Bakalářská práce

České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

# Aplikace pro správu ubytovacích zařízení

Martin Sebera

Vedoucí: doc. Ing. Ivan Jelínek, CSc.
Obor: Software
Studijní program: Otevřená informatika
Květen 2019

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Sebera** Jméno: **Martin** Osobní číslo: **466120**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Software**

## II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

**Aplikace pro správu ubytovacích zařízení**

Název bakalářské práce anglicky:

**Application for the Management of Accommodation Facilities**

Pokyny pro vypracování:

1. Proveďte rešerši současných aplikací pro správu ubytoven
2. Navrhněte strukturu databáze, webového API a mobilní aplikace pro správu ubytoven
3. Implementujte všechny tři součásti systému z bodu 2.
4. Navrhněte systém úspory přenosu dat a synchronizace
5. Navrhněte a implementujte vhodný způsob zabezpečení
6. Navrhněte uživatelský interface mobilní aplikace a funkcionalitu pro registraci uživatelů na webu
7. Navrhněte vhodný způsob testování a aplikaci otestujte, výsledky vyhodnoťte

Seznam doporučené literatury:

[1] https://kotlinlang.org/
[2] https://www.tutorialspoint.com/asp.net_core/
[3] https://stackoverflow.com/
[4] https://developer.android.com/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

**doc. Ing. Ivan Jelínek, CSc., kabinet výuky informatiky FEL**

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

Datum zadání bakalářské práce: **22.01.2019** Termín odevzdání bakalářské práce: **24.05.2019**

Platnost zadání bakalářské práce: **20.09.2020**

| doc. Ing. Ivan Jelínek, CSc.<br>podpis vedoucí(ho) práce | podpis vedoucí(ho) ústavu/katedry | prof. Ing. Pavel Ripka, CSc.<br>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 |
|---|---|

# Abstrakt

Tato bakalářská práce si klade za cíl vytvořit mobilní aplikaci pro správu ubytovacích zařízení. Hlavním účelem bude organizace příjezdů a odjezdů. Cílovou skupinou jsou menší a rodinné podniky. Jedná se o komplexní softwarový projekt zahrnující mobilní klientskou aplikaci a centrální server s databází

**Klíčová slova:** ubytovna, ubytovací zařízení, android

**Vedoucí práce:** doc. Ing. Ivan Jelínek, CSc.

# Abstract

This BA project aims to create a mobile application for management of accommodation businesses. The main function is the organization of check-in and check-out with a simple payments register also available. The application is intended for smaller and family businesses. It is a complex software project which includes a mobile application and a central server with database.

**Keywords:** hostel, accommodation, android

**Project supervisor:** doc. Ing. Ivan Jelínek, CSc.

# Prohlášení a poděkování

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

V Praze, dne 17.5.2019

Chtěl bych poděkovat panu doc. Ing. Ivanu Jelínkovi, CSc. za vedení mé práce. Děkuji také všem, kteří mě během studia jakkoli podporovali.

# Obsah


Abstrakt ..... 4

Abstract ..... 4

Prohlášení a poděkování ..... 5

1. Úvod ..... 8

1.1. O systému ..... 8

1.2. Cíl a přínos aplikace ..... 8

1.3. Popis vývoje projektu ..... 8

2. Analýza ..... 8

2.1. Operační systém android ..... 8

2.2. Rešerše existujících aplikací ..... 9

Previo ..... 9

Bananadesk ..... 10

Systém Bananadesk není český produkt, ani se nespecializuje na český trh. Zaregistrovat se mohou uživatelé z celého světa. K dispozici je dvoutýdenní verze zdarma, díky čemuž mohla proběhnout rešerše. Bananadesk bude rozebrán detailněji, neboť je jeho návrh podobnější návrhu ubysysu. ..... 10

2.3. Kvalitativní požadavky ..... 12

2.4. Funkční požadavky ..... 12

2.5. Použité jazyky a technologie ..... 13

Kotlin vs Java ..... 13

C# a ASP.NET Core ..... 14

MS SQL ..... 14

3. Návrh ..... 15

3.1 architektura systému ..... 15

3.2. Vznik návrhu ..... 15

3.2.1. Nálezy během formování návrhu a jejich řešení ..... 16

3.3. Organizační záležitosti ..... 16

3.3.1. seskupování hostů do skupin ..... 16

3.3.2. specifikace organizace ubytoven ..... 16

3.3.3 role agentur v provozu ubytovacích zařízení ..... 17

3.3.4. jednoduchá evidence plateb ..... 17

4. Implementace ..... 17

4.1. server ..... 17

4.1.1. struktura serveru ..... 17
4.1.2. kotrolery ..... 17
4.1.3. Databázové tabulky ..... 18
4.1.4. formát requestu ..... 21
4.1.5. kontrola requestu ..... 21
4.1.6. Formát response ..... 22
4.1.7. Version vector ..... 22
4.1.8. logování chyb ..... 22
4.1.9. Další zabezpečení ..... 22
4.1.10. Příklad těla metody služby poskytované serverem ..... 22
4.2. klientská aplikace ..... 23
4.2.1. struktura klientské aplikace ..... 23
4.2.2. Webové API – webový klient ..... 24
4.2.3. Webové API – přijímání odpovědí ze serveru ..... 25
4.2.4. Webové API – services ..... 25
4.2.5. Webové API – cache namísto stahování dat ..... 27
4.2.6. specifika registrace a přihlašování ..... 28
4.3. Popis a ukázka funkcionalit ..... 28
4.3.1. Kalendář ubytovacího zařízení ..... 28
4.3.2. Registrace/editace/smazání skupiny ..... 28
4.3.3. Vytvoření/editace/smazání ubytovacího zařízení ..... 29
4.3.4. Registrace/editace/smazání agentury ..... 29
4.4. uživatelské testování ..... 29
4.4.1. seznam úloh ..... 29
4.4.2. průběh testování ..... 30
4.4.3. nálezy a jejich řešení ..... 30
5. Závěr ..... 31
Citovaná literatura ..... 33
Odkazy ..... 33
Příloha A. Obrazovky aplikace ..... 34

# 1. Úvod

## 1.1. O systému

Cílem práce je vytvoření funkčního systému pro správu příjezdů a odjezdů v ubytovacím zařízení. Projekt dostal pracovní název Ubysys (zkratka ubytovací systém). Ubysys je komplexní softwarový projekt zahrnující centrální server s databází a mobilní aplikaci pro android. Hlavní úkol je správa příjezdů a odjezdů skupin, dále je umožněno evidovat platby a agentury, s nimiž ubytovna spolupracuje. Ubysys je jednodušší alternativa k již zavedeným systémům.

## 1.2. Cíl a přínos aplikace

Systém Ubysys si klade za cíl poskytnout jednoduché rozhraní pro správu ubytoven, přičemž se od známých systémů liší v několika rysech. Už samotná specializace na ubytovny je netypická, neboť většina podobných systémů uzpůsobuje svou organizaci hotelovým zařízením a penzionům. Oproti jiným systémům se Ubysys nezajímá o obsazenost konkrétních postelí na pokojích, ale pouze o obsazenost celého pokoje. To znamená, že systém ubytuje skupinu hostů, pokud pro ně existují volné postele, ale už neříká, které postele to jsou. Díky tomuto zjednodušení je možné bez dalšího přemýšlení využít kapacitu pokojů naplno a již není potřeba starat se o efektivní využití lůžek. Samozřejmě to s sebou přináší jednu nevýhodu – po odjezdu hostů nikde není psáno, které konkrétní lůžko se má uklidit. Dle slov uživatelů však toto není problém.

Systém je určen spíše pro menší a rodinné podniky. Umožňuje spravovat více ubytoven (limit je dán specifikací licence). Je dovoleno, aby jedna ubytovna měla více manažerů, ovšem majitel licence je považován za hlavního manažera. Je implementováno i zaměstnávání recepčních formou posílání pozvánek k registraci na email.

## 1.3. Popis vývoje projektu

Výběr právě tohoto tématu byl inspirován brigádou na hotelu a tím, že osobně znám několik provozovatelů různých typů ubytovacího zařízení. Projekt začal analýzou, při které bylo rozhodnuto, pro jaký operační systém se vytvoří a udělal se průzkum existujících řešení, ze kterých se dala načerpat inspirace. Během navrhování systému proběhly konzultace s provozovateli hotelu a ubytoven, což pomohlo formování požadavků apod. Při implementaci se uživatelé nevyjadřovali až do finálních uživatelských testů, které odhalily několik nedostatků.

# 2. Analýza

## 2.1. Operační systém android

Systém android je v současnosti nejrozšířenější na trhu. Za jeho vývoj zodpovídá společnost Google, která na něm aktivně pracuje. (1) Vývojářům může nabídnout perfektní detailní online dokumentaci a množství vývojářských nástrojů (oficiální web v roce 2019 byl *developer.android.com/reference*). V dnešní době se k vývoji nejčastěji využívá Android Studio (2), které je vlastně vyseknutý modul z

programu IDEA fungující jako samostatné IDE. Android Studio je vyvíjeno společně se systémem Android, takže podporuje vývoj aplikací na všech API levelech.

Pro vývoj aplikací poskytuje systém android API využívající platformu JVM. Existuje mnoho verzí API (tzv. API levels). Dnes máme již level 28. (3) Nicméně systém ubysys bude vyvíjen na levelu 25 kvůli kompatibilitě se staršími mobilními telefony.

Zajímavostí je, že již máme na výběr více programovacích jazyků než jen Java. V současnosti je velmi populární kotlin, který vychází z Javy a kompiluje se pro do stejného bytecodu jako java. Nabízí však mnohem příjemnější syntaxi a je technologicky vyspělejší. (4)

## 2.2. Rešerše existujících aplikací

Zde se text věnuje analýze existujících systémů. Rešerše byla zaměřena na hotelové i hostelové systémy, mezi nimiž hledá rozdíly a podobnosti. Některé vlastnosti těchto zavedených systémů budou naimplementovány i do systému ubysys.

### Previo

Hotelový systém Previo používalo na jaře 2019 cca 1750 ubytovatelů. (5) Je velmi populární a propracovaný. Je to původně český produkt (6) a na první pohled je jasné, že se na český trh i specializuje, neboť obsahuje funkce určené pro reportování českému statistickému úřadu či cizinecké policii České republiky (7). Za zmínku stojí i fakt, že se nejedná jen o hotelový systém, ale nabídka softwarových produktů a služeb je mnohem širší. Za další příplatky je možno využívat i restaurační systém, rezervační systém (ten je napojen na služby booking.com aj,), systém na hlášení úklidu a poruch, platební bránu a různé marketingové nástroje.

Většina běžných uživatelů však zná previo jen jako hotelový systém. Touto částí se může inspirovat i tento projekt, ale spíše jen okrajově. Previo se zaměřuje na hotely, které mají jinou organizaci příjezdů. Previo navíc obsahuje např. i funkce pro inventuru drobného prodeje, pokročilé účetnictví, různé reporty apod., což nebude v tomto projektu implementováno.


**Obrázek 2.1:** Hotelový systém previo

Ačkoliv existuje i mobilní aplikace previo, na hotelech je obvykle k vidění webová verze pro desktop. Obrázek ukazuje hlavní obrazovku. Uprostřed můžeme vidět kalendář skupin. Dá se scrollovat na obě strany libovolně daleko. Tato příjemná funkcionalita bude implementována i v systému Ubysys. Vlevo je panel na přepínání pater, což je skvělá funkce do velkých hotelů. Pro ubytovny a zejména rodinné podniky s malým počtem pokojů je taková funkce zbytečná.

