FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Diplomová práce

Inovace datového modelu inventáře a sbírky dokumentů

Tomáš Zikmund

PLZEŇ

2024

FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Diplomová práce

Inovace datového modelu inventáře a sbírky dokumentů

Bc. Tomáš Zikmund

Vedoucí práce
Ing. Martin Kryl

PLZEŇ **2024**

Citace v seznamu literatury:
ZIKMUND, Tomáš. Inovace datového modelu inventáře a sbírky dokumentů. Plzeň, 2024. Diplomová práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky. Vedoucí práce Ing. Martin Kryl.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Tomáš ZIKMUND**
Osobní číslo:	**A22N0062P**
Studijní program:	**N0613A140040 Softwarové a informační systémy**
Téma práce:	**Inovace datového modelu inventáře a sbírky dokumentů**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s problematikou návrhu datového modelu a principy vizualizace dat.
2. Analyzujte datové modely projektu Inventaria Rudolphina.
3. Navrhněte vhodné úpravy a rozšíření modelu na základě zjištěných poznatků s ohledem na okolí projektu.
4. Implementujte datovou pumpu pro převod dat do nových modelů.
5. Navrhněte vizualizaci dat podporující explorativní analýzu dat.
6. Vyhodnoťte vhodnost navržené vizualizace a diskutujte dosažené výsledky.

Rozsah diplomové práce: **doporuč. 50 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

dodá vedoucí diplomové práce

Vedoucí diplomové práce: **Ing. Martin Kryl**
Katedra informatiky a výpočetní techniky

Datum zadání diplomové práce: **8. září 2023**
Termín odevzdání diplomové práce: **16. května 2024**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 11. října 2023

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného akademického titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Západočeská univerzita v Plzni má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Plzni dne 15. května 2024

. . . . . .
Tomáš Zikmund

V textu jsou použity názvy produktů, technologií, služeb, aplikací, společností apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Abstrakt

Tato diplomová kvalifikační práce se věnuje inovaci datových modelů inventáře a sbírky dokumentů z období vlády císaře *Rudolfa II.*, využívajících databáze *Inventaria* a *Documenta Rudolphina* spravované Ústavem dějin umění Akademie věd ČR. Smyslem inovace je navrhnout úpravy současné podoby datových modelů a vhodně rozšířit vizualizační možnosti, které uživateli pomohou lépe interpretovat a porozumět uměleckohistorickým datům nashromážděných v rámci projektu.

V počátku této práce bylo vyžadováno důkladné seznámení s problematikou návrhu datového modelu a zásadami vizualizace dat. Následovala podrobná analýza obou datových modelů projektu *Rudolphina*. Na základě získaných poznatků a zohlednění kontextu a okolí projektu byly navrženy adekvátní úpravy a rozšíření modelů. Implementace a následné nasazení datové pumpy umožnilo konverzi dat z původních modelů. Pro nově vzniklý datový model *Inventaria* byla následně vytvořena vizualizační aplikace podporující explorativní analýzu dat.

V závěru práce byla provedena evaluace vhodnosti a přínosu navržené vizualizace s ohledem na nově navržený datový model a diskutovány dosažené výsledky včetně nedostatků. Práce představuje kritickou reflexi provedených inovací s ohledem na zlepšení efektivity a užitečnosti datových modelů *Inventaria* a *Documenta Rudolphina* pro další výzkumné účely.

Abstract

This diploma thesis delves into the possibilities of innovating the data model for inventory and document collection from the reign of Emperor *Rudolf II.*, specifically focusing on the databases *Inventaria* and *Documenta Rudolphina* administered by the *Institute of Art History of the Czech Academy of Sciences*.The objective of such innovation is to propose modifications to the current form of the data models and to appropriately extend the visualization possibilities that would help the users to better interpret and understand the art historical data collected within the mentioned project.

The initial stage of this research involved an in-depth examination of data model design principles and data visualization fundamentals. An analysis of existing data models within the *Rudolphina* project was then conducted. Insights from this analysis led to the proposal of targeted modifications and enhancements to these models. The implementation included a data conversion mechanism that allowed for the smooth transition of data from original to revised models. Additionally, a web-based tool was developed to enable exploratory analysis of the new *Inventaria* data model.

The final part of the thesis provided a comprehensive evaluation of the visualization techniques used, assessing their effectiveness and advantages in the context of the new data model. The discussion also covered the results achieved and potential limitations identified, offering a critical review of the enhancements made to improve the functionality and utility of the

Inventaria and *Documenta Rudolphina* data models for advanced research endeavors.

Klíčová slova

Inventaria Rudolphina • Datové modelování • *ETL* • Vizualizace • *Dashboard*

Poděkování

Na tomto místě bych rád vyjádřil svůj upřímný dík Ing. Martinu Krylovi za jeho neocenitelnou podporu, trpělivost a odborné vedení po celou dobu tvorby této diplomové práce.

Dále bych chtěl poděkovat Mgr. et Mgr. Markétě Ježkové, Ph.D. a Ing. Martinu Medunovi za jejich cenné rady a konzultace.

Děkuji rovněž své rodině a přátelům za jejich pochopení a za to, že vždy stáli při mně.

Tomáš Zikmund,
autor diplomové práce
(květen 2024)

Obsah

Seznam zkratek a cizích výrazů

ASCII	*American Standard Code for Information Interchange*, standard pro kódování textových dat
API	*Application Programming Interface*, rozhraní pro programování aplikací
AV	Akademie věd České republiky
BI	*Business Intelligence*, proces pro analýzu dat v rámci podniku
BSON	*Binary JSON*, binární formát pro serializaci dat
Bool	*Booleovské* operátory, logické operátory v informatice
Box-plot	Typ grafu - krabicový
Cache	Mezipaměť
Callback	Funkce, která je volána po dokončení určité operace
Cheat-sheet	Stručný přehled důležitých informací
Cluster	Skupina (shluk) počítačů, pracují jako jednotný systém
CRUD	*Create, Read, Update, Delete*, základní operace s databází
CSS	*Cascading Style Sheets*, jazyk pro stylování webových stránek
DataFrame	Datová struktura v programovacím jazyce *Python*
Dashboard	Uživatelské rozhraní, vizuálně zobrazuje klíčové informace
dev	Označení vývojářeké verze databáze *Inventaria*
DOM	*Document Object Model*, objektový model dokumentu
ELT	*Extract, Load, Transform*, proces zpracování dat
ERD	*Entity-Relationship Diagram*, diagram entit a relací
ETL	*Extract, Transform, Load*, proces zpracování dat
FIFO	*First In, First Out*, metoda zpracování dat
Frontend	Klientská část aplikace
Fulltext	Vyhledávání na základě celého znění textu
Getty/ULAN	*Getty Union List of Artist Names*, databáze jmen umělců
HTML	*HyperText Markup Language*, značkovací jazyk pro tvorbu webovů
Hostování	Poskytování webového prostoru pro stránky
I/O	*Input/Output*, vstup/výstup
IEC	Mezinárodní elektrotechnická komise
ISO	Mezinárodní organizace pro normalizaci
Join	Operace v databázích, spojení tabulek
jQuery	Knihovna *JavaScriptu* pro zjednodušení skriptování *HTML*
JSON	*JavaScript Object Notation*, formát pro uchovávání a výměnu dat
Layout	Rozložení prvků na stránce nebo v dokumentu
Localhost	Lokální server, běží přímo na uživatelském zařízení
Logování	Záznam událostí v systému

MB	*Megabyte*, jednotka digitální informace
M:N	Vazba *many-to-many*, typ relace v databázích
Node	Označení pro prostředí *Node.js*
NoSQL	*Not only SQL*, nerelační typ databází
Open-source	Software s otevřeným zdrojovým kódem
Overhead	Dodatečná režie nebo zátěž
Pattern	Vzor, opakující se design nebo kód
Pipeline	Sekvence zpracovatelských kroků, či instrukcí
PNG	*Portable Network Graphics*, obrázkový formát
Prealokace	Předem alokovaný prostor pro data
Provenience	Původ nebo zdroj uměleckého díla
Query	Dotaz například do databáze
Rastr	Obrázek složený z pixelů
React	Knihovna *JavaScriptu* pro tvorbu uživatelských rozhraní
Real-time	Zpracování dat v reálném čase
Rollback	Operace, vracející databázi do předchozího stavu před provedením změn
Rolling-up	Agregační proces při datové analýze
rudolf	Označení pro zdrojovou databázi *Inventaria*
RAM	*Random Access Memory*, operační paměť
Schema-less	Databáze bez pevně definovaného schématu
Shard(ing)	Horizontální škálování, rozdělení databáze na menší části
SSH	*Secure Shell*, protokol pro zabezpečené vzdálené připojení
Stage	Dočasné úložiště v *ETL*, přechodné ukládání dat mezi procesními fázemi
SQL	*Structured Query Language*, jazyk pro práci s relačními databázemi
Stack	Technologický zásobník, sada technologií, služeb a nástrojů
Storytelling	*Vyprávění* scénářů v kontextu prezentace dat
SVG	*Scalable Vector Graphics*, vektorový grafický formát
SWOT	Analýza silných a slabých stránek, příležitostí a hrozeb
Tag	Značka, používá se pro označení částí v *HTML*
TEI	*Text Encoding Initiative*, standard pro kodování textových dat
Term	Pojem nebo výraz
TTL	*Time To Live*, doba životnosti dat v *cache* nebo při síťové komunikaci
Trigger	Spouštěč, databázový mechanismus reagující na určité události
UI	*User Interface*, uživatelské rozhraní
UML	*Unified Modeling Language*, grafický jazyk pro modelování softwaru
UNIX	Operační systém, předchůdce *Linuxu*
UX	*User Experience*, uživatelská zkušenost
Widget	Malý prvek uživatelského rozhraní poskytující specifickou funkci
XSLT	*Extensible Stylesheet Language Transformations*, jazyk pro transformaci *XML* dokumentů
XML	Extensible Markup Language, značkovací jazyk pro strukturování dat
ÚDÚ	Ústav dějin umění Akademie věd ČR
Ústav	Ústav dějin umění Akademie věd ČR
UTF	*Unicode Transformation Format*, způsob kódování znaků
ZČU	Západočeská univerzita v Plzni

Úvod 1

Teze se primárně zabývá databází *Inventaria Rudolphina*, jakožto nedílné součásti projektu ***Umění na odiv**: obrazová sbírka císaře Rudolfa II. v kontextu uměleckého sběratelství kolem roku 1600*[1]. Pro úplné pochopení kontextu a okolností tohoto projektu je nezbytné představit rudolfínské sbírky umění, jejich vývoj a historii a dále také instituci, stojící za realizací projektu. Po úvodním představení následují dílčí součásti projektu v podobě databáze a webového serveru *Inventaria Rudolphina* a přilehlého okolí. Cílem je poskytnout pohled na strukturu aplikace, ukázku konkrétního použití a v neposlední řadě i náhled na uchovávaná data. Opomenout nelze ani provázání aplikace s externími službami a aktuální technologické provedení aplikace. Z těchto dílčích kroků by měla vzejít ucelená představa o kontextu a záběru projektu.

Středem zájmu diplomové práce je datová vrstva architektury aplikace, tedy její datový model. S tímto tématem souvisí i nutnost prozkoumat principy a koncepty datového modelování a zasazení zjištěných poznatků do širšího rámce databáze *Inventaria*. S datovým modelováním se pojí i možnosti využití konceptů jako je například datová pumpa. U toho konceptu je nastíněn princip fungování, možné způsoby implementace, nasazení a potenciální přínosy. Dále je nutné zhodnotit aktuální stav databáze na základě důkladné analýzy obsažených dat a diskusí s uživateli. Tím lze identifikovat klíčové nedostatky a konkrétní požadavky na úpravy a optimalizace. V případě odhalení nedostatků bude nutné transformovat stávající datový model do inovované podoby, která odpovídá nově definovaným specifikacím. Tuto transformaci lze provést implementací a použitím mechanismu výše zmíněné datové pumpy. Transformovaná data je žádoucí patřičně interpretovat. Základním prostředkem pro prezentaci dat a operace s nimi je vizualizace, a tak práce přináší přehled různých typů vizualizačních přístupů s důrazem na knihovnu *Dash* od komunity *Plotly*[2] využívající programovací jazyk *Python*.

Nově zavedený datový model bude podroben ověření pomocí vizualizace postavené na grafové *dashboardové*[3] platformě podporující explorativní analýzu dat. Výchozí stav *dashboardového* zobrazení v *Inventaria* nic takového neumožňuje. Tento přístup umožní identifikovat dosud neuváděné statistiky a případně i skryté věcné vztahy mezi datovými záznamy. Efektivita a přínosy navržené vizualizace spolu s ostatními dosaženými výsledky budou posouzeny a diskutovány.

[1]Grant: GAČR 20-15927S - [Úst19]
[2]https://dash.plotly.com/
[3]https://www.thoughtspot.com/data-trends/dashboard/what-is-a-dashboard

Představení *Inventaria Rudolphina* — 2

2.1 Umění na dvoře císaře *Rudolfa II.* - rudolfínské sbírky

Umění na dvoře císaře *Rudolfa II.* v Praze, známé jako rudolfínské umění, představuje klíčovou epochu mezi koncem 16. a počátkem 17. století. Toto období se vyznačuje císařovým mecenášstvím, které formovalo unikátní rudolfínský styl ovlivňující pozdější barokní a manýristické umění. S příchodem umělců jako *Bartholomeus Spranger* či *Hans von Aachen* v 80. letech 16. století se Praha stala prvním uměleckým centrem Evropy, přitahujícím tvůrce z celého kontinentu. Nastupující generace umělců v 90. letech 16. století přinesla inovace v oblasti alegorického a žánrového umění, což obohatilo kulturní prostředí a přispělo k rozvoji umění charakterizovaného hlubokými alegorickými významy a kombinací mytologie s alchymistickou symbolikou. Mecenášství *Rudolfa II.* nejenže posílilo Prahu jako centrum evropského umění, ale také podpořilo propojení vědy, filozofie a umění, což mělo dlouhodobý vliv na evropskou kulturu a uměleckou scénu. [Zla21]

2.2 Ústav dějin umění Akademie věd ČR, v. v. i.

Ústav dějin umění Akademie věd ČR (dále jen jakožto *„ústav"*, případně *„ÚDU"*) je veřejná výzkumná instituce a jedním z významných výzkumných pracovišť fungujících pod záštitou Akademie věd (následně v textu již jen jako *„AV"*). Jeho primárním cílem je provádění výzkumu v oblasti dějin a teorie výtvarného umění, architektury, historického urbanismu, památkové péče, výtvarné kritiky, estetiky a muzikologie. Ústav se zabývá několika klíčovými projekty, jako jsou *Dějiny českého výtvarného umění* a soupisy *Uměleckých památek Čech, Moravy, Slezska a Prahy*. Pracovníci instituce se podílejí jako autoři na mnoha publikacích, pořádají výstavní projekty, případně vyučují na univerzitách. Kromě výzkumu a publikací *Ústav* nabízí řadu služeb veřejnosti, včetně specializovaných knihoven a fotoateliéru s rozsáhlým fotoarchivem. Ústav také spravuje bohaté sbírky historických plánů, fotografií a písemných pramenů.

Toto významné akademické pracoviště také spolupracuje s mnoha jinými vědeckými institucemi a odborníky jak doma, tak i v zahraničí a je členem mezinárodní asociace uměleckohistorických institutů (*Research Institutes in the History of Art*). [Úst20d]

2.3 Grant *GAČR* č. 20-15927S - Umění *na odiv* (*Art for Display*)

Tento grant představuje významný výzkumný projekt prováděný *ÚDU AV ČR*, konkrétně týmem vedeným Štěpánem Váchou. Projekt probíhal v letech 2020-2022 a poskytuje detailní náhled do obrazové sbírky císaře *Rudolfa II.*, jež byla uložena v Praze a následně se rozptýlila během třicetileté války. Hlavním výstupem výzkumu je dle stanoviska [Úst20b] projektového týmu vyhotovit monografii v anglickém jazyce, v níž bude zhodnocen význam sbírky, a to zejména v kontextu dobových sběratelských činností. Dílčí výsledky by pak byly publikovány formou odborných článků.

V období *Rudolfovy* vlády patřila tato sbírka k těm nejvýznamnějším v Evropě a byla bohatě dokumentována inventáři, dobovými prameny a identifikovanými uměleckými díly. Cílem projektu bylo provést detailní heuristický výzkum této sbírky a vytvořit speciální databázi (***Inventaria Rudolfina***), která umožnila komplexní analýzu a interpretaci. Zkoumalo se rozložení žánrů, zastoupení malířů a jejich škol a prostorové uspořádání a provoz sbírky.

Každé dílo v databázi je doplněno o klíčová slova umožňující dohledání výčtu souvisejících děl. Jména autorů, či věcně souvisejících umělců jsou přímo propojena s databází ***Getty ULAN*** obsahující kromě základních informací o umělcích také bibliografické údaje a jejich umělecké vazby. Témata obrazů jsou přiřazena odpovídajícími sjednocenými číselníky ***IconClass***. Jedná se o hierarchický systém kategorizace uměleckých a ikonografických motivů, který usnadňuje popis a vyhledávání uměleckých děl podle jejich obsahu a tématiky. Okolí projektu od roku 2021 zahrnuje i databázi ***Documenta Rudolphina*** skládající se z hierarchie přepisů archivních textových pramenů z 16. a 17. století. Původním autorem transkriptů byl *Manfred Staudinger* († 2021). [Ins21a]

Prostřednictvím tohoto projektu by mělo být získáno hlubší pochopení kulturního dědictví *Rudolfa II.* a jeho sběratelských aktivit, což může přispět k rozvoji poznání a interpretace této klíčové éry evropského umění.

Databáze *Inventaria Rudolphina* a *Documenta Rudolphina*

3

Jako součást grantového projektu *Umění na odiv* byly vytvořeny databáze *Inventaria*[4] a *Documenta*[5] *Rudolphina*, umožňující detailní analýzu obrazové sbírky císaře *Rudolfa II.* a dobových textových záznamů.

3.1 Databáze a webová aplikace Inventaria Rudolphina

Webová aplikace *Inventaria Rudolphina* umožňuje oprávněným uživatelům procházet kolekci digitálních obrazových záznamů zařazených do příslušných dobových inventářů. V době vytváření této práce bylo zaznamenáno celkem třináct chronologicky uspořádaných inventářů, datovaných od roku 1595 do roku 2020. Každý inventář poskytuje detailní informace o obrazech a uživatelé mají možnost provádět komplexní vyhledávání napříč inventáři. Podrobná funkcionalita je popsána v rámci oddílu 3.1.1, věnovaného navigaci v aplikaci.

Každá obrazová položka v této databázi obsahuje klíčová slova, odkazy na umělce v databázi *Getty ULAN* (3.2.2) a přiřazuje obrazům tematické číselníky *Iconclass* (3.2.3). Dále, jsou-li tyto informace dostupné, zobrazuje původní umístění obrazů z pražských inventářů v císařské galerii na Pražském hradě a na základě zjištěných konkordancí umožňuje sledovat obměnu obrazů na Hradě v období mezi lety 1621 a 1648. Aplikace také poskytuje informace o prostorovém uspořádání císařových sbírek.

Níže následuje rozbor funkcionality webové aplikace *Inventaria* z hlediska běžného uživatele. Následuje obsáhlá analýza a popis příkladu uložených dat. Je nutné podotknout, že analýze je podrobena vývojářská (*dev*) verze z března roku 2023 provozovaná na serverové infrastruktuře Západočeské univerzity v Plzni (*ZČU*). Bylo tak učiněno proto, aby v průběhu testování nedošlo k narušení přístupnosti a konzistence nasazené databáze, která je denně přistupována výzkumnými pracovníky *ÚDU*.

[4] https://www.inventariarudolphina.com/
[5] http://documenta.rudolphina.com/

3.1.1 Technologická realizace (*stack*)

Z technologického hlediska je aplikace *Inventaria Rudolphina* postavena na *Node.js* serveru, data jsou uložena prostřednictvím databázové platformy *MongoDB*. *Frontend* čili přístupové rozhraní, bylo vytvořeno za použití knihovny *Semantic UI* a reaktivního frameworku *Vue.js*. Dílčí technologie jsou orientačně představeny v následujících oddílech. [Úst20a]

3.1.1.1 Prostředí Node.js

Node.js[6] (dále pouze jakožto *Node*) je *open-source*, multiplatformní prostředí pro vykonávání *JavaScriptového* kódu. *Node* se rozsáhle využívá pro programování serverů, což vývojářům umožňuje uplatnit jazyk *JavaScript* nejen pro klientský kód, ale i pro implementaci serverového kódu, aniž by bylo nutné se učit další programátorský jazyk. Ačkoliv bývá *Node* označován jakožto programovací jazyk, či softwarový vývojový *framework*, obě označení jsou technicky nepřesná - jedná se striktně o *runtime environment* (prostředí) pro *JavaScript*. [SD22]

3.1.1.2 Dokumentová databáze *MongoDB*

Dokumentově orientované databáze, patřící do kategorie *NoSQL*, představují flexibilní alternativu k tradičním relačním databázím (detailní srovnání viz odstavec níže) díky své schopnosti ukládat data v semi-strukturované nebo strukturované formě dokumentů, typicky ve formátech *JavaScript Object Notation* (*JSON*) nebo *Extensible Markup Language* (*XML*). [Pap21] Ukázka struktury *JSON* dokumentu je zachycena v Příloze C. Jedná se o zdrojový záznam položky *PrgC-1* z *Inventaria Rudolphina*.

Tato flexibilita umožňuje uživatelům ukládat dokumenty s různou strukturou bez nutnosti pevně definovaných schémat, což usnadňuje správu a adaptaci datových modelů. Navíc, tyto databáze jsou navrženy pro snadné horizontální škálování a distribuci, což z nich činí efektivní nástroje pro zpracování velkých objemů dat a zajištění vysoké dostupnosti. [Mon23e]

MongoDB[7] jako přední představitel dokumentově orientovaných databází, nabízí inovativní přístup k ukládání a manipulaci s daty. Jeho model založený na dokumentech umístěných v kolekcích poskytuje vývojářům vysokou úroveň flexibility při modelování dat. Procesu datového modelování je pak vyhrazena 5. kapitola. Dokumentově orientovaný přístup vyžaduje ve srovnání s tradičními databázemi i odlišné techniky. [Mon23e]

Vzhledem k tomu, že MongoDB využívá specifický model bez pevně daných schémat, který se značně liší od tradičních relačních databází, je procesu datového modelování věnována samostatná podkapitola. Tato podkapitola podrobně vysvětluje, jak efektivně využívat dokumentově orientovanou strukturu pro optimalizaci výkonu, zajištění konzistence dat a maximální využití vestavěných funkcí MongoDB, jako jsou indexy, agregace a transakce na úrovni dokumentů.

Kolekce. Kolekce v *MongoDB*, fungující jako ekvivalent relačních tabulek, sdružují sady dokumentů. Dokumenty pak představují základní datové jednotky a jsou tvořeny páry *klíč-hodnota* a

[6] https://nodejs.org/en/about
[7] https://www.mongodb.com/

podobají se formátu *JSON*. Konkrétně se jedná o binární variantu zvanou *Binary JSON* (*BSON*). [Zím23] *BSON* je binární reprezentace *JSON* dokumentů, která byla navržena pro efektivnější ukládání a přenos dat. *BSON* rozšiřuje standardní *JSON* o další datové typy a funkcionality, jako jsou binární data, datum či časové razítko (*timestamp*), respektive o podporu vnořených dokumentů a polí, jež *JSON* rovněž nativně nepodporuje. [Kha23]

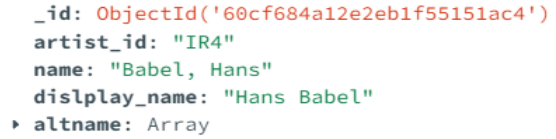
Projekt *Inventaria*, respektive *dev* verze databáze zvaná *rudolf*, se skládá z devíti dílčích kolekcí - *artists*, *concordances*, *iconclass*, *inventory*, *notes*, *related*, *rooms*, *ulan* a *users*. Stručný přehled je umístěn v tabulce 3.1.

Téměř každá z těchto kolekcí představuje specifickou oblast dat, která jsou zásadní pro správu, organizaci, dokumentaci a výzkum uměleckých děl v rámci *Inventaria Rudolphina*. V době psaní této teze obsahovala databáze v součtu necelých 14 000 dokumentů.

Tabulka 3.1: Přehled původních kolekcí v databázi *MongoDB* projektu *Inventaria Rudolphina*. Zdroj: Vlastní zpracování

Název kolekce	**Počet záznamů**	**Popis účelu**
artists	44	Původní abecední seznam umělců.
related	102	Prvotní spojový seznam souvisejících položek (inventář).
concordances	7 786	Uchovává informace o vztazích mezi dokumenty, mapuje vztahy v inventáři.
iconclass	654	Klasifikace obsahové náplně děl, použití číselníků *Iconclass*.
inventory	4 183	Uchovává detailní informace o uměleckých dílech.
notes	707	Uživatelské poznámky a komentáře k dílům.
rooms	34	Obsahuje komnaty a umístění na Pražském hradě (časoprostorové zasazení).
ulan	187	Obsahuje detailní informace o umělcích, načteny z databáze *Getty ULAN*.
users	-	Uchovává informace o uživatelích aplikace (správa přístupu, personalizace).

Původním účelem kolekce *artists* měl být ucelený abecední seznam autorů umění. Vyjma vlastního a zobrazovaného jména je k dispozici i pole *altname* umožňující ve formě samostatných řetězců uchovávat alternativní pojmenování autora. Kolekce pozbyla smysl, jelikož *Inventaria* integruje informace o autorech z databáze *Getty ULAN* (představena později - podsekce 3.2.2). Nadále není rozvíjena, ani se nepoužívá.



Obrázek 3.1: Ukázka struktury dokumentu z kolekce *artists*. Zdroj: Vlastní zpracování

Obdobně byl ukončen vývoj kolekce *related*. Ze strany výzkumníků se jednalo o prvotní pokus o sestavení spojovaného seznamu souvisejících položek, v podstatě tedy dílčího inventáře. Jednotlivé dokumenty reprezentující položky byly zachovány a zařazeny do aktuálních inventářů. Nesou však původní identifikátor v podobě *„Rel-XX"*. Struktura položky bude blíže představena u kolekce *inventory*.

Kolekce *concordances* slouží k uchování informací o vztazích mezi různými záznamy v databázi. Dokument v této kolekci může obsahovat identifikátory (*_id*), které odkazují na specifické zdrojové dokumenty (*src_xml_id*) a cílové dokumenty (*dst_xml_id*), mezi kterými byl stanoven věcný vztah. Dále zde můžeme pozorovat míru jistoty (*cert*) tohoto propojení a odpovědnost (*resp*) za vytvoření této vazby. Kolekce je esenciální pro mapování vztahů a propojení inventárních položek.


```
_id: ObjectId('63cc6c4ac070863a95aae129')
src_xml_id: "Inv-17"
dst_xml_id: "Bxl-1"
cert: "low"
resp: "SV"
```


Obrázek 3.2: Ukázka struktury dokumentu z kolekce *concordances*. Zdroj: Vlastní zpracování

Kolekce *iconclass* je určena pro klasifikaci a organizaci vizuálních děl pomocí mezinárodně uznávaného systému tematické klasifikace *Iconclass*. Dokumenty v této kolekci obsahují kódy *Iconclass* (*c*), které umožňují detailní kategorizaci obsahu obrazů, soch a jiných uměleckých děl. Kromě kódů zahrnuje každý záznam také textový popis (*txt*) v různých jazycích, klíčová slova (*kw*) pro usnadnění vyhledávání a navigace v databázi, a hierarchickou strukturu (*p*), která umožňuje sledovat vztahy mezi různými úrovněmi klasifikace.


```
_id: ObjectId('5edbfbf122a4e6cebaf6d0b7')
▸ c: Array
▸ r: Array
▸ txt: Object
▸ kw: Object
  n: "73B5"
▸ p: Array
▸ k: Array
▸ kws_all: Object
  iconclass_external_id: "73B5"
```

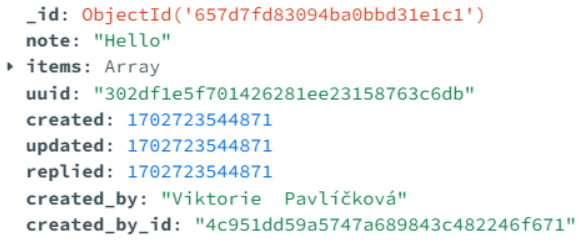

Obrázek 3.3: Ukázka struktury dokumentu z kolekce *iconclass*. Zdroj: Vlastní zpracování

Kolekce *inventory* v databázi *Inventaria Rudolphina* zaznamenává detailní informace o uměleckých dílech a předmětech, primárně obrazů, chronologicky členěných do jednotlivých inventářů. Každá položka obsahuje jedinečný identifikátor (*_id*), původní *XML* reprezentaci objektu (*original_xml*), pořadí umístění v inventáři (*order*), a *XML* identifikátor (*xml_id*). Dále jsou zde uvedeny tematické kategorie (*subject, search_subject*), vazby na další záznamy (*concordances*), textový popis a titulek díla (*text*, *title*), zdrojové informace (*src*), a specifická metadata o každém objektu včetně konkrétního obrazu (*object*), jeho názvu, typu, digitalizace, popisu, rozměrů, materiálu, datace, a provenience. V něm případně vnořený klíč *images* obsahuje rastrový soubor ilustrující daný obraz. Záznam dále obsahuje informace o inventáři (*inventory*), přístupová

práva (*access*) pro jednotlivé uživatelské role, informace o místnosti (*room*) a konkrétním umístění (*place*) díla na Pražském hradě (nachází-li se v některém z pražských inventářů), a také odkazy na předchozí a následující položku v příslušném inventáři (*prevItem*, *nextItem*).

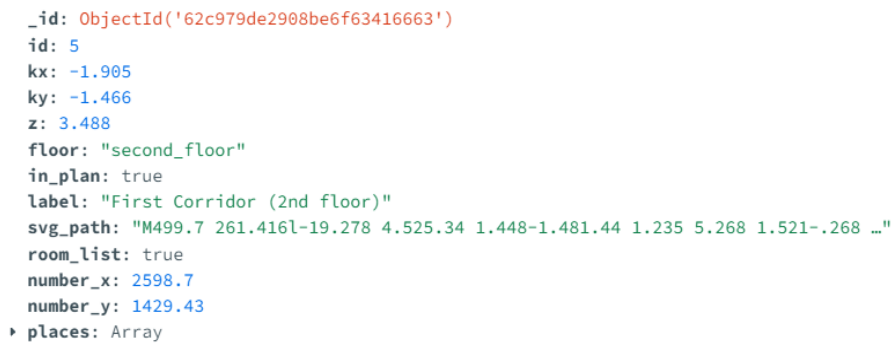
Obrázek 3.4: Ukázka struktury dokumentu z kolekce *inventory*. Zdroj: Vlastní zpracování

Oproti tomu kolekce *notes* shromažďuje poznámky nebo komentáře přidané k evidovaným dílům. Každá poznámka má svůj jedinečný identifikátor (*_id*), text poznámky (*note*), seznam identifikátorů děl, ke kterým se poznámka vztahuje (*items*), univerzální unikátní identifikátor (*uuid*), časové údaje o vytvoření poznámky a poslední aktualizaci (*created*, respektive *updated*) a informace o autorovi poznámky (*created_by*, *created_by_id*). Pole *replied* označuje čas poslední odpovědi nebo interakce s poznámkou.



Obrázek 3.5: Ukázka struktury dokumentu z kolekce *notes*. Zdroj: Vlastní zpracování

Kolekce *rooms* dokumentuje informace o místnostech, přesněji dobových komnatách nacházejících se na Pražském hradě, což zahrnuje jedinečný identifikátor místnosti (*_id*), identifikační číslo (*id*), souřadnice místnosti (*kx*, *ky*, *z*), patro (*floor*), informaci o zařazení do plánu (*in_plan*), popisek (*label*), *SVG* cestu pro vizualizaci (*svg_path*), příznak zařazení do seznamu místností (*room_list*), souřadnice pro číslování (*number_x*, *number_y*) a v neposlední řadě i seznam umístění (*places*) v místnosti s popisky (např. *„on the floor*"nebo *„on the bench*"). Tato kolekce poskytuje komplexní přehled o časoprostorové organizaci a lokalizaci obrazů v kontextu sběratelství *Rudolfa II*.



Obrázek 3.6: Ukázka struktury dokumentu z kolekce *rooms*. Zdroj: Vlastní zpracování

Následuje kolekce *ulan*. Obsahuje informace o umělcích, jejichž díla jsou součástí databáze. Zahrnuje jedinečný identifikátor (*_id*), externí identifikátor odpovídajícího záznamu v databázi *Getty* (*getty_external_id*), jméno umělce (*name*), popis (*description*), pohlaví (*gender*), odhadované období působnosti (*est_start*, *est_end*), místo narození a úmrtí (*birth_place_name*, *death_place_name*), zobrazované jméno (*display_name*), alternativní znění jména (*altname*), poznámku k dílům (*note*) a národnost (*nationality*). Tato kolekce je klíčová pro propojení uměleckých děl s jejich tvůrci.

Obrázek 3.7: Ukázka struktury dokumentu z kolekce *ulan*. Zdroj: Vlastní zpracování

Závěrečná kolekce *users* uchovává informace o uživatelích aplikace. Obsahuje jedinečný identifikátor (*_id*), uživatelské jméno (*username*), jméno (*first_name*), příjmení (*last_name*), heslo (šifrované, *password*), email (*email*), roli uživatele v systému (*role - admin*, *contributor*, *user*), barvu poznámek (*notes_color*), avatar (*avatar*) a univerzální unikátní identifikátor (*uuid*). Tato kolekce je nezbytná pro správu přístupu k databázi a personalizaci uživatelského prostředí.

Obrázek 3.8: Ukázka struktury dokumentu z kolekce *users*. Zdroj: Vlastní zpracování

Vlastnosti. *MongoDB* podporuje širokou škálu operací pro práci s daty, včetně *CRUD* operací (vytváření, čtení, aktualizace, mazání) a disponuje rozsáhlými možnostmi dotazování a indexace. Tato flexibilita a výkonnost činí *MongoDB* vhodnou pro široké spektrum aplikací, od jednoduchých projektů po komplexní podnikové systémy. [Mon23e]

Ačkoliv dokumentově orientované databáze včetně *MongoDB* nabízí řadu výhod, jako jsou flexibilita datových modelů, snadná škálovatelnost a efektivní práce s velkými objemy dat, nesou s sebou i potenciální nevýhody. K nim se řadí například větší komplexnost při zajištění konzistence dat mezi dílčími dokumenty nebo potenciální komplikace s optimalizací výkonu pro specifické typy dotazů. Přestože absence pevně definovaných schémat může být považována za výhodu, může také vést k nedostatkům v oblasti konzistence a validace dat. *MongoDB* se vyznačuje nedostatkem podpory pro transakční zpracování a složitým spojováním (*join*) dokumentů z různých kolekcí, což může být pro některé aplikace zcela nezbytné. Limitace dokumentového spojování s sebou může přinést datovou redundanci a tím i vyšší využití paměti, než by bylo nutné. Správné nastavení a řízení indexů je rovněž zásadní pro udržení vysokého výkonu. Všechny tyto aspekty je třeba zvážit při rozhodování o nasazení *MongoDB* pro konkrétní použití. [Gee23]

V případě aplikace *Inventaria Rudolphina* se *MongoDB* používá na základě schopnosti efektivně manipulovat s nestrukturovanými daty a i kvůli flexibilitě ve správě heterogenních datových souborů. *MongoDB* umožňuje dynamické schéma, což je vhodné pro akademické výzkumné projekty, kde se struktura dat může vyvíjet. Tato *NoSQL* databáze navíc podporuje *fulltextové* dotazování a horizontální škálování, což je u stále se rozrůstající sbírky, čítající v současnosti přes čtyři tisíce obrazových položek, žádoucí - především pro analýzu záznamů. [Mon23f] Na druhou stranu, nedostatky *MongoDB* lze pozorovat v oblastech konzistence dat a normalizace. Kvůli dynamickému schématu a tendenci k denormalizaci dat v rámci záznamů může být udržování konzistence napříč daty náročnější, což vyžaduje pečlivé plánování a zohlednění typických případů užití. Limitace mohou spočívat v náročnosti na správu a v optimalizaci pro konzistentní výkon při velmi velkých objemech dat a komplexních dotazech, které mohou vyžadovat pokročilé indexování a ladění. V kontextu *Inventaria Rudolphina* převládají v databázi operace čtení. Zápisy jsou ve srovnání s nimi zcela marginální, ať už z hlediska úprav/aktualizace záznamů, tak i z pohledu přidávání nových záznamů. Aplikování normalizace na např. redundantní data a ukládat je v separátní kolekci, by sice mohlo usnadnit správu konzistence dat, ale zároveň by mohlo i zvýšit komplexnost dotazů (typicky skrze spojování záznamů z více kolekcí), což by v konečném důsledku mohlo mít negativní dopad na rychlost vyhledávání napříč sbírkou. Méně časté zápisy navíc snižují riziko kolizí. Nicméně, je důležité zvážit optimalizaci schémat a indexů pro maximalizaci výkonu při čtení.

V konečném důsledku, ačkoliv dokumentově orientované databáze typu *MongoDB* představují cenný nástroj pro vývoj moderních aplikací, je žádoucí pochopit jejich charakteristiky, výhody i potenciální omezení, a to ještě před jejich začleňováním do technologických řešení.

Srovnání s relačními databázemi. *MongoDB* se odlišuje od relačních databází nejen datovým modelem, ale i použitým dotazovacím jazykem a strukturou. Srovnají-li se datové modely, tak relační databáze využívají tabulky, řádky a pevně definovaná schémata pro ukládání dat. Pro

zadávání dotazů se uplatňuje dotazovací jazyk *SQL*. Oproti tomu *MongoDB* pracuje s dokumenty a kolekcemi, kde jednotlivé dokumenty jsou uchovávány ve formátu *BSON*. [Zím23] Dotazování se provádí pomocí jazyka *MongoDB Query Language*. [Geo23] Z hlediska schématu platí, že relační databáze vyžadují předem definované schéma s fixními datovými strukturami, zatímco *MongoDB*, jak již bylo uvedeno výše v přehledu vlastností, je *schema-less*, což umožňuje agilní a flexibilní přístup k datům. Rozdíly je možné pozorovat i u škálování. Relační databáze se opírají o vertikální škálování, kdežto *MongoDB* podporuje kromě vertikálního škálování i horizontální, díky čemuž efektivně zvládá i větší objemy dat. Při výběru mezi *MongoDB* a relačními databázemi je klíčové zvážit specifické potřeby projektu a preferovaný datový model. Každý přístup má své výhody a nevýhody, rozhodnutí závisí na konkrétních požadavcích a cílech. [Gil23]

3.1.1.3 Framework *Semantic UI*

Uživatelská zkušenost (*UX*) a uživatelské rozhraní (*UI*) jsou úzce propojeny. Při vývoji webových stránek je nutné věnovat pozornost designu, jelikož rozhraní webové stránky by mělo být jednoduché a především snadno použitelné. Špatný *UX* může vést až k celkovému opuštění stránky. Design a uživatelské rozhraní mají zásadní vliv na použitelnost webu. [Pah23]

Semantic UI[8] lze charakterizovat jako *open-source* komponentovou knihovnu pro vytváření intuitivních uživatelských rozhraní s využitím technologií jako jsou kaskádové styly (*CSS*), *jQuery* a *HTML*. Poskytuje *Semantic* tématiky v podobě *CSS* stylů. Pomáhá vývojářům produkovat webové stránky s rychlým a stručným *HTML* ve spojení se schopností automaticky přizpůsobit rozvržení a obsah webové stránky tak, aby poskytovala optimální zobrazovací zážitek na široké škále odlišných zařízení a velikostí obrazovek. [Pah23]

I když *Semantic UI* umožňuje efektivní vývoj webových aplikací díky opětovnému využití komponent [Pah23], je důležité zvážit i jeho potenciální nevýhody. Mezi ně se řadí například strmá učební křivka, která vyžaduje větší časovou dotaci pro efektivní využívání *frameworku*. Omezená přizpůsobitelnost může představovat výzvu u projektů s unikátními designovými požadavky, zatímco větší objem kódu může negativně ovlivnit rychlost načítání stránek. Limitovaná podpora prohlížečů a závislost na názvech tříd mohou také komplikovat vývoj. [Ars23]

3.1.1.4 Framework Vue.js

Vue.js[9] představuje *JavaScriptový framework* pro vývoj webových uživatelských rozhraní, který rozšiřuje možnosti *HTML*, *CSS* a *JavaScriptu* prostřednictvím deklarativního a komponentově orientovaného programování. [You22] Jeho hlavní přednosti spočívají ve zjednodušení vývoje aplikací, podpoře znovupoužitelných *UI* komponentů a efektivní manipulaci s dokumentově orientovaným modelem (*DOM*), což přispívá k rychlejšímu a přehlednějšímu kódu. *Vue.js* je navíc navržen tak, aby se snadno integroval do stávajících projektů, díky čemuž je vhodný pro široké spektrum vývojových potřeb. [San24]

[8] https://semantic-ui.com/
[9] https://vuejs.org/

Framework se vyznačuje flexibilitou a schopností postupné integrace. To umožňuje vývojářům efektivně implementovat *Vue.js* do existujících aplikací bez nutnosti rozsáhlých změn kódu. [You22]

Vue.js sice nabízí řadu výhod, jako jsou podpora animací, multiplatformní vývoj a komponentová architektura, přináší však i několik úskalí. Patří mezi ně potenciální jazyková bariéra nebo nižší podpora ze strany velkých společností ve srovnání s jinými *frameworky*. Bariéra je způsobena tím, že majorita uživatelské komunity se nachází v Číně a značná část dostupných materiálů je čínštině. [Alt22b]

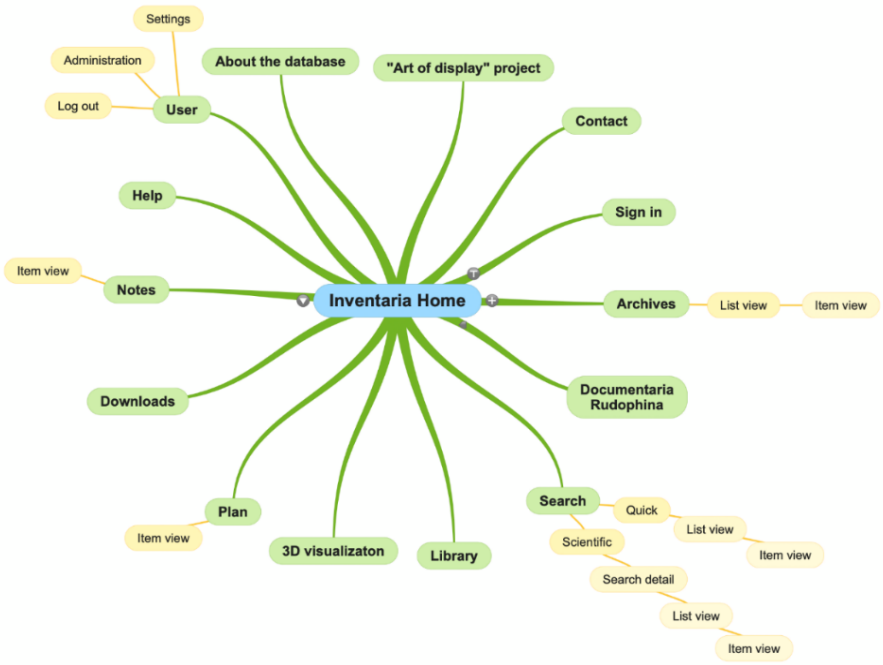
3.1.2 Navigace na webovém serveru

Popis navigace je strukturován podle jednotlivých obrazovek a sekcí aplikace, s důrazem na dva interakční scénáře - nepřihlášeného a přihlášeného (autentizovaného) uživatele. Tento přístup je zvolen z důvodu, že aplikace byla v době psaní této práce primárně koncipována pro uzavřenou komunitu výzkumníků z *ÚDU*, a tedy není plně zpřístupněna veřejnosti ve všech svých možnostech. Hlavní obrazovku včetně odkazů do jednotlivých sekcí je možné si prohlédnout na snímku B.1, umístěném v přílohách. V aplikaci *Inventaria* se lze přesměrovat do různých sekcí. Každá se vyznačuje odlišným účelem. Pro dodatečné upřesnění struktury aplikace byla níže vytvořena webová mapa (viz oddíl 3.1.2.1). Následuje stručný přehled:

- **Vstupní (indexová) brána**: Hlavní stránka aplikace pro všechny uživatele, včetně nepřihlášených.
- **Záložka *About the Database***: Informace o databázi, její obsah a zdroje.
- **Záložka *Art for Display***: Informace o stejnojmenném projektu věnující se digitální prezentaci uměleckých děl s historickým kontextem.
- **Záložka *Documenta Rudolphina***: Odkaz pro přesměrování na stránku projektu *Documenta Rudolphina*.
- **Záložka *Contact***: Kontaktní údaje na projektové členy.
- **Záložka *Library***: Přístup k evidovaným textovým publikacím, možnost stažení.
- **Záložka *Plan***: Plánková vizualizace Pražského hradu s *dashboardy* znázorňujícími plošné statistiky napříč *Inventariem*.
- **Záložka *3D Visualization***: V budoucnu by měla umožňovat trojrozměrnou virtuální prohlídku jednotlivých místností a umístění děl.
- **Záložka *Downloads***: Možnost stahování původních zdrojových *XML* souborů představující inventární položky.
- **Záložka *Notes***: Systém pro sdílení a správu poznámek - možnost komunikace.
- **Záložka *Help***: Návod pro porozumění věcným vztahům mezi uměleckými díly.
- **Záložka *User***: Uživatelská konfigurace a správa.
- **Záložka pro vyhledávání (*Advanced Search*)**: Filtrování a *fulltextové* vyhledávání inventárních záznamů v databázi. Vyhledává se na základě shody *termu* ze zadaného řetězce s množinou klíčových slov, která je vedena u každého inventárního záznamu.

3.1.2.1 Mapa webu *Inventaria*

Mapa webu funguje jako průvodce strukturou webové aplikace a umožňuje uživatelům představit si celkovou organizaci obsahu ve formě stromové hierarchie. Hlavní stránky webu slouží jako základní uzly tohoto stromu, zatímco další zanořené stránky představují větve, které se dále rozbíhají do dalších specifických sekcí. Tato vizuální reprezentace usnadňuje navigaci a upřesňuje vzájemné vztahy mezi stránkami, což je užitečné jak pro nové návštěvníky, tak pro správce webu při údržbě a aktualizacích obsahu. Mapa webu *Inventaria* je zachycena níže na obrázku 3.9. Následuje náhled na položky a zdrojová data databáze *Inventaria*.



Obrázek 3.9: Základní mapa webu *Inventaria Rudolphina*. Zdroj: Vlastní zpracování (Zikmund, Pavlíčková, 2023)

3.1.3 Rozbor obrazového záznamu z hlediska uživatelského rozhraní

Analýza vizuální reprezentace obrazového záznamu z hlediska *UI* se zaměřuje na rozložení stránky a detaily prezentace díla. Konkrétně se jedná o digitální zobrazení obrazu *Venus Cythereia - Flora with the City of Genua*[10] od *Jana Massyse*. Tento inventární záznam byl vybrán kvůli obsáhlým metadatům, která demonstrují možnosti vizualizace a výpisu obsažených informací. Uživatelské rozhraní v detailu inventárního záznamu zahrnuje následující prvky:

- **Navigační lišta:** Obsahuje standardní navigační prvky jako na úvodní stránce, s přídavkem seznamu inventářů v horní části.
- **Detailní zobrazení inventárního záznamu:** Prezentuje umělecké dílo s identifikátorem z inventáře, jménem autora a názvem díla, zmiňuje i časoprostorový kontext. Poskytuje rovněž přímý přístup k surovým datům (tj. k *XML* a z nich vytvořeným *JSON* souborům reprezentující položky).

