FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Bakalářská práce

Výukový kurz pro vývoj databázově orientovaných low-code aplikací

Jan Kadlec

PLZEŇ

2024

FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Bakalářská práce

Výukový kurz pro vývoj databázově orientovaných low-code aplikací

Jan Kadlec

Vedoucí práce
Ing. Jan Valdman, Ph.D.

PLZEŇ 2024

© Jan Kadlec, 2024.

Všechna práva vyhrazena. Žádná část tohoto dokumentu nesmí být reprodukována ani rozšiřována jakoukoli formou, elektronicky či mechanicky, fotokopírováním, nahráváním nebo jiným způsobem, nebo uložena v systému pro ukládání a vyhledávání informací bez písemného souhlasu držitelů autorských práv.

Citace v seznamu literatury:
KADLEC, Jan. *Výukový kurz pro vývoj databázově orientovaných low-code aplikací*. Plzeň, 2024. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky. Vedoucí práce Ing. Jan Valdman, Ph.D.

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

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jan KADLEC**
Osobní číslo:	**A20B0537P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační systémy**
Téma práce:	**Výukový kurz pro vývoj databázově orientovaných low-code aplikací**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s moderní problematickou low-code vývoje aplikací, obvyklými funkcemi a parametry low-code řešení a konkrétně s prostředím HCL Domino Leap.
2. Prostudujte dostupné online návody na vývoj aplikací na této platformě zhodnoťte je.
3. Popište možnosti platformy HCL Domino Leap z pohledu tvorby jednoduchých informačních systémů – např. datové modely, autorizační schémata, tvorba a nasazení aplikací, možnosti integrace, provozní aspekty atd.
4. Vytvořte on-line výukový kurz (tutorial), pomocí kterého se mohou běžní IT pracovníci naučit vyvíjet databázově orientované workflow aplikace na této platformě.
5. Kurz otestujte na vybrané skupině uživatelů.
6. Výsledky vyhodnoťte.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Jan Valdman, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **2. října 2023**
Termín odevzdání bakalářské práce: **2. května 2024**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

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

V Plzni dne 25. října 2023

Prohlášení

Prohlašuji, že jsem tuto bakalářskou 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 Českých Budějovicích dne 30. dubna 2024

. . . . . .

Jan Kadlec

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

Abstrakt

Pro platformu HCL Domino Leap sice existují materiály jako dokumentace nebo wiki. Žádný z nich však nepředstavuje způsob vhodný pro uživatele, který nemá s platformou žádné zkušenosti. Cílem této práce proto bylo vytvořit tutoriál pro platformu HCL Domino Leap, jež by tento nedostatek řešil. Výsledky této práce jsou pak přístupné on-line na webové stránce `https://dleap-tutorial.onrender.com/`.

Abstract

While there are materials for the HCL Domino Leap platform such as documentation or wiki. However, none of them present a method suitable for a user who does not have experience with the platform. Therefore, the aim of this work was to create a tutorial for HCL Domino Leap platform to address this shortcoming. The results of this work are then available online at `https://dleap-tutorial.onrender.com/`.

Klíčová slova

Low-code • No-code • HCL Domino Leap • tutoriál

Poděkování

Na tomto místě bych rád poděkoval vedoucímu této práce Ing. Jan Valdman, Ph.D. za poskytnutí rad a vedení mé práce. Dále pak Ing. Petrovi Kuncovi a Ing. Vítu Hlavovi za otestování a zpětnou vazbu k tutoriálu. V neposlední řadě bych pak poděkoval rodině za podporu při tvorbě této práce.

Obsah

1 Úvod **5**

2 Kontext low-code platforem **7**

2.1 Historický kontext vývoje softwaru 7

2.1.1 Počátky vývoje softwaru 7

2.1.2 Jazyky 2 a 3. generace 7

2.1.3 Vzestup jazyků čtvrté generace 8

2.1.4 Nástup rychlého vývoje aplikací 8

2.1.5 Zrod low code platforem 8

2.1.6 Vývoj a vyspělost 8

2.1.7 Shrnutí 9

2.2 Vývoj řízený modely 9

2.2.1 Základní principy MDD 9

2.2.2 Transformace mezi modely 10

2.3 Low code platformy 12

2.3.1 Definice low-code platforem podle analytických společností 12

2.3.2 Akademická definice 13

2.4 Porovnání low-code platforem a modelově řízeného inženýrství . 14

2.4.1 metodika lc 16

2.4.2 Uživatelé low-code platforem 16

2.5 No-code platformy 17

3 Příklady low-code platforem **19**

3.1 OutSystems 19

3.2 Mendix 20

3.3 Retool 21

4 HCL Software **23**

4.0.1 HCL Domino Leap: Shrnutí 41

5 On-line návody **43**
5.1 HCL Domino Leap dokumentace 43
5.2 HCL Domino Leap wiki 44
5.3 YouTube tutoriál 44
5.4 HCL Domino Leap fórum 45

6 Realizace tutoriálu **47**
6.1 Výběr nástrojů pro realizaci 47
6.2 Výběr vhodného značkovacího jazyka 48
6.3 Výběr z již dostupných řešení 48
6.3.1 Jednotlivé nástroje 48
6.4 Nasazení aplikace 50
6.5 Struktura a obsah tutoriálu 60
6.6 Sekce I. Domino Leap Basics 60
6.7 Sekce II. No-code Development 61
6.8 Sekce III. Low-code Basics 64
6.9 Object Reference 65
6.10 Advanced Techniques 65

7 Testování **67**

8 Závěr a zhodnocení **69**

Bibliografie **71**

Seznam obrázků **77**

Seznam tabulek **79**

Seznam výpisů **81**

Kamil Ekštein,
autor šablony

Úvod 1

Ve světě, kde technologie rapidně pokračují ve svém vývoji a digitalizace se stává nezbytnou součástí každodenního života, je potřeba efektivně a rychle vytvářet software. V posledních letech se koncept vývoje low-code stává čím dál více populární, což reflektuje narůstající potřebu po rychlejším a přístupnějším způsobu vývoje softwaru. Tato bakalářská práce se zaměřuje na vytvoření výukového kurzu pro vývoj databázově orientovaných low-code aplikací, což umožní i méně zkušeným programátorům efektivně vytvářet a spravovat databázové aplikace.

Hlavním cílem této práce je navrhnout a realizovat online výukový kurz, který by poskytl uživatelům bez hlubokých technických znalostí dovednosti potřebné pro vytvoření a správu databázových aplikací s využitím low-code platformy HCL Domino Leap. Důvodem je absence výukového materiálu vhodného pro začátečníka, který do styku s platformou přichází poprvé.

Kontext low-code platforem

2

2.1 Historický kontext vývoje softwaru

Vznik low code platforem představuje milník ve vývoji tvorby softwaru a odráží širší technologické a metodické změny v průběhu desetiletí. Tato kapitola se zabývá historickými událostmi, které připravily půdu pro nástup low-code platforem.

2.1.1 Počátky vývoje softwaru

Vývoj softwaru byl ve svých počátcích doménou odborníků. Programování vyžadovalo detailní znalost strojových jazyků a instrukcí specifických pro hardware, což z vývoje softwaru činilo složitou a časově náročnou činnost. V 60. a 70. letech 20. století se objevily vysokoúrovňové programovací jazyky, jako jsou FORTRAN a C, které část složitosti abstrahovaly, ale stále vyžadovaly značné technické znalosti [1].

2.1.2 Jazyky 2 a 3. generace

Programovací jazyky druhé generace, známé též jako assembler, představují klíčový vývoj v oblasti programování počítačů, protože poskytují srozumitelnější sadu instrukcí, která však úzce souvisí se strojovým kódem. Tyto jazyky používají symbolickou reprezentaci binárních strojových kódů, což programátorům umožňuje psát složitější programy, aniž by se museli zabývat složitými detaily strojového kódu specifického pro hardware. Programovací jazyky třetí generace (3GL), jako jsou C, Fortran a COBOL, zavedly další abstrakci a nabízejí syntaktické prvky vysoké úrovně, které jsou mnohem bližší lidskému jazyku. Tyto jazyky jsou navrženy tak, aby byly nezávislé na konkrétních typech počítačového hardwaru, díky čemuž jsou programy napsané v jazycích 3GL lépe přenositelné mezi různými systémy. Na rozdíl od svých předchůdců podporují 3GL složité datové struktury, řídicí struktury a usnadňují vývoj větších a složitějších aplikací. Tento významný posun nejen zvýšil efektivitu programování, ale také rozšířil možnosti vývoje softwaru a znamenal další zásadní okamžik v historii výpočetní techniky [2].

2.1.3 Vzestup jazyků čtvrté generace

V 80. letech 20. století došlo k nástupu jazyků čtvrté generace (4GL). Tyto jazyky, včetně jazyků SQL a MATLAB, byly abstraktnější než jejich předchůdci a jejich cílem bylo zvýšit produktivitu vývojářů tím, že umožňovaly přirozenější deklarativní styly programování. Ačkoli se nejednalo o přímé předchůdce low code platforem, představovaly 4GL snahu o zjednodušení a zpřístupnění programování [3].

2.1.4 Nástup rychlého vývoje aplikací

V 90. letech 20. století se začaly prosazovat modely rychlého vývoje aplikací (RAD). Tyto modely kladly důraz na rychlé vytváření prototypů, iterativní vývoj a zapojení uživatelů, což kontrastovalo s rigidními a lineárními metodikami, které převládaly dříve [4]. Principy RAD položily základy agilnějšího a flexibilnějšího přístupu k vývoji softwaru, který později ovlivnil metodiky low code vývoje.

2.1.5 Zrod low code platforem

Termín "low code"poprvé použila společnost Forrester Research v roce 2014, ale kořeny tohoto konceptu lze vysledovat již v dřívějších snahách o zjednodušení vývoje softwaru [5]. Mezi první příklady platforem, které vykazovaly vlastnosti nízkého kódu, patřily Visual Basic společnosti Microsoft a FileMaker společnosti Apple, které poskytovaly grafická rozhraní pro vývoj aplikací bez rozsáhlého kódování.

2.1.6 Vývoj a vyspělost

Od svého prvního formálního pojmenování se low-code platformy rychle vyvíjely, což bylo způsobeno rostoucí poptávkou po digitálních řešeních a potřebou překlenout propast mezi informačními technologiemi a podnikáním. Moderní low-code platformy nabízejí mnohem více než jejich dřívější obdoby, a to díky pokročilým funkcím, jako je vývoj s podporou umělé inteligence, rozsáhlé integrační možnosti a podpora komplexních podnikových aplikací [6]. Příkladem může být společnost Mendix nabízející dva způsoby využití umělé inteligence. Prvním způsobem je vývoj za asistence umělé inteligence (AIAD). V tomto případě umělá inteligence nabízí dle kontextu, ve kterém se uživatel nachází, asistenci s tvorbou aplikační logiky. Zároveň je umělá inteligence schopna předvyplňovat parametry funkcí. Dále umělá inteligence nabízí vývojáři doporučení týkající se takzvaných „best practice“. Důsledkem je pak vyšší výkon, bezpečnost a kvalita aplikace. Poslední aplikací umělé inteligence je pak funkcionalita generující postupy validace dat [7]. Druhým způsobem využití je integrace externích služeb třetích stran přímo do aplikace. Těmito externími službami jsou např.: Amazon Textract, Amazon Rekognition, ChatGPT, Google Bard, AWS IoT TwinMaker, Amazon Polly, Amazon Bedrock. Aplikace je tak

obohacena o funkcionality, jako je rozpoznávání textu, chat a analýza textu postavená na velkých jazykových modelech (LLM) a neuronových sítích. Poslední avšak neméně důležitou funkcí platformy je možnost napojení na modely strojového učení v aplikačních rámcích jako jsou Pytorch, TensorFlow, Caffe atd. [7].

2.1.7 Shrnutí

Historická cesta od specializovaného programování k uživatelsky přívětivým rozhraním odráží rozvoj v oblasti vývoje softwaru. Tento přechod k jednoduchosti, agilitě a dostupnosti není jen technologickou změnou, ale reakcí na stále rostoucí poptávku po rychlejších, efektivnějších a přístupnějších metodách vývoje softwaru.

2.2 Vývoj řízený modely

Vývoj řízený modely (MDD) je přístup softwarového inženýrství zaměřený na vytváření a využívání abstraktních modelů pro návrh a vývoj softwarových systémů. Tento přístup zdůrazňuje význam modelování jako klíčové aktivity v životním cyklu vývoje softwaru, přičemž modely jsou považovány za primární nástroje vývoje. MDD propojuje obchodní požadavky s technickou implementací, poskytuje vyšší úroveň abstrakce a usnadňuje komunikaci mezi různými zainteresovanými stranami. Zaštiťován je konsorciem pro standardy výpočetního průmyslu s názvem Object Management Group (OMG) [8], které vydalo v roce 2006 manuál zabývající se právě problematikou MDD.

2.2.1 Základní principy MDD

Z průvodce zpracovaného konsorciem OMG vyvozuje A. W. Brown [9] tyto čtyři základní principy:

- **Modely vyjádřené v dobře definované notaci:** Tato zásada zdůrazňuje, že modely vyjádřené v dobře definované notaci jsou nezbytné pro pochopení systémů, zejména na úrovni podnikových řešení.

- **Tvorba systému na základě modelů:** Systémy mohou být strukturovány kolem sady modelů, přičemž architektonický rámec řídí řadu transformací mezi těmito modely.

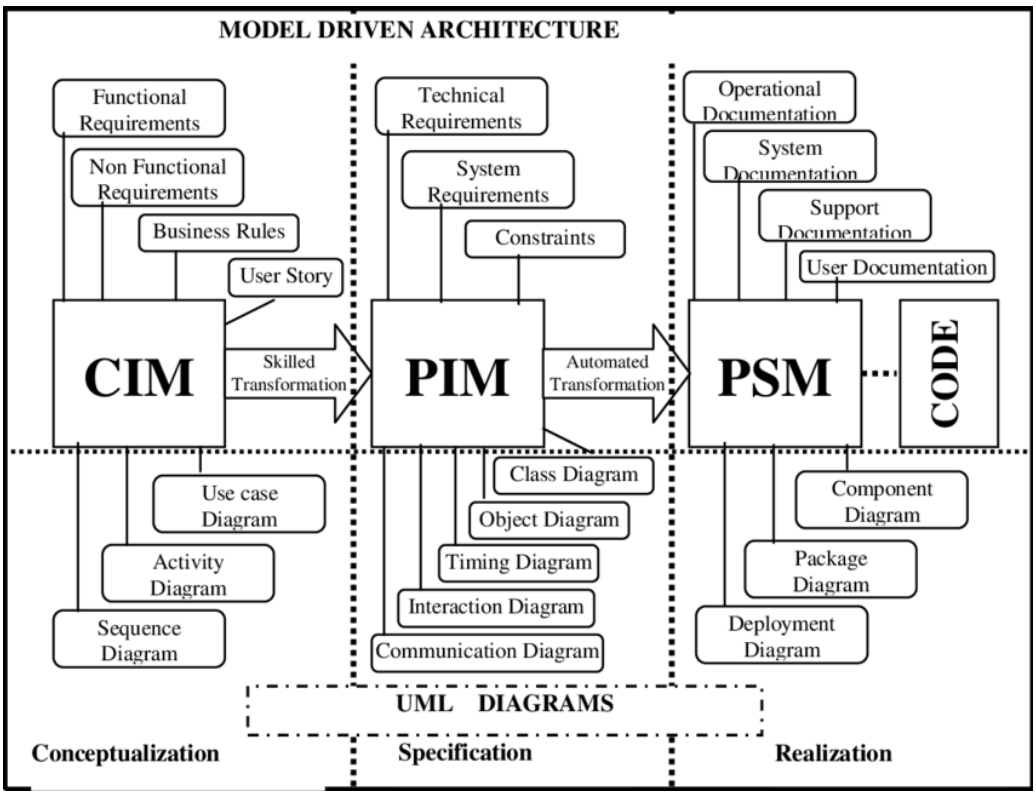
- **Formální podklad pro popis modelů:** Pro popis modelů je nutná sada metamodelů (model modelu). To usnadňuje smysluplnou integraci a transformaci mezi modely a slouží jako základ pro automatizaci pomocí nástrojů.

- **Průmyslové normy pro širší zavedení MDD:** Přijetí a široké osvojení tohoto přístupu založeného na modelech závisí na průmyslových standardech. Tyto

normy jsou nezbytné pro zajištění transparentnosti pro spotřebitele a pro podporu konkurence mezi dodavateli.

2.2.2 Transformace mezi modely

Abychom dobře porozuměli MDD, uveďme si diagram 2.1 ilustrující transformaci mezi jednotlivými modely a jak odděluje specifikaci funkčnosti systému od jeho implementace na konkrétní technologické platformě. Schéma je rozděleno do tří



Obrázek 2.1: Transformace mezi modely v MDD [10]

hlavních vertikálních částí:

- **CIM (Computation Independent Model):** Je to nejvyšší úroveň abstrakce v MDD. Jedná se o vrstvu konceptualizace kladoucí důraz na pochopení a popis problému, kterým se zabýváme. Obsahuje:
  - Funkční požadavky
  - Nefunkční požadavky
  - Obchodní pravidla
  - Uživatelský příběh

Související UML diagramy:

- Diagram případů užití
- Diagram aktivit
- Diagram sekvencí.

Tyto prvky reprezentují systém z pohledu oborových expertů a koncových uživatelů. CIM nezobrazuje podrobnosti o struktuře systému ani o jeho zpracování.

- **PIM (Platform Independent Model):** Tato prostřední část se zaměřuje na fungování systému a je nezávislá na technologických detailech. Zabýváme se jí ve fázi specifikace. Zahrnuje:

  - Technické požadavky
  - Požadavky na systém
  - Omezení

  Související diagramy UML:

  - Diagram tříd
  - Objektový diagram
  - Časový diagram
  - Diagram interakcí
  - Komunikační diagram

  PIM představuje strukturu a chování systému a může být transformován do jednoho nebo více modelů specifických pro platformu (PSM).

- **PSM (Platform Specific Model) a kód:** Tyto části představují realizaci systému v konkrétním technickém kontextu. PSM zpřesňuje PIM do podoby, která využívá specifické mechanismy platformy. Je součástí realizační fáze. Sekce kód obsahuje tyto prvky:

  - Provozní dokumentace
  - Systémová dokumentace
  - Dokumentace podpory
  - Uživatelská dokumentace.

  Související diagramy UML:

- Schéma komponent
- Schéma balíčků
- Diagram nasazení

Přerušované šipky označují proces transformace, přičemž „kvalifikovaná transformace“ spojuje CIM s PIM a „automatizovaná transformace“ spojuje PIM s PSM. Tuto transformaci lze provést pomocí nástrojů, které převádějí vysokoúrovňové abstrakce na podrobnější specifikace.

Zmíněné diagramy UML (Unified Modeling Language) jsou nástroje používané k vizualizaci různých aspektů modelů.

Celkově schéma ilustruje strukturovaný přístup k vyvíjení softwarových systémů, který začíná pochopením problémové oblasti na vysoké úrovni a postupně směřuje k podrobné technické implementaci.

2.3 Low code platformy

Low-code platformy se staly klíčovou technologií pro rychlý vývoj podnikových aplikací. Nabízejí dostupnější přístup k tvorbě softwaru a umožňují uživatelům s různou úrovní technických znalostí rychle vytvářet aplikace [11]. Tato část se zabývá definicemi low-code platforem, jak je definují dvě přední firmy zabývající se analýzou a poradenstvím v oblasti informačních systémů, informačních a komunikačních technologií: Forrester Research a Gartner. Dále se v této části budeme zabývat definicí podle rešerše analyzující ostatní akademické práce zabývající se problematikou právě low-code platforem.

2.3.1 Definice low-code platforem podle analytických společností

2.3.1.1 Definice low-code platforem podle společnosti Forrester Research

Společnost Forrester Research v dokumentu nazvaném "New Development Platforms Emerge for Customer-Facing Applications"autorů Claye Richardsona a Johna R. Rymera, zveřejněného 9. června 2014, se zabývá vznikajícím trendem low-code platforem při vývoji aplikací. Zavádí definici low-code platforem jako "Platformy, které umožňují rychlé dodání aplikace s minimem ručního kódování a rychlým nastavením a nasazením pro systémy zapojení (systems of engagement)."

2.3.1.2 Definice low code platforem podle společnosti Gartner

Společnost Gartner, další vlivná analytická a poradenská firma, nabízí trochu jiný pohled na low-code platformy [12] [13]. Společnost definuje low-code aplikační

platformy (LCAP) jako platformy pro rychlý vývoj a provoz vlastních aplikací díky minimalizaci a abstrakci programovacích jazyků. Podskupina tohoto odvětví podnikové LCAP je určena širšímu okruhu vývojářů, zejména podnikovým vývojářům, a nabízí funkce důležité pro poskytování a údržbu aplikací ve středních a velkých podnicích. Mezi charakteristické vlastnosti patří vysoký výkon, dostupnost a škálovatelnost aplikací, zotavení po havárii (přírodní pohroma, kybernetický útok) [14] , zabezpečení, přístup API k podnikovým cloudovým službám a službám třetích stran, monitorování používání, dohody o úrovni poskytovaných služeb, technická podpora a školení.