Kalendář previa je velmi barevný kvůli rozlišování stavů rezervací. Jsou to: opce, potvrzeno, ubytován, ukončeno, jiné, waiting list, storno. Rezervace ve stavu no-show se nezobrazuje v kalendáři, ale stále je dohledatelná v archivech.

Previo má komplexní evidenci hostů. Každý, kdo se někdy ubytovával, musí mít záznam v databázi. Kvůli předpisům je potřeba evidovat jméno, příjmení, číslo a typ dokladu, národnost apod. Dá se vyplnit však mnohem více údajů. Např. SPZ auta a titul před i za jménem. Host se dá označit různými štítky – např. alergik, problémový host, vegetarián, sportovec apod. Samozřejmostí je dodatečná poznámka.

Rezervace se dělí na dva typy – individuální a skupinovou. To má svůj základ v účetnictví, neboť se často stává, že se pro každý pokoj vystavuje účtenka zvlášť. Ve skupinové rezervaci může mít každý pokoj jiný datum příjezdu a odjezdu a dají se jim individuálně prodlužovat či zkracovat pobyty. Hosté se mohou ubytovat i nad kapacitu pokoje, ale potom nemají lůžko. Tato funkce se v praxi používá spíše zřídka.

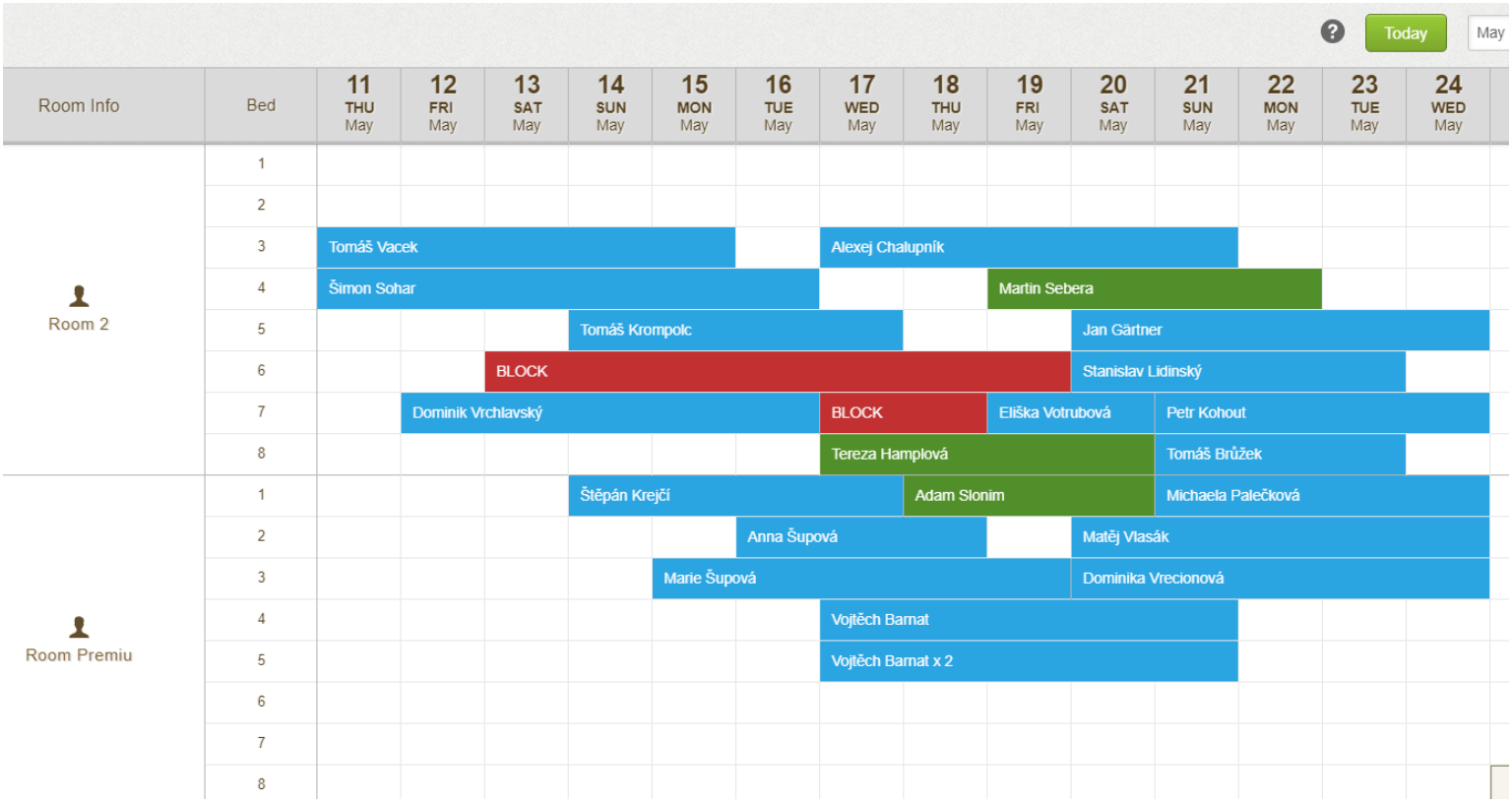
V rezervaci je možno pro každého hosta nastavit režim stravování – bez stravy, snídaně, polopenze, plná penze. Kvůli městským poplatkům se i rozlišuje, zda se jedná o služební cestu či rekreaci.

Tento systém se dá pro všechny provozovatele hotelu určitě doporučit. Má intuitivní ovládání, ale je potřeba naučit se některé pojmy (individuální vs. skupinová rezervace apod.). Jeho funkce pro reportování statistickému úřadu a cizinecké policii ušetří spoustu práce. Hodí se i jeho modul pro rezervace, který je napojen na služby booking.com a další. Jedná se prostě o velmi komplexní produkt. Je složitější tento systém pochopit a efektivně využívat, nicméně poté nabízí řešení pro všechny běžné problémy.

### Bananadesk

Systém Bananadesk není český produkt, ani se nespecializuje na český trh. Zaregistrovat se mohou uživatelé z celého světa. K dispozici je dvoutýdenní verze zdarma, díky čemuž mohla proběhnout rešerše. Bananadesk bude rozebrán detailněji, neboť je jeho návrh podobnější návrhu ubysysu.

Systém Bananadesk poskytuje více modifikací – pro hotely, ubytovny a jiné. Zdá se ale, že primární zaměření jsou ubytovny.




Obrázek 2.2: Systém bananadesk pro ubytovny

Na první pohled je vidět, jak se bananadesk od previa nápadně liší. Vidíme, že pokoje jsou rozděleny na lůžka. Systém ani neumožňuje v případě poruchy zablokovat celý pokoj, ale dají se zablokovat konkrétní lůžka (na obrázku červenou barvou). Pro zablokování celého pokoje je nutno zablokovat všechna lůžka. Chybí možnost uzavření na dobu neurčitou, což opět hodnotím jako chybu.

Správa pokojů není tak komplexní jako u previa, ale stále je možno nastavit spoustu vlastností. Pokoje se seskupují do typů, kde každý typ má daný počet lůžek, cenu a zda je mužský či ženský. Přitom u hostů se již pohlaví nerozlišuje. Tento rozpor je překvapivý. Zajímavostí je, že cena se dá nastavit pro každý den v týdnu jiná, přičemž se dá ještě upravit pro konkrétní data. Tím se dá automaticky vypočítat cena pobytu, avšak přináší to i pár nevýhod. Pokud bude cena pro nějaký den změněna, změní se i cena již vyřízených rezervací. Dle slov uživatele je navíc časté, že každému hostu dávají jiné ceny. Proto bylo rozhodnuto, že v systému ubysys nebude tato funkcionalita implementována.

Oproti previu došlo ke zjednodušení v evidování hostů. Zapisuje se jen jméno, datum narození, adresa bydliště, národnost a číslo a typ dokladu. Chybí informace o pohlaví, automobilu, vízu, nejde označovat hosta pomocí štítků (např. vegetarián, problémový host apod.) a není možnost připsat poznámku, což hodnotím jako chybu, neboť je často potřeba připojit doplňující informace. Databáze hostů je taková „volnější“.

Stejně jako v previu se hosté mohou seskupovat do skupin. Protože se však jedná o ubytovnu, je možné, aby každý host přijel a odjel jindy, dají se jim individuálně prodlužovat pobyty apod. Rezervace mohou mít dva stavy potvrzení – potvrzeno/nepotvrzeno a dva stavy uhrazení – zaplaceno/nezaplaceno. Oproti previu počet stavů ubyl.

GUI systému působí oproti previu mnohem jednodušeji. To by byla spíše výhoda, kdyby bylo stejně tak dobře propracované. GUI previa obsahuje spoustu popup oken a většina komunikace se serverem je asynchronní javascript, tudíž není prakticky nikdy potřeba načítat znovu celou stránku. Bananadesk častěji provádí reload celé stránky, což zhoršuje uživatelský zážitek, prodlužuje odezvy apod.

Tento systém není dokonalý, ale přesto se dá doporučit. Má intuitivní ovládání a jednoduchý návrh. Uživatelé by jistě uvítali více popup oken, psaní poznámek k hostům a možnost blokování celého pokoje

## 2.3. Kvalitativní požadavky

Tato funkcionalita vznikla na základě požadavků reálného uživatele.

**NFR1: Různá oprávnění uživatelů**
Je potřeba rozlišit majitele ubytovny a jeho spolumajitele, jež mají téměř stejná oprávnění (dále jen manažer) a běžného zaměstnance, např. recepčního (dále jen zaměstnanec)

**NFR2: Podpora operačních systémů**
Systém bude postaven na android API verze 25, aby systém běžel i na starších telefonech

## 2.4. Funkční požadavky

Jedná se o funkcionalitu, kterou systém poskytuje a která byla inspirována již existujícími systémy a potřebami reálného uživatele.

**FR1: Registrace příchozí skupiny**
Systém umožní manažerovi a zaměstnanci registrovat novou skupinu.

**FR2: Editace skupiny**
Systém umožní manažerovi a zaměstnanci editovat ubytovanou skupinu.

**FR3: Smazání skupiny**
Systém umožní manažerovi a zaměstnanci smazat ubytovanou skupinu

**FR4: Zobrazení kalendáře ubytovny**
Systém umožní manažerovi a zaměstnanci procházet grafické zobrazení příjezdů a odjezdů)

**FR5: Vytvoření nové ubytovny**
Systém umožní manažerovi vytvořit novou ubytovnu a nastavit její pokoje, pokud jeho limit pro počet ubytoven ještě nebyl vyčerpán.

**FR6: Editace ubytovny**
Systém umožní manažerovi editovat ubytovnu a její pokoje

**FR7: Smazání ubytovny**
Systém umožní manažerovi smazat ubytovnu. Provede se pouze soft delete.

**FR8: Registrace nové agentury**
Systém umožní manažerovi a zaměstnanci registrovat novou agenturu

**FR9: Editace agentury**
Systém umožní manažerovi a zaměstnanci editovat agenturu

**FR10: Smazání agentury**
Systém umožní manažerovi smazat agenturu. Provede se pouze soft delete.

**FR11: Pozvání zaměstnance ke spolupráci přes email**
Systém umožní manažerovi poslat pozvánku ke spolupráci na daný email.

**FR12: Pozvání spolumanažera ke spolupráci přes email**
Systém umožní manažerovi poslat pozvánku ke spoluřízení ubytovny na daný email.