[10] http://147.228.173.159/item/PrgC-1

- **Lišta se souvisejícími díly:** Zobrazuje karty s věcně souvisejícími díly, včetně vizuálních indikátorů pro konkordanci a nabízí možnost plynulého přesměrování mezi nimi.
- **Informační bloky:** Obsahují (jsou-li dostupné) podrobnosti o autorovi, díle, a jeho historii, provenienci, a to včetně číselníků a odkazů na další externí zdroje (*IconClass, Getty ULAN*).
- ***Widget* s plánkem Pražského hradu:** Interaktivně zobrazuje, kde bylo dílo pravděpodobně vystaveno na Pražském hradě, s možnostmi *zoomu* a detailního náhledu na příslušnou místnost.
- **Poznámky:** Boční panel slouží pro zobrazení a správu poznámek souvisejících s daným dílem, umožňuje uživatelům přidávat, upravovat a odpovídat na poznámky.

Detail inventárního záznamu je zachycen na obrázcích B.2 a B.3, které jsou uvedeny v přílohách, konkrétně jakožto součásti Přílohy B. Z této stručné analýzy lze vyvodit, že rozhraní bylo navrženo tak, aby poskytovalo uživatelům přístup ke všem relevantním informacím, a současně umožňovalo hlubší propojení s věcnými souvislostmi a historickým kontextem děl.

3.1.4 Rozbor obrazového záznamu z hlediska zdrojových dat

Tento oddíl je zaměřen na detailní analýzu položek v kontextu zdrojových *XML* a *JSON* souborů, obsahující informace o dílech uložených v inventářích *Inventaria Rudolphina*. Na ukázce již zmíněného obrazu *Venus Cythereia - Flora with the City of Genua*[11] jsou ilustrovány struktura a obsah obou datových formátů a je kladen důraz na porozumění jejich vzájemného vztahu a rozdílů. Oba formáty jsou určeny k ukládání a reprezentaci jednotlivých inventárních záznamů, avšak v odlišných organizačních strukturách. *XML* formát vychází z hierarchické struktury elementů, zatímco *JSON* využívá šablonu skládající se z různých položek.

Zdrojové *XML* soubory byly sepsány manuálně akademickými pracovníky a následně transformovány do formátu *JSON* dokumentů a vloženy do databáze *MongoDB*. Přepis byl motivován několika faktory. *JSON*, respektive *BSON*, dokumenty nabízejí efektivnější ukládání a rychlejší přístup k datům díky své struktuře a způsobu indexace, což je klíčové pro zpracování velkých datových objemů. Tento formát také umožňuje flexibilnější manipulaci s daty a jejich jednodušší integraci s moderními aplikacemi.

Pro úplnost je třeba uvést, že v rámci struktury těchto odvozených dokumentů je zápis původního *XML* uchován. V *JSON* dokumentu je vždy uvozen klíčem *original_xml*. Viz ukázka níže na obrázku 3.10. Koncepty struktur zdrojových souborů a vzájemné *namapování XML* elementů (a jejich atributů) na *JSON* klíče byly z důvodu přehlednosti umístěny do sekce příloh - konkrétně do Přílohy A.

[11] http://147.228.173.159/item/PrgC-1

Obrázek 3.10: Ukázka zachování původního *XML* ve struktuře odvozeného *JSON* dokumentu. Zdroj: Vlastní zpracování

3.1.4.1 Rozbor položky - *XML*

Níže následuje podrobná analýza hierarchické struktury *XML* souboru. Hierarchie souboru je konstituována z jednotlivých elementů. Pro koncept struktury viz Příloha A. Soubory byly konstruovány dle standardu *Text Encoding Initiative* (*TEI*)[12], což je konsorcium, respektive soubor směrnic, zaměřený na vývoj standardů pro digitální kodifikaci textů, sloužící k výzkumným účelům v oblasti humanitních a sociálních věd. *TEI* využívá *XML* pro definování a strukturování dat, což umožňuje přesné a detailní zpracování textu. *TEI* poskytuje soubor specifických elementů a atributů pro komplexní popis textů, a to včetně jejich struktury, jazyka a interpretace. *TEI* představuje důležitý nástroj pro digitalizaci a analýzu textových materiálů, umožňující vědeckým pracovníkům efektivně archivovat, sdílet a analyzovat data. [AT 23] Strukturu *XML* lze lépe ilustrovat na konkrétním případě. Níže rozebíraný kód popisuje konkrétní databázový objekt, a to výše zmíněnou ukázku obrazu z podsekce 3.1.3.

První uvedený prvek, nazvaný ***item***, obsahuje atributy ***n***, ***xml:id***, ***ana*** a ***corresp***.

- Atribut ***n*** nese hodnotu `1` a funguje jako identifikátor, respektive pořadí záznamu v inventáři.
- Atribut ***xml:id*** má hodnotu `PrgC-1` a funguje jako identifikátor *XML* záznamu.
- Atribut ***ana*** nese hodnoty `#th.erotic`, `#th.Genua`, `#th.city`, `#th.nude`, `#th.flower` a `#th.flowers`. Hodnoty fungují jako klíčová slova tématik vyobrazených na položce.
- Atribut ***corresp*** obsahuje hodnotu `Iconclass 96A23` sloužící jako odkaz na ikonografický záznam v databázi *IconClass*, blíže specifikující téma položky.
- Dále se v elementu ***item*** nachází vnořený element ***title***, zachycující název obrazu, konkrétně `Dea Flora ligt ganz nackent`.

Následuje element ***seg*** s atributy ***ana***, ***cert*** a ***resp***.

[12] https://tei-c.org/

- Atribut ***ana*** nabývá hodnoty `#src.original` a udává, že záznam popisuje originál, tedy původní dílo.
- Atribut ***cert*** s hodnotou `high` naznačuje vysokou míru jistoty při procesu identifikaci tohoto záznamu.
- Atribut ***resp*** byl přiřazen řetězec `#MJ` stanovující iniciály osoby odpovědné za ověření a publikování záznamu. Dodatečný text `vom` může být částí věty, ale bez kontextu není jasné, co přesně znamená.

Nyní element ***persName*** obsahující jméno autora obrazu, zde `Joann Massi`, a disponuje atributy ***role*** a ***corresp***.

- Atribut ***role*** udává roli osoby vůči dílu, v tomto případě se jedná o `artist` (umělce).
- Atribut ***corresp*** odkazuje skrze *URL* na záznam osoby v databázi *Getty ULAN*, kde je veden i autor `Joann Massi`. Umístěn je přímý odkaz[13].

Element ***note*** obsahuje řetězec „`1 stuckh. Ob identisch mit Dudik A 410 (S. XXXIX): Eine Flora?".` Poznámka poskytuje další informace o tématice záznamu a odkazuje na bibliografický zdroj, který by mohl být relevantní pro další výzkum.

Následují elementy ***ptr***, které odkazují na stanovené věcně související položky (s danou mírou pravděpodobnosti se jedná o totéž dílo, které je současně evidováno v dalších inventářích). Element se vyznačuje atributy ***sameAs***, ***cert*** a ***resp***.

- Atribut ***sameAs*** se odkazuje do jiných inventářů skrze identifikátory korespondujících děl. Zde se jedná o `PrgD-410` a `Stck-54`.
- Atributy ***cert*** a ***resp*** fungují obdobným způsobem jako u elementu ***seg*** uvedeného výše. ***Resp*** zde figuruje ve významu ověření provázanosti děl z jiných inventářů.

Element ***listObject*** obklopuje všechny následné elementy blíže specifikující inventární záznam. Uvnitř tohoto elementu se nachází ***object*** s atributem ***xml:id*** nabývající hodnoty `alt-1-PrgC-1`, jež představuje identifikátor objektu v inventáři.

Element ***object*** obsahuje několik vnořených elementů. Prvním je ***objectIdentifier*** s atributy ***cert*** a ***resp***.

- ***Cert*** uvádí úroveň jistoty identifikace díla (zde `high`), zatímco ***resp*** iniciály osoby zodpovědné za tento proces (`EW`) - obdobně jako u elementu ***seg***.

Uvnitř ***objectIdentifier*** jsou zanořeny elementy, jako ***objectName***, obsahující název objektu (`Venus Cythereia - Flora with the City of Genua`), ***idno*** s evidenčním číslem v rámci instituce, v němž se dílo aktuálně nachází (`NM 507`), ***institution*** obsahující název spravující instituce (`Nationalmuseum`) a ***address*** s geografickými a souřadnicovými údaji.

V ***address*** elementu se nachází ***country*** s přiřazenou hodnotou `Sweden`, dále ***settlement*** obsahující název sídla, v němž se spravující instituce nachází (`Stockholm`) a ***location*** obsahující geografické souřadnice v rámci elementu ***geo*** (`59.328611 18.078333`).

Dalším vnořeným elementem v rozsahu ***object*** je ***physDesc***, který obsahuje informace o fyzických parametrech objektu skrze element ***p***. Nachází se zde duplikátní zápis elementu

[13] http://vocab.getty.edu/page/ulan/500011809

PersName a souvisejících atributů, dále obsah bloku ***Description*** (viz informační bloky v *UI*, podsekce 3.1.3) a elementy *ref* obsahující reference do zdrojových institucí[14], případně na související materiály v atributu ***target***.

Element ***physDesc*** zahrnuje informace o použitých technikách a materiálech (`material - oil, on oak panel`), typu objektu (***objectType*** - `painting`), a rozměrech (***dimensions***) s atributy ***height*** (`130`) a ***width*** (`156`).

- V obou těchto případech nadefinován atribut ***unit*** upřesňující délkovou jednotku, zde se jedná o `cm`.

Závěrem pak ***object*** obsahuje ***history*** element, s informacemi o historickém vývoji objektu. Ve hnízděném elementu ***origin*** se nachází ***origDate*** s hodnotou `1561` stanovující rok původu. Současně je v témž elementu iniciován atribut ***when*** s duplikátní hodnotou.

V ***provenance*** je uveden předpokládaný původ obrazu (`Possibly from the collection of Rudolf II; possibly 1648 Queen Christina in Sweden; 1865 transferred from the Royal Museum.)`

XML kód poskytuje podrobné informace o inventárních objektech v rámci *Inventaria*. Míra a podrobnost těchto informací se však může u každé položky lišit. Poskytované informace jsou strukturované, což umožňuje snadný přístup a využití v dalších aplikacích. Zdrojové soubory jsou v rámci aplikace zkonvertovány do formátu *JSON* pro usnadnění vizualizace.

Je však nutno upozornit, že kód obsahuje strukturální nekonzistence, které by bylo žádoucí vyřešit. U tohoto konkrétního záznamu byl diagnostikován nejasný výskyt řetězce `undefined` napříč strukturou. Konkrétně se jedná o:

- **Nedefinovaný text pod nadpisem:**
  - V bloku ***title*** se nachází text „`Dea Floraundefined ligt ganz nackent`", kde se zdá, že řetězec `undefined` by neměl být součástí obsahu.
- **Nedefinovaný text pod <*persName*>:**
  - Pod blokem ***persName*** se vyskytuje znak tečky bez jasného kontextu.
  - Hodnota elementu ***persName***, zanořeného v elementu ***object***, obsahuje řetězec `undefined` bez jasného kontextu.
- **Nedefinovaný text v odkazech:**
  - V bloku ***ref*** se nachází `undefined`.
- **Nedefinované hodnoty v elementu *location*:**
  - V bloku ***location*** v poli ***geo*** jsou uvedeny hodnoty souřadnic ve formátu `59.328611undefined 18.078333`, což naznačuje, že může být něco nevyplněno.
- **Formát zápisu souvislého textu**

[14] Například: https://rkd.nl/explore/images/51705

– Při zápisu souvislého textu do hodnot atributů dochází k nekonzistencím v podobě náhodných výskytů nadbytečných mezer mezi jednotlivými slovy.

Je žádoucí prověřit tyto oblasti, opravit případné chyby či nekonzistence v hodnotách a zjistit, zda jsou všechny proměnné a hodnoty řádně definovány, aby bylo možné se v budoucnu těmto nejednoznačnostem vyhnout.

3.1.4.2 Rozbor položky - *JSON*

Struktura *JSON* vychází ze šablony obsahující sadu objektů tvořených ze dvojic klíč/hodnota. Přehled klíčů lze nalézt v konceptu struktury v Příloze A.

Vzhledem k tomu, že *JSON* byl zkonvertován z *XML*, je jeho obsah prakticky analogický a není tak nutné opakovaně rozebírat význam hodnot jednotlivých klíčů (tj. hodnot korespondujících *XML* elementů). Má význam se tak zaměřit primárně na strukturální rozdíly.

Srovnání struktur *XML* a *JSON* pro popis uměleckého díla, odhaluje zajímavé podobnosti a rozdíly mezi těmito dvěma formáty datového záznamu. Oba formáty jsou určeny k identifikaci a podrobnému popisu uměleckého díla, ale způsob prezentace informací se odlišuje.

Rozdíly mezi formáty lze identifikovat ve struktuře informací o inventárním záznamu. Zatímco *XML* využívá vnořené elementy, *JSON* volí pole objektů, což usnadňuje hierarchické uspořádání informací.

Ve struktuře *JSON* je pořadí záznamu v inventáři a identifikátor reprezentován v podobném duchu, avšak v případě tématik dochází k přidání pole ***search_object*** obsahující seznam řetězců klíčových slov pro vyhledávání, které byly přejaty z externího *IconClass* záznamu. Informace, zda se jedná o originální dílo a s jakou pravděpodobností to je jisté, se ukládá v klíči *src* (v *XML* odpovídá elementu *seg*). Věcně související položky jsou odkazovány v klíči ***concordances***.

I pro *JSON* platí, že pomyslným hlavním klíčem je pole ***object***, které obsahuje podrobnosti inventárního záznamu. Narozdíl od *XML* však tento klíč obsahuje i ucelené podrobnosti o umělci, včetně národnosti, pohlaví a dat narození a úmrtí v poli ***name***. V *XML* se jednalo o avizovaný element ***persName*** a podrobnosti se zjišťovaly externě z *ULAN*. Zvláštností je, že pole ***name*** se v rámci klíče ***object*** vyskytuje duplicitně - zde v obou objektových *literálech*.

Dalším rozdílem je, že *JSON* v ***object*** uchovává informaci o tom, zda dílo bylo digitalizováno (viz klíč ***digitalisation_layer***) a rovněž uchovává i zdrojové soubory s náhledy a popisky (pole ***images***). Zajímavostí je, že v případě původního elementu ***origDate*** v *JSON* již nedochází k duplicitnímu zaznamenání hodnot.

Struktura *JSON* je navíc obohacena o klíče nesoucí informace identifikující inventář v rámci Inventaria (***inventory***), místnost (***room***) a polohu v místnosti (***place***) na Pražském hradě a závěrem i identifikátor předcházející a následující položky ze seznamu (***prevItem***, respektive ***nextItem***).

Odlišnosti jsou patrné také v zápisu některých informací jako poznámek, či odkazů - v *XML* se řetězce zapisují do hodnot elementů a odkazy pak do příslušných atributů, v *JSON* se využívají *HTML* tagy (viz element ***ref*** s atributem ***target*** a klíč ***physical_description*** ve zdrojovém *JSON*).

Na základě tohoto rozboru lze konstatovat, že oba formáty jsou navzdory odlišným strukturám a organizaci dat schopny podrobně popsat umělecké dílo a přinášejí specifické výhody a úskalí v závislosti na požadavcích uživatele. Zdrojový soubor v obou formátech je v původní nezměněné podobě uveden v rámci Přílohy C. Zde ilustrované zdrojové soubory, dostupné přímo z webu *Inventaria*, obsahově i strukturálně prakticky odpovídají korespondujícím záznamům nacházejících se v databázi (3.1.1.2). Výjimku představuje několik záležitosti, a to unikátní identifikátor záznamu, který je v rámci databáze nezbytný, dále následuje klíč ***original_xml***, jež jako řetězcovou hodnotu obsahuje znění původního *XML*, z nějž byl *JSON*, respektive *BSON*, odvozen. Oproti tomu, *XML* je na webu zobrazeno samostatně. U webové reprezentace zdrojového kódu záznamu dochází k oddělení klíče udávající informace týkající se *IconClass* (obsah atributu ***iconclass_data*** - viz snímek 3.3 výše) do podoby samostatného dokumentu. Změny jsou tedy čistě vizuálního charakteru.

3.2 Okolí a kontext *Inventaria Rudolphina*

Následující sekce představuje okolí a kontext databáze *Inventaria*. Jmenovitě se jedná o databáze *Documenta Rudolphina* a *Getty ULAN*, respektive o klasifikační systém *ICONCLASS Illustrated Edition*.

3.2.1 Databáze *Documenta Rudolphina*

Okolí projektu *Inventaria*, jak již zaznělo v úvodním představení (sekce 2.3) grantu *Umění na odiv* (*Art for Display*), je od roku 2021 rozšířeno o kolekci textových přepisů dobových pramenů (tj. z období vlády *Rudolfa II.*) umístěných v databázi nesoucí název ***Documenta Rudolphina***. Stalo se tak po úmrtí původního autora pana *Manfreda Staudingera*. Databáze je k dispozici k volnému prohlížení, bez nutnosti přihlášení se pod uživatelským účtem, tak jak tomu je u *Inventaria*. [Ins21a]

3.2.1.1 Rozvržení uživatelského rozhraní a obsah

Rozvržení a design úvodní stránky[15] je přímočarý a obsahuje pouze několik záložek a anotaci v německém jazyce v níž je vysvětlen původní záměr autora digitalizovat a tím zachovat kulturní dědictví, jež přepisované texty mohou představovat.

První uváděná záložka *Personen* odkazuje na seznamy historických osob agregovaných dle počátečního písmene jejich příjmení, eventuelně jména, pokud není příjmení dostupné. Jedná se o historické figury vyskytující se v původních textových pramenech.

Druhá záložka *Dokumente* uživatele přesměruje na hierarchickou stromovou strukturu představující lokace a instituce, v nichž jsou původní texty uchovávány a v listech se rovněž nachází i přímé odkazy na odpovídající přepisy.

Tyto dvě výše uvedené záložky představují informační jádro *Documenta* a jsou detailněji analyzovány v odstavcích níže.

[15] http://documenta.rudolphina.com/

Následující záložka *Essays* pak odkazuje na seznam pojednání. V době psaní čítal seznam pouze jednu položku a to *Der Kuati*[16] datované k červnu roku 2014. Zbývají záložky *programming* a *info* poskytující základní přehled o použitých vývojových technologiích, respektive o zaštiťující organizaci.

3.2.1.2 Organizace souborů

Projekt se skládá z kolekce zdrojových (datových) *XML* souborů, systematicky strukturovaných do čtyř dílčích složek - ***Archiv*** (stromová struktura archivu), ***Indices*** (indexy osob), ***Namen*** (informace o osobách) a ***Regesten*** (přepisy textů). Předmětem rozboru jsou *XML* soubory obsahující indexaci, literární záznamy, hierarchii archivu, jména osob a přepisy listin dokumentů. Z této širší kolekce přepisů (***Regesten***) byla zvolena omezená podmnožina: *Archive > Praha > Archiv Pražského hradu > Dvorská komora*, a to z praktických důvodů, jelikož projekt celkově zahrnuje tisíce unikátních přepisů.

Princip fungování webu *Documenta Rudolphina* spočívá v transformaci zdrojových *XML* souborů pomocí technologie *eXtensible Stylesheet Language Transformation* (*XSLT*) *style-sheetů* a jejich prezentaci ve formě webového *HTML* zobrazení.

XSLT je stylingový jazyk pro *XML*, umožňující transformaci *XML* dokumentů do jiných formátů, například právě do *HTML*. *XSLT* umožňuje sofistikovanější transformace než *CSS*, včetně přidávání/odebírání elementů a atributů, řazení a filtrování elementů, testování a rozhodování o zobrazení, či skrývání prvků. [Ref23]

Každý *XML* soubor obsahuje odkaz na příslušný *XSLT style-sheet*, který se nachází v kořenovém adresáři s příponou *.xsl*. Tyto *style-sheety* využívají doprovodné kaskádové styly, jež jsou integrovány do kořenové struktury adresáře a určují způsob zobrazení dílčích elementů na stránce. Transformace *XML* do *HTML* umožňuje prezentaci obsahu webové stránky pomocí definovaných stylů a formátování.

3.2.1.3 Struktura souborů

Analýza se soustřeďuje na zkoumání konzistence napříč jednotlivými složkami z důvodu manuálního plnění a přidávání dat autorem (p. *Staudinger*). Identifikované nekonzistence a anomálie budou zdokumentovány a specifikovány, například pro možnost následné plošné transformace zdrojových souborů do jiného formátového typu.

Každý *XML* soubor obsahuje hlavičku s verzí, kódováním a odkazem na použitý *style-sheet*. Data jsou prezentována ve značkované (*tagované*) struktuře, kde elementy reprezentují význam jednotlivých částí textu. Při analýze je kladen důraz na gramatické aspekty a transformaci *non-ASCII* znaků, charakteristických pro dobovou *„ranou novou horní němčinu"*(*Frühneuhochdeutsch, fnhd.*). Pro korektní zobrazení se využívá vhodného kódování, jako například *ISO/IEC 8859-15* nebo *UTF-8*.

Následuje podrobný rozbor obsahu a struktury čtyř dílčích kolekcí.

[16] http://documenta.rudolphina.com/Essays/Coati.html

Složka *Archiv*. Kolekce ***Archiv*** obsahuje *XML* soubory se stromovou hierarchií lokací, z nichž pocházejí přepisované textové záznamy. Názvy souborů identifikují konkrétní lokaci a úroveň zanoření v hierarchii archivu. Hierarchie je rozčleněna do **devíti** úrovní (od nulté až po osmou) a struktura je hierarchicky organizována od nejvyšší úrovně reprezentující archiv s šestnácti městy až po nejnižší úroveň, obsahující přímé odkazy na jednotlivé přepisy textů (odkazy do *Regesten*).

ÚROVEŇ	PREFIX – název	OBSAH
0	**A_0**	Všech 16 měst
1	**A_1**	Jednotlivé město
2	**A_2**	Archiv / galerie/ univerzita v daném městě
3	**A_3**_<zkratka A_2>	Jednotlivé archivy
4	**A_4**_<zkratka A_2>_<zkratka A_3>	Název / číslo spisu
5	**A_5**_<zkratka A_2>_ ... _<zkratka A_4>	Název / kódové označení díla
6	**A_6**_<zkratka A_2>_ ... _<zkratka A_5>	Podnázev (Vazba / svazek /páska)
7	**A_7**_<zkratka A_2>_ ... _<zkratka A_6>	Podnázev (název / číslo)
8	**A_8**_<zkratka A_2>_ ... _<zkratka A_7>	Přepisy (odkazy do kolekce *Regesten*)

Obrázek 3.11: Struktura úplné větve logického stromu *Archivu*. Zdroj: Vlastní zpracování (Bozděch, Zikmund, Pavlíčková a kol., 2022)

Je důležité podotknout, že obsah na jednotlivých úrovních se může od obsahu tabulky lišit. Tabulka představuje pouze kompletní větve stromu lokací, tedy ty, které sahají od 0. až do 8. úrovně zapouzdření. Běžně se může stát, že je větev zakončena v jiné než osmé úrovni.

Například větvení děl evidovaných v Brně je zakončeno na třetí (respektive čtvrté úrovni, jelikož se začíná na nulté) úrovni zapouzdření, kde jsou již uvedeny přímé odkazy do kolekce *Regesten*. Konkrétně tato cesta by měla následující podobu:

A_0_Archive.xml (Kořen archivu) →
A_1_Archive_Brno.xml (*Brno*) →
A_2_Moravsky_zemsky_archiv.xml (*Moravský zemský archiv*) →
A_3_MZA_RAD_Inv_1898_46.xml (*RAD Inv. 1898-46 [Karton 421]*)

Struktura jednotlivých *XML* souborů se mění na každé úrovni zanoření, avšak princip struktury zůstává neměnný. Soubor disponuje větší mírou informací s každou další úrovní zanoření.

Jako ukázka byl zvolen generický soubor odpovídající druhé úrovni zanoření ***A_2_XXX.xml***. Ilustrační zdrojový soubor v původní podobě je umístěn v Příloze D - část *Archiv*.

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <?xml-stylesheet href="../view_archive.xsl" type="text/xsl"?>
3   <dr:doc xmlns:dr="http://documenta.rudolphina.org/"
4   xmlns="http://www.w3.org/1999/xhtml" uri="A_2_Nazev archivu" v="
    PUBLICATION_DATE"><title>Nazev archivu</title>
5   <!-- Cesta k tomuto souboru obsahujici odkazy na predchozi urovne
    -->
6   <div class="path">
7   <a href="../Archiv/A_0_Archive.xml">Archive </a> &gt;
8   <a href="../Archiv/A_1_Archive_Nazev mesta.xml">Nazev mesta</a> &
    gt; <span><Nazev
```

```
9      archivu>
10     </div>
11     <!-- Cesta s odkazy k souborum na nizsi urovni -->
12     <ul>
13     <a href="../Archiv/A_3_Zkratka mesta_Nazev archivu.xml">Nazev
       archivu </a>
14     <!-- ... -->
15     <a href="../Archiv/A_3_Zkratka mesta_Nazev archivu.xml">Nazev
       archivu </a>
16     </ul>
17     </dr:doc>
```

Zdrojový kód 3.1: Struktura dokumentu z kolekce *Archiv*

Poznatky:

- **Nultá úroveň:** Reprezentována výchozím souborem *A_0_Archive.xml* obsahujícím šestnáct měst. Názvy měst jsou v různých jazycích podle země, což může komplikovat přehlednost a navigaci pro uživatele.
- **Struktura:** Logicky členěná, ale s devíti úrovněmi a zkratkami může být s narůstající úrovní zanoření postupně znepřehledňována. Přehlednost by mohlo zvýšit použití jiných zkratek.
- **Větvení:** Některé větve obsahují odkazy do kolekce *Regesten* i na úrovních, kde stále dochází k větvení hierarchie.
- **Hlavička souboru:** Obsahuje parametry, včetně verzování. Verze může sloužit k monitorování provedených změn.
- **Konzistence:** Technicky vzato se jedná o strukturu seznamu s odkazy do vnitřního zapouzdření. Zápis se jeví konzistentním.

Tento způsob organizace záznamů nabízí jasnou hierarchii, i když s kompromisy týkající se srozumitelnosti zkratek archivů v nižších vrstvách. V zájmu zvýšení uživatelské přívětivosti by mohla být uvažována optimalizace tvorby zkratek a případná revize způsobu zobrazení.

Složka *Indices*. Složka *Indices* zahrnuje *HTML* soubory s abecedními seznamy historických osobností, kategorizovanými podle počátečního písmene příjmení, případně jména. Soubory obsahují reference do kolekce *Namen* na odpovídající osoby. Po rozkliknutí detailu osoby se zobrazí ucelený seznam přepisů s odkazy do záznamů z kolekce *Regesten*, kde je daná osoba zmíněna či uváděna. Platí, že ze jména osoby je možné přímo přejít až na konkrétní přepsané záznamy, v nichž osoba figuruje. Platnost je oboustranná - z přepisu záznamu je možné přejít na detail osoby v něm figurující.

Pro účely rozboru budou použity zdrojové soubory obsahující bibliografické seznamy literárních děl příslušného autora. Na jeden zdrojový soubor připadá jeden autor. Autorovi může být vyhrazeno více souborů. Názvy souborů nesou prefix „*Lit_*", jméno autora a případně i rok vydání díla.

Struktura je konzistentní pro všechny kontrolované soubory obsahující bibliografické seznamy literárních děl příslušného autora. Hlavička je standardní u všech literárních záznamů. Pořadí elementů (až na výjimky, viz poznatky níže) je konzistentní, pro ohraničení údajů se používají v každém souboru identické *tagy*. Elementy se nepřekrývají. Pro vysázení do *HTML* je použit *XSLT style sheet* (*view_literatur.xsl*). Obsahová část je uvozena elementem <*body*>. Řetězec *URI* kopíruje strukturu názvu souboru. Je tvořen dvojicí jméno autora a rok publikování díla. Níže následuje generická ukázka struktury záznamu v *Indices*. Ilustrační zdrojový soubor v původní podobě je možné najít v Příloze D - část *Indices*.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <?xml-stylesheet href="../view_literatur.xsl" type="text/xsl"?>
3  <body xmlns="http://www.w3.org/1999/xhtml" uri="Jmeno autora,
   pripadne i rok">
4  <!-- Text obsahujici seznam literarnich del prislusneho autora a
   informace o techto dilech -->
5  <!-- Stylizovany odkaz -->
```

Zdrojový kód 3.2: Struktura dokumentu z kolekce *Indices*

Poznatky:

- **Seznam děl:** Struktura seznamu děl je konzistentní a neměnná. Každý soubor obsahuje seznam děl, přičemž zápis textu je rovněž soudržný.
- **Informace o dílech:** Informace o dílech jsou ve zdrojových souborech odřádkovány, ale nepravidelně uvozovány *tagem* (<*br/*>). Zápis se jeví být konzistentní, výskyt *tagu* nepravidelný.
- **Obsah:** Veškerý informační obsah o dílech se vyskytuje ve formě prostého textu v těle jednoho elementu uvozeného tagem <*body*>.
- **Vložené externí odkazy:** Některé soubory (např. ilustrační ukázka - ***Lit_ Vignau-Wilberg_1992.xml***) obsahují vložené odkazy, jmenovitě třeba stylizované hypertextové odkazy na online publikaci souvisejících děl. Struktura informací o dílech je pravidelná, avšak výskyt externích odkazů nepravidelný. Doprovodné externí webové odkazy je žádoucí ponechat, ale zároveň transformovat do unifikované podoby.
- **Citace děl:** Soubory se seznamy citujících děl jsou dostupné pouze z detailu přepisu z kolekce *Regesten*. Přehled všech souborů s literaturou není k dispozici v ucelené podobě.

Obsah v těle *XML* souborů ze složky *Indices* se vždy vzájemně odlišuje, není zde jednotná konvence. Každý soubor obsahuje celé jméno autora, název díla a případně informace o umístění v archivu. Textové popisy díla obsahují rok vydání a některé i místo vydání. Struktura popisu může zahrnovat informace o zdrojových stránkách nebo svazcích. Vzhledem k variabilitě struktury lze do textového řetězce přidat různé informace o díle.

Složka *Namen*. *Namen* zahrnuje soubory vždy s detaily o jedné konkrétní historické osobnosti. U osob je evidováno jméno, případně titul. U západních jmen je použita interpretace v pořadí „*příjmení*", „*jméno*". Následuje přirozený zápis celého jména včetně přízvisek a titulů. Dále

zde mohou být uvedena místa a data narození, případně úmrtí a další jména, pod kterými je možné osobu identifikovat. V některých záznamech je možné najít i jména blízkých příbuzných (manžel/manželka) v doprovodu dalších detailů, jako výpis povolání. Dále jsou zde explicitně uvedeny reference na všechny textové záznamy, ve kterých osoba figuruje. Odkazováno je do kolekce *Regesten*. Jsou-li pro osobu evidována tištěná díla (tzn. *Gedruckte Werke*), následuje jejich výčet. Pro (budoucí) vyhledávání v rámci archivu byly pro každou osobu stanoveny význačné indexy, které se vytvářejí systematicky postupným přidáváním písmen obsažených v příjmení/titulu osoby (limitace maximálním počtem rovným hodnotě šest). Následuje obecná ukázka struktury zdrojového souboru. Ukázkový zdrojový soubor v původní nepozměněné podobě je možné najít v Příloze D - část *Namen*.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="../view_names.xsl" type="text/xsl"?>
<dr:doc xmlns="http://www.w3.org/1999/xhtml"
xmlns:dr="http://documenta.rudolphina.org/" v="PUBLICATION-DATE">
<title>Prijmeni, Jmeno</title>
<h3>Jmeno Prijmeni</h3>
<!-- Datum narozeni, umrti, informace o pribuznych, alternativni
jmena, povolani apod. -->
<div class="docindex"><b>Quellen:</b><a href="../Regesten/XXX.xml
" title="XXX">XXXX-XX-XX</a>
<!-- ... -->
</div>
<!-- Seznam del -->
<ul> ... </ul>
<dr:ix> IndexA XX XXX XXXX </dr:ix>
</dr:doc>
```

Zdrojový kód 3.3: Struktura dokumentu z kolekce *Namen*

Poznatky:

- **Struktura *XML* souboru**: Každý soubor obsahuje informace o konkrétní historické osobě, jména jsou pevně strukturována, zahrnuty jsou i životní jubilea, jména příbuzných a povolání. Obsahuje výčet odkazů na všechny textové záznamy, kde je daná osoba zmíněna (přesměrování do kolekce *Regesten*). Tištěná díla (*Gedruckte Werke*) jsou v rámci souboru uváděna nepravidelně (v závislosti na dostupnosti dané informace).
- **Struktura *XML* hlavičky:** Hlavička souboru obsahuje informace o poslední verzi (časové razítko). Zobrazení probíhá skrze *style-sheet*, obdobně jako u kolekce *Archives*.
- **Životní data a další identifikátory:** Životní data osoby a další identifikátory (* - datum narození; † - datum úmrtí; ∞ - datum, místo sňatku a jméno choti) jsou v souboru chápány jako jeden dílčí řetězec bez *tagů*. Zápis se jeví být konzistentní, avšak výskyt této struktury je nepravidelný, v závislosti na dostupnosti informace u osoby.
- **Informace o příbuzných:** Informace o příbuzných je vždy odřádkována (<*br*/>). Zápis je konzistentní, ale nevyskytují se pravidelně.

- **Povolání:** U výpisu povolání se naskytuje situace, kdy některé záznamy nejsou odřádkovány. V jiných případech jsou povolání odřádkována tagem <*br*/>, což vede k nekonzistenci zápisu.
- **Seznam děl:** Seznam děl se vkládá mezi reference na záznamy a vyhledávací indexy, jedná se o neseřazený seznam prostých textů. Formát je konzistentní, avšak výskyt nepravidelný. Uvozeno slovním spojením *Gedruckte Werke*. Viz ukázka:

```
1     <ul>Julius Alessandrini a Neustain, Antargenterica pro
      Galeno (Venetiis 1552). [ONB BE.9.N.65.(3).]< /ul>
2
```

Zdrojový kód 3.4: Ukázka výčtu seznamu děl v dokumentu z kolekce *Namen*

- **Odkazy na přepisy:** Platí, že osoby jsou propojeny skrze hypertextové odkazy se **všemi** textovými přepisy, ve kterých jsou zmíněny. Tato propojenost je zachována pro možnost uživatelské navigace napříč dokumenty.

Strukturu kolekce *Namen* lze charakterizovat jako robustní, avšak s několika nepravidelnostmi týkajících se odřádkování a formátování, což by mělo být vzato v úvahu při následných úpravách zdrojového formátu.

Složka *Regesten*. *Regesten* v rámci *Documenta Rudolphina* představuje kolekci souborů, které obsahují přepisy historických textů. Každý soubor má specifický název, složený z prefixu „*A*" označujícího archiv, dále data původu a pěticiferného indexu. Vybraná podmnožina pro rozbor pochází z Archivu Pražského hradu - *Dvorská komora*. Struktura souboru zahrnuje název záznamu, datum a místo vytvoření, metadata o osobách a reference na zainteresované osoby, následované přepisem původního textu. Dále obsahuje metadata o archivu s odkazy na hierarchické stupně, informace o dřívější katalogizaci a seznam pozdějších děl, která citují přepisovaný originál.

Soubory vykazují konzistentní pořadí elementů v rámci podmnožiny, v níž jsou údaje ohraničeny totožnými *tagy*. Hlavička dokumentu je shodná u všech zdrojů a pro vysázení do *HTML* je využíván identický *XSLT* (*view_regest.xsl*) s doprovodem *CSS*. Každý soubor zahrnuje referenční seznam osob, překlad vlastní textové části a hierarchii archivu, přičemž rozdíly v překládaných textech mohou vyplývat ze stylizačních úprav. Ukázka zdrojového souboru v původní nepozměněné podobě je umístěna v Příloze D - část *Regesten*. Níže následuje generická ukázka struktury zdrojového souboru:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet href="../view_regest.xsl" type="text/xsl"?>
3 <dr:doc xmlns="http://www.w3.org/1999/xhtml"
4         xmlns:dr="http://documenta.rudolphina.org/" v="PUBLICATION-
    DATE" uri="AXXX-XX-XX-XXXXX" a="MSt">
5     <title>TITLE</title>
6     <!-- Metadata o referovanych osobach/jevech (.pi/.si/.pa apod.)
    -->
7     TITLE
8     DATE, LOCATION
```

```
    <div class="namindex">
        <!-- Odkazy na osoby -->
    </div>
    <div class="zitat">
        <!-- Vlastni text prepisu -->
        <!-- Neserazeny seznam -->
        <ul> ... </ul>
    </div>
    <!-- Metadata zdrojoveho archivu (.i1) -->
    <div class="archiv">
        <!-- Reference do archivnich stupnu + cislo prepisu, svazku
   apod. -->
    </div>
    <!-- Drivejsi katalogizace -->
    <!-- Seznam del, citujici zde uvedeny prepis -->
    Copyright XXXX Manfred Staudinger
</dr:doc>
```

Zdrojový kód 3.5: Struktura dokumentu z kolekce *Regesten*

Poznatky:

- **Konzistence struktury:** Pořadí elementů v souborech (v rámci podmnožiny) je konzistentní. Všechny zkoumané soubory obsahují referenční seznam osob, překlad textových částí a hierarchii archivu (včetně zanoření).
- **Vysázení textu:** Vysázení textu přepisu řídí *style-sheet view_regest*, který může odkazovat na kaskádové styly. V tomto kontextu je třeba zmínit vysvětlivky nacházející se v záznamu současně s přepisovanými texty. Platí pro ně, že jsou zvýrazněny a umístěny v hranatých závorkách. Metadata o osobách a archivu jsou konzistentní a uvozena instrukčními znaky - např. *.pi* (uvozuje osoby), *.si* (uvozuje literaturu), *.pa*, *.i1* (uvozuje fyzické umístění původního textu), *.**, *.cs*.), avšak nejsou ve webovém zobrazení vysázena.
- ***Tagy <del>*:** Text přepisu může obsahovat tagy *~~* a *~~*, které ve vizualizaci představují přeškrtnutý text. Výskyt těchto *tagů* je z podstaty nepravidelný.
- Gramatické chyby a archaické zápisy jsou evidovány a zdůrazněny v závorkách.
- **Informace o verzi:** Podobně jako u kolekce *Archive*, zdrojový soubor obsahuje informace o verzi. Verzování umožňuje zobrazení návrhu citace přepisu.
- Přepisy zachovávají odkazy na osoby pomocí funkce *onmouseover = "highlightWords()"*. Uchovány jsou rovněž informace o odkazovaných osobách.
- Při uvádění, že přepisovaný záznam byl doložen v jiných dílech, je vypsán i seznam citující literatury.
- Odkazy do kolekce *Indices* jsou skryty za jménem autora a rokem publikace.

3.2.1.4 Výstup analýzy

Vzhledem k omezenému rozsahu zkoumaných souborů v rámci této analýzy a s ohledem na manuální proces tvorby těchto souborů lze předpokládat, že pravděpodobně existuje řada ne-

srovnalostí, chyb a odchylek, jež by se mohly objevit při zohlednění většího množství souborů.

Zvláště absence databázového systému se jeví jako faktor, který zhoršuje přehlednost. Pro přidání nového obsahu je nutné vytvářet nové *XML* soubory a propojovat je skrze kolekce, což přináší další vrstvu komplexity. Tento styl konfigurace není v souladu s moderními standardy (architekturami) a může omezovat efektivitu a uživatelskou přívětivost webového prostředí.

3.2.2 Databáze Getty ULAN

Union List of Artist Names, zkráceně ***ULAN*** (často označován jako *Sjednocený rejstřík umělců*), je elektronický archiv obsahující jména umělců a doplňkové informace o nich. Tuto unikátní databázi vytvořil a spravuje od 80. let 20. století institut *J. Paul Getty Trust* sídlící v Los Angeles (USA) jakožto součást programu ***Getty Research Institute*** (*GRI,* případně v textu jako *Getty)*. Ačkoli se *ULAN* prezentuje jako seznam záznamů, ve skutečnosti má spíše charakter tezauru - slovníku. Pro představu, v roce 2018 obsahoval přibližně 293 000 záznamů reprezentující umělce a organizace, přičemž se na ně bylo možné odkazovat skrze více než 720 000 unikátních jmen/označení. [The23b] K datu 5. března 2023 pak *ULAN* obsahoval již 525 990 záznamů o fyzických a právnických osobách na které referovalo celkem 1 470 932 unikátních označení. [The23a] Je patrné, že v rozpětí pěti let se archiv zvětšil o necelých 80 %, zatímco zásoba unikátních pojmenování se více než zdvojnásobila.

Projekt *Inventaria Rudolphina* databázi *ULAN* používá k čerpání detailních informací týkajících se umělců, již jsou s daným inventárním záznamem spjati.

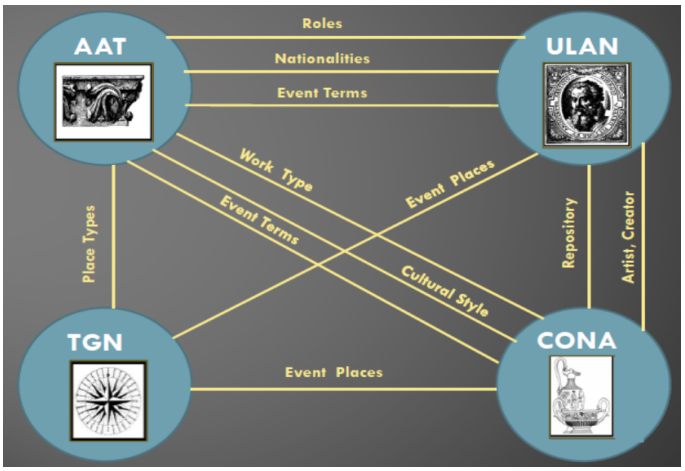
3.2.2.1 Koncept Getty slovníků

Kromě *ULAN*, *Getty* nabízí soubor vzájemně propojených slovníků (*Getty Vocabularies*), které zahrnují *Art & Architecture Thesaurus* (*AAT*), *Getty Thesaurus of Geographic Names* (*TGN*), *Cultural Objects Name Authority* (*CONA*) a *Getty Iconography Authority* (*IA*). *AAT* pokrývá termíny, vztahy a poznámky týkající se umění, architektury a kulturního dědictví, zatímco *TGN* se specializuje na globální tezaurus míst relevantních pro umění a architekturu. *CONA* se zaměřuje na názvy a klasifikaci architektonických a uměleckých děl, včetně informací o současných i historických dílech. *IA* obsahuje jména a data pro ikonografická propojení, s důrazem na východní [The19a] (asijskou) tématiku a odkazy na další zdroje pro evropská témata. Tyto slovníky poskytují komplexní zdroje pro odborníky v oblasti umění a architektury, podporující rozvoj a udržení kvalitního výzkumu v těchto oborech. [The19b]

Všechny tyto slovníky představují strukturované informační zdroje z oblasti výtvarného umění, které zahrnují umění, architekturu, dekorativní umění, archivní materiály, uměleckou konzervaci a další související aspekty. Jsou přístupné prostřednictvím online rozhraní a jsou dostupné v různých formátech. Splňují[17] mezinárodní standardy pro strukturované a řízené slovníky a poskytují odborné informace pro katalogizační agentury, výzkumníky a poskytovatele dat. *Getty Vocabularies* obsahují odkazy na jiné informační zdroje, kde se témata vzájemně prolínají. Nicméně, jsou výjimečné svým globálním záběrem a umožňují vzájemné propojení

[17] https://www.getty.edu/research/tools/vocabularies/

archivních a současných informací. Dále se dokážou přizpůsobit místy diskutabilní a nejednoznačné povaze informací v oblasti dějin umění a umožňují stanovit složité vztahy jak v rámci samotných slovníků, tak mezi nimi. Proto nelze tyto zdroje chápat pouze jako primitivní slovníky, nýbrž spíše jako rozsáhlé znalostní základny. [The23b] Níže (na snímku 3.12) je uveden diagram znázorňující vzájemné provázání *Getty* slovníků.



Obrázek 3.12: Diagram znázorňující provázání *Getty Vocabularies*. [Har14]

3.2.2.2 Funkcionalita a účel

Slovníky se využívají pro katalogizaci, čímž umožňují konzistentní dokumentaci a volbu mezi preferovanými termíny nebo synonymy. Dále usnadňují propojování dat prostřednictvím unikátních identifikátorů a podporují hledání informací díky rozsáhlé sadě synonym a kontextových dat, jež zvýrazňují vztahy mezi koncepty. ULAN je dostupný ve formátech jako *Linked Open Data* (Otevřená data), *XML*, relační tabulky a přes *API* webových služeb, přičemž *online* vyhledávací rozhraní *Getty* slovníků patří mezi nejčastěji[18] navštěvované nástroje na webu *Getty*, sloužící jak katalogizátorům, tak výzkumníkům. [The23a]

Hlavním cílovým publikem *Getty* slovníků jsou výzkumníci dějin umění, muzea, archivy, konzervátoři a další odborníci, kteří hledají informace o uměleckých konceptech. Kromě toho jsou využívány i studenty a širokou veřejností.

3.2.2.3 Rozsah a struktura

ULAN se zaměřuje na osobnosti a organizace z oblasti výtvarného umění a přidružených disciplín, jejichž díla jsou typicky shromažďována muzei a institucemi kulturního dědictví. Databáze obsahuje rozsáhlé záznamy, které zahrnují jména, vztahy, biografické informace, jakožto i údaje o mecenáších a umístění děl. Subjekty v *ULAN* se dělí do kategorií: *PERSONS, ARTISTS* (osoby, umělci), *CORPORATE BODIES* (právnické osoby), *NON-ARTISTS* (neumělci), *UNKNOWN PEOPLE BY CULTURE* (generické osoby) a *UNIDENTIFIED NAMED PEOPLE* (anonymní tvůrci). Každý záznam je identifikován unikátním numerickým kódem (viz ukázka záznamu

[18] Online Search [The23a]

zlatníka *Albrechta Dürera* - obrázek 3.13), databáze pokrývá časové období od antiky po současnost a zahrnuje různé formy jmen a příjmení, včetně pseudonymů a přízvisek v různých jazycích, přičemž pro každý záznam je určeno právě jedno preferované jméno. [The23a]

Obrázek 3.13: Ukázka záznamu v databázi *ULAN* (z důvodu přehlednosti uvedena jen část dostupných atributů). [The23c]

Detailní informace ohledně údajů uvedených u záznamů a osvědčené postupy pro přidávání i editaci záznamů lze získat přímo na stránkách *Getty*[19]. Analýza a popis atributů je nad rámec této diplomové práce.

