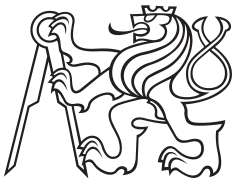
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Webová aplikace pro správu úkolů

Dan Nguyen

Vedoucí: Ing. Pavel Šedek
Obor: Softwarové inženýrství a technologie
Prosinec 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Nguyen** Jméno: **Dan** Osobní číslo: **465897**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Webová aplikace pro správu úkolů firmy

Název bakalářské práce anglicky:

Web-based company task management application

Pokyny pro vypracování:

Analyzujte požadavky zadavatele na aplikaci pro správu firemních úkolů. Na základě požadavků vyberte vhodné existující aplikace a stanovte náročnost úprav a integrace s podnikovým systémem. Dále stanovte náročnost implementace vlastního řešení. Porovnejte obě možnosti a vybranou variantu realizujte. Navrhněte testovací scénář a prověřte, zda ho realizované řešení splňuje.

Seznam doporučené literatury:

[1] Learning React Functional Web Development with React and Redux, Alex Banks, Eve Porcello
[2] UML 2 a unifikovaný proces vývoje aplikací, Ila Neustadt, Jim Arlow

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Pavel Šedek, katedra ekonomiky, manažerství a humanitních věd FEL

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

Datum zadání bakalářské práce: **27.02.2019** Termín odevzdání bakalářské práce: **24.05.2019**

Platnost zadání bakalářské práce: **19.02.2021**

Ing. Pavel Šedek 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

Poděkování

Tato práce navazuje na stávající informační systém společnosti Curso s.r.o. Díky mé spolupráci s touto společností jsem získal mnoho zkušeností s návrhem a vývojem softwarových řešení, které bych chtěl nadále prohlubovat a v tomto směru se dál vzdělávat. Tuto práci jsem si vybral díky této spolupráci, která mi umožnila skloubit zaměstnání s mým studiem.

Chtěl bych tedy tímto poděkovat za tuto příležitost společnosti Curso s.r.o. zejména Ing. Pavlovi Šedkovi za jeho vedení a pomoc při zhotovení této práce.

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně a použil jsem pouze literaturu uvedenou v přiloženém seznamu.

V Praze, 10. prosince 2019

Abstrakt

V této práci jsem navrhl rozšíření pro stávající informační systém společnosti Curso s.r.o. Toto rozšíření bude zaměřeno na správu a evidenci úkolů zadaných managementem svým terénním zaměstnancům.

Klíčová slova: Úkol, Správa, Software, Aplikace, Systém, Mobilní, Analýza

Vedoucí: Ing. Pavel Šedek

Abstract

In this work I designed an extension to a current information system for company Curso LLC. The extension is focused on managing tasks that are given to the company employees by the management.

Keywords: Task, Management, Software, Application, System, Mobile, Analysis

Title translation: Web-based application for task management

Obsah

1 Úvod **1**
1.1 O firmě 1
1.2 O aplikaci 1
1.3 Motivace 2
1.4 Cíle 2
1.5 Vymezení pojmů 2

2 Analýza a návrh řešení **5**
2.1 Požadavky 5
2.1.1 Funkční požadavky 6
2.1.2 Nefunkční požadavky 9
2.2 Existující řešení 10
2.2.1 Todoist 10
2.2.2 Trello 12
2.2.3 Vyhodnocení 14
2.3 Případy užití 14
2.3.1 Systém 16
2.3.2 Internal aplikace 17
2.3.3 Mobile aplikace 18
2.3.4 Scénáře 18
2.4 Doménový model 23
2.5 Shrnutí analýzy 28

3 Implementace **29**
3.1 Vybrané technologie 29
3.1.1 Programovací jazyky 29
3.1.2 Vývojové prostředí 30
3.2 Struktura aplikace 30
3.2.1 Datová vrstva 31
3.2.2 Aplikační vrstva 31
3.2.3 Prezentační vrstva 31

4 Testování **33**
4.1 Popis uživatele 33
4.2 Metody testování použitelnosti 33
4.3 Testy použitelnosti 34
4.3.1 UC-1 Vytvořit Case 34
4.3.2 UC-5 Zobrazit Task 37
4.3.3 UC-6 Dokončit úkol 37
4.3.4 UC-10 Přihlásit odběr ke Case . . 38
4.4 Vyhodnocení testů 40
4.4.1 Priority 40
4.4.2 Nálezy 41

5 Závěr **43**

A Literatura **45**

B Návod na lokální spuštění **47**

C Datový model **49**

D Případy užití **57**

Obrázky

2.1 Todoist dashboard 11
2.2 Todoist filter 11
2.3 Todoist komentáře 12
2.4 Trello dashboard 12
2.5 Trello karta 13
2.6 Trello filter 13
2.7 Diagram případu užití 16
2.8 Diagram případu užití Internal aplikace 17
2.9 Diagram případu užití Mobile aplikace 18
2.10 Doménový model 24

4.1 Digram určování priorit nálezů 40

C.1 Diagram tříd 49

Tabulky

2.1 Existující řešení 14

4.1 Nálezy testování 41

Kapitola 1
Úvod

1.1 O firmě

Tato bakalářská práce vznikla na základě mé spolupráce se společnosti Curso s.r.o. Než začnu popisovat samotnou aplikaci, je třeba společnost v rychlosti představit. Společnost Curso s.r.o. poskytuje komplexní službu pro správu krátkodobých a dlouhodobých pronájmů bytů patřících majitelům, kteří jsou zákazníci společnosti. Tato služba mimo jiné obsahuje následující [1]:

- Tvorba profilu bytu
- Stanovení ceny
- Starání se o hosty
- Právní servis
- Přehled pobytů a výdělků

1.2 O aplikaci

Tato práce se bude zabývat návrhem a implementací rozšíření systému, které bude poskytovat nástroje na zefektivnění každodenní práce terénních zaměstnanců a také managementu. Rozšíření poskytne rozhraní pro zadání, správu, evidenci a report úkolů, které budou zadány management svým zaměstnancům.
Praktický příklad by bylo nastání události, která by bránila hladkému průběhu správy bytů např. nefungující topení na bytě. Potom terénní zaměstnanec založí v systému nový případ ve kterém popíše problém. Manažeři pro tento případ naplánují úkoly, které je třeba splnit k vyřešení tohoto případu a jednotlivé úkoly přidělí terénním zaměstnancům. Terénní zaměstnanci po splnění jím přiřazeným úkolům, reportují přes systém aktuální stav úkolů a manažeři uzavřou případ, pokud tak uznají za vhodné.

1.3 Motivace

Aplikací na spravování úkolů je spousty a většina z nich je dobře optimalizovaných. Provedl jsem rešerši těchto aplikací a porovnal je s požadavky společnosti. Stávající systém využívá určité datové struktury a integrace existujících aplikací se systémem tak, aby byly tyto struktury použitelné podle požadavků společnosti, byla časově náročnější než implementace vlastního řešení, které navíc nabízelo i neomezenou možnost přizpůsobení.

1.4 Cíle

Cílem této práce je analýza, návrh a implementace aplikace pro správu úkolů společnosti Curso s.r.o. V rámci analýzy provedeme rešerši existujících řešení a porovnáme je s požadavky společnosti. Podle analýzy dále implementujeme a otestujeme aplikaci. Tato aplikace bude splňovat požadavky společnosti. Hlavními aspekty této aplikace bude jednoduchost a provázanost se stávajícím systémem a jeho datovými strukturami.

1.5 Vymezení pojmů

Booking

Rezervace a uskutečněný pobyt na bytě majitele, který je zákazník společnosti.

Case

Událost, kterou je třeba vyřešit ke správnému chodu správy bytu. Tato událost může mít několik úkolů, které je třeba splnit.

Listing

Byt majitele, který je klient společnosti.

Locality

Oblasti města Prahy, kde se vyskytují byty, které společnost spravuje.

Manager

Uživatel, který má přístup do částí aplikace pro manažerskou pozici.

Operative

Uživatel, který má přístup do částí systému pro pozici terénního zaměstnance.

Subtask

Podúkol obsažen v úkolech.

Task

Úkol týkající se správy bytu zákazníka společnosti, kterou je třeba vykonat. Tento úkol může být složen z několika podúkolů.

User

Uživatel, který má přístup do aplikace přes své přihlašovací údaje podle své pozice.

Kapitola 2
Analýza a návrh řešení

Nedílnou součástí všech softwarových řešení je analýza a návrh. Na jejich základě jdou určit požadavky na systém, odhalit úskalí a hranice systému a také slouží jako podklad pro developery, který jim usnadní práci s programováním.
Diagramy v analýze budu modelovat pomocí UML [2] (Unified Modeling Language), což je standardizovaný modelovací jazyk, který obsahuje nástroje pro vytváření modelů, které usnadňují tvorbu softwarových řešení. Tyto diagramy slouží k sestavení a vizualizaci návrhu systému.
Řešením bude rozšíření již existujícího systému tak, aby vyhovovalo požadavkům společnosti. Stávající systém slouží k evidenci a správě bytů, pobytů, vyúčtování, zaměstnanců, výkazů atd.
Řešení by mělo tyto funkce rozšířit tak, aby systém byl schopný spravovat a evidovat *Case*, které je třeba řešit a *Task* zadané managementem, které musí zaměstnanci splnit.
Rozšíření bude navrženo a implementováno do dvou hlavních modulů na klientské straně aplikace, které budou mít jednu společnou serverovou stranu. Tyto moduly jsou:

- **Internal aplikace** Klientská strana aplikace určena pro *Manager*.
- **Mobile aplikace** Klientská strana aplikace určena pro *Operative*.