Podle společnosti Gartner LCAP musí:

- Obsahovat nástroje pro vývoj kompletní aplikace a to s malým množstvím potřebného kódu v porovnání s běžným vývojem. Aplikací se v tomto případě rozumí aplikace včetně uživatelských rozhraní pro webové a mobilní platformy, aplikační logiku a ukládání dat.
- Podporovat modelově řízený nebo grafický programovací přístup doplněný o skriptování.
- Fungovat jako platforma pro běh aplikací vyvinutých v LCAP.

LCAP by měly navíc podle této společnosti poskytovat:

- Podporu všech fází životního cyklu vývoje softwaru včetně nasazení do více prostředí, sledovatelnost a správu systémovými administrátory.
- Podporu synchronizace změn od více vývojářů pracujících souběžně.
- Vlastní generování rozhraní API pro vytváření balíčků obchodních funkčností a sdílených rozhraní API.
- Konektory pro populární podnikové aplikace, platformy a systémy správy databází.
- Sady nástrojů pro testování nebo podporu integrace s externími nástroji pro správu testů a automatizaci [6].

2.3.2 Akademická definice

Rešerše literatury [15] autorů Karlis Rokis a Marite Kirikova vypracovaná v roce 2023 si klade za cíl objasnit hned několik otázek problematiky low-code platforem. První z těchto otázek se přímo věnuje definici low-code vývoje. Z jimi zanalyzovaných akademických prací pak vyplývá definice následující: *Low-code vývoj softwaru je přístup k vývoji, který podporuje rychlý, flexibilní a iterativní vývoj softwaru tím, že*

umožňuje rychlé převedení požadavků do aplikace prostřednictvím vizuálního programování s grafickým rozhraním, vizuální abstrakcí a minimálním ručním kódováním a zapojením odborníků s různým vzděláním a zkušenostmi s vývojem softwaru. V této práci se budeme řídit právě touto definicí. Z této definice pak můžeme vyvodit co jsou low-code vývojové platformy. Low-code platformy jsou platformy umožňující právě low-code vývoj.

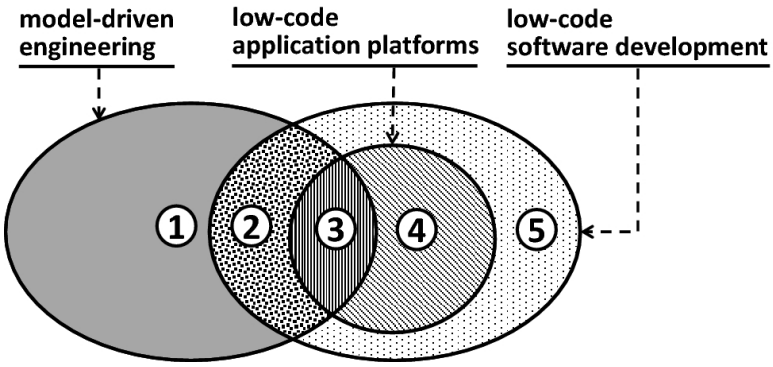
2.4 Porovnání low-code platforem a modelově řízeného inženýrství

Pohledem na to jak se jednotlivé přístupy k vývoji softwaru liší, se zabývala práce z roku 2022 [16], jež dobře ilustruje rozdíly mezi jednotlivými low-code platformami prostřednictvím Vennova diagramu. V této kapitole si shrneme její obsah.

Low-code vývojové platformy a modelově řízené inženýrství (MDE) nabízejí odlišné pohledy na vývoj softwaru s vlastními nástroji , metodikami a cíli. MDE se zaměřuje na vyzdvihnutí modelů jako primárních prvků v životním cyklu vývoje softwaru a využívá je pro širokou škálu úloh, jako je specifikace, simulace, verifikace a generování kódu. Cílem tohoto přístupu je zlepšit produktivitu a zvýšit kvalitu automatizací různých fází vývoje pomocí modelů a doménově specifických jazyků (DSL), čímž se proces vývoje stává více cíleným a snižuje se tím složitost.

Naproti tomu LCDP jsou navrženy tak, aby zjednodušily vývoj aplikací a umožnily uživatelům vytvářet aplikace prostřednictvím grafických rozhraní a minimálního ručního kódování. LCDP se zaměřují na širší publikum, včetně neodborných uživatelů, a snižují složitost spojenou s vývojem i správou aplikací tím, že poskytují cloudová řešení, která řeší různé aspekty životního cyklu aplikací.

netechnických



Obrázek 2.2: Vennův diagram znázorňující společné rysy a rozdíly mezi přístupem řízeným modely, platformami pro low-code aplikace a low-code vývojem softwaru [16]

Diagram na obrázku 2.2 důkladně popisuje společné vlastnosti a rozdíly mezi metodikami low-code a MDE a identifikuje pět hlavních oblastí:

Přístupy řízené čistě modelem (oblast 1): Tato oblast zahrnuje metodiky, které využívají modely jako strojově zpracovatelné prvky (artefakty), aniž by se nutně snažily minimalizovat množství ručně psaného kódu. Důraz je zde kladen na automatizaci složitých úloh, jako je simulace, formální verifikace a reverzní inženýrství. Příklady zahrnují optimalizaci návrhových modelů UML pro lepší výkon a nástroje pro extrakci modelů z existujících kódových databází.

Modely řízená redukce kódu (oblast 2): Přístupy v této oblasti rovněž považují modely za prvořadé koncepty a usilují o snížení objemu ručního kódování pomocí mechanismů, jako je generování kódu nebo jeho interpretace. Na rozdíl od oblasti 3 však tyto přístupy neposkytují možnosti nasazení nebo správy životního cyklu. Významnými příklady jsou JHipster a jeho jazyk JDL, Google Protocol Buffers a systém pro konverzi modelů OlivaNova.

Modely řízené integrované platformy (oblast 3): Tato oblast představuje platformy, které využívají modely ke snížení náročnosti kódování a nabízejí komplexní řešení pro nasazení a správu životního cyklu. Tyto platformy, jako jsou Codebots a Judo, nabízejí ucelenější přístup tím, že integrují vývoj řízený modely s provozní podporou v rámci stejného ekosystému.

Low-code platformy bez explicitního modelování (oblast 4): Tato oblast zahrnuje platformy, které usnadňují vývoj aplikací s minimálním kódováním, ale nespoléhají se na modely odpovídající explicitně definovaným jazykům nebo metamodelům. Místo toho mohou využívat jednodušší formy reprezentace dat, jako jsou XML/JSON bez schémat, nebo relační databázová schémata a poskytovat vestavěnou podporu pro nasazení a správu životního cyklu.

Zjednodušené přístupy k redukci kódu (oblast 5): Podobně jako v oblasti 4 se tyto metody zaměřují na snížení požadavků na kódování, ale nenabízejí vestavěné funkce pro nasazení nebo správu životního cyklu. Navíc nepoužívají explicitní modelovací jazyky. Příkladem jsou generátory řízené databázovými schématy a jednorázové generátory, jako jsou generátory poskytované aplikačním rámcem Ruby on Rails.

dodat zdroje na původní zdroje ve studii?

Dále se budeme věnovat dalším třem aspektům doplňující diagram 2.2.

Platformy. Srovnání se rozšiřuje i na platformy, kde low-code řešení, především cloudová, nabízejí výrazné usnadnění použití pro netechnické uživatele tím, že eliminují potřebu instalace vývojové platformy a nasazení aplikace. Některá řešení MDE se sice také přesouvají do cloudových prostředí, ale většina z nich zůstává založena na desktopech a zaměřuje se na technicky zdatnější uživatele.

Uživatelé. Demografické charakteristiky uživatelů dále rozlišují obě varianty, přičemž LCDP oslovují především občanské vývojáře a neprofesionální programá-

tory díky své jednoduchosti a přístupnosti. Naproti tomu nástroje MDE jsou často zaměřeny na profesionální vývojáře a k jejich plnému využití je zapotřebí hlubších technických znalostí.

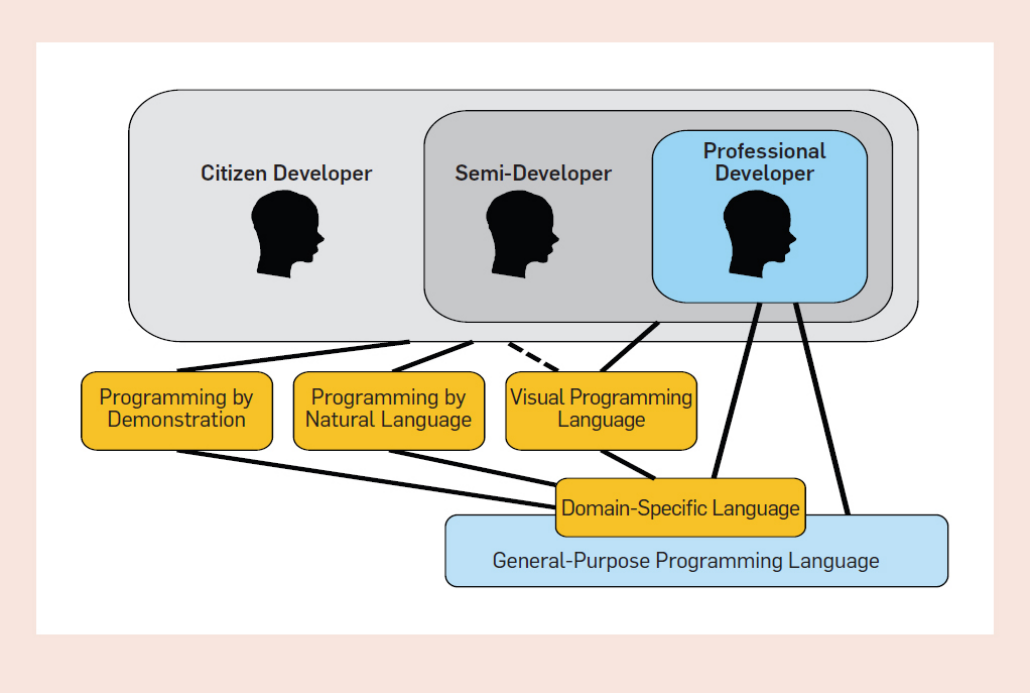
Oblasti užití. V oblasti použití nastávají rozdíly, kdy se low-code platformy zpočátku zaměřovaly na podnikové aplikace, ale postupně se rozšiřují do oblastí, jako je internet věcí, chatboti a strojové učení. MDE, ačkoli jsou schopny řešit i tyto oblasti, se tradičně zaměřují na techničtější oblasti vyžadující specializované znalosti, jako je automobilový průmysl, energetika a kyberneticko-fyzikální systémy.

Celkově tak studie poskytuje dobrý vhled na rozdílné a společné vlastnosti low-code platforem a modelově řízeného inženýrství.

2.4.1 metodika lc

2.4.2 Uživatelé low-code platforem

Martin Hinzel se ve své práci [17] mimo jiné zabývá spektrem uživatelů low-code platforem. Zavádí rozdělení na tři skupiny podle jejich zkušeností s vývojem softwaru viz. 2.3. Jsou jimi:



Obrázek 2.3: Low-code uživatelé [17]

- *Neodborný / příležitostný vývojář (Citizen developer)* je amatérský programátor s malým množstvím profesního programátorského vzdělání. Jeho kariéra je

založena na odborné znalosti z jiného odvětví. Low-code platformy umožňují takovýmto oborovým odborníkům, aby se stali *běžnými vývojáři* a vytvářeli softwarová řešení relevantní pro jejich oblasti expertízy.

- *Polo-profesionální vývojáři (Semi-developers)*: Tato skupina se nachází na pomezí mezi *profesionálními* a *běžnými vývojáři*.
- *Profesionální vývojáři (Professional Developers)*: Tito vývojáři mají programátorské vzdělání nebo kariéru v softwarovém vývoji. Při vývoji na low-code platformách využívají svých dovedností a zkušeností ke zvýšení svojí produktivity práce. Jsou na opačném konci spektra oproti *běžným vývojářům*.

2.5 No-code platformy

No-code vývoj představuje zásadní posun v softwarovém inženýrství, který umožňuje vytvářet a nasazovat aplikace osobám s minimálními znalostmi kódování. Tento přístup využívá grafická uživatelská rozhraní a konfiguraci namísto tradičního ručního programování na počítači, čímž výrazně snižuje bariéru pro vývoj softwaru a umožňuje širšímu okruhu lidí vytvářet digitální řešení.

Jedním z hlavních přínosů no-code vývoje je jeho potenciál zpřístupnit tvorbu softwaru i netechnickým uživatelům, jako jsou obchodní analytici, projektoví manažeři a podnikatelé. Tato míra zapojení neodborných vývojářů podporuje inovace a urychluje digitální transformaci v různých odvětvích tím, že umožňuje osobám s odbornými znalostmi v dané oblasti, ale s omezenými programátorskými dovednostmi, aby se přímo podílely na vývoji aplikací [18].

No-code platformy nabízejí řadu výhod, včetně rychlého vývoje aplikací, snížení nákladů a agility, která umožňuje rychle reagovat na potřeby trhu nebo podnikání. Tyto platformy často přicházejí s rozhraními typu drag-and-drop, předpřipravenými šablonami a integračními funkcemi, které zjednodušují proces vytváření složitých aplikací. Umožňují vývoj široké škály aplikací, od mobilních a webových aplikací až po sofistikovanější podnikové systémy, aniž by bylo nutné psát kód od začátku [19].

Vývoj no-code však není bez problémů. Obavy týkající se omezení přizpůsobitelnosti, škálovatelnosti a možnosti vytváření řešení typu "černá skříňka", kterým uživatelé plně nerozumějí, jsou na místě. Navíc, i když platformy no-code mohou výrazně snížit potřebu tradičního kódování, zcela ji neodstraňují. V některých případech může být k vylepšení nebo rozšíření možností no-code řešení stále zapotřebí minimální množství kódování nebo technických znalostí [19].

Integrace nástrojů umělé inteligence s no-code platformami je novým trendem, jehož cílem je dále rozšířit možnosti a dostupnost no-code vývoje. Nástroje poháněné umělou inteligencí mohou pomáhat v různých fázích procesu vývoje od

návrhu až po nasazení tím, že poskytují doporučení, automatizují úlohy a dokonce generují úryvky kódu na základě popisu v přirozeném jazyce. Očekává se, že díky těmto pokrokům bude vývoj bez kódu ještě výkonnější a přístupnější, což uživatelům umožní vytvářet sofistikovanější a přizpůsobitelnější aplikace s minimálním úsilím [20].

V oblasti specializovaných aplikací jako je umělá inteligence začínají platformy no-code nabízet řešení, která automatizují složité úlohy jako je například generování funkčních blokových schémat pro návrh průmyslových procesů přímo z dokumentace. Tento přístup nejen zjednodušuje proces vývoje, ale také otevírá nové možnosti pro uplatnění umělé inteligence a strojového učení v oborech, které tradičně spoléhaly spíše na manuální techniky programování [21].

Další inovativní aplikaci principů no-code lze pozorovat v Low-code Large Language Models (LLM), které usnadňují vytváření složitých logických a pracovních postupů prostřednictvím vizuálních programovacích rozhraní. To umožňuje uživatelům definovat, upravovat a spravovat logiku aplikací řízených umělou inteligencí, aniž by se museli hlouběji zabývat základním kódem, čímž se překlenuje propast mezi koncepčním návrhem na vysoké úrovni a technickou implementací [22].

Závěrem lze říci, že vývoj bez kódu mění podobu softwarového inženýrství tím, že zpřístupňuje tvorbu technologií, podporuje inovace v různých oblastech a urychluje digitální transformaci. Očekává se, že s dalším vývojem této technologie se jednotlivcům i organizacím otevře ještě více možností vytvářet vlastní řešení šitá na míru jejich konkrétním potřebám, aniž by k tomu potřebovali rozsáhlé znalosti programování.

No-code a low-code platformy a LLM

Příklady low-code platforem

3

Z důvodu, že low-code platforem je velké množství, je potřeba k otestování vybrat pouze některé z nich. Pro výběr jsou důležitá dvě kritéria. Prvním kritériem je to, aby platforma byla přímo testovatelná u poskytovatele. Nemusíme tedy spoléhat na jejich marketingová prohlášení a materiály. Druhým kritériem je, aby poskytovatel přímo ve svých materiálech uváděl termín „low code“ ve spojení se svojí vlastní platformou. Zde jsou platformy, které těmito kritérii prošly:

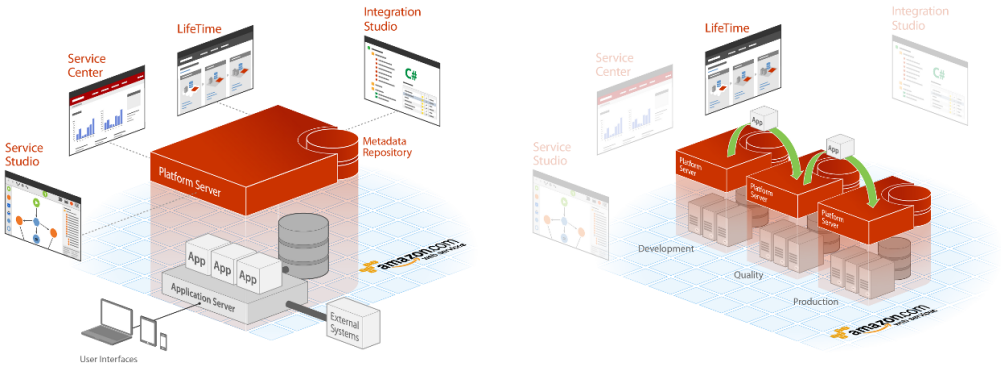
3.1 OutSystems

OutSystems je společnost původem z Portugalska založená Paulem Rosaldem v roce 2001. Platforma se skládá z několika hlavních komponent (viz. 3.1):

- *Platform Server*: Komponenta, která řídí orchestraci všech činností spojených s provozem, nasazováním a správou všech aplikací. *Platform Server* syntetizuje kód pro konkrétní sadu technolgií. Například pro Windows Server s využitím SQL Serveru to bude ASP.Net a SQL kód. Kompilovaná aplikace je poté nasazena na *Application Server*. *Platform Server* také zahrnuje Scheduler Service, který spravuje provedení kroků v rámci procesních modelů vytvořených pomocí procesně-byznysových technologií (BPT) a také naplánovaných úloh vytvořených pomocí časovačů (Timers) [23].

- *Service Studio*: Vizuální vývojový nástroj pro vytváření, úpravy a nasazování aplikací, který je instalován na pracovní stanici vývojářů. Service Studio je vývojové prostředí pro všechny doménově specifické jazyky (DSL) podporované OutSystems. Při publikování aplikace *Service Studio* uloží dokument s modelem aplikace a odešle jej na *Platform Server*. IDE je rozděleno do čtyř hlavních pohledů: modelování procesů, průchod uživatelským rozhraním, vlastní logika/API a modelování databáze [23].

- *LifeTime*: Centralizovaná konzole pro správu infrastruktury, prostředí, aplikací, IT uživatelů a bezpečnosti.
- *Service Center*: Správní konzole pro prostředí infrastruktury.
- *Integration Studio*: Vývojový nástroj pro vytváření konektorů pro integraci aplikací OutSystems s dalšími podnikovými systémy, instalovaný na pracovní stanici vývojářů.
- *Application Server*: Běží na platformě Oracle WebLogic, JBOSS nebo IIS. Server ukládá a spouští vyvinuté aplikace, které jsou propojeny s relačním databázovým systémem, jako je SQL Server, Oracle nebo MySQL. SQL kód generovaný *Platform Serverem* je specifický pro vybraný databázový systém [23].

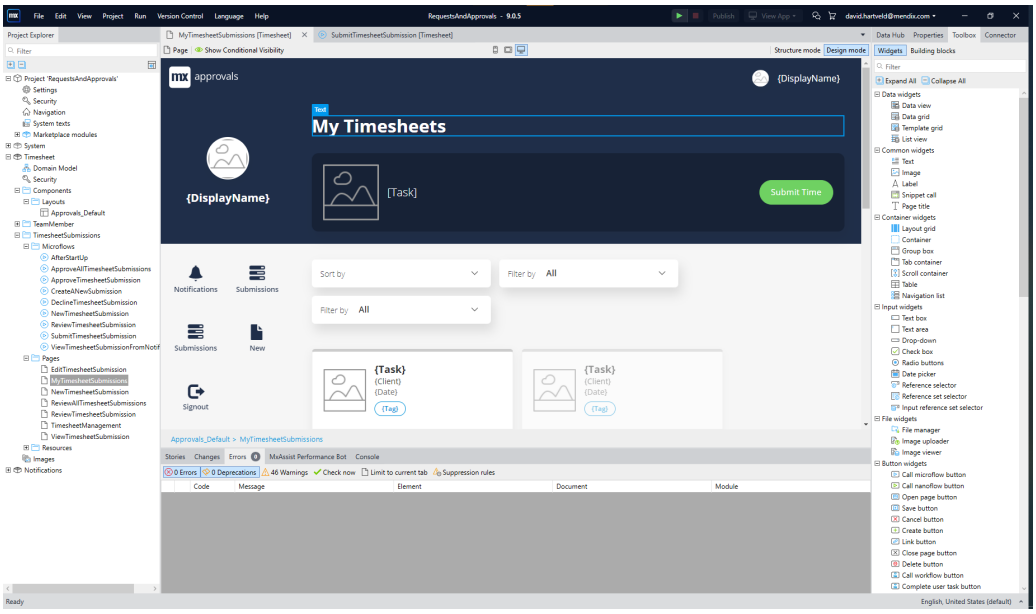


Obrázek 3.1: Jednotlivé komponenty platformy OutSystems [24]

3.2 Mendix

Mendix je společnostu původem z Nizozemska, sídlící v Bostonu, koupená v roce 2018 německou technologickou společností Siemens [25] .