3.2.3 Klasifikační systém *ICONCLASS Illustrated Edition*

Iconclass Illustrated Edition představuje komplexní klasifikační systém v *online* podobě, sloužící ke klasifikaci obsahu obrazů. Historie *Iconclass* sahá do myšlenek *Henriho van de Waala*, profesora dějin umění na Leidenské univerzitě, který položil základy tohoto systému ve 40. letech 20. století. Aktuální (tj. třetí) verze prohlížeče[20] *Iconclass* čítá na 28 000 unikátních klasifikačních typů [ree18] a nabízí rozsáhlý korpus klasifikovaných obrázků, které lze využít pro srovnání a inspiraci v oblasti ikonografie. [ICO22e]

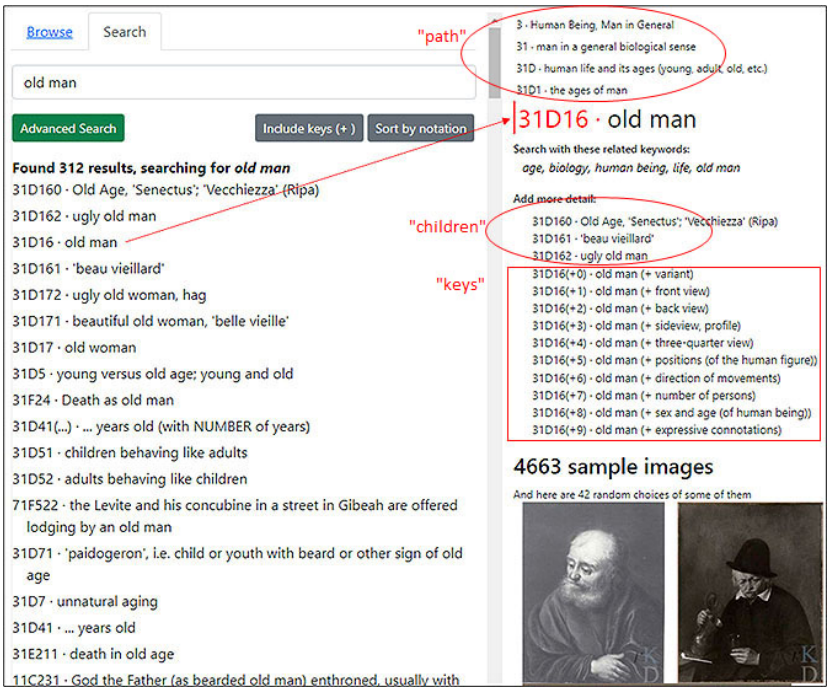
Systém nahlíží na svět obrazů skrze kategorie pojmů. Rozsah každé z těchto kategorií je demonstrován výběrem ukázek obrazů zařazených do dané kategorie. Každá kategorie se může skládat z podkategorií a toto hierarchické členění pokračuje v obdobném duchu i na dalších úrovních. Tímto způsobem dochází k formování klasifikačních ***konceptů***, též v kontextu této práce označovaných jako *číselníky* z analýzy detailu záznamu databáze *Inventaria*. Ty se uplatňují ke klasifikaci tématiky, respektive obsahové stránky inventárních položek. [ICO22e]

3.2.3.1 Notace

Klasifikační systém *IconClass* využívá alfanumerickou notaci společně s textovou definicí obsahu pro indexaci a popis tématiky obrazových děl. [ICO22e] Notace začíná číslicí 0-9, odpovídající jedné z deseti hlavních kategorií, a umožňuje popis i komplexních témat. Každá notace je

[19] https://www.getty.edu/research/tools/vocabularies/guidelines/index.html#ulan
[20] https://iconclass.org

dílčí částí hierarchické struktury, obsahující rozšiřující ikonografické pojmy. Notace je označována jako „***textový korelát***", což dle autorů představuje skutečnou definici konceptu. Koncepty jsou v rámci kategorií řazeny od nejvíce obecného ke specifickým, s možností rozdělení do devíti podkategorií přidáním číslic. Specifičnost je možné dále zvyšovat přidáváním velkých písmen a číslic bez mezer, což zároveň může zlepšit čitelnost notace. [ICO22d]



Obrázek 3.14: Struktura *IconClass* konceptu (*31D16 old man*). [ICO22f]

3.2.3.2 Dokumentace

Systém *Iconclass* měl za cíl vytvořit univerzální klasifikaci témat výtvarných děl. Během let prošel *Iconclass* mnoha revizemi a byl předmětem diskusí a konferencí. Zároveň se stal zdrojem pro získávání dodatečných informací pro digitální katalogy a různé webové stránky zabývající se historií umění. [ICO22c] Publikace založené na *Iconclass* slouží jako zdroje pro budoucí „technologickou archeologii", přičemž datové sady *Iconclass* se ukázaly být užitečné pro analýzu obrazů s využitím strojového učení. Dokumentace systému zahrnuje rozsáhlou bibliografii a usiluje o shromažďování pramenů jak v publikované, tak nepublikované formě, s odkazy na texty v plném znění, pokud jsou k dispozici. [ICO22a]

Dokumentace obsahuje seznam webových stránek muzeí, knihoven a dokumentačních institucí, které implementují *IconClass* pro sjednocení přístupu k ikonografickým informacím. Příkladem instituce využívající *IconClass* je právě i *Inventaria Rudolphina*, jež přejímá externí odkazy na koncepty, jejich názvy a seznam klíčových slov. Převzaté informace uchovává ve formátu *JSON* obsahující data přímo z *IconClass*, a to včetně klasifikačního názvu (klíč *txt*), klíčových slov (klíč *kws_all*) a specifické notace (klíč *iconclass_external_id*). Ukázku tohoto dokumentu lze nalézt v Příloze E. Bylo zjištěno, že existuje omezená jednotnost metodologie přístupu k ikonografickým informacím mezi institucemi využívajícími *IconClass*. Cílem současné verze prohlížeče je podpořit standardizaci v této oblasti. [ICO22b]

Datové modelování 4

V této kapitole jsou představeny obecné principy a techniky datového modelování, které jsou následně uplatněny při návrhu vylepšení sbírkových databází *Inventaria* a *Documenta*.

V dnešním daty řízeném světě mají data neocenitelnou hodnotu pro organizace ve všech odvětvích. Od základních operací až po komplexní rozhodovací procesy, jak efektivní správa, tak analýza dat hrají klíčovou roli v úspěchu podniku. Datové modelování je základním procesem, který umožňuje organizacím převést rozsáhlé množství nezpracovaných dat do strukturované a uplatnitelné formy. Následující odstavce poskytují přehled o datovém modelování, jeho významu, typech, procesu a klíčových aspektech, které je třeba zvážit pro úspěšné využití dat. [Sim21]

Datové modelování odkazuje na proces vytváření vizuální reprezentace dat a jejich vzájemných vztahů v rámci informačního systému nebo databáze. Jeho hlavním cílem je abstrahovat a definovat, jak budou data strukturována, uložena a vzájemně propojena, aby byla zajištěna jejich přesnost, efektivita a snadná správa. Datové modelování poskytuje plán pro budování a úpravu databázových systémů, umožňuje lepší porozumění datům a usnadňuje komunikaci mezi různě kvalifikovanými *stakeholdery* (například mezi technickými a obchodními týmy). [Sus23]

Efektivní datové modelování přináší organizacím mnoho výhod. Umožňuje lepší porozumění a správu dat, zlepšuje jejich kvalitu a konzistenci, a tím zvyšuje efektivitu rozhodovacích procesů. [Sim21] Datové modelování také usnadňuje dodržování interních předpisů tím, že poskytuje jasnou dokumentaci o struktuře, použití a správě dat. Kromě toho umožňuje organizacím rychle reagovat na změny v obchodním prostředí a technologickém vývoji. [Sus23]

4.1 Vývoj datového modelování

Datové modelování prošlo během let významným vývojem, což odráží pokroky v technologiích, metodách správy dat a firemních požadavcích. Z počátků, kdy se modelování zaměřovalo převážně na manuální a konceptuální práci, došlo k posunu ve využívání automatizovaných nástrojů, které umožňují pracovat na různých úrovních abstrakce. Současné trendy zahrnují využívání specializovaných modelovacích jazyků a standardů typu *SQL* a *UML* a integraci modelování s dalšími procesy správy dat. Tento vývoj naznačuje rostoucí význam efektivního řízení dat v současném obchodním ekosystému. [Sim21]

4.2 Nástroje pro datové modelování

Datové modelování lze vnímat jakožto proces uplatňování určitých technik a metodologií pro konverzi dat do užitečné podoby. Toho se dosahuje pomocí nástrojů, které pomáhají vytvářet požadovanou strukturu databáze z diagramů. Tyto nástroje mohou usnadnit i propojování dat. Existují tři základní techniky datového modelování: [Sim21]

- ***Entity-Relationship diagram*** **(*ERD*)**: Entitně-relační diagram. Tato technika slouží k modelování a návrhu relačních databází.
- ***Unified Modeling Language*** **diagram tříd (*UML*)**: *UML* je standardizovaný modelovací jazyk určený k vizualizaci, specifikaci, návrhu a dokumentaci softwarových systémů.
- ***Data Dictionary*** **(Technika modelování datového slovníku)**: Tato technika zahrnuje tabulkovou definici nebo reprezentaci datových aktiv.

4.3 Typy datových modelů

Existuje několik různých přístupů k datovému modelování, z nichž každý má své vlastní uplatnění v závislosti na specifických potřebách a cílech projektu: [Sim21]

- ***E-R*** **(*Entity-Relationship*) model**: Tento model se zaměřuje na reálné entity a vztahy (relace) mezi nimi. Vytváří soubory entit a vztahů, včetně obecných atributů a omezení. Tento model je často používán při návrhu relačních databází.
- **Hierarchický model**: Hierarchický model organizuje data do podoby stromu s jedním kořenem, ke kterému jsou připojeny další údaje. Hierarchie začíná u kořene a je rozšiřována dle pravidel stromu. Tento model se často aplikuje v systémech správy souborů.
- **Síťový model**: Tento model umožňuje vztahy *M:N* (*many-to-many*) mezi propojenými uzly. Data jsou uspořádána ve struktuře podobné grafu, což umožňuje komplexní vztahy.
- **Relační model**: Relační model organizuje data do tabulek s řádky a sloupci, což usnadňuje identifikaci vztahů mezi daty. Relační databáze jsou široce používané napříč oblastmi *IT*.
- **Objektově orientovaný model**: Model definuje databázi jako kolekci objektů nebo znovupoužitelných softwarových komponent s příslušnými metodami a funkcemi. Je často používán v objektově orientovaném programování.
- **Objektově-relační model**: Tento model kombinuje prvky objektového modelu s jednoduchostí relačního datového modelu. Je využíván v případech, kdy je potřeba sloučit objektově orientované *paradigma* s relačními databázemi.
- **Dimenzionální datový model**: Model prezentuje entity ve třírozměrných tabulkách. Variantou tohoto modelu je multidimenzionální datový model, kde každá tabulka obsahuje více než tři sloupce. Tento model slouží k uchovávání historických dat.
- **Dokumentově orientovaný model**: Tento přístup se zaměřuje na ukládání a správu dat ve formě dokumentů, obvykle ve formátech jako *JSON* nebo *XML*, místo tradičních tabulek nebo relací. Agregace dat do dokumentů umožňuje flexibilní a hierarchické

uspořádání. Tento model je vhodný pro aplikace vyžadující rychlou vývojovou iniciativu a schopnost pracovat s nestrukturovanými nebo semistrukturovanými daty.

4.3.1 Obecný proces datového modelování

Proces datového modelování je zásadním krokem při vývoji databázových a informačních systémů, který zahrnuje definici klíčových komponent datového modelu a jejich vzájemných vztahů. Tento proces začíná identifikací entit (např. obrazů, inventářů, autorů, …), které by byly reprezentovány tabulkami v databázi. Každá entita je poté charakterizována sadou atributů, jež se objevují jako sloupce v tabulkách a obsahují specifické informace, například identifikátor a název obrazu či inventáře. [Sus23]

Záznamy, prezentované řádky v tabulkách, představují konkrétní instance entit, jako jsou jednotlivé obrazy s unikátními identifikátory, názvy a parametry. Vztahy mezi entitami, definované jako relace *1:1*, *1:N*, nebo *M:N*, odhalují jak jsou entity vzájemně propojené. Kupříkladu vztah mezi obrazem a autorem, kde jeden obraz může náležet k jednomu autorovi, zatímco jeden autor mohl vyhotovit více obrazů. Dále má každá entita primární klíč, což je jedinečný identifikátor, který umožňuje jednoznačně identifikovat každý záznam v tabulce. Cizí klíče pak slouží k vytvoření vztahů mezi tabulkami, skrze odkazování na primární klíče v jiných tabulkách. Tím umožňují propojení informací mezi různými entitami. [Sus23]

Pochopení a správná implementace těchto základních konceptů jsou základem pro úspěšné datové modelování a to nejen relačních databází. Proces návrhu datového modelu je popsán i pro *MongoDB* v kapitole 5. jakožto zástupce dokumentových databází.

4.3.2 Úrovně datové abstrakce

Datové modelování zahrnuje několik úrovní vizuální abstrakce, které usnadňují porozumění a práci s daty: [Sim21]

- **Konceptuální úroveň** zahrnuje vysokoúrovňový pohled na data a jejich vztahy, což usnadňuje komunikaci a porozumění mezi různými zainteresovanými stranami.
- **Logická úroveň** přináší detailnější pohled na data, jejich strukturu a vztahy, ale stále nepřihlíží k fyzickému uložení dat.
- **Fyzická úroveň** se zaměřuje na konkrétní technické detaily týkající se uložení dat v databázi, včetně definic datových typů a indexů.

Z úrovní abstrakce lze odvodit i tři hlavní typy modelů: **konceptuální**, **logický** a **fyzický**. Konceptuální model poskytuje vysokoúrovňový přehled o entitách a vztazích bez zabíhání do technických detailů. Logický model přidává detaily o entitách, atributech, primárních a cizích klíčích a specifikuje, jak jsou data mezi sebou provázána. Fyzický model se zaměřuje na implementaci, včetně stanovení datových typů a struktur tabulek. [Sus23]

Proces datového modelování zahrnuje několik kroků, od počátečního shromažďování požadavků po konečnou implementaci v databázovém systému.

Tyto tři výše uvedené datové modely jsou vytvářeny postupně, přičemž každý model staví na předchozím. Začíná se s definicí požadavků a konceptuálním návrhem. [Sim21] I když neexis-

tují pevně stanovená pravidla, je žádoucí identifikovat entity, které budou v modelu zahrnuty, a následně definovat jejich vzájemné vztahy. Pokračuje se vytvořením logického modelu, který pomáhá identifikovat a shromáždit všechny požadavky kladené na systém a také porozumět tomu, jakým způsobem data proudí v rámci dílčích *business* procesů. Logický model je dále převeden na fyzický model, který specifikuje technické detaily, a následuje již implementace. Tento proces vyžaduje úzkou koordinaci mezi datovými modeláři, vývojáři, obchodními analytiky a dalšími *stakeholdery*, aby bylo zajištěno, že výsledný model splňuje všechny obchodní a technické požadavky. [Sus23]

Při vývoji může průběžné testování částí datových modelů vést ke včasnému detekování chyb a problémů, což se projeví snížením nákladů a minimalizací rizika výpadků funkcionality u koncových uživatelů. [Sus23]

4.3.2.1 Vytvoření logického datového modelu

Vytvoření logického datového modelu je klíčovým krokem ve vývoji efektivní databáze, který převádí vysokoúrovňové návrhy z konceptuálního modelu do detailně specifikované struktury. Tento proces zahrnuje: [Sus23]

- **Identifikaci atributů**: Pro každou entitu se identifikují všechny relevantní atributy, které popisují její vlastnosti.
- **Výběr primárních klíčů**: Pro každou entitu se vybírá jedinečný identifikátor (primární klíč), který umožňuje její jednoznačnou identifikaci.
- **Nalezení relací**: Definují se vztahy mezi entitami (*1:1*, *1:N*, nebo *M:N*).
- **Řešení *M:N* relací**: Transformace těchto vztahů tak, aby byly v relačním modelu reprezentovány efektivně. Standardně vede na rozklad.
- **Normalizace**: Aplikace normalizačního procesu s cílem minimalizovat redundanci a vylepšit úroveň konzistence dat.

Logická normalizace. Logická normalizace je proces, který uspořádává data v logickém modelu tak, aby se minimalizovala redundance a zvýšila konzistence. Klíčové cíle zahrnují: [Sus23]

- **Eliminace datových duplikátů**: Redukce opakování informací mezi tabulkami.
- **Zajištění správného umístění atributů**: Atributy jsou správně rozděleny mezi entity.
- **Podpora snadné údržby**: Zjednodušení správy a dotazování v datovém modelu.
- **Optimalizace stability**: Zvýšení robustnosti a efektivity datové struktury.
- **Zvýšení flexibility**: Model se stává adaptabilnějším na změny.

Závislost logického modelu na sémantické vrstvě. Logický datový model je integrální součástí sémantické vrstvy, která funguje jako most mezi fyzickým uložením dat a koncovými uživateli. Sémantická vrstva umožňuje netechnicky orientovaným uživatelům přístup k datům v kombinaci se zjednodušeným pohledem na data. Dále definuje metriky a výpočty, čímž umožňuje data analyzovat s minimálními technickými znalostmi. [Sus23]

4.3.2.2 Vytvoření fyzického datového modelu

Konverze logického modelu na fyzický zahrnuje specifikaci, jakým způsobem budou data v databázi uložena: [Sus23]

- **Konverze entit na tabulky**: Entitám odpovídají tabulky, atributům sloupce.
- **Definice vztahů a klíčů**: Vztahy jsou reprezentovány pomocí cizích klíčů.
- **Specifikace datových typů**: Každý atribut je definován konkrétním datovým typem.

Po dokončení struktury fyzického modelu je systém připraven na implementaci a následné naplnění daty.

4.4 Výhody a omezení datového modelování

Jak bylo zdůrazněno v předchozích odstavcích, modelování představuje zásadní proces v rámci vývoje softwarových aplikací a databázových systémů. Datové modely hrají klíčovou roli v uspořádání a analýze dat napříč průmyslovými odvětvími, jako jsou *e-commerce*, finanční služby a právě vývoj software, kde napomáhají v rozhodovacích procesech, zefektivnění operací a zvyšování bezpečnosti. Motivace k realizaci modelování spočívá v řadě výhod zmíněných v úvodních odstavcích této kapitoly. [Sim21] Navzdory uvedeným benefitům existují také některá omezení a výzvy, které je nutné brát v potaz: [Sim21]

- **Omezená flexibilita**: Datové modely nemusí být pružné. Ve výsledku se tak může zkomplikovat adaptabilita na měnící se požadavky, či datové struktury.
- **Komplexnost**: Modely mohou být složité a obtížné k pochopení, což může ztížit zapojení zúčastněných stran a vyústit neefektivní spoluprací.
- **Časová náročnost**: Datové modelování může být časově náročný proces, zejména při návrhu komplexních databázových systémů.

4.4.1 Běžné chyby a osvědčené postupy při datovém modelování

Přestože datové modelování nabízí řadu výhod, v praxi se lze často setkat s plejádou běžných chyb, které mohou značně ovlivnit kvalitu a funkčnost výsledných datových modelů. K těmto chybám se řadí nedostatečná flexibilita modelu, přílišná komplexita tabulek, nevhodné modelovací schéma, opomenutí potřeb koncových uživatelů, statický pohled na data bez možnosti jejich aktualizace, nekonzistentní *granularita* a pojmenování, příliš mnoho složitých pohledů, nedostatečná komunikace mezi *stakeholdery*, a vnímání modelování za jednorázový úkol bez potřeby dalších revizí a údržby. [Sus23]

Aby bylo možné těmto chybám předcházet a zajistit tak vytvoření kvalitního modelu, je důležité dodržovat soubor osvědčených metod a nejlepších postupů. Je žádoucí začít s hlubokým porozuměním podnikovým požadavkům, což je prerekvizita k tomu, že datový model bude odpovídat potřebám a cílům organizace. Zjednodušení a vizualizace datového modelu představují klíčovou prevenci vůči nežádoucím komplikacím, usnadňující pochopení modelu a jeho

údržbu. Zaměření se výhradně na relevantní data pro řešení obchodních záležitostí napomáhá vyhnout se informačnímu přetížení a udržování modelu v efektivní podobě. [Sus23]

Další důležitý aspekt spočívá v průběžné validaci modelu během jeho vývoje, aby bylo možné včas identifikovat a řešit potenciální problémy. Kompletní a přesná dokumentace všech aspektů modelu, zahrnující entity, vztahy a pravidla, je nezbytná pro jeho správné pochopení a další rozvoj. Spolupráce mezi všemi zainteresovanými stranami je klíčová pro zajištění, že datový model bude efektivně sloužit svému zamýšlenému účelu a bude schopen se přizpůsobit budoucím změnám. [Sus23]

4.5 Shrnutí datového modelování

Datové modelování představuje klíčový nástroj pro organizace, umožňující jim lépe využívat svá data pro strategické plánování, informované rozhodování, identifikaci nových příležitostí a zlepšení obchodních procesů. V této souvislosti je zásadní dodržovat osvědčené postupy a vyhýbat se běžným chybám pro jeho úspěšné využití v praxi. Tato praxe poskytuje solidní základ pro rozvoj databázových systémů, které jsou nezbytné pro efektivní ukládání, správu a získávání informací, zatímco korektně provedené datové modelování zvyšuje úroveň pochopení dat, jejich kvalitu a rozvíjí spolupráci mezi různými zaměstnanci v organizaci.

Význam datového modelování přesahuje pouhé technické aspekty a stává se nezbytným krokem pro zajištění konkurenceschopnosti a inovace v dynamickém obchodním prostředí. Pro úspěch v současném konkurenčním prostředí je nezbytné, aby organizace věnovala patřičnou pozornost procesu datového modelování, jeho neustálému zdokonalování a adaptaci na měnící se požadavky.

Datové modelování v *MongoDB*

5

Tato kapitola rozšiřuje obecné poznatky z předchozího textu, věnovaného datovému modelování, a zaměřuje se již konkrétně na modelování v prostředí *MongoDB*.

Datové modelování v *MongoDB* je zásadní pro efektivní využití této *NoSQL* databáze, přičemž klíčovou výzvou je dosažení rovnováhy mezi potřebami aplikace, výkonností databáze a vzory (*patterny*) pro vyhledání a přístup k datům. *MongoDB* nabízí flexibilní schéma, které umožňuje efektivní mapování datových entit a podporuje různé datové modely pro optimální využití v různých aplikacích. Cílem je vytvořit strukturu dokumentů a kolekcí. [Mon23b]

Na rozdíl od tradičních relačních databází, *MongoDB* umožňuje ukládat dokumenty bez předchozího stanovení pevného schématu. Tato schématická flexibilita umožňuje, že dokumenty v jedné kolekci nemusí mít stejnou sadu polí, a datové typy polí se mohou mezi dokumenty lišit. Tato vlastnost usnadňuje adaptaci databáze na dynamicky se měnící potřeby aplikací a umožňuje snadné uzpůsobení schématu bez nutnosti jeho kompletní redefinice. [Mon23b]

Při vývoji datového modelu je třeba analyzovat všechny čtecí a zapisovací operace aplikace s ohledem na *atomicitu*, model vnořených dat, transakce mezi více dokumenty, indexaci a *sharding* - všechny tyto aspekty jsou představeny v níže uvedených sekcích. [Mon23c]

5.1 Proces datového modelování v *MongoDB*

Aby bylo zajištěno, že datový model bude mít logickou strukturu a dosáhne optimálního výkonu, je nezbytné naplánovat a navrhnout databázové schéma ještě před nasazením do produkčního prostředí. Proces vytváření datového modelu lze charakterizovat následujícími návrhovými kroky: [Kha23]

1. **Identifikace požadavků aplikace**: Před návrhem datového modelu je důležité identifikovat požadavky aplikace a pochopit, jak budou data využívána. To zahrnuje identifikaci datových entit, jejich vztahů a dotazů, které budou nad daty prováděny.

2. **Navržení schématu dokumentu**: Na základě požadavků aplikace se navrhne schéma dokumentu pro uložení dat. Schéma by mělo reflektovat vztahy mezi datovými entitami a být optimalizováno pro dotazy.

3. ***Normalizace* nebo *denormalizace* dat**: V závislosti na požadavcích mohou být data *normalizována* nebo *denormalizována*. Normalizace zahrnuje rozdělení datových entit na menší, lépe zpracovatelné části pro snížení redundance a zlepšení integrity dat. Denormalizace zahrnuje zkombinování souvisejících datových entit do jednoho dokumentu pro zvýšení výkonu dotazování. Ukázky viz sekce 5.4.

4. **Optimalizace struktury dokumentu**: Po navržení schématu je důležité optimalizovat strukturu dokumentu pro výkon. To se skládá z použití vhodných datových typů, minimalizace výskytu vnořených dokumentů a vyhýbání se rozsáhlým polím.

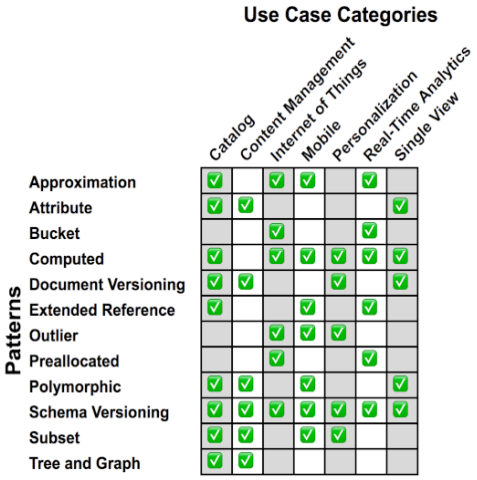
5. **Validace datového modelu**: Před nasazením datového modelu je důležité jej ověřit skrze testování na vzorových datech a provádění dotazů pro verifikaci očekávaného výkonu.

Proces skládající se z těchto kroků umožňuje nejen stanovit a předvídat potřeby aplikace v reálném provozu, ale také efektivně strukturovat data.

Pro zohlednění specifických aplikačních potřeb při návrhu datového modelu v *MongoDB* je významné zvážit různé návrhové vzory (*patterny*). Tyto vzory poskytují osvědčené strategie pro optimalizaci struktury, výkonu a správy dat a tím vypomáhají při řešení konkrétních výzev. Níže je uvedeno celkem dvanáct základních návrhových vzorů, které by měly být při návrhu schématu uváženy: [Aro22]

- ***Approximation*** (Aproximační vzor): Optimalizuje ukládání dat tím, že zachovává pouze přibližné hodnoty. Snižuje počet zápisů a výpočtů.
- ***Attribute*** (Atributový vzor): Zaměřuje se na indexaci a dotazování jen u specifické podmnožiny polí v rozsáhlých dokumentech.
- ***Bucket*** (Vzor *vědra*): Efektivní pro *streamovaná* data nebo aplikace internetu věcí, snižuje počet dokumentů a usnadňuje přístup k datům provedením agregace.
- ***Computed*** (Výpočtový vzor): Minimalizuje potřebu opakovaných výpočtů předpočítáním výsledků buď při operaci zápisu nebo v pravidelných intervalech.
- ***Document Versioning*** (Vzor verzování dokumentu): Umožňuje společnou koexistenci více verzí dokumentu pro efektivní správu.
- ***Extended Reference*** (Rozšířený referenční vzor): Omezuje potřebu spojení vkládáním jen často používaných polí, zjednodušuje strukturu a zvyšuje výkon.
- ***Outlier*** (Vzor pro výjimky): Navrhuje datové modely a dotazy s ohledem na běžné použití, minimalizuje vliv vymykajících se hodnot.
- ***Pre-Allocation*** (Vzor prealokace): Zlepšuje výkon prostřednictvím *prealokace* prostoru pro dokumenty, u nichž je struktura předem známá.
- ***Polymorphic*** (Polymorfní vzor): Poskytuje flexibilitu pro dokumenty s podobným účelem, ale rozdílnou strukturou.
- ***Schema Versioning*** (Vzor verzování schématu): Umožňuje adaptaci (změnu) schématu během životního cyklu aplikace bez nutnosti odstávky nebo technického dluhu.

- ***Subset*** (Podmnožinový vzor): Zvyšuje výkon ukládáním pouze části dat, která jsou aktivně využívána, a umožňuje lepší využití operační paměti (*RAM*).
- ***Tree*** (Stromový vzor): Ideální pro hierarchická data a jejich efektivní správu v dynamických strukturách.



Obrázek 5.1: Hlavní návrhové vzory pro databázové schéma a odpovídající případy užití. [Aro22]

Zahrnutí těchto návrhových vzorů do procesu datového modelování umožňuje navrhnout databázová schémata, která jsou nejen efektivní a škálovatelná, ale také vhodně přizpůsobená aplikačním potřebám. [Aro22]

Na základě uvedeného výčtu návrhových vzorů a analýzy zdrojového kódu se jeví, že *Inventaria Rudolphina* uplatňuje kombinaci několika odlišných návrhových vzorů, jmenovitě ***Extended Reference***, ***Attribute***, a pravděpodobně i vzor ***Polymorphic***. Následuje podrobnější náhled:

- ***Extended Reference***: Tento vzor je aplikován skrze použití identifikátorů a referencí (např. *getty_external_id*), což umožňuje efektivní spojování dat mezi dokumenty bez nutnosti denormalizovaného ukládání všech detailů přímo ve struktuře jednoho dokumentu.
- ***Attribute Pattern***: *Pattern* se zaměřuje na optimalizaci indexace a dotazování specifických polí v rozsáhlých dokumentech. Vzhledem k tomu, že dokument obsahuje specifická pole jako například *subject*, sear*ch_subject,* a *concordances*, která mohou být často dotazována, je pravděpodobné, že vzor *Attribute* byl využit pro zefektivnění právě těchto operací.
- ***Polymorphic Pattern***: Umožňuje flexibilitu pro dokumenty, které mají podobný účel, ale liší se ve struktuře. Vzhledem k rozmanitosti objektů uvnitř pole *object* a jejich různorodým atributům (např. *material*, *type*, *dimensions*), by mohl být tento vzor využit k zajištění flexibilní manipulace s různými typy uměleckých děl.

Strategie zkombinování několika návrhových vzorů může vést k zajištění efektivní manipulace s rozsáhlými a různorodými daty a rychlého vyhledávání.

5.2 Základní koncepty modelování v *MongoDB*

Jak již bylo zmíněno v úvodu kapitoly věnované modelování v *MongoDB*, cílem je vytvořit strukturu, která přesně reprezentuje data a je optimalizována pro potřeby aplikace. V *MongoDB* je datový model vytvořen pomocí *BSON* dokumentů. [Kha23] Následuje výčet některých ze základních konceptů datového modelování *v MongoDB*: [Kha23]

- **Dokumenty:** Dokument je chápán jako sada dvojic *klíč-hodnota*, kde každý klíč je název pole a hodnota může být libovolného (podporovaného) datového typu. Dokumenty v *MongoDB* mohou být vnořené, což znamená, že pole může obsahovat další dokument nebo pole dokumentů.
- **Kolekce:** Kolekce je skupina dokumentů, které se vyznačují podobnými poli a jsou organizovány dohromady. Lze je vnímat jako analogii k tabulkám v tradičních *SQL* databázích.
- **Pole:** Pole[21] je dvojice *klíč-hodnota* v dokumentu, která reprezentuje konkrétní atribut nebo segment dat. Každý dokument může mít odlišnou sadu polí v závislosti na reprezentovaných datech.
- **Datové typy:** *MongoDB* podporuje různé datové typy, včetně řetězců, celých čísel, desetinných čísel, *booleovských* hodnot a polí. Každé pole[22] v dokumentu může mít různý datový typ.
- **Návrh schématu:** Návrh schématu zahrnuje definování struktury dokumentu a jeho organizaci tak, aby odpovídal potřebám aplikace. Při návrhu schématu je důležité zvážit, jakým způsobem budou data dotazována a jaké typy indexů budou potřebné pro optimalizaci výkonu.
- ***Normalizace* a *Denormalizace*:** Oba přístupy mohou být použity pro optimalizaci výkonu dotazování a snížení redundance v datech.

Porozuměním těmto základním konceptům modelování mohou vývojáři vyhotovit optimalizované datové modely, které přesně reprezentují data a dostatečně vyhovují aplikačním potřebám.

5.3 Datové typy v *MongoDB*

MongoDB podporuje různé datové typy, které lze uplatnit pro ukládání dat v rámci dokumentů. Tyto datové typy jsou navrženy tak, aby byly v souladu s požadavkem na flexibilitu a škálovatelnost databáze. Podporovány jsou následující datové typy: [Kha23]

- ***String***: Řetězce jsou sekvence znaků a mohou být použity pro ukládání jakýchkoli textových dat. Řetězce mohou být i víceřádkové.
- ***Integer***: Slouží pro ukládání celých čísel. *MongoDB* podporuje *32* a *64bitová* celá čísla.
- ***Double***: Používá se pro ukládání čísel s desetinnými hodnotami.

[21] Pozn.: *Pole* ve smyslu *field*, nikoliv *array* (datový typ odpovídající seznamu seřazených hodnot)
[22] Pozn.: Zde již myšleno jako datový typ *array*

- ***Boolean***: *Booleovské* hodnoty slouží k ukládání binárních hodnot *TRUE* a *FALSE*.
- ***Date***: Nese informace o datu a čase. Data jsou uložena ve formátu milisekundového posunu od počátku *unixové* epochy (tzn. 1. leden 1970).
- ***ObjectID***: *ObjectID* jsou unikátní identifikátory, které jsou v *MongoDB* automaticky generovány při vložení dokumentu. *ObjectID* jsou *12bajtové* hodnoty zahrnující časové razítko, identifikátor stroje a počítadlo.
- ***Array***: Pole se používají pro ukládání seznamů hodnot. Mohou obsahovat prvky jakéhokoli datového typu, včetně dalších polí a dokumentů.
- ***Null***: Reprezentuje absenci hodnoty.
- ***Regular Expression***: Regulární výrazy se používají pro vyhledávání vzorů v řetězcích.

Je patrné, že *MongoDB* podporuje řadu datových typů, které lze použít pro ukládání různých dat. V kombinaci s uplatněním binárního formátu *BSON* je zajištěn efektivní způsob ukládání a načítání dat. [Kha23]

5.4 Struktura dokumentů a relace mezi daty

MongoDB podporuje dva základní přístupy k organizaci, strukturování a modelování vztahů mezi daty: model vnořených dokumentů (denormalizovaný datový model) a model s referencemi (normalizovaný datový model). Každý z těchto modelů se vyznačuje specifickým využitím a výhodami závislými na charakteru dat. [Aro22] Níže na snímcích 5.2 a 5.3 jsou zachyceny struktury ukázkových dokumentů, respektive relace mezi nimi.

5.4.1 Model vnořených dokumentů

Model vnořených dokumentů, často označovaný také jako denormalizovaný datový model, se jeví jako strategická volba v situacích charakterizovaných těsným vztahem mezi dvěma nebo více datovými sadami. Tento model je založen na principu ustanovení vztahů mezi datovými prvky skrze jejich uchování v rámci jedné integrované dokumentové struktury. V závislosti na specifických požadavcích aplikace mohou být relevantní informace uloženy buď v jednoduchém poli nebo v poli obsahujícím další dokumenty. [Aro22]

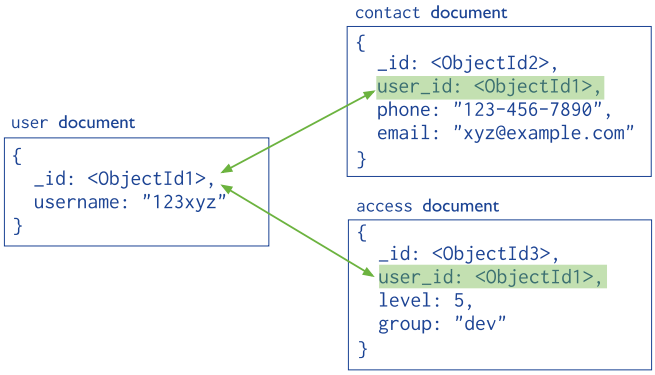
Jedním z klíčových přínosů vnořených dokumentů je značná optimalizace výkonu při čtení dat. Tato metodologie umožňuje seskupení souvisejících datových entit do jednoho dokumentu, což minimalizuje nutnost vykonávání mnohonásobných databázových dotazů a tím zvyšuje efektivitu přístupu k datům. Tento přístup se jeví jako obzvláště vhodný pro scénáře, kde jsou určité datové sady přistupovány a zpracovávány společně na pravidelné bázi, jelikož redukuje potřebu pro distribuované transakce a zjednodušuje databázové operace. [Mon23a]

Obrázek 5.2: Ukázka vnořených dokumentů v rámci dokumentu. [Mon23a]

5.4.2 Model s referencemi

Model s referencemi, respektive normalizovaný datový model, představuje alternativní *paradigma*, jehož cílem je optimalizovat strukturu datového úložiště prostřednictvím minimalizace redundance dat. Tento model se opírá o použití objektových referencí (odkazů) k modelování vztahů mezi jednotlivými datovými prvky nebo dokumenty. Klíčovou charakteristikou normalizovaného modelu je jeho schopnost efektivně dokumentovat *M:N* vztahy mezi datovými entitami bez nutnosti duplikace datových obsahů. [Aro22]

Aplikace normalizovaného modelu se ukazuje jako obzvláště přínosná v případech, kdy je potřeba modelovat rozsáhlé hierarchické datové sady s četnými vztahy napříč různými kolekcemi. Tato metodologie je preferována v situacích, kde u dat dochází k častým změnám nebo pokud jsou data charakterizována složitými vzájemnými vztahy. Redukce duplikace dat vede nejen k uvolnění úložných kapacit, ale především k zajištění vyšší integrity a konzistence dat, což je základním předpokladem pro udržení kvality a spolehlivosti datových systémů. Hodí se pro návrh a implementaci databázových systémů, které vyžadují komplexní správu a analýzu hierarchických nebo propojených datových struktur. [Aro22]



Obrázek 5.3: Ukázka referování mezi různými dokumenty. [Mon23a]

5.5 Atomicita a transakce

Atomicita, též nedělitelnost, operací na úrovni jednotlivých dokumentů v *MongoDB* umožňuje, že i složité aktualizace uvnitř dokumentu jsou atomické. Toto je klíčové pro udržení konzistence dat a zajištění, že buď všechny změny v dokumentu proběhnou úspěšně, nebo se

neprovede žádná. Pro situace vyžadující atomicitu napříč více dokumenty podporuje *MongoDB* tzv. *multi-dokumentové* (distribuované) transakce, které rozšiřují tuto záruku atomicity i na operace zasahující do více dokumentů, ať už v rámci jedné kolekce nebo napříč více kolekcemi. Aplikace musí provádět samostatné čtecí i zapisovací operace pro získání a modifikaci těchto souvisejících dat. [Mon23b]

5.6 Optimalizace a údržba návrhu

Optimalizace datového modelu v *MongoDB* vyžaduje komplexní přístup, který zohledňuje využití dat, duplikaci, konzistenci, indexaci a hardwarová omezení. Správným návrhem a implementací strategií pro práci s daty může vývojář zefektivnit dotazování, zvýšit propustnost operací (*CRUD*), zavést horizontální škálování a tím dosáhnout vysokého výkonu a efektivity aplikací využívajících *MongoDB*. [Mon23b]

5.6.1 Použití dat a výkonnost

Při návrhu datového modelu je klíčové zohlednit, jakým způsobem aplikace databázi využívá. Například, pokud aplikace využívá především nedávno vložené dokumenty, může být vhodné použít tzv. *Capped* kolekce, které omezují velikost kolekce a udržují dokumenty v původním pořadí (*FIFO*), v jakém byly vloženy. To je užitečné pro aplikace vyžadující rychlý přístup k nejnovějším datům nebo pro *logování*, kde je důležitá sekvenční integrita dat. Dále by při návrhu měla být zvážena i správa životního cyklu dat, jako je tomu u *TTL (Time-To-Live)* kolekcí pro expiraci dokumentů po určité stanovené době. Pokud je hlavní potřebou aplikace čtení z kolekce, přidání indexů pro podporu běžných dotazů může výrazně navýšit výkonnost a snížit dobu odezvy aplikace. [Mon23b]

5.6.2 Duplikace dat a konzistence

Duplikace dat mezi kolekcemi může sloužit jakožto taktika pro optimalizaci výkonu čtení, neboť umožňuje agregaci souvisejících informací o různých entitách prostřednictvím jediného dotazu. Avšak, při rozhodování o duplikaci dat je imperativní zvážit frekvenci, s jakou bude vyžadována jejich aktualizace. V případech, kdy dochází k častým aktualizacím, se jako efektivnější alternativa propojení souvisejících dat jeví využití referencí, čímž se minimalizují požadavky na udržování datové konzistence napříč kolekcemi. [Mon23a]

5.6.3 Indexace a sharding

Indexace je fundamentální pro zlepšení výkonnosti dotazů v databázových systémech. Strategické vytváření indexů na polích, která jsou často cílem dotazů, umožňuje vývojářům zabezpečit efektivitu dotazů, i když objem dat narůstá. Je nezbytné provádět pravidelné sledování využití indexů a přizpůsobovat indexační strategii aktuálním trendům v přístupu k datům a dynamice růstu databáze. [Mon23a]

Sharding (*shardování*), neboli horizontální škálování, umožňuje distribuci dat mezi několik serverů v *MongoDB*, což zvyšuje výkonnost a kapacitu systému. Ústředním prvkem tohoto

procesu je výběr *shard key*, který by měl být proveden s ohledem na rovnoměrné rozložení dat mezi dílčí *shardy*. Výběr má přímý dopad na efektivitu a výkonnost dotazování. [Kha23]

5.6.4 Hardwarová omezení

Při návrhu databázových schémat je klíčové brát v úvahu hardwarová omezení, především kapacitu dostupné operační paměti. Větší dokumenty spotřebují významné množství paměti *RAM*, což může vést k nutnosti čtení dat z disku, s negativním dopadem na výkon aplikace. Efektivním návrhem schémat, kde dotazy extrahují pouze relevantní data, lze omezit velikost pracovní sady a tím podpořit výkon. [Mon23a]

V určitých případech je preferováno distribuovat související informace do více kolekcí namísto jejich koncentrování v jediné. Tento přístup může optimalizovat výkon při vysoce náročném dávkovém zpracování, přestože každá kolekce a její indexy zabírají určité množství datového prostoru. Pro kolekce obsahující velké množství malých dokumentů může být efektivním přístupem *rolling-up* (tzv. „srolování") těchto malých dokumentů do větších agregátů, které zahrnují pole vnořených dokumentů. Tato metoda může zlepšit výkon dotazů snížením počtu náhodných přístupů k disku. [Mon23c]

5.7 Shrnutí datového modelování v *MongoDB*

Datové modelování v *MongoDB* nabízí adaptabilitu a výkon pro správu dat, což je zásadní pro optimalizaci aplikací. Klíčem k úspěchu je volba vhodného datového modelu, integrace atomických operací a pečlivé plánování schémat, což společně zvyšuje výkonnost a zlepšuje škálovatelnost aplikací založených na *MongoDB*. Tyto principy umožňují vývojářům plně využít potenciál *MongoDB* a zajistit, že aplikace mohou efektivně reagovat na rozmanité požadavky.

Návrh změn a vylepšení 6

V této kapitole je rozebrána metodika určená k detekci slabých míst projektu *Inventaria Rudolphina*, doplněná o návrh konkrétních řešení a změn, které by mohly být na základě zjištěných nedostatků implementovány.

6.1 Metodika zjišťování nedostatků a možných vylepšení

Tato sekce se zaměřuje na metodiku dotazování, která byla použita k identifikaci a formulaci následných návrhů změn a vylepšení v aplikacích *Inventaria* a *Documenta Rudolphina* s cílem optimalizovat funkčnost datových modelů. Proces dotazování byl zásadní pro získání hlubokého porozumění potřebám uživatelů a technickým výzvám spjatých s aktuálním stavem datových modelů. Metodika kombinovala *on-line* vedené rozhovory s výzkumnými pracovníky *ÚDU AV* Ing. Martinem Medunou a Mgr. et Mgr. Markétou Ježkovou, Ph.D. a realizaci vlastní analýzy zdrojových souborů a způsobů použití aplikace. Viz předešlé podsekce 3.1.4 a 3.1.2.

6.1.1 Metodologický rámec

1. **Definice cílů a otázek:** Před zahájením rozhovoru byly stanoveny cíle dotazování - hlouběji porozumět, jak se aplikace používá, identifikovat problémy, priority uživatelů a potenciální oblasti pro zlepšení v kontextu datového modelu. Příprava otázek byla zásadní pro zajištění, že rozhovory a analýzy pokryjí všechny klíčové aspekty datového modelu a jeho používání. Otázky zahrnovaly:

- Jakým způsobem aplikaci využíváte nejčastěji?
- Jaké problémy jste při používání aplikace identifikovali?
- Jaké máte nápady na zlepšení nebo optimalizaci datového modelu?

2. **Výběr respondentů:** Respondenti byli vybráni z řad výzkumných pracovníků *ÚDU AV* na základě jejich předchozích zkušeností s používáním *Inventaria* a *Documenta Rudolphina*. Důraz byl kladen na různorodost perspektiv zahrnující jak běžné uživatele, tak administrátora a vývojáře databáze.

3. **On-line rozhovory:** Rozhovory probíhaly v on-line prostředí platformy *Google Meet*, což vzhledem k předchozí komunikaci realizované v identickém duchu, umožnilo snadné zapojení respondentů. Byly použity návodné otázky pro zajištění otevřené a produktivní diskuse.

4. ***Use-case* scénáře:** K ověření vlastní analýzy a identifikaci specifických problémů byly vytvořeny a prozkoumány *use-case* scénáře. Tyto scénáře pomohly objasnit, jak uživatelé interagují s aplikací v konkrétních situacích a jaké výzvy se přitom objevují z pohledu datového modelu.

5. **Formulace zjištění:** Informace získané z rozhovorů a vlastního rozboru fungování aplikací byly pečlivě analyzovány. Cílem bylo identifikovat opakující se motivy, klíčové problémy a navrhnout vylepšení.

6.1.2 Klíčová zjištění

- **Prioritizace funkcionalit a rychlého přístupu k datům:** Rozhovory odhalily, že prioritou pro většinu uživatelů je intuitivní navigace a efektivní vyhledávání v databázi. Z pohledu datového modelu je pak žádoucí uchovat vysokou rychlost čtení, jelikož se jedná o nejčastější způsob manipulace s databází.
- **Technické výzvy:** Prezentace vlastní analýzy upozornila na specifické technické problémy, jako jsou obtíže s udržováním konzistence dat v *NoSQL* databázi *MongoDB*, respektive zastaralé provedení souborového systému u *Documenta*.
- **Návrhy na vylepšení:** Návrhy se často týkaly optimalizace datových modelů. Řešila se otázka transformace (normalizace/denormalizace) současného provedení struktury zdrojových souborů datového modelu *Inventaria*. V případě *Documenta* je pak nasnadě konverze původních zdrojových souborů do jiného formátu a změna způsobu uchovávání spočívající v zavedení vhodného databázového systému.

V kontextu optimalizace datového modelu *Inventaria* se narazilo na konkrétní problémy týkající se udržování konzistence dat, které by mohly navrhované změny řešit. Tyto obtíže plynou především z dynamické povahy a flexibility schémat dokumentů, což může komplikovat udržování konzistence při aktualizacích nebo změnách struktury dat.

Komplikace s konzistencí může nastat, když je například nutné aktualizovat informace o umělci, které jsou distribuované napříč více záznamy v databázi. V relační databázi by taková změna vyžadovala aktualizaci pouze v jednom místě. V databázi ***rudolf*** je každý záznam o díle navržen tak, aby obsahoval úplné informace o umělci, což znamená, že aktualizace se musí provést na mnoha místech současně, což zásadně zvyšuje riziko nekonzistence.

Další výzvou je správa složitých vztahů mezi entitami. Příkladem může být potřeba zachování konzistence mezi inventárními položkami a externími klasifikačními systémy, jako jsou *Iconclass* nebo *Getty ULAN*, týkající se umělců. Pokud dojde ke změně v klasifikačním systému, měla by se tato změna propagovat do všech relevantních inventárních záznamů, což je proces náročný na správu bez dostatečně navrženého systému referencí a aktualizací. Ve struktuře dokumentu inventární položky by se to týkalo například pole *search_subject*, jež může z *IconClass*

přejímat množinu klíčových slov. Obdobně pak pole *iconclass_external_id* obsahující konkrétní hodnotu *IconClass* číselníku.

Dále například pole *concordances* obsahuje identifikátory (hodnoty atributu *xml_id*) vztahující se k dalším položkám, které spolu jistým způsobem věcně souvisejí. Pokud dojde ke změně identifikátoru věcně souvisejícího díla, je vyžadována aktualizace všech záznamů, které na související dílo skrze původní identifikátor odkazují. Tato operace je náchylná k chybám a náročná na správu v původním denormalizovaném datovém modelu. Obtížně tak půjde zajistit, aby tyto změny byly adekvátně a konzistentně reflektovány ve všech relevantních záznamech. Obdobný problém pak může nastat například u polí *prevItem* a *nextItem*, v nichž se referuje totožným způsobem na předchozí, respektive následující, inventární položku ze seznamu.