2.1 Požadavky

Analýza požadavků [2] je proces definování uživatelských očekávání od daného softwarového řešení. Požadavek [2] je schopnost systému, která je vyžádaná uživatelem a která dokáže řešit požadované úkoly.

Tyto požadavky slouží nejen jako výčet a popis, ale také jako podklad pro programátory k implementaci. Katalog by měl být jasný a srozumitelný pro zadavatele i zprostředkovatele.

Požadavky budou mít tyto náležitosti:

- **Identifikátor** Ve tvaru FP-X pokud se jedná o funkční požadavek a NP-X pokud se jedná o nefunkční požadavek, kde X je pořadové číslo

• **Název**

• **Popis**

Požadavky se budou dále dělit do dvou modulů. Za normálních okolností bych udělal podkapitoly Manager, Operative a pod ně psal jednotlivé požadavky, ale kvůli existenci společných požadavků jsem se rozhodl jednotlivé požadavky dělit na User, Manager nebo Operative.

2.1.1 Funkční požadavky

Funkční požadavky [2] specifikují jaké úkoly bude systém schopný řešit. Popisují chování požadované funkce v systému a reakci na jistou událost. Měly by mít popsané pro koho slouží a čeho daný úkol docílí. Tyto požadavky slouží jako podklad pro případy užití.

FP-1 Přihlášení

Manager *Manager* se po zadání svých správných přihlašovacích údajů dostane na domovskou stránku *Internal aplikace*.

Operative *Operative* se po zadání svých správných přihlašovacích údajů dostane na domovskou stránku *Mobile aplikace*.

FP-2 Domovská stránka

Manager Pro *Manager* bude domovská stránka přehled *Case*, které mu jsou přiřazené k dokončení.

Operative Pro *Operative* bude domovská stránka přehled *Case*, které vytvořil, vlastní, alespoň jeden z *Task* daného *Case* mu je přiřazený nebo je *Operative* odběratelem.

FP-3 Case správa

User *User* bude mít k dispozici přehled všech *Case*. Přehled půjde filtrovat podle *CaseState* a půjde v něm vyhledávat podle *id* a *name*.
Jednotlivé *Case* budou mít svou detailní stránku, která bude obsahovat základní informace jako je *name, description, deadline, priority* atd. Dále bude obsahovat nahrané *Photo*, *Comment* a seznam *Task* spojené s daným *Case*.
User bude moct vytvářet nové *Case*, upravovat již existující *Case*, nahrávat/mazat *Photo* a přidávat *Comment* pro daný *Case*.

Manager Přehled půjde seřadit podle viditelných atributů.
Položky přehledu budou obsahovat *id*, *name*, *Priority*, *CaseState*, *Listing*, *Booking*, *owner*, *assignee*, *deadline*.
Manager bude možnost měnit *CaseState*.

Operative *Operative* bude mít k dispozici přehled *Case*, které jsou relevantních pro daného *Operative* tzn. daný *Operative* je *assignee, owner, createdBy* nebo jeden z *followers* pro daný *Case* nebo je *assignee* na jednom z *Task* na daném *Case*. Přehled bude seřazený sestupně podle data vytvoření a půjde v něm vyhledávat podle *id* a *name*. Položky přehledu budou obsahovat *name*, *Priority*, *CaseState* a *Listing*.
Dále se *Operative* bude moct přihlásit a zrušit odběr. Odběr umožní *Operative* dostávat notifikace o změnách na daném *Case*.

FP-4 Task správa

User *User* bude mít k dispozici *Task* přehled. Přehled bude rozdělen na dokončené a nedokončené *Task*.
Jednotlivé *Task* budou mít svou detailní stránku, která bude obsahovat základní informace jako je *name, description, deadline, priority* apod. Dále bude obsahovat seznamy *Subtask*, *Comment* a *Photo* spojené s daným *Task*.
User bude dále moct nahrávat/mazat *Photo* a přidávat *Comment*.

Manager Přehled půjde seřadit podle viditelných atributů
Položky přehledu budou obsahovat *id*, *name*, *Priority*, *Case*, *Locality*, *Listing*, *assignee*, *deadline*.
Manager může *Task* dokončit pokud *Task* není ještě dokončený.
Manager bude moct vytvářet nové *Task*, upravovat již existující *Task* a přidávat *Subtask* k *Task*.

Operative Na záložce pro nedokončené bude přehled seskupený a seřazený sestupně podle *deadline* a pak v rámci *deadline* skupiny podle *Locality*. Na záložce pro dokončené bude přehled seřazený sestupně podle *finishedOn* a bude nabízet možnost vyhledání podle *id* a *name*.
Položky přehledu budou obsahovat *name*, *Priority*, *Listing* a tlačítko na dokončení *Task*, pokud možno.
Operative může *Task* dokončit pokud je jeho *assignee*, *Task* není ještě dokončený a všechny *Subtask* daného *Task* mají *SubtaskState* jiný než *UNBEGUN*.

FP-5 Subtask správa

User *User* bude mít přehled *Subtask* u daného *Task*
Dále bude moct měnit *SubtaskState*.

Manager *Manager* bude moct *Subtask* vytvořit a upravit jej.

FP-6 Activity přehled

User *User* bude mít k dispozici *Activity* přehled, který bude obsahovat detailní popis dané *Activity*, *name, id, createdOn, createdBy* a *Assignable*, které se *Activity* týká.
Přes tuto *Activity* se *User* dostane na stránku detailu daného *Assignable*.

Operative *Operative* bude dostávat notifikace o nových *Activity*, které jsou pro něj relevantní.

FP-7 Listing správa

Manager *Manager* bude mít k dispozici *Listing* přehled. Přehled bude seřaditelný podle viditelných atributů a bude obsahovat id, name, address.
Jednotlivé *Listing* budou mít detailní stránky, které budou obsahovat základní informace a *Booking* přehled vázaný na daný *Listing*.
Dále moct *Listing* vytvářet a upravovat.

FP-8 Booking správa

Manager *Manager* bude mít k dispozici *Booking* přehled. Přehled bude seřaditelný podle viditelných atributů a bude obsahovat id, guestName, *Listing*, start, end.
Jednotlivé *Booking* budou mít detailní stránky, které budou obsahovat základní informace o *Booking*.
Dále moct *Booking* vytvářet a upravovat.

FP-9 Locality správa

Manager *Manager* bude mít k dispozici *Locality* přehled. Přehled bude seřaditelný podle viditelných atributů a bude obsahovat id, name.
Jednotlivé *Locality* budou mít detailní stránky, které budou obsahovat základní informace o *Locality*.
Dále moct *Locality* vytvářet a upravovat.

FP-10 Contact správa

User *User* bude mít k dispozici *Contact* přehled, který bude obsahovat *name, phone* pro aktuálně odpovědného *User* za dané oddělení.

Manager *Manager* bude mít k dispozici *Contact* detaily, které budou obsahovat základní informace a bude moct *Contact* vytvářet a upravovat.

FP-11 User správa

Manager *Manager* bude mít k dispozici *User* přehled, který bude obsahovat *id, firstname, lastname, username*. Přehled půjde seřadit podle viditelných atributů.
Jednotlivé *User* budou mít svou detailní stránku, která bude obsahovat základní informace jako je *firstname, lastname, username a phone*.
Manager bude moct *User* vytvářet a upravovat.

FP-12 Notifikace

Operative Při změně na *Case* nebo *Task* se do 5 minut zobrazí notifikace o nové *Activity* všem *User*, kteří jsou *assignee, createdBy* nebo v případě *Case* jsou *owner* nebo jedni z jejich *followers*.

2.1.2 Nefunkční požadavky

Nefunkční požadavky [2] specifikují jak by měl systém úkoly řešit. Popisují jak se systém chová a jaké jsou hranice jeho funkcionality. Měly by popisovat obecné charakteristiky systému.

NP-1 Integrace

Operative *Mobile aplikace* bude zintegrovaná se stávajícím systémem a bude schopna pracovat s existujícími daty v systému

Manager *Internal aplikace* bude rozšířena tak, aby nenarušila procesy stávajícího systému.

NP-2 Design

Operative *Mobile aplikace* bude vzhledem přizpůsobena převážně pro prohlížeče mobilních zařízení.

Manager *Internal aplikace* bude vzhledem přizpůsobena převážně pro prohlížeče počítačových zařízení.

NP-3 Doba zpracování požadavků

User Aplikace bude schopna zpracovat uživatelské požadavky do 1 sekundy.

NP-4 Reakce aplikace na požadavky

User Aplikace bude reagovat na uživatelské požadavky tak, aby uživatel dokázal vyhodnotit reakci aplikace a rozhodnout se jak dále postupovat.

NP-5 Bezpečnost

User Aplikace neumožní přístup k jednotlivým stránkám a informacím obsažených na těchto stránkách bez úspěšného přihlášení uživatele.

NP-6 Změna dat v databázi

User Po změně dat v databázi bude tato změna dostupná všem uživatelům do 2 sekund.

NP-7 Podporované prohlížeče

User Aplikace bude fungovat na prohlížečích od verzí: Google Chrome 70, Mozilla Firefox 63, Internet Explorer 16, Microsoft Edge 10, Safari 11.

NP-8 Rozšířitelnost

User Aplikace bude rozšířitelná o další funkcionality.