Na rozdíl od společnosti OutSystems nabízí Mendix jedno primární vývojové prostředí Mendix Studio Pro (viz. obrázek 3.2). Mendix Studio Pro představuje pokročilé vizuální integrované vývojové prostředí (IDE), které výrazně urychluje proces vývoje aplikací. Tato platforma je navržena tak, aby automatizovala všední úkoly a zjednodušila provádění složitých operací. Jako robustní, modelově řízené vizuální vývojové prostředí usnadňuje tvorbu aplikací na platformě Mendix. Studio Pro nabízí komplexní sadu nástrojů, které uživatelům umožňují bez námahy vytvářet, upravovat, integrovat, testovat a nasazovat aplikace, přičemž tyto funkce sjednocuje do jediného uceleného rámce [27].



Obrázek 3.2: Mendix Editor stránek [26]

Platforma Mendix nabízí rozsáhlou knihovnu modulárních zdrojů přizpůsobených pro různá odvětví podnikání. Tyto zdroje jsou přístupné prostřednictvím tržiště Mendix Marketplace. Platforma primárně využívá pro nasazení aplikací vyvinutých v rámci svého ekosystému Mendix Cloud hostovaný na Amazon Web Services (AWS). Kromě toho rozšiřuje podporu pro nasazení aplikací spravovaných samotnými klienty, jak na privátních cloudových prostředích založených na Kubernetes, tak ve vlastních datových centrech [6].

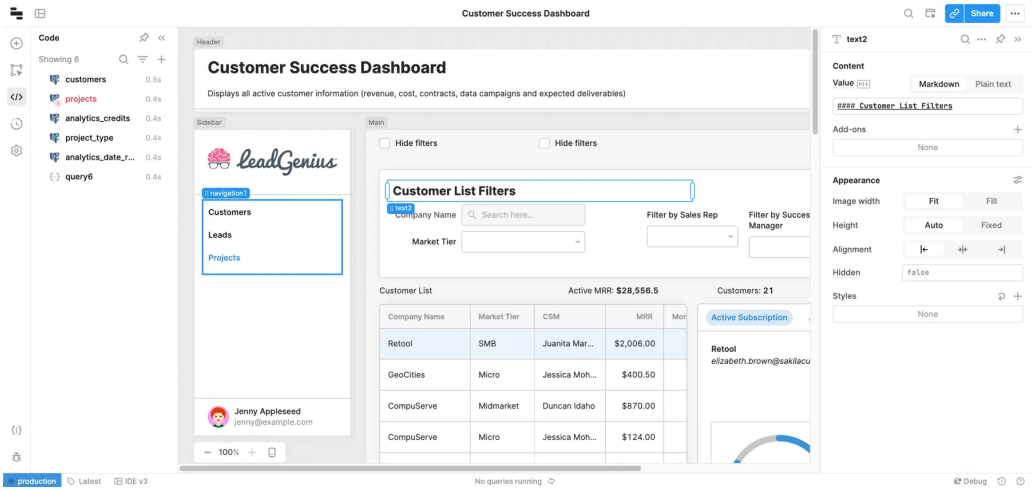
3.3 Retool

Společnost Retool byla založená v roce 2017 v San Francisku v Californii [28]. V porovnání s předchozími dvěma platformami pro použití Retoolu není potřeba si instalovat žádnou aplikaci na pracovní stanici. Pro práci s Retoolem se používá webový editor (viz obr. 3.3). Retool automaticky vytvoří instanci aplikace ve verzi určené k vývoji na doméně třetího řádu.

Architektura systému Retool je založena na modelu klient-server. Klientská strana nabízí bohaté interaktivní uživatelské rozhraní vytvořené pomocí moderních webových technologií. Na straně serveru Retool komunikuje s různými datovými zdroji a rozhraními API a zajišťuje doménovou logiku a zpracování dat.

Mezi klíčové funkcionality patří:

- Rozhraní Drag-and-Drop:Zjednodušuje tvorbu uživatelského rozhraní a umožňuje vývojářům soustředit se spíše na funkčnost než na složité grafické roz-



Obrázek 3.3: Retool editor [29]

vržení.

- Předpřipravené komponenty: Nabízí knihovnu komponent uživatelského rozhraní, jako jsou tabulky, tlačítka a formuláře, které lze podle potřeby upravit.
- Připojení k datům: Podporuje připojení k různým databázím (SQL a NoSQL) a rozhraním API, což usnadňuje manipulaci s daty a jejich integraci.
- Vlastní kód: Retool je sice primárně low code nástroj, ale umožňuje vkládat vlastní kód JavaScriptu, což nabízí flexibilitu pro složité úlohy.

HCL Software 4

Protože pro následnou práci je znalost plaformy HCL Domino Leap potřebná ve větším detailu než u platforem předchozích, bylo rozhodnuto přistupovat k analýze produktů HCL Software, konkrétně k HCL Domino Leap pomocí postupu popsaném v práci zabývající se analýzou deseti low-code platforem. Ulrich Frank, Pierre Maier a Alexander Bock se v teto práci zabývají právě problematikou low-code platforem [30].

Popis metodiky. Metodika se v základu zaměřuje na low-code platformy ze dvou úhlú pohledu. Prvním pohledem je pohled na historii a kontext low-code platforem. Zabývá se pozadím a historií dané platformy, profilem poskytovatele a nabízí pohled na to, jak jednotliví poskytovatelé směřují svůj marketing. Nabízí tak strukturu pro analýzu, jež je primárně zaměřená na charakteristiky low-code platforem a jejich funkcí lišících se mezi produkty (viz. tabulka 4.1).

Druhým úhlem pohledu je pohled zaměřující se přímo na detailní analýzu low-code platforem prostřednictvím klíčových konceptů. Jedním ze konceptů je koncept základních abstrakcí zaměřujících se na statické, funkční a dynamické perspektivy systémů s koncepčními modely jako základy vývoje. Opakované použití je podporováno prostřednictvím referenčních modelů a datových struktur, přičemž jazyky pro vytváření těchto modelů jsou buď obecné, nebo specifické pro danou oblast. Dále se autoři zaměřují na uživatelko-systémové interakce. Jedná se o modely, které reprezentují prvky a rozvržení uživatelského rozhraní. Zdůrazňují význam interaktivních prvků („widgetů“) v návrhu softwaru. Další významnou součástí je znovupoužití a adaptabilita. Text zdůrazňuje význam opakované použitelnosti v softwarových systémech, které mohou sahat od obecných aplikačních rámců(„frameworků“) až po modely specifické pro danou oblast. Přizpůsobivost těchto systémů je klíčová a závisí na úrovni abstrakce použitého jazyka a možnosti upravovat implementační dokumenty jako kód či schémata. Též je zde pamatováno na externí komponenty nebo služby a jejich modifikovatelnost. Práce dále srovnává low-code platformy s jinými prostředími pro vývoj softwaru a zohledňuje rozsah aplikací, pro které jsou vhodné, od malých až po rozsáhlé projekty. U komponent, které pracují s re-

Profil poskytovatele
Produktové Portfolio
• sada vývojových nástrojů • význam low-code platformy v rámci produktového portfolia • nabídka dalších softwarových systémů • podíl na trhu
Původ produktu
• produkt byl poprvé označen jako low-code platforma • předchůdci platformy • vývoj produktu
Zaměřeno na marketing
• jedinečný prodejní argument (z ang. Unique Selling Propositon) • cílové skupiny uživatelů • inzerované příklady užití • sliby

Tabulka 4.1: Původ a obraz dodavatele low-code produktů [30]

prezentacemi, jako jsou editory modelů, je zdůrazněna jejich úloha při zvyšování produktivity a posilování postavení uživatelů. Text se zabývá funkcemi pro spolupráci více uživatelů, jako jsou přístupová práva pro jednotlivé role, a problémy při nasazení na různých platformách. Jsou zmíněna řešení, jako je automatizovaná instalace softwaru a technologie pro škálování aplikací. Ekonomická část se zabývá ekonomickými aspekty, včetně pořizovacích nákladů, licenčních poplatků, zvýšení produktivity, nákladů na školení, údržbu a nasazení. Dotýká se také přenositelnosti aplikací v různých prostředích. V práci je dále uvedeno několik tabulek s podrobnými kritérii pro hodnocení low-code platforem, které zahrnují statické abstrakce, dynamické abstrakce, vývoj grafického uživatelského rozhraní a další aspekty hodnocení.

Zaměření na Reprezentace		
Konceptuální reprezentace	*Jazyky*	*Komponenty*
• datové typy • datové modely • objektové modely • ...	• GPML (např. entitně vztahový model) • DSML • proprietární jazyk	• editor modelů • generátor – schéma databáze (DB) (typ DB) – kód (programovací jazyk)
Zaměření na Opětovné Použití a Adaptabilitu		
Generické abstrakce	*Jazykové koncepty*	*Přístup k externím zdrojům a k dokumentům na úrovni implementace*
• generické abstrakce • doménově specifické abstrakce *Komentář:* reprezentováno jednou nebo více koncepčními reprezentacemi uvedenými v nejlevějším sloupci výše	• generalizace • zapouzdření • kompozice • ...	• datové modely • databáze • web • kód • kód (programovací jazyk) *Komentář:* přístup k externím zdrojům obvykle vyžaduje rozhraní jako jsou ODBC, SQL, HTTP atd.
Zaměření na Integraci		
Běžné statické abstrakce napříč škálou aplikací	*Komentář:* Pokud dvě aplikace mohou (znovu) používat stejnou statickou abstrakci, např. stejný datový model, výměna dat mezi těmito aplikacemi je usnadněna.	
Přístup k běžným datovým repozitářům napříč škálou aplikací	*Komentář:* Pokud k tomu obě aplikace mají přístup k společnému datovému repozitáři, např. společné databázi, je možná integrace na úrovni instancí (vyhnutí se redundanci dat).	

Tabulka 4.2: Koncepty charakterizující statickou perspektivu

Zaměření na Reprezentace		
Konceptuální reprezentace	*jazyky*	*Komponenty*
• hierarchie funkcí • tok dat • tok zpráv • obchodní pravidla • modely objektů • ...	• DFD • UML • proprietární jazyk • DSML, například, jazyk specifikace pravidel	• editor modelů • generátor – kód (programovací jazyk) – definice rozhraní *Komentář:* generátory vyžadují přístup k odpovídajícím statickým reprezentacím.
Zaměření na opětovné použití a adaptabilitu		
Referenční abstrakce	*Jazykové koncepty*	*Přístup k externím zdrojům a k dokumentům na úrovni implementace*
• obecné abstrakce • doménově specifické abstrakce – specifičnost *Komentář:* reprezentováno jednou nebo více konceptuálními reprezentacemi uvedenými v nejlevějším sloupci výše	• kompozice • ...	• (webové) služby • knihovny funkcí • kód knihovny tříd • kód • ... *Komentář:* přístup k externím zdrojům obvykle vyžaduje rozhraní jako jsou API a specifické protokoly jako REST nebo SOAP

Tabulka 4.3: Koncepty charakterizující funkční perspektivu

Zaměření na Reprezentace		
Konceptuální reprezentace	*Jazyky*	*Komponenty*
• modely obchodních procesů • obecné modely procesů • ...	• BPMN • UML • Petriho síť • stavový diagram • proprietární jazyk	• editor modelů • generátor – kód (programovací jazyk) – schéma pracovního postupu *Komentář:* generátory vyžadují přístup k odpovídajícím statickým a funkčním reprezentacím.
Zaměření na opětovné použití a adaptabilitu		
Referenční abstrakce	*Jazykové koncepty*	*Přístup k externím zdrojům a k dokumentům na úrovni implementace*
• obecné abstrakce • doménově specifické abstrakce – specifičnost *Komentář:* reprezentováno jednou nebo více konceptuálními reprezentacemi uvedenými v nejlevějším sloupci výše	• kompozice • ... *Komentář:* opětovné použití procesů čelí vážným překážkám	• schéma pracovního postupu • kód (programovací jazyk) • ... *Komentář:* provedení pracovních postupů může vyžadovat přístup k externím funkcím/službám.

Tabulka 4.4: Koncepty pro analýzu dynamických abstrakcí

Perspektiva interakce s uživatelem		
Konceptuální reprezentace	*Komponenty*	*Architektonické aspekty*
• modely GUI • obecné modely procesů • ...	• editor GUI • „táhni a pusť“ (drag-and-drop) editor • WYSIWYG • styl specifický pro platformu • generování kódu	přístup k integraci GUI se zbytkem aplikace, například vzor model-view-controller (MVC)
Zaměření na opětovné použití a adaptabilitu		
Referenční abstrakce	*Adaptabilita*	
• knihovna obecně použitelných a doménově specifických prvků GUI (rámy, widgety, ..) • příklad aplikací s opětovně použitelnými GUI	• přizpůsobení widgetů • (re)definice stylů a šablon Přizpůsobení specifickému vzhledu a pocitu platformy	

Tabulka 4.5: Koncepty pro analýzu podpory vývoje grafického uživatelského rozhraní

Přístupnost a pohodlí používání	
• návod - pokyny – podpora metodického vývoje – vestavěné pomocné funkce • uživatelské rozhraní low-code platforem – nedostatky uživatelského rozhraní low-code platforem – přizpůsobitelné různým uživatelským skupinám • modelovací jazyky – rozšířenost – složitost, výraznost	
Způsoby použití	
Platforma (tj. vývoj aplikací)	*Aplikace*
• definice a správa přístupových práv • podpora spolupráce (např. větvení) • použití v distribuovaných prostředích • použití v heterogenních prostředích	• definice a správa přístupových práv • použití v distribuovaných prostředích • použití v heterogenních prostředích • podpora spolupráce (např. společná úprava)
Zaměření na nasazení a škálovatelnost	
Cílová umístění	*Podpora*
• lokální stroj • datové centrum (cloud) •	• automatizovaná, transparentní instalace • ... napříč různými platformami • dostupnost nasazené aplikace (např. poskytnutá API) • další podpora nástrojů (např. kontejnery, Kubernetes)
Zaměření na umělou inteligenci	

• interní AI služby • podpora pro integraci externích služeb AI
Zaměření na školení a podporu
• Tutoriály (manuály, screencasty, atd.) • (online) kurzy • externí poskytovatelé služeb
Zaměření na ekonomiku (kriteria tištěná šedou nejsou pokryta)
• náklady na používání/vlastnictví (pořízení, licence a podobně) • dostupnost vyškolených vývojářů • příspěvek k produktivitě vývoje • ochrana investic

Tabulka 4.6: Další aspekty analýzy low-code platforem

4.0.0.1 HCL Software: Profil poskytovatele služby

Produktové portfolio. Společnost HCL Software (HCL) nabízí hned několik produktů určených pro vývoj low-code aplikací. HCL Volt MX je low-code platforma pro vývoj jak frontendu tak backendu. Společnost pak dále nabízí další dva nástroje. První je HCL Leap určený pro neodborné vývojáře. Druhou pak HCL Domino Leap sdílející stejný základní zdrojový kód s HCL Leap s tím rozdílem, že HCL Domino Leap poskytuje další funkcionality spojené s Domino platformou [31].

Původ produktu. Platforma HCL Domino vznikla pod názvem Lotus Notes ve společnosti Lotus Software. Tu koupila v roce 1995 společnost IBM a přejmenovala na IBM Notes a vytvořila aplikaci IBM Domino určenou pro servery [32]. Na základech IBM Domina byla postavena platforma s dnešním názvem IBM Domino Leap. Ve společnosti IBM dále pak vznikl IBM Forms Expirence Builder, který se později vyvinul do platformy HCL Leap. V roce 2019 pak firma HCL dokončuje akvizici části softwarových produktů od společnosti IBM společností HCL. [33]. V roce 2020 byla společností HCL představena platforma HCL Volt MX [34]. Vývoj a názvy platforem pak ilustruje diagram viz. 4.1.

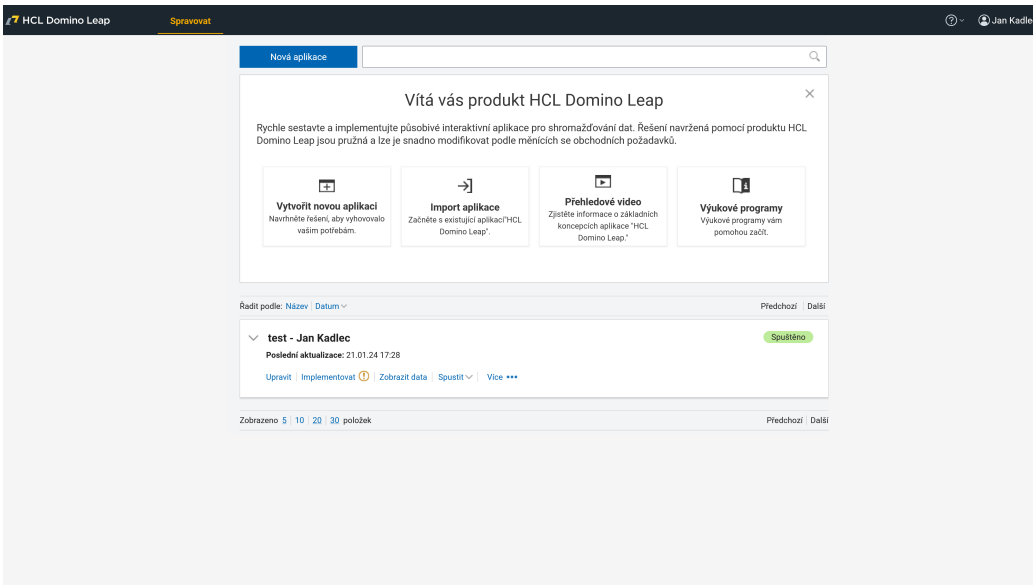
Obrázek 4.1: Vývoj jednotlivých low-code platforem společnosti HCL (dříve IBM)

Zaměřeno na Marketing - HCL Domino Leap. U produktu HCL Domino Leap se používají pojmy low-code a no-code bez rozdílu. V některých případech je platforma popisována jako low-code, v jiných zase jako no-code. Dále na svých webových stránkách určených tomuto produktu jsou uváděny jako hlavní benefity „Silné no-code schopnosti“, možnost tvořit aplikace až o 70% rychleji, a to, že je platforma postavena na Dominu [35]. Mezi další přínosy pak podle odkazované brožurky na webové stránce patří zlepšená agilita, snížené náklady, rychlejší transformace a vylepšená zákaznická zkušenost. V případě dalšího z produktů *HCL Volt MX* jsou jako výhody zmiňovány možnost zapojení všech zaměstnanců na řešení a spolupráci na řešení jejich digitálních výzev v bezpečném a spravovaném prostředí. Jako další výhoda je pak uváděna možnost být v čele inovací a podílet se na vytváření digitálních zážitků pro uživatele, a to od superaplikací (do jednoho ekosystému sdružené aplikace různých služeb [36] [37]) až po umělou inteligenci. Jako jeden z hlavních benefitů je uváděna účinnost doručování aplikací, která je podle webových stránek až o 60% větší než u podobných služeb s bezbackendovou komplexitou [38]. Poslední platformou zaměřenou na low-code je pak *HCL Leap* nabízející podobné vlastnosti jako *HCL Domino Leap*: Důvodem je, že HCL Domino Leap je oproti HCL Leap rozšířen o prvky určené pro zákazníky HCL Domino [31]. Jejich marketingová prohlášení jsou tedy v mnohém velmi podobná. Oproti HCL Domino Leap pak vyzdvihuje vlastnosti jako jsou dynamické pracovní postupy, přístup k aplikačnímu rozhraní (API) a vhodnosti pro mobilní zařízení [39].

Produktové Portfolio
• HCL Software nabízí několik produktů pro vývoj low-code aplikací. • HCL Volt MX: Low-code platforma pro vývoj frontendu i backendu. • HCL Leap: Určeno pro neodborné vývojáře. • HCL Domino Leap: Sdílí základní zdrojový kód s HCL Leap a poskytuje další funkcionality spojené s Domino platformou.
Původ produktu
• Platforma HCL Domino vznikla jako Lotus Notes ve společnosti Lotus Software. • IBM koupila Lotus Software v roce 1995 a přejmenovala produkt na IBM Notes a vytvořila IBM Domino pro servery. • Na základech IBM Domino byla postavena platforma IBM Domino Leap. • IBM Forms Experience Builder se později vyvinul do platformy HCL Leap. • V roce 2019 firma HCL dokončuje akvizici softwarových produktů od IBM. • V roce 2020 byla představena platforma HCL Volt MX.
Zaměřeno na marketing
• Pojmy low-code a no-code jsou používány bez rozdílu. • Hlavní benefity: "Silné no-code schopnosti", možnost tvořit aplikace až o 70% rychleji, založeno na Domino platformě. • Další přínosy: Zlepšená agilita, snížené náklady, rychlejší transformace, vylepšená zákaznická zkušenost. • HCL Volt MX: Zapojení všech zaměstnanců, spolupráce na digitálních výzvách, v čele inovací, účinnost doručování aplikací o 60% vyšší. • HCL Leap: Nabízí podobné vlastnosti jako HCL Domino Leap, s rozšířením o prvky pro zákazníky HCL Domino, vyzdvihuje dynamické pracovní postupy, přístup k API a vhodnost pro mobilní zařízení.