6.1.3 Možná řešení problémů

Na základě vstupů od respondentů a vlastní analýzy byly identifikovány specifické oblasti pro zlepšení datového modelu, včetně:

- **Optimalizace datového modelu:** Přepracování struktury dat pro lepší efektivitu a flexibilitu, využívání referencí a vnořených dokumentů pro zjednodušení dotazů a aktualizací. U *Documenta* pak integrace zdrojových dat do databázového systému.
- **Normalizace redundantních dat a centralizace:** Snížení duplikace a redundance dat tím, že opakující se informace budou uloženy v centrální kolekci a v záznamech děl budou pouze referovány. Tím se zjednoduší aktualizace těchto informací.
- **Indexace pro rychlejší dotazy:** Vytvoření indexů na často vyhledávané atributy pro zlepšení rychlosti vyhledávání. Případně lze také rozšířit množinu vyhledávaných atributů.
- **Unifikace a denormalizace duplicitních atributů:** Sjednocení opakujících se atributů v rámci položky, zjednodušení struktury dokumentů pro snadnější správu a dotazování.

6.1.4 Diskuse

V rámci diskuse bylo prioritou pečlivě vyhodnotit potenciální dopad navrhovaných změn na funkcionalitu a výkon *Inventaria* a *Documenta Rudolphina*. Zdůrazněna byla nutnost najít rovnováhu mezi uživatelským komfortem a technickou proveditelností, aby bylo zaručeno, že navrhované úpravy přinesou výhody jak pro běžné uživatele, tak pro administrátory databázových systémů.

Důraz na metodiku dotazování ukázal její přínos jako nástroje pro odhalení potřeb a problémů, které vyžadují prioritní řešení během dalšího vývoje aplikace. Systematický a uživatelsky orientovaný výzkum poskytl základ pro efektivní aktualizaci a rozšíření aplikace.

Závěry z této diskuse zdůrazňují kritický význam průběžného získávání zpětné vazby a adaptace datového modelu, aby odrážel měnící se požadavky uživatelů a držel krok s technologickým pokrokem. Tento proces umožnil položit základy pro chystané optimalizace, které by měly navýšit efektivitu, strukturovanost a přístupnost datových modelů.

Nasazení dotazovací metodologie na případ *Inventaria* a *Documenta Rudolphina* vycházelo z celkového porozumění uživatelským potřebám a technickým výzvám. Tento přístup vedl k identifikaci a formulaci specifických zlepšení, která přispívají k efektivitě datového modelu.

Na základě provedených rozhovorů s výzkumnými pracovníky *ÚDU AV* a detailní analýzy používání aplikace byla vypracována *SWOT* analýza datového modelu projektu *Inventaria Rudolphina* a jeho okolí. Tato analýza umožnila komplexní pohled na silné, slabé stránky, příležitosti a hrozby týkající se aktuálního stavu a budoucího vývoje aplikace.

Identifikace klíčových sil, jako je efektivní organizace (v případě *Inventaria*), indexace a potenciál zdrojových souborů pro hloubkové analýzy vztahů, potvrzuje solidní základ datového modelu. Naopak rozpoznání slabostí, včetně výzev týkajících se komplexnosti správy a možností redundance dat, indikuje oblasti vyžadující pozornost a vylepšení. Analýza také odhalila významné příležitosti pro integraci pokročilých technologií a nových přístupů, které by mohly rozšířit funkčnost a interaktivitu databází. Současně byly identifikovány potenciální hrozby, včetně rizik spojených s rychlými změnami v technologických standardech a výzvami ohledně integrity a bezpečnosti dat. *SWOT* analýza tak poskytla cenný rámec pro pochopení současného stavu datového modelu a pro plánování strategií jeho budoucího vývoje.

Tabulka 6.1: *SWOT* analýza současného stavu projektu *Inventaria* a *Documenta*. Zdroj: Vlastní zpracování

Kategorie	POMOCNÉ	ŠKODLIVÉ
VNITŘNÍ	**Strengths (silné stránky)**: • Datový model poskytuje základní organizaci a indexaci uměleckých děl a umělců • Rychlé vyhledávání • Potenciál pro hluboké analýzy vztahů mezi díly a tvůrci	**Weaknesses (slabé stránky)**: • Model může být komplexní pro správu • Použití vnořených dokumentů a referencí • Ztížení aktualizací – konzistence • Redundance dat • Zastaralá organizace souborů (*Documenta*)
VNĚJŠÍ	**Opportunities (příležitosti)**: • Integrace nových technologií a přístupů (strojové učení pro doporučování a analýzu trendů mezi díly) • Rozšíření modelů o další multimediální obsah nebo interaktivní prvky	**Threats (hrozby)**: • Rychlé změny v technologiích a ve standardu uchování dat • Frekventované aktualizace modelu • Riziko ztráty integrity dat • Bezpečnost dat (při nevhodné správě nebo návrhu databáze) • Integrita a aktuálnost údajů z externích zdrojů (*Getty*/*Iconclass*)

Níže již následují rozbory zjištěných nedostatků a konkrétní návrhy možných vylepšení, nejdříve pro *Inventaria* a následně pro *Documenta*.

6.2 Inventaria

Tento oddíl se věnuje návrhu změn a možných vylepšení datového modelu používaného v aplikaci *Inventaria Rudolphina*. Analýza zdrojových souborů v podsekci 3.1.4 odhalila řadu klíčových prvků, které tvoří základ současného datového modelu, včetně identifikátorů, tematické klasifikace, detailních popisů děl, informací o vztazích a provenience. Přestože tento model umožňuje efektivní prezentaci dat, bylo i na základě komunikace s uživateli identifikováno několik nedostatků, které jej mohou limitovat. Optimalizace datového modelu má za cíl adresovat identifikované nedostatky a zvýšit jeho efektivitu, přehlednost a zlepšit možnosti správy dat například z hlediska konzistence.

V případě datového modelu bude zachována databáze *MongoDB*, a to kvůli několika klíčovým vlastnostem, které vyhovují specifickým potřebám projektu. *MongoDB*, jakožto *NoSQL* databáze, v tomto případě spravuje heterogenní datové soubory, jejichž struktura se může lišit u každé inventární položky. Při výběru databázové technologie hrály roli i podpora komplexního dotazování a škálovatelnost. Měly by však být brány v potaz i limitace *MongoDB*. Specifická omezení, ovlivňující *Inventaria* z pohledu výkonu a správy, jsou především udržování konzistence a integrity dat při aktualizacích, či změnách struktury. S tím se pojí i složitá správa vztahů mezi entitami. Kvůli dynamickému schématu a tendenci k *denormalizaci* dat napříč inventárními záznamy může být udržení konzistence náročnější. V kontextu *Inventaria*, kde převažují operace čtení nad zápisy, by normalizace dat v podobě referování usnadnila správu vztahů a udržování konzistence. Nicméně, tato strategie může zvýšit složitost dotazů a ovlivnit výkon při čtení, popřípadě vyhledávání.

Model *Inventaria* je postaven na komplexní a flexibilní struktuře, která podporuje reprezentaci uměleckých děl, umělců a souvisejících metadat. Návrh optimalizace datového modelu se opírá o následující prvky obsažené v analyzovaném zdrojovém souboru záznamu (konkrétně *JSONu*, viz Příloha C):

- **Identifikátory a klasifikace**: Unikátní identifikátory (*xml_id*), klasifikace podle témat (*subject*, *search_subject*) a externí *ID* systémů *Getty ULAN* a *Iconclass*.
- **Detailní popisy děl**: Textové popisy, názvy a komplexní informace o jednotlivých objektech, včetně typu, digitalizace a údajů o umělcích (včetně odkazů do externích databází).
- **Vztahy a konkordance**: Informace o vztazích mezi různými záznamy, včetně míry shody a osob odpovědných za stanovení vztahu.
- **Fyzické dimenze a provenience**: Detailní údaje o objektech (tzn. obrazech), včetně materiálu, rozměrů, datace a provenience.
- **Umístění, zasazení do prostoru a inventární čísla**: Informace o umístění děl, včetně inventárních čísel a lokalit (jsou-li dostupné). Zasazení do prostoru u děl z pražských inventářů.

Bylo identifikováno několik strategií, které je možné uplatnit pro vylepšení struktury zdrojových dat a optimalizaci datového modelu:

6.2.1 Redukce duplicit

Následující návrhy optimalizace by mohli pomoci zredukovat duplicity nacházející se v datech a tím zjednodušit strukturu a v návaznosti s tím i manipulaci s daty. Není-li uvedeno jinak, návrhy se týkají změny struktury dokumentů z původní kolekce ***inventory*** - viz obrázek 3.4.

6.2.1.1 Normalizace redundantních dat - centralizace informací o umělcích

Minimalizace duplikace informací mezi dokumenty může zlepšit účinnost ukládání a aktualizací. Pro každý obraz je žádoucí uchovávat pouze specifické informace, případně unikátní atributy relevantní pro to konkrétní dílo. Všechny ostatní opakující se informace by měly být uvedeny pouze jednou na úrovni celého dokumentu nebo odkazovány z centrálně uložených dat. Kupříkladu, místy rozsáhlé informace o autorovi jsou v kompletním rozsahu ukládány u každého dokumentu v poli *object*. Tyto informace mohou být vedeny v samostatné kolekci a referovány v dokumentech odpovídajících uměleckých děl (viz klíč *object.name*). Vzhledem k tomu, že informace o umělci (*name.getty_data*) se (s výjimkou položky *role*, která zůstane zachována) opakují, by se mohla uplatnit již existující vytvořená kolekce *ulan*, obsahující záznamy o umělcích ve stejném rozsahu jako vnořené dokumenty v *object.name*. V návaznosti na to by se v dokumentech obrazů referovali prostřednictvím unikátního identifikátoru příslušného dokumentu z *ulan*. Chybějící autory je možné volně[23] získat z databáze *Getty ULAN*.

Tímto způsobem dojde ke snížení redundance dat a zároveň se usnadní udržování konzistence. Změny týkající se autora bude potřeba zapsat jen do jednoho dokumentu.

6.2.1.2 Unifikace a denormalizace polí klíčových slov

Duplicity jsou dále evidentní v polích *subject* a *search_subject*, která mají totožný význam - poskytují množinu klíčových slov vystihující danou položku pro účely vyhledávání. Pole *subject* obsahuje manuálně stanovená klíčová slova, zatímco *search_subject* obsahuje položky předešlého pole v doprovodu množiny extrahované z dokumentu v kolekci *iconclass* dohledatelného skrze hodnotu *Iconclass* číselníku, umístěného v atributu *iconclass_external_id*. Prvky v obou polích se mohou vzájemně překrývat. Pole by mohla být sjednocena do jednoho, bez duplicit. To by zjednodušilo strukturu záznamu, snížilo redundanci dat a zlepšilo srozumitelnost. Avšak zvolením tohoto přístupu by došlo ke ztrátě informace týkající se původu klíčového slova. Metadata týkající se původu klíčových slov by bylo možné zachovat zavedením anotací. Přístup by spočíval v tom, že by se klíčová slova neukládala pouze jako pole textů, ale jako pole objektů, v němž by každý objekt obsahoval jak samotné slovo, tak informaci o jeho původu. Například:

```
kws_all: [
    {kw: "flower", origin: ["manual"]},
    {kw: "fruit", origin: ["iconclass"]},
    {kw: "market", origin: ["iconclass", "manual"]}
]
```

Zdrojový kód 6.1: Ukázka anotovaných klíčových slov

[23] https://www.getty.edu/news/getty-union-list-of-artist-names-ulan-linked-open-data/

Výběr mezi oběma přístupy by se měl odvíjet dle konkrétních potřeb projektu.

Pole *subject* je žádoucí zachovat, přičemž množinu klíčových slov pocházející z *Iconclass* je možné načítat z již uvedené kolekce *iconclass*, jmenovitě ze zanořeného pole *kws_all.en*. Změní-li se záznam v *iconclass*, bude nutné pole *search_subject* manuálně aktualizovat.

U relačních databází by se v takovéto situaci nabízelo použití *triggerů* (spouštěčů) a definovat činnosti, které se mají provést v případě definované události nad databázovou tabulkou [WC95] - zde by se jednalo o aktualizaci záznamu. Nicméně, *NoSQL* databáze *MongoDB triggery* nativně nepodporuje. Nabízí však nástroj zvaný *Change Streams*[24], který umožňuje aplikacím reagovat na změny v databázi v reálném čase. Prostřednictvím implementace by bylo možné monitorovat změny v kolekci *iconclass* a následně aktualizovat dokumenty v kolekci *inventory* na základě těchto změn. Je důležité poznamenat, že *Change Streams* vyžadují replikovanou sadu, případně *sharded cluster* v *MongoDB*. Databáze je musí podporovat, což se týká *MongoDB* verze 3.6 a novějších. Při velkém objemu dat, případně častých aktualizacích se doporučuje testovat dopady na výkon a zvážit dodatečné optimalizace. [DD16]

Rozšíření procesu unifikace o toto řešení umožňuje dynamické aktualizace mezi kolekcemi s minimálními požadavky na manuální intervenci, což zjednodušuje správu dat a udržuje je synchronizovaná.

6.2.1.3 Změna reprezentace konkordancí, inventáře, *Iconclass* a položek v pořadí seznamu

Pole *concordances* obsahuje reference (*id*) na další věcně související záznamy a míru jistoty (*cert*). Na základě rozboru struktury databáze (v oddílu 3.1.1.2), bylo zjištěno, že tyto vztahy již jsou uchovávány v samostatné kolekci - *concordances.* Záznamy v této kolekci jsou navíc doplněny o identifikátor zdrojového díla (*src_xml_id*) a o informace o osobách, jež nesou za stanovení věcné souvislosti odpovědnost.

Nabízí se zjednodušit strukturu pole. Mohla být aplikována normalizace tím způsobem, že by se místo celého záznamu z *concordances* uchovával pouze identifikátor daného záznamu. Z odpovídajícího dokumentu v kolekci *concordances* by se referovaly hodnoty klíčů *dst_xml_id* a *cert* odpovídající věcně souvisejícímu záznamu, respektive míře stanovené jistoty konkordance. Obdobně by se dalo postupovat i v případě klíče *inventory*, který v sobě nese zanořený dokument obsahující název (*name*), popisek (*label*) a pořadí inventáře v *Inventaria* (*order*). I zde se nabízí vyhotovení samostatné kolekce (*inventory_info*), která by čítala třináct záznamů, jeden pro každý inventář. V rámci položky by se pro odkazování na dokument inventáře uplatnily reference. Ve výsledku by tato změna ulehčila správu metadat týkajících se inventářů. Dodatečné přidání reverzních odkazů do dokumentů v *inventory_info* reprezentující inventáře (např. do pole *item_ids*) usnadní propojení s jednotlivými položkami z kolekce *inventory* a zjednoduší formulaci vyhledávacích *query*, které jsou na *frontendu* poměrně běžné - vypisování obsahu inventáře. Tato změna by mohla vést k rychlejšímu přístupu k datům a redukci složitosti dotazů.

V rámci struktury jsou u některých položek v kompletním rozsahu uloženy informace týkající se *Iconclass* (viz *iconclass_data* v dokumentech z kolekce *inventory*). Týká se to těch

[24] https://www.mongodb.com/docs/manual/changeStreams/

položek, kterým byl přiřazen jednoznačný *Iconclass* číselník/identifikátor. Při rozboru dílčích kolekcí v oddílu 3.1.1.2 však bylo zjištěno, že pro metadata týkající se *Iconclass* je již vyhrazena samostatná dokumentová kolekce nesoucí týž název. Opět by se aplikovala stejná strategie jako v předchozích odstavcích, tedy normalizace a referování. Bylo by možné odkazovat se přes *_id* dokumentů z kolekce *iconclass*, přičemž tyto dokumenty jsou náplňově identické se vnořenými dokumenty ve struktuře položky (dokumentu v *inventory*). Přínos by mohl spočívat ve zpřehlednění struktury inventárních položek, případně zjednodušením správy a úprav informací získaných z *Iconclass*.

Normalizace již byla tímto způsobem aplikována u klíčů *room* a *place*. V těchto dvou případech se v dokumentech kromě číselného referenčního *id* vyskytuje klíč *original_description*. Tento klíč je v databázi zachycen pouze v rámci struktury inventární položky a nikoliv v kolekci *rooms*, kam je skrze *id* odkazováno. V této kolekci, kde společně s prostorovými souřadnicemi místnosti, popisku a namapování do *SVG* plánku je přítomné i pole *places*. To zahrnuje výčet všech umístění, evidovaných v rámci dané místnosti, což lze vzhledem k charakteru záznamu považovat za smysluplnou strukturu a není třeba záznam dále normalizovat.

V neposlední řadě je možné navrhnout alternativní odkazování mezi položkami seznamu, tj. reference na předcházející (*prevItem*), respektive následující (*nextItem*) položku v inventárním seznamu. Namísto řetězcových hodnot odpovídající atributu *xml_id* jiné položky, by bylo možné přímo uplatnit unikátní identifikátor dokumentu (*_id*) z kolekce *inventory* a tím usnadnit provázání položek a manipulaci se seznamem děl.

6.2.1.4 Normalizace a referování obrazů - úpravy pole *object*

Změna spočívá v ukládání dílčích rastrových obrazů a jejich popisu umístěných v poli *object* do samostatné kolekce. Platí, že pokud byl inventárnímu záznamu přiřazen konkrétní obraz, respektive obrazy, pole *object* obsahuje kromě výchozího generického záznamu rozebíraného v 3.1.4.2 i další. V případě vyššího počtu položek, *object* obsahuje opakující se informace o obrazech. Jedná se o popisy a charakteristiky (např. *relationship_type*; *iconclass_external_id*; *subject*), které by mohly být optimalizovány tak, aby se duplicitní informace neopakovaly, například skrze normalizaci buďto informací o obrazech, případně celých vnořených dokumentů reprezentujících obrazy, do samostatné kolekce s unikátními identifikátory pro každý obraz. Referování by bylo možné skrze unikátní *object_id*, který by byl každému obrazu přidělen, eventuelně zkonstruován z klíče *xml_id*, je-li u obrazu uveden. Není tomu však u všech obrazů.

V rámci dokumentu v poli *object* lze dále nahlédnout například na klíč *images*, tedy pole s názvem zdrojového rastrového souboru a doprovodným popiskem. Pro úplnost je třeba zmínit, že rastrové položky jsou uchovávány přímo na serveru, v instalačním adresáři[25] demoverze *Inventaria*. Namísto ukládání celého obrázkového záznamu přímo v dokumentu inventární položky by se mohly obrázky evidovat samostatně a referovat pomocí identifikátorů, což by vedlo nejen k efektivnějšímu ukládání, ale i přístupu k nim. V případě, že se referovaný zdrojový soubor vyskytuje ve více prvcích pole *object*, což může být dáno například tím, že je vyobrazen v rámci jiné položky, mohlo by tímto přístupem dojít k úspoře místa v úložišti, jelikož daný

[25] Adresář: *./opt/app/production/dist/static/images*

soubor bude v databázi uložen pouze jednou. Pokud by bylo nutné jej aktualizovat, změna by se prováděla pouze na centrálním místě. Situaci, při které je zobrazován identický ilustrativní obrázek, lze odhalit na bázi hodnoty konkordance, kde klíč *cert* u věcně související položky z jiného inventáře nabývá hodnoty `same_as`. To se týká například pražských inventářů *1621 Prague A* a *1621 Prague B*. Celkem bylo (ke dni publikace teze) na základě filtrace identifikováno 965 unikátních položkových párů, u nichž bylo stanoveno, že se jedná o identické obrazy.

```
_id: ObjectId('63cc6c4ac070863a95aae1a2')
src_xml_id : "PrgA-815"
dst_xml_id : "PrgB-815"
cert : "same_as"
resp : null
```

Obrázek 6.1: Záznam z kolekce *concordances* představující identický obraz. Zdroj: Vlastní zpracování

Zároveň by se však výše popisovaným přístupem snížila přehlednost - jednotlivé obrazy by bylo potřeba vždy namapovat k původním prvkům z pole *object*, u nichž samotných je plánovaná normalizace. V aktuální podobě je vše drženo na jednom místě ve formě vnořených dokumentů.

Informace, které jsou specifické pro každé dílo/obraz, jako jsou *cert*, *caption*, *object_name*, *institution*, případně data týkající se fyzických vlastností a provenience, mohou zůstat jako vnořené dokumenty v rámci jednotlivých objektů. To udržuje relevantní informace společně, zatímco by se zbytečné duplicity eliminovaly.

Tímto způsobem lze strukturu *object* ve struktuře inventárního záznamu optimalizovat pro lepší přehlednost, snazší správu a efektivnější využití dat.

6.2.2 Zlepšení přístupu k datům

V rámci zlepšení přístupu k datům a zvýšení efektivity práce s databází, je možné identifikovat dva zásadní aspekty, které by měly být předmětem optimalizace - *indexace* pro zrychlení dotazů a *validace* schématu pro zajištění integrity dat. Tyto návrhy jsou klíčové pro zlepšení celkového výkonu a spolehlivosti databázových operací.

Prvním krokem je implementace indexů pro vyhledávané atributy dokumentů z kolekce *inventory*, jako jsou *xml_id, title, subject, name.value* (jméno umělce) a *iconclass_external_id*. Indexace těchto atributů umožňuje databázi efektivněji lokalizovat a načítat požadované záznamy, čímž se výrazně snižuje doba reakce na dotazy. Tento proces je zvláště důležitý v prostředích s velkým objemem dat (zde se jedná o tisíce záznamů), kde může rozdíl v rychlosti vyhledávání mít významný dopad na celkovou uživatelskou zkušenost a výkon systému.

Druhým klíčovým aspektem je zavedení validace schématu dokumentů pro zajištění datové integrity. V *dev* verzi databáze *rudolf* bylo prakticky ověřeno[26], že nebyla stanovena žádná validační pravidla. Aplikace pravidel pro validaci schémat zajistí, že všechna data, u kterých se plánuje vložení do databáze (případně se jedná o již vložená data v databázi), splňují specifické požadavky na formát a strukturu. Tímto přístupem by se předcházelo potenciálním nekonzistencím a chybám v datech, které by mohly vést k nepřesnostem v analýze dat nebo i k dalším

[26] https://www.mongodb.com/docs/manual/core/schema-validation/view-existing-validation-rules/

problémům při zpracování. Validace schémat je jedním z pilířů pro udržení vysoké kvality a spolehlivosti datových zdrojů. Vzhledem k flexibilním schématům, se kterými *MongoDB* pracuje, a tedy rozdílné míře dostupnosti dílčích informací u jednotlivých inventárních položek, se nabízí stanovit volitelná pole, případně definovat několik různých validačních schémat. [Mon23d]

6.2.3 Evaluace návrhů vylepšení

Na základě provedené analýzy a identifikace klíčových oblastí pro zlepšení je možné konstatovat, že navrhované změny v datovém modelu přinášejí řadu významných výhod. Optimalizace struktury dat, redukce duplicit, normalizace redundantních informací a efektivnější přístup k datům nejen, že zlepšují efektivitu a přehlednost ukládaných informací, ale také usnadňují správu, aktualizace a rozšiřování databáze pro budoucí potřeby. Tato doporučení vycházejí přímo z analýzy současného stavu zdrojových dat.

Implementace navrhovaných řešení bude vyžadovat pečlivé plánování a koordinaci mezi vývojáři, databázovým administrátorem a uživateli. V kontextu zjištěných návrhů na zlepšení datového modelu lze očekávat snížení redundance dat a zlepšení celkové správy a udržitelnosti databáze.

6.2.4 Nastínění změn v ukázkové struktuře inventární položky

Následuje začlenění a vizualizace navrhovaných změn do ukázkové struktury inventární položky *PrgA-815*[27]. V případě, že dochází k normalizaci původních informací do nových kolekcí, je znázorněna ukázka struktury dokumentu z příslušné kolekce.

6.2.4.1 Upravená struktura inventární položky

Ukázka struktury dokumentu z kolekce *inventory* (viz odpovídající záznam v Příloze H) reprezentuje výše uvedenou inventární položku *PrgA-815* po zavedení navrhovaných změn. Pole *subject* a *search_subject* byla sloučena, následně došlo k normalizaci věcně souvisejících vazeb, vyčlenění *generického* prvku z pole *object* do samostatného vnořeného dokumentu a v neposlední řadě i k normalizaci zbývajících prvků a dokumentu reprezentující zdrojový inventář.

6.2.4.2 Struktura dokumentu referované konkordance

Je uplatněn původní dokument z existující kolekce *concordances*. Strukturu nebylo třeba upravovat, dokumenty disponují unikátním identifikátorem.

```
{
  "_id": {
    "$oid": "63cc6c4ac070863a95aae1a2"
  },
  "src_xml_id": "PrgA-815",
  "dst_xml_id": "PrgB-815",
  "cert": "same_as",
  "resp": null
```

[27] http://147.228.173.159/item/PrgA-815

```
}
```

Zdrojový kód 6.2: Struktura dokumentu konkordancí

6.2.4.3 Struktura dokumentu referovaných umělců

Používá se dokument z existující kolekce *ulan*, struktura nebyla upravena. Informační rozsah se shoduje s původní podobou vnořeného dokumentu. V inventární položce byly zachovány pouze dvojice původních klíčů *value* a *role*. Ukázková struktura je zachycena v rámci Přílohy H.

6.2.4.4 Struktura dokumentu původního prvku z pole *object*

Pro jednotlivé obrazy je v případě normalizace již potřeba vytvořit zcela novou kolekci (*separate_objects*). Normalizace se netýká *generického* prvku z pole *object*, který se ve srovnání s následujícími (jsou-li přítomny) strukturálně odlišuje. Obdobně jako u dokumentů z *inventory* i zde dochází k referování informací týkajících se autora v poli *name*. Každému dokumentu by bylo přiřazeno unikátní *object_id*. V rámci struktury (viz odpovídající oddíl Přílohy H) by rovněž bylo možné normalizovat dokumenty v poli *images*. Ukázka normalizace je uvedena za původním nezměněném znění. Normalizace obrazu v rastrové podobě je demonstrována níže v oddílu 6.2.4.6.

6.2.4.5 Struktura dokumentu referovaného inventáře

Normalizace vnořených dokumentů týkajících se informací o zdrojových inventářích by se uskutečnila prostřednictvím nově vytvořené kolekce (*inventory_info*) a pro účely referování by bylo možné aplikovat unikátní hodnoty klíče *_id*. Reverzní referování do kolekce *inventory* by se uskutečňovalo skrze pole *items_ids* obsahující identifikátory děl nacházejících se v inventáři.

```
[
 {
  "_id": ObjectId('inventory_1621_Prague_A_id'),
  "name": "1621 Prague A",
  "label": "1621 Prague A",
  "order": 6
  "item_ids": [ ... 821 Items ... ]
 }
]
```

Zdrojový kód 6.3: Struktura dokumentu referovaného inventáře

6.2.4.6 Struktura dokumentu referovaného grafického souboru

Řešeno obdobným způsobem jako například výše uváděné inventáře. Původně vnořené dokumenty skládající se ze zdrojového grafického souboru (tj. obrazu) a popisku, by se přesunuly do samostatně figurující kolekce (např. *images*). Odkazování by se uskutečňovalo v dokumentech z kolekce *separate_objects*. Pro tyto účely byl taktéž navrhnut unikátní identifikátor *_id*.

```
[
  {
    "_id": ObjectId('image_id_1'),
    "file": "alt-1-PrgA-B-815.jpg",
    "text": "Giuseppe Cesari (Cavaliere D'Arpino), Susanna at the
   Elders, ca. 1607, Pinacoteca Nazionale, Siena"
  },
  (Dalsi zdrojove soubory s rastry + popisky ...)
```

Zdrojový kód 6.4: Struktura dokumentu referovaného grafického souboru

6.3 Implementace změn - proces *ETL*

Pro implementaci výše stanovených strukturálních změn lze uplatnit variantu procesu *ETL* neboli *Extrakce*, *Transformace* a *Load*. [VS09] Ve své podstatě se jedná o fundamentálním krok v oblasti datové integrace, který umožňuje efektivní manipulaci s daty z různých zdrojů a jejich přípravu pro analýzu nebo ukládání v cílovém systému. Tento proces je klíčový pro zpracování velkých objemů dat, vyžadující jejich čištění, normalizaci a konsolidaci do uceleného formátu. [Qli24] V souvislosti s *MongoDB* a *JSON* dokumenty, tyto procesy čelí specifickým výzvám týkající se nestrukturované povahy dokumentů a potřeby jejich efektivní normalizace, agregace a optimalizace pro dotazování.

Následuje stručný přehled jednotlivých fází *ETL* procesu:

Extrakce - První fáze, extrakce, zahrnuje shromažďování dat z mnoha různých zdrojů, které mohou být strukturované nebo nestrukturované. Data mohou pocházet z databází, souborů, cloudových úložišť, *API* rozhraní, a dalších. Cílem je extrahovat relevantní data, která jsou následně připravena pro další zpracování. V našem případě by se jednalo o pouhý export původních dokumentů z *dev* databáze *rudolf*. Celkově proces extrakce představuje základní stavební kámen celého *ETL* cyklu. Jeho pečlivá implementace a provedení jsou zásadní pro úspěch následujících fází. [IBM23]

Transformace - Během transformace se extrahovaná data upravují a konvertují do formátu, který je vhodný pro analýzu nebo další použití. Tento krok může zahrnovat čištění dat od nepřesností a duplicit, normalizaci datových formátů, odstranění nebo šifrování citlivých informací, agregaci a mnoho dalších transformačních úkonů. Transformace je často nejnáročnější částí celého *ETL* procesu, jelikož vyžaduje důkladné pochopení jak zdrojových dat, tak požadavků cílového systému. [Keb22]

Nahrání - Poslední fáze, nahrání, představuje přesun transformovaných dat do cílového systému, jako je například datový sklad, databáze nebo jiná platforma pro ukládání dat. V této fázi je důležité zajistit, aby data byla správně namapována na cílovou strukturu a aby byly zachovány všechny relevantní vztahy mezi daty. Takto upravené dokumenty by se importovaly do předpřipravené *MongoDB* databáze složené z kolekcí definovaných v předchozí specifikaci změn. [IBM23]

ETL procesy mohou být implementovány ručně, avšak s rostoucím objemem a složitostí dat se stále více uplatňují specializované nástroje a služby, které nabízejí automatizaci, vizuali-

zaci procesů a pokročilé možnosti pro správu a monitorování datových toků. K populárním nástrojům patří jak *open-source* řešení (např. *Integrate.io*[28]), tak i komerční platformy (např. *IBM DataStage*[29], *AWS Glue*[30], *Azure Data Factory*[31]), které umožňují organizacím maximalizovat přidanou hodnotu získanou z dat a tím podpořit rozhodování a obchodní strategie. [Dat23]

Klíčovým aspektem úspěšné implementace částí *ETL* je pochopení specifických potřeb a výzev spjatých s konkrétními daty a cílovým systémem. To zahrnuje volbu vhodných metod a nástrojů pro extrakci, detailní naplánování transformační logiky a zajištění robustního a škálovatelného řešení pro nahrání dat. V kontextu manipulace s *JSON* dokumenty a databází *MongoDB* je důležité zvážit specifika těchto technologií, včetně struktury dat a možností pro jejich exportování, transformaci a ukládání v nových kolekcích. Jednalo by se tak o poměrně zjednodušený proces *ETL* s důrazem kladeným na transformaci původních dat, která se již nachází ve strukturované podobě.

Vzhledem k dynamické povaze datových ekosystémů je také důležité uvažovat o možnostech rozšíření *ETL* procesu, aby byly zachyceny všechny relevantní změny ve zdrojových souborech. Ačkoli se jedná o proces, který bude spuštěn vícekrát, zejména při upravování implementace pro zobrazení dat na *frontendu* a při přepracování původních dat do nové struktury, je cílem ukázat, že je možné efektivně upravit strukturu zdrojových souborů a ukládat je identickým způsobem. Tímto se navrhované řešení odlišuje od pravidelného nebo *realtime ETL* procesu, který by sledoval a importoval změny z externích zdrojů, typu *IconClass* či *Getty ULAN*.

V praxi vyžaduje implementace, ačkoliv primitivního *ETL* procesu pro dokumenty z *MongoDB*, detailní plánování a přípravu. Začalo se s analýzou struktury zdrojových dat a identifikací žádoucích transformací pro dosažení požadované struktury. Extrakci zdrojových dat by bylo možné uskutečnit aplikováním dedikovaných nástrojů a funkcionalit, přičemž transformační fáze může zahrnovat, mimo jiné, samostatné funkce pro normalizaci datových bloků a rozdělení komplexních dokumentů na jednodušší entity pro lepší správu. Import nově strukturovaných dokumentů by se opět uskutečnil skrze existující nástroje poskytované přímo *MongoDB*. Strategie pro nahrání transformovaných dat zahrnuje jak úpravu stávajících kolekcí, tak i vytváření nových, případně i definování nových indexů pro optimalizaci výkonu. Dílčí kroky jsou detailně rozebrány v následujících oddílech.

6.3.1 Aplikace *ETL* na dokumenty ze zdrojové databáze *rudolf*

Při rozhodování o implementaci *ETL* procesu pro extrakci dokumentů z původní databáze *rudolf* (obsahující řádově tisíce dokumentů), následnou transformaci struktury dokumentů a nahrání do jiné databázové instance *MongoDB*, byl nakonec zvolen vývoj vlastního skriptu v jazyce *Python*[32], namísto uplatnění specializovaných *ETL* nástrojů. Tento záměr vychází z

[28] https://www.integrate.io/
[29] https://www.ibm.com/products/datastage
[30] https://aws.amazon.com/glue/
[31] https://azure.microsoft.com/en-us/products/data-factory
[32] https://www.python.org/about/

několika konkrétních faktorů, které byly uváženy vůči potenciálním nevýhodám a problémům týkajících se tohoto přístupu.

Zásadní okolností pro toto rozhodnutí je jednorázová povaha požadovaného *ETL* procesu. Vzhledem k tomu, že není potřeba automatizované opakování procesu, nabízí vývoj specifického skriptu v *Pythonu* významnou flexibilitu a efektivitu [Kha24] bez zbytečných složitostí spojených s konfigurací a provozováním komplexních *ETL* řešení, jako jsou například *Mage.ai*[33], *Apache NiFi*[34] či *Pentaho Data Integration*[35].

Python, s jeho ekosystémem knihoven a zejména podporou knihovny *pymongo*[36], poskytuje dostačující nástroje pro práci s *MongoDB*. *Pymongo* umožňuje přímou manipulaci s dokumenty, jejich transformaci a zároveň naskýtá možnosti pro snadný import a export z, respektive do databáze. Tato integrace přímo vyhovuje potřebám projektu, umožňuje rychlou a přesnou práci se zdrojovými daty a minimalizuje potenciální rizika spojená s kompatibilitou a implementací. [Ohi11]

Přestože se vývoj vlastního řešení může potýkat s výzvami, jako jsou škálovatelnost, udržitelnost, integrace s jinými systémy, bezpečnost a spolehlivost, v kontextu jednorázového použití tyto potenciální nevýhody nepřevažují výhody. Manipuluje se s relativně malým datovým objemem (tisíce dokumentů, desítky *MB*) a zároveň není třeba definovat, plánovat, ani monitorovat složité průběžné *pipeline* (sady instrukcí). Bezpečnost a spolehlivost mohou být efektivně řešeny důkladným testováním a revizemi kódu.

Ve znamení těchto úvah je vývoj vlastního skriptu, kompletně pokrývajícího celý zjednodušený *ETL* proces, považován za vhodné řešení. Poskytuje nezbytnou míru přizpůsobivosti, umožňuje přímou a efektivní práci s daty a zároveň minimalizuje zbytečnou komplexitu, *overhead* a náklady spojené s používáním dedikovaných *ETL* nástrojů. Následující odstavce jsou věnovány rozboru zdrojových modulů formujících transformační skript, který byl vyvinut s ohledem na specifické potřeby *Inventaria*.

Transformační skript ***rudolfTransformation*** se skládá ze šesti zdrojových modulů v jazyce *Python*: *config.py, data_transformation.py, db_utils.py, main.py, transformation_functions.py a validation_schemes.py.* Cesta k umístění modulů je uvedena v rámci Přílohy G. Veškeré potřebné prerekvizity pro spuštění transformačního skriptu jsou uvedeny v projektovém *README.md* souboru.

Stručný popis náplně funkcionality modulů:

- ***main.py***: Hlavní spouštěcí konfigurace pro transformaci a migraci dat mezi *MongoDB* databázemi. Možnost opakovaného provádění *ETL* procesu.
- ***config.py***: Konfigurační soubor obsahující nastavení pro připojení k databázím a *SSH*, dále obsahuje konstantní hodnoty používané napříč zdrojovými soubory.
- ***db_utils.py***: Poskytuje pomocné funkce pro manipulaci s databázemi, včetně kopírování kolekcí a vytváření indexů. Obsahuje i funkce pro validaci schématu dokumentů v kolekcích.

[33] https://docs.mage.ai/introduction/overview
[34] https://nifi.apache.org/documentation/
[35] https://www.hitachivantara.com/en-us/products/pentaho-plus-platform/data-integration-analytics.html
[36] https://pymongo.readthedocs.io/en/stable/

- ***data_transformation.py***: Modul odpovídající za dílčí fáze *ETL* od extrakce a uložení, až po transformaci dat v cílové databázi pro jejich optimalizaci a normalizaci.
- ***transformation_functions.py***: Sada funkcí pro specifické transformace dat, jako je unifikace klíčových slov, transformace informací o umělcích, normalizace zdrojů obrázků, či tvorba reverzních referencí z inventářů na jejich položky. Zahrnuta je i funkce pro nahrání (aktualizaci) transformovaných dokumentů z kolekce *inventory*.
- ***validation_schemes.py***: Modul poskytující validační schémata pro dokumenty z kolekcí *images*, *inventory*, *inventory_info* a *separate_object*.

Pro snazší pochopení funkcionality kódu je popis rozdělen do tří segmentů reprezentující jednotlivé fáze *ETL*: extrakce (export) z databáze, transformace a nahrání do cílové databáze. Každá část je zkoumána s ohledem na implementační detaily a techniky. Zdrojový kód, uživatelská příručka, respektive dokumentace, jsou umístěny v přílohách, viz Příloha G.

V návaznosti na předchozí popis modulů, kde jsou rozděleny klíčové segmenty procesu do fází *ETL*, je důležité upřesnit, že přesnější charakterizací by bylo označení procesu jako ***ELT* (extrakce, nahrání, transformace)**. Toto upřesnění vychází z praktické realizace dílčích fází, kde transformace dat probíhá až po jejich nahrání do cílové databáze, což je odlišné od tradičního postupu *ETL*, v němž transformace předchází nahrání. Tento přístup umožňuje větší flexibilitu a efektivitu při zpracování dat, neboť manipulace s daty probíhá již přímo v prostředí, kde jsou data nakonec uložena. To může výrazně zjednodušit proces transformace a zvýšit celkovou výkonnost systému tím, že se eliminují některé překážky týkající se přenosu velkého objemu dat mezi různými systémy. Takový přístup reflektuje moderní trend [Nae+19] v oblasti datového inženýrství, kde se kladou požadavky na rychlost a škálovatelnost procesů zpracování dat.

6.3.1.1 Extrakce a nahrání dat

Proces je zahájen extrakcí dat ze zdrojové databáze ***rudolf***, a to prostřednictvím posloupnosti operací implementovaných, respektive volaných, v hlavním spouštěcím modulu *main.py*. Tento krok je zásadní pro zajištění, že veškerá data potřebná pro pozdější transformace jsou úspěšně přenesena do nového prostředí.

Nejprve je vytvořen bezpečný *SSH* tunel, který slouží jako zabezpečený most mezi lokálním pracovním prostředím a vzdálenou databází ***rudolf***. Uplatněna je externí knihovna *SSHTunnelForwarder*. Tento krok je zásadní pro ochranu dat během jejich přenosu, neboť veškerá komunikace probíhá šifrovaně, čímž se minimalizuje riziko neoprávněného přístupu nebo úniku informací. Uživatel je dotázán k zadání přístupových údajů, v případě hesla je pro uživatelský vstup použita knihovna *getpass* zajišťující, že při zadávání hesla nebude zobrazeno na obrazovce, což pomáhá chránit tento citlivý údaj před nahlížením třetích stran. V praxi by tyto přístupové údaje obvykle byly uloženy v konfiguračním textovém souboru a pro účely autentizace by se uplatnil speciální účet, případně vygenerovaný přihlašovací klíč, aby se nadále zvýšilo zabezpečení systému.

Po úspěšném navázání *SSH* tunelu následuje připojení k oběma databázím - k původní ***rudolf*** a nové, lokálně hostované databázi ***test_rudolf***, která bude sloužit jako úložiště pro transformovaná data. Takto je umožněn přímý přístup k datům a efektivní manipulace s nimi.

Hlavní částí procesu extrakce je kopírování vybraných kolekcí z původní databáze ***rudolf*** do nové databáze zavoláním funkce *copy_collections()* z modulu *db_utils.py* (k volání této i následujících funkcí uváděných v tomto odstavci dochází uvnitř těla funkce *etl_data()* definované v modulu *data_transformation.py*). Tato operace zajišťuje, že všechna relevantní data jsou dostupná pro následné fáze *ELT* procesu. Kromě kopírování standardních kolekcí je rovněž volána funkce *fill_inventory_info_collection()* pro vytvoření a doplnění kolekce *inventory_info*, pokud v cílové databázi chybí. Voláním této funkce je garantováno, že systém má k dispozici všechna potřebná metadata pro správnou identifikaci inventářů a kategorizaci položek v nich.

Pomocí této fáze jsou z původní databáze ***rudolf*** získána data, zkopírována do ***test_rudolf*** a připravena pro další krok zpracování - transformaci.

6.3.1.2 Transformace

Fáze transformace v procesu *ELT* je klíčová pro úpravu a optimalizaci dat tak, aby co nejlépe vyhovovala stanoveným potřebám cílového systému. V tomto případě se transformace dat provádí na nově extrahovaných datech z původní *MongoDB* databáze ***rudolf***, která byla přenesena do nové databáze ***test_rudolf*** provozované na *localhostu*. Proces transformace je spuštěn z hlavního modulu *main.py*, který ve smyčce vyvolá funkci *etl_data()*. V jejím těle je umístěno volání funkce *transform_data()* definované v modulu *data_transformation.py*. K cyklickému volání funkce může dojít na základě situace, kdy transformované dokumenty neprošly úspěšně procesem validace (viz podsekce 6.3.2 níže) a *ELT* je tak vhodné celé zopakovat. Uživatel se může rozhodnout, zda nechá celou smyčku opětovně proběhnout, nebo zda ji vlastní interakcí zruší. Dílčí transformační metody pak pocházejí z modulu *transformation_functions.py*.

Proces transformace začíná inicializací a přípravou pracovního prostředí, v němž jsou specifikovány dílčí kolekce ve struktuře *collections*. S ní se bude v průběhu transformace pracovat. Následně se ve smyčce pro každý dokument z kolekce *inventory* volají transformační funkce.

První z nich je funkce *unify_keyword()*, u které z názvu plyne, že řeší unifikaci klíčových slov přítomných v dokumentu. Tato funkce sjednocuje klíčová slova zadaná manuálně (v poli *subject*) a ta, která jsou k jednotlivým dokumentům přidružena prostřednictvím externího zdroje *Iconclass* (pole *search_subject*), čímž vytváří komplexní a významově bohatou sadu klíčových slov, prostřednictvím kterých je možné dokument dohledat.

Následuje řada transformačních kroků, které zahrnují transformaci dat týkajících se *Iconclass* (*transform_iconclass()* - nahrazení obsahu pole *iconclass_data* identifikátorem korespondujícího záznamu z kolekce *iconclass*), dále transformování původního pole věcných souvislostí *concordances* skrze funkci *transform_concordances()*, aby obsahovalo pouze odkazy na záznamy v doprovodné kolekci *concordances*. Na to navazuje proces normalizace informací o umělcích funkcí *transform_artists_info()*, která nově do dokumentu umístí referenci (*ulan_reference*) na záznam v kolekci *ulan*. Tyto funkce zabezpečují, že všechny relevantní informace jsou korektně aktualizovány, normalizovány a reflektují externí reference.

Dalším významným krokem je normalizace rastrových souborů obrazů a poskytování odkazů na ně v rámci inventárních položek. Použije se funkce *normalize_image_sources()*, která centralizuje správu obrazových zdrojů do jedné kolekce a zjednodušuje správu. Tento proces zahrnuje ověření na bázi shody názvů, zda se nově přidávaný obraz v kolekci již nenachází. Pokud ano, není již ukládán. Tímto způsobem je zamezeno vkládání duplicit.

Na základě předchozího vytvoření a vložení dokumentů do kolekce *inventory_info* je možné namísto udržování kompletního informačního rozsahu o zdrojovém inventáři, v němž se inventární položka nachází, uplatnit referenci na odpovídající dokument. Tuto dílčí normalizaci položek z *inventory* zaštiťuje funkce *normalize_inventory_info()*.

Ve fázi transformace je také zásadní volání funkce *split_objects()*, která normalizuje dokumenty obsahující více objektů v poli *object* na samostatné dokumenty v kolekci *separate_objects*. Tyto nově vzniklé dokumenty jsou v původních dokumentech referovány v poli *object_ids*. U každého původního dokumentu je také nově vytvořeno pole *generic_object* zahrnující generický objekt z původního pole objektů. Původní pole *object* je po zdárně provedené normalizaci ze struktury odebráno.

Funkce *transform_references()* je zásadní pro aktualizaci referencí uvnitř dokumentu na předcházející, respektive následující položku v inventárním seznamu. Zaměřuje se na převod identifikátorů položek z podoby řetězce nabývající hodnoty atributu *xml_id* do formátu unikátního identifikátoru dokumentu (*_id*). Tímto krokem se zvyšuje integrita a navigace mezi dokumenty se stává efektivnější.

Po průchodu a aktualizaci všech položek z kolekce *inventory*, je provedena finální transformační operace, a to propojení inventárních položek a záznamů o inventářích z kolekce *inventory_info*, což je realizováno funkcí *link_inventory_items_to_info()*. Funkce propojuje dokumenty na základě unikátního identifikátoru inventáře. Pro každý dokument v *inventory_info* vyhledá odpovídající položky v *inventory*, získá jejich *_id* a přidá je do dokumentu inventáře jakožto pole *item_ids*. Takto je zajištěno efektivní oboustranné referování mezi dokumenty těchto dvou kolekcí.

Fáze transformace převádí extrahovaná data do formátu, který nejlépe vyhovuje potřebám analýzy a dalšího využití. Díky pečlivě navrženým transformačním operacím jsou data optimalizována nejen pro výkon, ale i z hlediska sémantiky, což zvyšuje hodnotu informací dostupných pro koncové uživatele. Funkce *etl_data()* tak reprezentuje kompletní *ELT* proces, kde každý krok, od mazání starých kolekcí, přes kopírování dat, transformace až po vytváření indexů a validaci, je realizován sekvenčně.

6.3.1.3 Aktualizace transformovaných dat

Po provedení transformance jsou všechny dokumenty z kolekce *inventory* aktualizovány v cílové databázi ***test_rudolf***.

Funkce *update_document_in_collection()* hraje zásadní roli při této operaci. Je volána v rámci ústřední transformační funkce *transform_data()* z modulu *data_transformation.py*.

Funkce *update_document_in_collection()* slouží k aktualizaci již existujících dokumentů v kolekci *inventory*. Dokument je aktualizován v databázi tím, že se nastaví nové hodnoty (příkazem *$set*) a odeberou se nepotřebné atributy (specifikované v příkazu *$unset*).

Po aktualizaci všech relevantních dokumentů jsou v rámci databáze vytvořeny nové indexy prostřednictvím funkce *create_indexes()* z modulu *db_utils.py*. Indexy jsou definovány v konfiguračním souboru *config.py*. Po úspěšném propsání změn a vytvoření indexů navazuje již pouze validační etapa (sekce 6.3.2). Ukončení procesu *ELT* zahrnuje uzavření spojení s databázemi a *SSH* tunelu, čímž dojde k bezpečnému ukončení všech operací a uvolnění zdrojů.