NP-9 Počet uživatelů

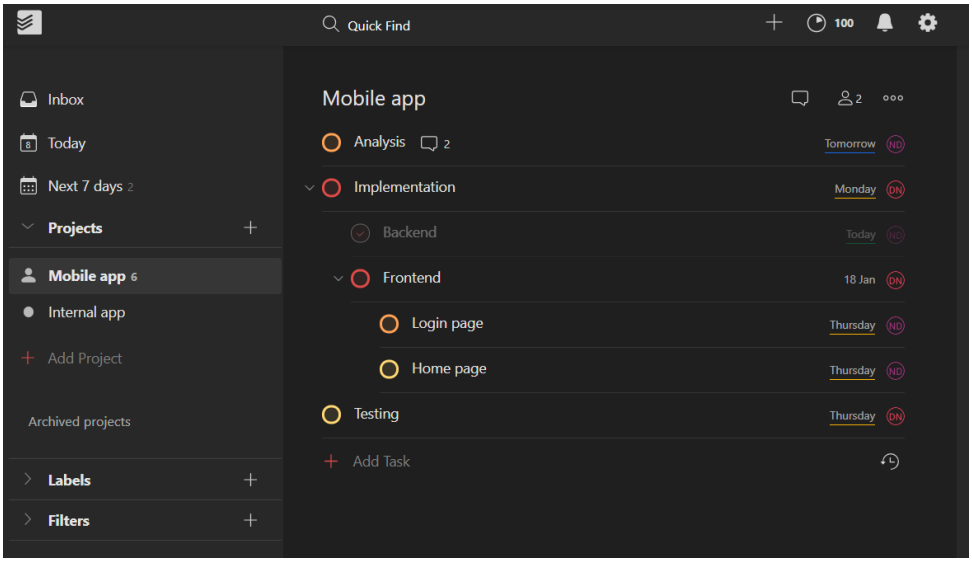
User Aplikace bude schopna provozu se 100 aktivními uživateli.

2.2 Existující řešení

Společnost potřebuje jednoduchý a přehledný systém na správu úkolů, který bude integrovatelný se stávajícím systémem. Systémů vyhovujících těmto kritériím je mnoho a mezi nimi vynikají dvě, které podrobněji popíši.

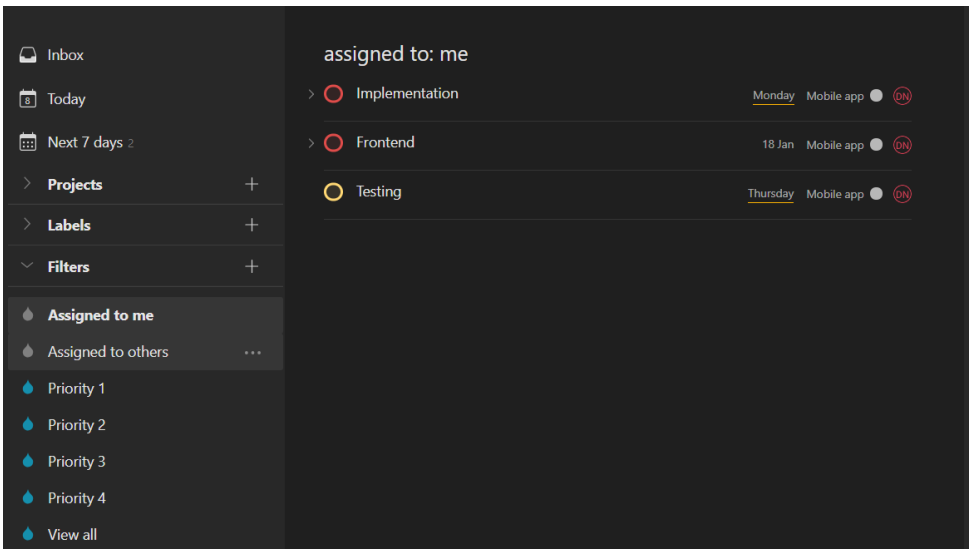
2.2.1 Todoist

Na Todoist mě nejvíce zaujala jednoduchost a intuitivnost používání aplikace. Aplikace má přehledný dashboard viz *Obrázek 2.1* a skoro všechny základní funkce pro task management.
Úkoly lze strukturovat do projektů a podúkolů a lze přidávat členy týmů do projektů. Úkolům lze přiřazovat priority, deadline, komentáře a člena který úkol splní viz *Obrázek 2.1*.



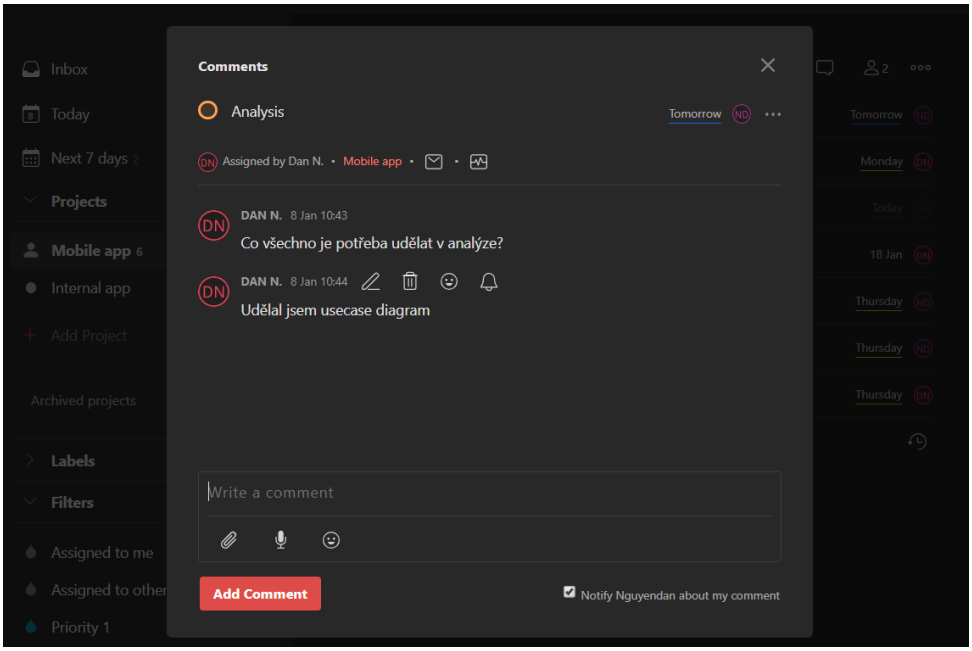
Obrázek 2.1: Todoist dashboard

Další užitečná funkce je filtr úkolů viz *Obrázek 2.2*. Tento filtr vytřídí úkoly podle člena, který jej má splnit nebo podle priority.



Obrázek 2.2: Todoist filter

Jedna věc co mi opravdu chybí v základních funkcí je popis jednotlivých úkolů. Toto lze nahradit komentáři viz *Obrázek 2.3*, ale ve větším množství komentářů se popis snadno ztratí.

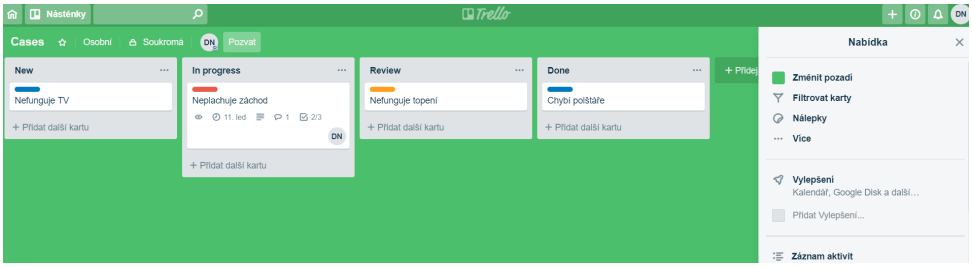


Obrázek 2.3: Todoist komentáře

Placená verze, která stojí 130 Kč na uživatele na měsíc, dále nabízí nahrávání fotek s komentáři, vlastní filtry, tagy a archív aktivit na jednotlivých úkolech.

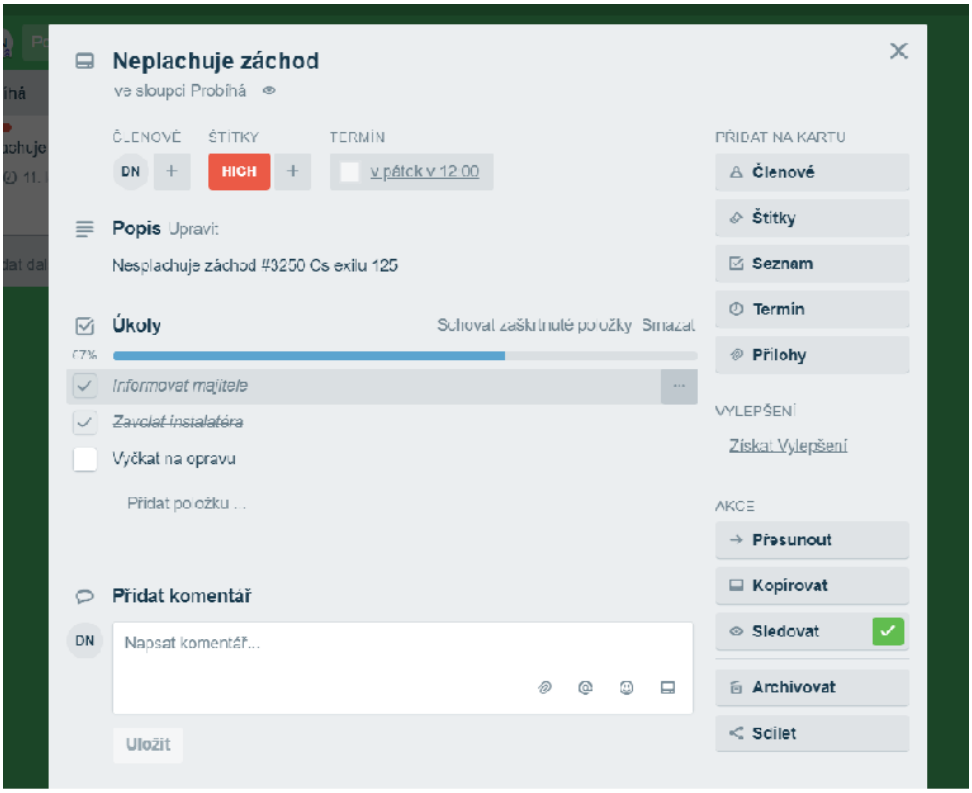
2.2.2 Trello

Na Trello mě nejvíce zaujalo do jaké míry si může uživatel přizpůsobit pracovní prostor. Stejně jako u Todoist, Trello je velice jednoduchá a intuitivní aplikace na používání. Aplikace má přehledný dashboard viz *Obrázek 2.4*, který lze rozdělit do různých sloupců podle stavu a skoro všechny základní funkce pro task management.



