Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Diplomová práce

Možnosti zabezpečeného mobilního spojení s nemocničním informačním systémem

Plzeň, 2016 Ladislav Račák

TATO STRANA BUDE OBSAHOVAT ORIGINÁL ZADÁNÍ

1. Seznamte se s možnostmi zabezpečeného mobilního spojení s nemocničním informačním systémem.
2. Na platformě Android navrhněte způsob bezpečného přenosu citlivých dat z mobilního zařízení do informačního systému.
3. Návrh realizujte v prostředí experimentálního medicínského systému.
4. Ověřte funkcionalitu, zejména z hlediska bezpečného přenosu a následné identifikace importovaných dat v experimentálním medicínském systému.
5. Zhodnoťte a navrhněte možná rozšíření.

NEVKLÁDAT DO VAZBY!!!!

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne ……

……

Ladislav Račák

Poděkování

Rád bych poděkoval zejména paní docentce Janě Klečkové, za poskytnutí potřebných informací a znalostí pro návrh a realizaci této DP. Velkou oporou mi byli také rodiče a přátelé, kteří mi dodávali potřebnou podporu ve studiu.

Abstrakt

Cílem této práce je návrh a implementace zabezpečeného mobilního spojení s experimentálním nemocničním informačním systémem. Po analýze možností zabezpečení jsou tyto znalosti využity k vytvoření nové mobilní aplikace pro platformu Android komunikující s nově vytvořeným modulem experimentálního nemocničního systému. Mobilní aplikace slouží k získávání dat pomocí definovatelných formulářů.

Klíčová slova: bezpečnost, Android, šifrování, kryptografie, elektronický podpis

Abstract

The objective of this study is to design and implement secure mobile connection with an experimental medical information system. After analyzing possibilities how to secure the link is the knowledge applied for creation new application for Android platform communicating with newly created module of experimental medical IS. The mobile application is used to collect and retrieve data using definable forms.

Key words: security, Android, cipher, cryptography, digital signature

Obsah

1 Úvod

Bezpečnost je jedna z klíčových oblastí moderní společnosti. V dnešní době je zejména šifrování dat a ověření identity velmi diskutovaným tématem. Citlivé informace musí splňovat přísné bezpečností zásady a nesmí podlehnout častým útokům na prolomení. Žijeme v digitální době a už i mobilní přístroje jsou šifrované, aby zabránili útočníkům v přístupu k citlivým osobním informacím.

Tato práce se zabývá aktuálními standardy v oblasti zabezpečení informací, dále návrhem mobilní aplikace pro platformu Android a serverovou částí sloužící lékařům, ale i pacientům k pohodlnému vyplňování definovaných formulářů. Návrh počítá s variantou, kdy komunikace mezi těmito systémy bude probíhat po nezabezpečené lince, a proto je nutné informace procházející skrz tuto linku zabezpečit jiným způsobem. Práce popisuje návrh a architekturu obou nově vzniklých systémů a vzájemnou komunikaci mezi nimi. Definuje aplikační rozhraní (API) a způsob volání.

V první kapitole jsou představeny metody pro zabezpečení samotné odesílané zprávy. Tato kapitola pojednává o algoritmech pro zašifrování zprávy, provádí klasifikaci těchto algoritmů do skupin a nabízí stručný přehled o dostupných, veřejně známých a používaných algoritmech. Představeny jsou jak symetrické šifry s jedním tajným klíčem, tak i šifry asymetrické fungující na principu dvojice klíčů (veřejný a soukromý). Dále jsou v této kapitole uvedeny další principy zabezpečení zprávy, jako je například ověření integrity nebo identity odesílatele. Opět jsou uvedeny zástupci používaných algoritmů. Na závěr této kapitoly je popsáno fungování a princip elektronického podpisu včetně právních aspektů na území ČR.

Druhá kapitola je věnována popisu zabezpečené komunikace, kde jsou využívány zejména principy popsány v předchozích kapitolách, jako je šifrování, haš a podobně. Bude popsána rodina protokolu TLS, sloužící k zabezpečení komunikace na úrovni transportní a relační vrstvy, která je využívaná zejména u zabezpečené verze protokolu http – https. Dále bude představena rodina protokolu IPsec, která funguje na nižší vrstvě ISO/OSI modelu a zabezpečuje samotné pakety pomocí šifrování a speciálních hlaviček

paketu. Na závěr této kapitoly je v jednoduchosti vysvětlen princip VPN, sloužící k vytvoření virtuální (zabezpečené) sítě mezi koncovými uzly.

Další kapitoly již popisují samotnou implementaci serverové a klientské části. Nejprve je představena každá část systému zvlášť spolu s vlastním modelem užití, základní architekturou a detailním popisem každé části včetně databázového modelu nebo například dostupných rozhraní. Spolu s řešením jsou krátce popsány technologie a nástroje použité při vývoji obou částí. Dále je vysvětlen princip tvorby a správy definic formulářů včetně vyplněných dat.

Samotnému popisu fungování komunikace mezi serverovou a klientskou částí je věnována zvláštní kapitola popisující dílčí úkony jako je přihlášení, příjem a odesílaní formulářových definic apod. Na závěr kapitoly jsou identifikovány některé základní hrozby týkající se prolomení zabezpečení spolu s navrhnutými protiopatřeními.

Na závěr jsou identifikována některá možná rozšíření týkající se celého navrhovaného systému.

2 Zabezpečení informací (dat)

Následující kapitola představuje základní principy a možnosti zabezpečení informací před případným odposloucháváním nebo jiným útokem. Budou popsány základní metody šifrování, hašování a elektronického podpisu, které budou v kapitole číslo 4 použity pro zabezpečení spojení mezi klientskou a serverovou aplikací. Pozornost bude věnována jen základním a běžně používaným metodám zabezpečení a ověření identity.

Účelem zabezpečení a šifrování informací je skrýt zprávu, která může být veřejně vystavena, před neautorizovanými příjemci. Současně je také nutné ověřit, že přijatá zpráva nebyla nikterak pozměněna nebo podstrčena, kterýmkoliv jiným subjektem, než tomu určeným.

V následujících podkapitolách budou pro účely příkladů uvažování 3 aktéři[1]:

- Alice – strana odesílající/přijímací zprávu protistraně (Bobovi);
- Bob – strana přijímací/odesílající zprávu protistraně (Alici);
- Eva – útočník snažící se zprávu neprávem přečíst.

2.1 Symetrické šifry

Následující kapitola vychází zejména ze zdroje (1 stránky 11,12,15,16,19). Budou popsány základní principy symetrických šifer (šifer se sdíleným klíčem) jak blokových, tak proudových (stream).

Symetrické šifry používají pro zašifrování a dešifrování zprávy jeden klíč, kterým musí obě strany před začátkem komunikace disponovat. Alice zašifruje zprávu smluveným společným klíčem za použití šifrovacího algoritmu, odešle zašifrovanou zprávu přes nezabezpečenou linku a Bob stejným klíčem zprávu dešifruje. Algoritmus pro šifrování může, ale nemusí, být veřejně známý, protože Eva bez znalosti klíče nemůže standartními

[1] Jména aktérů jsou volena v souladu s ustálenou tradicí a zjednodušením oproti názvům jako „strana A“, „strana B“ a „útočící/odposlouchávající strana“.

prostředky zprávu dešifrovat. Nevýhodou tohoto postupu je fakt, že obě strany musí mít před začátkem komunikace k dispozici smluvený klíč. Tento problém řeší různé algoritmy a postupy předání klíče, například použití asymetrické šifry (viz kapitola 2.2).

Zvolený algoritmus E/E^{-1} pro šifrování/dešifrování musí pro konstantní klíč k splňovat bijektivní zobrazení mezi zprávou M a šifrou C:

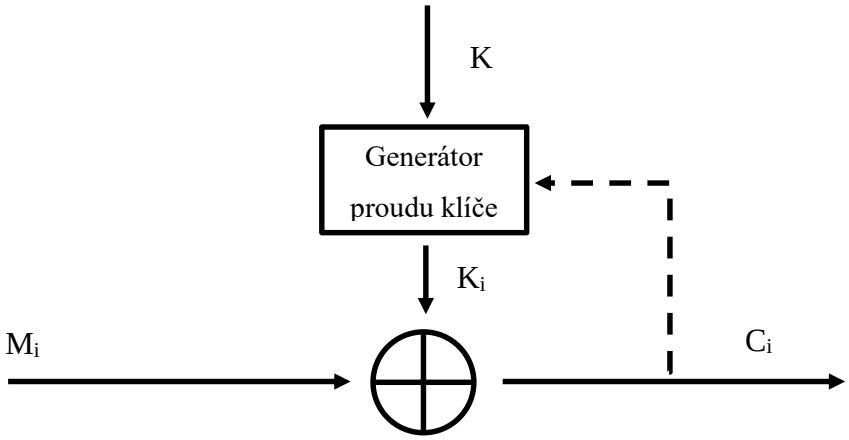
$$E(M,k) \rightarrow C, \qquad E^{-1}(C,k) \rightarrow M$$

Zároveň tento algoritmus nesmí bez znalosti klíče umožnit dešifrování šifry.

Rozlišují se dva základní typy symetrických šifer – blokové a proudové (stream). Blokové pracují s textem konstantní délky, zatímco proudové s textem znak po znaku ve formě proudu.

2.1.1 Proudové šifry

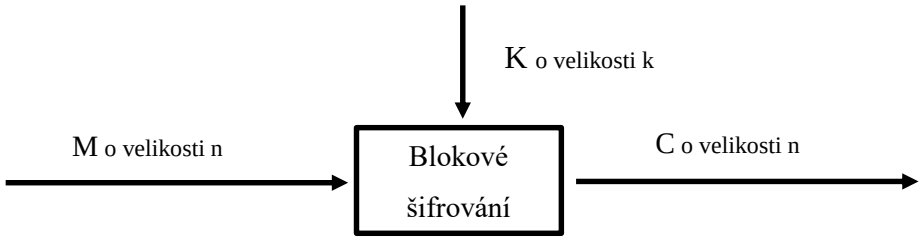
Následující kapitola vychází ze zdroje (2 stránky 30,31). Proudové šifry šifrují jednotlivé bity individuálně. Toto je dosaženo přidáním bitu z klíče do zdrojové zprávy. Proudové šifry se dají rozdělit na synchronní, kde přidávané bity záleží pouze na klíči, a asynchronní, kde přidávané bity záleží i na šifrovaném textu. Obrázek 3 znázorňuje zašifrování zprávy M na šifrovanou zprávu C pomocí klíče K (index i značí jednotlivé bity). Přerušovaná čára označuje tvorbu proudu klíče pro asynchronní šifry. Hlavní zástupci proudových šifer jsou například FISH (Fibonacci Shrinking), A5/1 apod.



Obrázek 1 Proudové šifrování (synchronní vs. asynchronní)

2.1.2 Blokové šifry

Blokové šifry šifrují celý blok zprávy současně za použití stejného klíče. Bloková šifra je funkcí, která mapuje n-bit dlouhý blok zprávy na n-bit dlouhý blok šifry (n je délka bloku). Funkce je parametrizována klíčem K o velikosti k. Následující obrázek znázorňuje fungování blokové šifry.



Obrázek 2 Bloková šifra

Hlavní zástupci blokových šifer jsou algoritmy AES (Advance Encryption Standard), DES nebo 3DES.

Mezi všemi šifrovacími algoritmy jsou symetrické šifry z hlediska hardwarové i softwarové implementace nejrychlejšími. Využití mají zejména v šifrování velkého objemu dat za použití chráněného klíče. Pro výměnu informací po nezabezpečené lince se lépe hodí asymetrické šifry (šifry s veřejným klíčem), které jsou popisovány v následující kapitole.

2.2 Asymetrické šifry (šifry s veřejným klíčem)

Tato kapitola se zabývá technikami pro šifrování s veřejným klíčem, také nazývané jako asymetrické šifry, vychází zejména ze zdroje (2 stránky 283,284). Při použití této metody je zapotřebí, aby všechny zúčastněné strany měly k dispozici veřejný klíč pro zašifrování a svůj privátní klíč (neveřejný klíč) pro dešifrování. Zároveň nesmí být v bezpečném systému možné získat privátní klíč při znalosti veřejného klíče. Nejznámější algoritmy (například: RSA nebo Rabin) využívají k ochraně soukromého klíče problém faktoriálu přirozených čísel, tak aby bylo výpočetně velmi náročné spočítat privátní klíč. Asymetrické šifry fungují na principu bloků bitů (viz kapitola 2.1.2).

Následující teoretický příklad popisuje nutný postup pro odeslání citlivých informací mezi Alicí a Bobem:

1. Alice si vyžádá veřejný klíč protistrany (Bob);
2. Alice zašifruje zprávu pomocí získaného klíče;
3. Alice odešle zašifrovanou zprávu po (ne)zabezpečené lince;
4. Bob přijme zprávu a dešifruje pomocí soukromého klíče, který tvoří pár s veřejným klíčem.

Při šifrování jsou data rozdělena do bloků určité velikosti a popřípadě je použit mód způsobu určení vstupů do algoritmu šifrování a doplňování bitů. Mezi nejznámější módy patří ECB (Electronic Codebook) nebo CBC (Cipher Block Chaining). Při použití ECB módu je zdrojová zpráva rozdělena do bloků a každý blok šifrován zvlášť. Mód CBC provede před samotným šifrováním operaci XOR mezi předchozím zašifrovaným blokem a aktuálním blokem zprávy, popřípadě inicializačním vektorem, pokud se jedná o první blok. Doplňování bitů znemožňuje útok na takzvaně často opakující se fráze nebo slovní spojení, jako je například oslovení. Ke každému šifrovanému bloku jsou přidána data navíc. Způsob doplňování bitů je definováno například PKCS (Public Key Cryptographic Standards).

Nevýhodou algoritmů s veřejným klíčem je výpočetní náročnost tvorby dvojice klíčů a samotného šifrování. Dalším rozhodujícím faktorem pro využití těchto algoritmů je fakt, že maximální délka šifrované zprávy závisí na délce klíče. Například u algoritmu RSA za použití 2048 bitů dlouhých klíčů je možné zašifrovat jen 245 bytů (1960 bitů) dlouhou zprávu. Tento problém se řeší použitím symetrické šifry, kde klíč použitý pro šifrování a dešifrování je zašifrován asymetrickou šifrou, ale data samotná jsou šifrována šifrou symetrickou.

2.3 Hašovací funkce

Následující kapitola vychází zejména ze zdroje (2 stránky 321, 322). Kryptografické hash (hašovací) funkce hrají fundamentální roli v moderní kryptografii. Používají se zejména k ověření integrity a autentifikace zprávy. Hašovací funkce přijímají zdrojovou zprávu jako vstup a vytváří tzv. haškód (hashcode) / otisk zprávy. Přesněji hašovací funkce H

mapuje řetězec bitů M libovolné konečné délky m na řetězec bitů N konečné délky n, kde m může být větší než n. Z principu je tato funkce neprostá, může nastat situace, kde pro rozdílné vstupní zprávy M_1, M_2 bude platit, že $H(M_1)=N$ a současně $H(M_2)=N$. Pravděpodobnost tohoto jevu je 2^{-n} (nezávislé na velikosti m).

Haš funkce se dají rozdělit do dvou hlavních skupin:

- Detekování modifikací MDC (modification detection codes), nebo také detekování manipulací a ověření integrity. Využívají se neklíčové haš funkce.
- Ověřování zprávy MAC (message authentication codes). Zde najdou využití haš funkce s dalším parametrem, který tvoří klíč.

Typický příklad použití neklíčové haš funkce pro ověření integrity je následující. Spočítá se haš/otisk pro nějakou danou zprávu. Integrita tohoto otisku je nějakým způsobem chráněna (například použitím šifry), zpráva samotná být chráněna nemusí. Po přijetí zprávy a otisku je spočítán otisk znovu za pomocí stejné haš funkce pro přijatou zprávu. Pokud se spočítaný otisk rovná přijatému, zpráva nebyla modifikována ani manipulována. Výhoda je ve velikosti otisku, protože zpráva samotná je libovolně velká, může být problematické (výpočetně/kapacitně) zajistit její integritu například šifrou. Zatímco otisk je vždy fixní délky a není žádný větší problém zajistit jeho integritu.

Klíčová haš funkce se používá pro ověření správného původu zprávy. Řeší problém, kdy útočník změní zprávu i otisk. Ten ale posléze neodpovídá otisku vytvořeném za použití správného klíče.

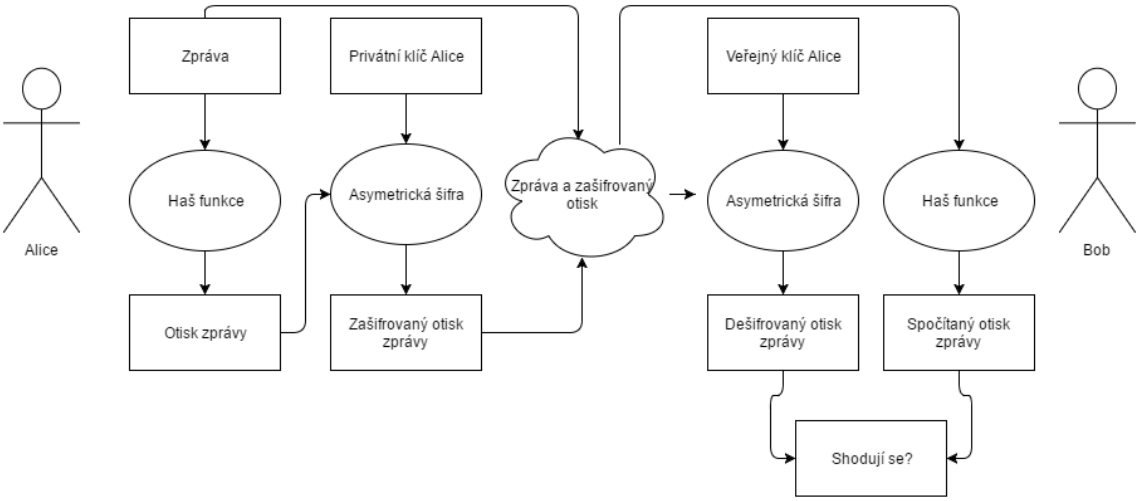
Typickými zástupci haš algoritmů jsou SHA1, SHA2, SHA3 nebo MD5.