6.3.1.4 Faktory ovlivňující rychlost *ETL* (*ELT*)

Doba vykonávání implementovaného *ELT* cyklu byla měřena a zaznamenána. V průběhu více testovacích běhů se tato doba pohybovala v rozmezí 30 až 40 vteřin. Testování bylo realizováno před zavedením procesu validace schémat dokumentů. Doba trvání procesu *ELT* závisí na několika klíčových faktorech, které ovlivňují jeho efektivitu a rychlost. Následuje shrnutí těchto faktorů společně s vysvětlením jejich vlivu:

1. **Velikost a komplexita dat:** Čas potřebný pro zpracování *ELT* procesů se značně liší v závislosti na velikosti a komplexnosti daných dat. Zatímco velký objem dat prodlužuje fáze extrakce a nahrání, složité aspekty (jako jsou vzájemné vztahy a normalizace) mají vliv na délku transformace. [MK17]

2. **Výkon zdrojového a cílového systému:** Výkon databázových systémů a serverů, z nichž se data extrahují a na které se data následně ukládají, je klíčový v určení délky trvání *ELT* procesu. Mezi nejdůležitější aspekty patří rychlost čtení a zápisu dat, výkon procesoru a dostupná kapacita operační paměti. [Pra08]

3. **Síťová propustnost a latence:** Kvalita a rychlost síťového připojení jsou zásadní pro *ELT* procesy, jež vyžadují přenos dat napříč různými systémy nebo umístěními. Tyto parametry mohou značně ovlivnit dobu potřebnou pro dokončení procesu, protože určují rychlost a spolehlivost datových přenosů. [Pra08]

4. **Složitost transformačních operací:** Pokud se v *ELT* procesu objevují složitější transformace, jako například provádění sofistikovaných agregací, komplexních výpočtů, nebo reorganizace a přejmenování objemných datových souborů, může to značně ovlivnit délku trvání celého procesu. Tyto operace vyžadují více času na zpracování, což vede k prodloužení doby, potřebné k dokončení *ELT* cyklu. [MK17]

5. **Úroveň paralelizace a automatizace:** Dodatečnou implementací paralelního zpracování a automatizačních technik by bylo možné významně snížit celkovou dobu potřebnou pro *ELT* proces. [Whi12]

6.3.2 Implementace procesu validace

V kontextu správy a zpracování dat se validace schémat vymezuje jako zásadní proces, zaměřující se na zajištění konzistence a splnění specifických kritérií či očekávání týkajících se dat před jejich dalším zpracováním nebo uložením. Tento proces je nezbytný pro odhalení a nápravu chyb v datech v co nejranější fázi, čímž se předchází možnému rozšíření nekonzistentních nebo neplatných dat do aplikací či databázových systémů.

Přesto je nutné si uvědomit dopady na výkon související s procesem validace, zejména v situacích, kdy se zpracovávají velké objemy dat. Efektivní strategie validace by měla harmonicky vyvažovat potřebu zachování kvality dat a operativní požadavky aplikace, aby nedocházelo k negativnímu ovlivnění celkového výkonu systému. V případě databáze ***test_rudolf*** má smysl implementovat validační schéma pro nově transformované dokumenty, potažmo kolekce. Tím je zajištěno, že nově transformovaná data splňují očekávané struktury a pravidla. Proces transformace zahrnuje složité manipulace s daty a kombinování dat z více kolekcí. Provedení validace po transformaci může zjednodušit proces identifikace a oprav chyb, jelikož lze aplikovat validační pravidla přímo na výsledek transformace, aniž by bylo nutné pro každý dokument validovat každou část transformace zvlášť. Dokumenty by již byly validovány v konečné podobě, tedy až po propsaní změn v databázi. Vycházíme z předpokladu, že zdrojová data vzhledem ke svému původu neobsahují nevalidní položky. Uvažujeme především hypotetickou situaci, kdy chyby mohou vzniknout na základě výpadku při aktualizaci souboru v databázi.

Implementace validace schémat za použití jazyka *Python* může pomoci identifikovat chyby nebo nekonzistence v datech před i po jejich uložením do databáze, případně před zpracováním dat v rámci transformačního skriptu. Pro tento účel lze využít knihovnu jako je *jsonschema*, která umožňuje definovat schémata a data validovat proti nim. Jazyk pro tvorbu definic schémat se nazývá obdobným způsobem - *JSON Schema*.

Nejdříve bylo nutné zmíněnou knihovnu nainstalovat a následně v rámci transformačního skriptu naimportovat. Následoval již samotný krok definice schémat a to pro dokumenty z kolekcí *image*, *inventory*, *inventory_info* a *seperate_object*. Například pro dokument z kolekce *images* schéma vypadá následujícím způsobem:

```
Images_schema = {
    "type": "object",
    "properties": {
    "_id": {"pattern": "^[a-f\\d]{24}$"},
        "file": {"type": "string"},
        "text": {"type": "string"}
    },
    "required": ["file"],
    "additionalProperties": True
}
```

Zdrojový kód 6.5: Definice validačního schématu pro dokumenty z kolekce *images*

Schéma reflektuje, že dokument má být objekt s vlastnostmi, které odpovídají jednotlivým polím. U každého pole je stanoveno, o jaký datový typ se jedná, případně zda má splňovat stanovený vzor daný regulárním výrazem a zda je výskyt pole povinný, či volitelný. Závěrem se uvádí, zda je povolena přítomnost dalších polí, která nejsou ve schématu explicitně definovaná.

V tento moment je možné zahrnout validační schémata do implementace, konkrétně po fázi transformace a indexace. Pro tento účel byly definovány funkce *validate_all-documents(db, validation_errors_filename)* a *validate_document(document, schema, collection_name, file)*.

Funkce *validate_all_documents()* zajišťuje hromadnou validaci dokumentů z vybraných kolekcí dle definovaných validačních schémat. Pro každou kolekci a přiřazené schéma tato funkce iteruje přes všechny dokumenty a používá funkci *validate_document()* pro jejich validaci a zápis

případných chyb do specifikovaného souboru *validation_errors.txt*. Funkce vrací pravdivostní hodnotu označující, zda během procesu validace byly nalezeny chyby.

Funkce *validate_document()* pak představuje jádro validačního procesu. Každý předaný dokument zkouší validovat proti danému schématu. V případě, že dojde k validaci a transformovaný dokument neodpovídá definovanému schématu, je vyvolána výjimka *ValidationError*, která je zachycena a informace o chybě jsou zaznamenány. Detaily chyby spolu s identifikací dokumentu a názvem kolekce jsou formátovány do *JSON* struktury a zapsány do předem specifikovaného souboru, což umožňuje pozdější analýzu a opravu dat. Přehled jednotlivých schémat se nachází ve zdrojovém souboru *validation_schemes.py*. Implementace validačních funkcí je možné najít ve zdrojovém souboru *db_utils.py*.

V současné konfiguraci procesu *ELT* v modulu *data_transformation.py* byla integrace validace realizována tak, aby se prováděla až po kompletním dokončení transformací a načtení změn do cílové databáze. Tento přístup reflektuje strategii, kdy se data nejdříve extrahují z původních zdrojů, poté načtou do cílové databáze, následně se provedou všechny požadované transformace a závěrem načtení změn do cílové databáze. Až po tomto načtení se provede validace transformovaných dat pomocí funkce *validate_all_documents()*.

Tento přístup umožňuje oddělit proces transformace dat od jejich validace, což přináší výhody v podobě vyšší efektivity a snadnějšího *debugování*. Vzhledem k tomu, že validace je prováděna až po načtení změn do databáze, mohou být veškeré transformace a aktualizace uskutečněny bez přerušení, což vede k rychlejšímu a hladšímu průběhu *ELT* procesu. Validace po dokončení transformací zároveň zvyšuje šanci na odhalení a identifikaci chyb v konečné podobě dat.

Pokud validace odhalí chybu, je možné podle potřeby rozhodnout o dalším postupu. Proces lze zastavit, poznamenat si chybu a případně data externě opravit. Tento přístup umožňuje, že transformační skripty budou robustnější a méně náchylné k problémům týkajících se nekonzistentních nebo nevalidních dat.

V praxi, zejména při používání *ELT* do cílového systému, však není vždy možné proces jednoduše zastavit a data opravit bez větších následků. V momentě, kdy jsou data již nahraná v cílovém systému, mohou být v nekonzistentním stavu, což komplikuje možnost jednoduchého *rollbacku*. V takovém případě je často nutné cílovou databázi před novým pokusem o nahrání dat kompletně vyčistit. Transformační skript ***rudolfTransformation*** uplatňuje právě tento přístup.

Na rozdíl od *ELT*, *ETL* procesy často zahrnují fázi *merge* (sloučení), což umožňuje inkrementální přidávání dat a nemusí dojít k úplnému vymazání již existujících dat v případě chyby. To přináší větší flexibilitu a efektivitu při manipulaci s daty. V návaznosti s tím se často pracuje s daty ve *stage* databázi, kde jsou data nejprve zpracována a teprve poté, co jsou verifikována a korektní, jsou přesunuta do produkčního prostředí. Tento postup efektivně redukuje riziko zanesení chybných dat do produkčního systému a umožňuje lepší kontrolu nad celým procesem. [Sou+19]

V případě zde rozebíraného *ELT* dojde po nalezení chyby k opakování celého procesu, pokud jej uživatel manuálně neukončí.

ale často se pracuje s daty ve stage databázi, kde jsou data nejprve zpracována a teprve

poté, co jsou verifikována a korektní, jsou přesunuta do produkčního prostředí. Tento postup efektivně redukuje riziko zanesení chybných dat do produkčního systému a umožňuje lepší kontrolu nad celým procesem. [Sou+19]

Zároveň však s sebou přináší jasnou výzvu - nalézt optimální rovnováhu mezi přesností validace a efektivitou zpracování. Takové rozhodnutí by mělo být učiněno s ohledem na specifické požadavky projektu a očekávanou frekvenci aktualizace dat.

Při integraci procesu validace do transformačního skriptu, bylo prakticky ověřeno, že doba vykonávání transformačního skriptu se prodloužila ve srovnání s transformací, která nezahrnovala validační etapu. Po zavedení hromadné validace transformovaných dokumentů se délka testovacího běhu z původních nižších desítek vteřin fakticky zdvojnásobila. Tento nález poukazuje na významný výkonnostní dopad, který s sebou validace schémat přináší. Je tedy žádoucí zvážit formu kompromisu mezi rychlostí transformace dat a důkladností validace.

Vzhledem k tomu, že transformační proces nebude (v reálném provozu) vyžadován, ani realizován frekventovaně, integrování validace dokumentů na úkor delší doby transformace se jeví jako smysluplné rozhodnutí. Tento přístup nabízí lepší garanci kvality a konzistence dat, což může být pro aplikace zpracovávající velké množství informací z dlouhodobé perspektivy až neocenitelné.

6.3.3 Shrnutí k *ETL* (*ELT*)

Navržený proces *ELT*, který byl následně aplikován na soubory z databáze ***rudolf***, představuje zásadní operaci, která je nezbytná pro optimalizaci a transformaci struktury dat. Vyvinutí specializovaného skriptu v *Pythonu* představuje řešení, poskytující nezbytnou flexibilitu a přesnost pro zpracování datových sad. Tento přístup byl upřednostněn oproti dedikovaným *ETL* nástrojům, především kvůli omezenému rozsahu transformační úlohy a snaze minimalizovat komplexitu a náklady spojené s využíváním komerčních řešení.

Využitím *Pythonu* a knihovny *pymongo* bylo dosaženo přímé manipulace s daty v *MongoDB*, což se ukázalo jako akceptovatelná varianta pro potřeby projektu. Přestože vývoj vlastního řešení přináší výzvy v oblastech škálovatelnosti, integrace, bezpečnosti a udržitelnosti, v tomto specifickém případě přínosy převažují nad těmito potenciálními nevýhodami.

Každý krok procesu byl promyšlen, aby zajistil efektivní zpracování dat při zachování jejich integrity a sémantické hodnoty.

Výše popisovaná implementace *ELT* nabízí potřebnou flexibilitu a minimalizuje nadbytečnou komplexitu a náklady. Došlo k normalizování struktury dokumentů z původní kolekce *inventory* a oddělení části vnořených dokumentů. Nově vzniklá kolekce *images* čítá 493 dokumentů, kolekce *inventory_info* 13 dokumentů a *separate_obejcts* 1 198 dokumentů. Objem dokumentů v databázi narostl o 1 704 položek. Ve srovnání s původním počtem se tak jedná o přírůstek ve výši přibližně 12 %.

Díky návrhu a realizaci *ELT* bylo možné dosáhnout požadovaných výsledků - optimalizovaných a transformovaných datových struktur, připravených pro další analýzu a využití, kterým může být například vizualizace probíraná v kapitole 7.

6.4 Documenta

V rámci modernizace a optimalizace *Documenta Rudolphina* by bylo žádoucí aplikovat sérii změn, které cílí na zlepšení přístupnosti, integraci s moderními technologiemi, standardizaci dat a zajištění jejich důvěryhodnosti a konzistence. Tyto kroky jsou nezbytné pro zvýšení použitelnosti a uplatnitelnosti historických přepisů pro širší spektrum uživatelů. Návrh a implementace změn vycházejí z předchozí týmové spolupráce a analýzy [Pav+23].

6.4.1 Modernizace aplikace

Celkem byly identifikovány dvě hlavní cesty pro modernizaci aplikace: zachování stávajících struktur s minimálními úpravami a vytvoření zcela nové systémové architektury. První přístup by zahrnoval přestylování současného rozhraní a integraci vyhledávání (*search*) přímo na webové stránce. Nicméně, tato možnost byla shledána nevhodnou, a to z důvodu pomalosti vyhledávání způsobené čtením z velkého množství *XML* souborů, jejichž počet se v době psaní odhadoval na přibližně *25 000*. Dalším důvodem bylo to, že současná struktura webu je značně zastaralá, nesrozumitelná a uživatelsky nepříliš přívětivá. Data uložená v samostatných *XML* souborech a následné stylování skrze *XSLT stylesheety* neodpovídají kvalitě moderních webů. Kompletní absence databáze zhoršuje přehlednost a v momentě přidávání nových stránek a obsahu je nezbytné vytvářet nové *XML* soubory.

Preferovaná alternativa se skládá ze zavedení třívrstvé architektury zahrnující *frontend*, *backend*, databázi a s ní související transformaci datového modelu, což umožní flexibilní rozvoj aplikace v závislosti na dostupných zdrojích. Tento přístup slibuje značné zlepšení výkonu, přehlednosti a uživatelské přívětivosti. V rámci teze jsou rozebírány primárně změny týkající se datového modelu.

6.4.1.1 Databáze

Při analýze vhodných databázových řešení byla zvažována možnost uchování dat v systémech *SQL* a *NoSQL*. Jako první varianta byla zkoumána *Oracle*[37] databáze s podporou *XML* souborů. Avšak z důvodu, že *XML* soubory nejsou v kontextu relačních databází standardně podporovány a implementace by byla náročná, nelze toto řešení doporučit. Navíc, *XML* soubory často obsahují kompletní *HTML* struktury, což komplikuje jejich zpracování.

Preferovaným řešením se stalo použití *NoSQL* databáze, konkrétně *MongoDB*. Tato platforma byla vybrána jako vhodné řešení pro ukládání semistrukturovaných dokumentů na základě podpory komplexních datových struktur a širokých možností, včetně základního *full-textového* vyhledávání a provádění složitějších dotazů. Převod *XML* dat do formátu *JSON* a jejich následná integrace do *MongoDB* představuje flexibilní a efektivní přístup k uchování a manipulaci s daty. Tento krok nejen, že zjednoduší správu dat, ale také umožní implementaci pokročilých dotazovacích funkcí.

[37] https://www.oracle.com/database/

Transformace *XML* do *JSON*. Z přehledu uvedeného v předchozí kapitole (viz 3.2.1) plyne rozhodnutí transformovat původní *XML* soubory do formátu *JSON* pro následné využití v *MongoDB*. Tento krok umožní přechod od statického, těžkopádného formátu ukládání a obsahu *XML* dokumentů k dynamickému a snadno uchopitelnému uspořádání dat. Pro účely projektu byly vytvořeny tři základní *JSON* kolekce - ***Person***, ***Document*** a ***ArchiveLinks***, které reprezentují osoby, díla a vztahy mezi archivy. Zdrojový kód transformačního skriptu napsaného v jazyce *Python* je pro všechny výše uvedené kolekce referován v Příloze F. Ukázky struktury vstupních a výstupních dat lze získat na základě informací uvedených v oddílu **elektronických příloh**.

Kolekce *Person* obsahuje klíčové informace o osobách, jako jsou jména, *aliasy*, data a místa narození či úmrtí a další relevantní údaje. Data a místa narození, či úmrtí nejsou ovšem zahrnuty u každé osoby. Lze je považovat za volitelné, doplňující informace.

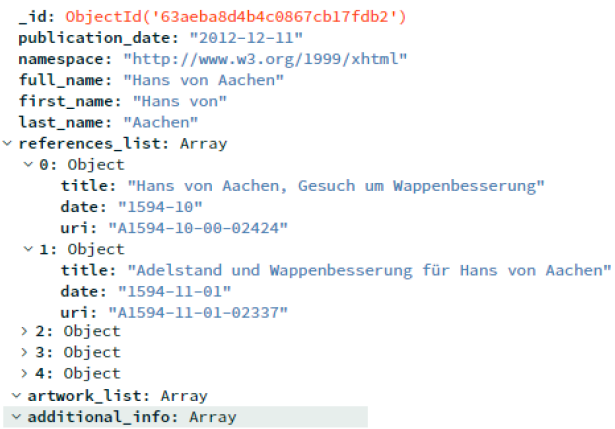
```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="../view_names.xsl" type="text/xsl"?>
<dr:doc xmlns="http://www.w3.org/1999/xhtml" xmlns:dr="http://documenta.rudolphina.org/" v="2012-12-11">
   <title>Aachen, Hans von</title>
   <h3>Hans von Aachen</h3>
   □1552 Köln, +4.3.1615 Prag; <b>Hans von Aach</b>
   <br/>1.1600-2.1612 Kammer-Maler
   <div class="docindex"><b>Quellen:</b>
   <a href="../Regesten/A1594-10-00-02424.xml" title="Hans von Aachen, Gesuch um Wappenbesserung">1594-10</a>
   <a href="../Regesten/A1594-11-01-02337.xml" title="Adelstand und Wappenbesserung für Hans von Aachen">1594-11-01</a>
   <a href="../Regesten/A1601-02-08-01657.xml" title="Eingangsprotokoll des Reichshofrates">1601-02-08</a>
   <a href="../Regesten/A1603-08-02-01960.xml" title="Schuldverschreibung Friedrichs Gf. zu Öttingen">1603-08-02</a>
   <a href="../Regesten/A1604-01-00-02017.xml" title="Daniel Fröschel an Philipp Lang">1604-01</a>
   <a href="../Regesten/A1604-04-27-02063.xml" title="Notariatsinstrument für Benedict Krueg">1604-04-27</a>
   <a href="../Regesten/A1606-05-31-16266.xml" title="Eingangsprotokoll der Hofkammer">1606-05-31</a>
   <a href="../Regesten/A1606-06-01-16370.xml" title="Ausgangsprotokoll der Hofkammer">1606-06-01</a>
   <a href="../Regesten/A1607-01-09-02267.xml" title="Resolutionsprotokoll des Reichshofrates">1607-01-09</a>
   <a href="../Regesten/A1608-03-29-02350.xml" title="Hans von Aachen an Kaiser Rudolph II.">1608-03-29</a>
   <a href="../Regesten/A1608-07-01-02363.xml" title="Befehl Kaiser Rudolfs II. an gfl. Öttingische Regierung zu Wallerstain">1608-07-01</a>
   <a href="../Regesten/A1608-08-12-02365.xml" title="Eingangsprotokoll der Hofkammer">1608-08-12</a>
   <a href="../Regesten/A1608-10-04-02375.xml" title="Gfl. Oettingische Vormundschaft an Kaiser Rudolph II.">1608-10-04</a>
   <a href="../Regesten/A1608-10-25-02388.xml" title="Eingangsprotokoll der Hofkammer">1608-10-25</a>
   <a href="../Regesten/A1609-11-19-02482.xml" title="Liste erkaufter Exotika">1609-11-19</a>
   <a href="../Regesten/A1612-02-00-02640.xml" title="Hofstaatsverzeichnis - Kammerpartei">1612-02</a>
   <a href="../Regesten/A1616-01-12-02721.xml" title="Beilage A zu Jacob Huefnagels Eingabe an die Hofkammer">1616-01-12</a>
 </div>
   <dr:ix> IndexA Aa Aac Aach Aache Aachen</dr:ix>
</dr:doc>
```

Obrázek 6.2: Ukázka původního *XML* souboru - *Personen*. Zdroj: Vlastní zpracování

Informace extrahované z původních *XML* souborů (viz snímek 6.2 výše) jsou považovány za dostatečné pro zachování funkcionalit před transformací do formátu *JSON*. Pro ilustraci transformačního procesu lze odkázat na snímek 6.3, který demonstruje, jak by mohl výsledek transformace vypadat. Transformační skript prochází zdrojový adresář *Person_Namen* a načítá každý *XML* soubor reprezentující osobnost.

Základním prvkem je atribut *full_name*, který se v dalším kroku rozdělí na *first_name* a *last_name*. Oddělovačem mezi jménem a příjmením v původním *XML* formátu slouží čárka. V případě, že čárka chybí, je celý text interpretován jako příjmení. Specifickým případem je situace, kdy jméno osoby neobsahuje jednoznačné rozdělení, jako například u jména `Friedrich I.`, kde se celé označení osoby považuje za příjmení a pole pro křestní jméno zůstává volitelným atributem s možností *NULL* hodnoty.

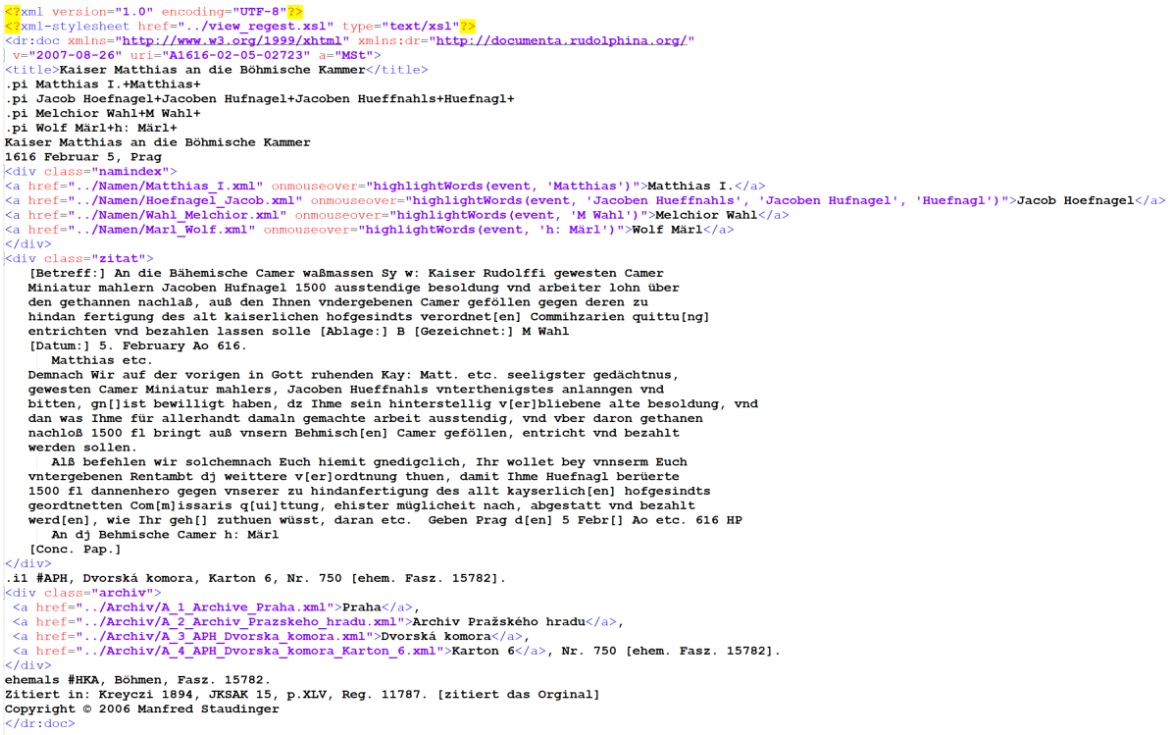
Pro každý načtený *XML* soubor z adresáře *Person_Name* se zavolá funkce *parsePersonXML()*, jež provádí syntaktickou analýzu *XML* struktury pomocí knihovny *minidom*. Z *XML* elementů extrahuje klíčové informace. Extrahovaná data se pak ukládají do objektu typu *Person* definovaného v modulu *Person.py*. Objekt *Person* se následně převádí do formátu *JSON* pomocí metody *to_json()*. Tato metoda generuje *JSON* dokumenty obsahující veškeré informace o dané osobnosti. Vygenerovaný *JSON* dokument se uloží do nového souboru s příponou *.json* v adresáři *Person_Namen_JSON*.



Obrázek 6.3: Ukázka transformace záznamu z Personen do *JSON*. Zdroj: Vlastní zpracování

Dodatečné informace, které nejsou ve formátu *XML* označeny žádným speciálním štítkem, jako jsou data a místa narození či úmrtí, společně s dalšími poznámkami, jsou uloženy jako volný text v atributu *additional_info*. Tento přístup umožňuje flexibilní manipulaci s daty a jejich efektivní využití v rámci nové struktury *JSON*, čímž se zajišťuje, že klíčové informace zůstanou přístupné a uchovány i po transformaci.

***Document* kolekce** se zaměřuje na díla, tedy na zdrojové soubory z původní kolekce *Regesten*, zahrnující názvy a texty spisů. Tato transformace zjednodušuje uchovávání a přístup k textovým obsahům, přičemž zachovává veškeré specifické značky a poznámky přímo v textu, což usnadňuje jejich další zpracování.



Obrázek 6.4: Ukázka původního *XML* souboru z *Regesten*. Zdroj: Vlastní zpracování

Pro každé dílo jsou jako základní a povinné atributy definovány *title* (název díla) a *text* (text díla). Tyto informace představují esenciální prvky každého uměleckého díla a jsou nezbytné pro jeho identifikaci a další zpracování.

Jelikož text díla může obsahovat další značky (např.: *[Betreff:], [Datum:]*), se kterými se dále na *frontendu* může pracovat, je celý text uložen z * * do atributu *text*. Tyto značky jsou součástí textu a jsou uchovávány přímo v atributu *text* bez dalšího rozlišení. Význam *tagů* nelze z kontextu přesně odvodit, mohou připomínat poznámky autora, nebo překladatelské poznámky. Rozhodnutí neuchovávat značky jako samostatné atributy vychází z aktuální praxe nevyužívání těchto značek ke specifickým účelům na původním webu, kde jsou zobrazeny jako regulérní součást textových řetězců. Přesto je možné tyto značky v budoucnu rozdělit a využít, pokud by se ukázalo, že jsou potřebné pro potřeby analýzy nebo výpisu na *frontendu*.

V původních dokumentech je název díla lokalizován mezi značkami *<title>* a *</title>*, tento specifický obsah je následně převeden do identicky pojmenovaného atributu *title*. Vedle názvu díla je rovněž zásadní uchovávat data o jeho vzniku či publikaci. Informace umožňující vzájemné propojení mezi osobou a dílem, a naopak, poskytuje unikátní identifikátor *uri*. Dalšími důležitými údaji, které si přejeme ze zdrojových souborů zachovat, jsou informace o archivu, jež nejsou v původních dokumentech omezeny konkrétním *tagem*, ale jsou identifikovatelné prefixem *.il*. Tyto informace jsou ukládány do atributu *archives* jako pole objektů.

Za relevantní z pohledu zachování jsou také považována jména figurující v dokumentech, která nejsou vymezena specifickým *XML tagem*, avšak jsou uvozena prefixem *.pi*. Tato jména jsou zachována v atributu *names* jako pole jmen, což umožňuje jejich efektivní využití a referování v rámci strukturovaných datových kolekcí.

Transformační proces probíhá obdobným způsobem jako u předešle rozebírané kolekce. Skript prochází zadaným adresářem *Regesten* a načítá *XML* soubory reprezentující textové přepisy. Pro každý načtený *XML* soubor se aplikuje funkce *parseDocumentXML()*, která provádí syntaktickou analýzu *XML* struktury skrze knihovnu *minidom*. Z *XML* elementů extrahuje dílčí informace. Extrahovaná data se pak ukládají do objektu typu *Document* definovaného v modulu *Document.py*. Objekt *Document* se následně převádí do formátu *JSON* pomocí metody *to_json()*. Vygenerovány jsou *JSON* dokumenty obsahující veškeré informace o daném přepisu. Nakonec následuje uložení do adresáře *Regesten_JSON_test*.


```
_id: ObjectId('63b7058a7f9fdb3587e81aed')
uri: "A1616-02-05-02723"
publication_date: "2007-08-26"
date: "1616-02-05"
index: "02723"
namespace: "http://www.w3.org/1999/xhtml"
title: "Kaiser Matthias an die Böhmische Kammer"
archives: Object
  A_1: "Archive_Praha"
  A_2: "Archiv_Prazskeho_hradu"
  A_3: "APH_Dvorska_komora"
  A_4: "APH_Dvorska_komora_Karton_6"
referedIn: "Kreyczi 1894, JKSAK 15, p.XLV, Reg. 11787. [zitiert das Orginal]"
text: "
        [Betreff:] An die Bähemische Camer waßmassen Sy w: Kaiser Rudolffi…"
names: Array
  0: Object
    link: "Matthias_I"
    name: "Matthias I."
  1: Object
    link: "Hoefnagel_Jacob"
    name: "Jacob Hoefnagel"
```


Obrázek 6.5: Ukázka transformace záznamu z *Regesten* do *JSON*. Zdroj: Vlastní zpracování

Poslední kolekce, *ArchiveLinks*, slouží k uchování informací o stromové struktuře archivů, což je klíčové pro orientaci v hierarchii uložených dat a jejich efektivní procházení. Skript načítá původní zdrojové soubory z adresáře *Archiv*. Pro načtené *XML* soubory se používá funkce

parseArchiveLinkXML(), provádějící syntaktickou analýzu *XML* struktury pomocí knihovny *xml.etree.ElementTree*. Z *XML* elementů extrahuje klíčové informace, jako je například název archivu, existence podřízených archivů, odkaz na nadřízený archiv a seznam podřízených archivů. Extrahovaná data se pak ukládají do objektu typu *ArchiveLink* nadefinovaného v modulu *ArchiveLink.py*. Objekt *ArchiveLink* se následně konvertuje do formátu *JSON* pomocí metody *to_json()*. Tato metoda generuje *JSON* dokumenty obsahující veškeré informace o daném archivu, včetně jeho interních struktur. Ukládány jsou v rámci adresáře *Archiv_JSON*.

Přechod z formátu *XML* do *JSON* a následné využití databáze *MongoDB* pro projekt *Documenta Rudolphina* představuje zásadní krok k modernizaci a zefektivnění práce s historickými daty. Tento proces usnadňuje manipulaci s daty a také otevírá nové možnosti pro dotazování a analýzu dat.

6.4.1.2 Formulace dotazů

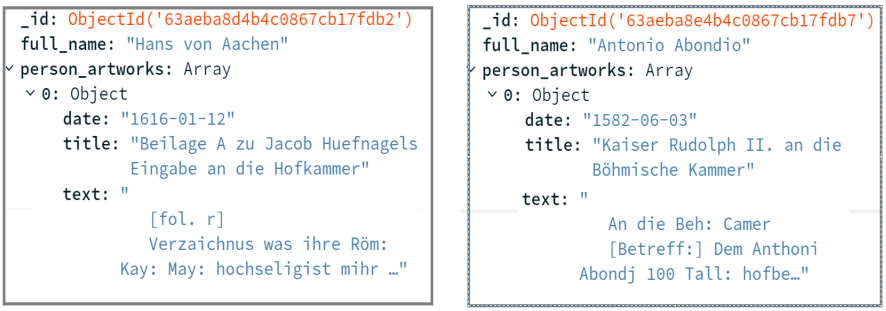
S kolekcemi *Person*, *Document* a *ArchiveLinks* lze provádět širokou škálu dotazů s vysokou mírou přesnosti a efektivity. Práce s těmito kolekcemi je intuitivní a umožňuje uživatelům snadno získávat specifické informace. Primitivní dotazy lze pokládat nad jednotlivými kolekcemi.

V případě kolekce *Person* je možné efektivně realizovat vyhledávání jednotlivců na základě abecedního řazení podle příjmení, aplikovat filtry vycházející z prvního písmene příjmení, a identifikovat osoby dle data narození či úmrtí, s možností specifikovat dotaz například na jedince z 16. století. Tato operace je ovšem limitována dostupností relevantních dat, což je dáno volitelnou povahou některých atributů.

Pro aplikaci verzování a historické analýzy dat se nabízí využití atributu *publication date*, který umožňuje filtrování osob na základě doby jejich přidání do systému a poskytuje základ pro eventuální revize informací.

V kontextu kolekce *Document* lze aplikovat jednoduché vyhledávací operace založené na názvu, obsahu textu, nebo identifikaci osob, které jsou s dílem spjaty. Toto přímé dotazování na atributy kolekce je základním nástrojem pro efektivní práci s daty.

Avšak pokročilé dotazování, které překračuje rámec jednotlivých kolekcí a zaměřuje se na jejich vzájemné propojení, otevírá další dimenze analýzy. Jako příklad lze uvést možnost generování seznamu všech děl nebo textů přiřazených ke specifické osobě (viz 6.6). Případně se lze z jednoho díla dotázat na historické osobnosti, jež v něm figurují (viz 6.7). Tato komplexní dotazovací schopnost je zásadní pro prohlubování porozumění vzájemným vztahům mezi osobami a díly v databázi. Dotazy mohou být konstruovány na základě názvu díla, nebo unikátního identifikátoru (*uri*). Následují výsledky výše popsaných ilustrativních dotazů:



Obrázek 6.6: Výsledky dotazu 1. - Výpis všech děl pro konkrétní osobu (dotaz dohledá díla ke každé osobě v databázi). Zdroj: Vlastní zpracování

Obrázek 6.7: Výsledky dotazu 2. - Výpis osob figurujících v díle (vypsáno pro každé dílo v databázi). Zdroj: Vlastní zpracování

Dále by nás mohlo zajímat například to, kolik děl se nachází v konkrétním archivu. Výsledek takto formulovaného dotazu (viz 6.8) by například pro Pražský archiv, konkrétně archiv s názvem *Karton_4* vypadal následovně:

Obrázek 6.8: Výsledky dotazu 3. - Výpis článků v pražském archivu Karton_4. Zdroj: Vlastní zpracování

Tento přístup k dotazování nejenže značně rozšiřuje možnosti analýzy a interpretace dat, ale také podporuje hlubší porozumění historickým souvislostem, které jsou v rámci *Documenta Rudolphina* evidovány.

Navrhované optimalizace aplikace *Documenta Rudolphina*, včetně přechodu na moderní datový model, mají za cíl výrazně zlepšit přístupnost, efektivitu a uživatelskou zkušenost. Nadcházející implementace pokročilého vyhledávání bude klíčovým prvkem pro podporu vědeckého bádání a šíření poznatků obsažených v této unikátní digitální knihovně.

Principy datové vizualizace

7

Vizualizace dat představuje univerzální koncept zaměřující se na pochopení významu dat prostřednictvím grafické reprezentace. Mnohé vzory, trendy a korelace, které by mohly na první pohled v rámci textově orientovaných dat zůstat uživateli skryté, jsou díky aplikaci softwarových řešení pro vizualizaci dat lépe identifikovatelné a rozpoznatelné. Ve své podstatě je vizualizace dat metodou prezentace informací v grafické podobě, což umožňuje transformaci datových sad, bez ohledu na jejich velikost, do vizuální podoby. Tento formát je pro uživatele snáze pochopitelný. Vizualizace se stává neodmyslitelnou součástí každodenního života, přičemž často se vyskytuje především ve formě grafů a diagramů. Efektivní vizualizace dat je založena na kombinaci různých disciplín, včetně komunikace, designu a zpracování dat. Správně realizované vizualizace poskytují cenné náhledy do komplexních datových sad způsobem, kterým intuitivně poskytují přidanou hodnotu. Tato kapitola se zaměřuje na představení konceptu vizualizace, její přínos, základní principy a techniky a v neposlední řadě i na implementaci reálné aplikace. [IJ19]

7.1 Úvod do datové vizualizace

V současnosti se stává běžnou praxí, že organizace ve stále větší míře integrují do svých interních procesů datové vizualizace a nástroje pro hlubší pochopení klíčových aspektů podnikání. Rozvoj v oblasti informačních technologií a uživatelsky přívětivého počítačového softwaru rozšiřuje analytické možnosti a posiluje datově řízené rozhodovací procesy. Zaměření na klíčové výkonnostní indikátory a datové *dashboardy* zdůrazňuje nutnost analýzy a průběžného sledování kvantitativních podnikových informací (např. objem prodeje, čtvrtletní příjmy, náklady, či tržní podíl a další). Nástroje pro vizualizaci dat mohou nabízet rozšířené prezentační možnosti, včetně různých infografik a interaktivních grafů, umožňující uživatelům proaktivně data zkoumat a analyzovat. [IJ19]

Grafické zobrazení složitých dat může usnadnit jejich pochopení, což kupříkladu podnikům umožňuje efektivněji identifikovat důležité vzory a koncepty ve srovnání s pouhou textovou nebo číselnou formou prezentace těchto dat. Využití grafů, diagramů, map a datových *dashboardů* představuje v procesu vizualizace dat nepostradatelné nástroje pro identifikaci klíčových problémů, formulaci optimálních strategií pro produkty a operace, předpovídání budoucího vývoje a další. [IJ19]

7.2 Cíle datové vizualizace

Vizualizace je fundamentálním nástrojem zpřístupňující složité datové koncepty širokému spektru uživatelů. Rozlišují se u ní tři hlavní cíle. [Ado24] Přičemž každý z těchto cílů odpovídá specifickým uživatelským potřebám a vyžaduje odlišné typy vizualizací pro maximální efektivitu.

- **Explorace**: Vizualizace umožňuje uživatelům objevovat doposud neznámé vzory v datech. Nástroje by měly být rychlé a intuitivní, aby podpořily prozkoumávání dat ve velkém rozsahu.
- **Monitorování**: Speciální vizualizační nástroje jako *dashboardy* umožňují sledovat například výkonnost a efektivitu procesů. Jsou navrženy tak, aby reflektovaly klíčové indikátory úspěchu podniku a byly přímo spojeny s akcemi, které lze učinit na základě získaných dat.
- **Poskytnout vysvětlení**: Některé vizualizace jsou vytvořeny speciálně pro to, aby vysvětlily příčiny a souvislosti stojící za zjištěnými datovými trendy. Takovéto nástroje jsou často uzpůsobeny pro širší publikum a snaží se komplexní témata interpretovat srozumitelněji.

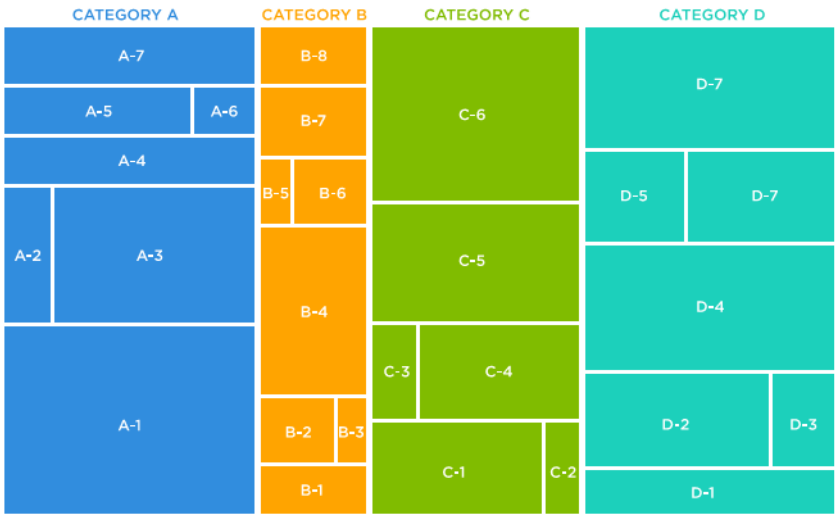
7.3 Historický kontext

Historie datové vizualizace sahá až do 2. století našeho letopočtu, kdy byly zaznamenány první pokusy o vizualizaci dat a to konkrétně v antickém Egyptě. Přední představitelé, jako *René Descartes*, *William Playfair*, *John Tukey* a *Edward Tufte*, měli zásadní vliv na vývoj moderních metod datové vizualizace. Významný pokrok v této oblasti však nastal až v 80. letech minulého století, zvláště s příchodem osobních počítačů a rozvojem informační vizualizace. [Few07]

7.3.1 Aktuální trendy v datové vizualizaci

Jak již zaznělo v předchozích odstavcích, v současnosti se datová vizualizace stává klíčovou složkou *business intelligence* (*BI*), s narůstajícím významem v oblasti výzkumu a praktického použití. Akademické instituce nabízejí specializované programy, které podporují inovace a interdisciplinární spolupráce, čímž napomáhají rozvoji nových aplikací a metod.

Přístup k vizualizaci dat se vyvíjí od jednoduchých grafů k sofistikovaným interaktivním nástrojům, které umožňují uživatelům nejenom data zobrazovat, ale také s nimi interaktivně pracovat, filtrovat a analyzovat. Technologie jako stromové mapy (*tree maps*) přinášejí nové možnosti pro reprezentaci a analýzu dat. [Few07]



Obrázek 7.1: Generická ukázka stromové mapy vizualizující data ve formě vnořených obdélníků. Zdroj: viz https://www.jaspersoft.com/articles/what-is-a-treemapping-chart

7.3.2 Negativní trendy

Trendy v oblasti datové vizualizace mohou vést k zavádějícím praktikám, kdy se kvůli rychlému nasazování nových produktů na trh neklade dostatek důrazu na kvalitu a účelnost. Marketingové kampaně mohou favorizovat vizuálně atraktivní, zato však málo užitečné nástroje, což může degradovat pověst a význam datové vizualizace. Často je termín *datová vizualizace* zneužíván a výsledné produkty nedosahují požadovaných funkcionalit ani účelu. [Few07]

V oblasti datové vizualizace se dále objevuje trend vytváření vizuálních reprezentací, který může vést k nedorozuměním kvůli nedostatečnému vyjádření variability a distribuce dat. Ukázka toho je použití grafů znázorňujících průměrné hodnoty s chybovými úsečkami, které často nedokáží adekvátně prezentovat skutečný rozptyl dat. To může vést k nesprávným závěrům, obzvlášť při analýze malých vzorků, kde extrémní a odlehlé hodnoty mohou zkreslit výsledky. Pro zajištění hlubšího porozumění datům by bylo vhodnější využít interaktivní grafy nebo vizualizace znázorňující jednotlivé datové body. Tyto alternativy, přestože nabízí užitečnější pohled na data, mohou být náročné na implementaci a technicky náročnější na pochopení pro určité uživatele, což vyžaduje pečlivé promyšlení návrhu vizualizací. [Wei+16]

7.3.3 Budoucí směrování - perspektivy a výzvy

Očekává se, že budoucí vývoj datové vizualizace bude směrovat ke zdokonalování stávajících praktik a zároveň bude reagovat na rostoucí počet uživatelů mobilních zařízení a na potřeby vývoje nových, imerzních informačních prostředí. Významným faktorem je také větší zapojení výzkumné komunity do praktického nasazení vizualizačních nástrojů právě v oblasti *BI*.

Smysluplné uplatnění vizualizačních nástrojů v praxi vyžaduje nejen technologické, ale i pedagogické úsilí zaměřující se na rozvoj dovedností potřebných pro práci s daty. [Few07]

7.4 Základní principy datové vizualizace

Efektivní vizualizace dat se zakládá na souboru principů, které usměrňují proces tvorby vizuálů s cílem zvýšit jejich přesnost, jasnost a celkovou efektivitu. Tyto principy pomáhají tvůrcům vytvořit grafická rozhraní, která účinně interpretují složité informace. Níže jsou uvedeny klíčové zásady, doporučené *Stephenem R. Midwayem* v rámci jím publikovaného článku *Principles of Effective Data Visualization*. [Mid20]

1. **Náčrtek (diagram) před vlastní tvorbou:** Před zahájením tvorby vizuálu je důležité mít konkrétní představu o informacích, které chceme sdílet. Tento krok zahrnuje zaměření se na hlavní zprávu a strukturu vizuálu před samotným výběrem a použitím softwaru.

2. **Výběr vhodného softwaru:** Smysluplná vizualizace vyžaduje adekvátní znalost vhodného softwarového řešení. Uvědomění si potřeby osvojit si nový software nebo rozšířit současné znalosti o již používaných programech je zásadní pro vytváření složitých a technicky pokročilých vizualizací.

3. **Použití efektivního způsobu zobrazení dat:** Následuje výběr správného způsobu (geometrie), který odpovídá typu dat a účelu vizualizace. Je doporučeno zobrazovat data přímo v grafu, čímž se zvyšuje jeho srozumitelnost.

4. **Barvy vždy mají svůj význam:** Barevná paleta je mocný vizualizační nástroj, který by měl být používán s úmyslem - barvy by měly být zvoleny tak, aby vyzdvihovaly hodnotu informací a napomáhaly v interpretaci dat.

5. **Zahrnutí nejistoty:** Prezentace nejistoty v datech je klíčová pro plné porozumění prezentovaným informacím. Nejistota by měla být v grafu zahrnuta, aby bylo patrné, co data znamenají a jakými limity disponují.

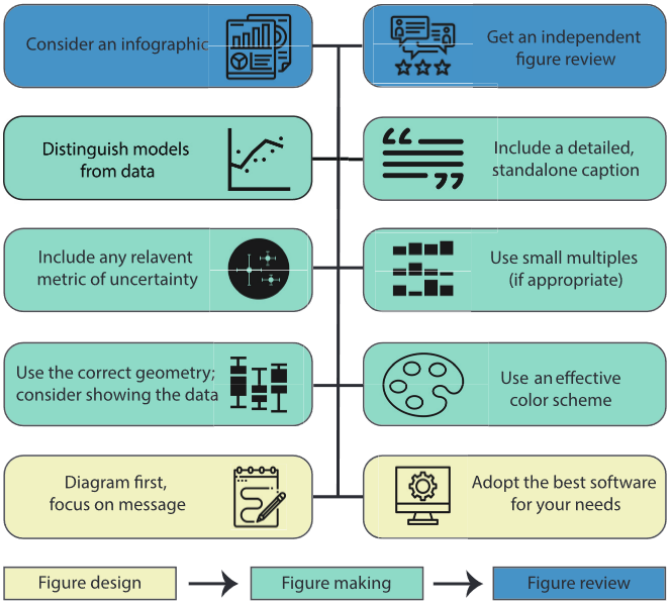
6. **Využití panelů, kdykoli je možnost** (*small multiples* [Tuf01]): Opakování podobně strukturovaného grafu pro zobrazení různých proměnných nebo časových kroků umožňuje snadné porovnání a zvýraznění rozdílů.

7. **Data a modely jsou odlišné záležitosti:** Je důležité rozlišovat mezi primárními surovými daty, zpracovanými daty a modelovými výsledky. Každá z těchto forem komunikace vyžaduje jinou techniku prezentace a vysvětlení.

8. **Jednoduché vizuály, detailní popisky:** Při zachování jednoduchosti designu je žádoucí v grafech poskytnout podrobné popisky, které vysvětlují vše, co je v nich zobrazeno.

9. **Zvážit použití infografik:** Infografiky, které kombinují text, obrázky a další vizuální prvky, mohou být užitečné pro komplexní prezentaci informací.

10. **Získání názoru a zpětné vazby:** Je vhodné získat zpětnou vazbu od kolegů v poli působnosti i od vnějších subjektů pro zhodnocení efektivity a srozumitelnosti vytvářeného vizuálu.