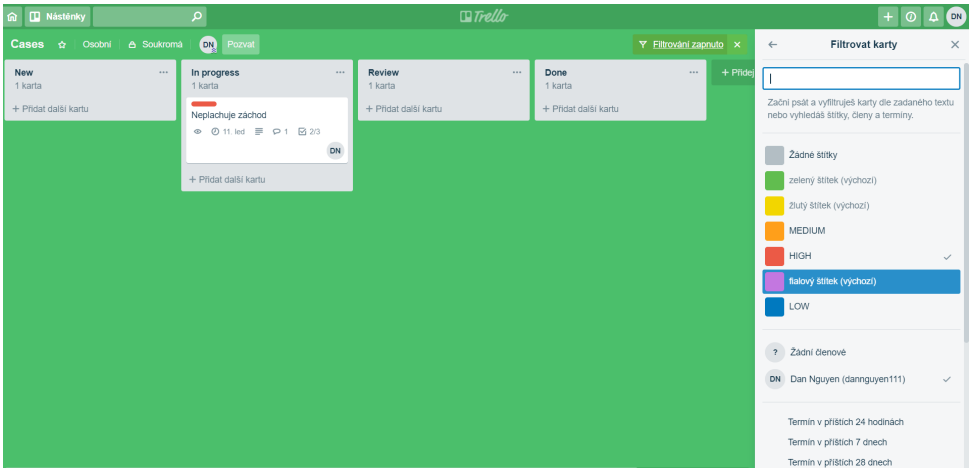
Obrázek 2.4: Trello dashboard

Aktivity ke splnění se nazývají **karty**, které lze strukturovat do nástěnek a lze k nim přidávat seznamy úkolů. Úkolům lze přiřazovat štítky, deadline, fotky, komentáře a člena který úkol splní viz *Obrázek 2.5*.



Obrázek 2.5: Trello karta

Další užitečná funkce je filtr úkolů viz *Obrázek 2.6*. Tento filtr vytřídí karty podle člena, který jej má splnit, štítků a deadline.



Obrázek 2.6: Trello filter

Na aplikaci mi chybí možnost přiřazení priority ke kartám. Toto lze nahradit štítky viz *Obrázek 2.6*, ale to nebrání uživateli takto přiřadit kartě více priorit než jednu. Je sice pěkné, že ke kartám můžeme přidat seznamy s úkoly, ale potřeboval bych i možnost k úkolům přidat podúkol, což už aplikace neumožňuje.

Placená verze, která stojí 234 Kč na uživatele na měsíc, umožní uživateli přidávat další pluginy do aplikace.

2.2.3 Vyhodnocení

Aplikace	Výhody	Nevýhody
Todoist	- Jednoduchost - Lze tvořit struktury úkolů a podúkolů	- Nelze přidat detailní popis úkolů
Trello	- Jednoduchost - Přehlednost karet a jejich stavů	- Nelze přidat prioritu úkolům
ToodleDo	- Přizpůsobitelné sloupce v seznamu	- Chybí rozdělení úkolů a podúkolů - Nelze přidávat komentáře a fotky k úkolům
Jira	- Historie změn na úkolech	- Přebytek funkcí zaměřených na vývoj softwarů jako je přiřazení sprintů apod.

Tabulka 2.1: Existující řešení

Existuje mnoho dalších řešeních kromě výše zmíněných jako jsou Wunderlist, ToodleDo, apod. Potom jsou takové, které jsme zavrhli kvůli složitému uživatelskému rozhraní, nepřehlednosti, přebytku funkcí nebo nedostatku funkcí jako jsou Jira, Clickup apod.

I přes tyto existující aplikace bylo zvoleno vlastní řešení dělané na míru požadavkům společnosti. Hlavním důvodem je náročnost integrace výše zmíněných aplikací s datovým strukturami, které jsou používané ve stávajícím systému společnosti jako jsou např. *Listing, Booking, Locality* a také omezenými možnostmi přizpůsobení dané aplikace.

2.3 Případy užití

Pro vizualizaci a usnadnění návrhu řešení jsem použil Use Case diagram [2] viz *Obrázek 2.9* a *Obrázek 2.8*.
Use Case diagram [2] popisuje sadu aktivit, které systém společně s externími uživateli vykoná. Diagram má následující elementy a vztahy mezi nimi [2]:

- **Případ užití** Specifikuje jakým způsobem budou uživatelé používat systém, aby splnili požadovaný úkol. Jsou odvozeny z funkčních požadavků.
- **Předmět** Je systém nebo část systému, která má definované určité chování, podle kterého se *případ užití* řídí
- **Aktér** Externí entita (v mém případě *User*), která využívá *předmět* ke splnění *případu užití*
- **Asociace** Vztah popisující interakci *aktéra* v daném *případu užití*
- **Include** Vztah mezi dvěma *případy užití*. Kroky z *případu užití*, do které *include* vchází (zdroj), jsou vloženy do *případu užití*, ze kterého include vychází (cíl). Zdrojový *případy užití* nemůže samostatně bez této vazby existovat.
- **Extends** Vztah mezi dvěma *případy užití*. Kroky z *případu užití*, do které *include* vchází (zdroj), jsou vloženy do *případu užití*, ze kterého include vychází (cíl). Zdrojový *případy užití* může samostatně bez této vazby existovat.

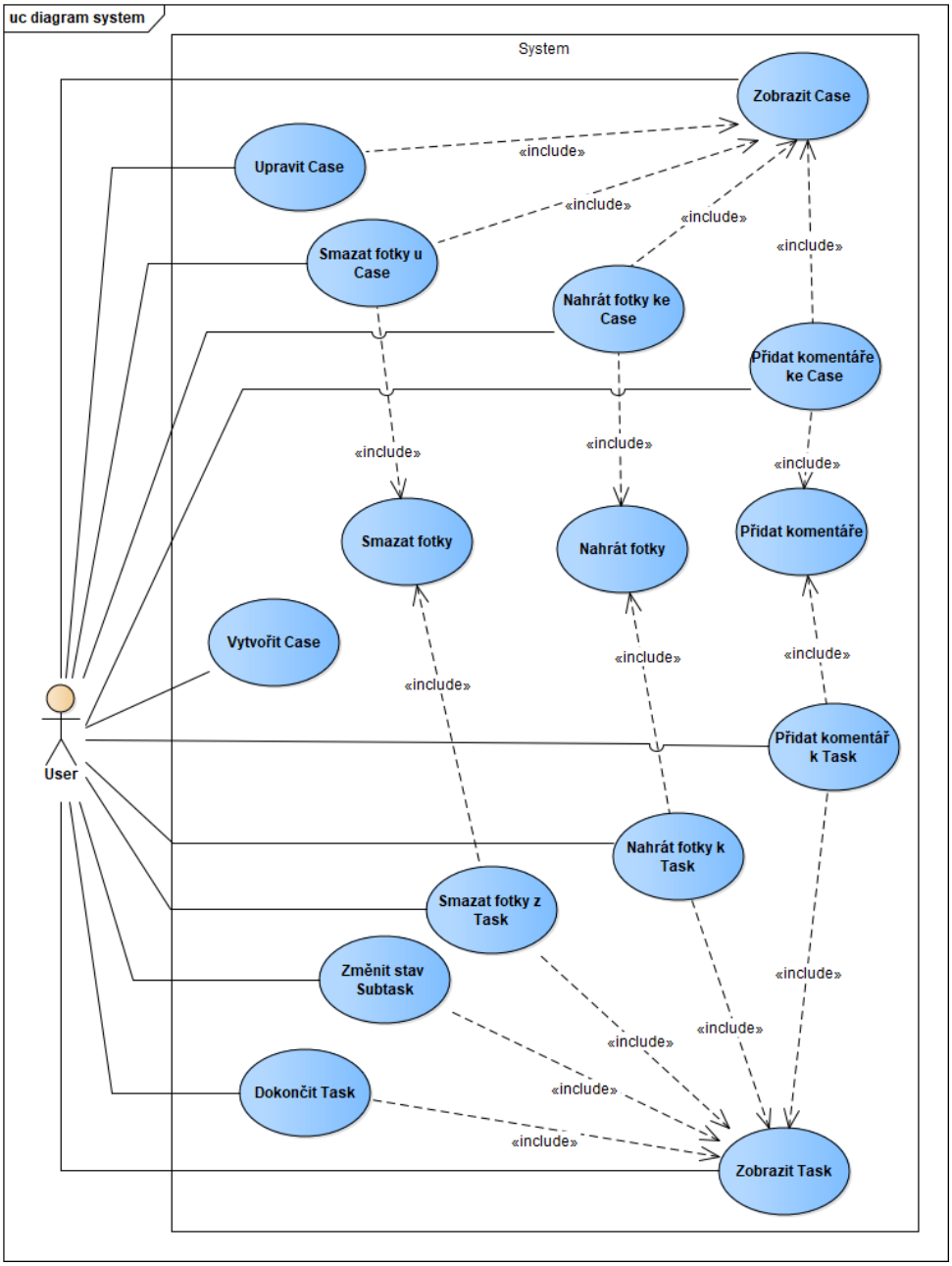
Dále má každý *případ užití* popsané tyto náležitosti:

- **Identifikátor** Ve tvaru UC-X, kde X je pořadové číslo případu užití.
- **Název**
- **Popis**
- **Aktéři** Jeden nebo více účastníků, kteří konají *případ užití*
- **Spoušť** Událost, která způsobí zahájení *případu užití*
- **Vstupní podmínky** Počáteční stav před začátkem *případu užití*
- **Výstupní podmínky** Cílový stav po dokončení *případu užití*

Use Case diagram [2] jsem pro přehlednost rozdělil na tři části, protože aktéři sdílí většinu případů užití, ale operují v jiných modulech. *Operative a Manager* můžou používat všechny případy užití co *User*.

