ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra počítačů

Android aplikace pro komunikaci v bytovém družstvu

Bakalářská práce

Studijní program: Otevřená informatika
Studijní obor: Software

Vedoucí práce: doc. Ing. David Šišlák, Ph.D.

Jiří Miroslav Kačena
Praha 2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kačena** Jméno: **Jiří Miroslav** Osobní číslo: **465874**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Software**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Android aplikace pro komunikaci v bytovém družstvu

Název bakalářské práce anglicky:

Android application supporting residental habitation

Pokyny pro vypracování:

Navrhněte a vytvořte víceuživatelskou aplikaci na bázi androidu, pro komunikaci v bytovém družstvu, ve které budou data uložena na serveru. V systému budou jednoznačně definované role jednotlivých uživatelů a jejich oprávnění. Řádně přihlášený uživatel získá přístup k finančním informacím o svém bydlení a také ke sdíleným datům v bytovém družstvu. Dle patřičné role mu bude aplikace zobrazovat jeho i společné data, případně mu povolí jejich úpravu. Při vytváření aplikace postupujte následujícím způsobem:
1) Vyhledejte a analyzujte dostupné systémy poskytující stejnou nebo obdobnou funkcionalitu.
2) Rozpracujte funkcionalitu vytvářeného systému a porovnejte s existujícími aplikacemi.
3) Seznamte se s technologiemi potřebnými pro vytvoření android aplikace komunikující se serverem prostřednictvím sítového spojení po internetu splňující běžné standardy.
4) Navrhněte vhodné a intuitivní grafické rozhraní aplikace.
5) Implementujte všechny komponenty systému včetně serverové části s její databázovou strukturou.
6) Vyzkoušejte funkčnost aplikace na uživatelích, tyto testy zdokumentujte a vyhodnoťte.

Seznam doporučené literatury:

[1] Schildt, H.: Mistrovstvi Java. Computer Press, 2014
[2] Biehl, Matthias: RESTful API Design (API-University Series) (Volume 3). CreateSpace Independent Publishing Platform; 1 edition, 2016.
[3] Tulach, J.: Practical API Design. Apress, 2008.
[4] Watson, R. T.: Data Management : Databases and Organizations. John Wiley & Sons, 2006.
[5] Fowler, M.: UML Distelled - Third Edition. Addison-Wesley, 2003.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. David Šišlák, Ph.D., centrum umělé inteligence FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **13.02.2019** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **20.09.2020**

doc. Ing. David Šišlák, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání

Podpis studenta

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 7.1.2020 Podpis . . . . . .

Poděkování

Chtěl bych vyjádřit poděkování zejména svému vedoucímu bakalářské práce doc. Ing. Davidu Šišlákovi, Ph.D., jakožto odbornému konzultantovi, za jeho ochotu a vřelý přístup. Dále bych chtěl poděkovat své rodině a přátelům za jejich podporu a pomoc během studia.

Abstrakt

Bakalářská práce se zabývá vývojem mobilní aplikace pro systém Android, která je určena pro usnadnění komunikace v bytových domech. Aplikace pracuje s několika rolemi. Je učena nejen pro domovníka, který se stará o bytový dům, ale také pro vlastníky a nájemníky jednotlivých jednotek. Aplikace je rozdělena na dvě části. V první může uživatel komunikovat s domovníkem nebo s ostatními obyvateli domu. Ve druhé části si uživatel může prohlédnout jemu dostupné finanční dokumenty. Tato druhá část byla vyvinuta ve spolupráci s firmou *Jirra software s.r.o.* Program *Inkasník*, který je určeny na správu bytových domů, posílá finanční a evidenční dokumenty na server, s nímž aplikace komunikuje. Druhá část aplikace je proto přístupná jen uživatelům, jejichž správci používají program *Inkasník*.

Klíčová slova: komunikace, bytový dům, mobilní aplikace, Android, server, Spring Boot, REST.

Abstract

This bachelor thesis deals with the development and design of an mobile application for the Android system. The application is intended to ease a communication in the apartment houses. The application works with several roles. It is not intended only for a caretaker who attends to the apartment house - it is intended for the owners and tenants of the apartment units, as well. The application is divided into two parts. In the first one, the user can communicate with the caretaker and the other residents of the house. In the second one, the user can view the available financial documents. The second part was developer in co-operation with the company *Jirra software s.r.o.* The program "*Inkasník*", intended for the management of apartment houses, sends the financial and evidence documents to a server with which the application communicates. The second part is therefore accessible only for the users whose managers use the program "*Inkasník*".

Keywords: communication, apartment building, mobile application, Android, server, Spring Boot, REST.

Obsah

1. Úvod 13
2. Existující řešení 16
   2.1. Domsys 16
   2.2. Domus 17
   2.3. Resident PortalApps 18
   2.4. SVJO – Společenství vlastníků jednotek online 19
   2.5. Spoluvlastníci 19
   2.6. Ema plus lam 19
   2.7. Závěr 20
3. Návrh aplikace 21
   3.1. Business požadavky 21
      3.1.1. Nefunkční požadavky 21
      3.1.2. Funkční požadavky 22
   3.2. Databázový model 25
4. Implementace 28
   4.1. Server 28
   4.2. Vstupní kód a token 29
   4.3. Aplikace 30
      4.3.1. Role 30
      4.3.2. Přihlašování 31
      4.3.3. První přihlášení do aplikace 32
      4.3.5. Menu možností 33
      4.3.6. Fragment 33
      4.3.7. Dialog 34
5. Testování 35
   5.1. Testování serverové části 35
   5.2. Testování mobilní aplikace 38
6. Závěr 41
7. Zdroje 42
8. Přílohy 44

Obrázky

Obrázek 1 - Aplikace Domsys Správce…….....16

Obrázek 2 - Aplikace eDomus…….....17

Obrázek 3 - Aplikace Green Portal…….....18

Obrázek 4 - Aplikace Leasa…….....19

Obrázek 5 – Databázový model…….....25

Obrázek 6 – Přihlašovací obrazovka…….....31

Obrázek 7 – Obrazovka se souhlasem …….....32

Obrázek 8 – Navigační menu…….....32

Obrázek 9 – Menu možností…….....33

Obrázek 10 – Kód pro testování třídy UserService ……..……...36

Obrázek 11 – Funkce pro testování funkce saveNote()……...37

Obrázek 12 – Obrazovka pro zobrazení informací o bytu…….…….....40

1.Úvod

Cílem této práce bylo vytvořit uživatelsky přívětivou aplikaci, která usnadní komunikaci v bytovém domě, a také umožní uživatelům si prohlédnout přehledně zobrazené finanční informace, které se vztahují k jejich bytu. Prvním impulzem pro vytvoření této aplikace byla nabídka od firmy *Jirra Software*[1], která se zabývá vývojem počítačových programů pro vyúčtování bytových domů, na vytvoření mobilní aplikace, jež by uživatelům umožňovala mít přehled o základních finančních údajích o bytu, ve kterém bydlí nebo jej vlastní. Taková aplikace by byla ovšem banální a uživatelům by nenabízela mnoho funkcí. Proto bylo potřeba se zamyslet nad tím, jak by mohla mobilní aplikace usnadnit život obyvatelům bytových domů. Po poradě s lidmi, kteří v bytových domech žijí, a domovníkem, který dům spravuje, jsme zjistili, že největším problémem je komunikace mezi obyvateli a domovníkem. Aplikace je proto zaměřená z části na komunikaci v bytovém domě, které probíhá v každém domě jinak, neboť existují více druhů bytových domů.

V České republice existují 3 druhy bytových domů. Prvním typem je společenství vlastníků jednotek (SVJ). Jednotky (prostory určené pro bydlení, či k jiným účelům) v bytovém domě vlastní několik vlastníků, kteří v bytech bydlí nebo je pronajímají, a kteří se společně podílí na správě společných prostor. Správné fungování SVJ zajišťuje správce, který se stará o veškeré vyúčtování domu a domovník, jenž zajišťuje komunikaci mezi vlastníky a správcem domu, a který také řeší problémy s domem spojené. Správce ani domovník ovšem nemusí být vlastníky bytu v daném SVJ a ani v něm nemusí žít. SVJ jako takové žádné jednotky nevlastní, neboť vlastníky jen sdružuje.

Bytové družstvo oproti SVJ bytové jednotky vlastní a obyvatelé jsou členy bytového družstva. Tento typ bytového domu byl velmi častý za komunismu a v současné době se již od tohoto způsobu upouští. Bytová družstva se snaží změnit na společenství vlastníků jednotek tím, že obyvatelům bytů dané byty postupně prodávají.

Dalším typem bytového domu je nájemní či činžovní dům, jenž vlastní majitel, který se o budovu stará sám nebo prostřednictvím správní firmy, a byty pronajímá nájemníkům. Největšími majiteli činžovních domů jsou města a obce, které si zřizují své správní firmy a ty se o domy starají.

Členové SVJ se pravidelně schází na shromážděních, kde řeší problémy SVJ a mimo tyto schůze spolu komunikují především přes e-mail. Obyvatelé činžovních domů tuto možnost nemají. Tito obyvatelé se většinou neznají a nemají na sebe ani žádný kontakt. Nemají tedy moc možností, jak by mohli ostatním obyvatelům cokoliv sdělit. Domovník s obyvateli komunikuje většinou pomocí e-mailu. Pokud má nějakou informaci, která je určena všem obyvatelů domu, může ji přidat na nástěnku, která se většinou nachází u vstupních dveří. Na tu se ovšem běžný obyvatel moc často nedívá, a proto může nějakou důležitou informaci lehce přehlédnout.

V současné době někteří správci bytových komplexů poskytují vlastníkům/nájemníkům přehledy o platbách ve formě textového dokumentu, které jim posílají e-mailem. Tyto výpisy obsahují mnoho informací a jsou často nepřehledné. Běžný vlastník/nájemník většinou tolik informací ani nevyžaduje. Stačí mu pouze pár základních údajů, jako například kolik má za měsíc celkově zaplatit. Informace, které uživatele nejčastěji zajímají, jsou zálohy na služby, pohledávky a vyúčtování.