4.0.0.2 HCL Domino Leap: Analýza funkcí platformy

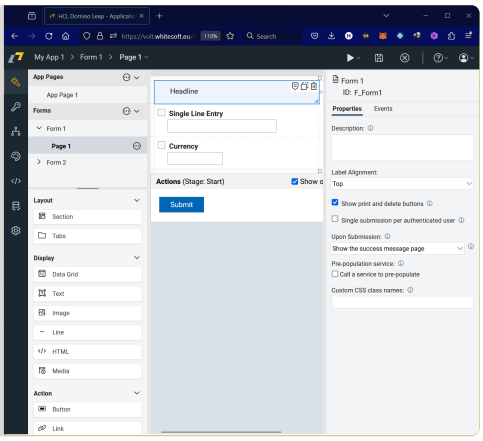
Domovská obrazovka HCL Domino Leap nabízí uživateli přehled jednotlivých aplikací. Dále pak umožňuje vytvořit novou aplikaci podle souboru Microsoft Excel, případně aplikaci importovat z proprietárního souboru s koncovkou `.dleap`. Na kartě náležející dané aplikaci pak můžeme vidět jednotlivé akce pro danou aplikaci, jmenovitě „Upravit“, „Implementovat“, „Zobrazit data“, „Spustit“, „Duplikovat“, „Odstranit“, „Exportovat“, „Přejít na vyšší verzi“ a „Historie“. Každá z aplikací se pak skládá ze stránek „Návrh“, „Přístup“, „Sled operací“, „Styl“, „Události“, „Ověření“ a „Nastavení“.



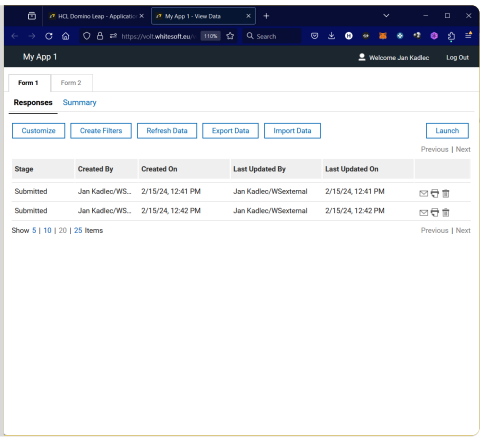
Obrázek 4.2: Výchozí obrazovka s přehledem aplikací

Statická perspektiva. Každá aplikace sestává z jednoho a více formulářů a případných pod-formulářů, zakomponovaných v jiném formuláři. Formuláře jsou prvky grafického uživatelského rozhraní (GUI). Každý formulář a pod-formulář pak odpovídá jednomu objektu, sestávajícímu z jednotlivých atributů s příslušnými datovými typy. Datových typů na serverové části je celkem šest a jsou jimi řetězec znaků, pravdivostní hodnota (pravda nebo nepravda), datum, čas, časová značka a číslo. Ten je pak následně ukládán do databáze přímo platformou. Platforma tak plní funkci objektově-relačního mapování objektu, jenž mapuje prvek grafického rozhraní na entitu v databázi. Objekt je reprezentován pomocí formuláře. Objektově relační mapování je nepřístupné a nemůže se tedy do něj zasahovat. Data jsou ukládána do Domino databáze. Příklad mapování a implementace pak můžeme vidět na obrázku 4.4. Formuláře mohou čerpat data z interních aplikací prostřednictvím takzvaných servisů, jimiž mohou být jiné aplikace na platformě Domino, avšak na stejném serveru. Servisy dále umožňují napojení na ostatní platformy pomocí API, Servletů (třídy obsluhující HTTP požadavky [40]) a webových služeb (web service).

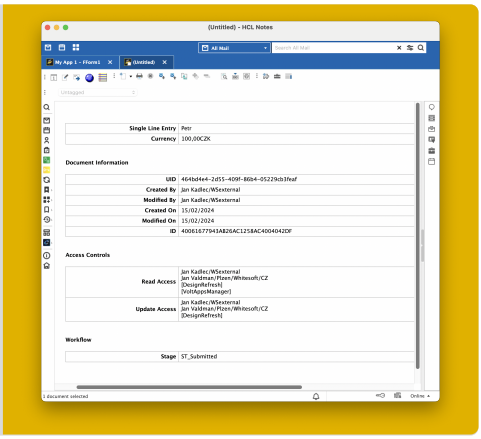
Reprezentace v grafickém rozhraní editoru



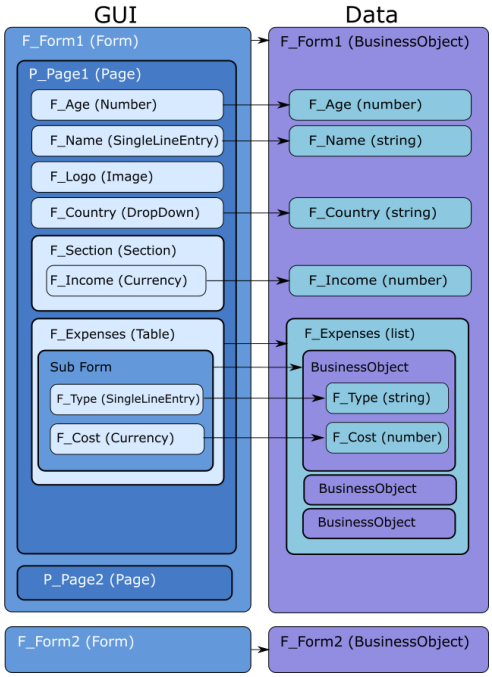
Reprezentace v náhledu na odeslaná data



Reprezentace v aplikaci HCL Notes



Obrázek 4.3: Zobrazení stejných dat v různých částech aplikace



Obrázek 4.4: Mapování prvků grafického rozhraní na data [41]

Zaměření na Reprezentace		
Konceptuální reprezentace	*Jazyky*	*Komponenty*
• datový model reprezentovaný prostřednictvím formuláře	• proprietární jazyk	• editor modelů a formulářů zároveň • NoSQL databáze • JavaSript ve vrstvě frontend
Zaměření na Opětovné Použití a Adaptabilitu		
Generické abstrakce	*Jazykové koncepty*	*Přístup k externím zdrojům a k dokumentům na úrovni implementace*
• obecné datové typy pro řetězce, čísla atd. • některé obecné typy obchodních dat jako "adresa"nebo "telefon" *Komentář:* reprezentováno jednou nebo více koncepčními reprezentacemi uvedenými v nejlevějším sloupci výše	• pouze one-to-one vztahy v DB	• žádný přístup k dokumentům (kódu) na úrovni implementace • persistence na platformě Domino
Zaměření na Integraci		
Běžné statické abstrakce napříč škálou aplikací	Datové modely nelze sdílet mezi aplikacemi HCL Domino Leap.	
Přístup k běžným datovým repozitářům napříč škálou aplikací	Datový model každé aplikace je spravován samostatně. Uživatelé HCL Domino Leap nemají přístup ke společnému úložišti dat.	

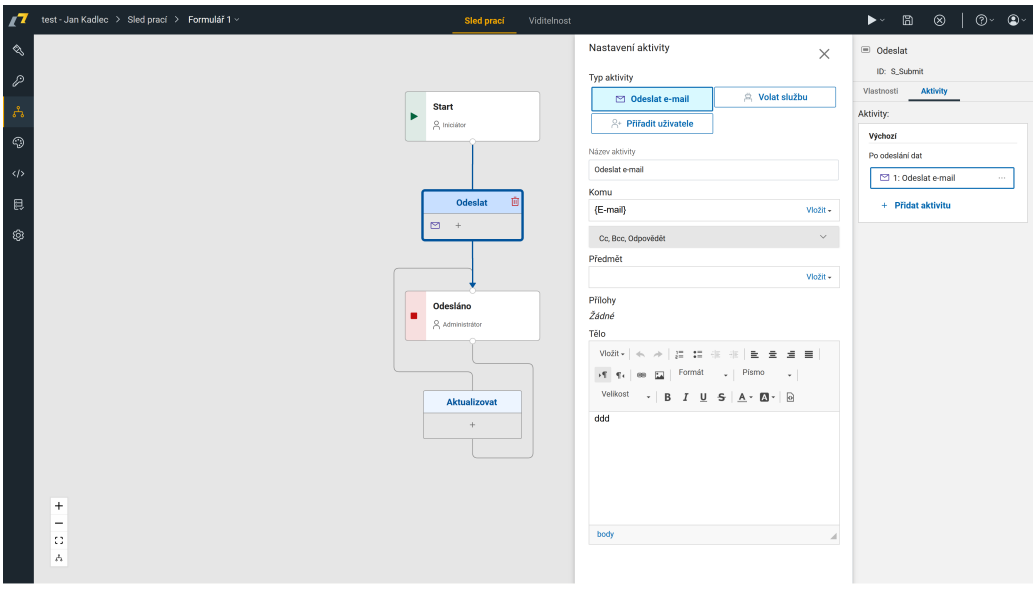
Tabulka 4.7: HCL Domino Leap statická perspektiva

Funkční perspektiva. HCL Domino Leap umožňuje spouštění vlastních funkcí napsaných v jazyce JavaScript. HCL Domino Leap nabízí pět kategorií událostí. Jsou jimi aplikační události, formulářové události, události na stránce, události na stránce aplikace a události prvku formuláře. Ke spuštění funkcí dochází při události, která nastala ve webovém prohlížeči. To mohou být události na úrovni aplikace, formuláře, stránky, prvku stránky v odpovědi na změnu stavu formuláře či uživatelské interakce. Funkce mohou být naprogramované přímo v aplikaci na několika místech např.: u formulářových prvků či v záložce „události“ ve vývojovém rozhraní. Další možností jak definovat funkce je pak přímo v souborech nahraných do aplikace a nastavit zavolání funkce ze souboru po nastání události. Při tvorbě skriptů nám aplikace nabízí datové objekty umožňují manipulaci s prvky na stránce. Manipulace s prvky je možná pomocí funkcí např.: `app.isSingleFormView()`, kdy `app` je datovým objektem obsahují data o aktuální aplikaci, kterou vytváříme a `isSingleFormView()` je funkce zavolaná nad objektem. Dále pak aplikace nabízí spuštění grafického rozhraní usnadňujícího výběr dat z formuláře pomocí klávesové zkratky `CTRL + MEZERNÍK`. Po kliknutí se nám vloží odkaz na proměnou obsahující data formuláře.

Zaměření na Reprezentace		
Konceptuální reprezentace	*jazyky*	*Komponenty*
• není k dispozici žádná konceptuální reprezentace funkcí	• nepřístupné	• funkce jsou specifikovány podle nabízených obrazovek grafického uživatelského rozhraní HCL Domino Leap, případně v dedikovaném souboru
Zaměření na opětovné použití a adaptabilitu		
Referenční abstrakce	*Jazykové koncepty*	*Přístup k externím zdrojům a k dokumentům na úrovni implementace*
• předimplementované obecné operace zaměřené na databázi	• nedostupné	• nelze přistupovat k dokumentům (kódu ...) na úrovni implementace • přístup k omezenému souboru externích funkcí

Tabulka 4.8: Shrnutí funkční perspektivy HCL Domino Leap

Dynamická perspektiva. Aplikace je procesně orientovaná. Aplikace má stránku s názvem „Sled prací“ určenou pro editaci workflow aplikace. Workflow je schéma provádění nějaké komplexnější činnosti, rozepsané na jednodušší činnosti a jejich vazby. HCL Domino Leap přistupuje k zobrazení pracovních postupů pomocí konceptu fází. Fáze mají několik důležitých vlastností. Fáze popisuje chování formuláře v době, kdy se nachází v určitém stavu (tj. jaké části formuláře jsou viditelné a/nebo pouze pro čtení, když je formulář v této fázi). Fáze popisuje tok, tedy to, do jakých dalších fází lze z této fáze přecházet. Fáze také obsahuje informace o tom, kdo má přístupová práva k formuláři, zatímco se formulář nachází v tomto stavu. Příkladem je pak schvalovací proces s fázemi jako „Požadavek na schválení“, „Čekání na schválení“ a „Schváleno“. Dále pak HCL Domino Leap poskytuje různé akce po odeslání formuláře. Těmi mohou být akce jako odeslání emailu, zavolání služby nebo přidání uživatele. Zároveň aplikace vždy zobrazí stránku informující uživatele o úspěšnosti odeslání dat. Každá aplikace má ve výchozím stavu tři fáze a to „Začátek“, „Schváleno“ a „Odesláno“ (viz. snímek obrazovky 4.5).



Obrázek 4.5: Workflow v HCL Domino Leap

Zaměření na Reprezentace		
Konceptuální reprezentace	*Jazyky*	*Komponenty*
• obecné modely procesů	• proprietární	• editor schémat
Zaměření na opětovné použití a adaptabilitu		
Referenční abstrakce	*Jazykové koncepty*	*Přístup k externím zdrojům a k dokumentům na úrovni implementace*
• nejsou k dispozici žádné referenční postupy	• fáze a akce	• nelze přistupovat k dokumentům na úrovni implementace

Tabulka 4.9: Shrnutí dynamické perspektivy HCL Domino Leap

Vývoj grafického rozhraní. HCL Domino Leap nabízí dvě stránky, ve kterých je možno editovat grafické rozhraní. Jsou jimi stránky „Návrh“ a „Styl“. Stránka „Návrh“ nabízí možnost přidávání stránek, formulářů a stránek formulářů. Dále pak umožňuje tvorbu formulářů pomocí „drag and drop“ (přetáhni a pusť). Drag and drop slouží k přetahování jednotlivých komponent formulářů na stránku formuláře. Komponenty jsou prvky formuláře, nebo rozvržení jako odkaz, tlačítko, e-mail, sekce, karta nebo heslo. U jednotlivých komponent pak můžeme definovat vlastní kaskádový styl (CSS) pomocí zadání názvu třídy definované na stránce „Styl“. Stránka „Styl“ nám dává možnost vybrat si jeden z předefinovaných motivů. Další variantou jak upravit uživatelské rozhraní je vložení vlastního CSS stylu pomocí odkazu na stránku se souborem, nahráním přímo souboru, nebo editaci CSS stylu přímo v aplikaci prostřednictvím textového pole. Spolu s CSS je také možnost přidávat vlastní fonty.

Perspektiva interakce s uživatelem		
Konceptuální reprezentace	*Komponenty*	*Architektonické aspekty*
• Stránky grafického uživatelského rozhraní jako "karta"aplikace	• Editor grafického uživatelského rozhraní • Navigační panel není možné nastavit	• více typů rolí je přiřazeno k jedné (systémem definované) stránce GUI s různými přístupovými právy a naopak • implementace vzoru MVC, přičemž aspekt „controller“ je do značné míry nepřístupný a neupravitelný
Zaměření na opětovné použití a adaptabilitu		
Referenční abstrakce	*Adaptabilita*	
• aplikace lze přizpůsobit	• pole lze uspořádat a upravit, stejně tak akce na stránce	

Tabulka 4.10: Koncepty pro analýzu podpory vývoje grafického uživatelského rozhraní

Další aspekty. Nasazené HCL Domino Leap aplikace jsou přístupné pouze v rámci samotné platformy. Společnostem je poskytnuta URL adresa prostřednictvím které je možno přistoupit na stránku s aplikací. HCL Domino Leap umožňuje export dat z aplikace dvěma způsoby. Prvním způsobem je export do formátu tabulkového procesoru Microsoft Excel, druhým pak prostřednictvím formátu určeného pro semistrukturovaná data s názvem Extensible Markup Language (XML). Jaká data budou exportována si může uživatel upravit podle svých potřeb. Domino Leap používá přizpůsobitelný model založený na rolích, který definuje, kdo má přístup k datům a kdo může aplikaci upravovat. Role umožňují přidělovat oprávnění přístupu k datům a k údržbě aplikací. Jednotlivci nebo skupiny jsou pak k rolím přiřazeni pomocí přístupové komponenty nebo programově prostřednictvím webových služeb. Třemi základními rolemi jsou: Role „Administrátor“, poskytující administrátorská oprávnění pro správu aplikace, role „Iniciátor“, umožňující zahájení vyplňování formuláře a jeho odeslání. Příkladem může být požadavek na firemní benefit v podobě vánočního stromku, o který si můžou zažádat pouze zaměstnanci dané firmy. Poslední předdefinovanou rolí je pak „Vlastník záznamu“ náležející uživateli, jenž záznam vytvořil, tedy vyplnil formulář. Dále aplikace nabízí tvůrci aplikace definovat vlastní role odpovídající potřebám organizace. Příkladem může být schvalovací proces ve kterém je potřeba mít různé role pro určité fáze schvalovacího procesu. Příkladem může být proces nástupu do práce a přiřazení přístupové karty, kdy oddělení lidských zdrojů podá žádost o vystavení karty. Administrativní oddělení

pak kartu vystaví a do systému napíše číslo karty. Oddělení lidských zdrojů pak kartu předá dotyčnému a označí ji v systému za vydanou. V tomto scénáři by byly fáze „Podáno“, „Vystaveno“ a „Vydáno“. Role by pak mohli být „Pracovník lidských zdrojů“ a „Pracovník administrace“ a tři již před-připravené role. Když se formulář nachází v určité fázi, má administrátor možnost tvůrce aplikace u každé role upravovat přístupová práva pro každou jednotlivou fázi. Přístupová práva pro každou fázi jsou specifikována podle „vytvořit, číst, editovat, smazat“ (CRUD) schématu. CRUD oprávnění se pak specifikují pro každou roli v aplikaci samostatně. Aplikace se může exportovat do souboru ve formátu `dleap` umožňující uchovávat všechna data o aplikaci samotné tak i data odeslaná prostřednictvím aplikace.

HCL Domino Leap nenabízí v rámci platformy žádné služby umělé inteligence ani integraci na externí služby umělé inteligence.

Přístupnost a pohodlí používání	
• žádná specifická metodická podpora vývoje • stránky grafického uživatelského rozhraní lze upravit podle různých uživatelských rolí • použití do značné míry nepřístupných, proprietárních modelovacích jazyků	
Způsoby použití	
Platforma (tj. vývoj aplikací)	*Aplikace*
• specifikace práv na základě CRUD • žádná podpora pro kolaborativní vývoj aplikací	• Specifikace přístupových práv založená na CRUD • omezené možnosti integrace mezi aplikacemi • žádná podpora pro kolaborativní využití platformy
Zaměření na nasazení a škálovatelnost	
Cílová umístění	*Podpora*
• cloudová platforma přístupná přes běžný webový prohlížeč	• vnější přístupnost prostřednictvím rozhraní RESTful API
Zaměření na umělou inteligenci	
• žádné služby AI zahrnuté v HCL Domino Leap	

Tabulka 4.11: Další aspekty analýzy low-code platforem

4.0.1 HCL Domino Leap: Shrnutí

Zaměření na oblasti vývoje aplikací. Ačkoli HCL Domino Leap nabízí samostatné funkce pro statické, funkční a dynamické aspekty vývoje aplikací, platforma se jed-

noznačně zaměřuje na statické aspekty a návrh grafického uživatelského rozhraní. Provádění uživatelsky definovaných funkcí je omezeno na události definované systémem. Platforma však nabízí běžné koncepty modelování pracovních postupů. Je možné přiřazovat uživatele k úlohám a sledovat pracovní postupy za běhu. Bohužel ale není řešen společný vývoj mezi vícero vývojáři najednou.

Poskytování abstrakcí. Abstrakce specifické pro danou doménu jsou v aplikaci HCL Domino Leap zakotveny jen zřídka, a to prostřednictvím poskytování některých obecných datových typů orientovaných na podnikání, jako je měna. Zvýšení produktivity je realizováno prostřednictvím vymizení implementačně specifických zvláštností. Příkladem může být editor vývojových diagramů nebo editor grafického uživatelského rozhraní typu drag-and-drop. Uživatelé platformy HCL Domino Leap nemají přístup k žádným základním implementačním dokumentům (např. aplikační kód) ani je nemohou upravovat.