2.4 Elektronický podpis

Tato kapitola popisuje fungování elektronického podpisu, jeho vyžadované vlastnosti a porovnává ho s klasickým ručním podpisem. Kapitola vychází zejména ze zdroje (2 stránky 427, 428).

Elektronický podpis slouží k podepsání elektronických dat a tím zaručení integrity a autenticity zprávy. Využívá princip asymetrické kryptografie. Nejprve podepisující

použije na zdrojovou zprávu hašovací funkci a tím vytvoří její otisk zaručující integritu zprávy (viz kapitola 2.3). Dále se tento otisk/haš zašifruje definovaným algoritmem pomocí privátního klíče podepisujícího. V této chvíli mohou být data a zašifrovaný otisk poslány příjemci. Příjemce zašifrovaný otisk zprávy dešifruje pomocí veřejného klíče odesílatele, tím získá otisk zprávy, který porovná se spočítaným otiskem zprávy nad přijatými daty. Pokud se shodují, zpráva je ověřena, pochází od odesílatele a nebylo s ní manipulováno. Celý proces je zobrazen na následujícím obrázku, kde Alice plní roli odesílatele a podepisující strany, Bob zprávu přijímá a ověřuje elektronický podpis.



Obrázek 3 Průběh tvorby a ověření elektronického podpisu

Obdobně jako je tomu u klasického podpisu, kde je zpráva (dokument) ověřena unikátním tvarem, velikostí a intenzitou ručního podpisu, tento podpis je ověřen unikátním veřejným klíčem podepisující strany, který s jeho privátním tvoří pár. Žádným jiným klíčem by zašifrovaný otisk neměl jít dešifrovat.

Pozn.: V České republice je elektronický podpis zakotven i v právním systému pod předpisem č. 227/2000 Sb. (viz zdroj (3)). Tento předpis definuje termín „zaručený elektronický podpis“, který musí splňovat následující kritéria:

1. je jednoznačně spojen s podepisující osobou,
2. umožňuje identifikaci podepisující osoby ve vztahu k datové zprávě,
3. byl vytvořen a připojen k datové zprávě pomocí prostředků, které podepisující osoba může udržet pod svou výhradní kontrolou,
4. je k datové zprávě, ke které se vztahuje, připojen takovým způsobem, že je možno zjistit jakoukoliv následnou změnu dat.

Legislativa dále vymezuje termín „uznávaný elektronický podpis“, který musí splňovat podmínku podle §3 odstavce 2, aby elektronický podpis byl vytvořen na základě kvalifikovaného certifikátu a obsahoval údaje, které umožní jednoznačnou identifikaci odesílatele jako osoby. Kvalifikovaný certifikát v současné době lze získat jen od vybraných certifikačních autorit:

- Certifikační autorita PostSignum pod Českou poštou;
- Certifikační autorita eIdentity a.s;
- První certifikační autorita, a.s.

3 Zabezpečená komunikace

Následující kapitola představuje základní principy a možnosti zabezpečení komunikace přes veřejnou síť internet (nebo jinou, ve výchozím stavu nezabezpečenou počítačovou sítí) mezi klientem a serverem. Budou popsány jen základní a běžně používané metody zabezpečení, jako je SSL (respektive TLS), IPsec a VPN. Níže uvedené metody vychází z představených principů pro zabezpečení dat.

3.1 TLS (SSL)

Následující kapitola vychází zejména ze zdroje (4) a (5).

Transport Layer Security (TLS) a jeho předchůdce, Secure Sockets Layer (SSL) jsou kryptografické protokoly, které poskytují možnost zabezpečení komunikace přes počítačovou síť. Fungují mezi 4. a 5. vrstvou ISO/OSI modelu. Protokol TLS je hojně využívaný zejména pro zabezpečení http protokolu pro komunikaci mezi klientem a serverem na síti internet. Protokol SSL byl od verze 3.0 přejmenován právě na TLS, kvůli možným licenčním problémům se společností Netscape Communications Corporation.

Princip TLS lze rozdělit do následujících tří fází:

1. dohodu účastníků na podporované verzi TLS a algoritmech, které budou použity;
2. výměna veřejných klíčů pro asymetrické šifrování (s veřejným klíčem) a autentizace vycházející z certifikátů;
3. samotné šifrování provozu symetrickou šifrou.

Server je považován za ověřenou stranu hned z počátku. Jeho certifikát (obsahující veřejný klíč) musí být digitálně podepsán nějakou uznávanou autoritou (certifikační autoritou) na straně klienta.

TLS je využíván v celé řadě dalších protokolů jako například HTTPS, SMTPS apod.

3.2 IPsec

V této kapitole je čerpáno ze zdroje (6).

IPsec neboli Internet Protocol security je bezpečnostní mechanismus na kryptografické bázi rozšiřující funkci 3. vrstvy ISO/OSI modelu, tudíž funguje přímo nad každým paketem a není nutná úprava na vyšších vrstvách. IPsec definuje dva bezpečnostní mechanismy:

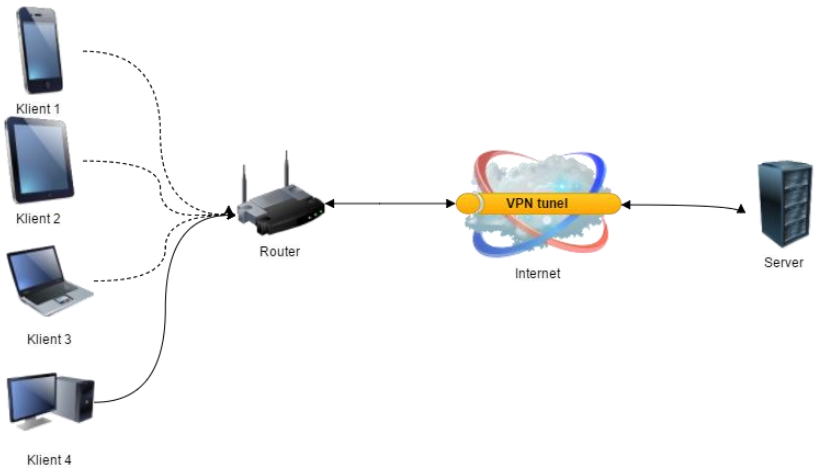
- Autentifikace – verifikace, že přijatá data pochází od skutečného odesílatele. Využívá se Authentication Header (AH)
- Kryptografie – zašifrování zprávy pomocí předem domluveného algoritmu. Využívá se Encapsulating Security Payload (ESP).

IPsec vytváří jednosměrné logické kanály, kde při využití ESP jsou pakety zašifrovány a opatřeny hlavičkou ESP, která definuje, jak pakety dešifrovat. Při použití AH je k odesílaným paketům připojena AH hlavička paketu. Ta slouží k ověření integrity a původu přijatých dat.

3.3 VPN tunel

Virtuální privátní síť (Virtual Private Network -VPN) je rozšíření klasické privátní sítě, které umožňuje komunikaci i přes sdílené nebo veřejné připojení (například internet). Chování je pro uživatele (aplikace) transparentní a je stejné, jako kdyby byly koncové uzly připojeny do jedné privátní sítě. Princip je založen na vytvoření vpn tunelu mezi koncovými body infrastruktury, viz Obrázek 4 VPN tunel. Aby se mohlo považovat spojení přes vpn jako zabezpečené, jen nutné použít některý z dostupných protokolů, například TLS (viz kapitola 3.1**Chyba! Nenalezen zdroj odkazů.**), IPsec (kapitola 3.2), Datagram Transport Layer Security (DTLS), Secure Shell (SSH) a podobně.

Výhodou použití vpn je transparentnost pro koncové uživatele (aplikace), jakmile je nastaven vpn tunel, komunikace probíhá skrze něj standartním způsobem, o směrování paketů a zabezpečení se stará právě jen software zajišťující vpn.



Obrázek 4 VPN tunel

3.4 Zabezpečení na aplikační úrovni

Dalším ze způsobů jak zabezpečit komunikaci přes nezabezpečenou linku je úprava přímo na aplikační úrovni ISO/OSI modelu. V takovém případě je nutné, aby obě strany používaly stejný princip komunikace. Výhoda tohoto řešení je možnost návrhu přímo pro konkrétní případ užití a přizpůsobit tak komunikaci požadavkům na celý systém. Na rozdíl například od protokolu TLS a IPsec, které fungují na bázi ověřování zařízení nebo koncové aplikace, umožňuje úprava na aplikační úrovni ověřovat přímo uživatele dané aplikace. Nemusí být vázána na kvalifikovaný certifikát a tak podobně.

Tato práce se zaměřuje právě na vlastní řešení, které využívá standardní metody pro zabezpečení dat, které byly představeny v kapitole 2. Konkrétní návrh komunikace je popsán v kapitolách níže.

4 Řešení

Tato kapitola a její podkapitoly popisují samotný návrh a implementaci zabezpečené komunikace mezi klientskou aplikací na platformě Android a serverovou částí, tvořící experimentální nemocniční systém. Každé části je věnována zvláštní kapitola, budou popsány použité technologie, architektura a implementace. Následně bude vysvětlen princip komunikace mezi těmito systémy včetně posílaných dat.

Všechny zdrojové kódy včetně instalačních souborů pro obě části systému jsou umístěny na přiloženém CD. Pro vývoj byl využíván i systém pro správu změn, konkrétně GIT, jehož vzdálený repositář je umístěn na serverech ZČU.

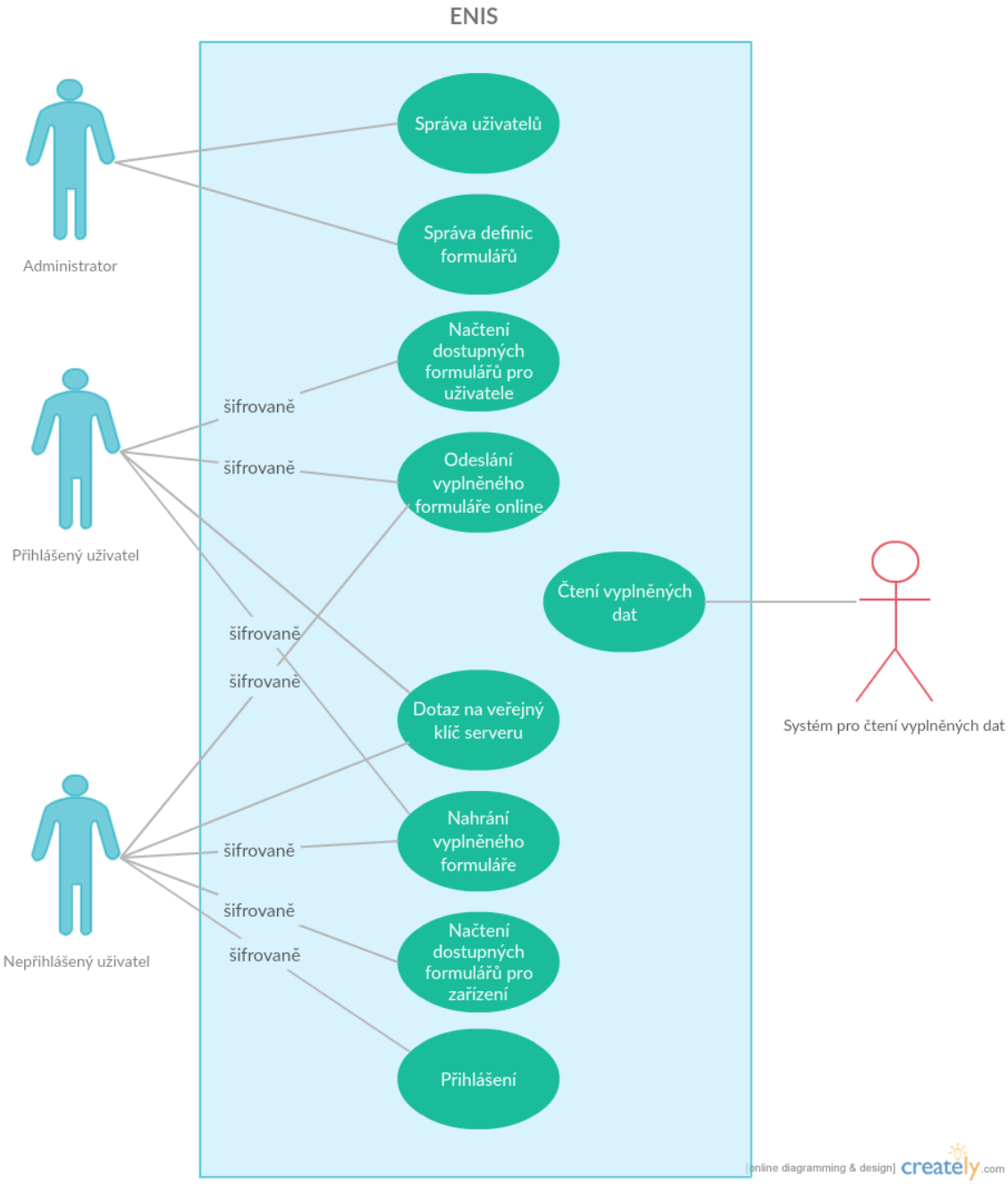
4.1 Serverová část (ENIS)

Serverová část představuje modul pro experimentální nemocniční informační systém (zkráceně ENIS) sloužící zejména k příjmu vyplněných formulářových dat z klientské aplikace. Dále umožňuje správu a ověření uživatelů, správu definic formulářů a další. Hlavním úkolem serverové části je zpracování příchozích formulářů a to buď online - nahráním na server přes http/https protokol, tak i offline – nahráním přímo do specifické složky na serveru.

Zdrojové kódy jsou umístěné na přiloženém CD ve složce source_codes/ENIS. Soubor enis-backend.war sloužící k nahrání na aplikační server se nachází ve složce dist.

4.1.1 Případy užití (use case) serverové části

Následující obrázek znázorňuje diagram případů užití (use case) definující jednotlivé požadavky na serverovou část.



Obrázek 5 Use case diagram serverové části

Administrátor může spravovat uživatelské účty, vytvářet nové uživatele včetně určení hesla. Dále určuje definice formulářů pro zařízení a uživatelské účty. Fungování definic formulářů včetně jejich vytváření je popisováno v kapitole 4.3.

Přihlášenému i nepřihlášenému uživateli je umožněno odesílat vyplněné formuláře online přes REST rozhraní, tak i offline nahráním souboru do specifikované složky a získat veřejný klíč serveru používaný k šifrování dat. Každý typ uživatele (přihlášený/nepřihlášený) má určené svoje definice formulářů, a proto se jedná o různé případy užití. Nepřihlášený uživatel je vázán na unikátní identifikátor zařízení a může se přihlásit, čímž se stává přihlášeným uživatelem. Poslední činitel je systém pro čtení vyplněných dat, kterému je umožněn přístup do složky, kam se nahrávají dešifrované vyplněné formuláře. Tento systém je může následně jakkoliv dále zpracovávat.

4.1.2 Architektura serverové části

V této kapitole bude popsána architektura serverové části, tedy struktura projektu, základní uspořádání jednotlivých komponent a vazby mezi nimi. Ve stručnosti budou popsány technologie, frameworky, knihovny a další nástroje, které byly použity v rámci vývoje serverové části systému.

4.1.2.1 Použité technologie

Programovací jazyk

Serverová část byla vyvíjena na platformě Java, konkrétně ve verzi 1.8.0_91, která oproti předchozím verzím (1.7, 1.6 atd.) umožňuje pokročilejší práci s kolekcemi ve formě takzvaných streamů (proudů), což výrazně zjednodušuje práci s různými seznamy zejména při filtrování a procházení. Další novinkou Java verze 1.8 jsou takzvané lamba výrazy, které opět zjednodušují zápis a to zejména při tvorbě takzvaných anonymních tříd, více informací, včetně příkladů použití, je možné nalézt ve zdroji (7). Dále pak bylo v práci využito nového Java time api, které změnilo způsob práce s daty a časy (více o tomto viz zdroj (8)) .

Správa závislostí / tvorba balíčku aplikace

Pro správu závislostí a tvorbu instalačního balíčku aplikace byl použit nástroj Apache Maven (viz zdroj (9)). Ten umožňuje definování závislostí, instalačních/testovacích skriptů, modulů a další v souboru pom.xml, který představuje takzvaný „project object model“ – popis projektu (jméno, vlastník, závislosti apod.). Dále uvedené použité knihovny a frameworky jsou dostupné vždy v centrálním repositáři Maven.

Běhové prostředí

Serverová část aplikace je navržena pro běh pod aplikačním serverem Apache Tomcat (10). Není však přímo vázána na právě tuto implementaci Java Servletů. Aplikace by tak mohla běžet eventuálně i v jiných aplikačních serverech. Byla ovšem testována jen pod Apache Tomcat, konkrétně ve verzi 8.0.32 a 7.0.68.

Aplikační Framework

Jako hlavní aplikační Framework serverové části byl zvolen Spring Framework (viz (11)), konkrétně ve verzi 4.2.4.RELEASE. Spring Framework a jeho další části jako například Spring Security, Spring Integration je velmi rozšířený Framework pro aplikace psané v programovacím jazyku Java. Umožňuje tzv. dependency injection (vkládání závislostí), která umožňuje tvorbu aplikačního rozhraní tak, že komponenta, která využívá api druhé, nemusí v době sestavování aplikace znát její implementaci. To ulehčuje tvorbu modulů aplikace a v případě potřeby jednoduché nahrazení dané implementace za jinou při zachování ostatních modulů. Vkládání závislostí při použití Spring Framework funguje následovně:

- Programátor označí požadovaný typ (nebo metodu vracející typ) anotací @Autowired. Například @Autowired UserService userService;
- Při inicializaci aplikace vyhledá Spring Framework možné implementace tohoto typu anotované @Component, @Service @Repository, nebo uvedené v kontextu aplikace. Například @Service UserServiceImpl implements UserService;
- Požadovaný typ je naplněn nalezenou implementací.

Spring Framework dále umožňuje například zjednodušenou tvorbu kontrolerů pomocí anotací @Controller a @RequestMapping("/").

Databáze/ORM

Pro objektově relační mapovaní a persistenci dat byla použita knihovna Hibernate ve verzi 5.1.0.Final spolu s Spring Data JPA. Jako systém řízení báze dat (SŘBD) byla pro svoji jednoduchost instalace a použití zvolena H2 databáze (viz zdroj (12)). Jedná se vestavěnou databázi, která běží současně s aplikací. H2 databáze spravuje svá data buď přímo v paměti, nebo ve specifikovaném souboru. Spring Data JPA umožňuje tvorbu tzv. repositářů, které jsou deklarované jen jako rozhraní s metody odpovídající dané konvenci.