Každý uživatel bytu platí zálohy na služby spojené s užíváním bytu, které jsou určeny výborem společenství vlastníků. Jedná se například o osvětlení společných prostor, společná anténa, úklid, čištění komínu, ohřev vody, a tak podobně. Součet těchto záloh tvoří předpis, který obyvatel nebo vlastník bytu musí každý měsíc zaplatit.

Pohledávky slouží jako přehled, kolik a kdy, měl daný obyvatel nebo vlastník bytu, za daný byt zaplatit, a také kolik a kdy zaplatil. Obvykle je hodnota pohledávky součtem všech záloh za období splatnosti (nejčastěji každý měsíc) za daný byt. Proto se v seznamu obvykle objeví dvanáct pohledávek za rok, a pokud uživatel pohledávky řádně platí, objeví se v seznamu i dvanáct plateb.

Jelikož se skutečný náklad za danou službu nemusí shodovat s její zálohou, musí se každý rok vytvořit vyúčtování vztahující se k danému bytu, ve kterém je uvedený součet všech skutečných nákladů za služby a součet všech záloh na služby za daný rok. Následně, když sečteme skutečný náklad a zálohu za službu, zjistíme, jestli má daný obyvatel nebo vlastník bytu doplatek či nedoplatek.

Po vyhodnocení výše uvedených informací jsme navrhli aplikaci obsahující nástěnku, jež uživatelům umožňuje přidávat jednotlivé poznámky, události či kontakty. Nabízí také veřejnou konverzaci určenou pro komunikaci všech obyvatelů domu či privátní konverzaci, určenou pro komunikaci obyvatelů s domovníkem. Domovníkovi aplikace nabízí možnost přehledně zobrazit jednotlivé obyvatele či vlastníky bytů a byty v daném domě. Nakonec, pokud budou finanční data daného domu spravována pomocí programu společnosti *Jirra Software*, mohou si obyvatelé a vlastníci bytů v aplikaci prohlédnout zálohy, pohledávky a vyúčtování vztahující se k jejich bytu.

Následující práce je rozdělená do několika částí, ve kterých se seznámíme s vývojem aplikace. Nejprve si ukážeme aplikace, které náš problém řeší a popíšeme jejich výhody a nevýhody. Následně si představíme použité technologie, s jejichž pomocí byla aplikace vytvořena a popíšeme si návrh aplikace. Na konec se budeme zabývat implementací a testováním aplikace.

2. Existující řešení

V současné době již existuje několik aplikací, které se zabývají podobnou tématikou. Aplikace jsou obvykle propojené s větším systémem, který se používá především na správu financí konkrétního bytového domu. Takový systém má několik uživatelských rozhraní, která jsou určená různým typům uživatelů. Správci používají desktopovou či webovou aplikaci pro vyúčtování a správu služeb. Vlastník či nájemník pak používá mobilní nebo webovou aplikaci ke komunikaci s ostatními návštěvníky a ke správě svých finančních informací vztahujících se k jeho bytu. Pokud však aplikace neobsahuje komunikační prvky, musí si uživatel vystačit s běžnými prostředky ke komunikaci, jako jsou nástěnka či email.

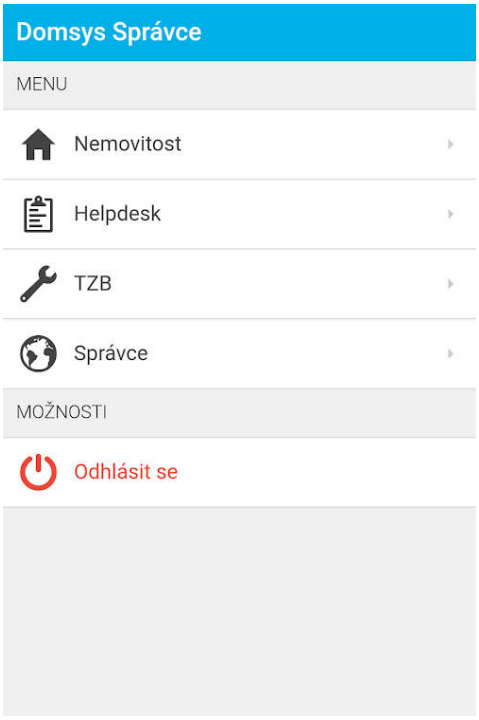
Mobilní aplikace v sobě zahrnují řadu různých funkcí pro komunikaci s ostatními obyvateli a pro zobrazování informací o platbách. Některé mobilní aplikace jsou určeny jen správcům a zaměstnancům budovy, které jim umožňují hlásit škody na majetku a zobrazovat informace o obyvatelích domu nebo kontakty na řemeslníky a opraváře.

2.1. *Domsys*

Domsys[2] je firma zabývající se vývojem on-line softwaru pro správu různých typů nemovitostí. Díky jejich softwaru může správce spravovat vlastníky či nájemníky bytů.

Systém správci umožňuje zobrazit veškeré údaje o vlastnících či nájemnících bytů a snadno s nimi komunikovat. Vlastníkům či nájemníkům bytu umožňuje si zobrazit své finanční informace. Pro tyto účely firma *Domsys* vyvinula dvě mobilní aplikace.

První aplikace *Domsys Portál*[3] je určena vlastníkům a nájemníkům jednotlivých bytů. Ti si skrze tuto aplikaci mohou detailně prohlídnout informace o svém bytu či o celém domě anebo si mohou zobrazit finanční informace vztahující se k jejich bytu, jako jsou například



Obrázek 1 - Aplikace Domsys Správce

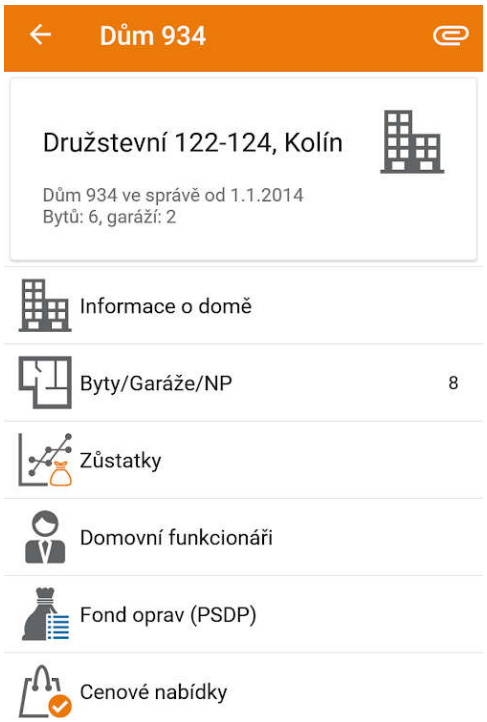
předpisy, platby, faktury atd. Dále aplikace nabízí uživatelům možnost zasílat správci různé typy žádostí. Správce je posléze o vývoji řešení informuje pomocí e-mailu. Uživatel si také může sám provádět kontrolu měřidel a výsledné informace odeslat správci.

Druhá aplikace *Domsys Správce*[4] je určena správci domu. S její pomocí si může prohlédnout veškerá data o všech domech, jednotkách a obyvatelích, které spravuje. Správce má také přístup ke všem uloženým souborům týkajících se domů, které spravuje.

Společnost *Domsys* se zabývá především vývojem webových aplikací pro správce či domovníky nemovitostí. Proto i jejich mobilní aplikace nemají tolik funkcionality jako webová aplikace. Aplikace jsou přesto uživatelsky přívětivé a poskytují mnoho funkcionality. Oddělení uživatelské a správcovské části nemusí vyhovovat správcům, kteří ve spravovaném domě zároveň bydlí, jelikož v mobilním telefonu musí mít dvě aplikace od stejné firmy. Obyvatelé domu díky aplikaci *Domsys Portál* mohou kontaktovat správce, ale již nemohou komunikovat s ostatními obyvateli domu.

2.2. *Domus*

Domus[5] je modulární systém pro správu nemovitostí a bytů vyvinutý společností *Anasoft*. Systém je rozdělen do několika speciálně zaměřených modulů pro správu domů, financí, mezd a energií. Společnost *Anasoft* k tomuto systému také vyvinula dvě mobilní aplikace *eDomus*[6] a *Odpočet měřičů – Poschodech*[7]. Tyto aplikace jsou učeny pouze zaměstnancům společnosti, která používá systém *Domus*.



Obrázek 2 - Aplikace eDomus

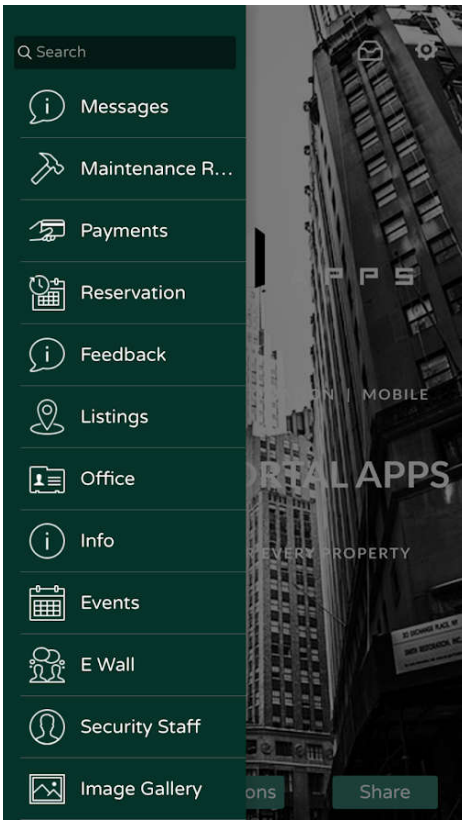
Aplikace *eDomus* umožňuje uživatelům online přístup k informacím uloženým v systému *Domus*. Uživatel si tak v mobilním telefonu či tabletu může zobrazit a editovat kontaktní údaje, popřípadě data jednotlivých obyvatel domu. Aplikace uživateli také umožňuje stáhnout si do zařízení různé dokumenty vztahující se k danému domu, kde s nimi může dále pracovat. Další funkcí aplikace *eDomus* je možnost vytváření záznamů vztahujících se k domu či bytu. Uživatel díky této funkci může nahlásit škodu na majetku či závadu a odeslat žádost o opravu.