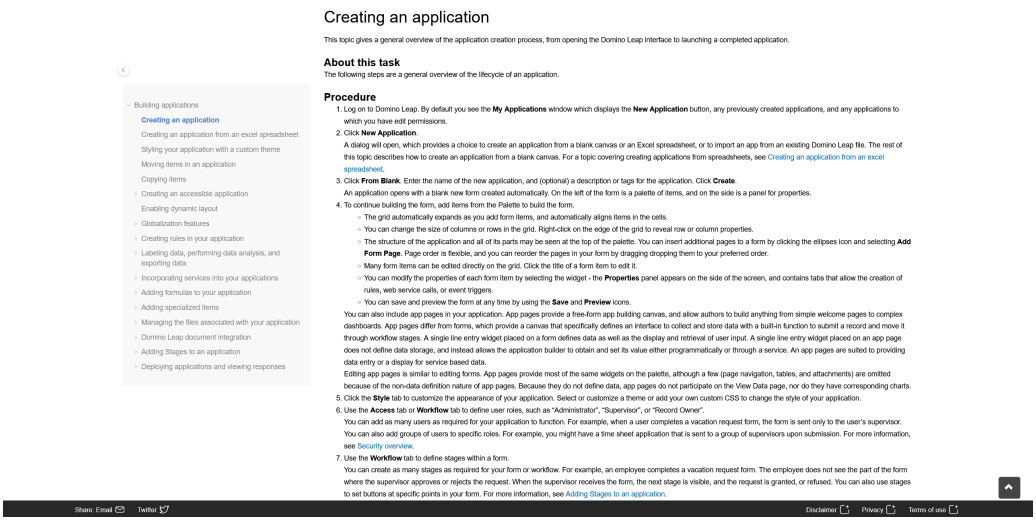
Úloha IT odborníků. Společnost HCL na svých webových stránkách výslovně inzeruje, že pro vývoj na platformě nejsou potřeba žádné speciální Domino nebo IT dovednosti. Zatímco většina funkcí uvažovaných v této analýze se zdá být obecně přístupná laickým vývojářům, jako výjimku zde lze uvést specifikaci aplikačních skriptů.

On-line návody 5

Pro práci s HCL Domino Leap existují čtyři zdroje informací, a to dokumentace, wiki, YouTube tutoriál a fórum. V této kapitole se budeme zabývat jejich analýzou.

5.1 HCL Domino Leap dokumentace

HCL Domino Leap dokumentace dostupná na adrese `https://help.hcltechsw.com/domino-leap/1.1.4/index.html` poskytuje dobrý přehled v referenci o metodách, objektech a událostech (events) této platformy. Nevýhodou je nepřehlednost způsobená špatnou strukturou. To má za následek, že daná informace se může v dokumentaci nacházet, uživatel ji ale nemusí nalézt. Dokumentace dále obsahuje návody (viz.) například na to, jak vytvořit aplikaci. Nevýhodou takovýchto návodů je to, že neobsahují téměř žádné obrázky a pro uživatele je pak velmi náročné si takto podané informace osvojit a vizuálně propojit.



Creating an application

This topic gives a general overview of the application creation process, from opening the Domino Leap interface to launching a completed application.

About this task

The following steps are a general overview of the lifecycle of an application.

Procedure

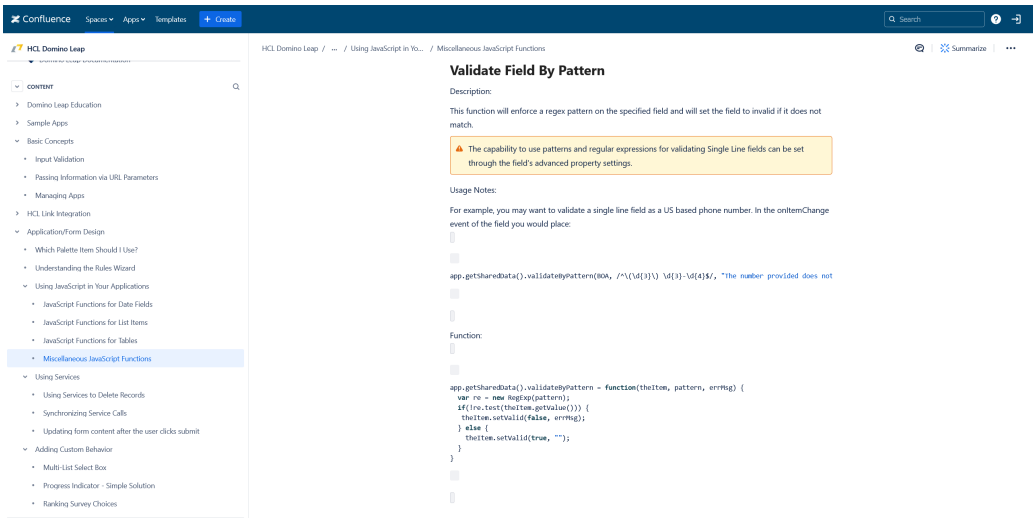
1. Log on to Domino Leap. By default you see the **My Applications** window which displays the **New Application** button, any previously created applications, and any applications to which you have edit permissions.
2. Click **New Application**.
   A dialog will open, which provides a choice to create an application from a blank canvas or an Excel spreadsheet, or to import an app from an existing Domino Leap file. The rest of this topic describes how to create an application from a blank canvas. For a topic covering creating applications from spreadsheets, see Creating an application from an excel spreadsheet.
3. Click **From Blank**. Enter the name of the new application, and (optional) a description or tags for the application. Click **Create**.
   An application opens with a blank new form created automatically. On the left of the form is a palette of items, and on the side is a panel for properties.
4. To continue building the form, add items from the Palette to build the form.
   - The grid automatically expands as you add form items, and automatically aligns items in the cells.
   - You can change the size of columns or rows in the grid. Right-click on the edge of the grid to reveal row or column properties.
   - The structure of the application and all of its parts may be seen at the top of the palette. You can insert additional pages to a form by clicking the ellipses icon and selecting **Add Form Page**. Page order is flexible, and you can reorder the pages in your form by dragging dropping them to your preferred order.
   - Many form items can be edited directly on the grid. Click the title of a form item to edit it.
   - You can modify the properties of each form item by selecting the widget - the **Properties** panel appears on the side of the screen, and contains tabs that allow the creation of rules, web service calls, or event triggers.
   - You can save and preview the form at any time by using the **Save** and **Preview** icons.

   You can also include app pages in your application. App pages provide a free-form app building canvas, and allow authors to build anything from simple welcome pages to complex dashboards. App pages differ from forms, which provide a canvas that specifically defines an interface to collect and store data with a built-in function to submit a record and move it through workflow stages. A single line entry widget placed on a form defines data as well as the display and retrieval of user input. A single line entry widget placed on an app page does not define data storage, and instead allows the application builder to obtain and set its value either programmatically or through a service. An app pages are suited to providing data entry or a display for service based data.
   Editing app pages is similar to editing forms. App pages provide most of the same widgets on the palette, although a few (page navigation, tables, and attachments) are omitted because of the non-data definition nature of app pages. Because they do not define data, app pages do not participate on the View Data page, nor do they have corresponding charts.
5. Click the **Style** tab to customize the appearance of your application. Select or customize a theme or add your own custom CSS to change the style of your application.
6. Use the **Access** tab or **Workflow** tab to define user roles, such as "Administrator", "Supervisor", or "Record Owner".
   You can add as many users as required for your application to function. For example, when a user completes a vacation request form, the form is sent only to the user's supervisor.
   You can also add groups of users to specific roles. For example, you might have a time sheet application that is sent to a group of supervisors upon submission. For more information, see Security overview.
7. Use the **Workflow** tab to define stages within a form.
   You can create as many stages as required for your form or workflow. For example, an employee completes a vacation request form. The employee does not see the part of the form where the supervisor approves or rejects the request. When the supervisor receives the form, the next stage is visible, and the request is granted, or refused. You can also use stages to set buttons at specific points in your form. For more information, see Adding Stages to an application.

Obrázek 5.1: Ukázka návodu popisujícího tvorbu aplikace na platformě HCL Domino Leap

5.2 HCL Domino Leap wiki

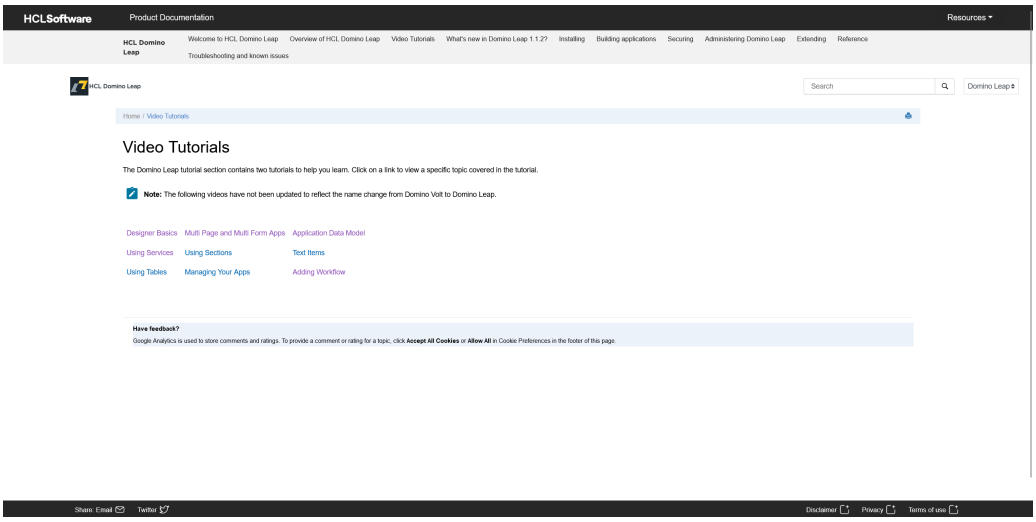
HCL Domino Leap wiki je postavená na databázi znalostí Confluence od společnosti Atlassian. Tato wiki je dostupná na webové stránce `https://hclwiki.atlassian.net/wiki/spaces/HDV/overview`. Výhodou wiki je velké množství materiálů, bohužel však rozdílné kvality a detailu. V některých případech je zde pouze nahraná výsledná aplikace, v jiných zas velmi podrobný popis. Zároveň tyto materiály nejsou v podstatě nijak logicky uspořádány a orientace v nich je chaotická. Jednotlivé stránky na sebe obsahově nenavazují. Pokud bychom se zaměřily na stránky samotné, tak formátování textu poskytují lepší, než je v dokumentaci, speciálně pak zvýrazňovaní syntaxe kódu je lepší, avšak k ideálu má stále daleko. Neumí například graficky odlišovat objekty a metody a nad úseky kódu přidává zvláštní čtverce (viz. obrázek 5.2).



Obrázek 5.2: Ukázka stránky vysvětlující validaci vstupu od uživatele pomocí regulárního výrazu

5.3 YouTube tutoriál

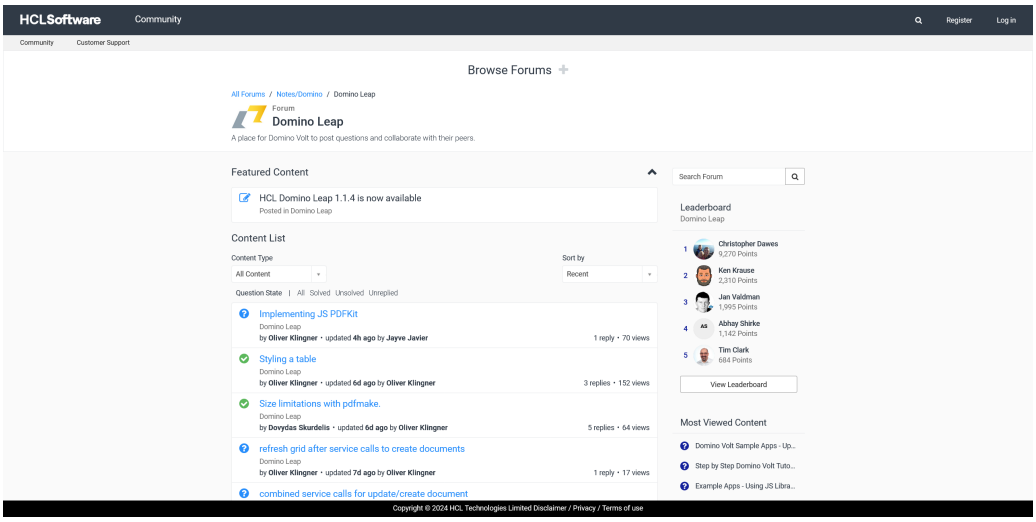
Pro platformu HCL Domino Leap byl vytvořen tutoriál, jež ukazuje, jak udělat pár základních věcí na již zmiňované platformě. Nevýhodou je to, že video je sdílené jako soukromé a není možné ho najít skrze vyhledávače jako je Google nebo Bing, protože ty soukromé video neindexují. Proto aby ho tedy uživatel našel, musí se prokliknout skrze jeden z odkazů uvedených v HCL Domino Leap dokumentaci. Nevýhodou tohoto videa je, že byť obsahuje vícero částí, jedná se o jedno dlouhé video a na jednotlivé části videa je pak odkazováno právě z dokumentace.



Obrázek 5.3: Odkazy do různých částí video tutoriálu

5.4 HCL Domino Leap fórum

Posledním zdrojem informaci je HCL Domino Leap fórum dostupné na webu poskytovaného pro tyto účely společností HCL Software. To obsahuje problémy nebo nejasnosti, na které narazili ostatní uživatelé platformy. Pro snazší nalezení potřebné informace obsahuje HCL Domino Leap funkci vyhledávání (viz obrázek 5.4. Zároveň je zde možnost pokládat dotazy a vzhledem k aktivní komunitě se dá očekávat brzká odpověď.



Obrázek 5.4: HCL Domino Leap fórum

Realizace tutoriálu 6

6.1 Výběr nástrojů pro realizaci

Cílem práce bylo vytvořit tutoriál na způsob již známých řešení jako je W3Schools. W3Schools je webová stránka v anglickém jazyce zaměřená na tutoriály týkající se problematiky webového vývoje a programovacích jazyků. W3School se zaměřuje na jednoduchost, snadno-pochopitelné tutoriály, příklady kódu a interaktivní vývojové prostředí nazvané "Try it Yourself". Pro tvorbu tutoriálu se zvažovaly dva možné způsoby řešení. Prvním možným způsobem řešení je přistoupení k vlastní implementaci, druhým pak využití již hotových nástrojů vytvořených pro tento účel.

Vlastní řešení. Vlastní řešení nabízí mnoho výhod. Hlavními jsou především možnost téměř neomezených úprav částí jako je uživatelské rozhraní, forma obsahu a další jiné funkcionality umožňující zlepšit zážitek z učení. Výhodou, ale zároveň i nevýhodou může být odlišnost od již existujících řešení. Může totiž znamenat zmatení uživatelů, což se eliminuje v případě již známého rozložení. Nevýhodami jsou pak zvýšené časové náklady spojené s tvorbou řešení. V takových případech může navíc dojít k nežádoucí situaci, kdy zůstane málo času na tvorbu obsahu samotného.

Dostupné nástroje. Jelikož tvorba tutoriálu je téměř neodmyslitelnou součástí každého nástroje či služby, existuje již na trhu velké množství ať již placených či neplacených řešení. Část neplacených řešení je navíc nabízena jako svobodný a otevřený software (FOSS). Výhodou využití již dostupných nástrojů je především značná časová úspora, možnost rychlého nastavení a jednoduchost použití. Nevýhodou by se mohla zdát nedostatečná možnost úprav. Tento problém však řeší nástroje poskytující zdrojový kód a licenci umožňující kód neomezeně upravovat a distribuovat (příkladem Astro Starlight [42]). Další výhodou již připravených řešení v porovnání s vlastním řešením je odpadnutí potenciálních problémů objevených až v průběhu řešení.

Z porovnání výše zmíněných přístupů k tvorbě tutoriálů vyplynula jako lepší

volba již existujícího nástroje. Především díky časovým limitům na projekt a množství předpokládaného času potřebného na tutoriál samotný. V další části se tedy budeme zabývat výběrem vhodného, již existujícího nástroje.

6.2 Výběr vhodného značkovacího jazyka

Pro psaní textů se typicky používají značkovací jazyky. Značkovací jazyk (markup language) je systém pro anotaci dokumentu, který se řídí syntaktickými pravidly pro formátování textu a dat. Značkovací jazyky nejsou primárně určeny k tomu, aby byly zobrazeny ve finální podobě, ale slouží k definování struktury a prezentace textových dat.

Pro potřeby tutoriálu je jako nejlepší deskriptivní jazyk Markdown obohacený o jazyk HTML. Důvodem je především široké použití Markdownu téměř ve všech novodobých nástrojích pro tvorbu dokumentace, tutoriálů atd.. Existují nástroje, které umožňují využití obou těchto jazyků zároveň. Kombinace těchto jazyků by se dala označit za zlatý standard pro tvorbu dokumentací, blogových článků a tutoriálů. Příklady takových jsou nástroje jako VuePress [43] nebo Docusaurus [44]. Princip práce na tutoriálu je takový, že autor tutoriálu standardně používá Markdown, v okamžiku, kdy však nastane situace, kdy Markdown nestačí, může autor použít HTML.

6.3 Výběr z již dostupných řešení

Z důvodu velkého množství dostupných řešení je potřeba množinu nástrojů pro výběr omezit. Logicky je dobré se omezit na množinu technologií známou tvůrci tutoriálu. Tato množina kromě klasických technologií jako je HTML, CSS a JavaSctipt zahrnuje také JavaScript framework Vue.js (zkráceně Vue).

6.3.1 Jednotlivé nástroje

VuePress. VuePress je generátor statických stránek navrhnutý speciálně pro Vue.js. VuePress nabízí více funkcionalit než např.: VitePress, výměnou za navýšení komplexnosti. Další nevýhodou VuePressu je že není příliš udržovaný. VuePress bohužel nenabízí funkcionalitu zvanou Hot Module Replacement (HMR) umožňující automatickou aktualizaci různých modulů za běhu aplikace, je tedy potřeba po každé změně v kódu spouštět proces sestavení. Tato vlastnost se používá pouze pro vývoj [43].

Nuxt.js. Nuxt.js je framework postavený nad Vue.js určený pro tvorbu jakýchkoli aplikací. Svým primárním účelem to není generátor statických stránek, má však

schopnost generovat statické stránky (SSG). Nuxt.js má mnoho funkcí, ale má i velké množství přednastavení, jež jsou vhodná pro velké aplikace. Pro jednoduché statické weby může být tato skutečnost spíše nežádoucí [45].

Gridsome. Gridsome je framework ze skupiny frameworků shourhně označovaných jako Jamstack. První tři písmena v názvu označují JavaScript, API a Markup. Jamstak tak odstraňuje potřebu aplikační logiky. Stejně jako Nuxt.js i Gridsome nabízí možnost statického generování stránek a HMR. Jeho výhodou je možnost získávat data z různých zdrojů. Nabízí uživatelům možnost využití již existují šablony. Bohužel pro tutoriál zatím žádnou šablonu nenabízí a tak by musela být speciálně pro tutoriál vytvořena [46].

VitePress. VitePress je generátor statických stránek stejně jako VuePress. Jak již název napovídá, využívá Vite, jež je lokální vývojový server. Kromě toho se Vite také stará o zabalení (bundling) všech souborů (HTML, CSS, JavaScript, ...) pro produkci. VitePress má funkcionalitu HMR poskytnutou právě Vitem. Jelikož je VitePress oproti VuePressu mladší, nabízí menší množství pluginů [47].

Srovnání nástrojů. Pro co nejlepší uživatelskou zkušenost s tutoriálem jsem nadefinoval několik kritérií, jež jsou vhodná, aby nástroj splňoval. Jsou jimi:

1. podpora hypertextu
2. možnost přidávání ukázek kódu
3. zvýraznění JavaSriptové syntaxe
4. možnost tvořit Step-by-step tutoriály
5. podpora komunitní spolupráce
6. podpora napojení na verzovací systém
7. náročnost použití
8. vyhledávání
9. Hot Module Replacement (HMR)

Z výše uvedených jsem vybral VitePress a to kvůli jednoduchosti a snadnému pochopení i pro začátečníka a podpoře HMR (viz. tabulka 6.1).

Kritérium	VuePress	Nuxt.js	Gridsome	Vitepress
1	ano	ano	ano	ano
2	ano	ano	ano	ano
3	ano	ano	ne	ano
4	ano	ano	ne	ano
5	ano	ano	ne	ano
6	ano	ano	ano	ano
7	střední	vysoká	vysoká	nízká
8	ano	ano	ne	ano
9	ne	ano	ano	ano

Tabulka 6.1: Srovnání nástrojů pro tvorbu tutoriálu

6.4 Nasazení aplikace

Pro nasazení a integraci změn v tutoriálu byl zvolen přístup softwarového inženýrství označovaný jako CI/CD, dále pak využití PaaS (Platform as a Service) pro hosting aplikace samotné.

Continuous Integration a Continuous Delivery. CI/CD, což jsou zkratky pro "Continuous Integration"a "Continuous Deployment"nebo "Continuous Delivery", představují metodiky softwarového inženýrství zaměřené na zlepšení a automatizaci procesů vývoje softwaru.

Continuous Integration (CI) je přístup, kde vývojáři pravidelně (často několikrát denně) integrují svůj kód do sdíleného úložiště. Každá integrace je automaticky testována pomocí různých automatizovaných testů, což umožňuje rychle identifikovat a opravit chyby a udržovat kvalitu softwaru na vysoké úrovni. Tento přístup zajišťuje, že softwarový produkt je stále ve stavu, kdy může být efektivně vydán, což výrazně snižuje rizika spojená s „integrací velkých změn“ ve větších časových intervalech.

Continuous Deployment a Continuous Delivery (CD) jsou postupy, které následují po CI a zaměřují se na automatické doručování nebo nasazování integrovaného kódu do produkčního prostředí. Continuous Delivery zahrnuje automatické nasazení všech kódových změn do testovacího nebo produkčního prostředí po úspěšné CI, ale obvykle vyžaduje manuální schválení k vydání. Continuous Deployment jde ještě dál a minimalizuje lidské zásahy, když každá změna, která prošla testy CI, je automaticky nasazena do produkce.