Knihovna se pak sama postará o implementaci tohoto rozhraní, tvorbu SQL dotazů a ORM mapování.

Logování

Serverová část využívá k zápisu aplikačních logů knihovnu Simple Logging Facade for Java (SLF4J) a Logback Project. Definice aplikačních logů je uložená například v souboru logback-test.xml, který umožňuje definovat formát výstupu, typ a například rolování logů, což umožňuje definovat pravidla, kdy se mají aplikační logy přesunout do archivního souboru.

Testy

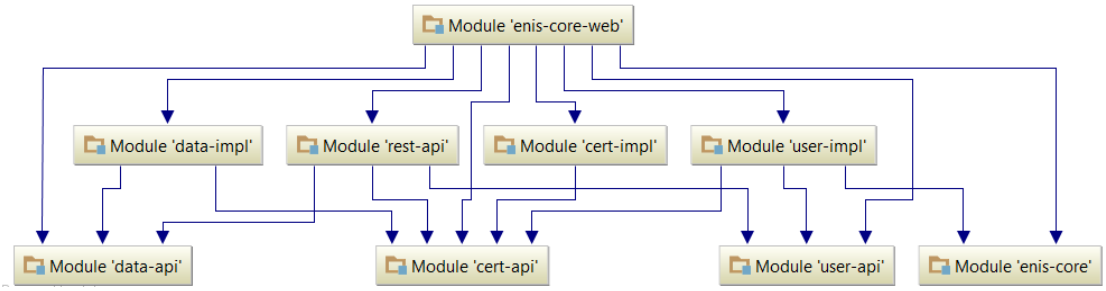
Serverová část využívá pro podporu jednotkových (unit) testů knihovnu JUnit ve verzi 4.12, která společně s komponentami frameworku Spring umožňuje používat „SpringJUnit4ClassRunner“ sloužící například k injektování závislostí a spouštění testů. Tvorba testů pak probíhá jednoduše za použití anotací @RunWith a @Test.

Další knihovny a nástroje

Jako další podpůrné knihovny a nástroje byly použity například projekt Lombok (zdroj (13), který se stará o tvorbu často se opakujících programových bloků, jako jsou například gettery a settery, konstruktory nebo definice logu. Dále byla použita knihovna Spring Ingeration, která umožňuje použití různých kanálů pro vstup souborů. Dokáže periodicky kontrolovat složku a volat definovanou metodu při nalezení nového souboru. Pro tvorbu REST rozhraní a umožnění komunikace na bázi JSON objektů byla použita knihovna Jackson JSON Processor (viz (14)). Tato knihovna dokáže z/do standardních Java POJO objektů konverzi do/z JSON formátu.

4.1.2.2 Členění projektu

Serverová část byla vytvořena od základu nová. Projekt byl pojmenován ENIS představující zkratku Experimentální Nemocniční Informační Systém. Všechny třídy a rozhraní jsou umístěny v balíčku „cz.zcu.kiv.enis“. Díky použití Apache Maven je projekt roztříděn do několika modulů (viz následující obrázek).



Obrázek 6 Vztah závislostí jednotlivých modulů serverové aplikace

Hlavní modul, který obsahuje závislosti na ostatních, je enis-core-web. Poskytuje také hlavní soubor web.xml, obsahující definici webové aplikace a základní konfiguraci frameworku Spring. Dále je zde základní soubor s definicí kontextu tohoto modulu (soubor enis-core-web-context.xml), kde se uvádí použité konvertory pro převod Java objektů na http odpovědi a příchozí parametry na Java objekty (více bude tato funkcionalita popsána u modulu rest-api). Při sestavování aplikace pomocí „maven package“ je právě v tomto modulu ve složce target vygenerován výsledný WAR soubor sloužící k nasazení na aplikační server.

Dalším modulem aplikace je enis-core, který obsahuje zejména definici připojení do H2 databáze, nastavení pro logování (soubor logback-test.xml) a testovací kontext. Tento kontext je používán ve vytvořených unit (jednotkových) testech, kde je potřeba práce s databázovou vrstvou. Dále obsahuje inicializační SQL skript pro vytvoření potřebných tabulek schématu. Samotné DB schéma, včetně ERA modelu bude popsáno dále. Tento modul také obsahuje soubor fallback.properties, který je součástí sestavení aplikace a slouží jako poslední možnost v seznamu hledání konfigurace (viz kapitola 4.1.3.3).

Modul, který spravuje požadavky na server, se nazývá rest-api. V kontextu modulu jsou definovány interceptory (zachycovače událostí/požadavků) sloužící k zabezpečení určitých požadavků na server. Dále obsahuje všechny kontrolery aplikace a DTO – data transfer object, objekty používané jen k přenosu dat, které jsou pomocí konvertoru převedeny na JSON a odeslány přes HTPP/HTTPS protokol. Převod zajišťuje knihovna „com.fasterxml.jackson“, která z datového objektu Java vytvoří JSON reprezentaci a naopak.

Dalšími moduly aplikace, které poskytují jen API pro ostatní moduly, obsahují tedy jen rozhraní definující metody, možné výjimky, DTO a entity, jsou data-api, cert-api a user-api. Tyto moduly představují komunikační rozhraní mezi ostatními moduly. Každý tento

modul má svoji implementaci pojmenovanou vždy *-impl. Právě tyto implementace a všechny již výše představené moduly používají jen definované aplikační rozhraní v modulech *-api.

Moduly cert-api a jeho implementace cert-impl poskytují metody pro správu a generování klíčů používaných při zabezpečování odesílaných dat. Dále také nabízí metody pro šifrování, dešifrování, hašování dat a tvorbu elektronického podpisu.

Moduly data-api a data-impl umožňují práci s definicemi formulářů, jejich načítání a správu pro jednotlivé uživatele nebo koncové zařízení. Modul obsahuje také metody pro uložení vyplněných formulářů do specifikovaného adresáře.

Poslední sadou modulů je use-api a user-impl, které slouží ke správě a persistenci uživatelů a koncových zařízení. Zároveň tento modul obsahuje nástroj pro generování nových uživatelů, který bude popsán dále.

Posledním modulem, který zastřešuje celou serverovou část je kořenový modul ENIS, který je uveden vždy jako rodič u svých dílčích modulů a obsahuje definice používaných verzí knihoven třetích stran. Dále obsahuje některé základní závislosti aplikace, které jsou používány napříč moduly, tak aby nemusely být uvedeny u každého zvlášť. Tento modul slouží také pro spouštění příkazů nástroje Maven, a tím pro sestavení celé serverové části.

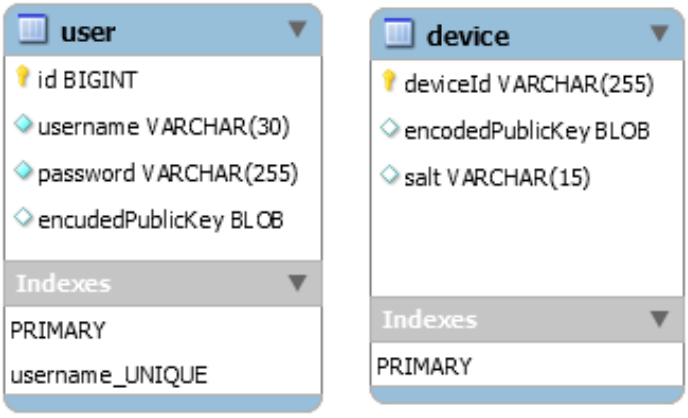
4.1.3 Implementace serverové části

Tato kapitola popisuje samotnou implementaci a způsob fungování serverové části. Navržené funkčnosti vycházejí z výše uvedených případů užití. Na zdrojovém CD ve složce ‘diagrams‘ se nachází také soubor ‘enis-class-diagram.png‘ představující diagram tříd všech vytvořených tříd a rozhraní aplikace ENIS.

4.1.3.1 Databázový model

Následující obrázek představuje databázový model. Model obsahuje jen dvě entity „user“ a „device“. User eviduje atributy username typu variabilní znak, nebo také text o maximální délce 30 znaků, sloužící k uchování uživatelského jména, dále atribut password stejného typu pro uložení otisku hesla a poslední atributem je encodedPublicKey binárního typu sloužící k uchování veřejného klíče pro daného uživatele. Primárním klíčem je automaticky generované id. Tabulka zároveň obsahuje

omezení v podobě unikátního indexu na atributu username. Druhou tabulkou ve schématu je „device“, která se váže ke konkrétnímu přístroji. Obsahuje atributy endodedPublicKey binárního typu, který slouží opět pro uchování veřejného klíče pro dané zařízení a atribut salt, který se používá při šifrování uživatelského jména a hesla uživatele během výměny informací (samotný princip zabezpečené komunikace je popsán v kapitole 4.4). Primárním klíčem je přímo unikátní identifikátor zařízení textového typu. Jedná se o takzvané Android ID, které by mělo být unikátní napříč všemi zařízeními s operačním systémem Android.



Obrázek 7 ERA model serverové části

Tabulky mezi sebou nemají žádnou vazbu. Z pohledu serverové části není důležité z jakého zařízení uživatel přistupuje, tudíž není nutné evidovat tuto informaci a vytvářet M:N vazby.

4.1.3.2 REST rozhraní

Enis obsluhuje 3 různé typy REST požadavků od uživatelů:

- Rozhraní, kdy při přístupu nejsou požadovány žádné speciální parametry, ani není prováděna žádná kontrola a šifrování.
- Rozhraní, kdy je vyžadována speciální hlavička („X-deviceId“) identifikující zdrojové zařízení. URL adresa obsahuje prefix „device/“.
- Poslední je rozhraní vyžadující autentifikaci uživatele. URL adresa obsahuje prefix „user/“. Požadavky obsahující tento prefix musí obsahovat speciální hlavičky „X-encodedUsername“, „X-encodedPassword“ a „X-deviceId“. První dvě jsou používány k ověření uživatele a nesou zašifrované uživatelské jméno a heslo. Poslední hlavička představuje identifikaci zdrojového zařízení.

Základní vstupní bod aplikace „http://<server>/<kontext>/“ (metoda GET) obsluhuje kontroler, který ověří aktuální nastavení instalace Javy, pro správný chod serverové části je nutná instalace Javy s neomezenou velikostí šifrovacího klíče. K tomu slouží soubory JCE (Java Cryptography Extension), které je potřeba umístit do složky <JAVA_HOME>/jre/lib/security. Po úspěšném nahrání souborů je možné používat větší než 128 bitů dlouhé šifrovací klíče. Pokud je při přístupu na úvodní stránku aplikace (základní vstupní bod) maximální dostupná délka menší než požadovaná, je zobrazena krátká zpráva informující právě o nutnosti instalace JCE.

Dalším požadavkem nevyžadujícím žádnou speciální autentifikaci je získání veřejného klíče serveru na adrese /publicKey metodou GET. Zároveň je možné odeslat id zařízení v požadavku a server vygeneruje kryptografickou sůl (salt) pro použití při šifrování hesla uživatele. Všechna binární data jsou pro přenos kódována Base64. Base64 je kódování založené na 64 znacích (malá, velká písmena anglické abecedy, číslice 0-9 a znaménka plus a lomítko), které převádí binární data na tisknutelné znaky (více viz zdroj (15)). Příklad požadavku a odpovědi ukazuje následující tabulka (data byla oříznuta z důvodu přehlednosti):

Požadovaná URL	http://localhost:8080/publicKey/androidID
Požadovaná metoda	GET
Odpověď	{"algorithm":"RSA", "encodedKey":"MIIBIjAN….9QIDAQAB", "salt":"$187115b7-9"}

Tabulka 1 Příklad požadavku a odpovědi na veřejný klíč serverové části

Odpověď serveru obsahuje jméno algoritmu, který byl použit pro vygenerování páru soukromého a veřejného klíče, samotný veřejný klíč kódovaný v Base64, a pokud požadavek obsahoval identifikaci zařízení je vrácena i neprázdná sůl. Zároveň proběhlo uložení vygenerované soli pro dané zařízení pro možnost pozdějšího použití při ověřování přijatého hesla.

Další sada požadavků slouží k registraci veřejných klíčů zařízení, respektive uživatele. Jedná se požadavky typu POST na adresu /device/publicKey pro zařízení, respektive /user/publicKey pro uživatele. Oba požadavky vyžadují, aby tělo obsahovalo objekt typu SecuredRestObject, který je používán při přenosu zakódovaných nebo zašifrovaných zpráv. Obsahuje identifikaci zařízení a objekt představující obsah požadavku (payload).

Jak již bylo řečeno výše, musí oba typy požadavků obsahovat dané hlavičky. Pokud požadavky splňují všechna kritéria, je přijatý klíč uložen k danému zařízení, respektive uživateli pro pozdější použití při ověřování elektronického podpisu. Oba požadavky vracejí hodnotu pravda (true), pokud byl veřejný klíč úspěšně uložen.

Poslední kontroler slouží k poskytování definic formulářů a příjem vyplněných. Obsluhuje následující požadavky:

- /user/availableForms/<uživatelské jméno> - slouží k načtení definic formulářů pro dané uživatelské jméno.
- /device/availableForms/<identifikátor zařízení> - slouží k načtení definic formulářů pro dané zařízení.
- /user/form/<uživatelské jméno> - ukládání vyplněného formuláře pro dané uživatelské jméno.
- /device/form/<identifikátor zařízení> - ukládání vyplněného formuláře pro dané zařízení.

Všechny požadavky pracují s datovou třídou SecuredFormDto, která obsahuje položky pro uložení elektronického podpisu, zašifrovaného symetrického klíče a samotná zašifrovaná formulářová data. Pro správné zpracování musí opět požadavky obsahovat konkrétní definované hlavičky.

4.1.3.3 Další komunikační a aplikační rozhraní

Serverová část poskytuje další vnější rozhraní pro správu uživatelů a formulářových definic pro konkrétní uživatele nebo zařízení. Tato funkčnost je přístupná pouze kvalifikované osobě, která musí disponovat přístupem do definovaných složek na souborovém systému. ENIS umožňuje specifikovat základní konfigurační parametry pro svůj chod uložené v souboru typu properties[2] třemi různými způsoby:

1. Definování systémové proměnné při spouštění aplikačního serveru odkazující na příslušný soubor properties, například:
    - -Dapp.config.file=file:C:\ENIS\config\app.properties

[2] Properties soubor je klasický textový soubor s položkami ve formátu klíč=hodnota. Klíč může být strukturován pomocí tečkové notace, např.: key.subkey=value.

2. Umístění konfiguračního souboru app.properties do složky config, jejíž umístění je relevantní k aktuálnímu běhovému prostředí aplikačního serveru, např:
    - C:/tomcat/bin/config/app.properties
3. Specifikování přímo v souboru fallback.properties v classpath modulu „enis-core“. Jedná se o poslední záložní řešení, které by mělo být vždy k dispozici, protože soubor je exportován při sestavování aplikace a je tedy její nedílnou součástí.

Načítání konfiguračního souboru je prováděno prioritně podle výše uvedeného seznamu, to znamená, že nejprve je zjištěno, zdali je použit systémový parametr, v případě že ne, nebo odkazuje na neexistující soubor, je použitý druhý přístup; jako poslední je třetí možnost. Načítání je možné upravit v souboru enis-core-context.xml, kde je definována třída DiscoveryPropertySourcesConfigurer, která slouží právě k načítání properties souboru. Načtený properties soubor musí obsahovat následující položky:

- security.keySize – velikost generovaných klíčů v bitech (např.:2048);
- security.algorithm – algoritmus pro generování klíčů (např.: RSA);
- security.publicKeyFilename – cesta pro uložení veřejného klíče serverové části (např.: C:/temp/enisPublicKey);
- security.privateKeyFilename – cesta pro uložení privátního klíče serverové části. K tomuto souboru by měl mít přístup jen proces serverové části, tak aby bylo zabráněno jeho čtení ostatními, popřípadě jeho modifikace (např.: C:/temp/enisPrivateKey);
- security.usersFile – cesta k souboru uživatelů. Více viz dále (např.: C:/temp/enisUsers.txt);
- db.path – cesta určující uložení databázového souboru H2 databáze. Tento parametr je platformě závislý, a proto musí být zpětná lomítka zdvojena, pokud aplikace běží pod operačním systémem Windows (vlnovka odkazuje na adresář uživatele, např.: ~\\OneDrive\\Skola\\DP\\database\\enis);
- formDefinition.inputDirectory – cesta k adresáři s definicemi formulářů (např.: C:/temp/enisInput);
- formDefinition.processedDirectory - cesta k adresáři se zpracovanými definicemi formulářů (např.: C:/temp/enisProcessed)
- filledForm.inputDirectory – cesta k adresáři pro zpracování vyplněných formulářů (např.: C:/temp/enisFilledInput)
- filledForm.processedDirectory – cesta k adresáři pro zpracované vyplněné formuláře (např.: C:/temp/enisFilledForms)

Pomocí výše uvedeného konfiguračního souboru lze definovat vnější rozhraní, tedy soubory a složky používané aplikací.

Správa uživatelů

Definovaný konfigurační parametr „security.usersFile“ odkazující na soubor s definicemi uživatelů je periodicky kontrolován aplikací každých 5 minut, pokud existuje, je načten seznam uživatelů včetně jejich otisku hesla. Pokud soubor neexistuje, jsou všichni uživatelé z interní databáze aplikace odstraněni. Načítání funguje jen na principu přidávání a úpravy stávajících, není možné selektivně mazat uživatele.

Soubor musí být ve formátu „<uživatelské jméno>:<otisk hesla se solí oddělený $ zakódovaný v Base64>, například:

test:oKMD/MzOfx3KZLIQIVOdtev/rlAnqkKRmsaCfxF7cAA=$c0qW6nB98WqPDr+81ZWGo/HLBd1o3scYdsqagcTug3U=

Ke správnému vygenerování řádky souboru byl vytvořen nástroj UserPasswordHashTool, umístěný v modulu user-api v balíčku cz.zcu.kiv.enis.util. Nástroj se spouští klasicky přes příkaz java (obsahuje metodu main). Po spuštění je uživatel vyzván k zadání uživatelského jména a následně hesla. Nástroj vygeneruje řádku s uživ. jménem a otiskem hesla se solí, který je možné umístit do specifikovaného souboru. Sůl a samotný otisk hesla jsou odděleny znakem $. Při následném ověřování hesla uživatele je při tvorbě otisku hesla použita právě tato sůl vztahující se přímo k tomuto uživateli.