Druhou aplikací od společnosti *Anasoft* je aplikace *Odpočet měřičů –Poschodech*, ta je určená zaměstnancům, kteří provádí odpočet měřičů. Zaměstnanci tak mohou lehce zapsat odečet měřiče do aplikace a rovněž k němu mohou pořídit fotografii měřiče. Po připojení k internetu aplikace záznam o měření automaticky uloží do systému a kopii pošle e-mailem vlastníkovi jednotky.

Obě aplikace od firmy *Anasoft* jsou určené pouze pro zaměstnance společnosti, která používá systém *Domus* a bohužel nemají žádnou aplikaci pro vlastníky či nájemníky bytů.

2.3. *Resident PortalApps*

Společnost *GreenApps*[8] vytváří mobilní aplikace pro správce obytných a obchodních nemovitostí. Správce si sám může určit, jaké funkce bude aplikace obsahovat. Tato aplikace usnadní obyvatelům nemovitosti komunikaci se správcem i s ostatními obyvateli. Správce nemovitosti může pomocí aplikace zasílat individuální nebo skupinové zprávy obyvatelům dané budovy. Správci budovy umožňuje kontrolovat informace o platbách či jiných poplatcích, které od obyvatel dostává. Obyvatelé si díky této aplikaci mohou zarezervovat konkrétní místnosti ve společných prostorách pro libovolnou událost, nebo mohou zasílat požadavky na údržbu s popisem problému, fotografií nebo videem. Správce muže poskytnout obyvatelům základní informace o domě, pravidlech a bezpečnosti.



Obrázek 3 - Aplikace Green Portal

Společnost *GreenApps* vytváří aplikace na míru, díky čemuž jsou aplikace efektivnější. Správce nemusí dělat žádné kompromisy a může navolit funkcionalitu, jakou potřebuje. Firma *GreenApps* se ovšem specializuje především na kanadský a americký trh, a proto jsou její mobilní aplikace v anglickém jazyce.

2.4. *SVJO – Společenství vlastníků jednotek online*

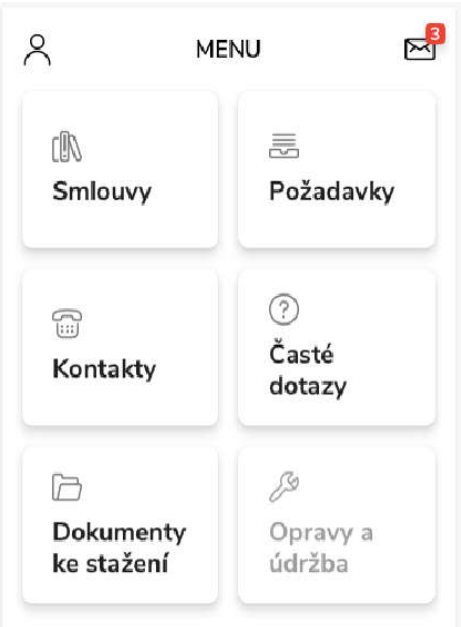
SVJO[9] je webová aplikace pro lepší komunikaci v bytovém domě. Je vytvořena tak, aby ji mohli jednoduše ovládat všechny věkové skupiny. Při zakoupení webové aplikace *SVJO* správce obdrží zabezpečené stránky, kam budou mít přístup pouze členové a kde mohou společně diskutovat a hlasovat. Webová aplikace obsahuje kalendář událostí, kde mohou uživatelé plánovat akce nebo si rozdělit práce v domě. Pomocí webové aplikace *SVJO* může správce zaslat SMS zprávu s upozorněním na blížící se schůzi nebo s jiným upozorněním. Uživatelé zde mohou nahlásit jakoukoliv závadu v bytě či domě. Webová aplikace eviduje měření v domě i jednotkách.

2.5. *Spoluvlastníci*

Webová aplikace *Spoluvlastníci*[10], vyvinutá společností *Artin*, je především zaměřená na komunikaci v bytovém domě. Webová aplikace umožňuje uživatelům používat různou funkcionalitu jako je například nástěnka. Zde mohou uživatelé sdílet důležité informace, které jsou podstatné i pro ostatní. Dále zde uživatelé mohou jednoduše hlasovat o různých věcech pomocí ankety, kde se dá dokonce nastavit i váha hlasu jednotlivých vlastníků podle podílů. Uživatelé také mohou vkládat dokumenty, u kterých mohou rozhodnout, kteří uživatelé k nim budou mít přístup. Nakonec skrze tuto webovou aplikaci uživatelé mohou hlásit závady na domě.

2.6. *Ema plus lam*

Ema plus lam[11] je webová aplikace určená pro správu a řízení nájemních vztahů v rámci komerčních i bytových portfolií, která byla vyvinutá společností *TescoSW*. Pomocí této aplikace má správce nájemního domu možnost pracovat s jednotlivými nájemníky a dokumenty či smlouvami, které se k nim vztahují. Správce ve webové aplikaci může definovat jednotlivé verze předpisů nájemních vztahů, záloh na služby a dalších služeb. Má také možnost zobrazit si jednotlivé faktury či vytvářet nové.



Obrázek 4 - Aplikace Leasa

Společnost *TescoSW* k webové aplikaci *Ema plus lam* vytvořila i mobilní aplikaci *Leasa*. Ta slouží pro komunikaci mezi správcem a nájemníkem. Uživatel zde má možnost zobrazit si smluvní informace týkající se nájmu nebo si může zobrazit rozpis plateb a vyúčtování. Aplikace uživateli také umožňuje zobrazit si uskutečněné platby nebo provést nové platby pomocí čárového či QR kódu.

2.7. *Závěr*

Všechny výše uvedené aplikace daný problém zčásti řeší. Aplikace *Domsys* je zaměřená především na zobrazování finančních informací, ale už neřeší většinu komunikačních problémů v bytovém domě. Stejným nedostatkem trpí i aplikace *Leasa*, která navíc je určena pouze pro nájemní domy. Aplikace od společnosti *Anasoft* jsou zaměřeny pouze na správce a zaměstnance bytových komplexů. Aplikace Resident *PortalApps* je zaměřená spíše na kanadský a americký trh. Webové aplikace *SVJO* a *Spoluvlastníci*, jež jsou určeny především pro SVJ, řeší skoro všechny problémy spojené s komunikací, neřeší však zobrazování finančních informací.

3. Návrh aplikace

Náš systém je navržen jako server-klient aplikace, kde klientskou část reprezentuje Android[12] aplikace. O serverovou část se stará Spring Boot[13] aplikace, která je umístěna na Apache Tomcat[14] serveru. Na serveru je také umístěna relační databáze, do které jsou ukládány data. Návrh obou aplikací je velmi důležitý pro snadný a rychlý vývoj sytému, proto musí být pečlivě promyšlen a otestován, aby se předešlo vývoji zbytečné funkcionality. Proto jsme návrh v průběhu vývoje testovali a předělávali.

3.1. *Business požadavky*

Business požadavky jsou požadavky, které jsou kladeny na systém. Analýza požadavků je důležitá pro správné navržení aplikace.

3.1.1. *Nefunkční požadavky*

Nefunkční požadavky jsou důležité pro vývoj kvalitní a stabilní aplikace. Jsou to především systémové požadavky:

a) **Škálovatelnost**
   Server dokáže přijímat větší množství požadavků, aniž by se to výrazně projevilo na délce čekání na jednotlivou žádost.

b) **Spolehlivost**
   I při zvýšené zátěži bude systém správně vykonávat požadavky, jak se od něj očekává.

c) **Rozšiřitelnost**
   Mobilní i webová aplikace budou vytvořeny tak, aby se v budoucnu daly lehce upravit či rozšířit.

d) **Bezpečnost**
   Server bude pracovat na zabezpečeném protokolu HTTPS. Systém umožní uživateli bezpečné přihlášení pomocí Google účtu. Po jeho přihlášení systém prověří, jestli má uživatel platný token a zda má právo pracovat s danými daty. Poté provede jeho požadavky.

e) **Ochrana osobních údajů**
   Systém umožní pracovat s citlivými údaji pouze osobě, která má na tyto údaje právo. Zároveň systém uživateli umožní, aby své údaje odstranil ze systému.

3.1.2. *Funkční požadavky*

Funkční požadavky definují, co uživatel může se systémem dělat. Co od systému očekáváme:

a) Uživatel

- **Přihlášení do systému**

  Systém bude umožňovat uživateli přihlášení do systému pomocí účtu od společnosti *Google*.

- **Odhlášení ze systému**

  Systém bude umožňovat uživateli odhlášení ze systému.

- **Zrušení účtu**

  Systém bude umožňovat uživateli odstranit své údaje ze systému.

- **Změna jazyka**

  Systém bude umožňovat uživateli změnit jazyk v aplikaci.

- **Vytvoření domu**

  Systém bude umožňovat uživateli vytvořit nový dům a stát se domovníkem.

- **Spojení s domem po zadání přihlašovacího kódu**

  Systém bude umožňovat uživateli zadat přihlašovací kód a poté ho propojí s jeho domem.

- **Vybrání role**

  Systém bude po přihlášení uživateli umožňovat volbu mezi domovníkem nebo obyvatelem v případě, že uživatel bude domovník i obyvatel zároveň.

- **Vybrání domu nebo bytu**

  Systém bude umožňovat uživateli, jenž je spojen s více domy či byty, si zvolit, do jakého domu či bytu se chce přihlásit.

b) Obyvatel

- **Správa poznámky, události nebo kontaktu**

  Systém bude umožňovat obyvateli vykonávat základní CRUD[1] operace s poznámkou, událostí nebo kontaktem, pokud na danou operaci bude mít právo.

- **Napsat veřejnou zprávu**

  Systém bude umožňovat obyvateli napsat zprávu do hromadné konverzace v daném domě.