**FR13: Registrace přes pozvání ke spolupráci**
Systém umožní příjemci pozvánky ke spolupráci zaregistrovat se do systému Ubysys. Vznikne nový uživatelský profil, který nemá vlastní licenci, ale je napojen na licenci svého zaměstnavatele (nebo spolumanažera)

**FR14: Zrušit dříve uzavřenou spolupráci**
Systém umožní manažerovi zrušit platnost profilu zaměstnance, který se dříve registroval přes pozvání ke spolupráci

**FR15: Přidělení práv zaměstnanci k práci na ubytovně**
Systém umožní manažerovi přidělit zaměstnanci oprávnění pracovat na dané ubytovně (registrovat tam skupiny, editovat je, evidovat platby apod.)

**FR16: Odebrání práv vykonávat práci na ubytovně**
Systém umožní manažerovi odebrat zaměstnanci práva k práci na dané ubytovně.

**FR17: Registrace nového hosta do databáze hostů**
Systém umožní manažerovi a zaměstnanci registrovat nového hosta do databáze hostů

**FR18: Vyhledávání hosta dle výrazu**
Systém umožní manažerovi a zaměstnanci vyhledávat hosta dle příjmení, jména a roku narození

## 2.5. Použité jazyky a technologie

### Kotlin vs Java

Pro vývoj mobilní aplikace byl vybrán jazyk kotlin kvůli jeho příjemné syntaxi a vyspělosti. Jeho vlastnosti umožňují rychlejší vývoj systému.

Jazyk byl uveden v roce 2011 společností IntelliJ jako náhražka javy. Měl být modernější, ale přitom plně kompatibilní. Dnes se dá kotlin kompilovat také do javascriptu i nativního kódu.

Následuje tabulka s detailním porovnáním těchto dvou jazyků. Nejsou uvedeny všechny rozdíly, jen ty, které argumentují ve prospěch kotlinu. (8)

| | |
|---|---|
| Java má mnohdy zdlouhavou syntaxi. Je například nutné psát mnoho getterů a setterů, což je sice triviální a automaticky generovaný kód, avšak je poměrně mohutný | Kotlin nabízí velké množství zkratek a syntax sugaru, což je nutné se nejprve naučit, nicméně přínosy z toho vyplývající jsou obrovské. Například není defaultně nutné psát gettery a settery, ale v případě potřeby není problém je doplnit.<br><br>Existuje více příjemných zkratek. Např. není nutné psát klíčové slovo public – je výchozí. |
| Java není null safety. Kromě primitivních datovch typů může všechno nabývat hodnoty null. Také proto je NullPointerException nejčastější chyba v javě | Kotlin byl od základu navržen tak, aby byl každý typ ve výchozím stavu nenulovatelný (tedy je nutné každou instanci už při vzniku inicializovat). Pokud je potřeba povolit hodnotu null u proměnné x typu T, definuje se s otazníkem navíc: `var x: T? = null` |
| Java používá tzv. „checked exceptions“, tzn. je nutné u definice metody vypsat seznam možných výjimek, což se ukázalo jako nešťastný návrh, který zbytečně způsoboval starosti. Navíc příliš neladí s funkcionálními principy. | Kotlin se poučil z antipatternů javy a mimo jiné odstranil nutnost uvádět seznam možných výjimek. |
| Java není čistě objektový jazyk, neboť používá tradiční primitivní datové typy (jen nemá unsigned typy). Tyto typy není možní používat jako parametry generik a není možné do nich uložit hodnotu null – proto existuje pro každý primitivní typ wrapper | Kotlin definuje primitivní datové typy jako třídy a jejich instance jsou plnohodnotné objekty. Jsou použitelné jako parametry pro generiky a je-li nutné povolit hodnotu null, definují se s otazníkem navíc, např: `var x: Int? = null` |
| Java vyžaduje za každým příkazem středník | Kotlin nevyžaduje psát středníky, což je příjemnější a pohlednější. Výjimka je, pokud máme na jednom řádku více příkazů. |
| Funkce a proměnné musí být definovány výlučně uvnitř typů. Uvnitř třídy mohou být definovány další třídy, a to buď statické, nebo nestatické (nestatické mají odkaz na instanci obalujícího typu) | Funkce a proměnné mohou být definovány jak ve třídě, tak mimo ni. Uvnitř třídy mohou být definovány další třídy, a to buď s modifikátorem inner, nebo bez. Třídy s modifikátorem inner mají odkaz na instanci obalujícího typu. |

### C# a ASP.NET Core

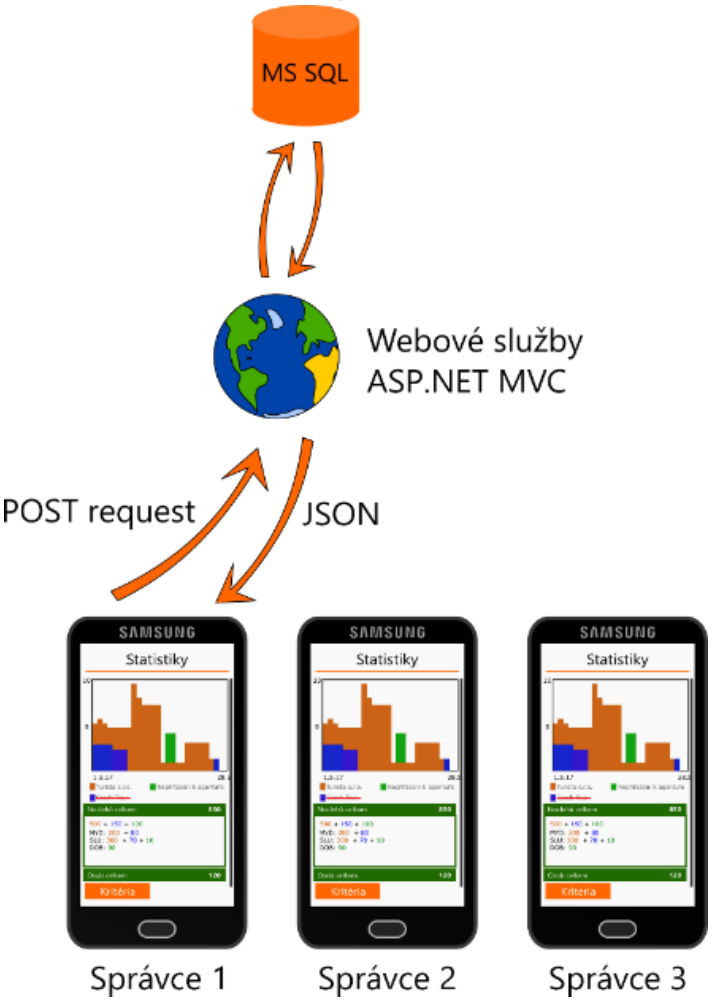
Jedná se o sofistikovaný jazyk a přehledný framework od společnosti Microsoft využívající architekturu MVC. S těmito technologiemi se setkávám na pracovišti.

### MS SQL

Výše zmíněné technologie výborně spolupracují s tímto databázovým produktem. Nechybí ani přímá podpora z visual studia.

# 3. Návrh

## 3.1 architektura systému




Architektura systému je trojvrstvá. Na nejnižší úrovni je centrální databáze, jež obsahuje až na výjimky všechna data.

Klienti nekomunikují s databází přímo, ale využívají služeb, jež jsou poskytovány webovou aplikací. Tyto služby reprezentují konkrétní úkoly – např. registruj skupinu, smaž ubytovnu, edituj hosta apod. Tomuto tématu bude věnována samostatná kapitola. Klienti posílají do této serverové vrstvy POST requesty, v jejichž těle jsou obsažena data ve formátu JSON. Server odpovídá jak http statusy, tak řetězci JSON.

Systém umožňuje, aby více správců spravovalo tytéž objekty. Synchronizaci dat napomáhá přijímání vektoru verzí v každé nechybové odpovědi. (více informací v kapitole o implementaci)

## 3.2. Vznik návrhu

Měl jsem možnost konzultovat s provozovatelem ubytoven. Právě jemu jsem dodal nějaké prototypy (obrázky apod., které představovali GUI), díky čemuž byl upřesňován návrh aplikace. Tyto konzultace formovaly požadavky na funkčnost tak, že do aplikace bylo rovnou naimplementováno vše potřebné

a nebylo nutné v pokročilých fázích projektu překopávat již zavedené funkce. Tyto konzultace byly ještě doplněny konzultacemi s provozovatelem hotelu, který používá systém previo. Systém Ubysys byl právě previem částečně inspirován.

### 3.2.1. Nálezy během formování návrhu a jejich řešení

Díky konzultování prototypů s provozovatelem bylo odhaleno několik chyb v návrhu aplikace. Návrh bylo potřeba doladit, aby z něho mohla vycházet implementace

- → O každém hostu je nutno zaznamenávat jméno, adresu a číslo dokladu apod.
  - Chyba byla v tom, že se hledělo na skupiny hostů jen jako na čísla udávající počet hostů. Tento příliš zjednodušený model nemohl odpovídat předpisům a přání uživatele
- → Každý host může mít jinou dobu příjezdu a odjezdu
  - Toto souvisí s předchozím problémem. Pokud by se na skupinu s pevně daným datem příjezdu a odjezdu hledělo jen jako na počet lidí, nemohl by být každý host individuálně spravován
- → Příliš mnoho proklikávání do hloubky
  - V prototypu nebylo hlavní menu vysouvací z levé strany, ale tvořilo úvodní stránku po přihlášení. Bylo tak zbytečně zdlouhavé se proklikat z tohoto menu do kalendáře ubytovny a následně např. zpět na úvod a do správy agentur. Nové menu nejen zjednodušilo ovládání, ale i odpovídá konvencím material designu.
- → Chybí možnost přepínání měny u plateb
  - Hosté mohou platit nejen v českých korunách, ale i v eurech.
- → Číslování skupin je zbytečné
  - Prototyp předpokládal, že každá skupina bude mít pořadové číslo, které uvidí uživatel. Ve skutečnosti byla tato funkcionalita spíše matoucí. Nehledě na to, že se nemusí dávat takový důraz na skupiny, protože uživatele budou poté spíše zajímat jednotliví hosté.

## 3.3. Organizační záležitosti

Jak na úrovni databáze a serveru, tak i na úrovni aplikace musí návrh systému odpovídat organizačním potřebám zařízení a jejich provozovatelů.

### 3.3.1. seskupování hostů do skupin

Jak pro hotely, tak i pro penziony a ubytovny je typické, že se hosté seskupují do skupin. Na hotelu často tvoří jednu skupinu obsazený pokoj, na ubytovně např. skupina dělníků. Účel skupiny je především jednotící (je potřeba vědět, kteří hosté patří k sobě). Také se díky skupinám snáze provádí různé fakturace. Platby za ubytování a doplňkové služby se často vážou právě na skupinu.

### 3.3.2. specifikace organizace ubytoven

Organizace ubytovny je z hlediska systému vlastně složitější, než je tomu u hotelů. Zde je několik rozdílů:

→ Na hotelu si hosté rezervují celé pokoje, zatímco na ubytovně jen lůžka. Proto se na pokoji mohou potkávat hosté z mnoha skupin
→ Na hotelu má celá skupina stejný datum příjezdu i odjezdu. Na ubytovně mají hosté ve skupinách individuálnější přístup. Každý host může mít jiný datum příjezdu i odjezdu, prodlužovat si pobyt nezávisle na ostatních apod.
→ Počítá se s dlouhodobými pobyty. Proto je zavedeno, že každá příchozí skupina může mít přiřazeno více plateb. Ne každý zaplatí takový pobyt najednou.
→ Na hotelu je běžné identifikovat příchozí skupinu příjmením hosta, nebo jménem firmy. Tento systém toto nebude vyžadovat. Skupiny mohou být bezejmenné

### 3.3.3 role agentur v provozu ubytovacích zařízení

Často se o ubytování hostů stará nějaká agentura. Při registraci skupiny je proto možné připojit konkrétní agenturu. Systém na tyto agentury uchovává různé kontakty, tudíž tato funkcionalita je používána i jako seznam telefonních čísel.