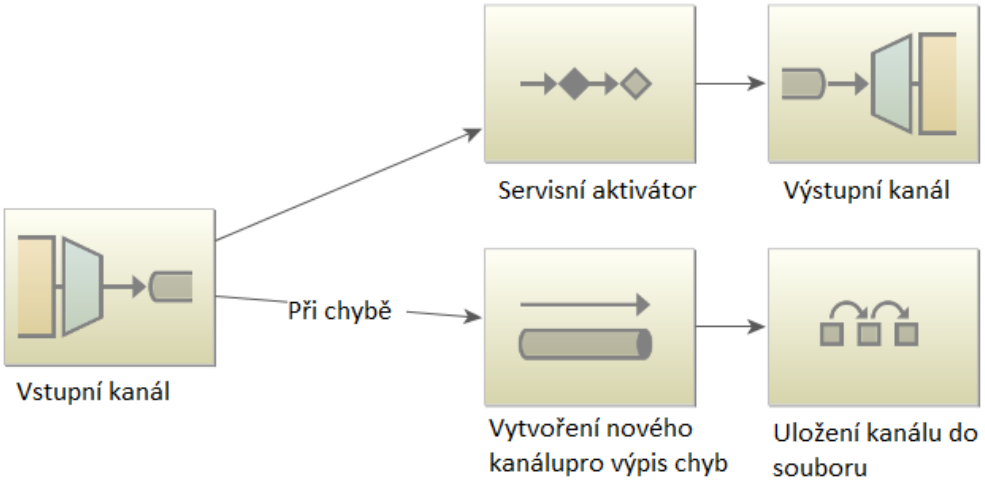
K vytvoření náhodné soli je použita třída SecureRandom za použití algoritmu SHA1PRNG (SHA1 pseudorandom generátor). Samotné vytvoření otisku hesla je provedeno pomocí algoritmu PBKDF2WithHmacSHA1, kde PBKDF2 je zkratka pro funkci pro vytvoření otisku na základ hesla. Hmac značí použití haš algoritmu za použití klíče a SHA1 je samotný algoritmus pro vytvoření otisku, vice informací je k dispozici ve zdroji (16).

Správa definic formulářů

Pro nahrání definic formulářů pro jednotlivé uživatele nebo zařízení je nutné nahrát XML popis formuláře do nakonfigurovaného adresáře (formDefinition.inputDirectory). Složka

je periodicky každých 5 vteřin monitorována a nové soubory zpracovávány. Aplikace si drží v paměti již zpracované jména souborů, proto nedojde k opětovnému zpracování.

Ke zpracování souborů se využívá Spring Integration. Na začátku je vytvořen vstupní kanál, do kterého jsou posílány nové soubory ke zpracování. Následně je volán servisní aktivátor, který zavolá definovanou metodu třídy se vstupním parametrem typu File (c.z.k.e.d.FormDefinitionLoaderImpl#handleFile). Metoda validuje zpracovávaný soubor oproti XSD souboru. V případě chyby vyhodí definovanou výjimku, která je zachycena knihovnou Spring Integration ve vstupním kanálu a předána novému kanálu určeného pro zpracování chyb. Tento nový kanál založí soubor se stejným jménem jako je aktuálně zpracovávaný soubor, ale s příponou .error, který obsahuje popis chyby tak, aby bylo snadné chyby identifikovat a případně opravit. Pokud validace oproti XSD souboru proběhla v pořádku, jsou vstupní data ve formátu XML převedeny do formátu JSON, který je datově méně náročnější a lépe vyhovuje pro mobilní platformy. Výsledný soubor je uložen do definované složky formDefinition.processedDirectory. Celý řetěz kanálů je zobrazen na následujícím obrázku.



Obrázek 8 Řetěz kanálů pro zpracování definice formuláře.

Příklad korektní i nekorektní definice formuláře s popisem chyby formuláře je k dispozici na přiloženém CD.

Pro určení jakému uživateli nebo zařízení je formulář určen slouží jeho název, který musí obsahovat uživatelské jméno nebo ID zařízení ohraničený podtržítkem z obou stran. Obě kritéria se mohou libovolně kombinovat, např.:

- form_testUsername_.xml – formulář je určen uživateli testUsername;
- form_user__11125dfdda_.xml – formulář určen jak uživateli user, tak i zařízení s ID 11125dfdda;
- _form__user_.xml – formulář určený pro uživatele form a user.

Tento jednoduchý přístup nevyžaduje žádné další požadavky na databázový model, či administraci, ale díky své jednoduchosti nemusí být ideální v případě, kdy jeden formulář bude využívat více uživatelů nebo zařízení – v takovém případě musí být soubor například duplikován, nebo obsahovat všechny potřebné informace přímo ve jménu (dlouhý název souboru).

Vyplněné formuláře

Posledním komunikačním rozhraním serverové části jsou složky pro vyplněné formuláře. Složka definovaná parametrem filledForm.inputDirectory slouží k nahrávání vyplněných formulářů obdobně jako při REST požadavku. Očekává zašifrovaný soubor, který vygeneruje klientská aplikace. Tato složka je také periodicky každých 5 vteřin monitorována.

Parametr filledForm.processedDirectory slouží k určení složky, kam se mají ukládat přijaté vyplněné formuláře. V této složce jsou již formulářová data nezašifrovaná a ve formátu RDF 1.1 N-Triples. Způsob komunikace, zabezpečení a formát dat bude popsán v samostatné kapitole 4.4 Komunikace mezi serverovou částí a aplikací.

4.1.3.4 Algoritmy

Následující kapitola popisuje některé klíčové algoritmy nebo datové toky v rámci serverové části.

Tvorba šifrovacích klíčů

O tvorbu klíčů používaných při šifrování dat se stará rozhraní CertificationManager a jeho implementace CertificationManagerImpl. Tvorba páru klíčů používaných při asymetrickém šifrování je závislá na definovaném algoritmu v konfiguračním souboru - výchozí algoritmus je RSA (více informací viz zdroj (17)). Při požadavku o získání veřejného nebo privátního klíče je nejprve zjištěno, jestli existuje soubor, jehož umístění je určeno konfigurací, představující serializovaný veřejný nebo privátní klíč. Pokud soubor neexistuje, je vygenerován nový pár klíčů a oba jsou uloženy pro pozdější použití.

Privátní klíč je uložen ve standardizovaném formátu PKCS#8 (více o tomto formátu viz (18)), veřejný klíč pak ve formátu X.509 (více viz zdroj (19)).

Dále tato třída umožňuje generování klíče pro použití při symetrickém šifrování. Pro generování klíče je použit algoritmus AES (Advance Encription Standard). Tento klíč není nikterak ukládán, je vždy použit pro každý požadavek nový.

Šifrování/dešifrování

Rozhraní CipherManager a jeho implementace CipherManagerImpl poskytují metody pro šifrování zpráv využívající veřejné klíče při šifrování a privátní klíče při dešifrování. Umožňují šifrovat a dešifrovat dlouhou zprávu, která svoji bitovou délkou přesahuje limit pro šifrování veřejným klíčem a tak musí být použita symetrická šifra a určený klíč.

Pro asymetrické šifrování je využitý výše uvedený algoritmus RSA spolu s módem ECB (Electronic Codebook) zejména z důvodu kompatibility s koncovými zařízeními. Pro doplňování bitů je použit standard PKCS#1 určený právě pro algoritmus RSA.

Symetrické šifrování používá algoritmus AES spolu s módem ECB a doplňováním bitů dle PKCS#5.

Následující tabulka poskytuje informace o možných šifrovacích algoritmech, módech a doplňovaní, které mohou být použity v serverové části (platforma Java). Informace vychází ze zdroje (20).

Jméno algoritmu	Mód	Doplňování (padding)
AES	ECB, CBC, PCBC, CTR, CTS, CFB, CFB8..CFB128, OFB, OFB8..OFB128	NOPADDING, PKCS5PADDING, ISO10126PADDING
AESWrap	ECB	NOPADDING
ARCFOUR	ECB	NOPADDING
Blowfish, DES, DESede, RC2	ECB, CBC, PCBC, CTR, CTS, CFB, CFB8..CFB64, OFB, OFB8..OFB64	NOPADDING, PKCS5PADDING, ISO10126PADDING
DESedeWrap	CBC	NOPADDING
PBEWithMD5AndDES, PBEWithMD5AndTriple	CBC	PKCS5Padding

DES[3]PBEWithSHA1AndDESede, PBEWithSHA1AndRC2_40		
RSA	ECB	NOPADDING, PKCS1PADDING, OAEPWITHMD5ANDMGF1PADDING, OAEPWITHSHA1ANDMGF1PADDING, OAEPWITHSHA-1ANDMGF1PADDING, OAEPWITHSHA-256ANDMGF1PADDING, OAEPWITHSHA-384ANDMGF1PADDING, OAEPWITHSHA-512ANDMGF1PADDING

Tabulka 2 Standardní Java provideři

Tvorba otisku hesla / haš

Při porovnávání přijatého hesla s přijatým je spočítán otisk přijatého za použití uložené soli, která je součástí uloženého hesla (část před znakem '$'). Spočítaný otisk hesla je ověřen oproti uloženému (část za znakem '$').

Elektronický podpis

Tvorba elektronického podpisu je prováděna za použití hašovacího algoritmu SHA256 a šifrovacího algoritmu RSA. Pomocí třídy java.security.Signature je vytvořena instance a následně je za použití privátního klíče volána samotná metoda pro vytvoření elektronického podpisu.

[3]PBEWithMD5AndTripleDES je proprietární algoritmus, který nebyl standardizován.

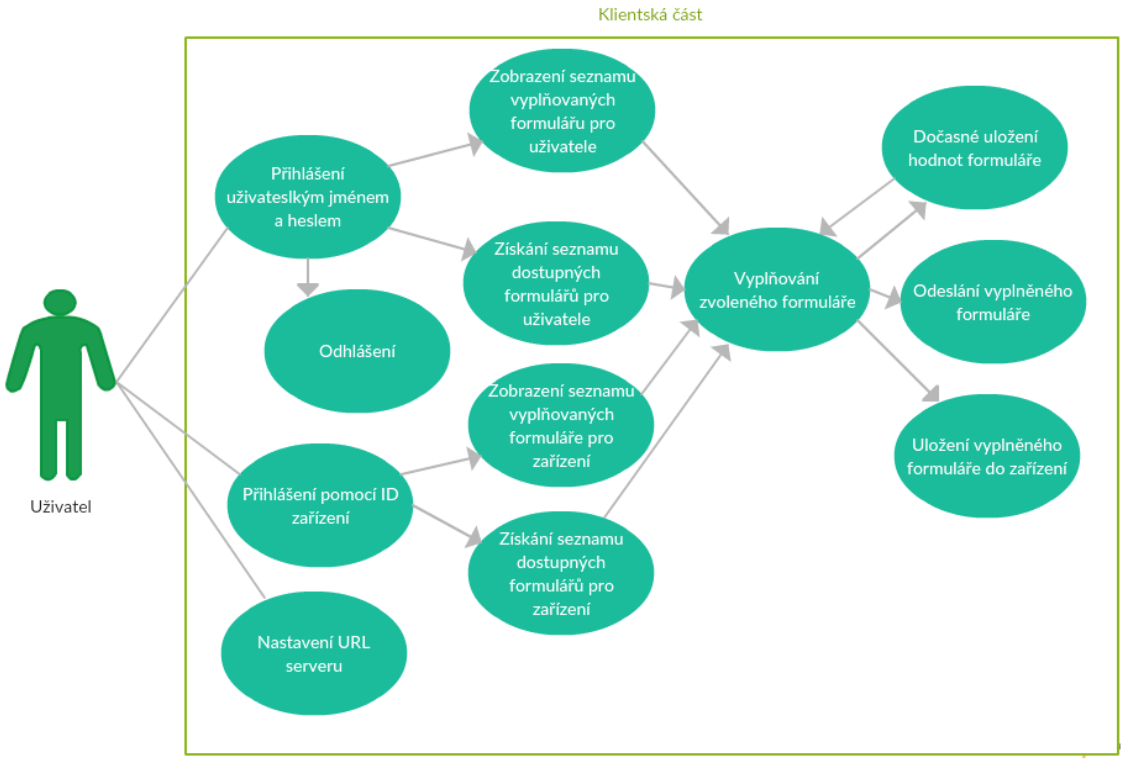
4.2 Klientská část (ENIS Klient)

Klientská část představuje aplikaci pro mobilní operační systém Android poskytující prostředky zejména pro vyplňování formulářových dat a jejich odesílání na serverovou část. Dále umožňuje přihlášení nadefinovaných uživatelů nebo načítání definic formulářů.

Zdrojové kódy jsou umístěné na přiloženém CD ve složce source_codes/ENIS_KLIENT. Soubor enis-klient.apk sloužící k instalaci na Android zařízení se nachází ve složce dist.

4.2.1Případy užití (use case) klientské aplikace

Následující obrázek představuje diagram případu užití klientské části. První nutnou funkčností je přihlášení uživatele pomocí definovaného jména a hesla. Serverová část musí ověřit správnost zadání. V případě zadání správné kombinace je uživateli nabídnut seznam aktuálně vyplňovaných formulářů. Druhá varianta použití je bez zadání uživatelského jména a hesla, jen za použití jedinečné identifikace zařízení. V takovém případě jsou zobrazeny vyplňované formuláře, které jsou dostupné jen pro dané zařízení. Při vyplňování formulářů musí být data ukládána do paměti zařízení tak, aby bylo možné formulář zavřít nebo opustit aplikace a při opětovném návratu byl opět načten s posledními zadanými daty. Poslední sada funkčností se týká již samotného odeslání vyplněných formulářových dat na server (viz ENIS), nebo uložit vyplněný formulář na externí paměť zařízení, odkud může být manuálně přenesen do určené složky přímo na serveru. Aplikace také umožňuje definovat adresu URL serveru pro veškerou komunikaci. Komunikace vně aplikace (odesílání, ukládání formulářů) je šifrovaná.



Obrázek 9 Use case diagram klientské části

4.2.2Architektura klientské aplikace

V této kapitole bude popsána architektura klientské části, tedy struktura projektu, základní uspořádání jednotlivých komponent a vazeb mezi nimi. Ve stručnosti budou popsány technologie, software development kit (SDK), frameworky, knihovny a další nástroje, které byly použité v rámci vývoje klientské části systému.

4.2.2.1 Použité technologie

Programovací jazyk / Android SDK

Vývoj pro platformu Android je silně spjat s programovacím jazykem Java. Platforma Android umožňuje vývoj nativních aplikací i pomocí jazyku C, ten je ale spíše využíván pro nízko-úrovňové volání systému apod. K vývoji aplikací poskytuje společnost Google, Inc. Android SDK (software develpment kit), jedná se o sadu nástrojů, frameworků, knihoven a další určených pro vývoj mobilních aplikací. Každá verze systému Android s sebou přináší i novou verzi SDK, tak aby bylo možné využít nově představené funkčnosti.

Do SDK verze N je možné pro vývoj používat jen Java do verze 1.7, až teprve vydaná ukázka Android N umožňuje využívat možnosti a novinek Javy 1.8. Tato verze však není určena pro vývoj nových aplikaci, díky nízkému podílu na trhu a nefinální podobě (21).

Pro využívání nových funkcionalit a současné zachování kompatibility se staršími verzemi OS Android vydává společnost Google, Inc. tzv. Support Library. Jedná se o sadu podpůrných knihoven umožňující právě využívání nové funkčnosti i ve starších verzích systému Android. V klientské části byly použity následující podpůrné knihovny:

- com.android.support:support-v4 – Základní podpůrná knihovna, přidává třídu Fragment (komponenta umožňující vytváření responsivního designu (tablet vs. telefon), umožňuje práci s novými notifikacemi apod.
- com.android.support:recyclerview-v7 – Knihovna umožňující využívat komponentu rycycler view, sloužící k zobrazení seznamu různých položek.
- com.android.support:appcompat-v7 – Knihovna přidávající podporu pro horní aplikační panel.
- com.android.support:design – Sada vizuálních prvků sloužící pro dodržování designových standardů definovaných společností Google, Inc.

Správa závislostí / tvorba instalačního souboru aplikace

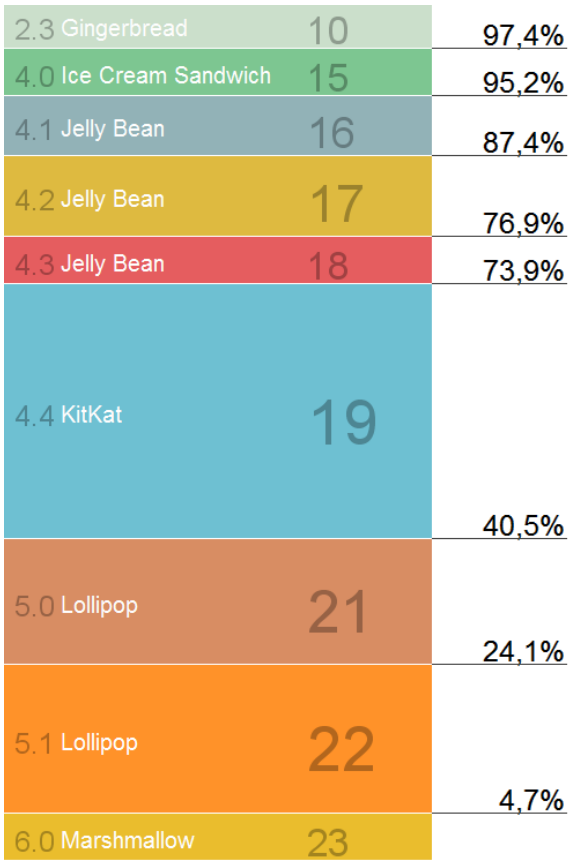
Vývoj pro platformu Android je silně spjat s nástrojem Gradle, který umožňuje provádět automatizaci častých úkonů. Jedná se o DSL (Domain Specific Language), jazyk specifikovaný na určitou doménu. Gradle je využíván zejména ke správě závislostí aplikace a její sestavení. Je často označován jako následník Apache Maven (22). V kombinaci s vývojem na platformu Android dokáže definovat verzi použitého SDK, verzi aplikace a mnoho dalšího. Gradle obsahuje nativní podporu pro Javu, Groovy, Scalu aj., umožňuje využívat i komunitní pluginy.