Obrázek 7.2: Přehled uvedených vizualizačních principů. [Mid20]

Tyto principy vycházejí z porozumění, že účinné vizuální sdělení vyžaduje více než jen schopnost umět používat software. Je vyžadována hloubková úvaha, plánování a designové rozhodnutí, které společně přispívají k lepšímu výkladu a sdílení poznatků. [Mid20]

7.4.1 Vizualizační metody a nástroje

Při studiu datové vizualizace, je důležité pochopit nejen principy, ale také techniky, metody a nástroje, které umožňují vizualizaci účinně provádět. [AA16]

7.4.1.1 Techniky a kategorie datové vizualizace

V oblasti datové vizualizace bylo v průběhu času vyvinuto mnoho metod, které umožnily efektivní reprezentaci a zkoumání rozsáhlých datových sad. K těmto metodám se řadí kupříkladu histogramy, liniové grafy, tabulky, koláčové grafy, sloupcové grafy, bodové diagramy, bublinové grafy, plošné grafy, tokové (*flow*) diagramy, *Vennovy* diagramy, diagramy toků dat, časové osy, diagramy vícenásobných datových řad, diagramy vztahů entit, kónické stromy, sémantické sítě, stromové mapy a paralelní souřadnice a mnohé další. Mezi těmito metodami lze vybírat na základě jejich schopnosti vizualizovat data, přičemž pro výběr lze využít různé hodnotící metriky jako použitelnost, interaktivitu a vlastnosti rozhraní. [AA16]

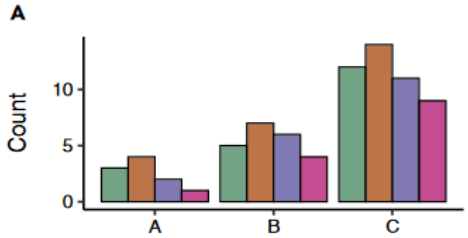
Kategorie datové vizualizace. U vizualizací se rozlišuje několik základních kategorií, dle kterých lze stanovit vhodný typ vizualizace v závislosti na charakteru a cíli reprezentace dat. Konkrétní vybrané techniky jsou popsány níže: [IJ19]

- **Časová** (*Temporal*): Časové vizualizace jsou lineární a jednorozměrné, často využívající linie k zobrazení dat s jednoznačně definovaným začátkem a koncem. Typickým příkladem jsou spojnicové, případně bodové grafy.

- **Hierarchická** (*Hierarchical*): Hierarchické vizualizace řadí podskupiny do větších skupin a jsou ideální pro zobrazování informačních shluků, které vycházejí z jednoho bodu. Pro vizualizaci hierarchických dat a struktur se hodí například stromová mapa (*tree map*).
- **Síťová** (*Network*): Síťové vizualizace znázorňují vztahy mezi datovými sadami umístěnými v síti bez nutnosti složitých popisů. Paralelní souřadnice jsou typicky užitečné pro zkoumání vztahů mezi různými proměnnými a identifikaci vzorců v síti.
- **Multidimenzionální** (*Multidimensional*): Vícedimenzionální vizualizace zahrnují více než dvě dimenze (proměnné) a často vytvářejí 3D datové vizualizace. Bublinový graf, jenž rozšiřuje graf bodový o třetí dimenzi, je ukázkou vícedimenzionální vizualizace. Obdobně lze využít i koláčové grafy, případně histogramy a *Vennovy* diagramy.
- **Geoprostorová** (*Geospatial*): Geoprostorové vizualizace se vztahují k fyzickým lokacím, kde překrývají mapy s datovými body. Standardně se využívají kartogramy, teplotní mapy (*heatmaps*), či mapy toku (*flow maps*). Pro podrobné zobrazení geografických dat společně s jinými souvisejícími atributy v kontextu geoprostorových analýz lze použít i tabulku.

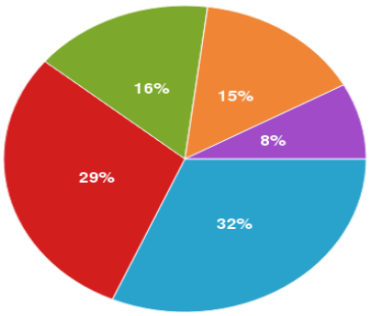
Vybrané techniky a jejich aplikace. Následuje krátký výčet jednotlivých vizualizačních technik, z nichž některé byly stroze uvedeny v předchozích odstavcích, s důrazem kladeným na jejich aplikaci a výhody pro konkrétní scénáře použití. [AA16]

- **Spojnicový graf** (*Line Chart*): Zachycuje vztahy mezi proměnnými a je vhodný pro porovnávání velkého množství položek současně.
- **Sloupcový graf** (*Bar Chart*): Umožňuje srovnání položek z různých skupin pomocí horizontálních nebo vertikálních sloupců.



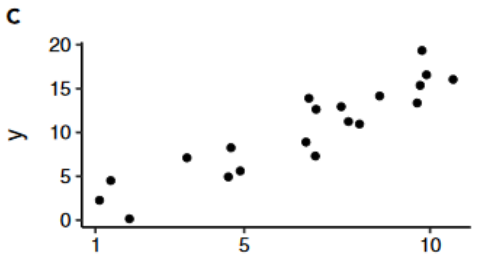
Obrázek 7.3: Ukázka shlukovaného sloupcového grafu. [Mid20]

- **Koláčový a prstencový graf** (*Pie/Doughnut Chart*): Znázorňuje statistiky a data ve formě výsečí, kde velikost výseče indikuje míru zastoupení prvku v dané množině.



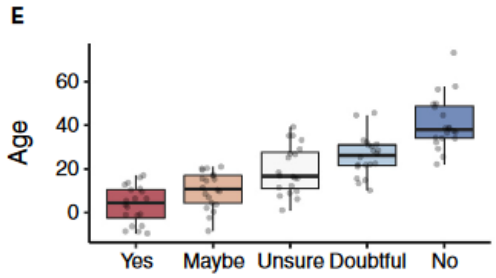
Obrázek 7.4: Ukázka koláčového grafu. [AA16]

- **Bodový graf** (*Scatter Plot*): Ilustruje vzájemnou korelaci dvou proměnných, přičemž jednotlivá data jsou znázorněna pomocí symbolů.



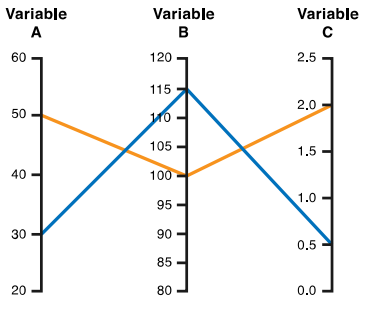
Obrázek 7.5: Ukázková reprezentace bodového grafu. [Mid20]

- **Bublinový graf** (*Bubble Chart*): Rozšiřuje bodový graf o dodatečnou dimenzi, kde velikost symbolu reprezentuje další proměnnou.
- **Krabicový graf** (*Box Plot*): Zobrazuje distribuci datové sady prostřednictvím *kvartilu*. Identifikuje potenciální odlehlé hodnoty, a tak umožňuje rychlý přehled o rozložení a rozptylu dat. [Uni24]



Obrázek 7.6: Krabicový graf seskupující zdrojová data (šedé body). [Mid20]

- **Paralelní souřadnice** (*Parallel Coordinates*): Umožňují vizualizaci vícedimenzionálních dat tím, že propojují body napříč mnoha dimenzemi na paralelních osách.



Data			
	Variable A	Variable B	Variable C
Item 1	50	100	2.0
Item 2	30	115	0.5

Obrázek 7.7: Anatomie grafu paralelních souřadnic, včetně tabulky. Zdroj: *The Data Visualisation Catalogue* (https://datavizcatalogue.com/methods/parallel_coordinates.html)

- **Tabulka** (*Table*): Slouží k systematickému uspořádání a prezentaci dat v řádcích a ve sloupcích, což umožňuje uživatelům snáze porovnávat hodnoty, identifikovat různé vzory a trendy, případně analyzovat vztahy mezi proměnnými. [BI24]
- **Stromová Mapa** (*Tree Map*): Zachycuje hierarchická data ve formě vnořených obdélníků, umožňuje srovnávání uzlů a poduzlů na různých úrovních.

Z výše uvedených metod je patrné, že každá z nich slouží ke specifickému účelu a je vhodná pro různé typy dat a analýz. Výběr správné vizualizační metody je rozhodující pro efektivní interpretaci dat. Je žádoucí, aby tvůrci vizualizací zvážili charakteristiky svých dat a předmět sdělení při výběru vizualizačních metod. [AA16]

7.4.2 Vizualizační aspekty při analýze dat

Podsekce se zaměřuje na rozbor klíčových vizualizačních aspektů, nezbytných pro efektivní prezentaci. Využity jsou informace z výzkumného článku *Cheat Sheets for Data Visualization Techniques*[38] nabízející přehled různých přístupů a technik v datové vizualizaci. Techniky, jako časové křivky, paralelní souřadnice, *box-ploty* a další, byly zkoumány s důrazem na jejich správné použití a rozpoznávání běžných chyb, ke kterým dochází při jejich aplikaci.

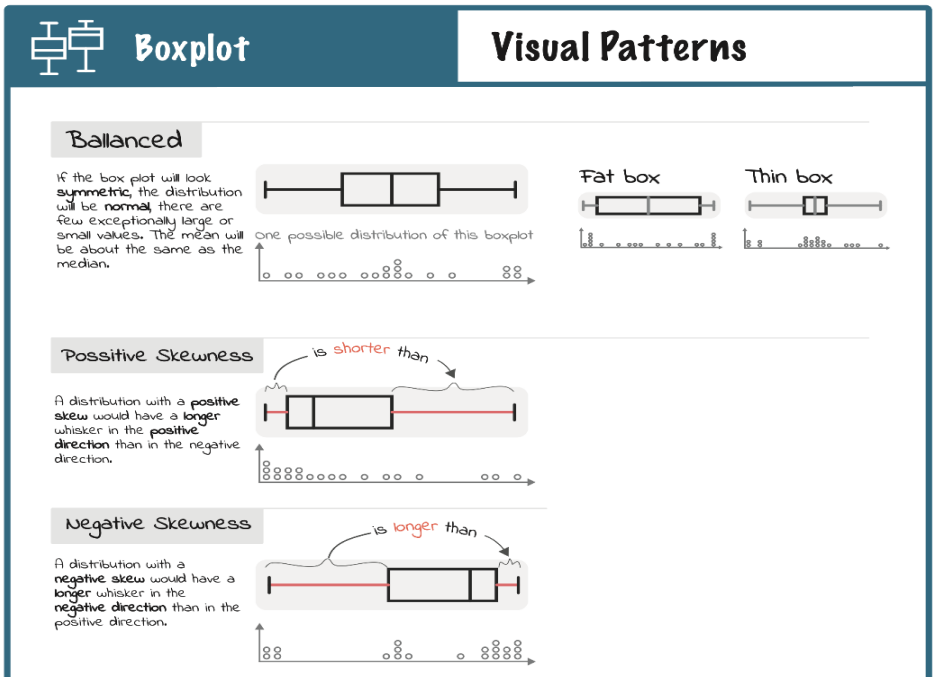
Článek zavádí koncept tzv. *Cheat sheets* jakožto přehledných vizuálních a textových návodů, které mohou uživateli usnadnit práci s konkrétními vizualizačními technikami. Tyto materiály umožňují rychle se zorientovat v klíčových aspektech dané techniky a náležitě ji používat v praxi.

Cheat sheets poskytují zjednodušené přehledy o tom, jak fungují různé vizualizační techniky, jaké mají vizuální vzory a jaká jsou potenciální úskalí jejich použití. Doprovodné příklady z *The Data Visualization Catalogue*[39] (kupříkladu obrázek 7.7 výše) napomáhají lepšímu porozumění a výběru techniky pro konkrétní úlohy.

7.4.2.1 Využití tvarů a rozvržení

Vizualizační techniky mnohdy využívají různé tvary a pečlivě navržený design prvků ke zvýraznění klíčových aspektů dat a zlepšení jejich srozumitelnosti. Barvy a tvary v časových křivkách mohou v rámci dat zvýrazňovat trendy, či významné změny. Krabicové grafy níže na obrázku 7.8 ilustrují, jak rozložení a orientace vizuálních prvků mohou zásadním způsobem ovlivnit interpretaci dat. [Wan+20]

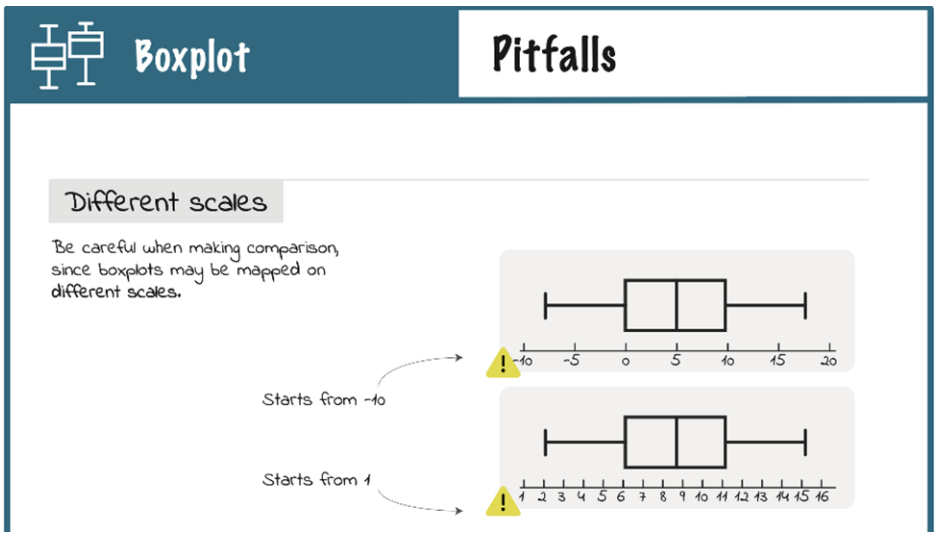
[38] *DOI*: https://doi.org/10.1145/3313831.3376271
[39] https://datavizcatalogue.com/



Obrázek 7.8: Příklad vizuálních vzorů pro krabicový graf. [Wan+20]

7.4.2.2 Interpretace a běžné chyby

Správná interpretace vizualizačních technik je klíčová, tudíž je žádoucí se u vybrané techniky vyvarovat běžným chybám. Například, u časových křivek je kriticky důležité správně pochopit metriku podobnosti, jež ovlivňuje interpretaci vzdáleností mezi body. U všech grafů je důležité si uvědomit, jakým způsobem interpretovat rozložení dat, aby se předešlo nesprávným závěrům. [Wan+20]



Obrázek 7.9: Příklad častých chyb při tvorbě krabicových grafů [Wan+20]

7.4.2.3 Designové principy a metodologie

Autoři článku[40] zdůrazňují význam modularity, kde se každý *Cheat sheet* zaměřuje na určitý aspekt vizualizace, čímž je redukováno informační přetížení a umožněna adaptabilita jimi poskytovaných materiálů na různé situace. Důležitá je také nezávislost na kontextu, z čehož plyne, že *Cheat sheets* by neměly záviset na specifických datových ukázkách, takže je možné je plošně využívat. [Wan+20]

Právě popsaná koncepce hraje zásadní roli v podpoře vizuální gramotnosti a pomáhá lépe porozumět složitým vizualizacím. Rafinovaně zvolené designové principy a veřejně přístupná kolekce těchto *cheat sheets* napomáhají k vytvoření zdroje materiálů, které rozvíjí dovednosti a usnadňují práci s rozličnými vizualizačními technikami.

7.4.2.4 Nástroje datové vizualizace

Vizualizační nástroje obvykle již v základu obsahují konektory k populárním datovým zdrojům, včetně běžných databází (relačních i *NoSQL*), *Hadoop* a různých cloudových úložišť. Uživatel může dle svého uvážení zvolit pro něj nejlepší způsob prezentace dat, přičemž některé nástroje již tento krok automatizují a optimálně vyberou typ grafu na základě analýzy dat. Standardně tyto softwary nabízejí *dashboardovou* komponentu umožňující uživatelům manipulovat v rámci jednoho rozhraní s více vizualizacemi a analýzami naráz, a to zpravidla ve webovém portálu. Následuje stručný přehled nástrojů používaných k datové vizualizaci. Každý z níže uvedených nástrojů nabízí různé možnosti pro tvorbu vizuálů, od jednoduchých grafů až po složité interaktivní vizualizace. Pro každý nástroj je uveden krátký popis jeho funkcionalit, výhod a potenciálních omezení, doplněný o příkladové ilustrativní obrázky. [IJ19]

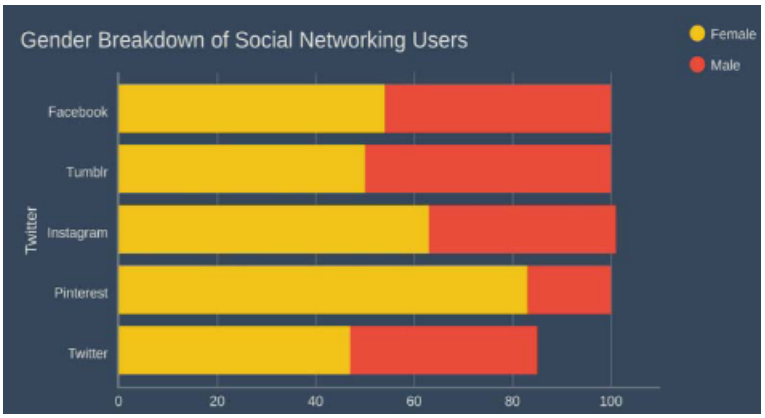
Tableau. Dle výrobce představuje *Tableau* robustní nástroj pro datovou vizualizaci se širokou škálou možností týkajících se formátu vstupních i výstupních dat. Nabízí mapovací funkce, které umožňují vytvářet barevně kódované mapy z geografických dat. *Tableau Public* je volně dostupná verze pro tvorbu veřejně přístupných vizualizací.

- **Výhody**: Široká škála možností vstupních dat, mapovací schopnosti, volně dostupná veřejná verze.
- **Limitace**: Vysoké měsíční náklady u placené verze, u volně dostupné varianty nelze uchovávat datové analýzy soukromě.

ChartBlocks. Umožňuje nahrání dat *odkudkoli* prostřednictvím jimi poskytovaného *API*. Aplikace podporuje rozsáhlou modulárnost vytvořené vizualizace a její výstupy jsou responzivní.

- **Výhody**: Dostupnost řešení, uživatelská přívětivost při vytváření základních grafů.
- **Limitace**: Nejasná robustnost *API*, neumožňuje vytváření map.

[40] *DOI*: https://doi.org/10.1145/3313831.3376271



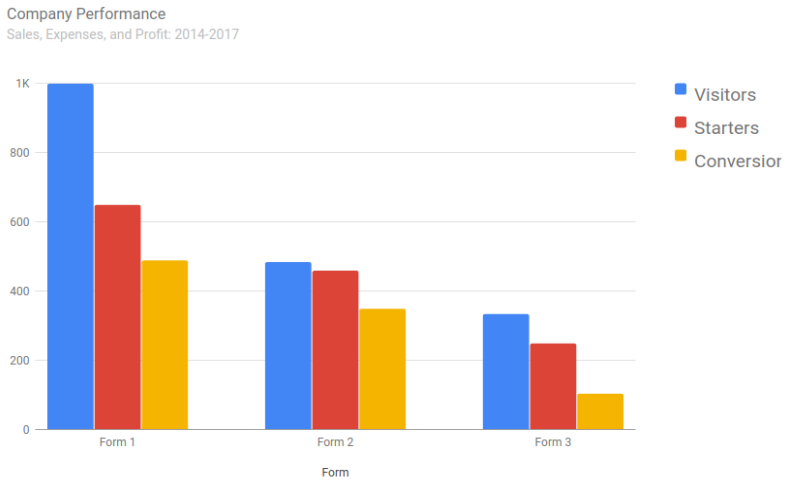
Obrázek 7.10: Ukázka vizualizace v *ChartBlocks*. [IJ19]

Datawrapper. Byl navržen speciálně pro přidávání grafů a map do novinových článků. Vizuály jsou interaktivní a určené pro přímočaré vložení na webové stránky zpravodajských serverů.

- **Výhody**: Specificky navržen pro potřeby redakcí, zahrnuje zajištění přístupnosti pro tělesně postižené (barvoslepost apod.).
- **Limitace**: Limitovaná podpora datových zdrojů, předplatné.

Google Charts. Jedná se o bezplatný nástroj pro tvorbu interaktivních grafů primárně určených pro vkládání do *online* platforem. Podporuje dynamická data a je kompatibilní napříč prohlížeči díky tomu, že jeho výstupy jsou založené na technologiích *HTML5* a *SVG.*

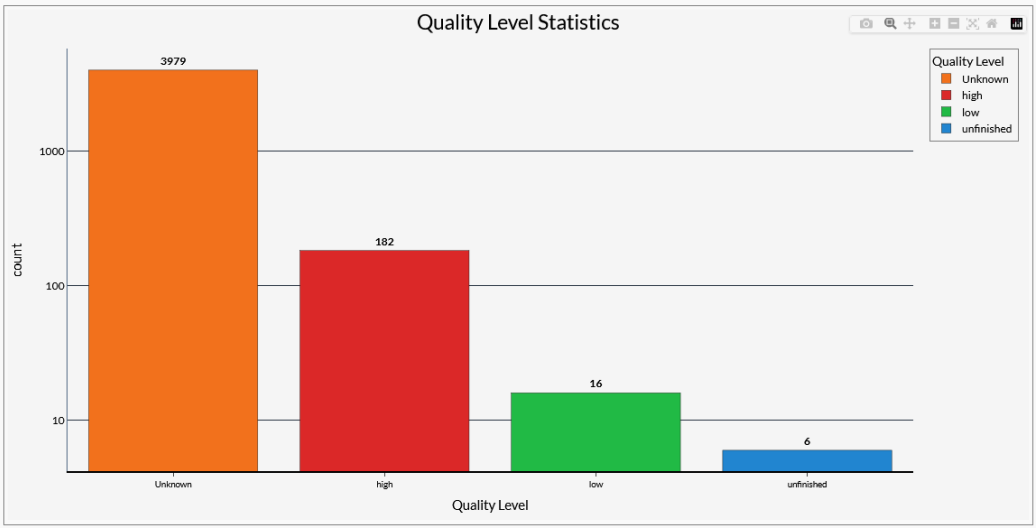
- **Výhody**: Zdarma, široká škála formátů grafu, kompatibilita napříč prohlížeči.
- **Limitace**: Omezená podpora mimo základní návody a komunitní fórum.



Obrázek 7.11: Příklad vizualizace v prostředí *Google Charts*. Zdroj: viz https://reflectivedata.com/dictionary/google-charts/

Dash (Plotly). Vyvinut firmou *Plotly*, jedná se o *framework* pro jazyk *Python*. Umožňuje tvorbu interaktivních webových aplikací a *dashboardů* bez nutnosti pokročilých znalostí *frontend* technologií. Využívá se především pro analytické a monitorovací účely v různých sférách. [Rod24]

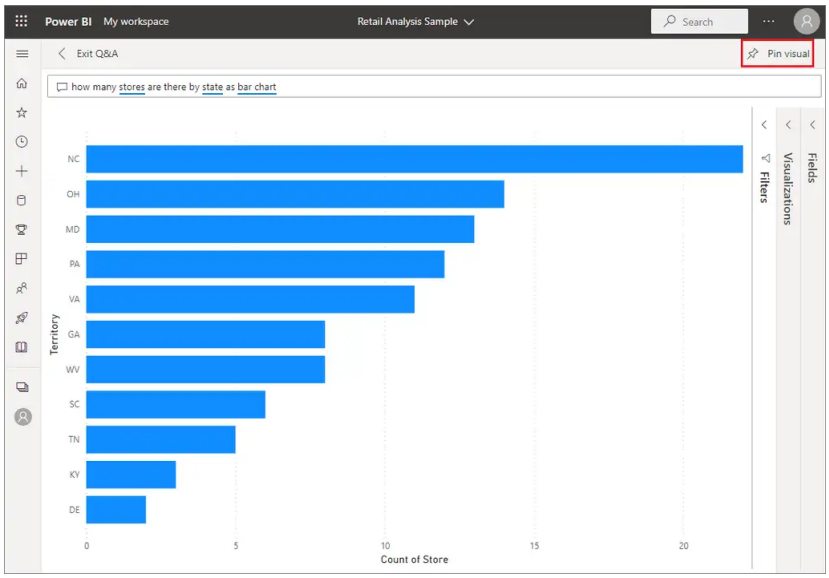
- **Výhody:** Umožňuje snadnou integraci s *Pythonem*, podporuje interaktivní prvky bez nutnosti používat *JavaScript*.
- **Limitace:** Vyžaduje znalost *Pythonu*, komplexnější aplikace jsou náročné na výkon.



Obrázek 7.12: Ukázkový sloupcový graf v *Dash*. Zdroj: Vlastní zpracování

Power BI. *Power BI* od *Microsoft* je analytická služba poskytující nástroje pro analýzu dat a sdílení informací. Je hojně využívána napříč podniky pro tvorbu reportů a *dashboardů*. [Alt22a]

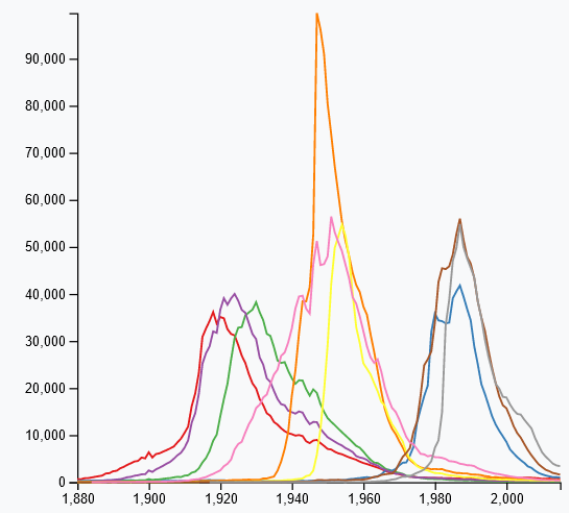
- **Výhody:** Integrace s dalšími produkty od *Microsoft*, uživatelsky přívětivé rozhraní.
- **Limitace:** Může být nákladné ve větším měřítku, limitované možnosti přizpůsobení.



Obrázek 7.13: Příklad jednoho z mnoha nástrojů (*Q&A*) dostupných v *PowerBI*. [Alt22a]

D3.js. *JavaScriptová* knihovna určená pro manipulaci s dokumenty založených na datech. Umožňuje vývojářům vytvářet detailní grafy integrované přímo do webových stránek. [Gee19]

- **Výhody:** Vysoká flexibilita a kontrola nad finálním výstupem, podpora velkého množství vizualizačních typů a technik.
- **Limitace:** Vyžaduje pokročilé znalosti *JavaScriptu* a *SVG*.



Obrázek 7.14: Ukázka několika spojnicových grafů v jednom zobrazení v rámci *D3.js*. Zdroj: viz https://d3-graph-gallery.com/graph/line_several_group.html

Qlik Sense. *Qlik Sense* představuje platformu pro analýzu dat, tvorbu interaktivních reportů a *dashboardů*, která poskytuje automatické generování datových vztahů. [Das23]

- **Výhody:** Intuitivní rozhraní pro koncové uživatele, silná podpora *samoobslužného BI*.
- **Limitace:** Omezená kontrola nad návrhem aplikací ve srovnání s tradičními *BI* nástroji, méně robustní u případů s velmi specifickými požadavky.

Přehled poskytuje ucelený pohled na různé metody a nástroje používané v oblasti datové vizualizace od jednoduchých aplikací pro laickou veřejnost po pokročilé nástroje vyžadující programovací znalosti.

Platí, že klíčovým krokem k úspěšné vizualizaci je porozumění datům a potřebám, které je nutné skrze datové reprezentace uspokojit. Kupříkladu časová data si žádají odlišný přístup než data hierarchická či geoprostorová a výběr správné techniky může zásadně ovlivnit interpretaci a porozumění datům. Nástroje jako *Dash*, *Tableau*, či *Google Charts* rozšiřují možnosti vizualizace tím, že nabízejí širokou paletu možností, od mapování přes interaktivní *dashboardy* až po sofistikované vícedimenzionální grafy. Výběr vhodného nástroje, který se dokáže patřičně adaptovat datovým specifikám i vizualizačním potřebám, je nezbytný pro efektivní sdílení a interpretaci obsažených informací. Úspěch vizualizace tedy spočívá ve správném sladění charakteristik dat, volby metody a využití nástrojů, které umožní zvolenou metodu co nejefektivněji aplikovat.

7.4.3 Principy tvorby *dashboardů*

Nyní budou představeny základní principy a vzory používané při tvorbě *dashboardů*. Diskutovány jsou klíčové aspekty jako význam a účel *dashboardů*, principy vizuálního vnímání a důležité aspekty efektivního designu a *layoutu* u *dashboardů*.

7.4.3.1 Definice a účel

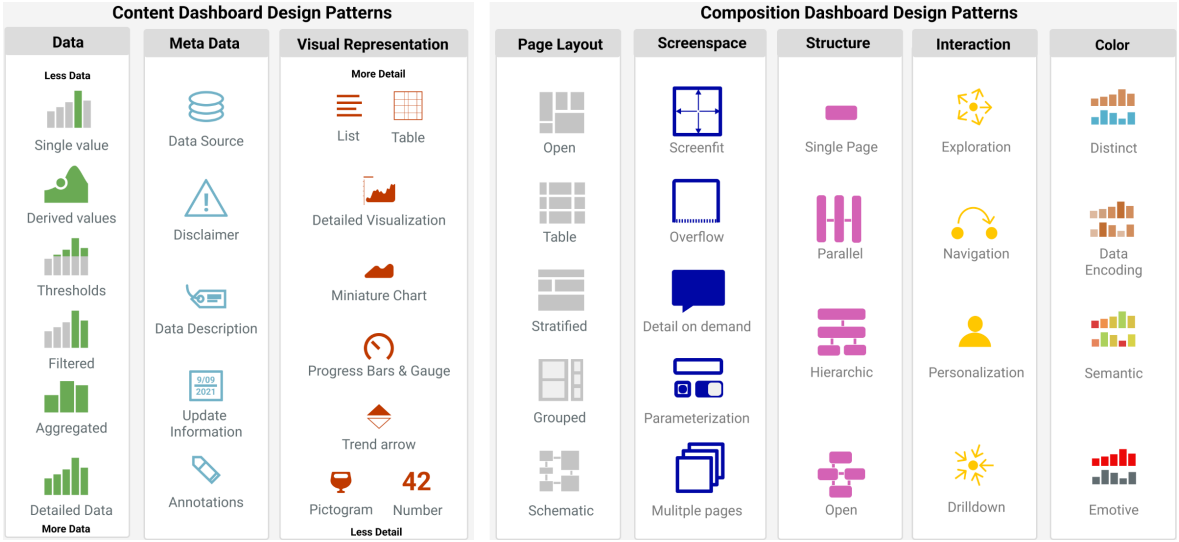
Dashboardy lze definovat jako nástroje optimalizované pro přehlednost a rychlé zorientování v komplexních datových souborech prostřednictvím vizualizačních metod. Tyto nástroje umožňují uživatelům monitorovat důležité informace a podporují rozhodovací procesy tím, že poskytují ucelený pohled a identifikují trendy a vzorce v rozsáhlých datech. [Bac+22]

7.4.3.2 Principy a vzory pro *dashboardy*

Při koncipování *dashboardů* je nutné aplikovat osvědčené vizualizační principy, které napomáhají k efektivnímu a intuitivnímu designu. Kromě základních principů datové vizualizace (viz výše sekce 7.4) je vhodné přihlédnout i k dalším dvěma sadám principů, které ve spojení s původními zvyšují funkčnost a estetickou hodnotu designu: [Bac+22]

- ***Gestalt* principy**: Tato teorie pomáhá uživatelům intuitivně přistoupit ke struktuře *dashboardů*. Vzhledem k univerzálnosti těchto principů je lze uplatnit i při samotné tvorbě grafů. Principy jako blízkost, podobnost a dostatečná kontinuita napomáhají logickému uspořádání obsahu. [Tod08]
- **Principy vizuálního vnímání**: Tyto principy zdůrazňují význam práce s barvami tak, aby byly informace na *dashboardu* prezentovány jasně a zároveň byly lehce rozlišitelné. Zásadní je správné využití kontrastu a barevné harmonie pro zdůraznění nejdůležitějších datových bodů.

Dashboardová vizualizace vyjma obecných principů uplatňuje i širokou paletu návrhových vzorů. Vzory lze dle článku *Dashboard Design Patterns*[41] rozdělit do osmi kategorií, pokrývající celé spektrum - od struktury obsahu po vizuální podání.



Obrázek 7.15: Souhrn osmi definovaných návrhových vzorů. [Bac+22]

Návrhové vzory jsou přehledně shrnuty na výše uvedeném obrázku 7.15, na němž jsou zachyceny všechny kategorie a jejich podskupiny. Tím poskytuje kompletní představu o mož-

[41] *DOI*: http://dx.doi.org/10.1109/TVCG.2022.3209448

nostech vizuálního designu *dashboardů*. Každý vzor přispívá unikátním způsobem k funkčnosti a estetické hodnotě *dashboardu*. Jmenovitě se jedná o: [Bac+22]

- **Data a metadata:** Zahrnují způsoby, jak prezentovat data, od jednoduchých hodnot po agregované sady, a jak je zasadit do kontextu pomocí metadat, jako jsou zdroje dat a anotace.
- **Vizuální reprezentace:** Rozsáhlé vizualizační možnosti, od číselných ukazatelů až po detailní grafy, umožňují uživatelům rychle absorbovat informace a identifikovat klíčové trendy, které mezi daty figurují.
- **Rozložení a struktura stránky:** Efektivní rozvržení a struktura *dashboardů*, od otevřeného po schematické rozložení, napomáhají v organizaci informací a srozumitelnosti prezentovaných dat. Detailní naplánování *layoutu* je základem pro snadnou navigaci a rychlé nabytí potřebných informací.
- **Využití prostoru a interaktivita:** Efektivní využití prostoru na obrazovce, například tak, že každý pixel hraje svou roli, a možnost zjistit více informací dle potřeby, napomáhá lépe porozumět obsahu a nabízí prostor pro objevování nových souvislostí.
- **Barevnost:** Vhodně zvolené barevné palety a odstíny nejen, že usnadňují orientaci v datech a vzájemné rozlišování, ale také mohou zvýšit výraznost vizualizace.

7.4.3.3 Typy *dashboardů*

Autoři skrze analýzu odhalili, že různé typy *dashboardů* sdílí tytéž charakteristické vlastnosti, podobně aplikují návrhové vzory a snaží se dosáhnout identických cílů. [Bac+22]

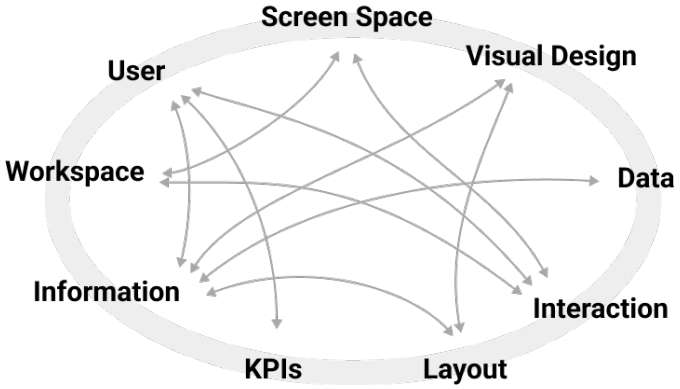
Na základě těchto poznatků se rozlišují dva specifické typy - kurátorské (*curated*), případně dozorované, *dashboardy*, které jsou vysoce selektivní v prezentaci dat a vizuálních prvků za účelem dosažení konkrétního cíle, a explorativní *dashboardy*, jejichž cílem je přenos velkých informačních objemů, aby uživatelé mohli vyhledávat pro ně nejrelevantnější informace. Kurátorské *dashboardy* lze charakterizovat jako autorem řízené vyprávění (*storytelling*[42]), zatímco u explorativních *dashboardů* je *storytelling* řízen čtenářem. [Bac+22]

Kurátorské *dashboardy* jsou navrženy tak, aby se standardně vešly na jednu obrazovku a typicky využívají uspořádání *screenfit* nebo *overflow* (viz kategorie *Screenspace* na obrázku 7.15). Nabízejí jen limitované možnosti úprav a vizuálního zobrazení dat, což se odráží v nízké parametrizaci (např. filtrování, řazení, výběr a další). Tvůrce těchto *dashboardů* musí pečlivě vybírat jak data, tak způsob jejich vizuálního zpracování, aby odpovídaly požadovanému účelu a poskytovaly cílené informace. Naproti tomu, explorativní *dashboardy* jsou navrženy pro poskytování rozsáhlého množství dat v různých formátech a pro různorodé analytické účely či sdílení. Záměrně není kladen důraz na omezení obsahu, aby uživatel mohl pohodlně nalézt jím požadované informace. Tyto *dashboardy* jsou vzhledem ke své variabilitě obvykle orientovány na širší publikum. [Bac+22]

[42] *Pozn.*: Koncept budování poutavého příběhu založeného na komplexních datech a analytických nástrojích, který pomáhá odvyprávět příběh s cílem ovlivnit a informovat konkrétní cílovou skupinu. [Mic24]

7.4.3.4 Kompromisy při návrhu *dashboardů*

Proces vytváření *dashboardů* zahrnuje neustálé vyvažování klíčových faktorů, část z nich je zachycena níže na snímku 7.16. Spojnice reprezentují vzájemný vztah mezi dvojicí faktorů. Jmenovitě se jedná například o prostor dostupný pro vizualizaci (*Screenspace*), počet stránek, na nichž jsou informace rozvrženy (*Number of Pages*), úrovně abstrakce (*Abstraction*) a míru interaktivity nutné pro přístup k informacím (*Interaction*). Každé dílčí rozhodnutí může vyžadovat kompromisy mezi těmito faktory. Například, zmenšení vizuálního prostoru může vést k nutnosti zvýšit počet stránek nebo interaktivitu, aby se uživatelé mohli účinně navigovat a získávat podrobnosti. Naopak, zvětšením vizuálního prostoru je možné snížit počet stránek a také limitovat potřebu interaktivity. Tento proces vyvažování je kontinuální a vyžaduje opakované iterace k dosažení optimálního designu, v němž je nerovnováha mezi faktory minimalizována. Cílem je najít takovou konfiguraci, která nejefektivněji využije dostupné zdroje a zároveň poskytne uživatelům co nejpřehlednější a nejintuitivnější zobrazení dat. [Bac+22]



Obrázek 7.16: Simplifikovaný model kompromisů při návrhu *dashboardů*. [Bac+22]

7.5 Praktická realizace datové vizualizace

Vizualizace je prakticky realizována nad transformovanou databází projektu *Inventaria Rudolphina*. Bylo tak rozhodnuto na základě kontextu zadání při zohlednění skutečnosti, že v *Inventaria* již existuje implementace *dashboardového* zobrazení, jak uvádí podsekce 3.1.2. Realizace vizualizací na transformovaných datech slouží k nastínění, jak by bylo možné původní *dashboardy* rozšířit o další funkcionalitu a zlepšit grafické zobrazení dat v souladu s moderními přístupy k datové vizualizaci, jež byly popsány v předešlé kapitole.

7.5.1 Volba vhodných technik a metod

Výběr technik, metod a nástrojů pro vizualizaci transformovaných dat byl koncipován s ohledem na specifika datové sady *Inventaria Rudolphina* a cíle projektu. Pro tvorbu *dashboardové* aplikace byla vybrána platforma *Dash* od komunity *Plotly*, která umožňuje zobrazování statistik přímo z *MongoDB* databáze. Navíc podporuje širokou škálu různých typů grafů, čímž uživatelům nabízí dostatečnou flexibilitu v prezentování dat. Aplikace byla navržena pro dynamické zobrazování dat pomocí interaktivních grafů a je určena pro uživatele, kteří se zajímají o grafický způsob datové reprezentace.

Plotly Dash je považován za flexibilní nástroj pro tvorbu interaktivních webových *dashboardů*. Oproti předešle uvedené (7.4.2.4) konkurenci nabízí několik specifických benefitů, které jej odlišují. Díky možnosti psát aplikace přímo v jazyce *Python*, mohou pokročilí uživatelé pohotově začít s vývojem. Zároveň i případná integrace dat z již existujících *Python* projektů je intuitivní a plynulá. [SMW22]

Kompaktnost kódu v *Dash* usnadňuje vytváření prototypů a iterování. To je užitečné v agilním vývoji, kde se požadavky na produkt neustále mění. *Dash* efektivně maskuje technickou složitost, jako je komunikace mezi *JavaScriptovým frontendem* a *Python backendem*, což snižuje potřebu aplikovat opakující se kód a ulehčuje definici *API* koncových bodů a správu *HTTP* požadavků. Navíc je platforma vyvíjena týmem *Plotly*, zajišťujícím propojení se stejnojmennou knihovnou pro tvorbu grafů. *Dash* je zkonstruován na *frameworku Flask*[43], čímž jsou mu poskytnuty široké možnosti nasazení, od plně spravovaných externích řešení po soukromé *hostování*.

Ačkoliv je platforma limitována na použití jazyka *Python*, lze ji rozšířit o elementy *CSS*, *JavaScriptu* nebo o vlastní komponenty vytvořené skrze *React* [44]. Dále dovoluje integraci i jiných grafových knihoven jako jsou *Matplotlib*[45] a *Seaborn*[46] do vytvořených webových aplikací. Pro *Matplotlib* lze použít nástroj *mpl_to_plotly* [47] z modulu *plotly.tools* pro převedení grafů do formátu kompatibilního s *Dash*. Pro *Seaborn* je možné grafy exportovat jako statické obrázky a následně je vkládat do *Dash* aplikací jako *HTML* obrázky. Přestože *mpl_to_plotly* nabízí přímou konverzi, novější verze *Matplotlib* mohou vyžadovat alternativní přístupy, jako je *mpld3* pro převod grafů na *HTML* kód, snadno integrovatelný do webových aplikací. [Sch23]

I přes mnohé přednosti má *Dash* i svá omezení. Pokud aplikace obsahuje velký počet komponent, případně zpracovává rozsáhlé datové sady, může docházet ke zpomalení. Zprovoznění může být složitější ve srovnání s některými nástroji postavených na metodách typu *no-code* či *low-code*[48]. Schopnost začlenění s jinými podnikovými softwary není na tak vysoké úrovni jako je tomu například u *PowerBI* a jeho propojení s dalšími produkty od *Microsoft*. Pro využití plného potenciálu je také nezbytné disponovat základními znalostmi *HTML* a *CSS*. [SMW22]

Pro případné začlenění vizualizací vytvořených v *Dash* do projektu *Inventaria*, který využívá předešle uvedený technologický *stack* (viz 3.1.1), lze uplatnit vnořený rám, tedy *IFrame*[49], prostřednictvím něhož se *Dash* aplikace vloží do *Vue.js frontendu* jako externí stránka. Tato metoda umožňuje udržet *Dash* aplikaci izolovanou od hlavního systému. Přesto může toto řešení působit jako nesourodý prvek v uživatelském rozhraní. Alternativu představuje přepsání vytvořených vizualizací pro zajištění přímé kompatibility s *Vue.js* a *Semantic UI*. To může být přímo realizováno pomocí knihovny *vue-plotly*[50]. Toto řešení zajistí konzistentní uživatelské rozhraní a odstraní potřebu správy dalších serverových zdrojů. Ačkoliv vyžaduje více vývojo-

[43] https://flask.palletsprojects.com/en/3.0.x/
[44] https://react.dev/learn
[45] https://matplotlib.org/stable/#learn
[46] https://seaborn.pydata.org/tutorial.html
[47] https://plotly.github.io/plotly.py-docs/generated/plotly.tools.mpl_to_plotly.html
[48] https://www.sap.com/products/technology-platform/low-code/what-is-low-code-no-code.html
[49] https://www.w3schools.com/html/html_iframe.asp
[50] https://www.npmjs.com/package/vue-plotly

vého úsilí než-li *IFrame*, výsledkem je plně integrované řešení, které lépe splyne s existujícím prostředím projektu *Inventaria*.

Výběr mezi těmito dvěma přístupy by měl zohlednit jak technické možnosti vývojářského týmu, tak dlouhodobé směrování projektu.

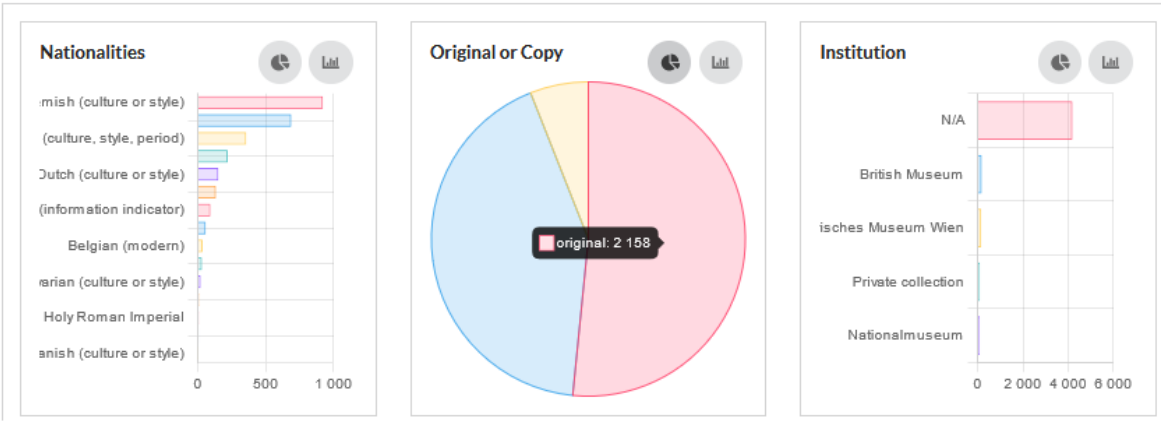
7.5.2 Rozbor implementace

Následující odstavce se zabývají funkcionalitami a implementací vizualizační aplikace zvané ***graphDash***. Pro spuštění a správný běh aplikace je nezbytné mít nainstalováno několik závislostí a knihoven, včetně *Pythonu* verze 3.11 nebo novějšího, dále mít zprovozněný *Communtity Server* od *MongoDB*, a různé *Python* knihovny a *frameworky* jako již zmíněný *Dash* či *Pandas*. Veškeré externí požadavky jsou specifikovány v souboru *requirements.txt* umístěném v kořenovém adresáři projektu aplikace (viz Příloha G). Další specifikace a kroky týkající se instalace a ovládání aplikace jsou podrobněji popsány v projektovém *README.md* souboru.

7.5.2.1 Možnosti využití

Aplikace poskytuje uživatelům interaktivní průzkum a vizualizaci předdefinovaných scénářů, které si uživatel může libovolně procházet. Jedná se tak o formu kurátorského vyprávění (*curated storytelling*). Byl zachován (případně vylepšen) původní rozsah, respektive statistiky napříč inventáři, znázorněné v rámci *Dashboard view Inventaria*. Jedná se o následující scénáře:

- Zastoupení uměleckého stylu z hlediska národnosti autora (***Nationalities***).
- Četnost stanovená podle *provenience*, případně pravosti díla (***Original or Copy***).
- Míra zastoupení uměleckých institucí - kolik inventárních děl je v rámci nich evidováno (***Institution***).
- Počet uměleckých děl připadajících na daného umělce (***Artists Name (ULAN only)***).
- Počet děl nacházejících se v příslušném inventáři (***Inventory***).
- Informace vyčíslující díla dle kvalitativního stavu (***Quality***).
- Míru zastoupení klíčových slov napříč díly (***Keyword***).
- Počet děl, nacházejících se v průběhu času v dané komnatě na Pražském hradě (***Room***).



Obrázek 7.17: Původní stav *dashboardového* zobrazení v *Inventaria*, ukázka scénářů a možností zobrazení. [Úst20a]

Těchto osm původních scénářů bylo dále rozšířeno o čtyři dodatečné: Množství položek (ne)obsahujících klíčová slova (***Documents with/without Keywords***); Počet položek, které obsahují ilustrativní ukázku obrazu (***Inventory Records Images***); Množství věcně souvisejících děl k dané položce (***Item Concordances***) a v neposlední řadě možnost zobrazit si časovou osu vývoje zvolené položky napříč inventáři (***Timeline***)[51]. Celkově lze procházet dvanáct samostatných *dashboardů*.