2.3.1 Systém

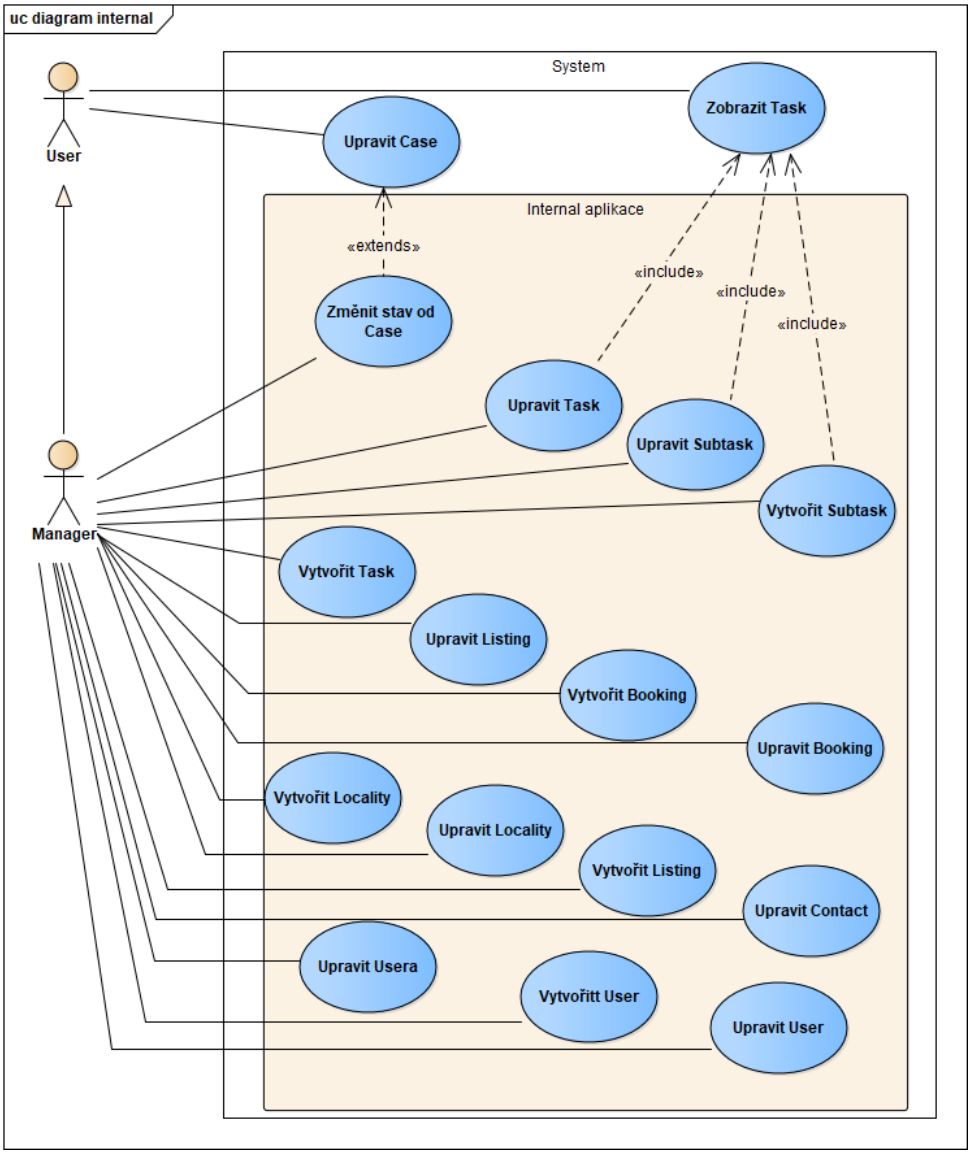
V *případech užití* pro celý *sysém* viz *Obrázek 2.7* je *aktér User*. To znamená, že všechny *případy užití* uvedné v tomto diagramu mohou používat *Operative i Manager*.



Obrázek 2.7: Diagram případu užití

2.3.2 Internal aplikace

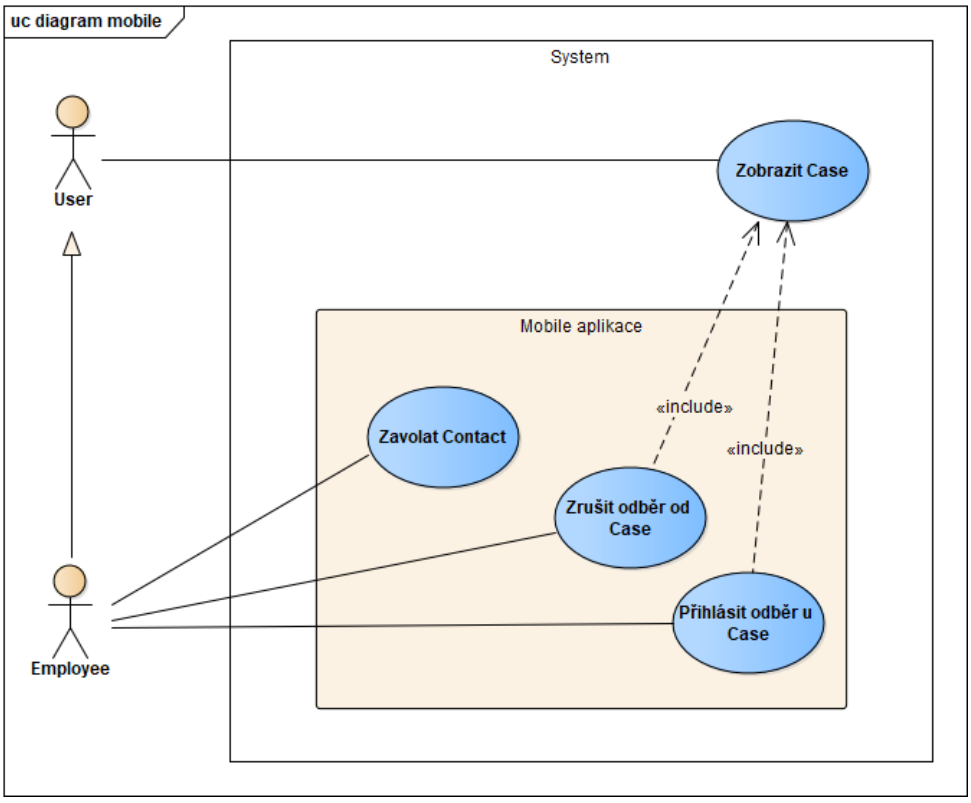
V *případech užití* pro *internal aplikaci* viz *Obrázek 2.8* je *aktér Manager*, který jako jediný má přístup do tohoto modulu.



Obrázek 2.8: Diagram případu užití Internal aplikace

2.3.3 Mobile aplikace

V *případech užití* pro *mobile aplikaci* viz *Obrázek 2.9* je *aktér Operative*, který jako jediný má přístup do tohoto modulu.



Obrázek 2.9: Diagram případu užití Mobile aplikace

2.3.4 Scénáře

V této kapitole detailně popisuji scénáře hlavních *případů užití*. Všechny detailně popsané *případy užití* viz *Příloha D*.

UC-1 Vyvořit Case

Aktéři	*User*
Popis	*User* vytvoří nový *Case* na základě události spojenou s *Listing*.
Spoušť	*User* chce založit *Case* na základě události, kterou je třeba řešit.
Vstup	*User* je přihlášen a nachází se na domovské stránce.

Výstup *Case* je vytvořen a uložen v databázi. *User* se nachází na stránce *Case* detailu.

Hlavní scénář

1. Uživatel přejde na stránku pro Case vytvoření.
2. Systém zobrazí stránku pro Case vytvoření.
3. Uživatel vyplní formulářové pole
4. Uživatel odešle formulář
5. Systém uloží Case do databáze
6. Systém přesměruje stránku na Case detail

Vedlejší scénáře

5a Pokud uživatel nevyplnil všechny povinné formulářové pole, tak ho systém o to požádá a změny neuloží. Případ užití se vrací na krok 3.

UC-2 Zobrazit Case

Aktéři *User*

Popis *User* vyhledá *Case* podle atributů, které zná. Tyto atributy můžou být *id, name* nebo *CaseState*.

Spoušť *User* chce vyhledat *Case*, aby s ním mohl dále pracovat.

Vstup *User* je přihlášen a nachází se na domovské stránce.

Výstup *User* nalezl *Case* a nachází se na stránce *Case* detailu.