Při vývoji klientské aplikace byl použit plugin Android-apt, který dokáže při sestavování aplikace zpracovávat anotované třídy i jejich vlastnosti a pomocí definovaných úkolů vytvářet pomocný kód. Například za použití anotace @org.parceler.Parcel a použití

pluginu Android-apt a příslušné závislosti, dojde k vygenerování potřebného kódu pro užití funkčnosti parcelování[4].

Podporovaná zařízení

Podporovaná zařízení velmi záleží na zvoleném minimálním Android API (SDK). Každá nová verze přináší některé novinky nebo změny, proto je nutné určit minimální hodnotu verze API, tak aby byla zajištěna kompatibilita s novějšími verzemi. Následující obrázek zobrazuje procentuální podíl na trhu v kombinaci s verzí Android API a SDK. Při zvoleném minimálním API 14, Android OS verze 4.0 se procentuální pokrytí trhu blíží k 96%. Tento parametr je zároveň jediným omezujícím kritériem, velikost obrazovky nebo některé další HW nároky nejsou aplikovány.



Obrázek 10 Procentuální podíl verzí systému Android k 1. 6. 2016 (zdroj: vývojové prostředí Android Studio)

[4] Parcelování je systém serializace a deserializace objektů v systému Android. Jedná se o preferovaný způsob serializace než přes klasické Java rozhraní Serializable (31).

Databáze/ORM

Systém Android obsahuje databázi SQLite verze 3 a vyšší. Databáze umožňuje provádět všechny základní DDL(create, drop, alter apod.) příkazy i DML (select, insert, update apod.). Bohužel neumožňuje provádět žádné DCL (grant, revoke apod.) příkazy, protože neobsahuje prostředky na správu uživatelů a jejich oprávnění. Systém Android předpokládá ošetření práv na úrovni aplikace.

Pro zjednodušení práce s databází a využití ORM, které systém Android v jeho čisté formě neumožňuje, byla použita knihovna Android-Orma (viz zdroj (23)). Knihovna výrazně zjednodušuje práci s databázovou vrstvou. Komponenty frameworku sami dokáží vygenerovat potřebné pomocné třídy umožňující přímo pomocí vlastností objektu definovat výsledný SQL příkaz.

Např. kód pro získání seřazeného listu vyplňovaných formulářů, kde uživatelské jméno není vyplněno, by vypadal následovně:

```
OrmaDatabase.builder(<aktuální_context>)selectFromFormFilling().usernameIsNull().orderByLastModificationDesc().toList();
```

Oproti standardnímu přístupu systému Android, jak ukazuje například zdroj (24), kde jednoduché vytvoření a dotazování nad databází je na několik desítek řádek kódu, je tento zápis výrazně jednoduší a za pomocí anotací @Table a @Column u entit a jejich vlastností dokáže framework Android Orma vytvořit i zakládací DDL skripty. Za pomoci pluginu android-apt, představeného výše, dokáže již při sestavování aplikace vygenerovat potřebný kód.

Další knihovny a nástroje

Mezi nejvýznamnější použité knihovny se řadí QMBForm (25), která umožňuje generování formulářových prvků na základě definice. Díky MIT licenci bylo možné její fungování upravit pro použití v klientské části, zejména se jednalo o ukládání a načítání hodnot formulářů. Příklad použití knihovny pro vygenerování jednoduchého textového prvku by mohl vypadat následovně:

SectionDescriptor.newInstance("tag","Title".addRow(
RowDescriptor.newInstance("text",RowDescriptor.FormRowDescriptorTypeText, "Text",
new Value<String>("test")));

Dalším nástrojem, který byl použit při vývoji klientské části, je framework AndroidAnnotations, který umožňuje v kombinaci s pluginem android-apt generovat často se opakující kód při vývoji na platformě Android. Jedná se zejména o tvorbu a spouštění úloh na pozadí, načítání prvků uživatelského prostředí pro manipulaci v kódu nebo také dependency injekction (vkládání závislostí). Pro jednodušší práci s REST rozhraním slouží oddělená část androidAnnotations rest-spring-api.

Pro použití tlačítka pro nabízené akce byla použita knihovna clans FloatingActionButton, která dokáže vykreslit nejen samotné tlačítko, ale i například menu. Příklad je uveden na následujícím obrázku.

Obrázek 11 Možnosti zobrazení FAB (tlačítko pro nabízení akce). Zdroj https://github.com/Clans/FloatingActionButton/blob/master/screenshots/menu_default_opened.png

4.2.2.2 Členění projektu

Projekt klientské části je členěn dle konvence PBL (Package By Layer), kde jsou seskupeny třídy a rozhraní na stejné úrovni (například všechny aktivity jsou v jednom balíčku apod.). Základní balíček tvoří „cz.zcu.kiv.enis.android.client“, který se dále větví na balíčky:

- activity – obsahuje všechny třídy aktivit, jednotlivé obrazovky aplikace;
- data – obsahuje pod-balíčky api a impl, pro definování rozhraní a jeho implementaci. Je zde definováno rozhraní pro práci se specifikacemi formulářů a jejich hodnot. Zároveň jsou zde umístěny entity FormFilling a FomSpecification reprezentující uložená data v SQLite databázi. Obě tyto entity implementují

rozhraní Form tak, aby bylo možné s nimi pracovat nezávisle na implementaci. V tomto balíčku se také nachází DTO třídy pro reprezentaci specifikace formuláře. Specifikace formuláře je přijata jako JSON objekt, který je pomocí knihovny fasterxml.jackson převeden na Java objekty, s kterými je následně dál pracováno;

- fragment – balíček obsahující všechny fragmenty aplikace (část uživatelského rozhraní);
- network – obdobně jako balíček data poskytuje tento opět pod-balíčky api a impl. Api slouží k definici rozhraní a jednotlivých datových tříd (DTO), používaných při REST požadavcích. Obsahuje i rozhraní ClientRestService, o jehož implementaci se stará knihovna androidannotations rest-spring-api. Opět je zde využito pluginu android-apt a pomocí anotací definovaná jednotlivá REST rozhraní;
- security – obsahuje podbalíčky api a impl. Slouží k definici a implementaci metod pro šifrování, tvorbu otisku nebo elektronického podpisu.
- view – balíček pro jednotlivé elementy uživatelského rozhraní, jako jsou různé adaptéry a seznamy apod. Obsahuje adaptér pro zobrazení listu vyplňovaných a nebo dostupných formulářů.

V kořenovém balíčku se nachází třída BaseApplication, která tvoří vstupní bod aplikace. Definuje závislosti, některé základní parametry apod.

Všechny další zdroje aplikace jsou umístěny ve složce res, kde jsou dále členěny do dalších podsložek podle jejich typu a určení. Platforma Android umožňuje definovat různé zdroje pro různé platformy, zobrazení, jazyky a další. K tomuto účelu slouží speciální složky se sufixem obsahující kritéria, např.: zdroje ve složce drawable-v21 budou načteny jen na zařízení mající verzi systému Android s API verze 21 a vyšší, apod.

Členění souborů ve složce res odpovídá standardům platformy Android, veškeré použité prostředky jsou z volně šiřitelných zdrojů, nebo přímo z Android SDK.

4.2.3 Implementace klientské části

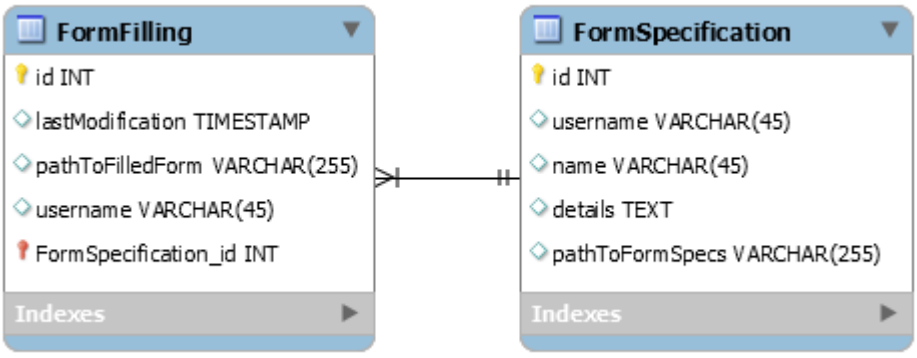
Tato kapitola popisuje samotnou implementaci a způsob fungování klientské části – Android aplikace. Navržené funkčnosti vycházejí z výše uvedených případů užití. Na zdrojovém CD ve složce ‘diagrams‘ se nachází také soubor ‘android-client-class-diagram.png‘ představující diagram tříd všech vytvořených tříd a rozhraní mobilní aplikace.

4.2.3.1 Databázový model

Následující obrázek představuje datový model klientské části. Aplikace musí držet informace o vyplňovaných formulářích a jejich definic, respektive jejich metadata.

První tabulkou je FormSpecification s číselným, automaticky generovaným primárním klíčem. Obsahuje atributy username textového typu, sloužící k identifikaci, jaký uživatel právě tuto definici stáhl/získal ze serverové části. Atribut name určuje jméno formuláře, jedná se o položku přímo z definice formuláře v jazyku, který je nastaven na zařízení. Atribut details je opět přímo z definice formuláře a stejně jako předchozí atribut je uložena jen ta část z definice, která odpovídá jazyku nastavenému na zařízení. Poslední atribut je pathToFormSpecs, který určuje absolutní cestu k samotnému souboru obsahující definici formuláře.

Druhou tabulkou, mající vazbu 1:N, je FormFilling. Jedna formulářová specifikace (záznam z tabulky FormSpecification) může mít více vyplňovaných formulářů (záznamy v tabulce FormFilling), ale jeden vyplňovaný formulář může vycházet jen z jedné definice. Tabulka FormFilling obsahuje časové razítko určující poslední modifikaci vyplňovaných dat. Dále je zde opět atribut username, který slouží k uložení informace, jaký uživatel tento formulář vyplňuje. Nutnost vlastního atributu pro identifikaci uživatele je z důvodu, kdy atribut username může být prázdný, pokud byla definice formuláře stažena pro zařízení, ale formulář začne vyplňovat přihlášený uživatel. Posledním atributem je pathToFilledForm, který obsahuje absolutní cestu k souboru obsahující zadaná data formuláře. Jedná se o binární soubor uložený v interní paměti aplikace, ke kterému nemá přístup žádný další uživatel ani proces systému, práva jsou nastavena na úrovni systému.



Tabulka 3 ERA model klientské části

Data obou tabulek jsou uloženy (persistovány) v nativní android SQLite databázi za použití ORM frameworku Android-Orma, starající se o objektově relační mapování a vytváření tzv. pomocných tříd umožňující jednoduší práci s entitami.

4.2.3.2 Uživatelské rozhraní

Uživatelské rozhraní bylo navrženo tak, aby v mnoha ohledech splňovalo designovou příručku společnosti Google, Inc. Aplikace využívá definovanou paletu barev, používá doporučené odsazení a rozložení, apod. Více o doporučeném designu je k dispozici ve zdroji (26). Všechny uživatelské prvky jsou dvou jazyčné – čeština a angličtina. Volba jazyka záleží na systémovém nastavení v zařízení. Výchozím jazykem je angličtina, to znamená, pokud je na zařízení nastaven jiný jazyk než čeština, jsou zobrazeny anglické popisky.

První obrazovka obsahuje vstupní pole pro jméno a heslo a dvě tlačítka pro přihlášení, nebo pro práci bez přihlášení. První tlačítko slouží k odeslání vyplněných informací na definovaný server, který je ověří a vrátí odpověď. Druhé tlačítko slouží pro použití v módu, kdy jsou formuláře specifikovány pro dané zařízení. Obrazovka je zobrazena na obrázku: Obrázek 12 část A.

Po úspěšném přihlášení je zobrazen seznam vyplňovaných formulářů, který je při prvním použití prázdný. Spolu se jménem a popisem formuláře v aktuálním nastaveném jazyce je zobrazena informace o tom, kdy byl formulář naposledy upraven, viz Obrázek 12 část B.

V levém horním rohu je zobrazeno tlačítko na zobrazení bočního menu, které lze vysunout i tažením prstu z levého okraje. Zde je zobrazeno uživatelské jméno nebo text

„Nepřihlášený uživatel“, pokud je aplikace využívána v módu zařízení[5]. Dále je zde hlavní menu pro navigaci (Obrázek 12 část C).

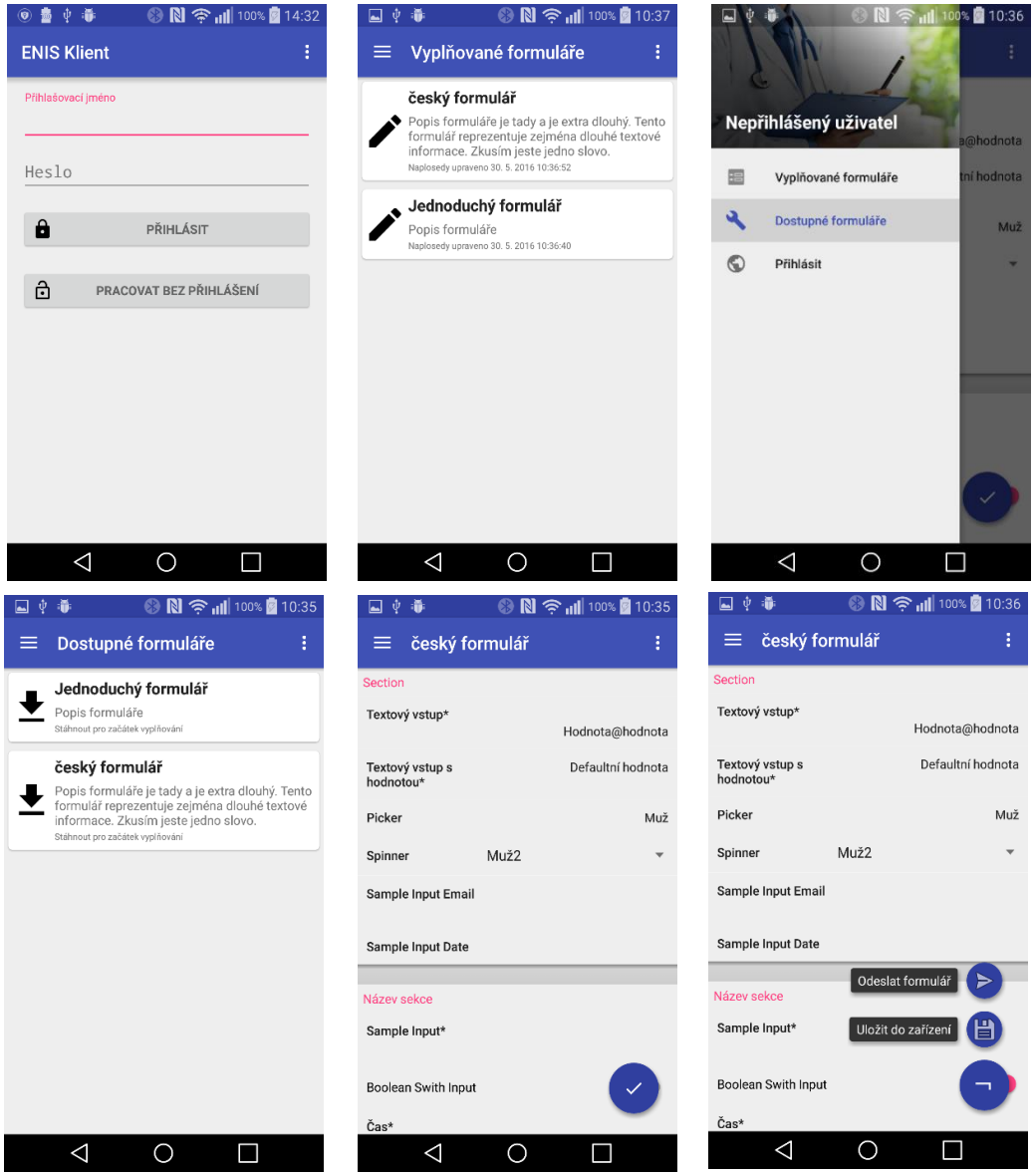
Další obrazovkou je seznam dostupných formulářů, kde je zobrazen nejprve seznam uložených definic formulářů přímo z databáze aplikace, pokud je tento seznam prázdný, dojde k načtení definic ze serverové části. Uživatel může vyvolat načtení tažením od horního okraje seznamu. Pro lepší uživatelskou interakci je zobrazena ikona stažení oproti ikoně úpravy, jak tomu bylo na obrazovky vyplňovaných formulářů (Obrázek 12 část D).

Ať už z obrazovky s dostupnými nebo vyplňovanými formuláři, je uživatel po kliku na příslušnou definici přesměrován na editaci vybraného formuláře. Zobrazené elementy uživatelského rozhraní vychází z definice formuláře (více viz kapitola 4.3). V pravém dolním rohu je zobrazeno tlačítko pro nabízené akce, které je v případě potřeby schováváno (například: při posunu seznamu dolu). Po stisku tohoto tlačítka se zobrazí nabídka možných akcí – odeslání formuláře online, nebo jeho uložení do paměti zařízení (viz Obrázek 12 část E a F).

Aplikace se skládá celkem ze tří aktivit (ucelený celek představující kontejner pro elementy obrazovky) a tří fragmentů. První aktivitu tvoří přihlašovací obrazovka, která je ukončena při přechodu na další. Další aktivita reprezentuje obrazovku s nastavením aplikace, kde je možné změnit adresu serverové části. Poslední aktivita představuje hlavní obrazovku s bočním navigačním panelem. Tato aktivita umožňuje měnit hlavní zobrazovací plochu pomocí definovaných fragmentů. Každý fragment představuje jednu položku z menu a jejich použité názvy v kódu aplikace jsou následující:

- FillingFormsFragment – seznam vyplňovaných definic formulářů;
- FillingFormDetailFragment – zobrazení formuláře a vyplňování hodnot;
- AvailableFormFragment – seznam dostupných definic formulářů.

[5] Obrázek použitý pro horní část pochází ze serveru pexels.com s licenci CC0 (Creative Commons), proto může být dále volně šířen, upravován a používán i pro komerční účely.



Obrázek 12 Uživatelské rozhraní aplikace, označeno od levého horního následovně: A, B, C, D, E, F.