- **Napsat zprávu domovníkovi**

  Systém bude umožňovat obyvatelovi napsat zprávu domovníkovi daného domu.

[1] CRUD (Create, Read, Update, Delete) – Zkratka shrnuje čtyři základní operace nad prvkem v databázi.Vytvořit, číst, editovat a smazat.

c) Obyvatel domu s programem od firmy *Jirra software*

- **Zobrazit pohledávky, vyúčtování nebo zálohy**

  Systém bude umožňovat obyvateli domu s programem od firmy *Jirra software* zobrazit jeho pohledávky, vyúčtování nebo zálohy jeho bytu.

d) Domovník

- **Správa poznámky, události nebo kontaktu**

  Systém bude umožňovat domovníkovi vykonávat základní CRUD operace s jakoukoliv poznámkou, událostí nebo kontaktem, která patří ke spravovanému domu.

- **Napsat veřejnou zprávu**

  Systém bude umožňovat domovníkovi napsat zprávu do hromadné konverzace daného domu.

- **Napsat zprávu jakémukoliv bytu**

  Systém bude umožňovat domovníkovi napsat zprávu jakémukoliv bytu v daném domě.

- **Upravit informace o domě**

  Systém bude umožňovat domovníkovi upravit informace o daném domě.

- **Odstranit dům**

  Systém bude umožňovat domovníkovi odstranit daný dům i všechna data s ním spojená.

- **Správa bytů**

  Systém bude umožňovat domovníkovi vykonávat základní CRUD operace s bytem v domě, který spravuje.

- **Vytvořit nového obyvatele bytu**

  Systém bude umožňovat domovníkovi vytvořit nového obyvatele v daném bytě a přiřadit mu roli v tomto bytě.

- **Upravit informace o obyvateli**

  Systém bude umožňovat domovníkovi upravit informace o obyvateli daného domu.

- **Odstranit obyvatele**

  Systém bude umožňovat domovníkovi odstranit obyvatele daného domu.

- **Propojit uživatele s bytem**

  Systém bude umožňovat domovníkovi propojit obyvatele s bytem v daném domě.

- **Odebrat uživatele z bytu**

  Systém bude umožňovat domovníkovi odebrat obyvatele z daného bytu v daném domě.

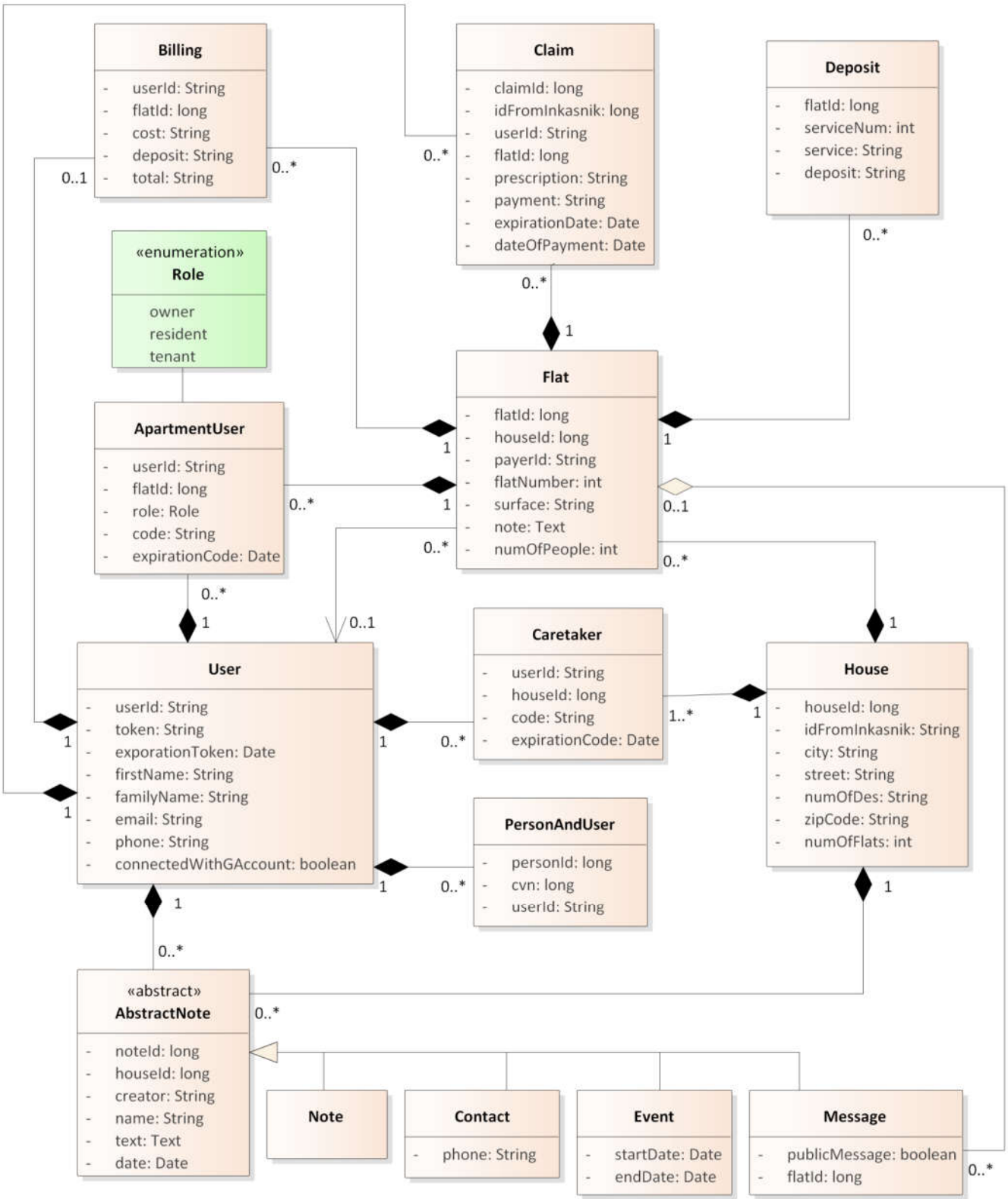
- **Upravit informace o obyvateli bytu**

  Systém bude umožňovat domovníkovi upravit informace o obyvateli daného bytu.
- **Vygenerovat vstupní kódy**

  Systém bude umožňovat domovníkovi vygenerovat vstupní kódy pro obyvatele domu.
- **Předat domovnická práva**

  Systém bude umožňovat domovníkovi předat domovnická práva jinému obyvateli bytu.

3.2. *Databázový model*



Obrázek 5 – Databázový model

Databázový model[15] zachycuje objekty uložené v relační databázi a vztahy mezi nimi. Každý objekt musí mít svůj unikátní identifikační klíč, pod kterým jej lze v databázi nalézt. Některé identifikační klíče mohou být utvořeny z několika parametrů.

Mezi hlavní objekty patří objekt uživatel (User). Každý uživatel má svůj identifikační klíč, který je buď vygenerován, nebo zděděn z Google účtu. Uživatel má také svůj unikátní token s exspirací, generuje se automaticky po přihlášení. Token složí pro bezpečnou komunikaci mezi mobilní aplikací a serverem. Jméno, přímení, e-mail a telefon jsou parametry, které jsou viditelné v aplikační vrstvě. Jsou to také parametry, které se dají v aplikační vrstvě měnit. Pro rozlišení, zda je uživatel virtuální nebo se již do aplikace přihlásil, slouží parametr connetedWithGAccount. Pro nalezení uživatele při importování dat z programu *Inkasník*[2], je objekt User propojen s objektem PersonAndUser. Objekt PersonAndUser obsahuje tři parametry. Identifikační klíč uživatele, identifikační klíč z databáze programu *Inkasník* a IČO dané databáze. Tyto tři parametry slouží k propojení uživatele uloženého v databázi, s uživatelem, který je obsažen v importovaných datech z programu *Inkasník*.

Další důležitým objektem je objekt dům (House). I dům má svůj identifikační klíč, který se automaticky generuje při vytvoření. Dále má parametry město, ulice, číslo popisné a poštovní směrovací číslo, ty se v aplikační vrstvě dají změnit. Uvedené parametry jsou ošetřené tak, aby nemohl byt vytvořen dům se stejnou adresou. Parametr počet bytů se mění podle počtu bytů v domě. Parametr idFromInkasník má stejnou úlohu jako objekt PersonAndUser. Slouží pro nalezení domu při importu dat z programu *Inkasník* a tvoří jej dva parametry IČO a identifikační klíč z databáze daného programu *Inkasník*. Každý dům je spojen s uživatelem, který má roli domovníka. Toto spojení zajišťuje objekt domovník (Caretaker), jenž obsahuje identifikační klíče uživatele a domu. Objekt domovník také obsahuje vstupní kód a datum exspirace tohoto kódu. Vstupní kód je vygenerován při vytvoření domovníka a je odeslán na jeho e-mail. Po přihlášení uživatel zadá tento kód a propojí se s domem.

Dům obsahuje byty, objekt byt (Flat) je tedy důležitý. Byt obsahuje identifikační klíč a také identifikační klíč domu, ve kterém se nachází. Dále obsahuje parametr číslo bytu, který je unikátní pro daný dům. Číslo bytu je zadáno při vytvoření bytu a pak se již nemůže změnit – na rozdíl od parametrů plocha a poznámka. Tyto parametry lze v aplikační vrstvě kdykoliv změnit. Parametr počet osob udává počet obyvatel konkrétního bytu. Byt by měl mít také svého plátce, proto zde je parametr plátce bytu, který propojuje byt s uživatelem, který za byt platí.

[2] Inkasník je program od společnosti Jirra Software. https://jirra.cz/inkasnik/

Objekt ApartmentUser slouží pro propojení uživatele s bytem. Jeho identifikační klíč je tvořený ze dvou identifikačních klíčů, uživatele a bytu. Pro určení role uživatele je zde parametr role, který lze nastavit na obyvatele, vlastníka nebo nájemce. Parametry kód a exspirace kódu zde slouží podobně jako u objektu domovník pro propojení uživatele a bytu. Aby se uživatel mohl propojit s bytem, musí domovník vygenerovat vstupní kód, který uživateli předá a on ho zadá do aplikace.