Platform as a Service. Platform as a Service (PaaS), česky platforma jako služba je model cloud computingu, který poskytuje sadu nástrojů a služeb navržených tak, aby umožnil vývojářům snadno a efektivně vytvářet, nasazovat a spravovat aplikace

bez nutnosti starat se o základní infrastrukturu, jako jsou servery, sítě, úložiště a databáze, které jsou nezbytné pro běh aplikace.

PaaS je založen na konceptu virtualizace zdrojů, kde jsou tyto zdroje poskytovány jako služba přes internet. To umožňuje vývojářům pracovat na aplikacích s vyšší úrovní abstrakce, neboť fyzická infrastruktura a platformní vrstvy jsou spravovány poskytovatelem služby. Výhody PaaS zahrnují snížení nákladů na hardware, zjednodušenou správu infrastruktury, snadné škálování aplikací a podporu pro kolaborativní vývojové procesy.

PaaS nabízí také řadu vývojových nástrojů a komponent, jako jsou předkonfigurované frameworky (např. Django, Vue.js, ...), knihovny, databázové systémy, a také možnosti pro integraci s jinými službami a API. To výrazně zrychluje vývoj nových aplikací a umožňuje vývojářům soustředit se na inovace a kreativitu bez obav z omezení infrastruktury.

Příklady poskytovatelů PaaS zahrnují služby jako Google App Engine, Microsoft Azure, Heroku, Railway.app, Render a Red Hat OpenShift. Každý z těchto poskytovatelů nabízí různé sady služeb a nástrojů, které mohou lépe vyhovovat specifickým potřebám různých vývojových projektů.

Porovnání jednotlivých Paas . Pro tutoriál bylo tedy potřeba vybrat platformu, jež by splňovala tyto základní požadavky:

1. hosting statických stránek
2. podporu CI/CD
3. zdarma (v míře pro potřeby tutoriálu)

Kritérium	Railway.app	Render.com	Cloudflare Pages	Heroku	Netify
1	ano	ano	ano	ano	ano
2	ano	ano	ano	ano	ano
3	ne	ano	ano	ne	ano

Tabulka 6.2: Porovnání hostingových platforem

Protože pro hosting tutoriálu byly vhodné tři platformy (Render.com, Cloudflare Pages, Netify), rozhodl jsem se pro výběr platformy Render.com, jelikož jsem měl s ní už předchozí zkušenosti. Platforma Render.com (jako i jiné plaformy jako Netify nebo Railway.app) ke své funkci využívají Github Apps, jež je dobré si vysvětlit pro pochopení, jak celé nasazení tutoriálu probíhá.

GitHub Apps. GitHub Apps představují specializovaný typ integrace, který umožňuje externím službám interagovat s platformou GitHub na více úrovních. Tato

technologie poskytuje aplikacím specifická oprávnění pro přístup k úložištím (repos) a uživatelským datům, čímž zvyšuje efektivitu a bezpečnost při automatizaci různých procesů v softwarovém vývoji.

GitHub Apps jsou navrženy tak, aby sloužily jako vysoko konfigurovatelné nástroje, které mohou jednat autonomně nebo na popud uživatele. Tyto aplikace mohou být přizpůsobeny pro konkrétní úložiště nebo být použity napříč mnoha úložišti v rámci organizace. Významným aspektem je, že každá GitHub App má svou vlastní identitu na platformě, což zahrnuje samostatné uživatelské účty a možnost přihlášení do rozhraní API pod svým jménem.

Oprávnění GitHub Apps jsou velmi detailní a rozdělená podle jednotlivých úloh, což umožňuje správcům přesně definovat, co aplikace může a nemůže dělat. Například, aplikace může mít oprávnění pro čtení kódu, ale nikoli pro jeho změnu. Další oprávnění mohou zahrnovat správu problémů, pull requestů, wiki stránek a dalších [48].

Na rozdíl od tradičních OAuth aplikací, které fungují na základě rozsáhlých oprávnění uživatele, GitHub Apps poskytují více granulární kontrolu nad oprávněními. To redukuje riziko bezpečnostních problémů, protože aplikace může interagovat pouze s částmi platformy, ke kterým má explicitně přidělený přístup [48].

Vývojáři mohou vytvářet GitHub Apps pomocí GitHub API a konfigurovat je přes GitHub webové rozhraní nebo programově. Aplikace mohou být nainstalovány buď přímo uživatelem, nebo na úrovni organizace, kde mohou být sdíleny mezi více úložišti [48].

GitHub Apps zahrnují automatizaci rutinních úloh jako jsou testování kódu, integrace změn, správa závislostí, aktualizace dokumentace, a sledování stavu projektu. Tím, že automatizují tyto procesy, GitHub Apps zvyšují efektivitu vývoje a umožňují vývojářům soustředit se na klíčové aspekty softwarového projektu.

GitHub Apps představují pokročilý nástroj pro automatizaci a integraci procesů v softwarovém vývoji na platformě GitHub. Díky jejich schopnosti poskytovat přizpůsobená oprávnění a působit ve prospěch konkrétních úloh, představují klíčovou komponentu pro vývojářské týmy, které hledají způsoby, jak optimalizovat své vývojové procesy.

Přidání GitHub Apps do repositáře tutoriálu. V našem GitHub repositáři je využívána GitHub App společnosti Render pro nasazení aplikace na servery Renderu. Přidání probíhalo tak, že jsem byl v prvním kroku vyzván aplikací Render pro připojení jedné z verzovacích aplikací. Připojil jsem tedy verzovací systém, v našem případě tedy GitHub. Následně jsem vybral konkrétní repositář, kde jsou uložené zdrojové kódy tutoriálu. Toto přidání bylo potřeba také schválit. Schválení probíhá tak, že je uživatel přesměrován na stránku GitHubu, která obsahuje tlačítko pro schválení (předpokládáme, že je uživatel již přihlášen), po němž je uživatel přesmě-

rován zpět na stránku Renderu.

Proces nasazení změn. Nasazení sestává z několika kroků. Prvním krokem při nasazování aplikace je automatické spuštění sestavovacího (build) procesu po každém `push` příkazu do repositáře. Parametry toho, jaké příkazy se provedou pro sestavení se nastavuje na platformě Render (viz. obrázek 6.1). Následně po procesu sestavení se statické soubory nahrají na server poskytnutý Renderem a jsou dostupné na doméně také od Renderu. Na platformě Render je také uživatel schopen vidět konzoli s výpisem programů spuštěnými během `build` procesu, což je především vhodné pro případné odstraňování chyb.

Komunitní údržba a rozšiřovaní. Aby tutoriál zůstal relevantní, je potřeba ho neustále aktualizovat. K tomu slouží tzv. `pull requesty`. Pokud přispěvatel se chystá udělat nějaké vylepšení či změnu, jednoduše vytvoří `pull request`, který v sobě obsahuje změny, jež se přidají a následně i projeví po schválení. Po vytvoření `pull requestu` může probíhat diskuse nad navrhovanými změnami. Všichni si také mohou navrhované změny kódu prohlédnout. GitHub zároveň nabízí přehled všech `pull requestů` ve stavech `open` a `closed` (viz obrázek 6.2).

6.4.0.1 Práce s textem a formátováním tutoriálu

Jak už bylo zmíněno, tutoriál pro tvorbu obsahu využívá značkovací jazyk Markdown. Ten je navíc rozšiřitelný o rozšíření nabízející další funkce. V této kapitole se budeme věnovat základnímu použití Markdownu a rozšířením, jež jsem v tutoriálu využil.

Použití Markdownu. V následující tabulce 6.3 je uvedena základní syntaxe Markdownu používaná v tutoriálu [49].

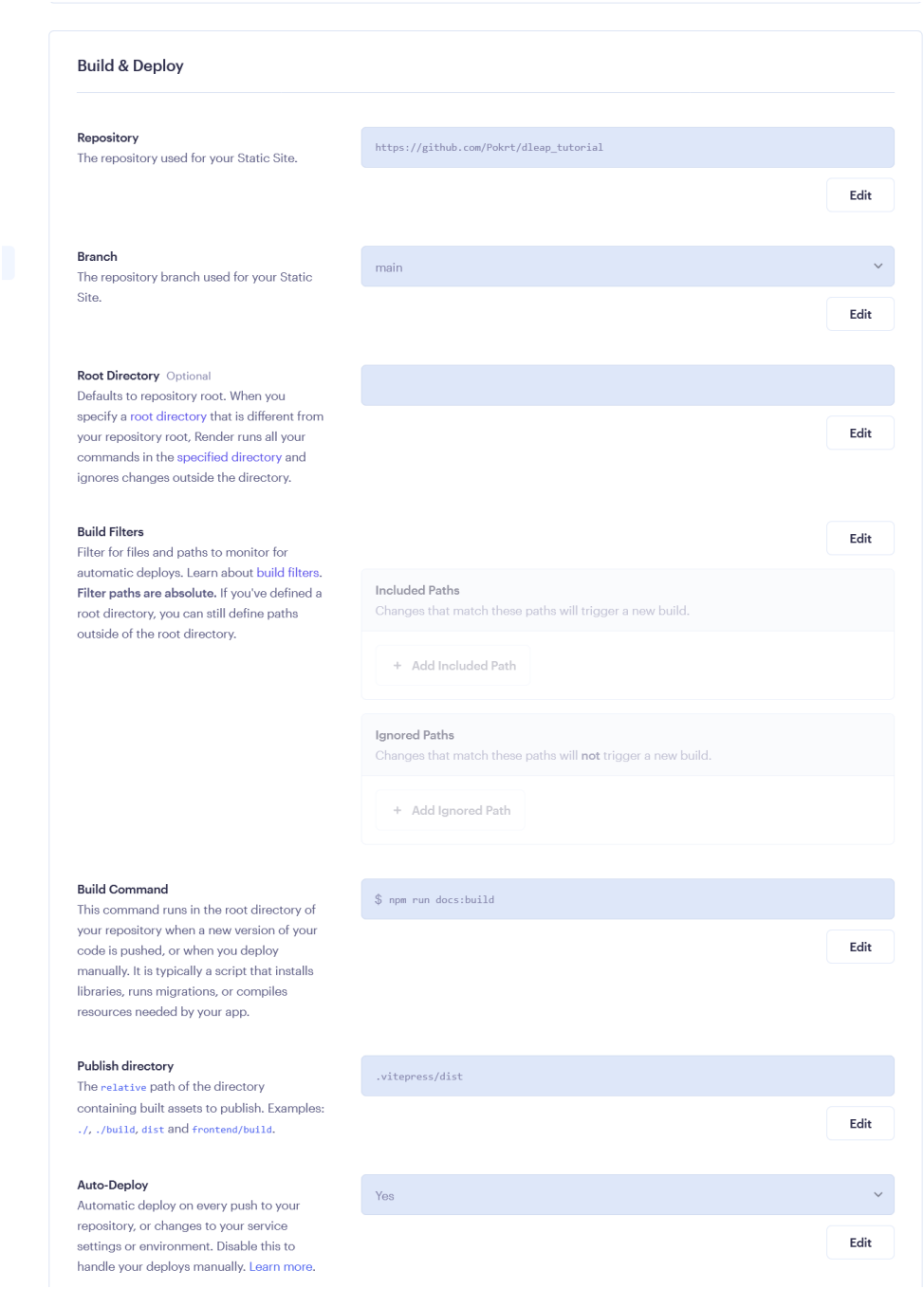
Grafický výstup těchto elementů by pak po zobrazení VitePressem vypadal tak, jak je zobrazeno níže na obrázku 6.3.

Markdown rozšíření. VitePress v základu nabízí značné množství rozšíření. Úplný seznam všech rozšíření je dostupný v oficiální dokumentaci [50]. Níže jsou popsaná rozšíření, jež jsou v tutoriálu použity.

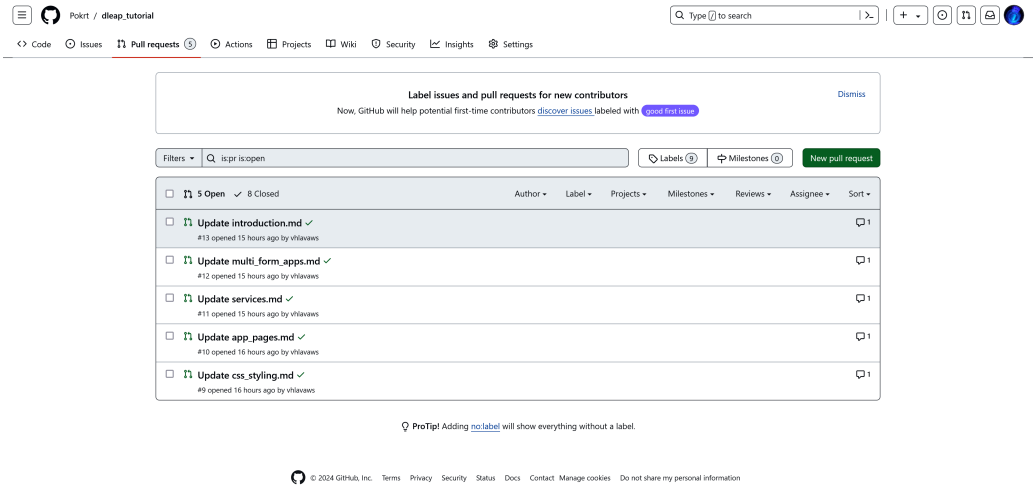
Kontejnery. Kontejnery jsou v tutoriálu používány pro tvorbu zadání cvičení, vytváření tipů a rozbalovacího okna s postupem krok za krokem. Níže následuje názorná ukázka:

Zdrojový kód 6.1: Ukázkový rozšířený Markdown kód pro tvorbu kontejnerů z kapitoly Hello World v tutoriálu

```
1 ::: tip Exercise 1
```



Obrázek 6.1: Ukázka nastavení platformy Render pro sestavení a nasazení tutoriálu



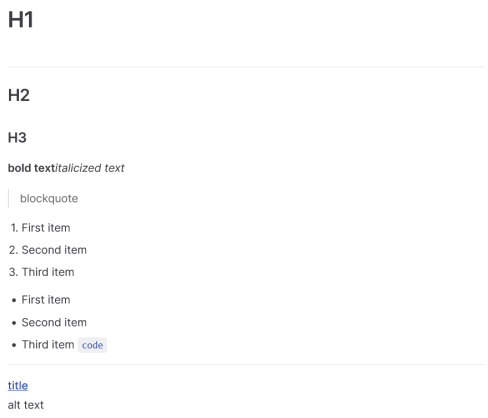
Obrázek 6.2: Ukázka toho, jak jiní uživatelé mohou přispívat do tutoriálu pomocí pull requestů

Element	**Markdown Syntaxe**
Nadpis	# H1
	## H2
	### H3
Tučně	**bold text**
Kurzíva	*italicized text*
Citace	> blockquote
Seřazený list	1. First item
	2. Second item
	3. Third item
Neřazený list	- First item
	- Second item
	- Third item
Kód	‘code‘
Vodorovná čára	—
Odkaz	[title](https://www.example.com)
Obrázek	![alt text](image.jpg)

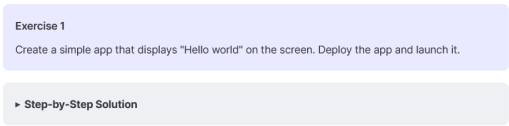
Tabulka 6.3: Základní Markdown syntaxe

```
Create a simple app that displays "Hello␣world" on the screen
    . Deploy the app and launch it.
:::

::: details Step-by-Step Solution
1\. Navigate to URL of your HCL Domino Leap
...
```



Obrázek 6.3: Grafický výstup základní syntaxe v Markdown



Obrázek 6.4: Grafický výstup kontejnerů

```
:::
```

Na obrázku 6.4 pak můžeme vidět jak kód 6.1 bude vypadat.

Zvýraznění syntaxe kódu. Dalžím rozšířením používaným pro tvorbu tutoriálu je zvýraznění syntaxe kódu. VitePress v základu zahrnuje pro tento účel knihovnu Shiki [51] [50]. Následuje ukázka použití této knihovny pro zvýraznění syntaxe JavaScriptového kódu:

Zdrojový kód 6.2: Ukázka zdrojového kódu v rozšířeném Markdownu pro zobrazení zdrojového kódu JavaScriptu se zvýrazněním syntaxe z kapitoly Service Calls v tutoriálu

```
‘‘‘javascript
// Retrieve the form object
var form = app.getForm(’F_Form1’); // Replace ’F_Form1’ with
    your actual Form ID

// Create a handle for ’serviceA’
var srvA = form.getServiceConfiguration(’SC_serviceA’);

// Connect a function to the ’onCallFinished’ event of ’
    serviceA’
srvA.connectEvent("onCallFinished", function (success) {
    // Verify that ’serviceA’ call was successful
    if (success) {
```

```
// Retrieve the form object
var form = app.getForm('F_Form1'); // Replace 'F_Form1' with your actual Form ID

// Create a handle for 'serviceA'
var srvA = form.getServiceConfiguration('SC_serviceA');

// Connect a function to the 'onCallFinished' event of 'serviceA'
srvA.connectEvent("onCallFinished", function (success) {
    // Verify that 'serviceA' call was successful
    if (success) {
        // Trigger 'serviceB'
        form.getServiceConfiguration('SC_serviceB').callService();
    }
});
```

Obrázek 6.5: Zobrazení uživateli tutoriálu

```
        // Trigger 'serviceB'
        form.getServiceConfiguration('SC_serviceB').
    callService();
    }
});
```

Začlenění Markdown souboru. Posledním využívaným rozšířením v tomto tutoriálu je rozšíření pro začlenění souboru. To jsem použil k tomu, abych nemusel pokaždé do sekce s referencí kopírovat metody, události, nebo objekty. Do souboru stačí přidat komentář (viz.) a rozšíření následně automaticky zobrazuje obsah souboru odkazovaném v komentáři.

Zdrojový kód 6.3: Zahrnutí obsahu jiného Markdown souboru

```
<!--@include: ./common/events.md-->
```

Frontmatter. Frontmatter je rozšíření sloužící pro konfiguraci různých rozložení stránky. Na stránky tutoriálu VitePress automaticky aplikuje rozložení `docs`. Toto výchozí rozložení se přepisuje na stránce `intex.md` v kořenovém adresáři obsahujícího zdrojové soubory. Tato stránka slouží jako domovská stránka obsahující úvodní informace o tutoriálu, a proto bylo pro ni použito rozložení `home`, jež je v základu poskytováno VitePressem a s dodatečnou konfigurací .

Zdrojový kód 6.4: Konfigurace Frontmatteru pro domovskou stránku

```
---
layout: home

hero:
  name: "HCL Domino Leap Tutorial"
  text: "From zero to hero in few days."
  tagline: "This tutorial will help you learn how to develop workflow applications in HCL Domino Leap using no-code and low-code (JavaScript) skills. "
  actions:
    - theme: brand
      text: Tutorial
```

```
11        link: /basics/introduction
12      - theme: alt
13        text: Reference
14        link: /object_reference/overview
15
16  ---
```

6.4.0.2 Konfigurace VitePressu

VitePress poskytuje možnost úprav vzhledu a funkcionalit prostřednictvím konfiguračního souboru, který se nachází ve složce `.vitepress/config.mjs`. Pro potřeby tutoriálu byl použit tento konfigurační soubor.

Zdrojový kód 6.5: Část konfiguračního souboru tutoriálu

```
1  import {defineConfig} from 'vitepress'
2
3  // https://vitepress.dev/reference/site-config
4  export default defineConfig({
5      sitemap: {
6          hostname: 'https://dleap-tutorial.onrender.com/'
7      },
8      title: "HCL␣Domino␣Leap␣Tutorial",
9      description: "Tutorial␣for␣HCL␣Domino␣Leap",
10     lastUpdated: true,
11
12     markdown: {
13         theme: {light: 'light-plus', dark: 'night-owl'}
14     },
15     themeConfig: {
16         // https://vitepress.dev/reference/default-theme-
       config
17         editLink: {
18             pattern: 'https://github.com/Pokrt/dleap_tutorial
       /blob/main/:path'
19         },
20         search: {
21             provider: 'local'
22         },
23         nav: [
24             {text: 'Tutorial', link: '/basics/introduction'},
25             {text: 'Reference', link: '/object_reference/
       overview'},
26         ],
27
28         socialLinks: [
29             {icon: 'github', link: 'https://github.com/Pokrt/
       dleap_tutorial'}
```

```
30          ],
31          outline: [2, 4],
32          sidebar: [
33              {
34                  text: 'I. Domino Leap Basics',
35                  collapsed: false,
36                  items: [
37                      {text: 'Introduction', link: '/basics/
    introduction'},
38                      {text: 'Your First App', link: '/basics/
    hello_world'},
39                      {text: 'Where is your Data', link: '/
    basics/data-storage'},
40                  ]
41              },
42              ...
43          ]
44      },
45  })
```

Popis konfigurace.

sitemap.hostname. Tato vlastnost slouží ke specifikaci domény, na které výsledný tutoriál běží.

title. Udává název stránky.

description. Slouží pro přidání metadat do stránky, konkrétně vložení HTML `<meta>` tagu.

lastUpdated. Umožňuje zobrazení data a času poslední změny dole na stránce.

markdown.theme. Upravuje výsledný vizuální vzhled tutoriálu. Konkrétně v tomto případě nastavuje Shiki [52]

themeConfig.editlink.pattern. Nastavuje cestu do kořenu Git repositáře. VitePress automaticky nastaví úplnou cestu až k souboru v repositáři, na kterou se čtenář tutoriálu může dole na stránce prokliknout pro provádění změn jako třeba opravu překlepů či jiné vylepšování tutoriálu.

themeConfig.search.provider. Přidává do tutoriálu možnost v něm vyhledávat. Je možnost přidat i placený vyhledávač Algolia, pro potřeby tutoriálu je však vyhledávač poskytnutý v základu dostačující.

themeConfig.nav Slouží ke konfiguraci navigace zobrazené v horní části stránky.

themeConfig.socialLinks. Slouží k přidání odkazu na ostatní platformy jako je GitHub, FaceBook nebo X pomocí objektů v poli. V tomto případě je zde přidán odkaz na GitHub tutoriálu.

themeConfig.outline. Tato konfigurace určuje jakou hloubku zanoření nadpisů bude pravý boční panel tutoriálu zobrazovat.

themeConfig.sidebar. Toto nastavení slouží k nastavení navigace levého bočního panelu. VitePress zde umožňuje zanoření do hloubky, dělení do skupin a možnost nastavení, zda bude skupina nebo podskupina tzv. „sbalovací“.