4.2.3.3 Algoritmy

Použité algoritmy pro šifrování, dešifrování, tvorbu a ověření elektronického podpisu se shodují s výše uvedenými v kapitole 4.1.3.4 (mírně upravené z důvodu rozdílného logování a konverzí BASE64). Jsou umístěné ve třídě SecurityManagerImpl implementující rozhraní SecurityManager.

4.3 Tvorba formulářů

Definici formuláře popisuje XML soubor s definovanými elementy a atributy, tak aby bylo možné na základě definice formuláře a vyplněných dat sestavit soubor ve formátu RDF 1.1 N-Triples (viz zdroj (27)).

Tento formát se skládá z trojice údajů podmět, vlastnost, předmět nebo také subjekt, predikát a objekt (subject, predicate, object), kde jednotlivé elementy jsou reprezentovány daným URI, například řádka:

<http://form> <http://row> "value"^^<http://www.w3.org/2001/XMLSchema#string> .

představuje v tomto případě formulář s definovaným URI http://form, položku s URI http://row a hodnota je „value“ typu string.

Takto vytvořené trojice hodnot jsou odděleny vždy znakem pro novou řádku, na které může být další libovolná trojice hodnot. Tento formát je použit pro ukládání výsledných souborů obsahující vyplněné údaje, soubory mají příponu nt.

Pro umožnění validace formuláře byl vytvořen soubor XSD (obsažen na přiloženém CD ve zdrojových souborech serverové části a modulu data-api). XSD soubor definuje možné elementy a jejich atributy včetně pořadí. Je možné ho přilinkovat přímo ke zdrojovému XML souboru a za pomoci některého z dostupných editorů získat napovídání v reálném čase apod. Přilinkování do XML souboru se provede následujícími atributy u kořenového elementu:

- *xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"*
- *xsi:noNamespaceSchemaLocation="<cesta>/formSchema.xsd"*

Hlavním, kořenovým elementem je formDescriptor, který musí definovat zdrojové URI formuláře (atribut uri), to je použito jako podmět (první hodnota řádku v souboru nt).

Další elementy tvoří uživatelský popis formuláře, který je zobrazen před samotným vyplňováním formuláře, jako první je formName s atributem lang, který obsahuje jméno formuláře. Druhým je formDescription pro popis formuláře, opět se stejným atributem.

Atribut lang musí obsahovat validní kód jazyka podle normy ISO 639-1, seznam validních kódů je dostupný například ve zdroji (28). Elementy s atributem lang se mohou opakovat, ale vždy by měli definovat jiný jazyk tak, aby mohl být vybrán klientskou aplikací ten vhodný, podle nastavení zařízení. Elementy s atributem lang budou dále uváděny jen jako multi-jazyčné.

Dalším elementem je section, který se může opakovat a definovat tím nové sekce formuláře. Každá sekce obsahuje multi-jazyčný element sectionTitle s názvem sekce, který nemusí být použit. V takovém případě bude v mobilní aplikaci vykreslena sekce bez titulku. Každá sekce může obsahovat libovolný počet formulářových řádek – element row, který musí definovat atribut uri. Hodnota tohoto atributu je následně použita pro druhý údaj na řádce v soubor nt (vlastnost). Řádek má definovaný multi-jazyčný popisek elementem title, datový typ (element type). Možné typy formulářových prvků jsou uvedeny v následující podkapitole. Řádek musí být unikátně identifikován, aby bylo možné s ním v klientské aplikaci pracovat (ukládat/načítat hodnoty apod.), k tomuto účelu slouží element uniqueName. Každý řádek může mít definovánu výchozí hodnotu, k tomuto účelu slouží multi-jazyčný element value.

Pro definování možných hodnot, kterých může formulářový prvek nabývat a tím omezit obor hodnot, slouží element values a vnořené elementy value. Tento výčet je brán v úvahu jen, pokud je zvolen jako typ selectorSpinner, nebo selectorSegmentedControl.

Ke každé řádce formuláře je také možné definovat požadované validace pomocí elementu validations a validationType. Při zvolení validationType „NOT_EMPTY“, bude brána řádka formuláře jako povinně vyplnitelná, k popisku bude přidán znak hvězdičky (*) a nebude povolena prázdná hodnota. Další možnou validací je „EMAIL“, která umožní odeslat/uložit jen validní emailovou adresu.

Příklad definice formuláře je dostupný ve zdrojových souborech serverové části modulu data-api ve složce resources.

4.3.1Možné typy formulářových prvků

V této kapitole bylo vycházeno zejména ze zdroje (25). Následující tabulka určuje možné typy formulářových prvků včetně jejich popisu a způsobu vykreslení. Většina formulářových prvků má dvě varianty vykreslení:

- Standardní – vykreslení popisku a samotného prvku zvlášť na dvou řádkách;
- Inline – vykreslení popisku na stejné řádce jako formulářový prvek. Kód položky obsahuje sufix „Inlline“.

Kód položky	Popis položky, vykreslovaný element
text	Vstup pro standardní text.
detail	Vstup pro víceřádkový text.
textView	Vykreslení textu bez možnosti editace.
url	Vstup pro text představující adresu URL. Softwarová klávesnice je přizpůsobena zadávání URL, není zapnuta predikce ani automatické mezery apod.
email	Vstup pro text představující emailovou adresu. Klávesnice může obsahovat například znak @, .com a podobně.
password	Zadávání hesla. Jednotlivé znaky jsou pro uživatele skryty.
number	Standardní vstup pro číslo. Klávesnice a samotný element umožňují pohodlnější zadání čísla.
currency	Vstup umožňující zadání čísla představující měnu.
phone	Vstup pro zadávání telefonního čísla.
integer	Celočíselný vstup.
selectorSpinner	Rozevírací seznam umožňující vybrání jedné z definovaných možností.
date	Dialog pro zadání data. Používá se standardní dialog systému Android.
time	Dialog pro zadání času. Používá se standardní dialog systému Android.
booleanCheck	Element představující klasické zaškrtávací tlačítko. Booleanovská hodnota pravda/nepravda.
booleanSwitch	Vstup umožňující pomocí přepínače zvolit hodnotu pravda/nepravda.
selectorSegmentedControl	Element vykreslovaný do horizontálního seznamu položek, kde uživatel může zvolit jeden z nabízených vstupů.

Tabulka 4 Možné formulářové prvky

4.4 Komunikace mezi serverovou částí a aplikací

Následující kapitola a její podkapitoly popisují samotnou komunikaci mezi serverovou a klientskou částí. Je popsán způsob a formát dat včetně použitých rozhraní.

4.4.1 Přihlášení

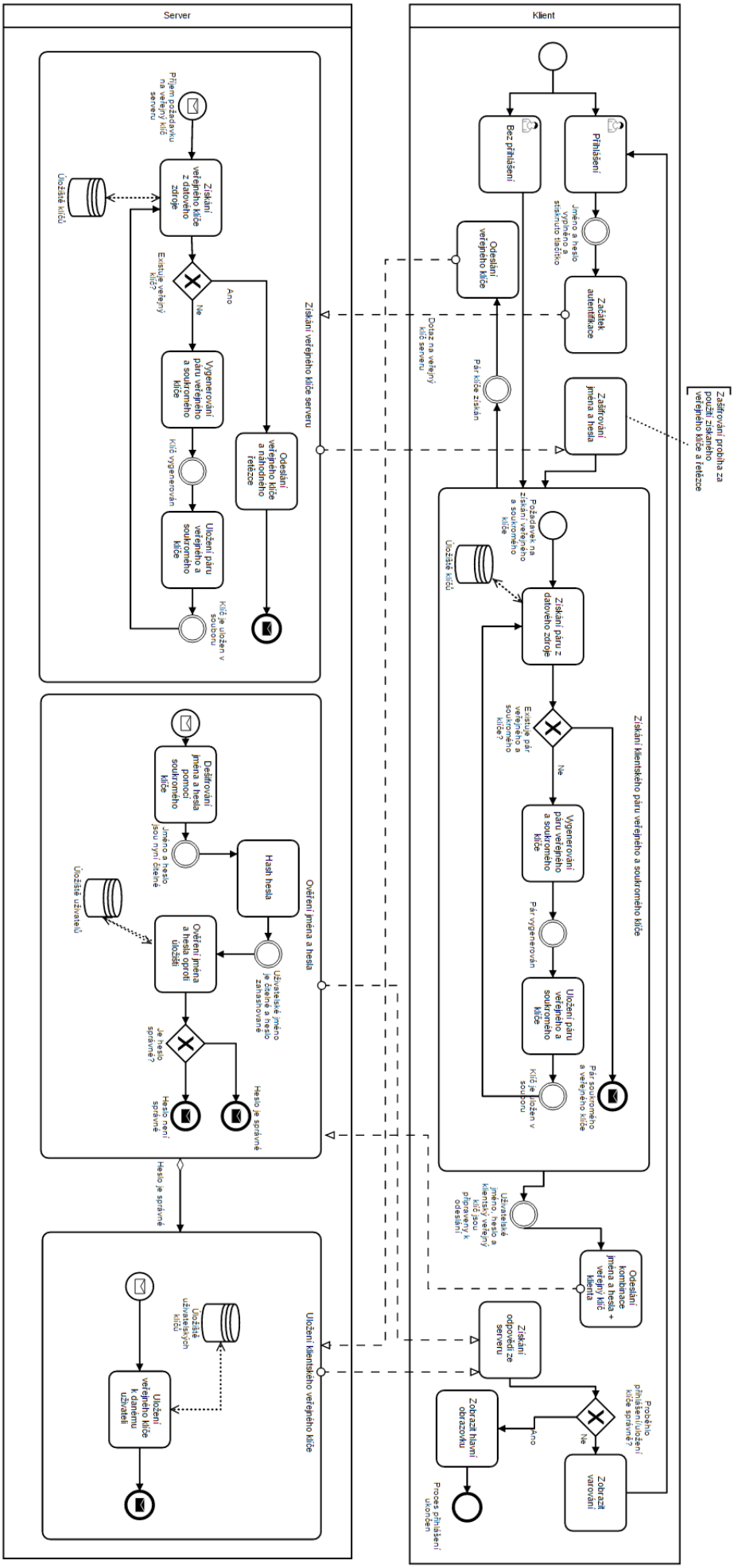
Obrázek 13 popisuje proces přihlášení pomocí BPMN 2.0 diagramu (jeho vektorová verze pro lepší čitelnost je umístěna na přiloženém CD ve složce diagrams soubor login-diagram.svg). Proces začíná spuštěním mobilní aplikace, kde uživatel může zvolit dva různé způsoby použití aplikace.

První možnost je přihlášení bez uživatelského jména a hesla v módu, kdy se definice formulářů získávají podle unikátního identifikátoru zařízení. Nejprve je získán pár soukromého a veřejného klíče pro zařízení. Pokud pár klíčů neexistuje, je vygenerován nový a uložen do souboru v interní paměti aplikace. Následně je odeslán veřejný klíč spolu s identifikací zařízení na serverovou část. Pokud je server nedostupný, nebo nastane jiná chyba, je zobrazeno informační okno o chybě a uživatel zůstává na přihlašovací obrazovce (aktivitě). V opačném případě, kdy server je dostupný a úspěšně zpracuje požadavek – uloží k danému zařízení získaný veřejný klíč, je uživatel přesměrován na hlavní obrazovku (aktivitu), kde je zobrazen seznam aktuálně vyplňovaných formulářů.

Druhá možnost přihlášení je pomocí uživatelského jména a hesla a použití aplikace v módu, kdy se definice formulářů vážou na přihlášeného uživatele. Nejprve je nutné získat veřejný klíč serverové části tak, aby bylo možné zašifrovat odesílané údaje. Aplikace tedy odešle požadavek na získání veřejného klíče, serverová část buď okamžitě odpoví zprávou obsahující daný klíč načtený ze souboru, nebo jej nejprve vygeneruje specifikovaným algoritmem, uloží do souboru a vrátí. Spolu s veřejným klíčem se odesílá i náhodný řetězec představující kryptografickou sůl pro použití při šifrování. Přijaté hodnoty jsou uloženy v klientské části do interní paměti. Dále klientská aplikace zašifruje zadané uživatelské jméno a heslo pomocí veřejného klíče a získané soli, která je přidána k zadaným datům. Důvod použití soli bude diskutován v kapitole 4.4.4. Po úspěšném zašifrování uživatelského jména a hesla je získán pár veřejného a soukromého

klíče pro daného uživatele[6]. Tato trojice hodnot je následně odeslána na serverovou část. Serverová část provede dešifrování získaného uživatelského jména a hesla pomocí vlastního soukromého klíče. V tuto chvíli jsou data čitelná a je možné porovnat, jestli se shoduje část obsahující kryptografickou sůl se solí odeslanou danému zařízení (kryptografická sůl se váže na unikátní identifikaci zařízení). V případě neúspěchu je požadavek zamítnut. Pokud se část shoduje, jsou načteny údaje týkající se uživatelského jména z databáze serverové části. Pokud takový uživatel neexistuje, je opět požadavek vrácen s chybovým stavem. V opačném případě je pomocí hašovací soli, získané z databáze pro daného uživatele, proveden otisk zadaného hesla a porovnán s uloženým. Pokud se otisky hesel shodují, je uložený pro daného uživatele přijatý veřejný klíč a odpoví se kladnou zprávou. Klientská aplikace podle přijaté odpovědi zobrazí hlavní obrazovku se seznamem vyplňovaných formulářů, nebo informační okno o chybě při přihlášení.

[6] Pár klíčů je závislý na uživatelském jménu a id zařízení. Pro každého uživatele a zároveň pro každé id zařízení je generován zvláštní pár klíčů, aby nedošlo ke znovu použití.



Obrázek 13 Proces přihlášení namodelovaný pomocí BPMN 2.0

4.4.2 Příjem/odesílání definic formulářů

Při přístupu na obrazovku s dostupnými formuláři je nejprve načten lokální seznam z databáze klientského zařízení. Pokud seznam obsahuje hodnoty, není proveden žádný požadavek na serverovou část a klientská aplikace nabídne jen lokálně uložené definice formulářů z důvodu urychlení práce a zlepšení uživatelského dojmu.

V opačném případě, kdy seznam je prázdný, nebo je vyžádáno znovu načtení definic formulářů pro daného uživatele/zařízení manuálně, je nejprve uskutečněn požadavek na veřejný klíč serverové části, čímž se současně znovu načte a přepíše kryptografická sůl pro dané zařízení.

Pokud je aplikace používána v módu bez přihlášení, získá se privátní (soukromý) klíč pro dané zařízení z interní paměti aplikace a odešle se požadavek na server spolu s identifikací zařízení. Serverová část v tomto případě načítá definice formulářů pro id zařízení.

V druhém případě, kdy je aplikace používána se zadaným uživatelským jménem a heslem, je jméno a heslo zašifrováno spolu s nově přijatou kryptografickou solí pomocí veřejného klíče serverové části. Následně je proveden požadavek, který obsahuje právě zašifrované uživatelské jméno a heslo a nezašifrované id zařízení. Serverová část načítá definice formulářů pro zadané uživatelské jméno.

Jakmile serverová část přečte dané definice formulářů ze souborů ve formátu JSON, načte se veřejné klíč pro dané zařízení (respektive uživatele), který byl získán v procesu přihlašování (viz předchozí kapitola). Následně pro všechny nalezené definice formulářů se vygeneruje tajný klíč pro symetrické šifrování, který se použije pro její zašifrování. Tento tajný klíč se taktéž zašifruje, ale za použití získaného veřejného klíče zařízení, nebo uživatele. Posledním krokem je elektronické podepsání zašifrované definice formuláře, které se provede za použití hašovacího algoritmu a privátního klíče serverové části. Celý seznam obsahující:

- zašifrované definice formulářů pomocí tajného klíče;
- zašifrovaný tajný klíč pomocí veřejného klíče uživatele/zařízení;
- elektronický podpis vytvořený ze zašifrované definice pomocí privátního klíče serverové části;

je následně odeslán ve formátu JSON jako odpověď.

Klientská část odpověď zpracuje a převede ji do Java objektů pro lepší manipulaci. Pro každou položku ze seznamu je nejprve ověřen elektronický podpis pomocí veřejného klíče serverové části. V případě úspěchu je dešifrován tajný klíč pro symetrické šifrování pomocí privátního klíče zařízení nebo uživatele. Získaný dešifrovaný tajný klíč je následně použit pro dešifrování definic formulářů, čímž je opět získán text ve formátu JSON, který je převeden opět do Java objektů reprezentující strukturu definice formuláře.

Získané objekty jsou uloženy do datových souborů v zařízení a jsou vytvořeny patřičné záznamy v interní databázi (tabulka FormSpecification). Současně jsou vybrány jen ty multi-jazyčné záznamy, které odpovídají aktuálně nastavené lokalizaci zařízení. Pokud tento jazyk není obsažen v seznamu multi-jazyčných položek, je vybrána první uvedená hodnota.

4.4.3 Příjem/odesílání vyplněných formulářů

Po úspěšném vyplnění formulářových hodnot – všechny validační podmínky jsou splněny, je možné odeslat, nebo uložit vyplněný formulář. Klientská aplikace převede vyplněné hodnoty do formátu RDF 1.1 N-Triples dle následujícího postupu:

1. Získání URI vyplňovaného formuláře (atribut uri elementu formDescriptor v xml definici formuláře).
2. Získání URI pro každý řádek formuláře (atribut uri elementu row).
3. Získání zadané, nebo zvolené hodnoty.
    a. Pokud se jedná o zvolenou hodnotu ze seznamu možných hodnot, je jako hodnota považována uri uvedená u zvolené hodnoty.
    b. Pokud uživatel zadal hodnotu vyplněním, nebo zvolením data a času[7], je jako hodnota považován přímo zadaný vstup, ke kterému je doplněno URI identifikující datový typ[8].

[7] Datum je převedeno do formátu ISO 8601.
[8] Datový typ je uvozen URI http://www.w3.org/2001/XMLSchema#, za kterou je uveden název datového typu (například string, float, date apod.).

Výsledná řádka ve formátu RDF 1.1 N-Triples je pak složena následovně (konec řádky je uvozen tečkou):

<uri_formuláře> <uri_řádky> hodnota_s_datovým_typem_nebo_zvolené_uri .