AbstractNote je abstraktní objekt, jehož parametry dědí čtyři další objekty. Každý potomek objektu AbstractNote má následující parametry. Identifikační klíč a datum, které jsou vygenerovány při ukládání do databáze. Identifikační klíč uživatele, který objekt vytvořil. Identifikační klíč domu, kam objekt patří. A nakonec jméno a text. Základním objektem děděným od AbstractNote je poznámka (Note), jež nemá žádné další parametry na rozdíl od události (Event). Ta má dva další parametry – začátek a konec události. Kontakt (Contact) má navíc telefonní číslo. A nakonec zpráva (Message) má dva parametry. Parametr, který rozlišuje, zda je zpráva veřejná, a identifikační klíč bytu, ke kterému zpráva patří. Pokud se jedná o veřejnou zprávu, je číslo bytu nevyplněno a tuto zprávu může vidět kdokoliv z daného domu. Pokud se však jedná o privátní zprávu, může ji vidět jen domovník nebo obyvatelé bytu, kterému zpráva náleží.

Posledními třemi objekty jsou zálohy (Deposit), pohledávky (Claim) a vyúčtování (Billing). Tyto objekty mohou být vytvořeny pouze při importu dat z programu *Inkasník*. Zálohy se vztahují ke konkrétnímu bytu, ke kterému se nemohou vztahovat dvě stejné zálohy, a proto je identifikační klíč zálohy tvořený z identifikačního klíče bytu a čísla služby. Záloha dále obsahuje konkrétní zálohu za danou službu a částku s ní spojenou. Objekt vyúčtování se vztahuje k uživateli a bytu. Uživateli, který je propojen s konkrétním bytem, může být zobrazeno jen jedno vyúčtování za předchozí rok, proto je identifikačním klíčem objektu vyúčtovaní tvořené z identifikačního klíče uživatele a bytu. Dále obsahuje parametry náklad, záloha a výsledek. Pohledávek vztahujících se ke konkrétnímu bytu a uživateli může být několik, proto objekt pohledávka musí obsahovat identifikační klíč, který je vygenerovaný při vytvoření. Dále obsahuje parametry den splatnosti, předpis, den platby a platba. Nakonec obsahuje také parametr idFromInkasnik, který slouží, stejně jako u objektu dům, pro nalezení pohledávky při importu dat z programu *Inkasník*.

4. Implementace

Aplikace je rozdělena na serverovou část a Android aplikaci. Obě tyto části jsou psané v programovacím jazyku Java[16] především proto, že s tímto jazykem mám největší zkušenosti. Serverová a aplikační část spolu komunikují přes vytvořené Rest API[17] pomocí zabezpečeného protokolu HTTPS[3]. Posílaná data jsou převedena do formátu JSON[18], který je vložen do těla HTTP[4] požadavku. Aplikace odešle tento požadavek na server, který na základě adresy URL a HTTP metody rozhodne, jakou operaci má s daným požadavkem vykonat. HTTP metody použité v této aplikaci jsou GET, POST, PATCH a DELETE. Metoda GET slouží pro získání dat ze serveru, DELETE na jejich odstranění, POST na uložení dat a PATCH na změnu dat. Metody GET a DELETE nepotřebují žádná data v těle požadavku. Každý požadavek obsahuje hlavičku, která obsahuje parametry. V našem případě každá hlavička obsahuje token na ověření uživatele. Server požadavek odeslaný z aplikace zpracuje a zareaguje na něj svou odpovědí. Ta obsahuje hlavičku, tělo a status. Status obsahuje číselný kód, který značí, jak server daný požadavek zpracoval. Protokol HTTP definuje, co tyto kódy značí a kdy se mají použít.

4.1. *Server*

Pro implementaci serverové části jsem zvolil Java Spring Boot framework především z důvodu snadného vývoje aplikace, díky jednoduché konfiguraci a také díky velkému množství dostupných materiálům o vytváření Spring Boot[19] aplikace. Právě díky jednoduché konfiguraci stačí v aplikaci správně naimplementovat třídy představující controller, service a repository a o ostatní funkcionalitu se postará Spring boot. Controller slouží k obsluze požadavků, které mobilní aplikace posílá na server. Services jsou určeny pro implementaci business logiky. A Repository slouží k propojení aplikace s databází.

V aplikaci je mnoho tříd s anotací @Service, které slouží k různým účelům a nejvíce z nich se stará o validaci. Každý controller má vlastní service, který slouží k ověření požadavku a přípravě dat, jež se k tomuto požadavku vztahují. Další services obsahují složitější metody, které pracují s některými entitami. Například CodeService obsahuje několik metod, které slouží k vygenerování unikátního vstupního kód nebo ke kontrole vstupného kódu.

[3] HTTPS - Hypertext Transfer Protocol Secure je zabezpečený internetový protokol HTTP
[4] HTTP - Hypertext Transfer Protocol je internetový protokol pro komunikaci s webovými servery

Třída s anotací @Repository poskytuje mechanismus provádět CRUD metody s objektem uloženým v databázi. Proto má každá entita, která se ukládá do databáze, svou vlastní třídu s anotací @Repositrory. S těmito třídami pracují i třídy s anotací @Service a @Controller.

Poslední třídou je třída s anotací @Controller nebo také @RestController, který slouží k obsluze požadavků, které mobilní aplikace posílá na server. Na základě zmapovaných cest se požadavek dostane ke konkrétní metodě ve třídě s anotací @Controller, kde je jeho tělo, pokud je přítomno, převedeno z JSON formátu na konkrétní entitu. Tato entita s hlavičkou požadavku je předána validační třídě, která rozhodne, zda je tento požadavek validní a nalezne data, která jsou potřebná pro danou metodu. Metoda zpracuje daný požadavek a odešle odpověď.

Pro zachycení dat poslaných z programu *Inkasník* slouží třída InkasnikController. Tato třída obsahuje metodu *import()*, která zpracovává požadavky zaslané z programu *Inkasník*. Metoda *import()* zkontroluje, zda požadavek obsahuje správný ověřovací token a následně pracuje pouze s tělem požadavku, které je ve formě textového XML[5] dokumentu. Příklad tohoto XML dokumentu můžete vidět níže. Tento dokument obsahuje několik elementů, které jsou v aplikaci reprezentovány třídami s výjimkou IČO. XML dokument je převede pomocí JAXBContext od Java objektu *Inkasnik*, který obsahuje proměnou cvn(IČO) a listy objektů, které reprezentují jednotlivé entity XML dokumentu. S většinou objektů se v aplikaci dále pracuje s výjimkou objektů Admin, Person a Service. Tyto objekty musí být nahrazeny objekty Caretaker, User a Deposit. Před uložením objektu do databáze se nejdříve ověří, zda se daný objekt v databázi již nenachází. Pokud tomu tak není, objekt je upraven a uložen do databáze. Při ukládání nového domovníka je vygenerován vstupní kód, který mu je poslán na e-mail, aby se mohl přihlásit do aplikace.

4.2. *Vstupní kód a token*

Každý požadavek, který mobilní aplikace pošle na server, musí obsahovat token, podle kterého server pozná, jaký uživatel daný požadavek poslal. Při přihlášení uživatele do aplikace server vygeneruje token, který odešle zpátky uživateli, a také nastaví exspiraci tokenu, která určuje, zda je token validní. Token exspiruje po třiceti minutách od posledního přijatého požadavku od uživatele. Při každém přijatém požadavku od uživatele je znovu nastavena i exspirace tokenu.

[5] XML - eXtensible Markup Language je jazyk určený pro výměnu dat mezi aplikacemi a pro publikování dokumentů.

Vstupní kód je tvořen z osmi náhodných znaků a slouží pro propojení obyvatele či domovníka s jeho domem. Tento kód rovněž generuje server, který jeho exspiraci nastaví na třicet dní od vygenerování. Po zadání kódu je nastaven jako neplatný a již se sním nelze znovu přihlásit. Domovníkovi, který se stará o dům, jehož data jsou spravována pomocí programu *Inkasník*, je vstupní kód zaslán e-mailem, který je vygenerován při prvním importu dat z programu *Inkasník*. Obyvateli či vlastníkovi bytu vstupní kód předá domovník, jenž jej získá ze serveru při zaslání žádosti o jeho vygenerování.

4.3. *Aplikace*

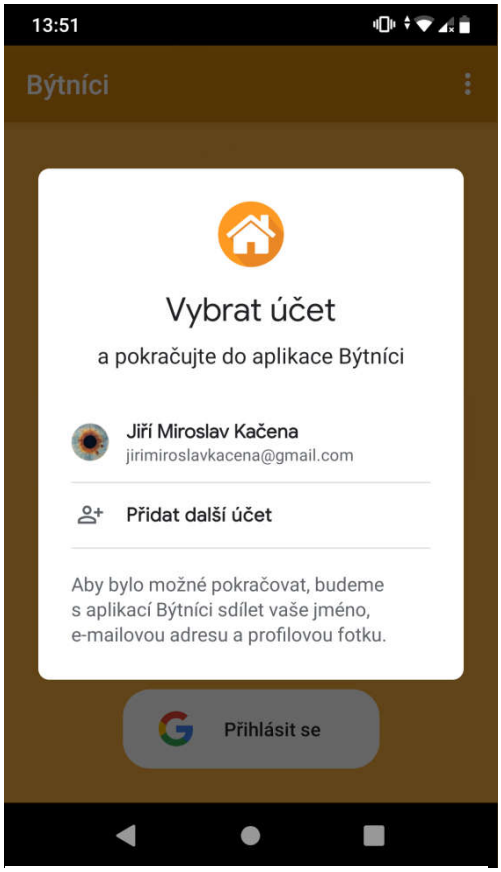
Aplikace je určena především pro uživatele z České republiky, kde je rozšířenějším mobilním operačním systémem operační systém Android. To byl jeden z hlavních důvodů, proč je aplikace vyvinuta právě pro operační systém Android. Aby aplikace fungovala i na starších zařízeních, byla vytvořena pro verze Android 5.0 (API level 21) a vyšší. Cílem bylo vyvinout uživatelsky přívětivou aplikaci, ve které se bude uživatel dobře orientovat, a proto má aplikace jednotný vzhled všech oken. Pro komunikaci se serverem aplikace používá klienta OkHttp[20], což je efektivní klient HTTP a HTTP/2 pro Android i Java aplikace. Komunikace mezi mobilní aplikací a serverem může být náročná jak časově, tak i na množství posílaných dat, proto při implementaci byla snaha co nejvíce tuto komunikaci omezit. Z tohoto důvodu aplikace většinu dat ukládá do mezipaměti telefonu, ale i tak se komunikace nedá úplně omezit.