Adresářová struktura. VitePress automatiky pro cestu k Markdown souboru automaticky generuje HTML stránku např. `<složka tutoriálu>/basics/hello_world.md` bude dostupné na stránce `<doména>/basics/hello_world.html`. Za zmínku stojí, že adresářová struktura nemá žádný vliv na to, jaké bude složení bočního navigačního panelu, ten se konfiguruje ve výše zmíněném konfiguračním souboru.

6.5 Struktura a obsah tutoriálu

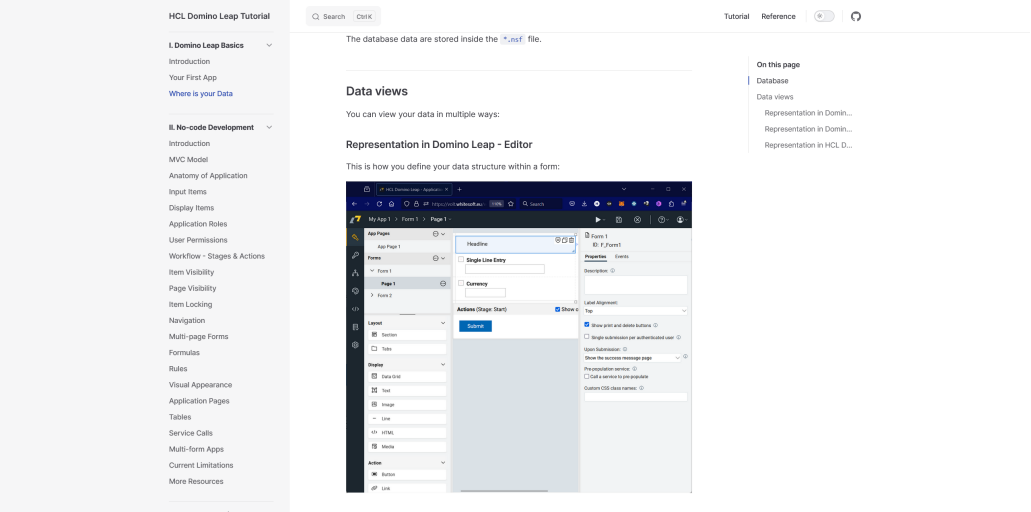
V této kapitole se budeme zabývat strukturou a obsahem tutoriálu. Cílem této kapitoly je poskytnout představu o tom, jak je tutoriál strukturován. Materiál je rozdělen do jednotlivých kapitol. Některé kapitoly pak obsahují další podkapitoly pro zlepšení orientace v tutoriálu uživatelem. Tutoriál je dostupný na webové adrese `https://dleap-tutorial.onrender.com/basics/introduction.html`. Tutoriál se skládá z pěti hlavních sekcí:

- I. Domino Leap Basics,
- II. No-code Development,
- III. Low-code Basics,
- IV. Object Reference,
- V. Advanced Techniques.

6.6 Sekce I. Domino Leap Basics

Sekce Domino Leap Basics sestává ze tří kapitol - Introduction, Your First App a Where is your Data (viz. obrázek 6.6). Tyto tři kapitoly seznamují uživatele s tím úplně nejzákladnějším.

Kapitola `Introduction` seznamuje uživatele se strukturou zbytku tutoriálu a jeho obsahem. Dále se zabývá typografickými konvencemi jako je značení jednolivých metod, objektů a událostí. Přibližuje uživateli označení používaná pro volání metod nad jednotlivými typy objektů - objekty typu `item` a `apItem`. Dále pak uživateli popisuje možnost vyzkoušet si nejdříve úkol samostatně a v případě tápání, nebo nejasností použít rozbalovací okénko nazvané *Step-by-Step solution*. Toto okénko uživateli po rozbalení, jak již název napovídá, ukáže detailní řešení popsané



Obrázek 6.6: Kapitola Where is your Data ze sekce Domino Leap Basics

v jednotlivých krocích s využitím velkého množství obrázků zachycující jednotlivé kroky. Díky tomu se uživatel snadno orientuje v platformě HCL Domino Leap.

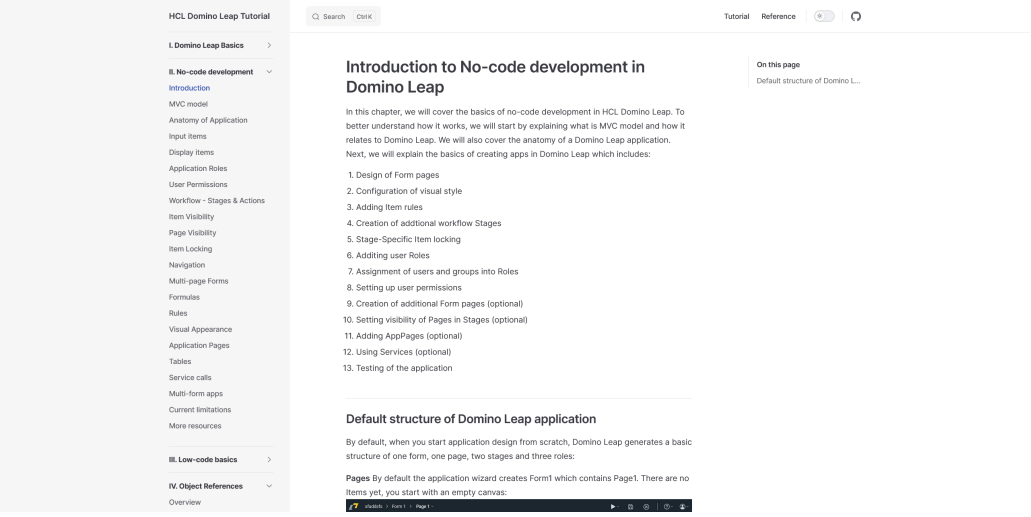
Kapitola `Your First App` seznamuje uživatele s prvními kroky při vytváření aplikace a základními komponentami aplikace jako je Editor, Formulář, Stránka a Prvek stránky. Ukazuje uživateli základní formulář obsahující název formuláře, pole pro zadání dat do formuláře a tlačítko odesílající data. Věnuje se také podobě formuláře a jeho odeslání a náhledu na již odeslaná data. Na konci kapitoly má pak uživatel možnost vyzkoušet si pro lepší osvojení znalostí cvičení s úkolem vytvořit jednoduchou aplikaci zobrazující nápis „Hello world“, tu pak nasadit a spustit.

Kapitola `Where is your Data` popisuje ukládání dat do Notes Storage Facility (NSF) souborů a vysvětluje uživateli jakou strukturu a funkci tyto soubory mají. Seznamuje uživatele také s přístupovými praávy k jednotlivých souborům prostřednictvím Access Controll Listů (ACL). Zabývá se také náhledem na data. Ukazuje uživateli, jak jsou data zobrazena v pohledech z aplikací jako HCL Domino, HCL Domino Leap v náhledu na odeslaná data a pohled prostřednictvím editoru HCL Domino Leap.

6.7 Sekce II. No-code Development

Druhá sekce se přibližuje vývoj v aplikaci HCL Domino Leap uživateli bez potřeby znalostí JavaScriptu. V této sekci nalezneme devatenáct hlavních kapitol. Kapitola `Introduction` (viz. obrázek 6.7 seznamuje čtenáře tutoriálu s obsahem sekce `No-code Development`.

Kapitola `MVC Model` vysvětluje koncept Model, View, Controller. Nejdříve vy-



Obrázek 6.7: Ukázka ze sekce No-code Development

světluje uživateli tyto tři prvky modelu z obecné perspektivy. Pro lepší představu je zde uveden diagram znázorňující již zmiňovaný Model, View, Controller. V další části této kapitoly jsou jednotlivé prvky grafického rozhraní HCL Domino Leap přiřazeny ke komponentám MVC modelu. Uživatel získá konkrétní představu a lépe pochopí princip fungování HCL Domino Leap. Toto vysvětlení je ještě obohaceno o snímky obrazovky uživatelského rozhraní se zvýrazněnými určitými prvky grafického rozhraní k příslušné komponentě MVC modelu. Kupříkladu jsou u komponenty `View` vyznačeny stránky, prvky pohledu (display items), šířka (určující, jak široce se bude vstupní pole zobrazovat) a CSS třídy (umožňují upravit právě vzhled).

Kapitola `Anatomy of Application` vysvětluje reprezentaci entity a jí příslušných metod a atributů. Kapitola dále tento koncept entit přibližuje uživateli pomocí příkladů entit z reálného světa jako jsou zákazníci, objednávky, produkty či zaměstnanci. Kapitola dále pokračuje popsáním reprezentace entit v HCL Domino Leap. Objasňuje, jak jsou prvky grafického rozhraní jako formuláře a vstupní pole mapované na datový objekt jako např. políčko pro zadání textu (prvek grafického rozhraní) je mapované na atribut datového objektu. V závěru pak kapitola přibližuje uživateli analogie mezi HCL Domino a HCL Domino Leap.

Kapitola `Input Items` tento koncept vstupních prvků (input items) zmíněný v předchozí kapitole dále popisuje, jak jsou tyto prvky používány pro ukládání a vizuální reprezentaci. Kapitola dále zmiňuje, jaké vlastnosti upravující vzhled a chování prvky mají. Konkrétně zmiňuje čtyři nejpoužívanější vlastnosti. Jsou jimi šířka, výplňový text (placeholder), formát a vlastní CSS třídy. Pro zlepšení orientace uživatelem je odstavec opatřen obrázkem. Následuje odstavec ukazují uživateli, jak se nastavují prvky jako požadované (required). Odstavec opět obsahuje ilustrativní

obrázek. V poslední části kapitoly jsou pak uživateli vysvětleny typy jednotlivých vstupních prvků. Jsou jimi prvky jako jednořádkový vstup, měna, číselný vstup. U každého ze vstupních prvků je uveden i ilustrativní obrázek.

Kapitola `Display Items` popisuje prvky sloužící ke zobrazení pouze různých druhů dat, jako jsou obrázky, text nebo datová síť. Zabývá se také tím, jak zobrazovat hodnoty jiného vstupního prvku na více místech zároveň pomocí tzv. `data mirroring`. Většina ze zobrazovacích prvků má uveden jeden či více ilustrativních obrázků.

Kapitola `Application Roles` vysvětluje, k čemu role jsou, dále pak jejich typy (otevřené a zavřené) a v závěru pak kapitola popisuje postup pro přidání nové role.

Kapitola `User Permissions` popisuje jednotlivé typy oprávnění a proces přiřazení oprávnění k roli.

Kapitola `Workflow - Stages & Actions` seznamuje uživatele s konceptem fází a akcí a prací s nimi. Dále pak kapitola nabízí možnost vyzkoušet přidání si nové fáze nebo větvení a a v případě nesnází jim poskytuje řešení „Krok za krokem“.

Kapitola `Item Visibility` popisuje nastavení, zda je nějaký prvek skrytý či viditelný, a to podle fáze, ve které formulář je, nebo podle podle určité podmínky ve formuláři např. vstup od uživatele v jiném políčku je menší než 300 dolarů.

Kapitola `Page Visibility` se zabývá viditelností různých stránek v různých fázích.

Kapitola `Item Locking` se věnuje uzamykání, tedy procesu, kdy se stane daná stránka nebo prvek pouze ke čtení.

Kapitola `Navigation` se zaměřuje na možnosti vytvoření navigace mezi stránkami. Možnosti jsou dvě - navigace pomocí navigačního prvku nebo pomocí tlačítka.

Kapitola `Multi Page Forms` popisuje, jak vytvářet více stránkové formuláře.

V kapitole `Formulas` jsou popsány vzorce umožňující přidat výpočet sčítání, násobení, průměr tabulky, suma tabulky a další užitečné funkce. Kapitola je zakončena cvičením zaměřeného na přidávání vzorců do formuláře.

Kapitola `Rules` seznamuje čtenáře se funkcionalitou pravidel jako je skrývání určitých tlačítek. Dále se pak kapitola zabývá zjišťováním přítomnosti již nastavených pravidel. Na konci kapitoly si může nabyté znalosti upevnit prostřednictvím cvičení.

Kapitola `Vissual Appearence` ukazuje, jak si tvůrce aplikace může přizpůsobovat vzhled prostřednictvím editoru.

Kapitola `Application Pages` se zabývá tvorbou tzv. aplikačních stránek, tedy stránek, které jsou pouze pro čtení a uživatel nemá možnost, jak manipulovat data aplikace. V kapitole je též popsána tvorba takovýchto stránek.

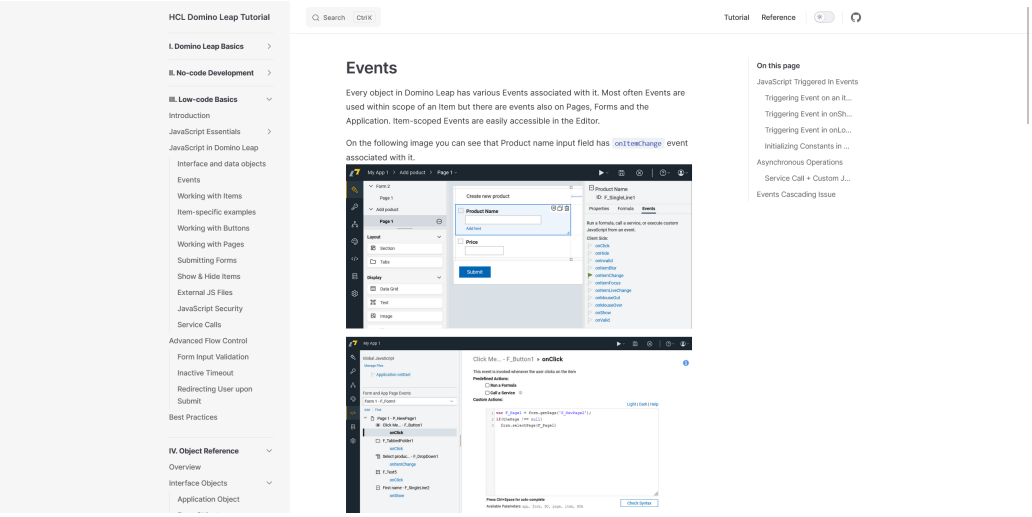
Kapitola `Tables` pak popisuje prvek, jež umožňuje zobrazení dat aplikace v tabulce a nabízí opět možnost si uživatelem vyzkoušet nejdříve sám a případně sáhnout po řešení krok za krokem.

Kapitola `Service Calls` popisuje funkcionalitu „servisů“. Dále pak zmiňuje jejich typy, jejich automatické generování a práci s nimi. V závěru je pak přiložený video tutoriál.

Kapitola `Multi-fom Apps` popisuje, jak může probíhat výměna dat mezi vícero formuláři. Tato kapitola je zakončena cvičením zaměřeného na získání dat z jiného formuláře.

6.8 Sekce III. Low-code Basics

Tato sekce se sestává ze třech hlavních celků a to `Javascrit Essentials`, `JavaScript in Domino Leap` a `Advanced Flow Control`. Tyto tři kapitoly seznamují čtenáře s tím, jak udělat některé pokročilejší věci v HCL Domino Leap za pomoci využití JavaSriptu a objektů, událostí (viz.) a metod poskytnutých platformou.



Obrázek 6.8: Ukázka kapitoly z tutoriálu o událostech

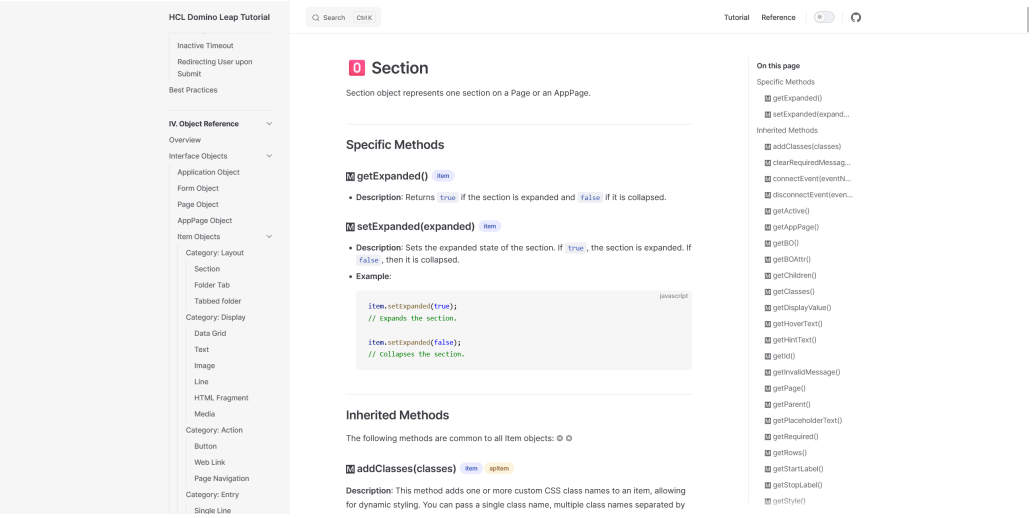
Javascrit Essentials. Tato sekce vysvětluje úplné základy JavaScriptu potřebné pro práci s HCL Domino Leap. Sekce je určená pro začátečníky, kteří ještě do kontaktu s JavaScriptem nepřišli.

JavaScript in Domino Leap. V této sekci se tutoriál věnuje objektům, událostem a metodám. U prvků se pak tato sekce také zaměřuje na jejich specifika, jmenovitě nastavovaní stisknutelnosti tlačítek, programově spuštěnou navigaci mezi stránkami formuláře, získávání možností u průzkumů, nebo volání JavaScriptových funkcí z externích souborů.

Advanced Flow Control. Tato sekce se věnuje řízení průchodu aplikací uživatelem a seznamuje ho s koncepty jako je validace uživatelského vstupu pomocí regulárních výrazů, odeslání formuláře pro uplynutí určitého časového kvanta nebo přesměrování uživatele.

6.9 Object Reference

Tato sekce poskytuje vylepšenou verzi oficiální dokumentace. Je vylepšená především o zvýraznění syntaxe a ikonky, ilustrující, zda-li se jedná o objekt, metodu či událost (viz. obrázek 6.9).



Obrázek 6.9: Ukázka reference vytvořené pro objekt sekce

6.10 Advanced Techniques

Tato sekce je zde připravená pro rozšíření tutoriálu o další části zaměřené na méně obecná témata, jako je např. tvorba QR kódů nebo tisk formulářů.