Příklady vyplněného formuláře ve formátu RDF 1.1 N-Triples:

```
<http://formular1> <http://radka1> "hodnota"^^<http://www.w3.org/2001/XMLSchema#string> .
<http://formular1> <http://radka2> <http://zvolena_ciselnikova_hodnota> .
<http://formular1> <http://radka3> "2016-05-
14T14:17:21"^^<http://www.w3.org/2001/XMLSchema#date> .
```

Takto vytvořená struktura je následně zašifrována pomocí vygenerovaného tajného klíče pro symetrické šifrování. Šifrovaná data jsou elektronicky podepsána pomocí privátního klíče uživatele, nebo zařízení. Tajný klíč použitý pro šifrování formulářových dat je zašifrován pomocí veřejného klíče serverové části. Posledním krokem je odeslání nebo uložení této struktury obsahující zašifrovaná data, tajný klíč a elektronický podpis. Pokud je zvoleno odeslání, je proveden požadavek na serverovou část, která příchozí data začne ihned zpracovávat. V opačném případě je datový soubor uložen do zařízení do adresáře form-data v systémové složce „stažené soubory“ (downloads). Při uložení do souboru je ke zvolenému jménu přidána informace o jménu uživatele nebo id zařízení. Tak, aby bylo později možné určit správný původ dat a tím i veřejný klíč pro ověření elektronického podpisu.

Při zpracovávání vyplněných dat přijatých ať už prostřednictvím REST rozhraní, nebo nahráním do specifikovaného adresáře, je ověřen elektronický podpis pomocí veřejného klíče uživatele, nebo zařízení získaného při přihlášení. Následně se dešifruje klíč pro dešifrování přijatých dat pomocí privátního klíče serverové části. Posledním krokem je již samotné dešifrování příchozích dat pomocí dešifrovaného klíče. Získaná data jsou uložena do souboru se jménem obsahujícím uživatelské jméno, nebo id zařízení, které data odeslalo, a časovým razítkem ve specifikovaném adresáři. Koncovka souboru je .nt. Pokud při verifikaci dat nebo dešifrování nastala chyba je uživatel informován přímo v aplikaci zobrazením informačního okna, nebo pomocí souboru s koncovkou .error při nahrávání souboru do sledované složky.

Aplikace dále nezpracovává přijaté hodnoty, výsledkem jsou uložené soubory ve formátu RDF 1.1 N-Triples ve specifikované složce.

4.4.4 Možné útoky a ochrana před nimi

Následující kapitola se věnuje základním technikám útoků na informace a způsobu ochrany. Pozornost bude věnována základním typům útoků a bude předpokládáno, že zvolené kryptografické algoritmy jsou bezpečné (neexistuje způsob jejich prolomení ve výpočetně akceptovatelném čase). Zároveň nebudou zohledněny eventuální slabá hesla uživatelů nebo malá zvolená délka kryptografických klíčů.

4.4.4.1 Odposlouchávání dat

Základním typem útoku je pasivní odposlouchávání dat. Útočník má přístup k veškeré síťové komunikaci a může nahlížet do odesílaných/přijímaných zpráv. Toto je možné například při připojení zařízení do nezabezpečené Wi-Fi sítě, kde bezdrátový směrovač zachytává veškerou komunikaci.

Systém (serverová i klientská část) je navržen tak, aby všechny citlivé informace (včetně definic formulářů, které mohou obsahovat výchozí hodnoty) byly šifrované, tudíž pro útočníka neznající soukromý klíč nečitelné. Jedinými nešifrovanými posílanými údaji jsou veřejné klíče serverové i klientské části a id zařízení, které útočník nemůže použít k dešifrování komunikace díky použití asymetrické kryptografie.

4.4.4.2 Modifikace dat

Dalším typem útoku je modifikace požadavku za účelem získání dat, na které uživatel nemá právo. Předpokládá se, že útočník má přístup ke komunikaci mezi klientskou a serverovou částí, kterou může i modifikovat. Opět to může být způsobeno připojení zařízení do nezabezpečené Wi-Fi sítě, či nainstalováním nevhodného software na zařízení.

Útočník může například při požadavku na definice formuláře změnit odesílané id zařízení, a tím si vyžádat definice určené jinému. Serverová část načte definice pro podvrhnuté zařízení, ale ty jsou před odesláním šifrovány pomocí generovaného klíče, který je šifrován veřejným klíčem pro přijaté id zařízení. Útočník tedy bez znalosti privátního klíče pro dané id zařízení není schopen dešifrovat klíč pro dešifrování definic.

Dalším příkladem modifikace dat je úplné nahrazení vyplňovaných dat, nebo definic formulářů. Útočník může nahradit obsah JSON požadavku kompletně novými hodnotami. Pokud bude dodržena struktura a formát zprávy, dojde k jejímu zpracování,

ale při pokusu o ověření elektronického podpisu pomocí veřejného klíče odesílatele dojde k chybě, protože podpis nebude korespondovat s obsahem požadavku, nebo nebude souhlasit použití páru soukromého a veřejného klíče.

4.4.4.3 Krádež identity

Dalším možným útokem je využití odesílané zprávy při autentifikaci pro pozdější využití a simulování tak odeslání přihlašovacích údajů. Útočník by mohl odposlechnout data posílaná během přihlašování a následně je odeslat na server s podvrhnutým veřejným klíčem.

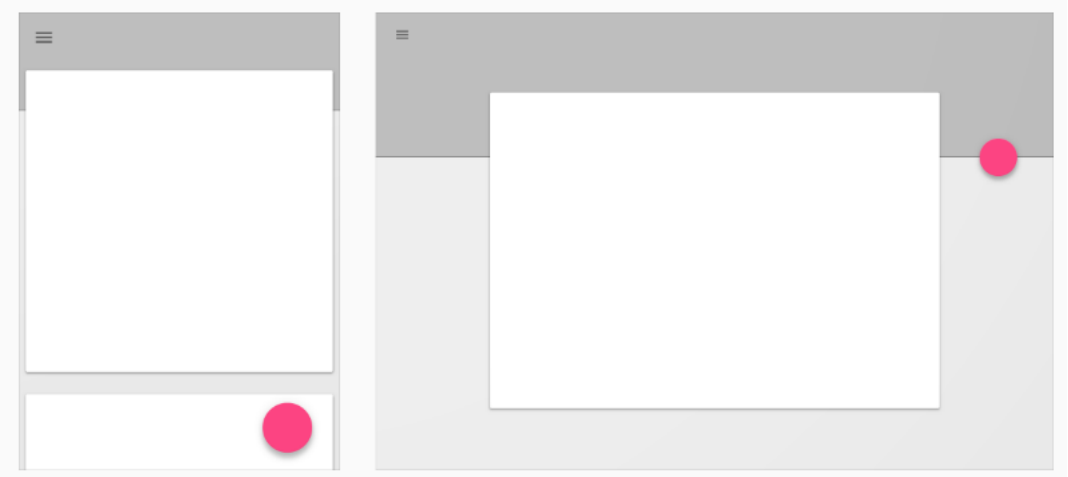
Serverová část však při každém požadavku na veřejný klíč serveru znovu vytváří kryptografickou sůl, která je použita při šifrování uživatelského jména a hesla. Tímto postupem je zaručena změna výsledné šifry uživatelského jména a hesla. Proto když se útočník pokusí odeslat přihlašovací údaje znovu, budou zamítnuty, jelikož již byla změněna kryptografická sůl.

5 Možná rozšíření

Následující podkapitoly představují některá možná rozšíření, která byla identifikována v průběhu implementace. Jedná se zejména o zlepšení uživatelského dojmu nebo využitelnosti aplikace.

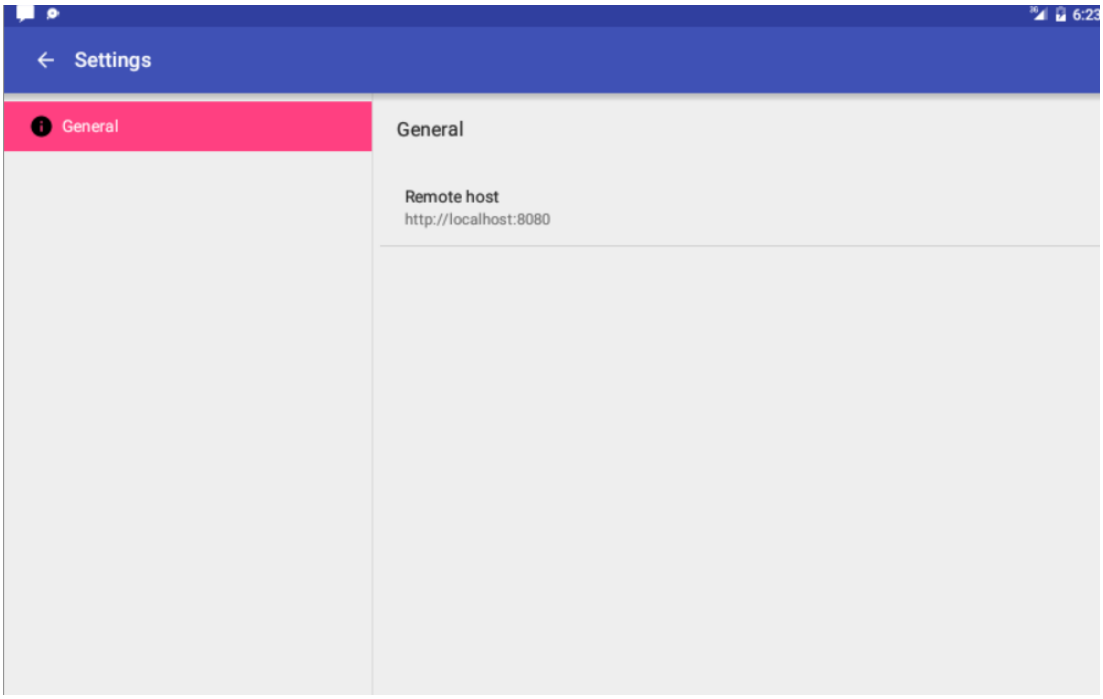
5.1 Uzpůsobení zobrazení pro zařízení typu tablet

Aplikace byla vyvíjena a designována primárně pro mobilní zařízení standardních rozměrů. I když některé obrazovky aplikace jsou uzpůsobeny větším úhlopříčkám, některé ovládací prvky, zejména pak při vyplňování formulářů nejsou pro velké obrazovky ideální. Design aplikace by mohl vycházet z oficiální příručky pro návrh rozhraní Android aplikace. Následující obrázek uvádí příklad návrhu obrazovky pro standardní rozměr telefonu a tabletu. Vzhledem k faktu, že by se zachovala stávající funkcionalita a byly rozšířeny jen možnosti zobrazení, nejedná se o časově náročný zásah do aplikace.

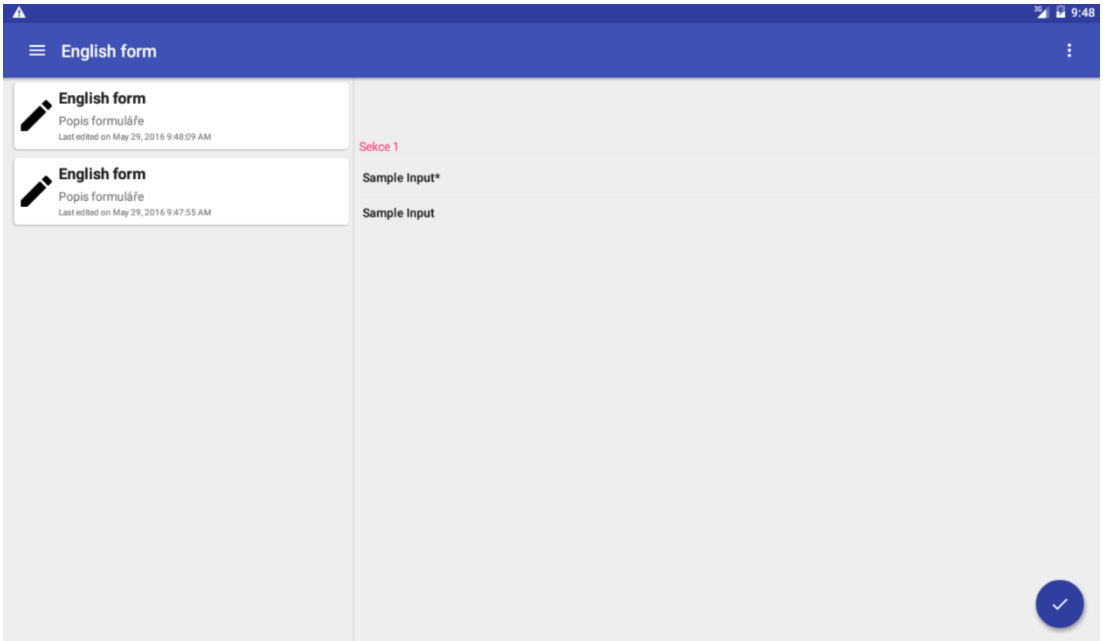


Obrázek 14 Návrh uživatelského rozhraní v porovníní telefon vs. tablet. Zdroj: https://www.google.com/design/spec/layout/responsive-ui.html#responsive-ui-patterns

Některé uživatelské obrazovky, které jsou již připraveny pro zobrazení na větší úhlopříčce, ukazují následující snímky obrazovky.



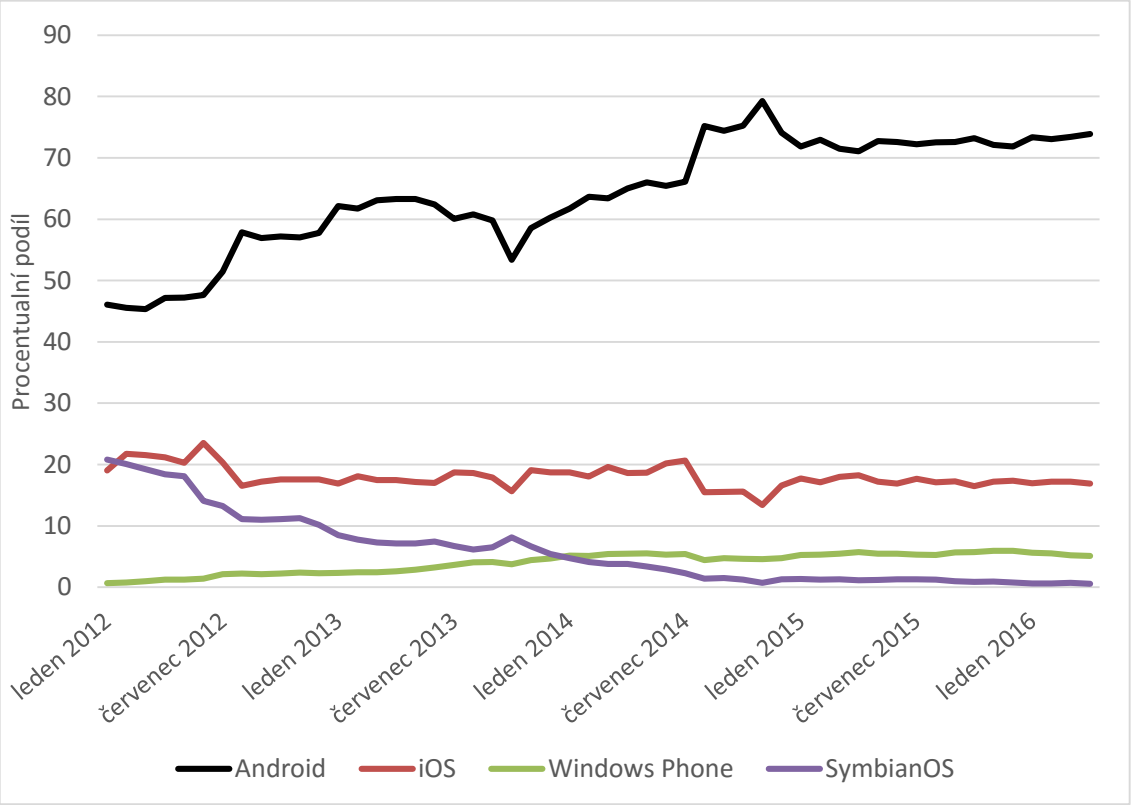
Obrázek 15 Tabletové zobrazení nastavení aplikace ENIS Klient



Obrázek 17 Tabletové zobrazení seznamu vyplňovaných formulářů vlevo a samotného formuláře v pravé části.

5.2 Aplikace pro další platformy

Z důvodu majoritního tržního podílu platformy Android a malými vstupními bariérami pro vývoj a distribuci aplikace byla zvolena jen platforma Android. Další platformy jako například systém iOS od společnosti Apple vyžadují certifikované vývojové prostředí běžící na zařízení od této společnosti. Nicméně díky oblibě tohoto systému lze doporučit vyvinout popisovanou klientskou aplikaci i pro platformu iOS. Další mobilní systémy mají jen velmi malý tržní podíl a vývoj by nebyl ekonomicky výhodný. Následující graf ukazuje historický vývoj zastoupení jednotlivých mobilních operačních systémů.



Graf 1 Historický vývoj tržního podílu mobilních platforem (29)

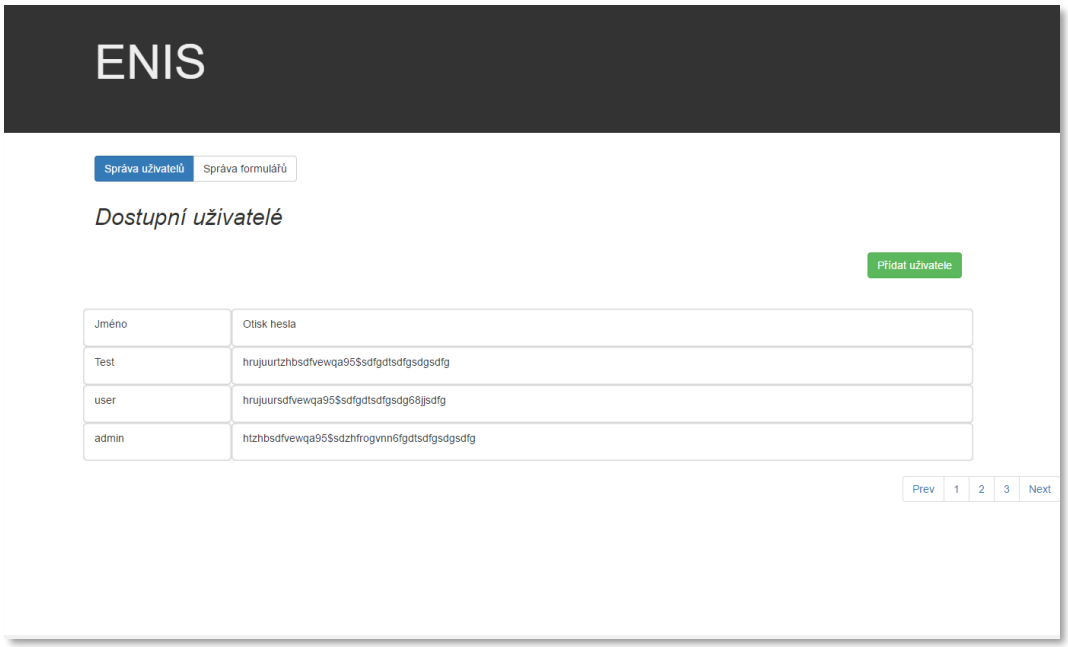
5.3 Využití protokolu https

Pro zvýšení úrovně zabezpečení je vhodné použít komunikační protokol https, který pomocí protokolu TLS umožňuje vytvořit zabezpečené spojení mezi klientskou a serverovou části při komunikaci přes REST rozhraní.