4.3.1. *Role*

Aplikace pracuje s dvěma základními rolemi, a to Domovník a Obyvatel. Obyvatelem se uživatel může stát pouze po zadání vstupního kódu, který mu dal Domovník. Domovníkem se může uživatel stát buď zadáním vstupního kódu, který obdržel na e-mail nebo mu jej dal stávající domovník, či vytvořením nového domu v aplikaci. Pro každou z těchto rolí aplikace zobrazuje jiné funkce, které uživatel může využívat. Domovník má oproti obyvateli více pravomocí, jako například spravovat obyvatele či byty jednotlivých obyvatelů v daném domě. Na rozdíl od domovníka má obyvatel možnost prohlédnout si své finance (pokud jsou data o domě importovány program *Inkasník*). Pokud je uživatel obyvatelem či domovníkem více bytů či domů, je mu umožněno při přihlášení zvolit si roli a následně vybrat, do jakého domu, respektive bytu, chce být přihlášen. V aplikaci se objevují i další role, jako jsou Plátce bytu, Obyvatel bytu, Nájemník bytu a Vlastník bytu, ale na rozdíl od rolí Domovníka a Obyvatele jsou tyto role jen informativního charakteru.

4.3.2. *Přihlašování*

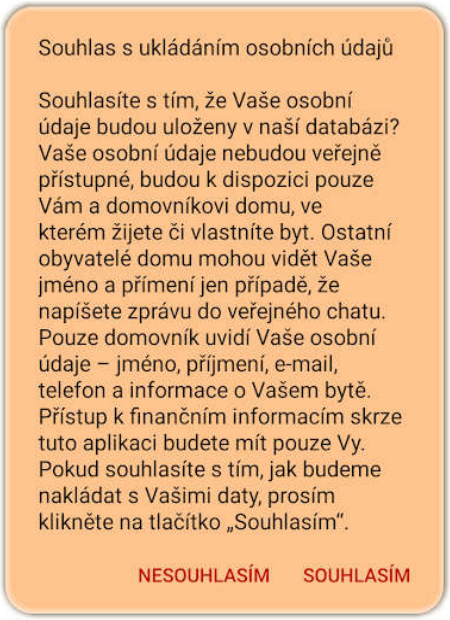
Pro bezpečné přihlášení do aplikace se uživatel musí přihlásit pomocí Google účtu[21]. To usnadní práci jak uživateli, tak i vývojáři. Jelikož je aplikace určena pouze pro zařízení s operačním systémem Android, tak lze předpokládat, že uživatelé účet od společnosti Google již mají. Uživatelé si tedy nemusí pamatovat další heslo, a zároveň jim aplikace umožňuje tiché přihlášení, kde pokaždé nemusí zadávat své přihlašovací údaje. Výhoda pro vývojáře je taková, že nemusí mít uložená uživatelská hesla a také nemusí kontrolovat, jestli je jejich heslo dostatečně silné. Samotné přihlášení začíná v mobilní aplikaci. Pokud uživatel již není přihlášen, musí se přihlásit pomocí Google účtu. Následně aplikace pošle požadavek o jednorázový autorizační kód na Google server, který, pokud je uživatel správně přihlášen, poskytne a aplikace jej přidá do požadavku o přihlášení, který následně odešle na server. Po obdržení požadavku sever odešle tento kód zpátky na Google server a ten mu vrátí uživatelovi údaje s uživatelovým identifikačním klíčem. Sever zjistí, jestli se v databázi vyskytuje uživatel s tímto identifikačním klíčem. Pokud ne, tak uživateli uloží a následně vygeneruje nový token a uživatelova data odešle do aplikace, kde s nimi aplikace dále pracuje.



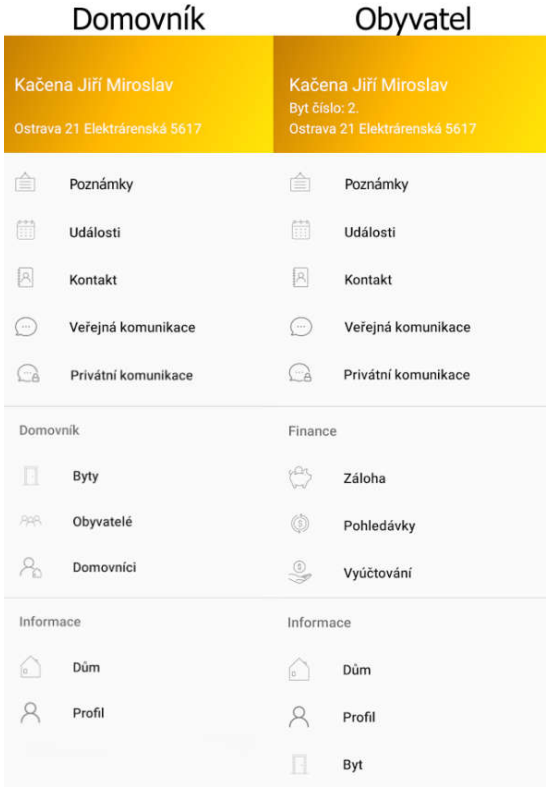
Obrázek 6 – Přihlašovací obrazovka

4.3.3. *První přihlášení do aplikace*

Uživateli, jenž se do aplikace přihlásí poprvé, je zobrazeno okno (viz. obr. 7), ve kterém nalezne vysvětlení, jak aplikace bude nakládat s jeho osobními údaji. Pokud uživatel nesouhlasí s tím, aby aplikace pracovala s jeho osobními údaji, aplikace odešle na server požadavek pro odstranění uživatele z databáze. Uživatel má také možnost kdykoliv v průběhu používání aplikace svůj účet odstranit. Pokud uživatel s podmínkami souhlasí, může změnit své údaje a následně si vybere ze tří akcí - zobrazit si informace o aplikaci, vytvořit nový dům a stát se domovníkem daného domu nebo zadat vstupní kód a spárovat se se svým bytem či domem. Pro vytvoření nového domu musí uživatel vyplnit formulář, ve kterém zadá údaje o domě. Aplikace následně pošle požadavek pro vytvoření nového domu na server, který dům vytvoří a nastaví uživatele jako domovníka daného domu. Poté aplikace přepne uživatele na hlavní obrazovku. Při zadávání kódu je kód odeslán na server, který kód zkontroluje a podle něj najde uživatelova data. Data spáruje s uživatelem a následně je pošle zpět do aplikace. Nakonec přepne uživatele do hlavní aktivity.



Obrázek 7 – Obrazovka se souhlasem



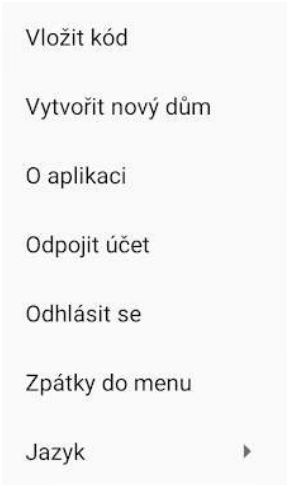
Obrázek 8 – Navigační menu

4.3.4. *Hlavní aktivita*

Hlavní aktivita je centrem celé aplikace. Skrze ni se může zobrazit mnoho různých fragmentů, ze kterých uživatel může vybírat v navigačním menu. To se skrývá na levé straně hlavní obrazovky a je možné jej vysunout poklepáním na ikonu v levém horním rohu nebo přejetím prstem z levého okraje obrazovky na střed. Jeho obsah se liší podle uživatelovy role (viz. obr. 8). Prvních pět položek mají všichni uživatelé stejné. Jedná se o položky, s jejichž pomocí spolu mohou obyvatelé domu komunikovat. Další společnou skupinou položek jsou Informace, které s výjimkou Bytu, který vidí pouze uživatelé s rolí obyvatel, vidí také všichni uživatelé. Zde si uživatelé mohou prohlédnout informace o daném domě, bytě a jich samotných, a pokud na to mají právo, mohou je i změnit. Skupinu položek Domovník vidí pouze domovník a s jejich pomocí může spravovat daný dům. Poslední skupinou položek v navigačním menu jsou Finance. Ty může vidět pouze obyvatel, jehož data byla importována programem *Inkasník*.

4.3.5. *Menu možností*

Menu možností neboli Option menu se vždy skrývá v právem horním rohu obrazovky. Při kliknutí na jeho ikonu se zobrazí menu možností. Obsah se mění podle aktivity či fragmentu, který je právě zobrazován. Například v aktivitě pro přihlašování se menu možností zobrazí pouze položky O aplikaci, Odhlásit, Odpojit účet a Jazyk. Vzhled menu možností v hlavní aktivitě můžete vidět na obrázku, které je obohaceno o možnosti Vložit kód, Vytvořit nový dům a Zpátky do menu.