Hlavní scénář

1. *User* vyfiltruje *Case* přehled podle *CaseState*.
2. Systém zobrazí *Case* přehled podle vyfiltrovaného *CaseState*.
3. *User* zadá do vyhledávání *id* nebo *name* daného *Case*.
4. Systém zobrazí všechny *Case*, které splňují vyhledávací kriteria.
5. *User* přejde na detail hledaného *Case*.
6. Systém zobrazí *Case* detail.

Vedlejší scénáře

3a *User* nezná *id* ani *name* a nalezne hledaný *Case* scrollováním přehledu.

3b *User* nezná *id* ani *name* a dál nehledá. Případ užití tímto končí.

UC-3 Změnit stav od Case

Aktéři *Manager*

Popis *Manager* změní *CaseState* u daného *Case*.

Spouští *Manager* chce změnit *CaseState* u daného *Case*.

Vstup Tento *případ užití* rozšiřuje (**UC-4** Case úprava). *Manager* jej iniciuje.

Výstup *Manager* změnil *CaseState* u daného *Case*.

UC-4 Case úprava

Aktéři *User*

Popis *User* upraví některé z *Case* následujících atributů: *name, description, Listing, Booking*.

Spouští *User* chce upravit atributy u daného *Case*.

Vstup *User* je přihlášen a nachází se na domovské stránce. *User* je *owner* nebo *createdBy* pro daný *Case*.

Výstup *User* upravil vybrané atributy u daného *Case*

Hlavní scénář

1. *User* najde daný *Case* detail (**UC-2**)
2. *User* zvolí možnost úpravy na daném *Case*
3. *User* upraví vybrané atributy a pokud chce tak i *CaseState* (**UC-3**)
4. *User* potvrdí změny
5. Systém uloží změny
6. Systém se vrátí na stránku *Case* detail

Vedlejší scénáře

4a *User* se vrátí zpět na *Case* detail. Případ užití se vrací na krok 2.

5a *User* nechal povinné údaje prázdné, tak ho systém vyzve k úpravě chyb. Případ užití se vrací na krok 3.

UC-5 Zobrazit Task

Aktéři *User*

Popis *User* vyhledá *Task* podle atributů, které zná. Tyto atributy můžou být *id, name* nebo *finished*.

Spouští *User* chce vyhledat *Task*, aby s ním mohl dále pracovat.

Vstup *User* je přihlášen a nachází se na domovské stránce. *User* je *assignee* hledaného *Task*.

Výstup *User* nalezl *Task* a nachází se na stránce *Task* detailu.

Hlavní scénář

1. *User* přejde na stránku s *Task* přehledem
2. Systém zobrazí Task přehled.
3. *User* vyfiltruje *Task* přehled podle atributu *finished*.
4. Uživatel pomocí seskupení přehledu podle deadlinu a místa konání nalezne hledaný Task
5. *User* přejde na detail hledaného *Task*.
6. Systém zobrazí *Task* detail.

Vedlejší scénáře

3a *Task* se nachází v přehledu dokončených *Task*, tak se uživatel pokusí najít *Task* pomocí *id* nebo *name*.

3b *Task* se nachází v přehledu dokončených *Task* a uživatel nezná *id* ani *name* tak se pokusí najít Task scrollováním.

3c *Task* se nachází v přehledu dokončených *Task*, uživatel nezná *id* nebo *name* a už dál nehledá. Případ užití tímto končí.

UC-6 Dokončit Task

Aktéři *User*

Popis *User* reportuje, že *Task* je hotov tím, že ho v systému jako *finished*.

Spoušť *User* chce reportovat dokončení *Task*.

Vstup *User* je přihlášen a nachází se na domovské stránce. *User* je *assignee* daného *Task*, který ještě není dokončený. *Task* má všechny *Subtask* v jiném *SubtaskState* než *UNBEGUN* nebo neobsahuje *Subtask*.

Výstup *User* označil *Task* jako *finished* a nachází se na stránce *Task* detailu. Systém uložil změny do databáze.

Hlavní scénář

1. *User* přejde na stránku s *Task* přehledem
2. Systém zobrazí *Task* přehled.
3. *User* najde daný *Task* detail (**UC-5**)
4. *User* zvolí možnost dokončení *Task*.

Další

Plné znění těchto *případů užití* viz *Příloha D*.

- **UC-7 Nahrání fotky** *User* nahraje *Photo* k danému *Assignable*, aby přiblížil ostatním situaci.
- **UC-8 Smazat fotky** *User* smaže *Photo* z daného *Assignable*.
- **UC-9 Přidat komentář** *User* okomentuje *Assignable*.
- **UC-10 Přihlásit odběr ke Case** *Operative* bude odebírat notifikace o změnách na daném *Case*.
- **UC-11 Case zrušení odběru** *Operative* přestane odebírat notifikace o změnách na daném *Case*.
- **UC-12 Nahrát fotky ke Case** *User* nahraje *Photo* k danému *Case*, aby přiblížil ostatním situaci.
- **UC-13 Smazat fotky z Case** *User* smaže *Photo* z daného *Case*.
- **UC-14 Přidat komentář ke case** *User* okomentuje *Case*.
- **UC-15 Nahrát fotky k Task** *User* nahraje *Photo* k danému *Task*, aby přiblížil ostatním situaci.
- **UC-16 Smazat fotky z Task** *User* smaže *Photo* z daného *Task*.
- **UC-17 Přidat komentář k Task** *User* okomentuje *Task*.
- **UC-18 Zavolat Contact** *Operative* zavolá danému *Contact*.
- **UC-19 Vytvořit Task** *Manager* vytvoří nový *Task*.
- **UC-20 Upravit Task** *Manager* upraví atributy u *Task*.
- **UC-21 Vytvořit Subtask** *Manager* vytvoří nový *Subtask* u daného *Task*.
- **UC-22 Upravit Subtask** *Manager* upraví atributy u *Subtask*.
- **UC-23 Vyvořit Listing** *Manager* vytvoří nový *Listing*.
- **UC-24 Upravit Listing** *Manager* upraví atributy u *Listing*.
- **UC-25 Vytvořit Booking** *Manager* vytvoří nový *Booking*.
- **UC-26 Upravit Booking** *Manager* upraví atributy u *Booking*.
- **UC-27 Vytvořit Locality** *Manager* vytvoří nový *Locality*.
- **UC-28 Upravit Locality** *Manager* upraví atributy u *Locality*.
- **UC-29 Vytvořit User** *Manager* vytvoří nový *User*.

- **UC-30 Upravit User** *Manager* upraví atributy u *User*.
- **UC-31 Vytvořit Contact** *Manager* vytvoří nový *Contact*.
- **UC-32 Upravit Contact** *Manager* upraví atributy u *Contact*.

2.4 Doménový model

Doménový model [18] popisuje data, jejich strukturu a vztahy mezi nimi. Vizualizaci doménového modelu [18] provedu pomocí diagramu viz *Obrázek 2.10*, který obsahuje následující elementy a vztahy mezi nimi [2]:

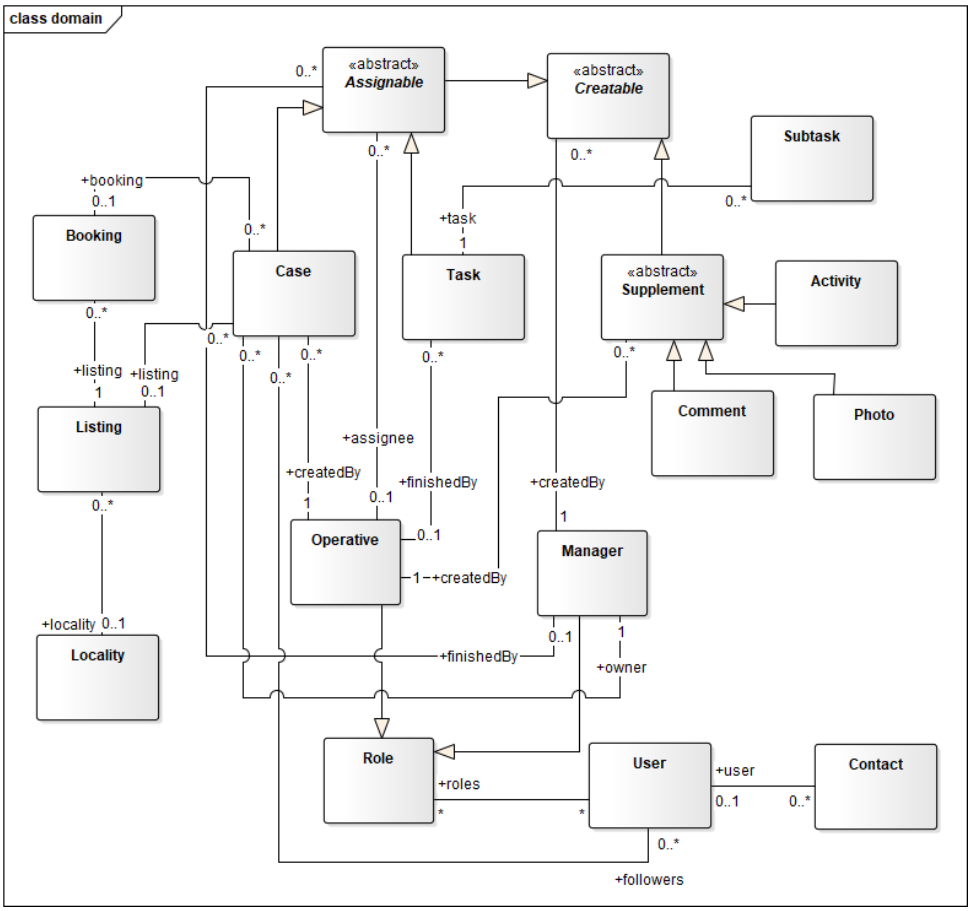
- **Třída** Je datová struktura, která popisuje všechny objekty, které sdílí společné atributy, vztahy, pravidla a operace. Tyto objekty jsou instance dané *třídy*.
- **Abstraktní třída** *Třída*, ze které ale nejdou tvořit instance. Slouží k tomu, aby ostatní *třídy* z ní mohli dědit její vztahy, operace a atributy.
- **Výčtový typ** Datový typ s výčtem hodnot, které jsou používány v instancích *tříd*.
- **Multiplicita** Určuje počet elementů vstupujících do vztahu.
- **Asociace** Vztah mezi dvěma *třídami*, popisující interakci mezi jejich instancemi.
- **Agregace** Druh *asociace*, který značí, že instance *třídy* je část daného celku, který je instance *třídy*. Část dokáže nezávisle na celku existovat.
- **Kompozice** Druh *asociace*, který značí, že instance *třídy* je část daného celku, který je instance *třídy*. Část nemůže existovat nezávisle na celku.
- **Generalizace** Vztah mezi dvěma *třídami*. Zdroj nazývám subclass a cíl nazývám superclass. Superclass je obecnější a subclass, která je specifičtější z ní dědí operace, atributy a vztahy.