Fungování protokolu https je pro mobilní aplikaci a serverovou část transparentní. Při správné konfiguraci aplikačního serveru a použití důvěryhodného certifikátu stačí upravit komunikační port nastavený v mobilní aplikaci. V případě použití certifikátu, který je digitálně podepsán neznámou certifikační autoritou z pohledu android zařízení, je nutné provést její manuální instalaci.

5.4 Grafické uživatelské rozhraní pro serverovou část

V současnosti je serverová část navržena jen pro komunikaci přes REST rozhraní a definované složky/soubory bez použití dalšího grafického uživatelského rozhraní. Pro lepší správu možných uživatelů, která je momentálně řešena pomocí definovaného souboru, a dalších funkčností, by bylo vhodné navrhnout jednoduché rozhraní. Návrh obrazovky na správu uživatelů je zobrazen na následujícím obrázku.



Obrázek 18 Návrh obrazovky pro serverovou část

5.5 Rozšíření možnosti odesílání formulářů

Pro zjednodušení práce s vyplňovanými formuláři je možné vytvořit nové komunikační kanály pro jejich příjem. Jedním z nejpoužívanějších prostředků komunikace je v dnešní době email. Serverová část by mohla kontrolovat definovanou emailovou schránku na nové zprávy a v případě nalezení vhodné přílohy v definovaném formátu obsahující šifrovaná vyplněná data by takovou zprávu zpracovala. Již nyní je serverová část napojena na různé kanály za použití Spring Integration, přidání nového by tedy nebylo časově příliš náročné, samotné zpracování přijatých dat by zůstalo nezměněné.

6 Závěr

Tato práce představila některé základní metody pro zabezpečený přenos informací mezi dvěma systémy spojenými nezabezpečenou linkou. Tyto metody byly využity při implementaci části experimentálního nemocničního informačního systému, sloužící ke správě formulářových definic včetně sběru jejich dat. Části systému se skládají z nově vytvořené mobilní aplikace na platformě Android komunikující pomocí moderních kryptografických metod s taktéž nově vytvořenou serverovou částí.

Práce poskytla detailní popis obou implementovaných částí i navržené zabezpečené komunikace včetně určení možných hrozeb a způsobu ochrany. V serverové části byla identifikována odesílaná data z Android zařízení, jak pomocí REST rozhraní, tak manuálním vložením do konfigurovatelné složky. Byl definován způsob tvorby formulářů a jejich správa včetně vymezení pro jaké zařízení nebo uživatele je daný formulář určen.

Obě části systému jsou navrženy s ohledem na budoucí rozšiřitelnost. Serverová část je rozložena do několika modulů, každý poskytující ucelenou sadu funkcionalit přes definovaná API (správa uživatelů, formulářů, zabezpečení apod.). Implementace jednotlivých modulů je oddělena, tak aby bylo možné ji kdykoliv nahradit jinou. Klientská část využívá moderní knihovny pro zajištění kompatibility napříč zařízeními a nabízí komfortní vyplňování formulářových dat.

Závěrem práce byla představena možná rozšíření týkající se, jak serverové, tak klientské části. Rozšíření byla zhodnocena v rámci časové náročnosti potřebných úprav.

7 Slovníček pojmů

API	Application Programming Interface, sbírka metod, procedur, funkcí a tříd, které je možné využít vně knihovny/modulu/aplikace.
Autorizace	Proces řízení přístupu.
BPMN 2.0	Business Process Model and Notation – způsob zápisu procesů.
DTLS	Datagram Transport Layer Security, protokol pro zabezpečení například UDP paketů.
Entita	Objekt představující záznam v DB. Řádek databáze při použití ORM
Framework	Aplikační rámec. Souhrn podpůrných programů, knihoven, api apod.
Haš/hash	Generovaný otisk nad daty.
Interceptor	Zachycovač událostí/požadavků na server ve framoworku Spring.
ISO/OSI Model	Model počítačové komunikace přes síť. Rozdělení do 7 vrstev: fyzická vrstva, linková (spojová) vrstva, síťová vrstva, transportní vrstva, relační vrstva, prezentační vrstva a aplikační vrstva.
Java EE	platforma jazyka Java pro tvorbu podnikových aplikací
Javadoc	Vygenerovaná programová dokumentace přímo ze zdrojových kódů.
JSON	JavaScript Object Notation je struktura pro popis uložených dat.
JUnit	Framework pro psaní automatických jednotkových testů.
Keš (cache)	Informace uložené přímo v paměti pro rychlý přístup.
Kontroler/Controller	Část třívrstvé architektury MVC, která slouží k manipulaci s prezenční vrstvou a modelem. Kontroluje datový tok.
Metadata	Data o datech. Například čas změny, práva apod.
ORM	Objektově relační mapování. Mapování atributů tabulky na proměnné ve třídě/objektu a naopak.
Plugin	Zásuvný modul, rozšíření funkčnosti.
POJO	Plain Old Java Object. Obyčejný Java objekt bez žádných speciálních/dalších závislostí.
REST/RESTfull	Representational State Transfer. Datově orientovaná architektura pro aplikační volání založená na CRUD operacích pro http protokol (GET, POST,DELETE,PUT).

SSL	Secure Sockets Layer – předchůdce TLS, z právních důvodů přejmenován právě na TLS.
Symetrické šifry	Šifry používající jeden tajný klíč k zašifrování i dešifrování.
TLS	Transport Layer security - kryptografický protokol poskytující možnost zabezpečené komunikace.
XML	Extensible Markup Language, značkovací jazyk.
XSD	XML (Extensible Markup Language) Schema Documentation / Popis XML schématu.
XSD	XML schema definition, definice XML schématu.

8 Reference

1. Delfs, Hans a Knebl, Helmut. *Introduction to Cryptography - Principles and Applications.* Berlin : Springer Science & Business Media, 2012. 978-3-540-49244-3.

2. Alfred J. Menezes, Paul C. van Oorschot a Scott A. Vanstone. *Handbook of Applied Cryptography.* Florida : CRC Press, 1996. 0-8493-8523-7.

3. Česká republika. Zákon č. 227/2000 Sb., o elektronickém podpisu. 2000.

4. Network Working Group. The Transport Layer Security (TLS) Protocol. *rfc5246.* [Online] [Citace: 8. 5 2016.] https://tools.ietf.org/html/rfc5246#section-1.1.

5. *Transport Layer Security.* [Online] [Citace: 8. 5 2016.] https://en.wikipedia.org/wiki/Transport_Layer_Security.

6. Tomáš Peter, Dalibor Michalec. IPSec. [Online] [Citace: 8. 5 2016.] http://www.cs.vsb.cz/grygarek/TPS-0304/projekty0304/ipsec/ipsec.html.

7. Oracle and/or its affiliates. Lambda Expressions. [Online] [Citace: 21. 5 2016.] https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html.

8. Ben Evans, Richard Warburton. Java SE 8 Date and Time. [Online] Oracle. [Citace: 16. 5 2016.] http://www.oracle.com/technetwork/articles/java/jf14-date-time-2125367.html.

9. Apache Maven Project. [Online] [Citace: 16. 5 2016.] https://maven.apache.org/.

10. Foundation, The Apache Software. Apache Tomcat. [Online] [Citace: 16. 5 2016.] http://tomcat.apache.org/.

11. Spring Framework. [Online] Pivotal Software. [Citace: 16. 5 2016.] https://projects.spring.io/spring-framework/.

12. H2 Database Engine. [Online] [Citace: 16. 5 2016.] http://www.h2database.com/html/main.html.

13. Project Lombok. [Online] [Citace: 16. 5 2016.] https://projectlombok.org/.

14. Jackson JSON Processor Wiki. [Online] [Citace: 16. 5 2016.] http://wiki.fasterxml.com/JacksonHome.

15. Josefsson, S. The Base16, Base32, and Base64 Data Encodings. *Request for Comments: 4648.* [Online] [Citace: 22. 5 2016.] https://tools.ietf.org/html/rfc4648.

16. Bisschops, Ralph. PBKDF2WithHmacSHA512 Vs. PBKDF2WithHmacSHA1. [Online] [Citace: 22. 5 2016.] http://stackoverflow.com/questions/19348501/pbkdf2withhmacsha512-vs-pbkdf2withhmacsha1.

17. EMC Corporation. PKCS#1 v2.2: RSA Cryptography Standard. [Online] 2012. [Citace: 26. 5 2016.] http://www.emc.com/emc-plus/rsa-labs/pkcs/files/h11300-wp-pkcs-1v2-2-rsa-cryptography-standard.pdf.

18. S. Turner. RFC5958 - Asymmetric Key Packages. [Online] https://tools.ietf.org/html/rfc5958. 2070-1721.

19. Orion Security Solutions. RFC4158 - Internet X.509 Public Key Infrastructure:. [Online] 2005. [Citace: 26. 5 2016.] https://tools.ietf.org/html/rfc4158.

20. Oracle and/or its affiliates. Java Cryptography Architecture Oracle Providers Documentation for Java Platform Standard Edition 7. [Online] [Citace: 25. 3 2016.] http://docs.oracle.com/javase/7/docs/technotes/guides/security/SunProviders.html.

21. Google, Inc. Java 8 Language Features. [Online] [Citace: 30. 5 2016.] https://developer.android.com/preview/j8-jack.html.

22. Kotačka, Vít. Gradle, moderní nástroj na automatizaci. *Zdrojak.* [Online] [Citace: 29. 5 2016.] https://www.zdrojak.cz/clanky/gradle-moderni-nastroj-na-automatizaci/.

23. FUJI Goro. Android Orma. *GitHub.* [Online] [Citace: 29. 5 2016.] https://github.com/gfx/Android-Orma.

24. Saving Data in SQL Databases. [Online] Google, Inc. [Citace: 29. 5 2016.] https://developer.android.com/training/basics/data-storage/databases.html.

25. Quemb. QMBForm. [Online] [Citace: 2. 5 2016.] https://github.com/quemb/QMBForm.

26. Google, Inc. Google Design. [Online] [Citace: 30. 5 2016.] https://design.google.com/.

27. W3C. *RDF 1.1 N-Triples.* [Online] [Citace: 30. 5 2016.] https://www.w3.org/TR/n-triples/.

28. Wikipedia. Seznam kódů ISO 639-1. [Online] [Citace: 30. 5 2016.] https://cs.wikipedia.org/wiki/Seznam_k%C3%B3d%C5%AF_ISO_639-1.

29. StatCounte. Top 8 mobile OS in Czech Republic. [Online] [Citace: 8. 5 2016.] http://gs.statcounter.com/#mobile_os-CZ-monthly-201201-201604.

30. Jak funguje IPSEC ? [Online] [Citace: 8. 5 2016.] http://www.security-portal.cz/clanky/jak-funguje-ipsec.

31. Breault, Philippe. Parcelable vs Serializable. *Developer Phil.* [Online] [Citace: 29. 5 2016.] http://www.developerphil.com/parcelable-vs-serializable/.

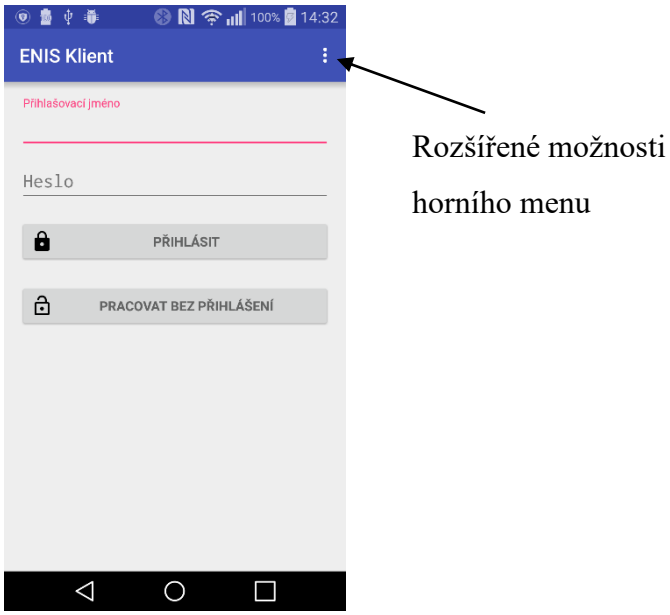
9 Přílohy

9.1 Uživatelská dokumentace aplikace

Nejprve je nutné samotnou aplikaci nainstalovat. Instalace se provede nahráním apk souboru do zařízení a jeho následném spuštění. Kvůli bezpečnosti systému Android je nutné v nastavení povolit instalaci aplikací z neznámých zdrojů tak, aby bylo možné instalovat aplikace i mimo Google Play Store.

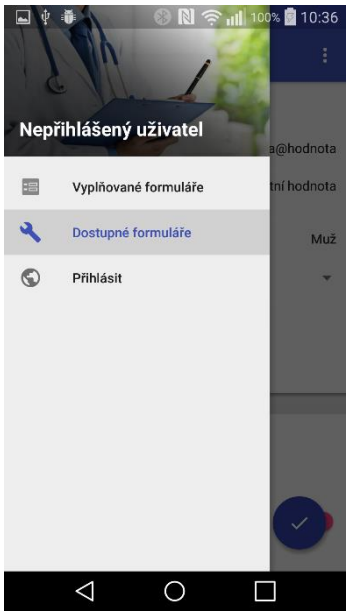
Po úspěšné instalaci je mezi aplikacemi zobrazena následující ikona s titulkem ENIS Klient. Po poklepání se spustí samotná aplikace.

Úvodní obrazovka umožňuje přístup do nastavení přes rozšířené možnosti horního menu (viz následující obrázek), které obsahuje i možnost zobrazení id zařízení, nebo zadání uživatelského jména a hesla, či použití v módu identifikace zařízení.



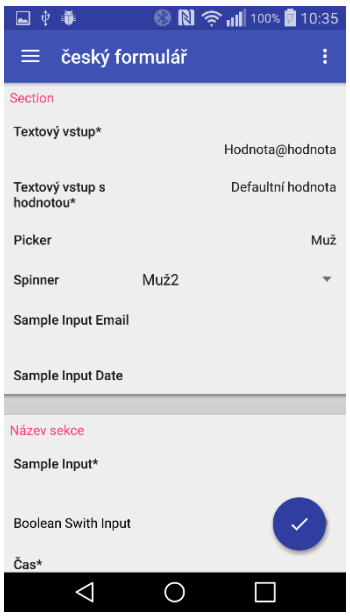
Obrázek 19 Úvodní obrazovka aplikace

Po přihlášení do aplikace jsou přes levé menu (otevírání tahem z levé strany displeje, nebo stisknutím tlačítka v horní části) zobrazeny jednotlivé možnosti navigace.



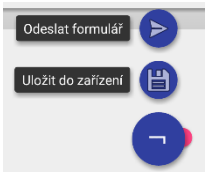
Obrázek 20 Možnosti navigace

Obrazovka s vyplňovanými formuláři obsahuje seznam formulářů, které se již začaly editovat. Po kliknutí na daný formulář je uživatel přesměrován na obrazovku sloužící k vyplňování (viz následující obrázek).



Obrázek 21 Vyplňování formuláře

Odeslání nebo uložení vyplněného formuláře se provede stiskem na modré tlačítko zobrazující možnosti (viz následující obrázek).



Obrázek 22 Možnosti uložení

První možnost provede automatické odeslání formulářových dat na konfigurovaný vzdálený server. Druhá možnost otevře dialogové okno s možností zvolení jména souboru. Výsledný soubor bude uložen v systémové složce pro stažené soubory a bude obsahovat za zvoleným názvem identifikaci uživatele, nebo zařízení.

9.2 Uživatelská dokumentace serverové části

Instalace serverové části se provádí nasazením war souboru na specifikovaný aplikační server. Při použití například Apache Tomcat je možné nahrát soubor do adresáře a aplikační server automaticky spustí serverovou část. Jako ověření správného nasazení a spuštění aplikace slouží adresa <adresa_serveru>/kontext (například.: http://mre.kiv.zcu.cz:8080/enis-backend/). Při nasazení aplikace se načítá konfigurační soubor ve formátu properties určující konfigurační parametry, které jsou následující:

- security.keySize – velikost generovaných klíčů v bitech (např.:2048);
- security.algorithm – algoritmus pro generování klíčů (např.: RSA);
- security.publicKeyFilename – cesta pro uložení veřejného klíče serverové části (např.: C:/temp/enisPublicKey);
- security.privateKeyFilename – cesta pro uložení privátního klíče serverové části. K tomuto souboru by měl mít přístup jen proces serverové části, tak aby bylo zabráněno jeho čtení ostatními, popřípadě jeho modifikace (např.: C:/temp/enisPrivateKey);
- security.usersFile – cesta k souboru uživatelů. (např.: C:/temp/enisUsers.txt);
- db.path – cesta určující uložení databázového souboru H2 databáze. Tento parametr je platformě závislý, a proto musí být zpětná lomítka zdvojena, pokud aplikace běží pod operačním systémem Windows (vlnovka odkazuje na adresář uživatele, např.: ~\\OneDrive\\Skola\\DP\\database\\enis);
- formDefinition.inputDirectory – cesta k adresáři s definicemi formulářů (např.: C:/temp/enisInput);

- formDefinition.processedDirectory - cesta k adresáři se zpracovanými definicemi formulářů (např.: C:/temp/enisProcessed)
- filledForm.inputDirectory – cesta k adresáři pro zpracování vyplněných formulářů (např.: C:/temp/enisFilledInput)
- filledForm.processedDirectory – cesta k adresáři pro zpracované vyplněné formuláře (např.: C:/temp/enisFilledForms)