Obrázek 9 – Menu možností

4.3.6. *Fragment*

Fragment[22] reprezentuje část nebo celé uživatelské rozhraní nějaké Activity i s příslušnými metodami. Díky tomu může být v hlavní aktivitě použito mnoho fragmentů, o které se stará třída FragmentFactory. V aplikaci jsou použity dva typy fragmentů. První zobrazuje seznam položek nějakého seznamu, například seznamu poznámek nebo seznamu kontaktů. Tento fragment většinou obsahuje i tlačítko pro přidání nové položky do seznamu. Druhý typ fragmentu se používá pro zobrazení jednotlivé položky ze seznamu. Při zobrazení položky ze seznamu, pokud má uživatel práva na úpravu položky, se v pravém horním rohu zobrazí tlačítka po editaci a odstranění položky. Tyto dva fragmenty na sebe většinou navazují, díky čemuž si uživatel může vybrat položku ze seznamu a tu následně zobrazit.

4.3.7. *Dialog*

Dialog je okno, které nevyplní celou obrazovku a může sloužit k různým účelům. V aplikaci se vyskytují tři typy dialogů, načítací dialog, rozhodovací dialog a dialog pro vkládání dat. Dialog pro načítání se zobrazí, když aplikace zpracovává, ukládá nebo načítá data. Rozhodovací dialog nutí uživatele vybrat si mezi dvěma možnostmi. Nejčastěji se jedná o potvrzení provedení nějaké akce, která se již nedá vrátit zpět. Tento dialog se proto zobrazí před odstraněním jakéhokoliv objektu. Dialog chrání před omylem spuštěnou akcí a taky uživatele informuje o důsledcích spuštěné akce. Poslední dialog, který je v aplikaci použit, slouží pro vkládání a úpravu informací o daném objektu. Tento dialog je složitější než předchozí dva, protože obsahuje mnohem více funkcionality, které se starají o zobrazování, vkládání, kontrolu a ukládání dat, proto je implementace dialogu pro vkládání dat řešena pomocí děděné třídy DialogFragment.

5. *Testování*

Pro vytvoření kvalitní aplikace je nutné ji otestovat. Aplikace by se měla testovat po celou dobu jejího vývoje a měly by se použít různé druhy testů. Testy nám pomáhají odhalit vady a nedostatky aplikace a také nám mohou ušetřit čas. Například, když na začátku vývoje pořádně otestujeme návrh aplikace, můžeme se tak vyhnou zbytečnému vývoji funkcí, které by se ve finální aplikaci nevyskytly. Proto byl návrh aplikace několikrát zkonzultován a poté přepracován, aby se předešlo zbytečnému vývoji.

Tato aplikace se skládá ze dvou částí. Serverové a aplikační. Obě tyto části je potřeba otestovat. Serverová část je testována pomocí jednotkových testů a integračních testů. Aplikační část je testována za pomoci uživatelů, kteří spadají do cílové skupiny aplikace.

5.1. *Testování serverové části*

Testy nám slouží k otestování funkčních celků. U jednotkových testů se většinou jedná o jednotlivé funkce a u integračních testů testujeme chování více funkcí najednou. Jelikož úkolem serverové části je práce s databází, většina testovaných funkcí pracuje s daty uložených v databázi, a proto je potřeba nejprve do databáze nahrát testovací data a následně funkce otestovat. Po ukončení testu je rovněž potřeba testovací data z databáze odstranit. Na straně serveru je třeba otestovat jednotlivé services a controllers.

Jednotlivé funkce v servise v zásadě neobsahují velké množství funkcionality, proto na jejich otestování postačí jednotkové testy. Na následujícím obrázku můžete vidět příklad jednotkového testu pro třídu UserService. Z důvodu přehlednosti jsou vynechány importy a také jsou uvedeny pouze tři testy pro třídu UserService. Na začátku testu funkce *setup()* nahraje do databáze testované data a na konci testu funkce *after()* tato data zase z databáze odstraní. V tomto případě jsou testovanými daty uživatel a byt, kteří jsou vzájemně propojeny. Následně jsou testovány tři funkce.

Funkci *isHisHouse()* vrací kladnou, nebo zápornou hodnotu, podole toho, je-li uživatel spojen s domem. Podobně tak funkce *isTokenValid()* zkontroluje, zdali je uživatelův token validní a podle toho vrátí kladnou, nebo zápornou hodnotu. Poslední funkce *splitName()* se používá při importu dat z programu *Inkasník* a jejím úkolem je rozdělit celé jméno na jméno a příjmení.

```
@RunWith(SpringRunner.class)
@SpringBootTest
public void UserServiceTest {
	@Autowired
	UserService userService;
	@Autowired
	ApartmentUserRepository apartmentUserRepository;
	@Autowired
	FlatRepository flatRepository;
	private User user;
	@Before
	public void setup() {
		Flat flat = new Flat(HOUSE_ID, FLAT_NUM);
		flatRepository.save(flat);
		FLAT_ID = flat.getFlatId();
		apartmentUserRepository.save(new ApartmentUser(USER_ID, FLAT_ID));
		user = new User(USER_ID);
		userService.saveUser(user);
	}
	@After
	public void after() {
		apartmentUserRepository.delete(apartmentUserRepository
				.findByUserIdAndFlatId(USER_ID, FLAT_ID));
		flatRepository.delete(FLAT_ID);
		userService.delete(user);
	}
	@Test
	public void isHisHouseTest() {
		assertThat(userService.isHisHouse(user, HOUSE_ID)).isTrue();
	}
	@Test
	public void isTokenInValidTest() {
		user.setToken("token");
		user.setExpirationToken(DateUtils.addMinutes(new Date(), 30)) ;
		assertThat(userService.isTokenValid(user)).isTrue();
	}
	@Test
	public void splitNameTest() {
		String name = "Kacana Jiri Miroslav";
		String expectedFirstName = "Jiri Miroslav";
		String expectedLastName = "Kacana";
		String actualFirstName = userService.splitName(name, true);
		String actualLastName = userService.splitName(name, false);
		assertThat(actualFirstName).isEqualTo(expectedFirstName);
		assertThat(actualLastName).isEqualTo(expectedLastName);
	}
}
```

Obrázek 10 – Kód pro testování třídy UserService

Na rozdíl od jednotkových testů, integrační testy se používají na testování více komponent. V našem případě jsou integrační testy použity pro testování controllerů, respektive jejich funkcí. Za pomoci Spring MockMvc[23], který slouží jako hlavní vstupní bod pro testování na jaře MVC, můžeme snadno otestovat controllery. V následující ukázce testu funkce *saveNote()* je na začátku vytvořena poznámka, která je následně podstrčena požadavku POST /Note. Server požadavek zkontroluje, poznámku uloží do databáze a následně poznámku obohacenou o nový identifikační klíč vrátí zpět. Následně testujeme, zdali byla poznámka řádně uložena a nakonec poznámku z databáze odstraníme.

```
@Test
public void testSaveNote() throws Exception {
        Note expectedNote = new Note(HOUSE_ID, USER_ID, NAME, TEXT, DATE);
        MockHttpServletRequestBuilder builder =
                MockMvcRequestBuilders
                        .post("/Note")
                        .header("token", USER_TOKEN)
                        .contentType(MediaType.APPLICATION_JSON)
                        .content(gson.toJson(expectedNote)) ;
        ResultActions resultActions = this.mockMvc
                    .perform(builder)
                    .andDo(MockMvcResultHandlers.print())
                    .andExpect(MockMvcResultMatchers.status()
                        .isOk());

        MvcResult result = resultActions.andReturn();
        String contentAsString = result.getResponse().getContentAsString();
        Note actualNoteReturn = gson.fromJson(contentAsString, Note.class);
        assertThat(actualNoteReturn).isNotNull();
        Note actualNote = noteRepository.findByNoteId(actualNoteReturn.getNoteId());
        assertThat(actualNote).isNotNull();

        assertThat(actualNote.getHouseId()).isEqualTo(HOUSE_ID);
        assertThat(actualNote.getCreator()).isEqualTo(USER_ID);
        assertThat(actualNote.getName()).isEqualTo(NAME);
        assertThat(actualNote.getText()).isEqualTo(TEXT);

        noteRepository.delete(actualNote) ;
}
```

Obrázek 11 – Funkce pro testování funkce *saveNote()*

5.2. *Testování mobilní aplikace*

Mobilní aplikace se testovala pomocí uživatelských testů, které pomohly odhalit problémy s používáním aplikace. Testování se zúčastnili tři uživatelé, kteří procházeli různé testovací scénáře. Každému uživateli jsme vysvětlili jejich roli a cíl, kterého by měli dosáhnout. Následně byli uživatelé ponecháni, aby sami provedli daný scénář. V průběhu testu jsme uživatelům neposkytli žádnou radu.

Prvním testujícím byl muž ve věku 57 let, který má dobré zkušenosti s ovládáním chytrého mobilního telefonu. Jeho role byla Domovník, jehož dům je spravován pomocí programu *Inkasník* a jeho cílem bylo se přihlásit do již vytvořeného domu a provést zde několik změn. Testujícímu byl předložen následující scénář.

1. Přihlásit se do aplikace a vytvořit svůj účet.
2. Zadat kód přihlašovací kód „rk5sot1s“.
3. Změnit jméno u obyvatele „Černý Jiří“ na jméno „Modrý Jan“.
4. Nastavit obyvatele „Žlutý Jan“ jako domovník.
5. Vygenerovat přihlašovací kód pro obyvatele „Oranžový Josef“.
6. Přidat obyvatele „Oranžový Josef“ do bytu číslo 2.
7. Vygenerovat přihlašovací kódy pro všechny obyvatele.
8. Odebrat obyvatele „Oranžový Josef“ z bytu číslo 1.
9. Odebrat poznámku „Upozornění pro vlastníky aut“.
10. Napsat zprávu bytu číslo 6.

Druhým testujícím byla žena ve věku 28 let, která má dobré zkušenosti s ovládáním chytrého mobilního telefonu. Její role byla domovník a jejím cílem bylo vytvořit nový dům a provést v něm několik operací. Testujícímu byl předložen následující scénář.

1. Přihlásit se do aplikace.
2. Vytvořit nový dům.
3. Přidat nový byt číslo 1.
4. Vytvořit nového uživatel „Jan Modrý“ v bytu číslo 1.
5. Vygenerovat vstupní kód pro uživatele „Jan Modrý“.
6. Přidat uživatele „Jan Modrý“ do nového bytu číslo 2.
7. Vygenerovat všechny vstupní kódy.
8. Odebrat uživatele „Jan Modrý“ z bytu číslo 2.
9. Odstranit uživatele „Jan Modrý“.