### 3.3.4. jednoduchá evidence plateb

Systém Ubysys si neklade za cíl konkurovat účetním systémům. Pouze umožňuje jednoduchou evidence plateb u skupin. Je povoleno mít různé platby v různých měnách

# 4. Implementace

## 4.1. server

### 4.1.1. struktura serveru

Srdcem systému je centrální server s databází. Jedná se o aplikaci ASP.NET Core, což je framework .NETu. Každá taková aplikace má architekturu MVC (model-view-controller). Pro tuto webovou aplikaci však není použiví viewů příliš typické, neboť většina jejích služeb vrací JSON s daty pro klientskou aplikaci

### 4.1.2. kotrolery

Kontolery jsou třídy, jejichž metody se vyvolávají po přijetí http požadavku.

**UbysysApiControllerBase**

Toto je základní kontoler – abstraktní třída dědící z třídy Controller (ta je poskytována frameworkem). Každý kontoler, jenž je součástí systému ubysys dědí z tohoto základního konroleru. K základním operacím přidává pár nových, Jsou to tyto metody:

→ CheckRequest – hlavním úkolem je ověřit platnost requestu. Zkontroluje hash dat a z přijatého podpisu requestu zjistí, který uživatel poslal požadavek, zkontroluje, zda má na danou operaci oprávnění
→ DataResponse – generická metoda vracející data pro aplikaci včerně version vectoru ve formátu JSON
→ OkResponse – dává najevo, že vše dobře dopadlo a vrací JSON bez dat, jen vektor verzí
→ ErrorResponse – vrací http status 500 a JSON s popisem chyby

Všechny tyto metody jsou nezbytně nutné. Každý request se zkontroluje metodou CheckRequest a většina requestů vrací JSON s daty nebo popisem chyby.

#### UbysysHouseController

Obsahuje metody pro základní CRUD operace s ubytovnami. Jsou to: Create, Delete, Edit, FetchAll, FetchDetails.

FetchAll načte jen základní informace o všech ubytovnách, což se používá např. pro výpis seznamu všech ubytoven, zatímco FetchDetails načte všechny informace o jedné ubytovně.

#### UbysysAgencyController

Obsahuje operace s agenturami. Jsou to pouze CRUD operace – create, edit, fetchAll a delete

#### UbysysGroupController

Je to spíše komplexnější kontoler, protože práce se skupinami je složitější.

- → Create – zaregistruje novou skupinu, pokud není překročena kapacita
- → Edit – upraví již zaregistrovanou skupinu. Funkce přijímá `GroupEditModel`, protože `Group` je složitější entita a je rychlejší a bezpečnější předávat jen změněná data, místo celé entity.
- → Delete – smaže skupinu podle id
- → FetchById – načte všechny informace o skupině s daným id
- → FetchSleepDisplayInfo – načte jen základní informace (nutné k vykreslení kalendáře) o skupinách, které se na ubytovně vyskytují v zadaném časovém úseku.
- → FetchFreeRooms – přijímá `FreeRoomsRequest` a spočítá počet volných lůžek v každém pokoji v daném termínu.

#### UbysysGuestController

Pracuje s evidencí hostů a je spíše jednodušší. Má metody Create, Edit, FetchById a FetchGuestsBySearchPattern

- → FetchById načte všechny informace o hostu s daným id
- → FetchGuestsBySearchPattern vrátí seznam hostů, jejichž jméno, příjmení či rok narození obsahují daný výraz.

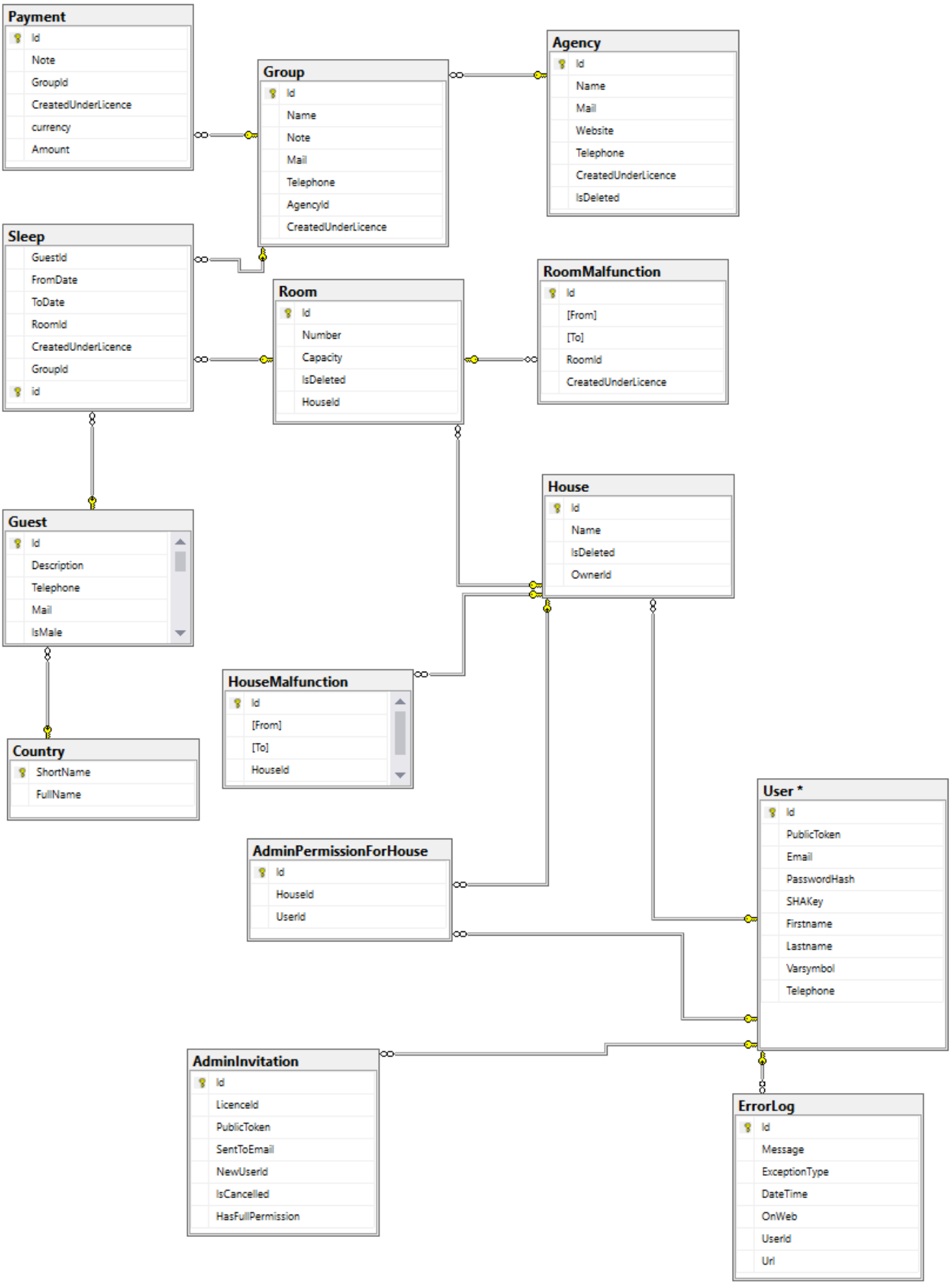
#### UbysysUsersController

Slouží k přihlašování a registraci z úvodní obrazovky aplikace. Obsahuje tedy metody Login a Register. Pro zjednodušení testování a demonstrace projektu je zavedeno, že každý registrovaný uživatel dostane roční licenci.

### 4.1.3. Databázové tabulky

Všechny důležitá data, která systém shromažďuje, končí v databázi. Sada tabulek a vztahy mezi nimi také souvisí s organizací aplikace a ubytovacího zařízení.

Každá tabulka má vygenerovanou třídu – tzv. entitu, která je obrazem této tabulky. Většina takových tříd implementuje interface IApiEntity, což znamená, že se jejich instance dají hashovat. Hash je důležitý prvek zabezpečení, jelikož tato data často proudí mezi klientem a serverem (tomuto bude věnována zvláštní kapitola)


Payment
Id
Note
GroupId
CreatedUnderLicence
currency
Amount
Group
Id
Name
Note
Mail
Telephone
AgencyId
CreatedUnderLicence
Agency
Id
Name
Mail
Website
Telephone
CreatedUnderLicence
IsDeleted
Sleep
GuestId
FromDate
ToDate
RoomId
CreatedUnderLicence
GroupId
id
Room
Id
Number
Capacity
IsDeleted
HouseId
RoomMalfunction
Id
[From]
[To]
RoomId
CreatedUnderLicence
House
Id
Name
IsDeleted
OwnerId
Guest
Id
Description
Telephone
Mail
IsMale
HouseMalfunction
Id
[From]
[To]
HouseId
Country
ShortName
FullName
User *
Id
PublicToken
Email
PasswordHash
SHAKey
Firstname
Lastname
Varsymbol
Telephone
AdminPermissionForHouse
Id
HouseId
UserId
AdminInvitation
Id
LicenceId
PublicToken
SentToEmail
NewUserId
IsCancelled
HasFullPermission
ErrorLog
Id
Message
ExceptionType
DateTime
OnWeb
UserId
Url

V obrázku schématu pro přehlednost chybí tabulka Licence, protože je na ni napojeno téměř všechno a pro ukázku schématu není podstatná.

Ve schématu jsou tabulky propojeny šipkami s bílým a žlutým koncem. To je zakreslení relace 0..n nebo 1..n. Žlutý konec symbolizuje 1, bílý konec n. Např. mezi tabulkami Payment a Group vidíme šipku, jež značí, že jedna skupina může mít n plateb.

### User

Uživatel systému. Význam některých vlastností je zřejmý: Id, Email, Firstname, Lastname, Telephone a PasswordHash. Jiné tak zřejmé nejsou a měly by být objasněny.

### Licence

Prvnotní význam je informace o licenci k používání softwaru. Je zde informace o platnosti. Uživatel může pracovat s aplikací, pokud má vlastní platnou licenci, nebo je jeho uživatelský účet napojen na cizí licenci (pak se jedná o účet zaměstnance, či sekundárního manažera

Vedlejší význam licence (a proto se na ni většina tabulek odkazuje) je informace o tom, pod jakou licencí byl který objekt vytvořen. To se používá na serveru ke kontrole před editací, stahováním či mazáním dat, aby se zjistilo, jestli uživatel nechce přepsat či číst data jiného uživatele.

### House

Ubytovna. Má libovolný počet pokojů a může mít teoreticky libovolný počet poruch (i když systém pracuje vždy jen s jednou). Má vlastníka, a proto se tato tabulka nemusí odkazovat na licenci.

### Room

Pokoj patří právě jedné ubytovně a může mít teoreticky libovolný počet poruch (i když systém pracuje vždy jen s jednou). Odkazuje se na dům, který má vlastníka, takže tato tabulka se také neodkazuje na licenci.

### Group

Hosté se seskupují do skupin. Pokud přijede jeden člověk, bude vytvořena skupina s jedním člověkem. Entita sama o sobě obsahuje v podstatě jen kontaktní údaje, poznámku a vybranou agenturu. Je zde odkaz na licenci.

### Guest

Informace o hostu jako o člověku – jméno, kontaktní údaje, národnost, číslo průkazu apod. Každý ubytovaný host musí být v evidenci hostů.