Po spuštění je aplikace dostupná skrze webový prohlížeč, kde uživatelé mohou provádět následující interakce:

- **Vybrat si typ grafu:** Uživatelé mohou volit mezi sloupcovými a prstencovými grafy, které jsou doprovázeny tabulkou pro detailnější analýzu dat.
- **Dynamicky filtrovat a řadit:** Možnosti zahrnují vyhledávání položek dle názvu, filtrování podle počtu naráz zobrazovaných položek a dle volby porovnávacího operátoru a celočíselné hodnoty. Dále je možné zobrazované položky řadit, a to abecedně nebo dle četnosti výskytu.
- **Manipulovat s dedikovanou nástrojovou lištou**: Nástrojová lišta umožňuje u sloupcových grafů přiblížení a oddalování zobrazení, včetně resetování do původní podoby a další možnosti.
- **Aplikovat globální křížové filtrování (*crossfiltering*)**: Tato funkcionalita synchronizuje vybraný filtr napříč všemi *dashboardy*. Specificky, dle uživatelem zvoleného inventáře. Tímto způsobem může dojít k zefektivnění a zpřesnění datového průzkumu.

Pro detailní informace o aplikaci je doporučeno nahlédnout do projektového *README.md* souboru, v němž je zahrnuta i uživatelská dokumentace.

Architektura. Aplikace ***graphDash*** se skládá celkem ze čtyř zdrojových modulů, které společně definují funkcionalitu aplikace. Kompletní zdrojový kód je referován v rámci Přílohy G.

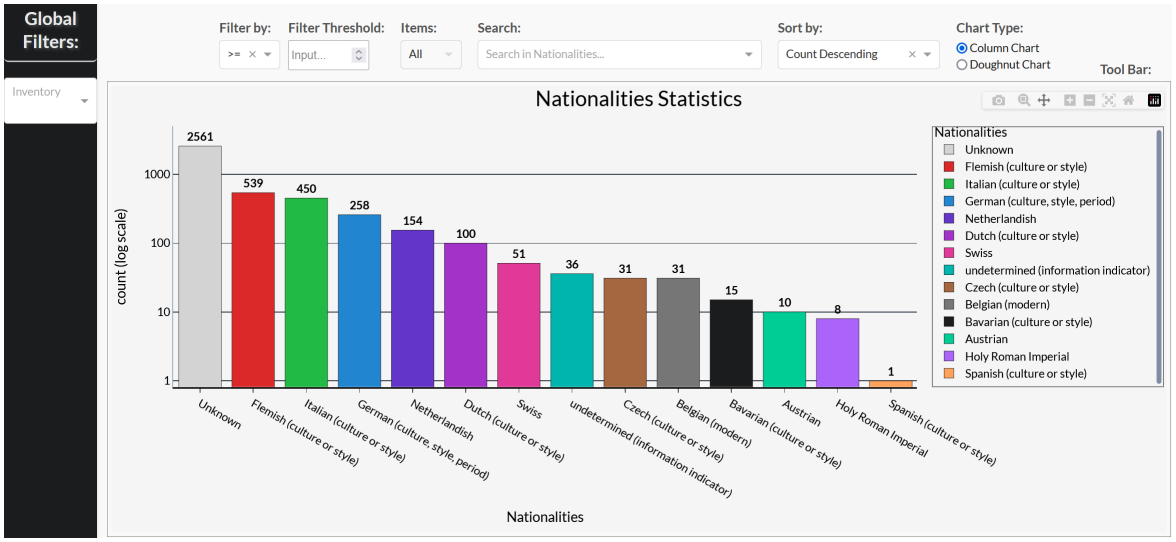
- ***main.py***: Spouštěcí modul, konfiguruje a inicializuje *Dash* aplikaci a definuje její základní *layout*. Zajišťuje obsluhu zpětných volání (*callbacks*).
- ***mongodb_query.py***: Modul pro interakci s databází *MongoDB*, obsahuje funkce pro dotazování (formulaci databázových *queries*). Funkce vracejí data ve dvojdimenzionální struktuře, se kterou aplikace nadále pracuje (*Pandas DataFrames*[52]).
- ***graph_functions.py:*** Modul zodpovědný za přípravu rozvržení aplikace, generování, stylizaci a integraci grafů. Definované funkce zpracovávají data získaná z *MongoDB* a transformují je do grafické podoby.
- ***config.py***: Konfigurační soubor uchovávající všechna potřebná nastavení a konstanty pro správnou funkčnost ostatních modulů.

Vizuální styly aplikace jsou definovány v kaskádových *CSS* souborech ***style.css*** a ***semantic.min.css*** z adresáře *assets*, zajišťující konzistentní a jednotné uživatelské rozhraní napříč

[51] *Pozn.*: Tento *dashboard* je na rozdíl od ostatních dostupný pouze ve sloupcovém provedení a umožňuje přehledně sledovat vývoj inventárních položek v časoprostoru.

[52] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

různými aplikačními komponentami. Druhý zmíněný soubor integruje styly *frameworku Semantic UI*, které jsou také aplikovány ve všech částech uživatelského rozhraní *Inventaria*. Tento přístup zaručuje, že všechny vizualizace mají konzistentní vzhled, který věrně odpovídá stylizaci *Inventaria*. Následuje podrobnější rozbor jednotlivých modulů.



Obrázek 7.18: Ukázka výchozího stavu *dashboardu* graficky znázorňujícího statistiky týkající se národností (*Nationalities*) v logaritmickém měřítku. Zdroj: Vlastní zpracování

Na výše uvedeném obrázku 7.18 je zachyceno ukázkové rozložení *dashboardu* znázorňujícího národnostní zastoupení napříč položkami v *Inventaria*. Je možné si povšimnout i interaktivních ovládacích prvků. Vysouvací seznam v postranním sloupci levé části obrazovky se používá jako globální filtr inventářů (tzn. volba inventáře se propíše do všech *dashboardů*). V rámci již zvoleného *dashboardu* je možné dále filtrovat přes sadu vstupních dialogových oken a vysouvacích seznamů (`Filter by`, `Filter Threshold`, `Items` a `Search`) v horní části obrazovky. Jsou zde umístěny i selektor pro výběr typu řazení (`Sort by`) a přepínání mezi grafovými typy (`Chart Type`). Možnosti pro přibližování, oddalování či posouvaní grafu se ukrývají pod příslušnými ikonami v nástrojové liště, označené jako `Tool Bar`. Legenda se vyskytuje vždy v pravé části obsahové náplně *dashboardu* a umožňuje dodatečnou interakci s grafem v podobě selektivního výběru zobrazených, respektive skrytých, položek.[53]

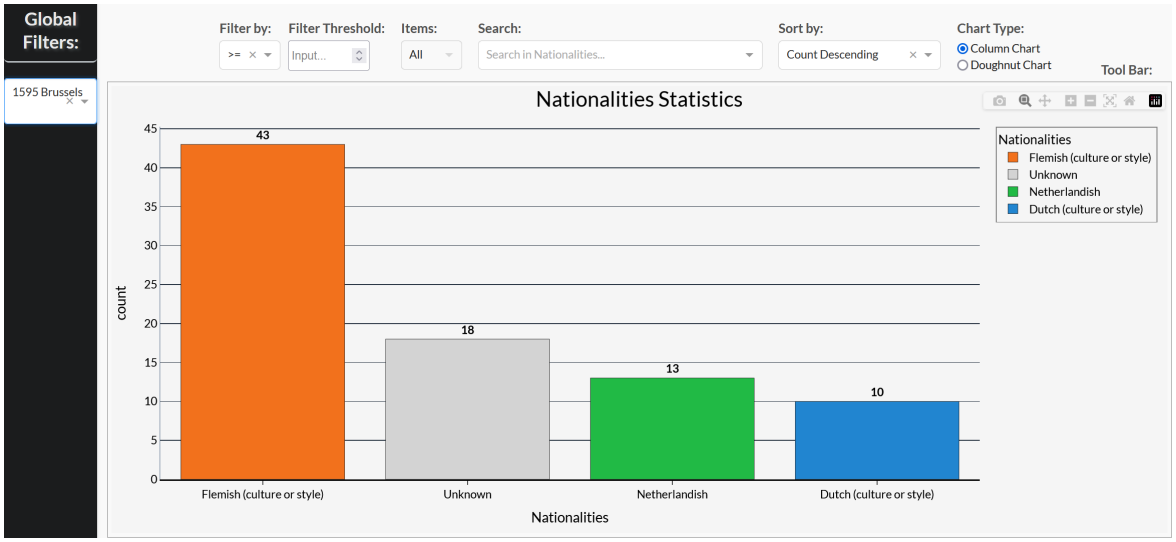
Konfigurace a inicializace. Modul *main.py* tvoří jádro vizualizační aplikace. Zodpovídá za inicializaci aplikace, definici uživatelského rozhraní a správu odpovědí na uživatelské interakce. Orchestruje nejen interaktivitu, ale i vizuální reprezentaci dat. Své fungování zahajuje vytvořením instance *Dash* aplikace, která slouží jako centrální platforma pro všechny další operace. Aplikace je konfigurována s dynamicky generovaným *layoutem*, který je vyhotoven voláním funkce *generate_layout()* importovanou z modulu *graph_functions.py*. Rozvržení definuje strukturu a vizuální styl uživatelského rozhraní.

Zásadní roli v modulu *main.py* sehrají *callback* funkce (tzv. funkce *zpětných volání*) *update_graph()* a *update_dropdown_options()*, které umožňují aplikaci dynamicky reagovat na uživatelem provedené interakce, jako je výběr specifických filtrů, aktualizace možností ve vyhle-

[53] Pro dodatečné podrobnosti viz *README.md* projektu ***graphDash***

dávacích rozbalovacích seznamech a změna počtu, či formátu zobrazovaných dat. Tyto funkce monitorují změny ve stavech interaktivních prvků aplikace a dle potřeby aktualizují vyobrazené informace zavoláním patřičných funkcí. Zpětná volání jsou zásadní pro zajištění, že vizualizace odpovídají aktuálně uživatelem vybraným parametrům. Podstatné je volání funkce *create_figure()* z modulu *graph_functions.py*, která na základě naformátovaných dat z databáze a zjištěné uživatelské konfigurace zformuje korespondující grafy. V závěru skriptu se nachází kód pro spuštění serveru hostující aplikaci. Tento krok je kritický pro zajištění dostupnosti aplikace a je realizován pouze tehdy, když je skript spuštěn jako hlavní program. Funkce tohoto modulu jsou nezbytné pro zajištění správné funkčnosti celé aplikace.

Následuje modul *mongodb_query.py* navržený za účelem interakce s *MongoDB* databází, nezbytný pro extrakci a agregaci dat potřebných pro vizualizační účely aplikace. Tento modul obsahuje definice funkcí pro každý vizualizační scénář, prostřednictvím kterých se získává množina dat reflektující požadované aspekty (např. zvolenou instituci, či národnost). V rámci modulu je zahájeno připojení k lokální instanci databáze ***test_rudolf***, čímž je umožněno zadávání dotazů. Pro každý typ dotazu modul vytváří agregační *pipeline*, která dynamicky zpracovává a filtruje data dle požadavků uživatele. To zahrnuje datové operace jako jsou filtrace (*$match*), projekce (*$project*), rozbalení (*$unwind*) a seskupení (*$group*) dat. Výstup z agregační *pipeline* je před předáním z funkce zkonvertován do formátu *DataFrame*. Jednotlivé funkce podporují podmíněné argumenty (*inventory_label*), což umožňuje použít globální křížový filtr a vhodně *pipeline* rozšířit. Funkce tak mohou získat data filtrovaná dle zvoleného inventáře.



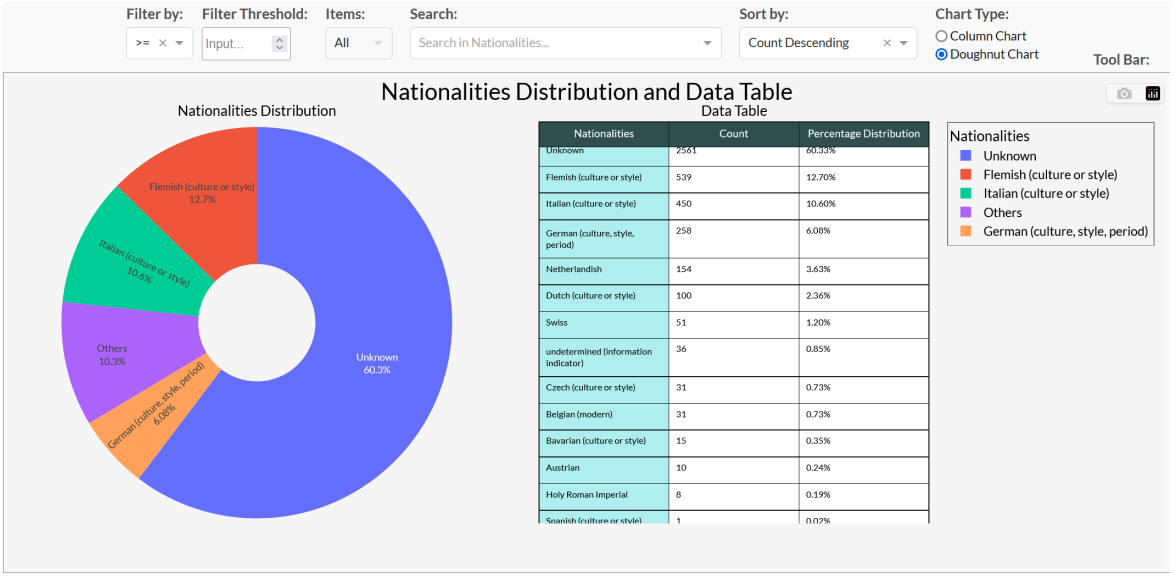
Obrázek 7.19: Ukázka propsání globálního křížového filtru (výběr konkrétního inventáře – *1595 Brussels*) na scénáři *Nationalities*. Zdroj: Vlastní zpracování

Jedná se o kritický prvek celé aplikace, jelikož zajišťuje, že načítaná data jsou korektně zpracována a připravena pro uživatelské vizualizace. Funkcionalita modulu poskytuje datový základ pro explorativní nástroje, které aplikace nabízí.

Soubor *config.py* uchovává pevně stanovené datové sady, konstanty a definice, které jsou používány napříč ostatními moduly. Tento konfigurační soubor zahrnuje definice vizualizačních scénářů a specifikací, které stanovují, jakým způsobem budou data uživatelům prezentována. Definuje, jaké funkce budou použity pro získávání dat u konkrétních scénářů a jak budou tato

data zorganizována ve vizualizačním výstupu. Udržuje vizuálně-funkční konzistenci napříč aplikací. Je zde uchováváno i nastavení pro grafické prvky aplikace, jako jsou barvy, typy popisků a stylizace, které napomáhají udržet estetickou jednotu. Tato nastavení vyjma toho, že zlepšují vizuální stránku aplikace, také redukují opakující se bloky kódu.

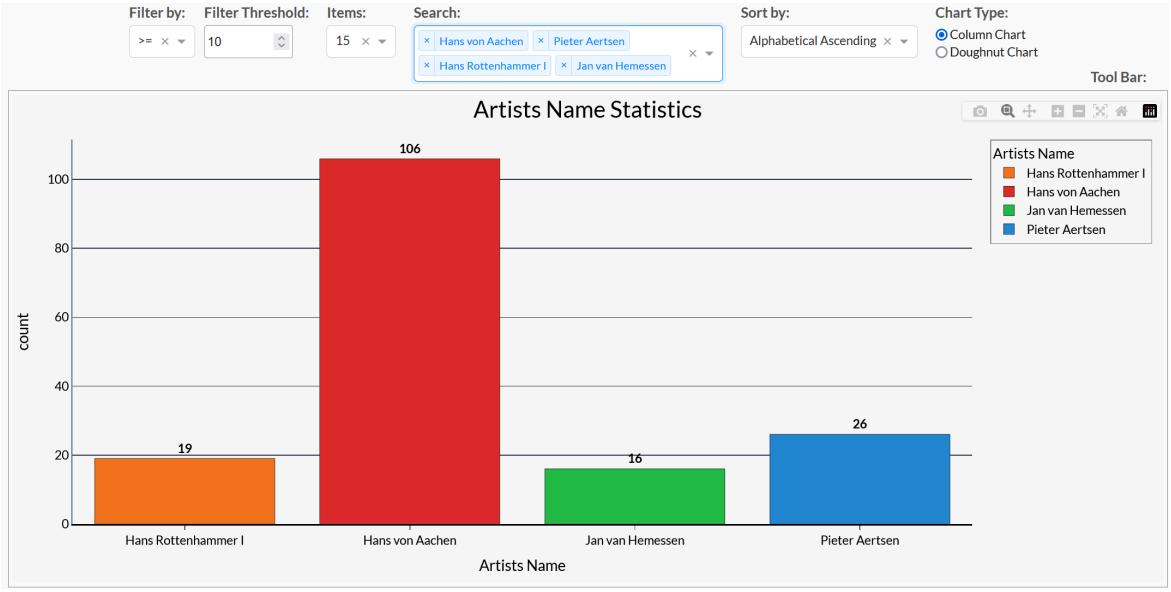
Vizualizační logika. Modul *graph_functions.py* plní funkci technické kostry pro vizualizační aspekty aplikace, protože poskytuje esenciální funkce pro celkové rozvržení aplikace (*generate_layout()*) a tvorbu a úpravu grafů - již avizovanou funkci *create_figure()*. Modul přímo zodpovídá za transformaci načtených datových souborů do podoby sloupcových a prstencových grafů. Dále implementuje filtrování a řazení, umožňující uživatelům částečně si přizpůsobit zobrazení dat. Obsah grafů je generován na základě uživatelské interakce.



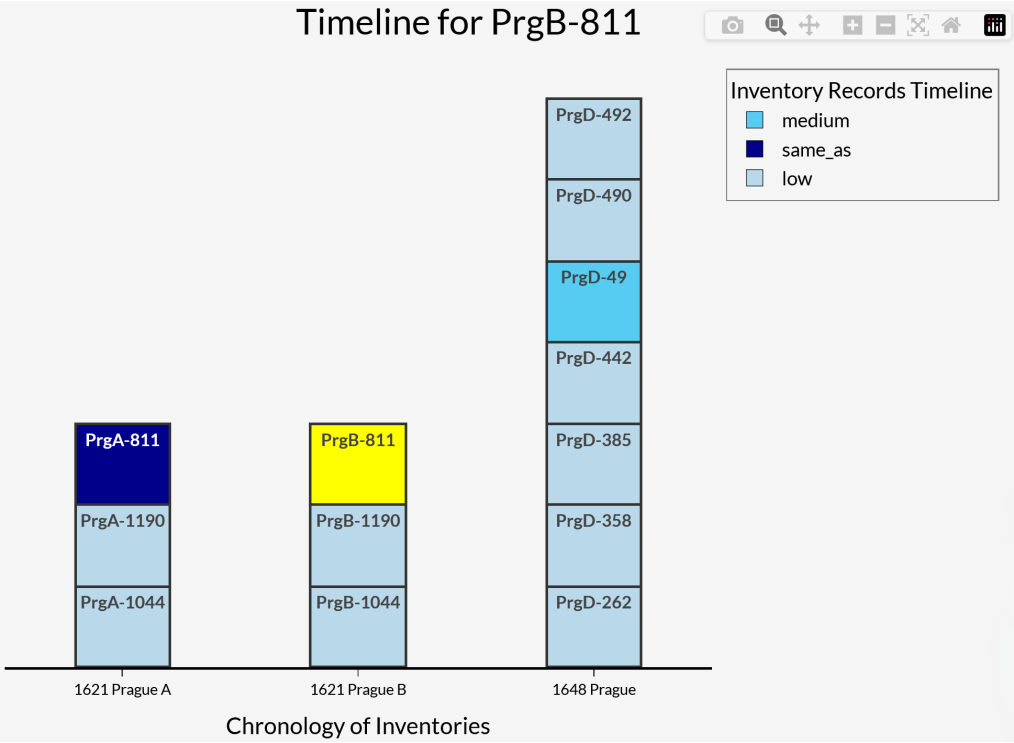
Obrázek 7.20: Ukázka *dashboardu* s prstencovým typem grafu a doprovodnou tabulkou u statistiky *Nationalites*. Zdroj: Vlastní zpracování

Pomocí integrovaných stylovacích funkcí jsou grafy přizpůsobeny tak, aby odpovídaly specifickým potřebám a preferencím uživatelů. Modul zahrnuje rozmanité možnosti nastavení vizuální stránky grafu od barev, až po specifické vlastnosti dané dle zvoleného typu grafu. Pro tyto účely jsou uplatňovány oba výše uvedené *CSS* soubory *style.css* a *semantic.min.css*. Design aplikace reflektuje úsilí o propojení vizuální přívětivosti a analytické funkčnosti, uvažující jak uživatelské preference, tak odborné principy. Rozhodnutí reprodukovat vzhled *Inventaria* bylo učiněno s cílem udržet vizuální konzistenci, jež napomáhá uživateli zorientovat se v datech a zjednodušuje jejich interpretaci. Výběr barevného schématu a grafických prvků byl učiněn tak, aby alespoň částečně respektoval teoretické základy vizuální reprezentace informací.

Kohese mezi teorií a praktickým provedením vzhledu *dashboardů* podstatně zvyšuje srozumitelnost při prezentaci datové sady. Aplikace tak umožňuje rychlejší a přesnější pochopení prezentovaného obsahu i pro ty uživatele, kteří nejsou detailně obeznámeni s obsahem *Inventaria*, případně s metodami datové analýzy. Díky této přístupnosti se aplikace může stát užitečným nástrojem nejen pro výzkumné pracovníky, ale i pro laickou veřejnost.



Obrázek 7.21: Příklad uživatelské interakce s *dashboardem*. Filtrování požadovaného počtu výskytů, vyhledání specifických položek, abecední řazení. Zdroj: Vlastní zpracování



Obrázek 7.22: Ukázka časové osy položky *PrgB-811* (žlutě vyznačené) a stanovených věcných souvislostí dle míry jistoty napříč inventáři. Zdroj: Vlastní zpracování

7.5.2.2 Zhodnocení vizualizace

Framework Dash byl zvolen pro jeho schopnost flexibilně vytvářet komplexní a interaktivní webové aplikace zaměřené na vizualizaci dat. Primárním důvodem volby byla snadná integrace s *Python backendem* a *MongoDB* databází, která zefektivnila vývojový proces díky minimalizaci

potřeby pokročilých znalostí v oblasti *frontendového* vývoje a programovacího jazyka *JavaScript*. [Hue20] Navíc bylo možné využít technologický *stack* z předešlého vývoje ELT pumpy. K interpretaci dat byly zvoleny sloupcové a prstencové grafy, jelikož dokáží způsobile reprezentovat statistické informace. Sloupcové grafy jsou vhodné pro srovnávání absolutních hodnot, zatímco prstencové grafy v kombinaci s doprovodnou tabulkou uceleně a přehledně prezentují procentuální rozložení napříč množinou[54], což je klíčové pro pochopení distribuce zkoumaných položek v inventářích. Stávající provedení (viz snímek 7.17) *dashboard view Inventaria* navíc již tyto dva typy grafů uplatňuje[55], takže by bylo možné na původní funkcionalitu plynule navázat, vhodně ji upravit a rozšířit o dodatečné možnosti. Nicméně, volba tohoto technologického *stacku* čelí řadě omezení, která je třeba zmínit nejen pro úplnost, ale také kvůli pochopení potenciálních výzev.

Závislost na externích knihovnách a *frameworcích* vyžaduje průběžné sledování změn a zabezpečení kompatibility s novými verzemi, což může představovat výzvu v kontextu pravidelnosti aktualizací a údržby nástroje. S rostoucím objemem dat může být také ovlivněn výkon aplikace, zejména při složitějších agregačních dotazech nad databází a následném generování komplexních vizualizací. To v konečném důsledku může vyžadovat dodatečnou práci na optimalizaci výkonu i navzdory tomu, že spojení *MongoDB* a jazyka *Python* umožňuje aplikace škálovat s ohledem na rostoucí datové objemy. [Ohi11]

V průběhu realizace se narazilo na několik technologických omezení, která naznačila potřebu integrace vizualizačního nástroje přímo do stávající webové aplikace *Inventaria*. Týkalo se to především implementace scénáře ***Timeline***. Vylepšení vizualizace časové osy inventárního záznamu by mohlo spočívat v poskytnutí přímého náhledu na jednotlivé věcně související položky. Tento náhled by uživatelům kupříkladu při najetí kurzorem na položku umožnil získat konkrétní detaily, a tím objasnit, proč byla míře podobnosti mezi položkami přiřazena daná hodnota. Tento přístup by mohl významně navýšit uživatelskou přívětivost a přidanou hodnotu vizualizačního scénáře.

Předešlé úpravy datového modelu ze sekce 6.2 spočívaly primárně v nasazení konceptu normalizace a tedy redukci redundancí a duplicit tím, že došlo přesunutí původně zanořených dokumentů do samostatných kolekcí, na něž se z původních zdrojových dokumentů nyní referuje přes unikátní identifikátory, mají zásadní dopad na návrh a implementaci vizualizace. Přestože tento přístup vede k efektivnějšímu ukládání a údržbě dat, může komplikovat formulaci a složitost databázových dotazů. Ve srovnání s původním denormalizovaným modelem jsou vyžadovány složitější dotazy pro rekonstrukci původních vztahů mezi daty. Je to dáno tím, že zavedení datové normalizace přináší nutnost uskutečňovat dotazy, které překračují rámce jednotlivých kolekcí. Tyto operace obvykle zahrnují spojování hned několika kolekcí (jako je tomu například u funkce *get_inventory_details()* z modulu *mongodb_query.py*), což může navýšit čas potřebný pro získání odpovědí z databáze. [Mon24c] S každou úrovní normalizace roste komplexita dotazů a navyšuje se doba výpočtu při provádění dotazů do databáze, protože databázový systém musí provést více operací spojení (*join*) pro agregaci dat umístě-

[54] *Pozn.*: U prstencového grafu jsou popisky u výsečí znázorněny vždy. Je to dáno tím, že minimálně zastoupené položky (tj. položky jejichž zastoupení nepřesahuje 5% v dané množině) jsou agregovány pod hodnotu *Others*

[55] *Pozn.*: Namísto sloupcových grafů se v původní realizaci uplatňují pruhové.

ných v různých zdrojích. Pozorovatelné je to především při snaze integrovat *crossfiltering* do dotazovacích funkcí, což vyžaduje rozšíření původních agregačních *pipeline*. Kupříkladu je-li funkci *get_related_items_data()* předán konkrétní argument *inventory_label* je původní *pipeline* rozšířena o dvě nezbytné operace spojení.

Zvýšená latence u přímého přístupu do databáze je obzvláště znatelná v systémech, které vyžadují *real-time* zpracování velkého množství dat. V tomto případě, kdy se manipuluje s přibližně 15 000 dokumenty, by se mohlo zdát, že uchovávání dat přímo v operační paměti aplikačního serveru je vhodné řešení. Nicméně, uvažování o implementaci mezipaměťové (*cache*) vrstvy by mohlo být efektivnější, zejména s ohledem na to, že například výchozí vizualizační pohledy se opakují a mohly by být vhodně *cachovány*.

Uchovávání databázových záznamů v operační paměti (*RAM*) může značně zrychlit přístup k datům, protože *RAM* je řádově (10^5krát [Gil03]) rychlejší než diskové úložiště. Tato praxe je vhodná u systémů, které vyžadují časté dotazování, jelikož se tím obratně vyhnou neustálým čtecím operacím z pomalých disků. [Mon24a] Ve své dokumentaci [Mon24b] *MongoDB* sice upozorňuje na riziko ztráty dat při výpadku napájení vzhledem k volatilní povaze *RAM*, avšak v praxi toto riziko může být minimální, jelikož v systému *Inventaria* převažují operace čtení. Dále také platí, že udržování velkého množství dat v *RAM* může být finančně náročné vzhledem k vyšší ceně ve srovnání s diskovými úložišti. [Mon24a]

I když proces načítání velkých objemů dat z databáze do operační paměti může být časově náročný, stojí za zmínku, že tento proces je typicky jednorázová záležitost při spuštění serveru, a nikoliv opakovaně se vyskytující problém. [SZT12]. Efektivita přenosu dat může být dále snížena kvůli potřebě transformovat data do požadovaných formátů, aplikací filtrů během načítání, případně validací dat po uskutečněném přenosu, což dále zvyšuje celkovou odezvu. Při manipulaci s výše uvedeným počtem záznamů je vhodné vybalancovat potřebu rychlosti a četnost dotazů vůči potenciálním rizikům a nákladům. V případě zde rozebírané vizualizační aplikace platí, že dotazování probíhá s každou uživatelskou interakcí a změny ve vizualizaci by se měly propisovat ideálně co nejrychleji. U autorem předpřipravených *dashboardových* scénářů se databáze používá pouze ke čtení, nikoliv k zápisu hodnot ze strany uživatele. Odpadá tak nutnost zajistit datovou konzistenci a při výpadku je možné data opětovně z databáze do operační paměti načíst a používat.

Stále však neodpadají výzvy týkající se variability původních dat. Proměnlivá obsáhlost záznamů z *MongoDB* může komplikovat jejich načítání do operační paměti a následné zpracování. Jelikož formát *JSON* umožňuje, že některé atributy nemusí být vždy vyplněny nebo jsou dostupné v různých formátech (vzhledem k manuálnímu původu dat), může tento faktor ztížit proces *serializace* a *deserializace* dat [EH11], klíčový pro efektivní *cachování* dat do *RAM*. [Jan+20]

Strukturální přechod záznamů k normalizované formě může zlepšit organizaci a správu databáze, ale zároveň vyžaduje promyšlené plánování struktury dotazů a optimalizaci databázových indexů, aby bylo možné smysluplně minimalizovat negativní dopady na výkon při dotazování. Výzkumní pracovníci a vývojáři z *AV* by měli tyto aspekty zvážit při budoucím návrhu dodatečných optimalizací databázového modelu, aby vyvážili efektivní správu a rychlost dotazů, to vše při zohlednění vlastních potřeb a požadavků.

Interaktivní průzkum dat nabízí rozsáhlé analytické možnosti, ale může zároveň komplikovat orientaci v aplikaci pro některé uživatele, kteří pak nejsou schopni využít její plný potenciál. Z toho důvodu je potřeba věnovat dodatečnou pozornost návrhu uživatelského rozhraní a poskytnout detailní uživatelskou dokumentaci.

Omezení použitých grafových typů výhradně na sloupcové a prstencové může vést k nedostatečnému vyobrazení jemnějších nuancí v datech. Ačkoliv *Dash* umožňuje relativně jednoduché přizpůsobení uživatelského rozhraní, může být finální vzhled a uživatelská přívětivost omezena základními styly a komponentami, kterými *Dash* disponuje.

Přestože tyto problémy mohou pro projekt představovat potenciální omezení, je možné je řešit důsledným plánováním, průběžnou údržbou a systematickým uživatelským testováním. Hlubší pochopení a aktivní přístup k těmto omezením může projektu přidat na hodnotě a podpořit do budoucna jeho udržitelnost.

7.5.2.3 Shrnutí

V této podkapitole byly zkoumány různé možnosti a koncepty vylepšení vizualizací nad upravenou datovou sadou projektu *Inventaria Rudolphina*. Přechod od původního textově orientovaného zobrazení k interaktivním grafickým *dashboardům* obohacuje způsob prezentace dat. Využitím platformy *Dash* od komunity *Plotly* bylo možné dosáhnout značné flexibility ohledně vizuálního formátu grafů, které nyní zahrnují dynamické zobrazování statistik přímo z databáze a interaktivní možnosti manipulace jako jsou řazení a filtrování. Zároveň byly diskutovány potenciální návrhy na dodatečná vylepšení a optimalizace manipulace s daty.

Dosažené výsledky.

- **Rozšíření funkcionality**: Původní myšlenka *dashboardového* zobrazení byla rozšířena o nové funkce, které umožňují uživatelům provádět podrobnější analýzy na základě interaktivní manipulace s grafy. Nastíněna byla implementace globálních filtrů a to na ukázce výběru konkrétního inventáře. Původní sada interakčních scénářů byla rozšířena o čtyři další, čímž byly uživateli představeny další klíčové statistiky a odhaleny původně ne zcela zřetelné vazby mezi inventárními záznamy. Za zmínku stojí především možnost nahlédnout na věcné souvislosti vybrané položky s jinými inventáři, neboli *dashboard* ***Timeline***.
- **Vylepšení grafického zobrazení**: Využití osvědčených metod uvedených v teoretické sekci 7.4 datové vizualizace se projevuje tím, že informace jsou nyní prezentovány vizuálně atraktivnějším a srozumitelnějším způsobem, což zvyšuje jejich analytický přínos.

Možná vylepšení. V průběhu realizace bylo zjištěno, že platforma *Dash*, navzdory tomu, že poskytuje nezanedbatelné výhody v oblastech flexibility a uživatelské interakce, má také svá specifická omezení, zejména ve zpracování velkých datových objemů, což může vést k pomalejší odezvě aplikace. Tato problematika naznačuje nutnost dalšího zkoumání možností zlepšení výkonu aplikace, zejména ve vztahu k manipulaci s daty a agregačními dotazy do *MongoDB*. Níže uvedený výčet inovací by nejen mohl snížit odezvu aplikace, ale také rozšířit možnosti uživatelské interakce a personalizace *dashboardů*.

- **Uchovávání databázových dat v paměti**: Implementace *in-memory* uchovávání dat z databáze by mohla zvýšit rychlost dotazování a zpracování dat, čímž by se snížila latence a další funkcionální rozšíření se staly rychlejšími a efektivnějšími. Je však žádoucí zohlednit dopad volby formátu, v němž by byla data uložena a opomenout nelze ani proměnlivost obsahu datových záznamů z níž se odvíjí složitost konstrukce dotazů. Dále by bylo potřeba řešit, jakým způsobem a jak často propagovat změny v databázi, aby se v paměti uchovávala pouze aktuální data.
- **Personalizace *dashboardů***: Dalším vylepšením, které by mohlo aplikaci rozšířit, je zavedení funkcionality pro generování *dashboardů* na základě individuálních uživatelských požadavků. Toto rozšíření by uživatelům dovolilo vytvářet vlastní vizualizace dat podle specifických kritérií a preferencí. Došlo by ke zvýšení interaktivity a osobního zapojení uživatelů do procesu datové analýzy.
- **Vylepšení vizuálního podání**: Případné vylepšení grafického podání dat by mohlo spočívat například v použití národních trikolór nebo vlajek přímo ve vizualizacích, namísto barevné palety *Semantic UI*. Byl by tak přidán nový sémantický rozměr a zároveň zvýšena estetická hodnota grafů. Tento přístup by mohl být obzvláště efektivní v případech, kde jsou sloupce grafů rozsáhlé a barevné rozlišení může výrazně zlepšit interpretaci dat.

Aplikace ***graphDash***, pracující s daty *Inventaria Rudolphina,* rozšířila původní textovou reprezentaci dat a úspěšně prokázala, že lze efektivně využít nástroje pro datovou vizualizaci ke zlepšení porozumění a interpretace uměleckohistorických dat. Zároveň prakticky nastínila a ověřila možné funkcionality, o které lze *dashboardové* zobrazení rozšířit. Budoucí vývoj by měl směřovat k dosažení rovnováhy mezi technologickou sofistikovaností a snadností použití, přičemž hlavním cílem zůstává neustálé zlepšování výkonu a rozšiřování funkčnosti *dashboardů* podporujících exploraci zdrojových dat.

Závěr 8

Tato diplomová práce se zaměřila na detailní rozbor a optimalizaci datové vrstvy architektury aplikací *Inventaria* a *Documenta Rudolphina*, které jsou součástí projektu *Umění na odiv*. Práce postupně posuzovala hlavní aspekty datového modelování a vizualizace v kontextu zdrojových uměleckohistorických dat. Na základě analýzy databázových záznamů byly stanoveny klíčové nedostatky a navrženy inovace pro vylepšení datových modelů. Cíle práce, které byly stanoveny na základě počátečního výzkumného záměru prezentovaného v úvodu, byly splněny s několika významnými přínosy, ale i s určitými limitacemi.

Výsledné transformace struktury datových modelů přinesly značné výhody. V případě *Inventaria* se jedná o efektivnější přístup k datům, redukci duplicit a redundancí, zvýšení přehlednosti a zlepšení celkové správy a údržby databáze. Tato transformace byla uskutečněna pomocí procesu *ELT*, jež zajistil požadovanou optimalizaci struktury dat. Transformace datového modelu *Documenta Rudolphina* do databázové podoby vedla k zásadnímu zlepšení struktury a správy dat ve srovnání s původním stavem. Modernizace umožnila efektivnější správu datových zdrojů, což přispělo k lepší integraci a koordinaci mezi jednotlivými databázovými entitami. Tato architektonická změna nejen, že zjednodušuje procesy ukládání a získávání dat, ale také usnadňuje aktualizace a rozšiřování aplikace s ohledem k budoucím výzkumným a vývojovým potřebám. Celkově tato transformace zajišťuje, že databáze bude i s přibývajícím počtem uživatelů nadále schopna efektivně reagovat na požadavky a adaptovat se novým výzkumným směrům.

Důležitou součástí práce bylo také zavedení nového přístupu k vizualizaci dat umístěných v *Inventaria*, a to pomocí platformy *Dash*. Implementace interaktivních grafových *dashboardů* rozšířila původní možnosti uživatelské interakce a datové analýzy, případně alespoň koncepčně nastínila možná vylepšení. Využitím zmíněné vizualizační platformy v kombinaci s teoretickými koncepty bylo dosaženo vyšší uživatelské flexibility při manipulaci s daty a také lepší vizuální interpretace datových statistik, což podporuje porozumění datům.

V počátku této práce bylo cílem poskytnout ucelený pohled na strukturu, fungování a využití aplikace *Inventaria Rudolphina*, čehož bylo úspěšně dosaženo. Původní představy o rozsáhlejší identifikaci nových skrytých vazeb mezi daty nebyly zcela naplněny kvůli povaze dat a omezením v dostupných nástrojích a metodách datové analýzy. Toto omezení bylo částečně kompenzováno rozvojem nových funkcionalit a vizualizačních scénářů, které mohou výzkumníci nově využívat.

Do budoucna se doporučuje zaměřit na rozšíření funkcionality *dashboardů* a prozkoumat

možnosti pro zlepšení výkonu aplikace, zejména v kontextu manipulace s velkými datovými objemy. Zásadní je pokračovat ve vývoji a testování nových vizualizačních nástrojů, což by mělo přispět ke zvýšení výkonu a zlepšení uživatelské interakce. Souběžně by měla proběhnout důkladná analýza metadat inventárních záznamů, což by mohlo poodhalit neobjevené vztahy a dále rozšířit možnosti pro výzkum a interpretaci dat. Tato opatření by pak měla přispět k efektivnějšímu a uživatelsky přívětivějšímu přístupu k datům a současně posílit analytické schopnosti projektu.

Závěrem lze konstatovat, že diplomová práce přináší značné množství inovací a vylepšení v oblasti správy, analýzy a prezentace uměleckohistorických dat. Budoucí vývoj by měl nadále směřovat k posilování technologického zázemí a rozšiřování analytických možností pro akademické i praktické uplatnění.

Struktura zdrojových souborů (*XML* a *JSON*)

A

Koncepty struktura zdrojových souborů *Inventaria* (*XML* a odvozený *JSON*) a vzájemné *namapování XML* elementů (a jejich atributů) na *JSON* klíče:

Struktura zdrojového *XML*.

- Element <item> reprezentuje dané dílo a má atributy *n*, *xml:id*, *ana* a *corresp*, které plní úlohu identifikace a anotace.
- Element <title> reprezentuje název díla.
- Element <seg> obsahuje textové informace a má atributy *ana*, *cert* a *resp*, které jsou určeny k anotaci informace.
- Element <persName> reprezentuje autora díla.
- Element <note> obsahuje poznámku ohledně díla a elementy <ptr> spojují dílo s dalšími díly v inventáři.
- Element <listObject> obsahuje informace o objektu v inventáři, který je asociován s tímto dílem.
- Element <object> reprezentuje daný objekt a disponuje atributem *xml:id*.
- Element <objectIdentifier> obsahuje identifikaci objektu a informace o něm.
- Element <physDesc> obsahuje fyzický popis objektu - tj. materiál, typ a rozměry.
- Element <history> obsahuje informace o historii objektu, jeho původu a provenienci.

Struktura odvozených *JSON* souborů.

Inventární položka:

- *order*: Určuje pořadí díla v rámci inventáře.
- *xml_id*: Identifikuje dílo pomocí *XML ID.*
- *original_inventory_id*: Původní inventární číslo (pořadí) díla.
- *subject*: Seznam klíčových slov, které charakterizují tématiku díla.
- *search_subject*: Seznam klíčových slov, které rovněž popisují náplň díla a mohou být využity při vyhledávání.
- *iconclass_external_id*: Identifikátor *IconClass* pro téma díla.
- *text*: Popisek díla.
- *title*: Název díla.
- *src*: Informace o zdroji a pravosti díla, míra jistoty identifikace záznamu, iniciály zodpovědné výzkumné osoby.
- *note*: Poznámky a přidané informace k dílu.

- *concordances*: Seznam odkazů na věcně související díla z jiných inventářů, míra jistoty konkordance a informace o osobě odpovědné za stanovení těchto vazeb
- *object*: Pole, které není nikdy prázdné. Minimálně je vždy obsažen alespoň výchozí generický záznam skládající se z klíčů *caption, type, digitalisation_layer* a *name*. Ucelený seznam podrobností o díle, zahrnuje:
  - *caption*: Název díla.
  - *type*: Typ díla.
  - *digitalisation_layer*: Indikace, zda byla malba digitalizována.
  - *name*: Jméno autora, včetně informace jako *role*, reference do *Getty ID* a další.
  - *xml_id*: Identifikátor zdrojového *XML* souboru.
  - *relationship_type*: Určení vztahu mezi fyzickým (skutečným) obrazem a zde rozebíraným záznamem.
  - *images*: Odkaz na zdrojový soubor s náhledem na záznam a textový popis.
  - *cert*: Míra jistoty identifikace.
  - *caption*: Název/popisek díla.
  - *object_name*: Název díla.
  - *id_number*: Inventární číslo díla ve spravující instituci.
  - *institution*: Instituce, v němž je fyzické dílo evidováno.
  - *country*: Země, kde se instituce nachází.
  - *city*: Město, kde se instituce nachází.
  - *latitude*: Souřadnice zeměpisné šířky spravující instituce.
  - *longitude*: Souřadnice zeměpisné délky spravující instituce.
  - *physical_description*: Dodatečné informace - údaje o dataci, odkazy na bibliografické zdroje a do další souvisejících institucí.
  - *name*: Role autora, odkaz do odpovídajícího *Getty ULAN* záznamu.
    - Data převzatá z *ULAN* jsou uchována v klíči *getty_data*.
  - *material*: Popis použité umělecké techniky, případně typu podkladu.
  - *type*: Stanovení druhu díla.
  - *dimensions*: Specifikace rozměrů díla (použité jednotky, výška, šířka).
  - *origDate*: Původní datum (rok) vyhotovení díla.
  - *provenance*: Provenience díla (historický vývoj z hlediska vlastnictví).
- *inventory*: Informace o inventáři, v němž je dílo katalogizováno (název, štítek, pořadí).
- *room*: Identifikátor a odkaz na místnost, v němž se obraz dle domněnek výzkumníků nacházel na Pražském hradě (dostupné pouze u záznamů pražských inventářů).
- *place*: Popisek a identifikátor konkrétního umístění v dané místnosti (dostupné pouze u záznamů pražských inventářů).
- *prevItem*: Ukazatel/odkaz na předcházející položku v inventáři, případně seznamu výsledků.
- *nextItem*: Ukazatel/odkaz na následující položku v inventáři, případně seznamu výsledků.

***IconClass*:**

- *txt*: Nabízí krátký popis nebo název tématu v angličtině, zde *„(story of) Flora*", což naznačuje zaměření dokumentu na příběh *Flory*[56].
- *iconclass_external_id*: Identifikátor *IconClass* **96A23** představuje unikátní kód, který se vztahuje k tématu *(story of) Flora* a je spojen s několika klíčovými slovy, jako jsou *„Flora*", *„Primavera*", *„Roman god*", *„spring*"a další, které odkazují na mytologii a historii spojenou s *Florou*.
- Klíčová slova (*kws_all*): Seznam klíčových slov v angličtině, včetně jmen spjatých s *Florou*, a atributů, jako jsou *„fertility*"(plodnost) a *„ancient history*"(starověká historie), které jsou pro kontext tématu relevantní.

Namapování struktur.

1. *item* **XML element:**
    - **JSON klíč:** *order*
    - **JSON hodnota:** Hodnota atributu *n* v XML elementu *item*
2. *xml:id* **XML atribut:**
    - **JSON klíč:** *xml_id*
3. *corresp* **XML atribut:**
    - **JSON klíč:** *iconclass_external_id*
4. *title* **XML element**:
    - **JSON klíč**: *title*
5. *seg* **XML element**:
    - **JSON klíč**: *src*
    - **JSON hodnota**: Pole objektů s následujícími klíči a jejich hodnotami (*value*, *indication*, *cert*, *resp*)
6. *persName* **XML element:**
    - **JSON klíč:** *object*
    - **JSON hodnota:** Pole objektů s následujícími klíči a hodnotami:
        - *value*: Textová hodnota obsažená v XML elementu *persName*
        - *role*: Hodnota atributu *role* v XML elementu *persName*
        - *getty_external_id*: Hodnota atributu *corresp* v XML elementu *persName*
7. *note* **XML element**:
    - **JSON klíč**: *note*
8. *listObject* **a** *object* **XML elementy:**
    - **JSON klíč:** *object*
    - **JSON hodnota:** Kompletní pole objektů (23 položek) - např.:
        - *caption*: Text obsažený v zanořeném XML elementu *objectName*
        - *type*: Text obsažený v zanořeném XML elementu *objectType*

[56] *Flora* - římská bohyně jara a květin

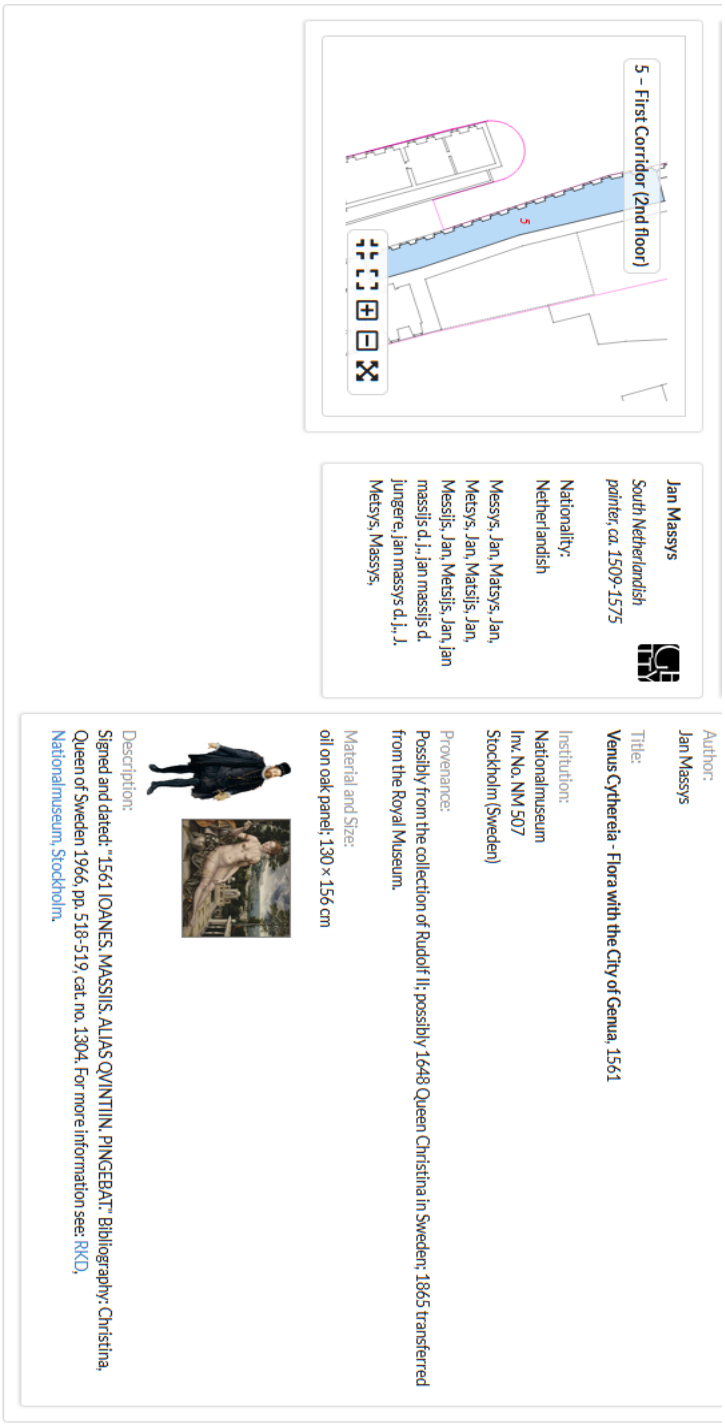
9. *objectIdentifier* **XML element**:
    - **JSON klíč**: *object*
    - **JSON hodnota**: Pole objektů s následujícím klíčem a hodnotou:
        - *caption*: Text obsažený v XML elementu *objectIdentifier*
10. *physDesc* **XML element**:
    - **JSON klíč**: *object*
    - **JSON hodnota**: Pole objektů s následujícími klíči a hodnotami:
        - *physical_description*: Text obsažený v XML elementu *p* a *ref*
        - *material:* Text obsažený v XML elementu *material*
        - *type:* Text obsažený v XML elementu *objectType*
        - *dimensions:* Hodnoty obsažené v XML elementu *dimensions*
11. *history* **XML element**:
    - **JSON klíč**: *object*
    - **JSON hodnota**: Pole objektů s následujícími klíči a hodnotami:
        - *originDate:* Hodnota obsažená v zanořeném XML elementu *originDate* elementu *Origin*
        - *provenance:* Hodnota obsažená v zanořeném XML elementu *provenance*

Uživatelské rozhraní *Inventaria*

B

Obrázek B.1: Domovská stránka webové aplikace *Inventaria Rudolphina*. [Úst20c]

Obrázek B.2: Inventární záznam *PrgC-1* (*Venus Cythereia - Flora with the City of Genua* od *Jana Massyse*). [Úst20c]



Obrázek B.3: Inventární záznam *PrgC-1* (*Venus Cythereia - Flora with the City of Genua* od *Jana Massyse*) - dodatečné informace. [Úst20c]

Odkaz na složku obsahující dodatečné *screenshoty* z prostředí webové aplikace *Inventaria Rudolphina* a detailu položky *PrgC-1*: https://1drv.ms/f/s!AhxWhpDaXdqMloVtypBND8eIaa2i-w?e=9Cv1Sm.