Posledním testujícím byl muž ve věku 41 let, který má základní zkušenosti s ovládáním chytrého mobilního telefonu. Jeho role byl obyvatel a jeho cílem bylo se přihlásit do domu a provést zde několik operací. Testujícímu byl předložen následující scénář.

1. Přihlásit se do aplikace a vytvořit svůj účet.
2. Zadat kód přihlašovací kód „sd84wa5q“.
3. Vytvořit novou poznámku.
4. Upravit poznámku.
5. Vytvořit novou událost.
6. Vytvořit nový kontakt.
7. Zavolat na číslo uvedené kontakt.
8. Upravit informace o sobě.
9. Odhlásit se z aplikace.

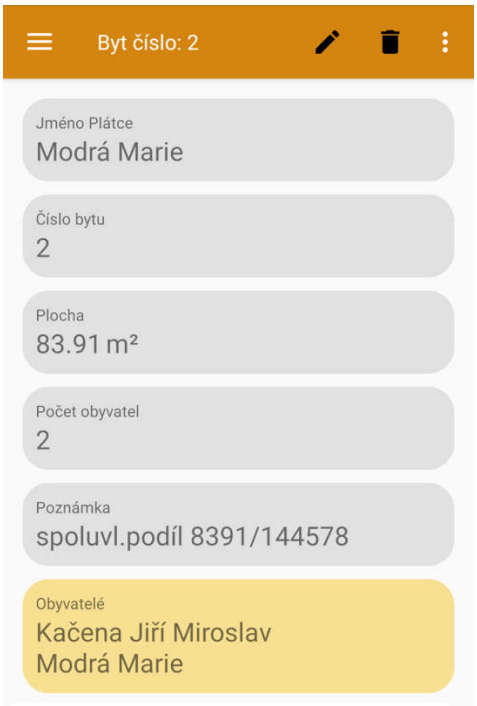
Všichni testující bez větších komplikací zvládli projít scénář, avšak se zde objevilo několik problémů. Většina těchto problému byla způsobena tím, že si testující na začátku nepřečetli informace o aplikaci. Tyto problémy byly z části odstraněny, ale pro úplné vyřešení problémů s neznalostí funkcí v aplikaci bychom museli vyvinout průvodce aplikací. Průvodce by se spustil při prvním přihlášení do aplikace a ukázal by uživateli, jak s aplikací správě pracovat.

Třetí testující měl problém nalézt položku „Profil“ ve skrytém menu, která je poslední v seznamu a pro její zobrazení uživatel musel srolovat až na konec menu. Uživatel si totiž na začátku nepřečetl informace o aplikaci, a proto nevěděl, že se dá menu rolovat, tudíž nemohl položku nalézt. Nakonec však položku našel. Problém byl z části vyřešen úpravou informací o aplikaci.

Všichni testující se také museli vypořádat s následujícím problémem: při úpravě jakéhokoliv objektu testující nejprve klikli na zobrazené informace, místo aby klikli na tlačítku ve tvaru tužky, které je k tomu určené. Tento problém jsme neodstranili především proto, že testující sice byli pozdrženi při první úpravě, avšak následně nalezli tlačítko, které je pro danou operaci určené.

Posledním problém, který objevili testující v roli domovníků, byl spjat s funkcí pro zobrazení obyvatel v daném bytě. Pro jeho zobrazení je nutné kliknout na položku Obyvatelé v informacích o bytu. Jelikož měla položka stejnou barvu jako ostatní, testující nevěděli, že se dá na položku kliknout. Proto byl seznam s obyvateli bytu barevně odlišen od ostatních informací (viz. obr. 12).

Testující také upozornili na pár gramatických chyb a na špatně zobrazený text. I tyto chyby byly odstraněny.



Obrázek 12 – Obrazovka pro zobrazení informací o bytu

6. Závěr

Zadání bakalářské práce bylo splněno. Navrhl a vytvořil jsem víceuživatelskou aplikaci pro komunikaci v bytovém domě, ve které si někteří uživatelé mohou zobrazit své finanční informace. Také jsem navrhl a vytvořil serverovou část s databází. Vývoj této aplikace byl náročný, jelikož jsem na začátku měl jen malé zkušenosti s vývojem mobilní aplikace, ale díky tomu jsem se seznámil a naučil pracovat s novými technologiemi, které jsem dříve neznal.

V první fázi vývoje aplikace jsem vyhledal a zanalyzoval existující aplikace a vytvořil jsem první návrh své aplikace, který jsem dále vylepšoval. Následně jsem se seznámil s potřebnou technologií pro vytvoření aplikace. Nejvíce času mi zabralo seznámit se s vývojem Spring Boot aplikace, jelikož jsem s ní neměl žádné zkušenosti. Také jsem se seznámil se způsobem komunikace mobilní aplikace se serverem přes zabezpečený protokol HTTPS. Další fází vývoje aplikace bylo vytvoření vhodného grafického rozhraní, aby bylo co nejvíce intuitivní a vyhovovalo všem uživatelům. Následně jsem vyvíjel mobilní aplikaci i serverové části zároveň tak, aby spolu vzájemně správně komunikovali. V průběhu vývoje jsem obě tyto části testoval, přepracovával a vylepšoval.

Když byla aplikace hotová, provedl jsem testování aplikace na uživatelích. Díky těmto testům jsem odhalil několik chyb v systému, které jsem odstranil. Z uživatelských testů jsem také zjistil, že před nasazením aplikace, by bylo dobré vytvořit nějakého průvodce aplikací, kde by uživatelé zjistili, jak s aplikací správně pracovat.

Aplikace je vytvořena tak, aby se dala do budoucna dále vylepšovat. Mezi možné zlepšení do budoucna patří například vylepšení komunikace aplikace se serverem, nebo přidání notifikací. Dále je možné přidat další funkce, které by uživatelé ocenili, například anketu, kde by mohli obyvatelé hlasovat, nebo kalendář prací a služeb, kde by se mohli obyvatelé domu například domlouvat na tom, kdo má tento měsíc na starost vynášení popelnic před dům nebo kdo má na starost úklid domu. Dalším zlepšením by mohla být úprava grafického rozhraní tak, aby aplikace lépe vyhovovala uživatelům vlastnícím tablet.

7. Zdroje

1. **Jirra Software** [online] [cit. 08.12.2019]. Dostupné z: https://jirra.cz/

2. **Domsys** [Online] [Citace: 16.12.2019] Dostupné z: https://www.domsys.cz/

3. **Domsys Portál** [online] [cit. 16.12.2019]. Dostupné z: https://play.google.com/store/apps/details?id=eu.domsys.Board

4. **Domsys Správce** [online]. [cit. 16.12.2019]. Dostupné z: https://play.google.com/store/apps/details?id=eu.domsys.Management&hl=en_US

5. **DOMUS** [online]. Anasoft [cit. 16.12.2019]. Dostupné z: https://www.anasoft.com/domus/cz/

6. **eDOMUS** [online]. Anasoft [cit. 16.12.2019]. Dostupné z: https://www.anasoft.com/domus/cz/home/eDOMUS/

7. **Odpočet měřičů – Poschodech.** [Online] [Citace: 16.12.2019] Dostupné z: https://play.google.com/store/apps/details?id=com.anasoft.domus.odpocet

8. **GreenApps,** 2016 [online]. [cit. 17.12.2019]. Dostupné z: https://www.greenapps.com/

9. **Pavel Šiška,** *SVJO – Společenství vlastníků jednotek online.* [Online] [Citace: 3. Ledna 2020] https://www.svjo.cz/

10. **Spoluvlastníci.** [Online] [17.12.2019] https://www.spoluvlastnici.cz/

11. **Ema plus lam.** [Online] [17.12.2019]https://lam.emaplus.cz/

12. **LACKO, Ľ.** *Mistrovství - Android.* Computer Press, 2017. 648 p. ISBN 978-80-251-4875-4.

13. **WALLS, C.** *Spring Boot in Action.* Manning Publications, 2015. 264 p. ISBN 978-16-172-9254-5.

14. **Apache Tomcat,** 1999 [online] [cit. 21.12.2019]. Dostupné z: http://tomcat.apache.org/

15. **Richard T. Watson.** *Data Management: Database and Organizations.* John Wiley & Sons, 2003

16. **SCHILDT, H.** *Mistrovství - Java.* Computer Press, 2014. 1224 p. ISBN 978-80-251-4145-8.

17. **BIEHL, M.** *Restful Api Design*. CreateSpace Independent Publishing Platform, 2016. 294 p. ISBN 978-15-147-3516-9.

18. **SRIPARASA, S. S.** *JavaScript and JSON Essentials*. Packt Publishing, 2013. 120 p. ISBN 978-17-832-8603-4.

19. **Spring Boot.** [Online] [Citace: 28.12.2019] https://spring.io/projects/spring-boot

20. **OkHttp.** [Online] [Citace: 28.12.2019] Dostupné z: https://square.github.io/okhttp/

21. **Google Developers.** *Integrating Google Sign-In into your web app.* [Online] [Citace: 28.12.2019] Dostupné z: https://developers.google.com/identity/sign-in/web/sign-in

22. **Matěj Konečný.** *Dej Androidu tablety! 2*012 [Online] [Citace: 28.12.2019] Dostupné z: https://www.zdrojak.cz/clanky/dej-androidu-tablety/

23. **Lokes Gupta.** *Spring boot MockMVC example.* [Online] [Citace: 3. Ledna 2020] Dostupné z: https://howtodoinjava.com/spring-boot2/testing/spring-boot-mockmvc-example/

8. Přílohy

Android

- zdrojové kódy BPApp
- instalační soubor app.apk
- README.txt

Server

- zdrojové kódy bp-tomcat-application
- vygenerovaný soubor bp-tomcat-application.war
- README.txt

Dokumenty

- bakalářská práce (PDF)
- bakalářská práce (DOCX)