### Sleep

Každý host se může na ubytovně ubytovat libovolněkrát, přičemž po každém příjezdu není nutné znovu vyplňovat informace o hostu. Entita Sleep je vlastně informace o jednom pobytu (překlad Sleep je tedy nepřesný, ale už to tak zůstalo). Obsahuje datum příjezdu, odjezdu, příslušnost ke skupině a odkaz na použitý pokoj. Každý host může mít libovolně pobytů.

### Agency

Každá skupina může přijet s nějakou agenturou. Může se jednat o cestovní kanceláře.

### Payment

Ke každé skupině lze přiřadit 0..n plateb. Platba má název, částku, měnu a poznámku

#### RoomMalfunction
Porucha či uzavření pokoje. V tomto pokoji nebude možné počas poruchy ubytovávat hosty. Má Počáteční a koncové datum. Pokud je koncové datum null, chápe se to jako uzavření na dobu neurčitou.

#### HouseMalfunction
Porucha či uzavření ubytovny. V této ubytovně nebude možné počas poruchy ubytovávat hosty. Má Počáteční a koncové datum. Pokud je koncové datum null, chápe se to jako uzavření na dobu neurčitou.

#### AdminInvitation
Manažer ubytovny může přizvat ke spolupráci 0..n lidí. Pozvání se pošle na email. Příjemce dostane odkaz k registraci. Po registraci se tato pozvánka potvrdí a nový uživatel bude mít příslušná oprávnění vycházející právě z této pozvánky. Když chce manažer zaměstnance vyhodit, zruší platnost této pozvánky.

Pozvánka ke spolupráci je dvojího typu – nový uživatel se stane buďto zaměstnancem, nebo sekundárním manažerem.

#### AdminPermissionForHouse
Uživatel, který se zaregistroval na pozvání manažera a nemá manažerské oprávnění, může manipulovat jen s těmi ubytovnami, ke kterým má udělen přístup. Toto oprávnění je jednoduchá entita – obsahuje jen id ubytovny a id uživatele.

### 4.1.4. formát requestu
http request může mít v těle připojená data. Vždy se jedná buďto o primitivní datový typ, nebo o JSON v případě složitějšího objektu. Současně se ještě posílá veřejný token uživatele a request hash. Z veřejného tokenu uživatele se pozná, který uživatel poslal požadavek. Request hash je hash dat připojených v těle požadavku. Hash se vypočítává algoritmem HMACSHA256. Každý uživatel má svůj jedinečný klíč, aby se pouhou záměnou veřejného tokenu nemohl uživatel vydávat za někoho jiného. I když by znal cizí token, neodpovídal by hash requestu.

Toto zabezpečení requestu kontroluje již zmíněná metoda CheckRequest, která může neplatný požadavek odmítnout.

### 4.1.5. kontrola requestu
Zde se opět vrátíme k metodě CheckRequest, která se volá při každém požadavku. Nejprve pomocí veřejného tokenu uživatele zjistí, kdo poslal request. Pokud nebyl nalezen uživatel s daným tokenem, vrátí chybu `UserNotFound.` Zjistit odesilatele je nezbytné pro kontrolu hashe dat, neboť každý uživatel má jedinečný hashovací klíč. V případě, že server spočítá jiný hash, než přiložil uživatel v requestu, vrátí se chyba `WrongRequestHash`.

Dále metoda kontroluje oprávnění uživatele (jestli se jedná o hlavního manažera, manažera či zaměstnance). Každá operace má předepsáno, jaký stupeň oprávnění je potřeba. Pokud nemá uživatel dostatečná oprávnění, vrátí se chyba `OperationNotAllowed`. Pokud se jedná o vyhozeného zaměstnance, vrátí se chyba `AdminPermissionCancelled`.

Na závěr metoda kontroluje licenci. Pokud uživatel pracuje pod prošlou licencí, vrátí se chyba `LicenceExpired`. Pokud request prošel úspěšně každou kontrolou, je konečně vykonán.

Jelikož je tato kontrola nutná u každého requestu, má každá serverem poskytovaná služby v podstatě stejné schéma implementace.

### 4.1.6. Formát response

Server může vrátit tři typy odpovědí. Data a version vector ve formátu JSON, popis chyby ve formátu JSON, nebo chybový http status. Pro první dva případy existují již zmíněné metody DataResponse, OkResponse a ErrorResponse. Klientská aplikace umí tyto JSONy přeložit na objekty responsů a z nich extrahovat stažená data.

### 4.1.7. Version vector

Každá licence má přímo v paměti webové aplikace uložen version vector. Některé entity se totiž mění zřídkakdy a je lepší je cachovat na straně klienta. Version vector je informace o tom, v jaké verzi (tedy kolikrát byla upravena) daný typ entit. Pokud se verze změní, klient ví, že má stáhnout čerstvá data.

Toto téma bude plně objasněno v sekci o klientské aplikaci.

### 4.1.8. logování chyb

Chyby vzniklé neočekávanou výjimkou při vykonávání requestu se logují do databáze kvůli případné nápravě. Třída ErrorLogger s metodou Log(exception, url, userId) zaloguje informace o výjimce, kde k chybě došlo, kterému uživateli se stala a kdy se stala. Neočekává se, že by docházelo k výjimkám při logování, ale kdyby náhodou, tato chyba se již neloguje. Téměř všechny chyby vzniklé při logování souvisí s nemožností připojit se k databázi.

### 4.1.9. Další zabezpečení

Pro potřeby této bakalářské práce asi není nutné řešit zabezpečení na úrovni protokolu – tedy zprovoznění HTTPS. Je však vhodné zmínit, že v reálném provozu je toto zabezpečení nezbytně nutné, neboť zajišťuje bezpečné šifrování dat na cestě mezi klientem a serverem. Zabezpečení systému je však důležité i na jiných úrovních než komunikační protokol.

Metoda CheckRequest je z hlediska zabezpečení nezbytná. Její univerzalita bohužel nemůže ošetřit každé riziko. Nezabezpečí různé pokusy o hacknutí systému upravenými záškodnickými requestami. Časté je, že před smazáním či editací dat se musí ověřit, zda uživatel (resp. Skupina uživatelů) editovaná data skutečně vlastní. Je snadné poslat záškodnický požadavek na smazání ubytovny s cizím id. Proto má většina entit vlastnost CreatedUnderLicence. Je to id licence, pod kterou byla daná entita vytvořena. Tyto informace potřebuje jen server a nikdy se neposílají klientovi. Další rizika jsou spíše ojedinělá. Např. při vyhazování zaměstnance je nutné zkontrolovat, zda se manažer nepokouší vyhodit hlavního manažera apod.

### 4.1.10. Příklad těla metody služby poskytované serverem

V každém requestu je nutné provést víceméně stejné kontroly, takže serverem poskytované služby mají stejné schéma implementace:

```
User userSender = null;
try
{
	// security (request hash a public user token) a data jsou součástí requestu.
	// Framework je přeloží na objekty
```

```
        if (CheckRequest(out ErrorStatuses errorStatus, security, data, out userSender,
        PermissionLevel.<POŽADOVANÁ ÚROVEŇ OPRÁVNĚNÍ>))
        {
                // Většinou je potřeba znát id licence, pod niž uživatel operuje
                Licence licence = userSender.GetAssociatedLicence();
                // Zde se provede požadovaná operace a vrátí se data
                return DataResponse(<DATA PRO APLIKACI>, licence.Id);
        }
        else
        {
                return ErrorResponse(errorStatus);
        }
}
catch (Exception ex)
{
        // Neočekávané výjimky se logují
        ErrorLogger.Log(ex, "<URL>", userSender?.Id ?? 0);
        return ErrorResponse(ErrorStatuses.ServerError);
}
```

Jak vidno, všechny kontroly a operace logování nejsou zrovna triviální, ale jsou nezbytně nutné.

## 4.2. klientská aplikace

Klientská aplikace je nativní aplikace pro android vytvořená v android studiu. Nejprve bude ukázána struktura projektu a až poté budou vysvětleny jeho jednotlivé části

### 4.2.1. struktura klientské aplikace

- ◘ `ff` – rootový package projektu
  - ◘ `extensions` – obsahuje různé rozšiřující metody, které se nedají zařadit jinam. Jsou to různé formátovací metody, kontroly formátu, transformace dat a funkce, které mnohdy zjednodušují zápis
    - 🗋 `standardGson.kt` – soubor s jedinou veřejnou funkcí `getStdGson`, která vrací instanci typu `Gson`. To je objekt, který umí překládat objekty na JSON a naopak. Smyslem tohoto souboru je vytvářet standardní objekt `Gson`. To znamená, že objekt musí mít různé adaptéry na často serializované typy a vhodně nastavenou strategii includování a excludování různých speciálních členských proměnných.
    - ☉ `TextChanged` – je vlastně zjednodušující třída na typ `TextWatcher`, který poskytuje android pro sledování změn v textovém poli. TextWatcher si vynucuje implementovat metody `onTextChanged`, `beforeTextChanged` a `afterTextChanged`. V aplikaci se však používá jen `afterTextChanged`, takže tento zjednodušený typ umožňuje implementovat jen tuto jednu metodu a zbylé dvě nechává prázdné.
  - ◘ `models` – obsahuje datové třídy. Některé tyto třídy přicházejí jen jako odpověď ze serveru, některé se i běžně posílají. Některé tyto třídy odpovídají databázovým entitám.
    - ☉ `Entity` – abstraktní třída společná všem datovým třídám, které se posílají na server. Pro takové třídy je klíčové, aby měly implementovanou funkci `getEntityHash()`, která vrací pole bajtů vzniklé hashem HmacSha256. Dále implementuje funkci `jsonForRequest()`, která pomocí standardního objektu `Gson` převede instanci na JSON.

- ◘ `net` – toto je kompletní webové API ve všech jeho úrovních a podobách
  - ◘ `services` – obsahuje třídy, které jsou obrazem služeb, jež poskytuje server. V rámci webového API je to nejvyšší úroveň. Servisám je věnována samostatná kapitola.
  - ◘ `utils` – toto je package s funkcionalitou nižší úrovně. Tuto funkcionalitu používají spíše jen services, ale jinde v aplikaci není potřeba. Výjimkou je výčet chybových stavů `ApiErrorCodes`.
    - 🗋 `URLs.kt` – soubor obsahující jen readonly property vracející url konkrétné služby serveru.
    - ⊙ `Cache<T>` – generická třída tvořící cache konkrétního typu. Cache souvisí s úsporou přenosu dat a tomuto je věnován samostaný odstavec.
    - ⊙ `Caches` – singleton objekt obsahující všechny cache, které aplikace používá
- ◘ `ubysys` – Samotná aplikace. Jsou zde definovány všechny aktivity (obrazovky aplikace), dialogová okna a navíc objekt `GlobalData`.
  - ⊙ `GlobalData` – singleton obsahující data, která jsou potřeba prakticky pořád, a proto by měla být vždy po ruce. Jsou to informace o přihlášeném uživateli a URL adresa serveru aplikace.

### 4.2.2. Webové API – webový klient

Pojem webové API zahrnuje sadu tříd a funkcí na různých úrovních, které zajišťují komunikaci se serverem. API je samozřejmě vybudováno nad rozhraním, které poskytuje framework operačního systému android.

Nejnižším elementem webového API je objekt WebClient. Ten obsahuje jednu metodu `sendRequest`. Ta zkonstruuje POST request a na samostatném vlákně jej pošle na server. Pak čeká na odpověď.

```
fun sendRequest(requestInfo: RequestInfo, downloadingCompletedCallback:
(ApiRawResponse) -> Unit)
```

- → `requestInfo` obsahuje cílovou URL, data ve formátu JSON a request hash.
- → `downloadingCompletedCallback` se zavolá po odpovědi serveru.
- → Třída `ApiRawResponse` obsahuje buďto http status a JSON, jenž přišel ze serveru, nebo obsahuje interní chybový kód ubysysu (typické pro timeout atd., server nedostupný a jiné situace, kdy nepřijde vůbec žádná odpověď). Jedná se o „surovou“, nijak nezpracovanou odpověď.