Zdrojové kódy položky *PrgC-1*

C

Zdrojové kódy položky *PrgC-1* (*Jan Massys - Venus Cythereia - Flora with the City of Genua*[57]) ve formátech *JSON* a *XML*:

JSON.

```
{
    order: 1,
    xml_id: "PrgC-1",
    original_inventory_id: 1,
    subject: [
        "erotic",
        "Genua",
        "city",
        "nude",
        "flower",
        "flowers"
    ],
    search_subject: [
        "erotic",
        "Genua",
        "city",
        "nude",
        "flower",
        "flowers",
        "Auxo",
        "Carpo",
        "Chloris",
        "Flora",
        "Primavera",
        "Roman god",
        "Thallo",
        "Ver",
        "ancient history",
        "classical antiquity",
        "fertility",
        "goddess",
        "gods",
        "history",
        "mythology",
        "spring"
    ],
    iconclass_external_id: "96A23",
    text: "Dea Flora, ligt ganz nackent, vom Joann Massi. ",
    title: "Dea Flora, ligt ganz nackent",
```

[57] http://147.228.173.159/item/PrgC-1

```
    src: {
        value: "original",
        indication: [
            "vom"
        ],
        cert: "high",
        resp: "#MJ"
    },
    note: "1 stuckh. Ob identisch mit Dudik A 410 (S. XXXIX): Eine Flora?",
    concordances: [
        {
            id: "PrgD-410",
            cert: "high",
            resp: "HZ"
        },
        {
            id: "Stck-54",
            cert: "high",
            resp: "MJ"
        }
    ],
    object: [
        {
            caption: "Venus Cythereia - Flora with the City of Genua",
            type: "painting",
            digitalisation_layer: true,
            name: [
                {
                    value: "Joann Massi",
                    role: "artist",
                    getty_external_id: "500011809",
                    getty_data: {
                        getty_external_id: "500011809",
                        name: "Massys, Jan",
                        description: "South Netherlandish painter, ca. 1509-1575",
                        gender: "male",
                        est_start: "1500",
                        est_end: "1575",
                        birth_place_name: "Antwerpen",
                        death_place_name: "Antwerpen",
                        display_name: "Jan Massys",
                        altname: [
                            "Messys, Jan",
                            "Matsys, Jan",
                            "Metsys, Jan",
                            "Matsijs, Jan",
                            "Messijs, Jan",
                            "Metsijs, Jan",
                            "jan massijs d. j.",
                            "jan massijs d. jungere",
                            "jan massys d. j.",
                            "J. Metsys",
                            "Massys"
                        ],
                        note: "Comment on works: Religious; History",
                        nationality: "Netherlandish"
                    }
                }
            ]
        },
        {
            xml_id: "alt-1-PrgC-1",
            relationship_type: "identification",
            images: [
```

```
            {
                file: "PrgC-1_PrgD-410.jpg",
                text: "Jan Massys, Venus Cythereia - Flora with the City of Genua, 1561, Nationalmuseum, Stockholm"
            }
        ],
        cert: "high",
        caption: "Venus Cythereia - Flora with the City of Genua",
        object_name: "Venus Cythereia - Flora with the City of Genua",
        id_number: "NM 507",
        institution: "Nationalmuseum",
        country: "Sweden",
        city: "Stockholm",
        latitude: 59.328611,
        longitude: " 18.078333",
        physical_description: "Signed and dated: \"1561 IOANES. MASSIIS. ALIAS QVINTIIN. PINGEBAT.\" Bibliography: Christina, Queen of Sweden 1966, pp. 518-519, cat. no. 1304. For more information see: <a href=\"https://rkd.nl/explore/images/51705\" target=\"_blank\">RKD</a>, <a href=\"http%3A%2F%2Femp-web-84.zetcom.ch%2FeMP%2FeMuseumPlus%3Fservice%3DExternalInterface%26module%3Dcollection%26objectId%3D17510%26viewType%3DdetailView\" target=\"_blank\">Nationalmuseum, Stockholm</a>.",
        name: [
            {
                role: "artist",
                getty_external_id: "500011809",
                getty_data: {
                    getty_external_id: "500011809",
                    name: "Massys, Jan",
                    description: "South Netherlandish painter, ca. 1509-1575",
                    gender: "male",
                    est_start: "1500",
                    est_end: "1575",
                    birth_place_name: "Antwerpen",
                    death_place_name: "Antwerpen",
                    display_name: "Jan Massys",
                    altname: [
                        "Messys, Jan",
                        "Matsys, Jan",
                        "Metsys, Jan",
                        "Matsijs, Jan",
                        "Messijs, Jan",
                        "Metsijs, Jan",
                        "jan massijs d. j.",
                        "jan massijs d. jungere",
                        "jan massys d. j.",
                        "J. Metsys",
                        "Massys"
                    ],
                    note: "Comment on works: Religious; History",
                    nationality: "Netherlandish"
                }
            }
        ],
        material: "oil on oak panel",
        type: "painting",
        dimensions: {
            unit: "cm",
            height: 130,
            width: 156
        },
        origDate: "1561",
        provenance: "Possibly from the collection of Rudolf II; possibly 1648 Queen Christina in Sweden; 1865 transferred from the Royal Museum."
```

```
        }
    ],
    inventory: {
        name: "1635_Prague",
        label: "1635 Prague",
        order: 11
    },
    room: {
        original_description: "Im forderisten gang",
        id: 5
    },
    place: {
        original_description: "oben aufm gesimbs",
        id: 3
    },
    prevItem: false,
    nextItem: "PrgC-2"

}
```

XML.

```
<item n="1" xml:id="PrgC-1" ana="#th.erotic #th.Genua #th.city #th.nude #th.flower #th
    .flowers" corresp="Iconclass 96A23">
  <title>Dea Floraundefined ligt ganz
                                          nackent</title>
  undefined
  <seg ana="#src.original" cert="high" resp="#MJ">vom</seg>
  <persName role="artist" corresp="http://vocab.getty.edu/page/ulan/500011809">Joann
    Massi</persName>
  .
  <seg ana="#src.original" cert="high" resp="#MJ"/>
  <note>1 stuckh. Ob identisch mit
                                          Dudik A 410 (S. XXXIX): Eine Flora?</note>
  <ptr sameAs="PrgD-410" cert="high" resp="HZ"/>
  <ptr sameAs="Stck-54" cert="high" resp="MJ"/>
  <listObject>
    <object xml:id="alt-1-PrgC-1">
      <objectIdentifier cert="high" resp="EW">
        <objectName>Venus Cythereia - Flora with the City
                                                of Genua</objectName>
        <idno>NM 507</idno>
        <institution>Nationalmuseum</institution>
        <address>
          <country>Sweden</country>
          <settlement>Stockholm</settlement>
          <location>
            <geo>59.328611undefined 18.078333</geo>
          </location>
        </address>
      </objectIdentifier>
      <physDesc>
        <p>
          <persName role="artist" corresp="http://vocab.getty.edu/page/ulan/500011809"
    />
          Signed and dated: "1561 IOANES. MASSIIS. ALIAS
                                                QVINTIIN. PINGEBAT." Bibliography:
    Christinaundefined
                                                Queen of Sweden 1966undefined pp.
    518-519undefined cat. no. 1304.
                                                For more information see:
          <ref target="https://rkd.nl/explore/images/51705">RKD</ref>
          undefined
          <ref target="http%3A%2F%2Femp-web-84.zetcom.ch%2FeMP%2FeMuseumPlus%3Fservice
```

```
    %3DExternalInterface%26module%3Dcollection%26objectId%3D17510%26viewType%3
    DdetailView">Nationalmuseumundefined Stockholm</ref>
38          .
39          <material>oil on
40                                                  oak
41                                                  panel</material>
42          <objectType>painting</objectType>
43          <dimensions>
44            <height unit="cm">130</height>
45            <width unit="cm">156</width>
46          </dimensions>
47        </p>
48      </physDesc>
49      <history>
50        <origin>
51          <origDate when="1561">1561</origDate>
52        </origin>
53        <provenance>Possibly from the collection of Rudolf
54                                                  II; possibly 1648 Queen Christina in
     Sweden; 1865
55                                                  transferred from the Royal Museum.</
    provenance>
56      </history>
57    </object>
58  </listObject>
59 </item>
```

Zdrojové *XML* z *Documenta*

D

Archiv.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="../view_archive.xsl" type="text/xsl"?>
<dr:doc xmlns:dr="http://documenta.rudolphina.org/"
 xmlns="http://www.w3.org/1999/xhtml" uri="A_2_Moravsky_zemsky_archiv" v="2007-04-21">
 <title>Moravský zemský archív</title>
 <div class="path">
 <a href="../Archiv/A_0_Archive.xml">Archive</a> &gt;
 <a href="../Archiv/A_1_Archive_Brno.xml">Brno</a> &gt;
 <span>Moravský zemský archív</span>
 </div>
 <ul>
 <a href="../Archiv/A_3_MZA_RAD_Inv_1898_46.xml">RAD Inv. 1898-46 [Karton 421]</a>
 </ul>
 </dr:doc>
```

Indices.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="../view_literatur.xsl" type="text/xsl"?>
<body xmlns="http://www.w3.org/1999/xhtml" uri="Vignau-Wilberg_1992">
Thea Vignau-Wilberg, Joris Hoefnagel, the illuminator, p.15 - 28; The constructed
    alphabet, p. 313 - 409. In:
Lee Hendrix, Thea Vignau-Wilberg,
Mira calligraphiae monumenta: a sixteenth-century calligraphic manuscript inscribed by
     Georg
Bocskay and illuminated by Joris Hoefnagel,
The J. Paul Getty Museum 1992.
<a style="background: white; display: inline-block;" href="http://books.google.at/
    books?id=drlDAgAAQBAJ&amp;printsec=frontcover&amp;redir_esc=y#v=thumbnail&amp;q&
    amp;f=true">Online</a>
</body>
```

Namen.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="../view_names.xsl" type="text/xsl"?>
<dr:doc xmlns="http://www.w3.org/1999/xhtml" xmlns:dr="http://documenta.rudolphina.org
    /" v="2012-12-11">
   <title>Alexandrin, Julius</title>
   <h3>Dr. Julius Alexandrin von Neuenstein</h3>
   +1590;
   <br/>Leibmedicus
   <div class="docindex"><b>Quellen:</b><a href="../Regesten/A1565-02-00-02874.xml"
    title="Eingangsprotokoll der Hofkammer">1565-02</a></div>
   Gedruckte Werke:
   <ul>
   Julius Alessandrini a Neustain, Antargenterica pro Galeno (Venetiis 1552). [ÖNB BE
    .9.N.65.(3).]
```

```
12    Julius Alessandrini a Neustain, Antargentericorum suorum defensio adversus Galeni
       calumniatores (Venetiis, Bologninus Zalterius 1564). [ÖNB 70.E.51.]
13    Julius Alessandrini a Neustain, Galeni enantiomaton aliquot liber. Ejusdem, Galeni
       encomium (Venetiis, Junta 1548). [ÖNB 68.K.31.]
14    Julius Alessandrini a Neustain, Paedotrophia, sive de Puerorum educatione, liber ab
        auctore recognitus. Eiusdem Carmina aliquot alia (Tridenti, Joan. Baptista et
       Jacobus Fratres de Gelminis de Sabbio 1586). [ÖNB *38.Y.74.]
15    Giulio Alessandrini, Antargentericorum suorum Defensio, adversus Galeni
       Calumniatores (Viennae Austriae, Michael Zymmermann 1558). [ÖNB 36.C.82.]
16    Giulio Alessandrini, De medicina et medico dialogus, libris quinque distinctus (
       Tiguri, Andreas Gesnerus 1557). [ÖNB *69.H.49.]
17    Giulio Alessandrini, Galeni praecipua scripta Annotationes, quae Commentariorum
       loco esse possunt. Accessit trita illa de Theriaca Quaestio (Basileae, Petrus
       Perna 1581). [ÖNB 67.E.6.]
18    Giulio Alessandrini, Salubrium, sive de sanitate tuenda libri trigintatres (
       Coloniae Agrippinae, Gervinum Calenium &amp;amp; haeredes Quentelios 1575). [ÖNB 69.C
       .29.]
19    </ul>
20    <dr:ix> IndexA Al Ale Alex Alexa Alexan</dr:ix>
21 </dr:doc>
```

Regesten.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <?xml-stylesheet href="../view_regest.xsl" type="text/xsl"?>
3  <dr:doc xmlns="http://www.w3.org/1999/xhtml" xmlns:dr="http://documenta.rudolphina.org
       /" v="2007-08-26" uri="A1616-02-05-02723" a="MSt">
4  <title>Kaiser Matthias an die Böhmische Kammer</title>
5  .pi Matthias I.+Matthias+
6  .pi Jacob Hoefnagel+Jacoben Hufnagel+Jacoben Hueffnahls+Huefnagl+
7  .pi Melchior Wahl+M Wahl+
8  .pi Wolf Märl+h: Märl+
9  Kaiser Matthias an die Böhmische Kammer
10 1616 Februar 5, Prag
11 <div class="namindex">
12 <a href="../Namen/Matthias_I.xml" onmouseover="highlightWords(event, 'Matthias')">
       Matthias I.</a>
13 <a href="../Namen/Hoefnagel_Jacob.xml" onmouseover="highlightWords(event, 'Jacoben
       Hueffnahls', 'Jacoben Hufnagel', 'Huefnagl')">Jacob Hoefnagel</a>
14 <a href="../Namen/Wahl_Melchior.xml" onmouseover="highlightWords(event, 'M Wahl')">
       Melchior Wahl</a>
15 <a href="../Namen/Marl_Wolf.xml" onmouseover="highlightWords(event, 'h: Märl')">Wolf M
       ärl</a>
16 </div>
17 <div class="zitat">
18    [Betreff:] An die Bähemische Camer waßmassen Sy w: Kaiser Rudolffi gewesten Camer
19    Miniatur mahlern Jacoben Hufnagel 1500 ausstendige besoldung vnd arbeiter lohn über
20    den gethannen nachlaß, auß den Ihnen vndergebenen Camer geföllen gegen deren zu
21    hindan fertigung des alt kaiserlichen hofgesindts verordnet[en] Commihzarien quittu
       [ng]
22    entrichten vnd bezahlen lassen solle [Ablage:] B [Gezeichnet:] M Wahl
23    [Datum:] 5. February Ao 616.
24       Matthias etc.
25    Demnach Wir auf der vorigen in Gott ruhenden Kay: Matt. etc. seeligster gedächtnus,
26    gewesten Camer Miniatur mahlers, Jacoben Hueffnahls vnterthenigstes anlanngen vnd
27    bitten, gn[]ist bewilligt haben, dz Ihme sein hinterstellig v[er]bliebene alte
       besoldung, vnd dan was Ihme für allerhandt damaln gemachte arbeit ausstendig, vnd
       vber daron gethanen
28    nachloß 1500 fl bringt auß vnsern Behmisch[en] Camer geföllen, entricht vnd bezahlt
29    werden sollen.
30       Alß befehlen wir solchemnach Euch hiemit gnedigclich, Ihr wollet bey vnnserm
       Euch
31    vntergebenen Rentambt dj weittere v[er]ordtnung thuen, damit Ihme Huefnagl berüerte
32    1500 fl dannenhero gegen vnserer zu hindanfertigung des allt kayserlich[en]
       hofgesindts
```

```
33    geordtnetten Com[m]issaris q[ui]ttung, ehister müglicheit nach, abgestatt vnd
       bezahlt
34    werd[en], wie Ihr geh[] zuthuen wüsst, daran etc.  Geben Prag d[en] 5 Febr[] Ao etc
       . 616 HP
35      An dj Behmische Camer h: Märl
36    [Conc. Pap.]
37 </div>
38 .i1 #APH, Dvorská komora, Karton 6, Nr. 750 [ehem. Fasz. 15782].
39 <div class="archiv"><a href="../Archiv/A_1_Archive_Praha.xml">Praha</a>, <a href="../
       Archiv/A_2_Archiv_Prazskeho_hradu.xml">Archiv Pražského hradu</a>, <a href="../
       Archiv/A_3_APH_Dvorska_komora.xml">Dvorská komora</a>, <a href="../Archiv/
       A_4_APH_Dvorska_komora_Karton_6.xml">Karton 6</a>, Nr. 750 [ehem. Fasz. 15782].
40 </div>
41 ehemals #HKA, Böhmen, Fasz. 15782.
42 Zitiert in: Kreyczi 1894, JKSAK 15, p.XLV, Reg. 11787. [zitiert das Orginal]
43 Copyright © 2006 Manfred Staudinger
44 </dr:doc>
```

Zdrojový kód *IconClass* položky *PrgC-1*

E

Ukázka původního zdrojového *JSON* souboru obsahující informace převzaté z klasifikačního systému *IconClass* spjatých s dílem *PrgC-1* z *Inventaria Rudolphina*.

JSON.

```
{
    txt: {
        en: "(story of) Flora"
    },
    kws_all: {
        en: [
            "Auxo",
            "Carpo",
            "Chloris",
            "Flora",
            "Primavera",
            "Roman god",
            "Thallo",
            "Ver",
            "ancient history",
            "classical antiquity",
            "fertility",
            "goddess",
            "gods",
            "history",
            "mythology",
            "spring"
        ]
    },
    iconclass_external_id: "96A23"
}
```

Dotazy do databáze *Documenta* a transformační skript

F

Ukázkové dotazy realizované nad prototypovou databází reprezentující transformované kolekce *Documenta Rudolphina*. Zdrojové soubory, viz elektronické přílohy - adresář: **tgi_sp**, podadresáře: **\backend\convert**, **\backend\model** a **\backend\scripts** (pro dílčí kolekce ***Person***, ***Document*** a ***ArchiveLinks***).

```
#Query1: Osoba a vsechny jeji dila

[
  {
    '$lookup': {
      'from': 'Artwork',
      'localField': 'references_list.uri',
      'foreignField': 'uri',
      'as': 'person_artworks'
    }
  }, {
    '$project': {
      'full_name': 1,
      'person_artworks.title': 1,
      'person_artworks.text': 1,
      'person_artworks.date': 1
    }
  }
]

#Query2: Dilo a info o~osobach, jez v~nem konfiguruji

[
  {
    '$lookup': {
      'from': 'Person',
      'localField': 'uri',
      'foreignField': 'references_list.uri',
      'as': 'artwork_persons'
    }
  }, {
    '$project': {
      'uri': 1,
      'title': 1,
      'text': 1,
      'artwork_persons': 1
    }
  }
]
```

```
#Query3: Vypis archivu Praha - Karton_4

[
  {
    '$lookup': {
      'from': 'Artwork',
      'as': 'ArtworksInArchive_A4',
      'pipeline': [
        {
          '$match': {
            'archives.A_4': {
              '$regex': '.*Karton_4'
            }
          }
        }, {
          '$project': {
            'uri': 1,
            'title': 1,
            'text': 1,
            'publication_date': 1
          }
        }
      ]
    }
  }, {
    '$project': {
      'name': 1,
      'ArtworksInArchive_A4': 1
    }
  }
]
```

Obrázek F.1: Ukázková trojice dotazů do prototypové *MongoDB* databáze *Documenta*. Zdroj: Vlastní zpracování

Zdrojové kódy aplikací *rudolfTransformation a graphDash*

G

Aplikace *rudolfTransformation* - kompletní zdrojové kódy *ELT* procesu v jazyce *Python*, včetně uživatelské příručky/dokumentace lze nalézt v elektronických přílohách. Umístěny jsou v podadresáři ./**rudolfTransformation**, který lze najít v hlavním adresáři komprimovaného souboru s přílohami. Po úspěšné extrakci lze v daném inventáři dohledat požadované moduly, metadata a *README*.

Aplikace *graphDash* - kompletní zdrojové kódy vizualizační aplikace vystavěné na platformě *Dash*, včetně uživatelské příručky a dokumentace lze také nalézt v elektronických přílohách. Viz podadresář: ./**graphDash**, který lze najít v hlavním adresáři komprimovaného souboru s přílohami. Po úspěšné extrakci lze v daném inventáři dohledat požadované moduly, metadata a *README*.

Upravené struktury databázových dokumentů *Inventaria*

Upravená struktura inventární položky (**záznam z kolekce *inventory***):

```
{
_id: ObjectId('634310e3b80cc979d90b08e7')
"original_xml": "<item n="3" xml:id="BxlA—3" ... "
"order": 5,
"xml_id": "PrgA—815",
"original_inventory_id": 3,
"subject":  [
"erotic",
"bath"
],
"search_subject": [
"erotic",
"bath"
],
"concordances": [
"63cc6c4ac070863a95aae1a2"
],
"text": "Ein Baad von Josep Arpinas. Orig.",
"title": "Ein Baad",
"src": {
"value": "original",
"indication": "Orig.",
"cert": "high",
"resp": "#MJ"
},
"inventory": "inventory_id_1621_Prague_A",
"access": ... Array (3) ...,
"room": {
original_description: "Folgen ferner die Gemähl, welche in den fördern gang sein.",
  id: 5
},
"place": {
original_description: "oben auf dem gesims",
id: 3
},
"prevItem": ObjectID('634310e3b80cc979d90b08e6'),
"nextItem": ObjectID('634310e3b80cc979d90b08e8'),
"generic_object": {
"caption": "Susanna at the Elders",
"type": "painting",
"digitalisation_layer": true,
      "name": [
        {
```

```
        "value": "Josep Arpinas",
        "role": "artist",
    "ulan_reference": "6197ce9c329ac9e33225bf14"
        }
      ]
  },
"object_ids": [
"object_id_1",
"object_id_2",
"object_id_3"
]
}
```

Struktura dokumentu referovaných umělců (**záznam z kolekce *ulan***):

```
[
 {
  "_id": {
    "$oid": "6197ce9c329ac9e33225bf14"
  },
  "getty_external_id": "500115051",
  "name": "Cesari, Giuseppe",
  "patron": [
    "Orsini, Corradino",
    "Orsini family",
    "Sannesio family"
  ],
  "description": "Italian painter, 1568–1640",
  "gender": "male",
  "est_start": "1568",
  "est_end": "1640",
  "birth_place_name": "Arpino",
  "death_place_name": "Roma",
  "display_name": "Giuseppe Cesari",
  "altname": [
  ... 191 Items ...
  ],
  "note": "Artist specializing in decorative cycles (especially fresco painting) ... .
    ",
  "nationality": "Italian (culture or style)"
 }
]
```

Struktura dokumentu původního **prvku z pole *object***:

```
 [
    {
      "_id": ObjectId('634310e3b80cc979d90b08e7'),
      "xml_id": "alt-1-PrgA-B-815",
      "relationship_type": "identification",
      "iconclass_external_id": "71P4121",
      "subject": [
            "erotic",
            "bath"
      ],
      "images": [
        {
          "file": "alt-1-PrgA-B-815.jpg",
          "text": "Giuseppe Cesari (Cavaliere D'Arpino), Susanna at the Elders, ca.
    1607, Pinacoteca Nazionale, Siena"
        } (Alternativně: "images": [ObjectId ('image_id_1')], )
      ],
      "cert": "medium",
      "caption": "Susanna at the Elders",
```

```
      "object_name": "Susanna at the Elders",
      "id_number": "515",
      "institution": "Pinacoteca Nazionale",
      "country": "Italy",
      "city": "Siena",
      "latitude": 43.3157,
      "longitude": 11.3305,
      "physical_description": "D'Arpino made several versions of ...",
      "name": [
    {
 "role": "artist",
       "ulan_reference": "6197ce9c329ac9e33225bf14"
    }
      ],
      "dimensions": {
        "unit": "cm",
        "height": 52.8,
        "width": 37.1
      },
      "material": "oil on panel",
      "type": "painting",
      "origDate": "ca. 1607",
      "provenance": "Gift of Spanocchi."
    }
]
```

Elektronické přílohy

Elektronické přílohy se skládají z podadresářů a souborů, které jsou popsány v textovém souboru *Readme.txt*. Následuje detailní popis jednotlivých komponent.

Aplikace a knihovny

graphDash (Inventaria Rudolphina)

Adresář *graphDash* obsahuje zdrojový kód (*.py* moduly) stejnojmenné vizualizační aplikace. Dále zahrnuje:

- **assets** - Adresář s použitými vizuálními prvky a *screenshoty* aplikace.
- **LICENCE.txt** - Licenční ujednání.
- **README.md** - Uživatelská dokumentace k vizualizační aplikaci.
- **requirements.txt** - Seznam požadovaných knihoven a balíků pro zajištění funkcionality aplikace.

rudolfTransformation (Inventaria Rudolphina)

Adresář *rudolfTransformation* obsahuje zdrojový kód (*.py* moduly) transformačního skriptu v kombinaci s následujícími soubory:

- **LICENCE.txt** - Licenční ujednání.
- **README.md** - Uživatelská dokumentace popisující transformační skript.
- **requirements.txt** - Seznam požadovaných knihoven a balíků pro zajištění funkcionality skriptu.

tgi_sp (Documenta Rudolphina)

Zahrnuje podadresáře *docs* (projektová dokumentace), *frontend* (*UI*) a *backend*. Pro účely transformace je nosný *backend*. Obsahuje zdrojové kódy skriptu pro transformaci původních *XML* souborů *Documenta* do *JSON* dokumentů a jejich nahrání do *MongoDB* databáze. Struktura adresáře dále zahrnuje následující podadresáře a soubory:

- **convert** - Moduly s funkcemi pro konverzi *XML* dokumentů do *JSON*.

- **database** - Inicializace *MongoDB* databáze.
- **Dockerfile** - Konfigurační soubor pro vytvoření *Docker image*[58].
- **model** - Definice objektů pro nově transformované kolekce dokumentů.
- **scripts** - Zdrojové soubory pro konverzi dokumentů.
- **test_data** - Ukázky původních *XML* souborů a transformované *JSON* dokumenty (viz sekce *Vstupní data* a *Výsledky* níže).
- **README.md** - Uživatelská dokumentace/manuál.
- **requirements.txt** - Seznam požadovaných knihoven a balíků.

Poster

Adresář obsahuje veškeré soubory týkající se *posteru*.

Text práce

Kompletní zdrojové soubory a znění diplomové práce.

Vstupní data

Vstupní data jsou umístěna v podadresáři *test_data* a zahrnují:

- **Archiv** - Vstupní data odpovídající *XML* záznamům hierarchii archivů.
- **Artwork_Regesten** - Vstupní data představující podmnožinu textových přepisů.
- **Person_Namen** - Podmnožina záznamů reprezentující historické osobnosti.
- **README.txt** - Obsahuje stručný popis původních dokumentových kolekcí.

Výsledky

Transformovaná data ve formátu *JSON*. Opět se nacházejí v podadresáři *test_data*.

- **Archiv_Link_JSON** - Obsahuje zdrojový soubor s unifikovanou hierarchií archivů převedenou do formátu *JSON*.
- **Artwork_Regesten_JSON** - Adresář s dílčími přepisy historických textů, již ve formátu *JSON* dokumentů.
- **Person_Namen_JSON** - Adresář s konvertovanou podmnožinou historických osob, opět v podobě *JSON*.

[58] https://docs.docker.com/guides/docker-concepts/the-basics/what-is-an-image/

Bibliografie

[Das23] (DASHBOARDFOX), Team Post. *Qlik Sense Pros and Cons (Straight Talk Review)* [5000fish, Inc.]. 2023-07. Dostupné také z: https://dashboardfox.com/blog/pros-cons-qlik-sense-straight-talk-review/.

[Ado24] ADOBE. *Data Visualization Fundamentals*. 2024. Dostupné také z: https://spectrum.adobe.com/page/data-visualization-fundamentals/.

[AA16] AJIBADE, S.; ADEDIRAN, A. An Overview of Big Data Visualization Techniques in Data Mining. *Journal of Data Mining*. 2016, roč. 4, s. 105–113.

[Alt22a] ALTEXSOFT. *The Good and the Bad of Microsoft Power BI Data Visualization*. 2022-08. Dostupné také z: https://www.altexsoft.com/blog/power-bi-pros-cons/.

[Alt22b] ALTEXSOFT. *The Pros and Cons of Vue.js*. 2022-08. Dostupné také z: https://www.altexsoft.com/blog/pros-and-cons-of-vue-js/.

[Aro22] ARORA, Y. *Understanding MongoDB Data Modeling: A Comprehensive Guide*. Hevo Data Inc, 2022-02. Dostupné také z: https://hevodata.com/learn/mongodb-data-modeling/#1.

[Ars23] ARSALAN, M. *CSS Frameworks List and Their Pros and Cons*. 2023-02. Dostupné také z: https://makemychance.com/css-frameworks-list-and-their-pros-and-cons/.

[AT 23] AT TEI-C.ORG. *About – TEI: Text Encoding Initiative*. 2023-11. Dostupné také z: https://tei-c.org/about/.

[Bac+22] BACH, B. et al. Dashboard Design Patterns. *IEEE Transactions on Visualization and Computer Graphics*. 2022, roč. 29, č. 1. Dostupné z DOI: 10.1109/TVCG.2022.3209448.

[BI24] BI, Dundas. *Using a table visualization*. Insightsoftware, 2024. Dostupné také z: https://www.dundas.com/support/learning/documentation/data-visualizations/using-a-table-visualization.

[Dat23] DATACAMP, Inc. *A List of The 19 Best ETL Tools And Why To Choose Them*. 2023-12. Dostupné také z: https://www.datacamp.com/blog/a-list-of-the-16-best-etl-tools-and-why-to-choose-them.

[DD16] DWIVEDI, K.; DUBEY, S. K. Implementation of Data Analytics for MongoDB Using Trigger Utility. In: *Computational Intelligence in Data Mining—Volume 1: Proceedings of the International Conference on CIDM, 5-6 December 2015*. Springer India, 2016, s. 39–47.

[EH11] ERIKSSON, M.; HALLBERG, V. *Comparison between JSON and YAML for data serialization*. 2011. The School of Computer Science a Engineering Royal Institute of Technology.

[Few07] FEW, S. Data Visualization - Past, Present, and Future. 2007. Dostupné také z: https://www.perceptualedge.com/articles/Whitepapers/Data_Visualization.pdf.

[Gee19] GEEKSFORGEEKS. *D3.JS (Data Driven Documents)*. Sanchhaya Education Private Limited, 2019-01. Dostupné také z: https://www.geeksforgeeks.org/d3-js-data-driven-documents/.

[Gee23] GEEKSFORGEEKS. *MongoDB Advantages & Disadvantages*. Sanchhaya Education Private Limited, 2023-12. Dostupné také z: https://www.geeksforgeeks.org/mongodb-advantages-disadvantages/.

[Geo23] GEOPITS. *MongoDB Query Language*. 2023-08. Dostupné také z: https://www.geopits.com/blog/mongodb-query-language.html.

[Gil03] GILHEANY, S. *RAM is 100 Thousand Times Faster than Disk for Database Access*. Directions Magazine, 2003-01. Dostupné také z: https://www.directionsmag.com/article/3794.

[Gil23] GILLIS, A. *What is MongoDB? A definition from WhatIs.com*. Ed. BOTELHO, B. TechTarget - DataManagement, 2023. Dostupné také z: https://www.techtarget.com/searchdatamanagement/definition/MongoDB.

[Har14] HARPING, P. *Getty Vocabularies and LOD Conversion*. Getty Vocabulary Program, 2014-11. Dostupné také z: https://www.slideshare.net/AAColaborative/getty-vocabularies-and-lod-conversion.

[Hue20] HUE, A. *Plotly Dash or React.js + Plotly.js? A side by side comparison*. Medium, 2020-11. Dostupné také z: https://towardsdatascience.com/plotly-dash-or-react-js-plotly-js-b491b3615512.

[IBM23] IBM. *What is ETL (extract, transform, load)?* 2023. Dostupné také z: https://www.ibm.com/topics/etl.

[ICO22a] ICONCLASS. *A. Bibliography of publications about Iconclass and of publications using Iconclass for iconographic indexing*. 2022. Dostupné také z: https://iconclass.org/help/abouta.

[ICO22b] ICONCLASS. *C. Web-based pictorial information systems using Iconclass for subject access*. 2022. Dostupné také z: https://iconclass.org/help/aboutc.

[ICO22c] ICONCLASS. *General introduction*. 2022. Dostupné také z: https://iconclass.org/help/about.

[ICO22d] ICONCLASS. *Iconclass basics*. 2022. Dostupné také z: https://iconclass.org/help/basics#notations.

[ICO22e] ICONCLASS. *Iconclass Illustrated Edition*. 2022. Dostupné také z: https://iconclass.org/.

[ICO22f] ICONCLASS. *The Illustrated Iconclass: Search, Browse and Edit*. 2022. Dostupné také z: https://iconclass.org/help/search.

[Ins20] INSTITUTE OF ART HISTORY OF THE CZECH ACADEMY OF SCIENCES. *Annual Report 2019*. 2020. Dostupné také z: https://www.udu.cas.cz/archiv/content_cz/finalni_verze.pdf. Original work published 2019.

[Ins21a] INSTITUTE OF ART HISTORY OF THE CZECH ACADEMY OF SCIENCES. *About Documenta Rudolphina*. 2021. Dostupné také z: http://Documenta.rudolphina.com/Indices/Ind_Spec.xml.

[Ins21b] INSTITUTE OF ART HISTORY OF THE CZECH ACADEMY OF SCIENCES. *Annual Report 2020*. 2021. Dostupné také z: https://www.udu.cas.cz/archiv/content_cz/udu_annual_report_2020_web.pdf. Original work published 2020.

[Ins22] INSTITUTE OF ART HISTORY OF THE CZECH ACADEMY OF SCIENCES. *Annual Report 2021*. 2022. Dostupné také z: https://www.udu.cas.cz/archiv/content_cz/rooeenka_2021.pdf. Original work published 2022.

[Ins23] INSTITUTE OF ART HISTORY OF THE CZECH ACADEMY OF SCIENCES. *Annual Report 2022*. 2023. Dostupné také z: https://www.udu.cas.cz/archiv/content_cz/UDU_annual_report_2022-1688027820.pdf. Original work published 2023.

[IJ19] ISLAM, Mohammad; JIN, Shaowen. An Overview of Data Visualization. In: *2019 International Conference on Information Science and Communications Technologies (ICISCT)*. 2019, s. 1–7. Dostupné z DOI: 10.1109/ICISCT47635.2019.9012031.

[Jan+20] JANG, Junhyun et al. A Specialized Architecture for Object Serialization with Applications to Big Data Analytics. In: *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, s. 322–334.

[Keb22] KEBOOLA. *ETL Process Overview: Design, Challenges and Automation*. 2022-09. Dostupné také z: https://www.keboola.com/blog/etl-process-overview.

[Kha24] KHAN, Fahad. *ETL Using Python: Pros vs. Cons*. Astera, 2024-02. Dostupné také z: https://www.astera.com/type/blog/etl-using-python/.

[Kha23] KHAN, Shahrukh. *A Comprehensive Guide to Data Modeling in MongoDB*. 2023-03. Dostupné také z: https://medium.com/@skhans/a-comprehensive-guide-to-data-modeling-in-mongodb-b63b2df9d9dd.

[Mic24] MICROSOFT. *Co je to vyprávění příběhů dat?* 2024. Dostupné také z: https://powerbi.microsoft.com/cs-cz/data-storytelling/.

[Mid20] MIDWAY, Stephen R. Principles of Effective Data Visualization. *Patterns*. 2020, roč. 1, č. 9. Dostupné z DOI: 10.1016/j.patter.2020.100141.

[Mon23a] MONGODB, Inc. *Data Modeling — MongoDB Manual*. 2023. Dostupné také z: https://www.mongodb.com/docs/manual/data-modeling/.

[Mon23b] MONGODB, Inc. *Data Modeling Introduction — MongoDB Manual*. 2023. Dostupné také z: https://www.mongodb.com/docs/v5.0/core/data-modeling-introduction/.

[Mon23c] MONGODB, Inc. *Operational Factors and Data Models — MongoDB Manual*. 2023. Dostupné také z: https://www.mongodb.com/docs/v5.0/core/data-model-operations/.

[Mon23d] MONGODB, Inc. *Schema Validation — MongoDB Manual*. 2023. Dostupné také z: https://www.mongodb.com/docs/manual/core/schema-validation/.

[Mon23e] MONGODB, Inc. *What is a Document Database?* 2023. Dostupné také z: https://www.mongodb.com/document-databases.

[Mon23f] MONGODB, Inc. *Why Use MongoDB and When to Use It?* 2023-01. Dostupné také z: https://www.mongodb.com/why-use-mongodb.

[Mon24a] MONGODB, Inc. *In-Memory Databases Explained*. 2024. Dostupné také z: https://www.mongodb.com/databases/in-memory-database.

[Mon24b] MONGODB, Inc. *In-Memory Storage Engine — MongoDB Manual*. 2024. Dostupné také z: https://www.mongodb.com/docs/manual/core/in-memory-storage-engine/.

[Mon24c] MONGODB, Inc. *Monitor and Improve Slow Queries*. 2024. Dostupné také z: https://www.mongodb.com/docs/atlas/performance-advisor/.

[MK17] MUKHERJEE, Rajdeep; KAR, Piyali. A comparative review of data warehousing ETL tools with new trends and industry insight. In: *2017 IEEE 7th International Advance Computing Conference (IACC)*. IEEE, 2017.

[Nae+19] NAEEM, M. et al. Trends and future perspective challenges in big data. In: *Advances in Intelligent Data Analysis and Applications: Proceeding of the Sixth EuroChina Conference on Intelligent Data Analysis and Applications*. Arad, Romania, 2019.

[Ohi11] O'HIGGINS, Niall. *MongoDB and Python: Patterns and processes for the popular document-oriented database*. O'Reilly Media, Inc., 2011.

[Pah23] PAHUJA, Shraddha. *Introduction to Semantic UI React - Building Blocks of React*. 2023-09. Dostupné také z: https://www.knowledgehut.com/blog/web-development/semantic-ui-react.

[Pap21] PAPIERNIK, Marek. *An Introduction to Document-Oriented Databases*. DigitalOcean, 2021. Dostupné také z: https://www.digitalocean.com/community/conceptual-articles/an-introduction-to-document-oriented-databases.

[Pav+23] PAVLÍČKOVÁ, Viktorie et al. *Analýza webových stránek Documentaria Rudoplhina*. 2023. Tento zdroj je umístěn v elektronických přílohách práce. Viz podadresář *./tgi_sp/docs*, soubor *documentaria_rudolphina.pdf*.

[Pra08] PRABHU, C. S. R. *Data warehousing: concepts, techniques, products and applications*. PHI Learning Pvt. Ltd., 2008.

[Qli24] QLIKTECH INTERNATIONAL AB. *What is ETL?* 2024. Dostupné také z: https://www.qlik.com/us/etl.

[ree18] REEMWEDA. *Enriching Iconclass LOD by linking keywords to AAT concepts* [Iconclass Blog]. 2018-01. Dostupné také z: https://iconclassblog.com/2018/01/09/enriching-iconclass-lod-by-linking-keywords-to-aat-concepts/.

[Ref23] REFSNES DATA. *XSLT Introduction*. 2023-06. Dostupné také z: https://www.w3schools.com/xml/xsl_intro.asp.

[Rod24] RODZIEWICZ, Daria. *Overview of the Python Dash Framework from Plotly for Building Dashboards* [Appsilon Sp. z o.o.]. 2024-04. Dostupné také z: https://www.appsilon.com/post/overview-of-dash-python-framework-from-plotly-for-building-dashboards.

[San24] SANITY. *What is Vue.js? - Brief Framework Intro / Overview - Glossary*. 2024-01. Dostupné také z: https://www.sanity.io/glossary/vue-js.

[SD22] SHELDON, Ross; DENMAN, Jason. *Node.js (Node)* [WhatIs.com]. 2022-11. Dostupné také z: https://www.techtarget.com/whatis/definition/Nodejs.

[Sch23] SCHROEDER, Adam. *Creating an Interactive Web App with Matplotlib, Python, and Dash*. 2023-06. Dostupné také z: https://plotly.com/blog/dash-matplotlib/.

[SMW22] SCHROEDER, Adam; MAYER, Christian; WARD, Ann Marie. *The Book of Dash*. No Starch Press, 2022.

[SZT12] SCHWARTZ, Baron; ZAITSEV, Peter; TKACHENKO, Vadim. *High performance MySQL: optimization, backups, and replication*. O'Reilly Media, Inc., 2012.

[Sim21] SIMPLILEARN. *What is Data Modelling? Overview, Basic Concepts, and Types in Detail*. 2021-06. Dostupné také z: https://www.simplilearn.com/what-is-data-modeling-article.

[Sou+19] SOUIBGUI, Manel; ATIGUI, Faten; ZAMMALI, Saloua; CHERFI, Samira; YAHIA, Sadok Ben. Data quality in ETL process: A preliminary study. *Procedia Computer Science*. 2019, roč. 159.

[Sus23] SUSZTEROVA, Simona. *What Is a Data Model?* [GoodData]. 2023-04. Dostupné také z: https://www.gooddata.com/blog/what-a-data-model/.

[The19a] THE J. PAUL GETTY TRUST. *About CONA and IA (Getty Research Institute)*. 2019. Dostupné také z: https://www.getty.edu/research/tools/vocabularies/cona/about.html.

[The19b] THE J. PAUL GETTY TRUST. *About the AAT (Getty Research Institute)*. 2019. Dostupné také z: https://www.getty.edu/research/tools/vocabularies/aat/about.html.

[The19c] THE J. PAUL GETTY TRUST. *About the TGN (Getty Research Institute)*. 2019. Dostupné také z: https://www.getty.edu/research/tools/vocabularies/tgn/about.html.

[The23a] THE J. PAUL GETTY TRUST. *About the ULAN (Getty Research Institute)*. 2023-03. Dostupné také z: https://www.getty.edu/research/tools/vocabularies/ulan/about.html#top.

[The23b] THE J. PAUL GETTY TRUST. *Getty Vocabularies FAQs (Getty Research Institute)*. 2023-06. Dostupné také z: https://www.getty.edu/research/tools/vocabularies/faq.html.

[The23c] THE J. PAUL GETTY TRUST. *ULAN Full Record Display (Getty Research)*. 2023. Dostupné také z: https://www.getty.edu/vow/ULANFullDisplay?find=Durer&role=&nation=&prev_page=1&subjectid=500115493.

[Tod08] TODOROVIC, Dejan. Gestalt principles. *Scholarpedia*. 2008, roč. 3, s. 5345. Dostupné také z: http://www.scholarpedia.org/article/Gestalt_principles.

[Tuf01] TUFTE, Edward R. *The Visual Display of Quantitative Information*. Sv. 2. Cheshire, CT: Graphics Press, 2001.

[Uni24] UNIVERSITY, Newcastle. *Box and Whisker Plots* [Numeracy, Maths and Statistics - Academic Skills Kit]. 2024-04. Dostupné také z: https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/data-presentation/box-and-whisker-plots.html.

[Úst19] ÚSTAV DĚJIN UMĚNÍ AV ČR, V.V.I. *Umění na odiv*. 2019. Dostupné také z: https://www.udu.cas.cz/cz/projekty-a-granty/umeni-na-odiv.

[Úst20a] ÚSTAV DĚJIN UMĚNÍ AV ČR, V.V.I. *Inventaria Rudolphina*. 2020. Dostupné také z: https://www.inventariarudolphina.com/about_database.

[Úst20b] ÚSTAV DĚJIN UMĚNÍ AV ČR, V.V.I. *Inventaria Rudolphina*. 2020. Dostupné také z: https://www.inventariarudolphina.com/about_project.

[Úst20c] ÚSTAV DĚJIN UMĚNÍ AV ČR, V.V.I. *Inventaria Rudolphina*. 2020. Dostupné také z: https://www.inventariarudolphina.com/item/PrgA-811.

[Úst20d] ÚSTAV DĚJIN UMĚNÍ AV ČR, V.V.I. *Základní informace*. 2020. Dostupné také z: https://www.udu.cas.cz/cz/ustav-dejin-umeni/zakladni-informace.

[VS09] VASSILIADIS, Panos; SIMITSIS, Alkis. Extraction, Transformation, and Loading. In: *Encyclopedia of Database Systems*. 2009, sv. 10.

[Wan+20] WANG, Zezhong; SUNDIN, Lisa; MURRAY-RUST, Dave; BACH, Benjamin. Cheat Sheets for Data Visualization Techniques. *ACM Conference Proceedings*. 2020, s. 1–13. Dostupné z DOI: 10.1145/3313831.3376271.

[Wei+16] WEISSGERBER, Tracey L.; GAROVIC, Vesna D.; SAVIC, Miroslav; WINHAM, Stacey J.; MILIC, Natasa M. From Static to Interactive: Transforming Data Visualization to Improve Transparency. *PLOS Biology*. 2016, roč. 14, č. 6, e1002484. Dostupné také z: https://doi.org/10.1371/journal.pbio.1002484.

[Whi12] WHITE, Tom. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 2012.

[WC95] WIDOM, Jennifer; CERI, Stefano [ed.] *Active Database Systems: Triggers and Rules for Advanced Database Processing*. Morgan Kaufmann, 1995.

[You22] YOU, Evan. *Introduction | Vue.js*. 2022. Dostupné také z: https://vuejs.org/guide/introduction.html.

[Zím23] ZÍMA, Martin. *MongoDB databáze*. 2023. Dostupné také z: https://www.kiv.zcu.cz/studies/predmety/db2/cviceni-mongodb.html.

[Zla21] ZLATOHLÁVEK, Martin. *Přehled dějin českého umění - Umění na dvoře Rudolfa II*. 2021. Dostupné také z: https://udu.ff.cuni.cz/wp-content/uploads/sites/166/2021/03/41-Umeni-doby-Rudolfa-II.pdf.

Seznam obrázků

Seznam tabulek

Seznam výpisů