Každá dále popsaná *třída* nebo *výčtový typ* bude obsahovat:

- Název
- Typ - Může být *třída, abstraktní třída, výčtový typ*
- Dědičnost - *Třída*, ze které element dědí vztahy, operace a atributy, pokud taková je.
- Popis
- Atributy

- Vztahy

Pro tuto práci použiji doménový model viz *Obrázek 2.10*. Z tohoto doménového modelu odvodím diagram tříd viz *Příloha C*, protože nejlépe a do dostatečného detailu zachycuje objektově orientovaný přístup v programování, pomocí kterého budu aplikaci implementovat.



Obrázek 2.10: Doménový model

Hlavní *třídy* v této kapitole do detailu popíšu. Diagram tříd odvozený z doménového modelu, detailní popis všech *tříd* a *výčtových typů* viz *Příloha C*.

Creatable

Typ: Abstraktní třída.

Popis: Třídy odvozené z Creatable jsou ty, které mají evidovat datum a čas vytvoření instance a *User*, který ji vytvořil.

Atributy: **createdOn** je datum vytvoření

Vztahy: **createdBy** je *User*, který instanci vytvořil

Assignable

Typ: Abstraktní třída.

Dědičnost: Odvozená z *Creatable*

Popis: Třídy odvozené z Assignable jsou činnosti, které jsou přiřaditelné ke splnění určitému *User*, mají deadline a evidují datum splnění.

Atributy:
assignedOn je datum přiřazení činnost určitému *User*
deadline je nejpozdější datum, do kterého je třeba činnost splnit
finishedOn je datum, kdy byla činnost splněna
name je název činnosti
description je popis činnosti
createdOn

Vztahy:
assignee je *User*, kterému byla činnost přidělena
finishedBy je *User*, kterým byla činnost splněna
priority je *Priority* činnosti
createdBy

User

Typ: Třída.

Popis: Uživatel aplikace.

Atributy:
id je identifikátor
firstname je jméno
lastname je příjmení
username je přihlašovací jméno
password je heslo
phone je telefonní číslo

Vztahy:
roles je seznam rolí daného uživatele, které mohou být Operative nebo Manager

Case

Typ: Třída.

Dědičnost: Odvozená ze třídy *Assignable*

Popis: Je událost, kterou je třeba vyřešit ke správnému chodu správy bytu. Tato událost může mít několik *Task*, které je třeba splnit.

Atributy:
id je identifikátor

createdOn

assignedOn

deadline

finishedOn

name

description

Vztahy: **state** je *CaseState*, ve kterém se Case nachází

followers je seznam *User*, kterým přichází notifikace o změnách na daném Case

booking je *Booking*, ke kterému se Case vztahuje

listing je *Listing*, ke kterému se Case vztahuje

createdBy

assignee

finishedBy

priority

Task

Typ: Třída.

Dědičnost: Odvozená ze třídy *Assignable*

Popis: Úkol týkající se správy bytu zákazníka společnosti, kterou je třeba vykonat. Tento úkol může být složen z několika *Subtask*.

Atributy: **id** je identifikátor

finished je ukazatel, jestli je úkol dokončen nebo ne

createdOn

assignedOn

deadline

finishedOn

name

description

Vztahy: **case** je *Case*, ke kterému je Task přiřazený

createdBy

assignee

finishedBy

priority

Subtask

Typ: Třída.

Popis: Podúkol, který může *Task* mít

Atributy:
- **id** je identifikátor
- **name** je datum název

Vztahy:
- **task** je *Task*, ke kterému se Subtask vztahuje
- **sate** je *SubtaskState*, ve kterém se Subtask nachází

Listing

Typ: Třída.

Popis: Byt zákazníka společnosti.

Atributy:
- **id** je identifikátor
- **address** je adresa bytu
- **name** je zobrazený název bytu

Vztahy:
- **locality** je *Locality*, kde se byt nachází

Booking

Typ: Třída.

Popis: Rezervace nebo uskutečněný pobyt na bytě majitele, který je zákazník společnosti.

Atributy:
- **id** je identifikátor
- **start** je datum začátku pobytu
- **end** je datum konce pobytu
- **guestName** je jméno hosta
- **canceled** popisuje jestli je nebo byl pobyt zrušen

Vztahy:
- **listing** je *Listing*, ke kterému se Booking vztahuje

Další

Plné znění *tříd* a *výčtových typů* v datovém modelu viz *Příloha C*.

- **Contact** Kontakt na osobu, která aktuálně zodpovídá za telefonické hovory pro dané oddělení
- **ContactType** Typ/oddělení, ke kterému se *Contact* váže.
- **Priority** Priorita daného *Case* nebo *Task*.

- **CaseState** Popisuje stav daného *Case*.
- **SubtaskState** Popisuje stav daného *Subtask*.
- **Locality** Oblasti města Prahy, kde se vyskytují byty, které společnost spravuje.
- **Supplement** Jedná se o doplňující objekty, které se vážou na *Case* nebo *Task*.
- **Photo** Fotka spojená s daným *Case* nebo *Task*.
- **Comment** Komentář spojený s daným *Case* nebo *Task*
- **Activity** Jedná se o záznam o provedení změny daného *Case* nebo *Task*. Tento záznam přichází uživatelům jako notifikace.
- **Role** Popisuje uživatelskou roli, která může být buď Operative nebo Manager.

2.5 Shrnutí analýzy

Navzdory existujícím aplikacím jsme po analýze požadavků zvolili vlastní řešení, protože existující aplikace nesplňovaly zásadní požadavky viz kapitola *Existující řešení*. Navíc integrace těchto aplikací se stávajícím systémem se odhaduje na 50 MD a spolu s pravidelnými poplatky za užívaní aplikací v plném rozsahu se tyto varianty nevyplatily oproti vlastnímu řešení, které se odhaduje na 40 MD.

Kapitola 3
Implementace

3.1 Vybrané technologie

Technologie jsem vybíral převážně na základě požadavků společnosti, technologiím použitých ve stávajícím systému a mých zkušeností s jednotlivými technologiemi.

3.1.1 Programovací jazyky

Frontend

HTML [3] Jazyk sloužící k tvorbě webových stránek. Dokument vytvořený v tomto jazyku je schopný prohlížeč přeložit a zobrazit požadovaný obsah stránky.

CSS [4] Jazyk popisující způsob a styl zobrazení HTML elementů na webové stránce.

Javascript [5] Jazyk umožňující dynamicky manipulovat s HTML a CSS prvky. Dále spravuje logiku a stav klientské strany. V této aplikaci používám následující knihovny a frameworky:

- **React [7]** Javascriptová knihovna sloužící k tvorbě uživatelského rozhraní. Tato knihovna výrazně zjednodušuje psaní uživatelského rozhraní oproti čistému HTML, CSS a Javascriptu.
- **Redux [7]** Javascriptový framework sloužící k uchování dat, přiřazování dat k elementům a ke změně dat v aplikaci. Tento framework také nutí vývojáře psát kód podle určitých pravidel a tím přispívá ke konzistenci kódu.
- **RequireJs [8]** Knihovna sloužící pro vkládání a načítání externích Javascriptových knihoven.
- **Material UI [9]** Komponentová sada pro tvorbu uživatelského rozhraní pomocí React. Obsahuje již před-

definované a komponenty jako jsou tlačítka, nadpisy, seznamy atd.

Backend

Java [10] Objektově orientovaný programovací jazyk se zjednodušenou syntaxí vycházející z C a C++.

Java EE [11] Dnes s názvem Jakarta EE, je kolekce frameworků sloužících k vývoji serverové strany aplikací. Pro vývoj používám následující specifikace:

- **JPA** [12] poskytuje rozhraní pro mapování tříd do tabulek v databázi
- **JTA** [13] poskytuje rozhraní pro užívání transakcí. Transakce je složená z úkolů, které musí být všechny splněny, aby byla transakce dokončena.
- **JAXB** [14] poskytuje rozhraní pro konverzi Java objektů na XML a naopak.
- **JAX-RS** [16] poskytuje rozhraní pro implementaci RESTful API.

Databáze

PostgreSQL [17] Objektově-relační databázový systém, který používá a rozšiřuje jazyk SQL.

3.1.2 Vývojové prostředí

Atom IDE používám pro vývoj frontendu, jelikož spolu s vhodnými pluginy dokáže zvýrazňovat syntaktické chyby, doplňovat a formátovat kód v HTML, CSS a Javascriptu.