Ohledně androidu je zavedená poučka, že networking by neměl být vykonáván na hlavním vlákně. Proto se komunikace se serverem vykonává na vedlejším vlákně a po dokončení se zavolá callback s přijatými daty a http statusem

### 4.2.3. Webové API – přijímání odpovědí ze serveru

Výše bylo zmíněno, že třída `ApiRawResponse` může obsahovat http status a odpověď ve formátu JSON. Tato textová odpověď v případě chyby obsahuje informace o chybě, v případě úspěchu pak žádaná data a navíc i vektor verzí dat. Je očividné, že textové informace z instance `ApiRawResponse` by se měly přeložit na nějaký objekt, se kterým se bude pracovat efektivněji. K tomu slouží její metoda

```
fun <TData> parseResponse(targetTypeToken: Type) : Response
```

- → `TData` je typový parametr určující očekávaný typ odpovědi. Pokud například voláme službu vracející seznam ubytoven, víme, že `TData` bude `ArrayList<House>`
- → `targetTypeToken` je typový token typu `DataResponse<TData>`, na který se bude odpověď překládat. Parsování z JSONu provádí knihovna GSON. Problém JVM je, že nemá dokonale implementovány generiky. Pod pokličkou se místo generických parametrů dosazuje všude `Object`. Proto se musí přidávat token typu jako informace navíc.
- → `Response` je základní třída pro `DataResponse<TData>` a `ErrorResponse`. Metoda může proto v závislosti na obsahu surové odpovědi vrátit buďto datový objekt, nebo informace o chybě

### 4.2.4. Webové API – services

Již bylo ukázáno, jak se posílá požadavek na server a jak se získané odpovědi převádí na finální data. S touto funkcionalitou by se však stále pracovalo nepohodlně. Chybí ještě jedna vrstva:

Services je množina tříd, které definují konkrétní úkoly. Tyto servicy jsou vlastně obrazem služeb, které poskytují kontrolery na serveru. Například `HouseController` definuje služby `create`, `fetchAll`, `fetchDetails`, `edit` a `delete`. Třída `HouseServices` definuje přesně ty samé funkce.

Některé z nich jsou ukázány zde:

```
fun create(house: House, callback: (House) -> Unit, errorCallback: (ApiErrorCodes)
-> Unit)
```

- → `house` je entita představující ubytovnu. Obsahuje informace o pokojích a zaměstnancích, kteří mají oprávnění tuto ubytovnu spravovat
- → `callback` se zavolá v případě úspěšného uložení. Zde parametr typu `House` je tentýž dům, který se ukládá do databáze, jen má navíc přiřazeno id.
- → `errorCallback` se zavolá v případě chyby. `ApiErrorCodes` je výčet chybových kódů, které mohou nastat.

```
fun delete(houseId: Int, callback: () -> Unit, errorCallback: (ApiErrorCodes) ->
Unit)
```

- → houseId je id ubytovny, jež se má smazat
- → Další dva parametry mají stejný význam, jako u předchozí ukázky.

```
fun fetchAll(callback: (ArrayList<House>) -> Unit, errorCallback: (ApiErrorCodes)
-> Unit)
```

- → oba parametry mají stejný význam, jako u předchozích ukázek

Je vidět, že všechny metody servis mají velmi podobný tvar. Mají callback, který se volá po stažení dat, druhý callback, který se volá v případě chyby a samozřejmě vstupní data. S těmito metodami se

pracuje již velmi pohodlně a efektivně. Již se nemusí při každém volání řešit multithreading, konstrukce POST requestu, převod vstupních dat na JSON, parsování příchozího JSONu apod. Dá se říct, že byl tímto implementován návrhový vzor fasáda – skryli jsme velmi složitou implementaci.

Systém má šest tříd servis. Jsou to:

### AdminInvitationServices

Zabývá se pozvánkami ke spolupráci a zaměstnanci

- → `fetchAll` – stáhne všechny potvrzené pozvánky (kde proběhla registrace přes webový formulář. Jedná se tedy o zaměstnance) i nepotvrzené pozvánky
- → `fetchAdmins` – stáhne všechny zaměstnance. Stahuje více informací o uživatelích, než služba fetchAll
- → `cancelInvitation` – manažer může zrušit pozvánku ke spolupráci. Pokud se její příjemce ještě nezaregistroval, už to ani nebude možné. Pokud se již zaregistroval a je zaměstnancem, je vlastně propuštěn.
- → `newInvitation` – server vygeneruje novou pozvánku. Dostane informaci, zda se bude jednat o pozvání spolumanažera či zaměstnance.

### AgencyServices

Jedna z nejjednodušších servisních tříd. Zabývá se agenturami.

- → create
- → edit
- → delete
- → fetchAll

k tomu není co dodat.

### GroupServices

Zabývá se skupinami hostů

- → `create` – vytvoří skupinu. Server kontroluje, zda se uživatel nepokouší zaregistrovat skupinu, která by převýšila kapacity volných lůžek. Při převýšení kapacity se skupina nezaregistruje.
- → `fetchSleepDisplayInfo` – stáhne informace o skupinách pro vykreslení kalendáře. Nestahuje kompletní detailní informace. Vyžaduje id ubytovny, informaci od kdy do kdy mám kalendářní data již stažená a od kdy do kdy chci stáhnout nová data. Tímto se mohou stáhnout jen informace jen o těch skupinách, které ještě nemám.
- → `fetchFreeRooms` – vrátí seznam pokojů s počtem volných míst v daném termínu.
- → `edit` – edituje skupinu. Jako parametr překvapivě přijímá instanci `GroupEditModel`, protože entita `Group` je složitější – obsahuje mj. seznam hostů a plateb, takže je efektivnější a bezpečnější neposílat vždy celou entitu, ale jen změněné údaje.
- → `delete` – smaže skupinu. Obsazená lůžka se tak znovu uvolní

GuestServices

I když se jeden host ubytovává opakovaně, jakožto osoba pořád má jen jeden záznam v tabulce hostů. Pokaždé, když se host v rámci skupiny registruje k ubytování, recepční nemusí znovu a znovu vyplňovat všechny údaje, ale jen spojí dříve uloženého hosta s novou skupinou.

Entita `Guest` je navíc obsáhlejší. O hostu se evidují různé kontakty, adresy, jméno a příjmení, národnost, datum narození atd. takže je velmi výhodné, když se vše vyplní jen jednou. Samotná třída servis je jednoduchá:

- → `create` – vloží nového hosta do databáze hostů
- → `fetchGuestsBySearchPattern` – podle zadaného řetězce vyhledá hosta. Zadaný výraz se porovnává s příjmením, jménem a rokem narození hostů
- → `fetchById` – stáhne veškeré údaje o hostu s daným id.

HouseServices

- → `create` – vytvoří novou ubytovnu. Je možno rovnou přidat pokoje, zaměstnance a poruchy
- → `fetchAll` – stáhne základní informace o všech ubytovnách (ne pokoje, zaměstnance, nebo poruchy)
- → `fetchDetails` – stáhne kompletní informace o jedné ubytovně
- → `edit` – přijímá `HouseEditModel`, protože entita `House` je složitější – obsahuje mj. seznam pokojů, zaměstnanců a různé poruchy, takže je efektivnější a bezpečnější neposílat vždy celou entitu, ale jen změněné údaje.
- → delete – smaže ubytovnu. Provede se pouze soft delete, pokud na této ubytovně někdy bydlel alespoň jeden host

UserServices

Je určeno pro registraci a přihlašování z úvodní obrazovky aplikace

- → `register` – přijímá UbysysRegisterModel, který obsahuje mail, heslo, potvrzení hesla, jméno a příjmení uživatele
- → `login` – přijímá UbysysLoginModel, který obsahuje email a heslo

### 4.2.5. Webové API – cache namísto stahování dat

Prakticky každá obrazovka aplikace potřebuje nějaká data ze serveru. Tím by se aplikace zpomalovala a objem přenášených dat by rostl, což uživatelé mobilních telefonů nechtějí. Networking je tedy drahá operace. Nabízí se otázka, zda bychom nemohli taková data, která se zřídkakdy mění, cachovat.

To samozřejmě možné je. Typickým příkladem málokdy měněných dat je seznam ubytoven a informace o nich. Většina majitelů ubytoven málokdy kupuje nové budovy, nebo staví nové pokoje. Nebo seznam agentur – dle slov uživatelů nepřibývají spolupracující agentury příliš často. Takové informace se vyplatí stáhnout jednou a aktualizovat je, až (pokud vůbec) dojde ke změně. Opakem jsou informace ohledně příjezdů a odjezdů. Každý den se mohou mnohokrát změnit.

Nyní přichází ke slovu vektor verzí, který posílá server s každou odpovědí typu `DataResponse`. Každý uživatel má uloženou verzi informací o agenturách, ubytovnách, detailních informacích. Je to vektor

celých čísel. Při přidání, smazání, či editaci entity se verze daného typu inkrementuje. Když klient dostane v rámci nějakého requestu odlišný vektor, než má uložen na svém zařízení, dojde ke zinvalidnění dané cache, což aplikaci donutí při další potřebě stáhnout čerstvá data.

Každá cache má omezenou životnost, která se s každým přijetím vektoru refreshuje. Pokud by došlo k pokusu o použití cache, která by byla již neplatná, zinvaliduje se a stáhnou se čerstvá data.

Vektor verzí je uložen na serveru ve webové aplikaci přímo v paměti – tedy ne v databázi. To proto, že to není nijak velký objem dat (každý uživatel potřebuje jen pár bajtů) a při každém requestu se kontroluje. Databáze by tyto operace zbytečně zdržovala.

### 4.2.6. specifika registrace a přihlašování

Kvůli testování s uživateli a kvůli možným prezentacím došlo k jedné úpravě – když se uživatel zaregistruje přes úvodní obrazovku aplikace, dostane licenci na rok. Stává se z něj tedy hlavní manažer – majitel licence.

Pokud chce hlavní manažer pozvat další manažery nebo zaměstnance, přes aplikaci vygeneruje pozvánku, která se pošle na zadaný email. Příjemce pozvánky klikne na odkaz a přes webový formulář provede registraci. Tím se vytvoří uživatelský profil zaměstnance či spolumanažera a uživatel se již bude přihlašovat standardně přes úvodní obrazovku aplikace.

## 4.3. Popis a ukázka funkcionalit

Zde je ukázáno, jak fungují některé důležité obrazovky aplikace. Zde ukázané obrazovky se již nerozebírají v příloze.

### 4.3.1. Kalendář ubytovacího zařízení

Viz. Obrázek A3. Toto je jedna z nejdůležitějších funkcionalit, a proto je logické, že se zobrazí hned po přihlášení. Je to asi jediná obrazovka, která vypadá lépe v otočení na šířku.

Můžeme vidět, že je to kalendář pokojů, který se dá scrollovat doleva i doprava nekonečně daleko. Pokud chce uživatel zkontrolovat nějaké vzdálené datum, je lepší tam rovnou přeskočit. To se dělá kliknutím na tlačítko s letopočtem v rohu. Zobrazí se systémový dialog pro výběr data. V levém sloupci jsou vidět čísla pokojů. Záhlaví tabulkového kalendáře tvoří datumy a dny v týdnu, přičemž jsou barevně odlišeny víkendy a dnešní datum. Vpravo dole je modlé tlačítko, které slouží k registrování nové skupiny. V záhlaví obrazovky máme tlačítka dvě – pro nastavení ubytovny a výběr ubytovny (lokality)

Čísla v buňkách kalendáře znamenají počet bydlících osob v daném dni. Po rozkliknutí se ukáže seznam bydlících hostů, z něhož je možné přejít do editace skupiny.