Testování 7

Tutoriál byl testován dvěma zkušenými vývojáři nejen v HCL Domino Leap. Po týdenním testování tutoriál hodnotili takto:

> “ HCL Domino Leap Tutorial, navržený, sestavený a publikovaný Janem Kadlecem, vnímám jako užitečnou věc. Pracuji ve společnosti HCLSoftware jako technický specialista pro tuto oblast software a věřím, že firmy používající Leap, budou z tohoto Tutorialu benefitovat.
>
> Připravený obsah je dobře strukturován a člověk může začít na té části, která je pro něj vhodná: začátečníci na kapitole First App, pokročilejší v sekcích low-code/no-code, a ti zkušení si vyhledají řešení na svůj specifický problém. Velmi oceňuji, že jednotlivé kapitoly mají Step-by-step sekci, kde jsou jednotlivé kroky přesně popsány a doprovozeny screenshoty. Další věc, kterou vývojáři v Leapu ocení, jsou připravené snippety ukázkového kódu, kterou mohou vzít a použít ve své aplikaci.
>
> Přehledně je také zpracována sekce Reference, včetně příkladů volání jednotlivých metod či práce s objekty. Protože se produktové API může v čase měnit, doporučuji sekci Reference doplnit číslem verze produktu a datem poslední aplikace, aby bylo vidět k čemu přesně se vztahuje.
>
> Tutorial je publikován formou otevřené dokumentace, se zdrojovými kódy v MarkDownu na GitHub. Díky tomu je možno jej snad upravovat a okamžitě publikovat změny, a to i více autory paralelně.
>
> HCL Domino Leap Tutorial budu během své práce doporučovat společnostem, které produkt používají, nebo o jeho pořízení uvažují.
>
> **Petr Kunc**
>
> HCLSoftware
>
> *Technical Advisor, EMEA* ”

> “ HCL Domino Leap je platformou, která v umožňuje tvorbu firemních aplikací i běžným uživatelům s minimálním znalostí problematiky programování. Předpoklad povrchní znalosti terminologie, návrhových vzorů, datových modelů apod. velmi ztěžuje dokumentaci podobných systému.
>
> Tato práce pro chytře využívá webové Markdown prezentace pro vytvoření přehledného uživatelského návodu. Struktura kapitol logicky provází uživatele problematikou tvorby aplikací a upozorňuje na problematická místa. Pomocí step by step obrazových návodů dokáže i naprostý laik vytvořit jednoduchou Enterprise aplikaci.
>
> Některé na pochopení náročnější kapitoly např. aplikační role a uživatelská práva by právě proto, že jsou určeny laikům, mohly obsahovat obsáhlejší a více návodný popis.
>
> V návazných kapitolách upozorňuje na slabá místa vývoje v tomto prostředí, která nelze snadno řešit bez více či méně složitého programování a nabízí návod na řešení problematických scénářů. U pozdějších kapitolách určených spíše vývojářům již možná nadbytečně supluje výuku programování v JavaScriptu.
>
> **Vít Hlava**
>
> Whitesoft
>
> *IT konzultant* ”

Obě recenze společně poskytují pozitivní hodnocení HCL Domino Leap tutoriálu, i když z různých perspektiv. Petr Kunc, technický specialista, vnímá tutoriál jako velmi užitečný, dobře strukturovaný a snadno použitelný pro uživatele na různých úrovních zkušeností. Oceňuje step-by-step průvodce, připravené úryvky kódu a navrhuje drobné vylepšení v sekci Reference. Zdůrazňuje také flexibilitu tutoriálu publikovaného na GitHubu, což umožňuje snadné aktualizace a spolupráci více autorů.

Vít Hlava, IT konzultant, posuzuje tutoriál jako přístupný i pro uživatele s minimálními znalostmi programování, díky jeho logické struktuře a názorným návodům. Zároveň poukazuje na určité složitější kapitoly, které by mohly být lépe přizpůsobeny laikům, a vyzdvihuje návody na řešení komplikovanějších scénářů. Kriticky hodnotí nadměrné zaměření na výuku JavaScriptu v pozdějších kapitolách, které jsou spíše určeny pokročilým uživatelům.

Celkově obě recenze naznačují, že HCL Domino Leap tutoriál je kvalitní a přínosný zdroj pro uživatele různých úrovní dovedností, který efektivně podporuje tvorbu aplikací v rámci této platformy.

Závěr a zhodnocení 8

V této bakalářské práci jsem se zaměřil na vytvoření výukového kurzu pro vývoj aplikací na platformě HCL Domino Leap. Cílem bylo poskytnout uživatelům, kteří nejsou hluboce technicky zdatní, schopnosti potřebné pro efektivní vytváření a správu těchto aplikací.

Během práce byl realizován on-line výukový kurz dostupný na adrese `dleap-tutorial.onrender.com`, který demonstruje základní principy tvorby aplikací na zmíněné platformě. Kurz klade důraz na praktické využití teoretických znalostí, což bylo demonstrováno řadou praktických příkladů a cvičení.

Kurz byl testován na skupině uživatelů a zpětná vazba byla pozitivní. Uživatelé ocenili jeho strukturovanost, přístupnost a užitečnost. Též ocenili možnost do tutoriálu přispívat prostřednictvím GitHubu.

Závěrem lze říci, že cíle práce byly úspěšně splněny. Vytvořený kurz představuje cenný zdroj pro začínající i pokročilé uživatele, kteří chtějí rozšířit své dovednosti v rychle se rozvíjejícím oboru low-code aplikací. Budoucí práce by mohla kurz dále rozšířit o pokročilejší techniky a integraci s dalšími technologickými nástroji.

Bibliografie

1. MARK RANDALL HAVENS. *ITNEXT*. A Decade by Decade Retrospective: The Evolution of Programming Languages | by Mark Randall Havens | ITNEXT. 2023. Dostupné také z: https://itnext.io/a-decade-by-decade-retrospective-the-evolution-of-programming-languages-92d2c72afe5f#:~:text=Programming%20in%20the%201970s%3A%20A,programming%20environments%2C%20such%20as.
2. BROOKSHEAR, J. Glenn.; SMITH, David T. (David Timothy); BRYLOW, Dennis. Computer science : an overview. 2012, s. 609. ISBN 978-0-13-256903-3.
3. MARTIN, James. *Application development without programmers*. Prentice-Hall, 1982.
4. MARTIN, James. *Rapid Application Development*. Macmillan Publishing Co. Inc, 1991.
5. RICHARDSON, C; RYMER, J. *New Development Platforms Emerge for Customer-Facing Applications*. Forrester Research, 2014.
6. OLEKSANDR MATVITSKYY et al. *Gartner Magic Quadrant for Enterprise Low-Code Application Platforms*. 2023. Tech. zpr. Dostupné také z: https://www.gartner.com/en/documents/4843031.
7. *AI Low-Code Application Development Platform | Mendix*. [B.r.]. Dostupné také z: https://www.mendix.com/platform/ai/.
8. *Home - Object Management Group*. [B.r.]. Dostupné také z: https://www.objectmanagementgroup.org/.
9. BROWN, Alan W. Model driven architecture: Principles and practice. *Software and Systems Modeling 2004 3:4*. 2004, roč. 3, č. 4, s. 314–327. ISSN 1619-1374. Dostupné z DOI: 10.1007/S10270-004-0061-2.
10. SINGH, Yashwant; SOOD, Dr. Manu. The Impact of the Computational Independent Model for Enterprise Information System Development. *International Journal of Computer Applications*. 2010, roč. 11, č. 8, s. 21–26. Dostupné z DOI: 10.5120/1602-2153.

11. *What is Low-Code Development? | Mendix*. [B.r.]. Dostupné také z: https://www.mendix.com/low-code-guide/#what-is-lowcode.
12. *Gartner Research : TechWeb : Boston University*. [B.r.]. Dostupné také z: https://www.bu.edu/tech/about/professional-resources/gartner-research/.
13. *Gartner | IT Stock Price, Company Overview & News*. [B.r.]. Dostupné také z: https://www.forbes.com/companies/gartner/?list=worlds-best-management-consulting-firms&sh=60ab3e877faf.
14. *What is Disaster Recovery? - Definition & Benefits | VMware*. [B.r.]. Dostupné také z: https://www.vmware.com/topics/glossary/content/disaster-recovery.html.
15. ROKIS, Karlis; KIRIKOVA, Marite. Exploring Low-Code Development: A Comprehensive Literature Review. *Complex Systems Informatics and Modeling Quarterly*. 2023, roč. 0, č. 36, s. 68–86. ISSN 2255-9922. Dostupné z DOI: 10.7250/CSIMQ.2023-36.04.
16. DI RUSCIO, Davide et al. Low-code development and model-driven engineering: Two sides of the same coin? *Software and Systems Modeling*. 2022, roč. 21, č. 2, s. 437–446. ISSN 16191374. Dostupné z DOI: 10.1007/S10270-021-00970-2/FIGURES/8.
17. HIRZEL, Martin. Low-Code Programming Models. *Communications of the ACM*. 2023, roč. 66, č. 10, s. 76–85. ISSN 0001-0782. Dostupné z DOI: 10.1145/3587691.
18. YAN, Zhaohang. The Impacts of Low/No-Code Development on Digital Transformation and Software Development. 2021. Dostupné také z: https://arxiv.org/abs/2112.14073v1.
19. LUO, Yajing; LIANG, Peng; WANG, Chong; SHAHIN, Mojtaba; ZHAN, Jing. Characteristics and challenges of low-code development: The practitioners perspective. *International Symposium on Empirical Software Engineering and Measurement*. 2021. ISBN 9781450386654. ISSN 19493789. Dostupné z DOI: 10.1145/3475716.3475782.
20. RAO, Nikitha; TSAY, Jason; KATE, Kiran; HELLENDOORN, Vincent J.; HIRZEL, Martin. AI for Low-Code for AI. 2023. Dostupné také z: https://arxiv.org/abs/2305.20015v1.
21. OGUNDARE, Oluwatosin; ARAYA, Gustavo Quiros; QAMSANE, Yassine. No Code AI: Automatic generation of Function Block Diagrams from documentation and associated heuristic for context-aware ML algorithm training. 2023. Dostupné také z: https://arxiv.org/abs/2304.04117v1.

22. CAI, Yuzhe et al. Low-code LLM: Visual Programming over LLMs. 2023. Dostupné také z: https://arxiv.org/abs/2304.08103v2.

23. HENRIQUES, Henrique; LOURENÇO, Hugo; AMARAL, Vasco; GOULÃO, Miguel. Improving the developer experience with a low-code process modelling language. *Proceedings - 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2018*. 2018, s. 200–210. ISBN 9781450349499. Dostupné z DOI: 10.1145/3239372.3239387.

24. *2023*. Setting Up OutSystems - OutSystems 11 Documentation. [B.r.]. Dostupné také z: https://success.outsystems.com/documentation/11/setup_and_maintain_your_outsystems_infrastructure/setting_up_outsystems/.

25. FREDERIC LARDINOIS. Siemens acquires low-code platform Mendix for $700M | TechCrunch. *TechCrunch*. 2018. Dostupné také z: https://techcrunch.com/2018/08/01/siemens-acquires-low-code-platform-mendix-for-700m/.

26. *Mendix*. [B.r.]. Dostupné také z: https://docs.datadoghq.com/integrations/mendix/.

27. *Mendix Marketplace - Studio Pro 10.5.0 - Download*. 2023. Dostupné také z: https://marketplace.mendix.com/link/studiopro/.

28. *What is Retool? A Developer's Introduction*. 2023. Dostupné také z: https://www.superblocks.com/compare/what-is-retool.

29. *App IDE | Retool Docs*. [B.r.]. Dostupné také z: https://docs.retool.com/apps/web/concepts/ide.

30. FRANK, Ulrich ; MAIER, Pierre ; BOCK, Alexander. *Low code platforms: Promises, concepts and prospects. A comparative study of ten systems*. Essen: Essen: Universität Duisburg-Essen, Institut für Informatik und Wirtschaftsinformatik (ICB), 2021. Tech. zpr. Universität Duisburg-Essen, Institut für Informatik und Wirtschaftsinformatik (ICB). Dostupné z DOI: 10.17185/DUEPUBLICO/75244.

31. MARTIN LECHLEIDER. *HCL Domino Leap 1.1 - New Name and Loaded With New Features*. 2022. Dostupné také z: https://www.hcl-software.com/blog/domino/hcl-domino-leap-1-1-new-name-and-loaded-with-new-features/.

32. SIMON SHARWOOD. IBM offloads Notes and Domino to India's HCL Technologies • The Register. *2017*. [B.r.]. Dostupné také z: https://www.theregister.com/2017/10/30/ibm_hcl_notes_domino_partnership.

33. CAMPBELL KWAN. HCL now fully controls IBM software including Notes and Domino | ZDNET. *2019*. [B.r.]. Dostupné také z: https://www.zdnet.com/article/hcl-now-fully-controls-ibm-software-including-notes-and-domino/.
34. CRISTINA CARDOZA. HCL Software unveils the Volt MX low-code solution for professional developers - SD Times. *2020*. [B.r.]. Dostupné také z: https://sdtimes.com/low-code/hcl-software-unveils-low-code-solution-for-professional-developers-hcl-volt-mx/.
35. *Low-Code App Development (LCAP) For Citizen Developers - Domino Leap*. [B.r.]. Dostupné také z: https://www.hcl-software.com/domino/offerings/domino-leap?referrer=www.google.com.
36. Budoucnost je v superaplikacích - CFOworld.cz. *CFOworld*. 2021. Dostupné také z: https://www.cfoworld.cz/clanky/budoucnost-je-v-superaplikacich/.
37. FILIP ZELENKA. Revolut se chce rozkročit do šíře a stát se superaplikací | e15.cz. *e15*. 2023. Dostupné také z: https://www.e15.cz/technoverse-filipa-zelenky/finance-zazitky-a-ubytovani-revolut-se-chce-rozkrocit-do-sire-a-stat-se-superaplikaci-1399390.
38. *HCL Volt MX - Low Code App Development Platform*. [B.r.]. Dostupné také z: https://www.hcl-software.com/volt-mx.
39. *Enterprise Application Integration Platform | Create Mobile Form - HCL Leap*. [B.r.]. Dostupné také z: https://www.hcl-software.com/leap.
40. *Úvod do webových aplikací – FI WIKI*. [B.r.]. Dostupné také z: https://web.archive.org/web/20140905233724/http://kore.fi.muni.cz/wiki/index.php/%C3%9Avod_do_webov%C3%BDch_aplikac%C3%AD#Prvn.C3.AD_servlet_2.
41. *JavaScript API*. [B.r.]. Dostupné také z: https://help.hcltechsw.com/domino-leap/1.1.3/ref_javascript_api.html.
42. *Starlight Build documentation sites with Astro*. [B.r.]. Dostupné také z: https://starlight.astro.build/.
43. *Home | VuePress*. [B.r.]. Dostupné také z: https://v2.vuepress.vuejs.org/.
44. *Build optimized websites quickly, focus on your content | Docusaurus*. [B.r.]. Dostupné také z: https://docusaurus.io/.
45. *Nuxt: The Intuitive Vue Framework · Nuxt*. [B.r.]. Dostupné také z: https://nuxt.com/.

46. *Modern Site Generator for Vue.js - Gridsome*. [B.r.]. Dostupné také z: `https://gridsome.org/`.
47. *VitePress | Vite & Vue Powered Static Site Generator*. [B.r.]. Dostupné také z: `https://vitepress.dev/`.
48. *GitHub Apps documentation - GitHub Docs*. [B.r.]. Dostupné také z: `https://docs.github.com/en/apps`.
49. CONE, Matt. *The Markdown Guide*. 2020. ISBN 979-8656504492.
50. *Markdown Extensions | VitePress*. [B.r.]. Dostupné také z: `https://vitepress.dev/guide/markdown`.
51. *shikijs/shiki: A beautiful yet powerful syntax highlighter*. [B.r.]. Dostupné také z: `https://github.com/shikijs/shiki`.
52. *Themes | Shiki*. [B.r.]. Dostupné také z: `https://shiki.style/themes`.

Seznam obrázků

2.1 Transformace mezi modely v MDD [10] 10
2.2 Vennův diagram znázorňující společné rysy a rozdíly mezi přístupem řízeným modely, platformami pro low-code aplikace a low-code vývojem softwaru [16] 14
2.3 Low-code uživatelé [17] 16

3.1 Jednotlivé komponenty platformy OutSystems [24] 20
3.2 Mendix Editor stránek [26] 21
3.3 Retool editor [29] 22

4.1 Vývoj jednotlivých low-code platforem společnosti HCL (dříve IBM) . 31
4.2 Výchozí obrazovka s přehledem aplikací 33
4.3 Zobrazení stejných dat v různých částech aplikace 34
4.4 Mapování prvků grafického rozhraní na data [41] 35
4.5 Workflow v HCL Domino Leap 38

5.1 Ukázka návodu popisujícího tvorbu aplikace na platformě HCL Domino Leap 43
5.2 Ukázka stránky vysvětlující validaci vstupu od uživatele pomocí regulárního výrazu 44
5.3 Odkazy do různých částí video tutoriálu 45
5.4 HCL Domino Leap fórum 45

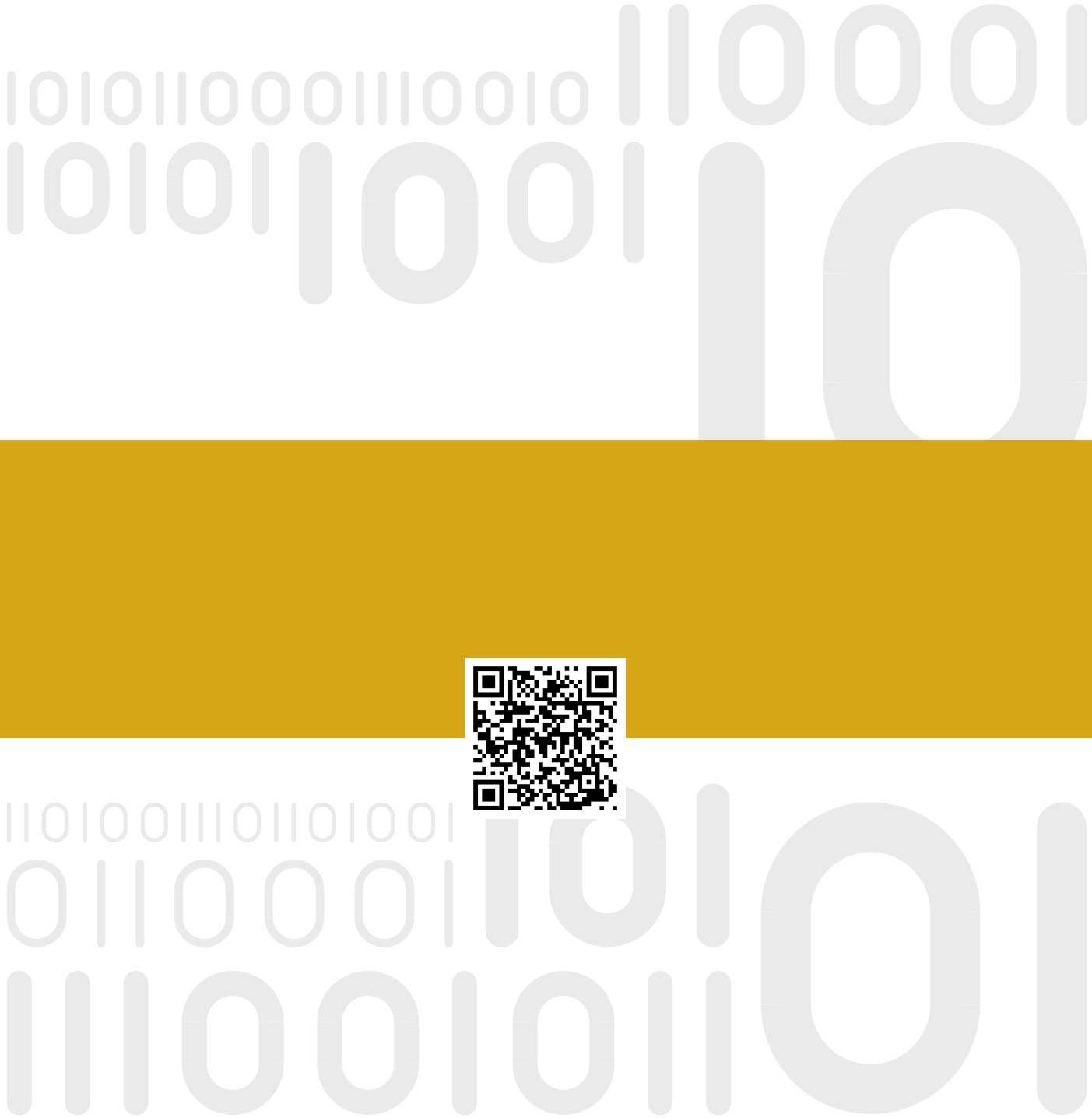
6.1 Ukázka nastavení platformy Render pro sestavení a nasazení tutoriálu 54
6.2 Ukázka toho, jak jiní uživatelé mohou přispívat do tutoriálu pomocí pull requestů 55
6.3 Grafický výstup základní syntaxe v Markdown 56
6.4 Grafický výstup kontejnerů 56
6.5 Zobrazení uživateli tutoriálu 57
6.6 Kapitola Where is your Data ze sekce Domino Leap Basics 61
6.7 Ukázka ze sekce No-code Development 62
6.8 Ukázka kapitoly z tutoriálu o událostech 64

6.9 Ukázka reference vytvořené pro objekt sekce 65

Seznam tabulek

4.1 Původ a obraz dodavatele low-code produktů [30] 24
4.2 Koncepty charakterizující statickou perspektivu 25
4.3 Koncepty charakterizující funkční perspektivu 26
4.4 Koncepty pro analýzu dynamických abstrakcí 27
4.5 Koncepty pro analýzu podpory vývoje grafického uživatelského rozhraní 28
4.6 Další aspekty analýzy low-code platforem 30
4.7 HCL Domino Leap statická perspektiva 36
4.8 Shrnutí funkční perspektivy HCL Domino Leap 37
4.9 Shrnutí dynamické perspektivy HCL Domino Leap 39
4.10 Koncepty pro analýzu podpory vývoje grafického uživatelského rozhraní 40
4.11 Další aspekty analýzy low-code platforem 41

6.1 Srovnání nástrojů pro tvorbu tutoriálu 50
6.2 Porovnání hostingových platforem 51
6.3 Základní Markdown syntaxe 55

Seznam výpisů

6.1 Ukázkový rozšířený Markdown kód pro tvorbu kontejnerů z kapitoly Hello World v tutoriálu 53
6.2 Ukázka zdrojového kódu v rozšířeném Markdownu pro zobrazení zdrojového kódu JavaScriptu se zvýrazněním syntaxe z kapitoly Service Calls v tutoriálu 56
6.3 Zahrnutí obsahu jiného Markdown souboru 57
6.4 Konfigurace Frontmatteru pro domovskou stránku 57
6.5 Část konfiguračního souboru tutoriálu 58