### 4.3.2. Registrace/editace/smazání skupiny

Viz. Obrázek A4 a Obrázek A5. Vytváření skupiny je složitější operace. Kromě základních informací se také vyplňuje seznam hostů a seznam plateb. Pokud je skupina již zaregistrována, nahoře v záhlaví se zobrazuje tlačítko pro smazání skupiny. Samozřejmě se automaticky uvolní obsazená lůžka.

Na obrázku vlevo je vidět výběr hostů. Zelené tlačítko [+] slouží k zaregistrování nového hosta.

### 4.3.3. Vytvoření/editace/smazání ubytovacího zařízení

Viz. Obrázek A6  Vytváření a editace ubytovny je podobně komplexní záležitost. Vyplňuje se seznam pokojů a seznam zaměstnanců *, Každý pokoj se dá dočasně uzavřít, k čemuž slouží žluté tlačítko. Dá se i odstranit, či přidat nový.

* Je zbytečné přidávat na seznam zaměstnanců manažery, protože ti mají automaticky přístup ke všem ubytovnám.

### 4.3.4. Registrace/editace/smazání agentury

Viz. Obrázek A7 a Obrázek A8 O agenturách se uchovávají jen kontaktní informace, takže je práce s nimi jednodušší. Na seznamu agentur jsou dle vyplněných údajů viditelná až dvě tlačítka pro komunikaci – jedno vytočí číslo na agenturu, druhé pošle email. Právě tato operace je vidět na obrázku

## 4.4. uživatelské testování

Uživatelské testování mělo ověřit intuitivnost aplikace a odhalit případné chyby v návrhu, které by znemožňovali pohodlné a efektivní využívání aplikace.

Test proběhl s pěti účastníky, z čehož dva pracují jako poskytovatelé ubytování. Jeden provozuje hotel, kde se používá systém Previo, druhý provozuje několik ubytoven, kde se zatím žádný speciální systém nepoužívá. Obecně platí, že všichni uživatelé mají zkušenost s chytrými telefony a řadou aplikací. Výsledky testů by tak měly vynést cenné informace.

Testování proběhlo tak, že uživatelé dostali dvanáct úloh, které měli postupně vykonat. Přitom na ně bylo dohlíženo. Už v průběhu tohoto testování bylo odhaleno několik nedostatků návrhu, další náměty na vylepšení vyplynuly ze závěrečného rozhovoru s uživateli.

### 4.4.1. seznam úloh

1. Zaregistrujte se
    - Zde stačilo zadat fiktivní email, neboť nebylo nutné ho ověřit a sloužil jen jako přihlašovací údaj. Další testování probíhalo po přihlášení na jiný uživatelský účet, kde již bylo uloženo více ubytoven.
2. Vytvořte novou ubytovnu s názvem „Les“ se dvěma dvoulůžkovými pokoji
    - Tento úkol vedl na povšimnutí si tlačítka [+] v dialogu
3. Ubytujte Karla Svobodu na celý červen. Přijel s agenturou „Cestuj CZ“ a zaplatil 500€
    - Uživatelé si měli všimnout tlačítka [+] vpravo dole a správně vyplnit informace o nové skupině a ověřit, zda Karel Svoboda je v databázi hostů. První uživatel ho tedy musel vytvořit, další uživatelé jen správně použít vyhledávání hostů
4. Ubytujte čtyři hosty do nějaké ubytovny na červen
    - Zde bylo podstatné, že aktuálně používaná ubytovna Les má málo volných míst. Uživatelé měli přepnout na jinou ubytovnu.
5. Přejmenujte ubytovnu „Les“ na „Háj“
    - Uživatel se měl vrátit do nastavení ubytovny

6. Smažte rezervaci Karla Svobody
7. Smažte agenturu „Cestuj CZ“
8. Smažte ubytovnu Háj
9. Zavřete ubytovnu „Magyar“ na celý rok 2019
10. Zkontrolujte, zda je v ubytovně Magyar nějaký host 5.1.2026
    - Bylo očekáváno, že nikdo nebude scrollovat v kalendáři ubytovny, ale uživatelé budou hledat tlačítko pro zobrazení konkrétního data. Následně byla ověřena jejich schopnost chápání systémového dialogu pro výběr data.

Úlohy byly seřazeny tak, aby na sebe logicky navazovaly.

### 4.4.2. průběh testování

Uživatelé dostali mobilní telefon s běžící aplikací na přihlašovací stránce. Vždy jim byl přečten následující úkol, který pod dohledem vykonávali. V případě zmatení jim byla poskytnuta nápověda. Výstupem testování byly poznámky popisující nalezené problémy – tj. pokud aplikace padala, nebo nějaká část GUI byla matoucí. Někteří uživatelé zkoušeli pracovat se systémem nad rámec zadaných úkolů, což přispělo k odhalování chyb.

### 4.4.3. nálezy a jejich řešení

→ Na úvodní obrazovce uživatelé přehlíželi tlačítko k registraci (snažili se rovnou přihlásit)
- Tlačítko je implementováno jako textview a má vzhled jako odkaz na webu (Ještě nemáte účet, registrujte se). Proto bylo zvýrazněno a byla zvolena křiklavější barva, aby nedošlo k záměně s textem.

→ Aplikace při vytváření ubytovny padala
- Tato chyba byla způsobena nechtěným zásahem do kódu, kdy původně fungující řádek nyní obsahoval nesmyslné parsování z příchozího JSONu. Namísto entity `House` se aplikace snažila zkonstruovat instanci úplně jiného typu. Nápravou bylo přepsání tohoto řádku

→ Někteří uživatelé nepochopili dialog vyhledávání hostů (mysleli si, že se jedná o registraci hosta)
- Dialog byl přejmenován. Původní nadpis „hosté“ je méně intuitivní, než „vyhledávání hostů“. Pole pro psaní vyhledávaného výrazu má hint „hledaný výraz“, takže toto nedorozumění je překvapivé. Pro jistotu vedle vstupního pole přibyla ikonka lupy – běžný symbol pro vyhledávání

→ Pokoje v kalendáři nebyly seřazeny podle čísel
- Uživatelé správně očekávali opak. Tato chyba byla při vývoji přehlídnuta. K nápravě došlo na straně klienta.

→ Někteří uživatelé si nevšimli přepínače měny v editačním dialogu plateb
- Přepínač byl původně umístěn pod vstupním polem „částka“. Uživatelé uvedli, že kdyby bylo vpravo od pole, bylo by to přehlednější.

→ Počet volných míst na pokoji se zobrazuje správně jen při vytváření skupiny, ne při editaci
- Toto byla chyba v kódu, ne v návrhu. Ze všech chyb byla nejzávažnější a byla odstraněna na straně serveru.

Každý testovaný subjekt přispěl nějakým nálezem. Některé nálezy byly nahlášeny hned několikrát.

| Nález | Hlášen | Priorita | Poznámka |
|---|---|---|---|
| Přehlížení odkazu k registraci | 4x | High | Tlačítko, které je implementováno jako klikací text potřebovalo hodně zvýraznit. Po zvýraznění uživatelé uvedli, že přihlašovací obrazovka je mnohem čitelnější |
| Pád aplikace při vytváření domu | 1x | High | Po prvním výskytu byla chyba opravena, protože znemožňovala testování s dalšími subjekty. Jednalo se o závažný bug. |
| Nepochopení dialogu vyhledávání hostů | 2x | Medium | Většina uživatelů pochopila rozdíl mezi vyhledáním a registrací hosta, ale přesto byla i tato funkcionalita zpřehledněna |
| Nesprávné řazení pokojů v kalendáři | 2x | Low | Uživatelé očekávali opak, takže přemýšleli, podle jakého klíče jsou pokoje řazeny. |
| Přehlédnutí přepínače měny | 1x | Low | Pouze jeden uživatel přehlédl přepínač, přesto byl proveden redesign formuláře |
| Chybné počítání volných míst | 1x | High | Chyba byla odhalena pouze jednou, což poukazuje na její nenápadnost a zákeřnost. Počítání volných míst je kriticky důležité. |

# 5. Závěr

Během posledních dvou semestrů vznikl systém, který je použitelný a splňuje zadání práce. Zadání se skládalo ze sedmi podúloh a zde je rozepsáno jejich vyřešení.

Pro pohodlnější uživatelské testování a případné demonstraci projektu bylo zavedeno jedno zjednodušení. Každý nově registrovaný uživatel dostane roční licenci zdarma. Byl však implementován náznak licenčního systému – např. každý uživatel má svůj variabilní symbol, jímž se identifikují platby.

**1. Proveďte rešerši současných aplikací pro správu ubytoven**

Byly zkoumány různé již zavedené systémy, a to jak pro hotely, tak pro ubytovny. Bylo potřeba je porovnat a pochopit rozdíly v organizaci hostů v daných typech ubytovacího zařízení. Bylo zjištěno dost rozdílů, které ale většinou nepřekvapili. I tak se ukázalo, které funkce bude do tohoto systému potřeba naimplementovat a které jsou zbytečné. Tento projekt byl inspirován hlavně systémem Bananadesk (viz. rešerše), přičemž systém ubysys neopakuje nedostatky v jeho návrhu. Některé funkce inspirované Bananadeskem byly zjednodušeny. Systém ubysys cílí na menší a rodinné podniky, takže jednoduchost je vítána.

**2. Navrhněte strukturu databáze, webového API a mobilní aplikace pro správu ubytoven**

Databáze a centrální server s poskytovanými službami byly vytvořeny již minulý semestr jako samostaný projekt. Při vývoji mobilní aplikace však bylo ještě potřeba doladit pár nedostatků, které byly odhaleny detailnější rešerší a uživatelským testováním.

Mobilní aplikace je nejnovější součást systému. Její GUI bylo inspirováno již existujícími systémy, což může usnadnit práci všem uživatelům, kteří mají s podobnými systémy zkušenost.

### 3. Implementujte všechny tři součásti systému z bodu 2.

Stejně jako návrh, i implementace byla po rešerši i uživatelském testování doladěna.

- Aplikace je nativní androidová, napsaná v jazyce kotlin.
- Server je napsán v C# a postaven na frameworku ASP.NET Core. Databáze je typu MS SQL, protože tato technologie výborně spolupracuje s C# a nechybí ani přímá podpora z visual studia.

### 4. Navrhněte systém úspory přenosu dat a synchronizace

Některá data se mění zřídkakdy a je lepší uchovávat jejich lokální kopii, než je pořád znovu stahovat. Takový systém byl navržen.

Bez takovéto cache by nebylo nutné řešit synchronizaci, protože by každý uživatel při každé operaci stahoval čerstvá data. S využitím cache nastává problém – pokud jeden uživatel uchovává svou kopii dat a jiný uživatel provede změnu, musí být přítomen systém, který umožní ostatním zjistit, že došlo ke změně. Toto bylo implementováno pomocí vektoru verzí dat.

### 5. Navrhněte a implementujte vhodný způsob zabezpečení

Zabezpečení bylo implementováno na více vrstvách. Pro potřeby demonstrace této bakalářky není nutné zavádět HTTPS připojení, avšak v reálném provozu je to nezbytné.

Zabezpečené je posílání požadavků na server, a to tak, že může posílat požadavky jen řádně přihlášený uživatel. Toho bylo dosaženo hashovacím algoritmem, přičemž každý uživatel má přiřazen jiný hashovací klíč.

Na serveru se při každém requestu kontroluje, zda má uživatel platnou licenci a zda se nepokouší přepsat či smazat cizí data. V databázi je zabezpečeno ukládání hesel rovněž hashováním HMACSHA256.

Zabezpečení jsou tak implementována na všech úrovních.

### 6. Navrhněte uživatelský interface mobilní aplikace a funkcionalitu pro registraci uživatelů na webu

Téměř všechno GUI, které je součástí systému, je součástí mobilní aplikace. Jediná výjimka je formulář pro registraci uživatele, který se registruje na pozvání ke spolupráci. Tento formulář je na webu a registrace je tak rychlejší, protože pozvánky ke spolupráci chodí na email, v němž je odkaz s daným tokenem. Stačí tak jen kliknout na odkaz. Nehledě na to, že uživatel, který teprve dostal pozvání ke spolupráci, nemusí mít ještě aplikaci staženou.

Veškeré GUI v aplikaci otestovali uživatelé.

### 7. Navrhněte vhodný způsob testování a aplikaci otestujte, výsledky vyhodnoťte

V takových aplikacích je důležitý vhodný návrh GUI, aby bylo pohodlné a intuitivní. Proto proběhlo testování s pěti uživateli, z nichž dva byly experti z oboru ubytování – jeden provozovatel hotelu a jeden provozovatel ubytoven. Takové testování je nenahraditelné žádným automatickým testem.

Výstupem uživatelských testů byl seznam nálezů chyb, případně některé názory a náměty. Každý nález musel být napraven.

Naopak server lze testovat nějakým integračním testem, který se tváří jako klientská aplikace – tzn. posílá ty samé požadavky a umí přijímat ty samé odpovědi. Vyhodnocují se hlavně změny v databázi, případně příchozí odpovědi.

# Citovaná literatura


1. macworld.co.uk. [Online] [Citace: 13. Květen 2019.] https://www.macworld.co.uk/feature/iphone/iphone-vs-android-market-share-3691861/.

2. *amarinfotech.com*. [Online] [Citace: 13. Květen 2019.] https://www.amarinfotech.com/best-android-development-tools-2018.html.

3. source.android.com. [Online] [Citace: 13. Květen 2019.] https://source.android.com/setup/start/build-numbers.

4. *kotlinlang.org*. [Online] [Citace: 13. Květen 2019.] https://kotlinlang.org/docs/reference/android-overview.html.

5. *previo.cz*. [Online] [Citace: 13. Květen 2019.] https://www.previo.cz/.

6. *previo.cz*. [Online] [Citace: 13. Květen 2019.] https://www.previo.cz/obchodni-zastupci#company-info.

7. *previo.cz*. [Online] [Citace: 13. Květen 2019.] https://www.previo.cz/co-mi-previo-prinese.

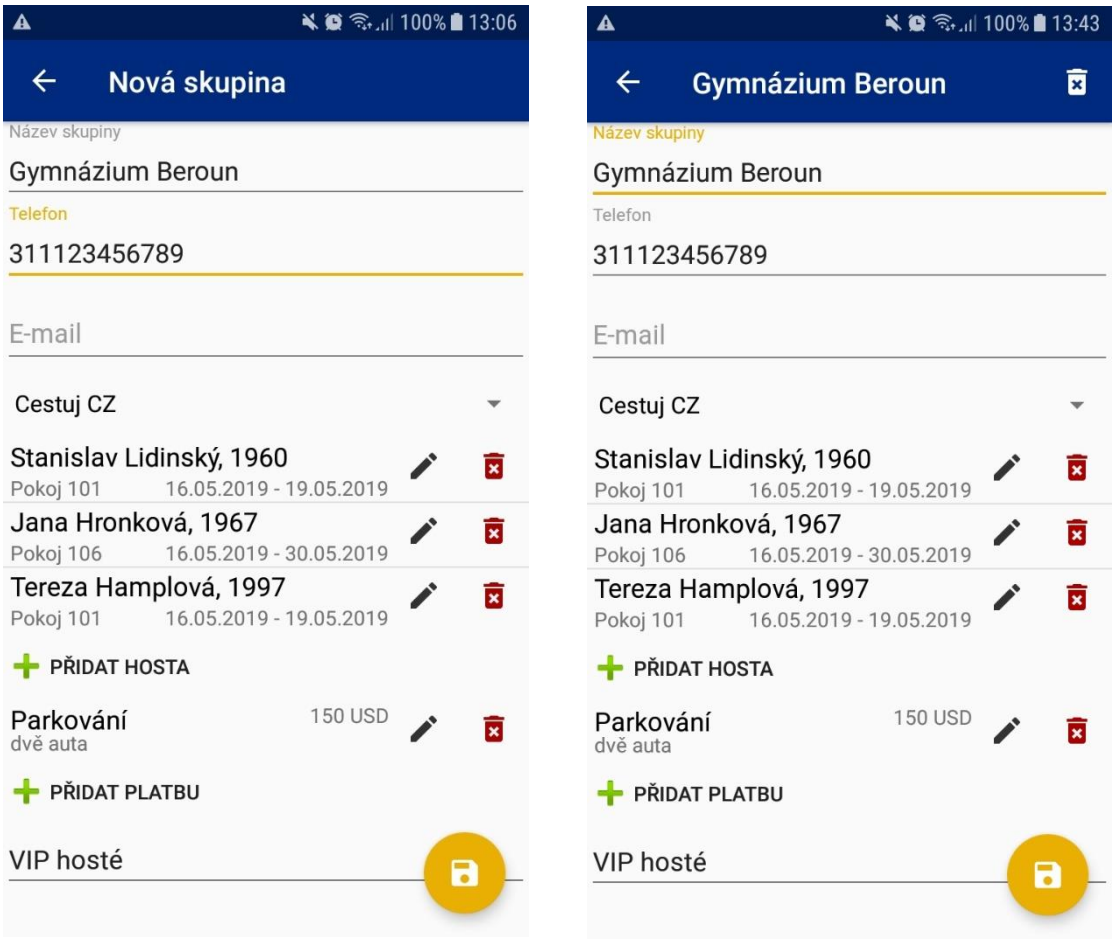
8. *kotlinlang.org*. [Online] [Citace: 13. Květen 2019.] https://kotlinlang.org/docs/reference/comparison-to-java.html.


## Odkazy

| | |
|---|---|
| previo.cz | Oficiální web společnosti previo provozující stejnojmenný systém (2019) |
| bananadesk.com | Oficiální web společnosti bananadesk provozující stejnojmenný systém (2019) |

# Příloha A. Obrazovky aplikace

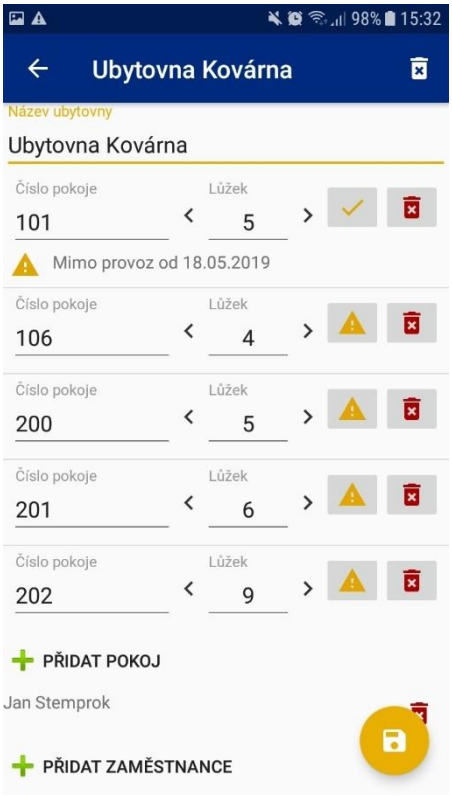

Obrázek A3 Kalendář ubytovny, který se zobrazí hned po přihlášení



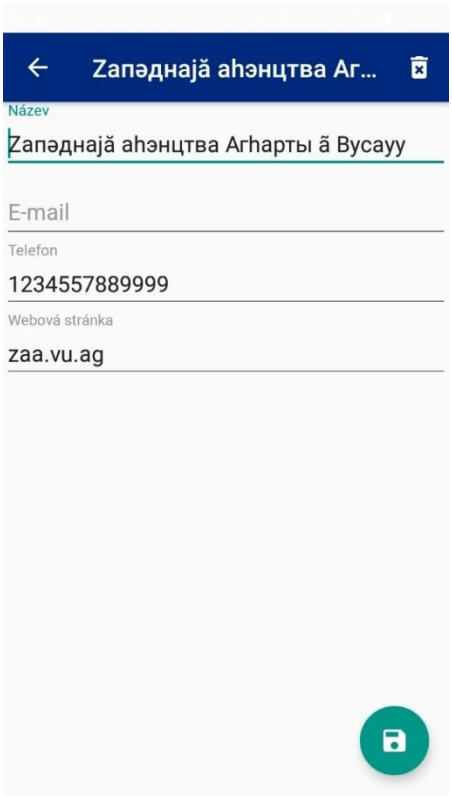

Obrázek A4: Vytváření nové skupiny a editace registrované skupiny s tlačítkem smazat.


Obrázek A5: Výběr hosta při přidávání hosta při editace skupiny

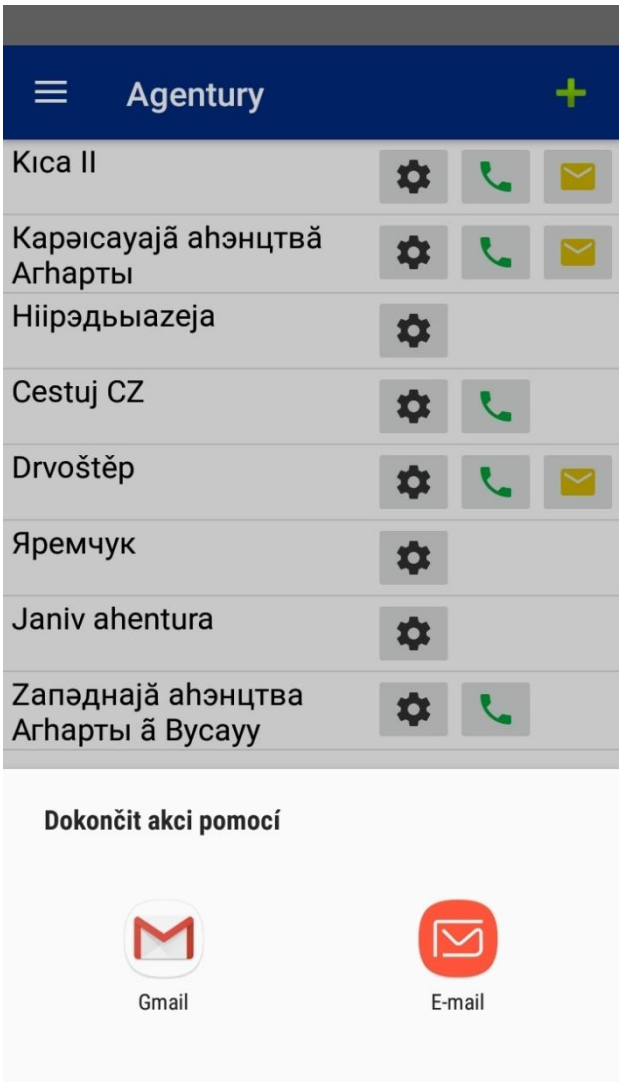
Obrázek A3:




Obrázek A6: Editace ubytovny s tlačítkem smazat




Obrázek A7: Editace agentury s tlačítkem smazat




Obrázek A8: Seznam agentur s různými kontaktními údaji s rozkliknutou operací poslat email.