Netbeans používám pro vývoj backendu, jelikož obsahuje vhodné nástroje pro vývoj v Javě včetně zvýrazňování syntaktických chyb, doplňování a formátování kódu.

PgAdmin používám při práci s databází jako např. kontrola dat, záloha a úprava tabulek, testování PSQL dotazů apod.

3.2 Struktura aplikace

Pro implementaci je použita třívrstvá architektura (anglicky Three tier architecture). Její výhoda je oddělení jednotlivých vrstev tak, aby byly na sobě nezávislé [22]. Toto umožňuje jednoduché přidání dalších modulů k již existující aplikaci. Tyto vrstvy jsou:

1. Datová vrstva (Database tier)
2. Aplikační vrstva (Business tier)
3. Prezentační vrstva (Presentational tier)

3.2.1 Datová vrstva

Datová vrstva zajišťuje ukládání, úpravu, mazání a výběr dat. [22]. Jak již bylo zmíněno výše, jedná se o integraci se stávajícím systémem, který má již většinu databáze implementovanou. Chybějící tabulky databáze se přidají pomocí tříd, které budou namapované na dané tabulky v databázi pomocí JPA [12].

3.2.2 Aplikační vrstva

Aplikační vrstva zajišťuje výpočty a operace nad dotazy z prezentační vrstvy, zpracované data dále předa datové vrstvě k uložení a odešle adekvátní odpověď zpět do prezentační vrstvy. Tato vrstva slouží jako prostředník mezi prezentační vrstvou a datovou vrstvou [22]

DAO

DAO, neboli Data access object, je programátorský koncept, který odděluje business logiku od datové vrstvy. DAO obsahuje obsahuje operace pro přidávání, mazání, úpravy a prohlížení dat z dané tabulky (tzn. CRUD [19] operace). Pro každou entitu existuje samostatný DAO. [21]

REST služby

Tyto služby slouží jako přístupové body pro komunikaci mezi klientem a serverem. Vstupní a výstupní data se budou přenášet v JSON (JavaScript Object Notation) [23] formátu. Tento formát jsem zvolili kvůli jednoduchému mapování na Javascript a Java objekty.

Tyto služby obsahují operace, které zpracují přijaté data od klienta, na základě těchto dat aktualizuje databázi a pošle klientovi adekvátní odpověď.

3.2.3 Prezentační vrstva

Prezentační vrstva má na starost zobrazení daného obsahu pro uživatele, podle aktuálních dat a stavu aplikace. Tato vrstva posílá požadavky a přijímá odpovědi z aplikační vrstvy [22].

Ducks

O stav prezentační vrstvy se postará knihovna Redux. Tento stav je dále přístupný všem komponentám, které zobrazují obsah uživateli. Při změně stavu v React, se překreslí komponenta, která stav vlastní. Výhodou použití Redux je, že tento stav naprosto odděluje od komponent, a na rozdíl od Reactu stav aplikace zpřístupní

globálně všem komponentám. Takže pokud změním stav v Redux, mohu překreslit více komponent na více místech najednou. [7]

Struktura pro tuto část je dělána podle konceptu Re-ducks. Pro každou novou funkcionalitu, je vytvořená složka (duck), která obsahuje veškerou logiku spojenou s touto funkcionalitou. [20]

Components

Components nebo česky komponenty slouží k zobrazování obsahu uživateli. Obsah je často určen stavem aplikace a mění se v průběhu používání aplikace. Pro přehlednost jsou komponenty dělené ještě na Containers a na Templates.

Containers mají za úkol zpracovávat data, které získá z Reduxu a dále je předat Templates. Dále Containers mohou vyvolávat operace, které mění stav aplikace nebo komunikují se serverem. Templates přijaté data prezentují uživateli.

Kapitola 4
Testování

4.1 Popis uživatele

Uživatelé aplikace jsou zaměstnanci společnosti, kteří pracují v terénu (dále jen operativní zaměstnanci). Operativní zaměstnanci jsou zpravidla studenti ve věku od 18 do 25 let a nemají problém s běžným užíváním inteligentních mobilů a počítačů. Operativní zaměstnanci však ještě nikdy nepracovali s touto aplikací. Předpoklady uživatele:

- Zná procesy a hodnoty společnosti
- Zná specifické označení společnosti jako jsou Case, Task, Listing apod.
- Naučí se pracovat s většinou nových aplikací za krátký čas
- Zná své přihlašovací údaje

4.2 Metody testování použitelnosti

K otestování aplikace jsme zvolili scénáře tak, aby se co nejvíce přibližovali reálnému použití v praxi. Každý scénář bude otestován jednou ze dvou následujících metod:

Kognitivní průchod [24] Tato metoda je navržena, aby zjistila jestli je uživatel schopen splnit dané testovací scénáře pro testovanou aplikaci. U testování se nejprve zeptáme, jestli uživatel ví co má dělat (Q0) a pak u každého kroku scénáře se ptáme na následující otázky:

- Q1: Ví uživatel jakou akci má zvolit?
- Q2: Je správná akce jasně viditelná pro uživatele?
- Q3: Dostane uživatel jasnou zpětnou vazbu o úspěšném vykonání akce?

Heuristická analýza [25] Ttato metoda určuje, jestli je aplikace v souladu s určitými heuristikami. Tyto heuristiky jsou:

- Viditelnost stavu systému
- Spojitost mezi systémem a reálným světem
- Uživatelská svoboda a kontrola
- Konzistence a dodržení standardů
- Prevence chyb
- Rozpoznání místo vzpomínání
- Flexibilita a efektivnost užívání
- Estetický a minimalistický design
- Pomoc uživatelům rozpoznat, pochopit a vzpamatovat se z chyby
- Nápověda a dokumentace

Výhodou těchto metod je, že jsou velice efektivní v poměru s jejich náklady, jelikož můžou být realizovány samotným vývojářem, a není třeba připravovat žádné testovací prostředí pro uživatele. Používají se i v návrhové části, což může potenciálně podchytit chyby, ještě než začne vývoj.

4.3 Testy použitelnosti

Testovací scénáře vychází z případů užití. Testy jsou zaměřeny zejména na uživatelské rozhraní. Zde jsou vypsané testovací scénáře, ve kterých se vyskytují závažnější nálezy.

4.3.1 UC-1 Vytvořit Case

Vytvořit Case po zjištění problému na Listing nebo Booking. Tímto zaměstnanec umožní managementu rychlý a detailní popis nalezeného problému a dalšími informacemi jako jsou: kdo nalezl problém, kdy byl problém nalezen, kdo byl na bytě zrovna ubytován apod. Nyní může management problém analyzovat a naplánovat řešení.

Aktér	Operativní zaměstnanec
Počáteční stav	Aktér nalezl problém na bytě, který je třeba nahlásit k dalšímu řešení.
Konečný stav	Problém byl nahlášen přes aplikaci a aktér dále kontroluje byt.
Prerekvizity	Aktér je přihlášen a nachází se na hlavní obrazovce aplikace, kde je přehled Case, které jsou relevantní pro aktéra (aktér je řešitel, zakladatel nebo odběratel Case).
Metoda	Kognitivní průchod [24]

TS-11 Hlavní scénář

1. Zvolit akci pro přidání nového Case
2. Vyplnit povinné formulářové pole
3. Vyplnit nepovinné formulářové pole
4. Potvrdit

Q0: Aktér chce nahlásit problém managementu.

Krok 1. Zvolit akci pro přidání nového Case

Q1: Ano, akce by měla být jasná uživateli.

Q2: Ne, ikonka pro přidání Case není dost viditelná v množství informací, které jsou zobrazeny na stránce.

Q3: Ano, aplikace přesměruje uživatele na stránku s přidáním nového Case.

Krok 2. Vyplnit povinné formulářové pole

Q1: Ano, akce by měla být jasná uživateli.

Q2: Ano, povinné pole jsou označené hvězdou.

Q3: Ano, aplikace zobrazuje vyplněné pole.

Krok 3. Vyplnit nepovinné formulářové pole

Q1: Ano, akce by měla být jasná uživateli.

Q2: Ano, nepovinné pole nejsou označené hvězdou.

Q3: Ano, aplikace zobrazuje vyplněné pole.

Krok 4. Potvrdit

Q1: Ano, akce by měla být jasná uživateli.

Q2: Ano, ikonka pro potvrzení by měla být viditelná a srozumitelná.

Q3: Ano, aplikace zobrazí stránku s detailem vytvořeného Case.

TS-12 Vedlejší scénář

1. Zvolit akci pro přidání nového Case
2. Vyplnit nepovinné formulářové pole
3. Potvrdit
4. Vyplnit povinné formulářové pole po výzvě systémem.

5. Potvrdit

Q0: Aktér chce nahlásit problém managementu.

Krok 1. Zvolit akci pro přidání nového Case

Q1: Ano, akce by měla být jasná uživateli.

Q2: Ne, ikonka pro přidání Case není dost viditelná v množství informací, které jsou zobrazeny na stránce.

Q3: Ano, aplikace přesměruje uživatele na stránku s přidáním nového Case.

Krok 2. Vyplnit nepovinné formulářové pole

Q1: Ano, akce by měla být jasná uživateli.

Q2: Ano, nepovinné pole nejsou označené hvězdou.

Q3: Ano, aplikace zobrazuje vyplněné pole.

Krok 3. Potvrdit

Q1: Ano, akce by měla být jasná uživateli.

Q2: Ano, ikonka pro potvrzení by měla být viditelná a srozumitelná.

Q3: Ano, aplikace zvýrazní nevyplněné povinné pole.

Krok 4. Vyplnit povinné formulářové pole

Q1: Ano, akce by měla být jasná uživateli.

Q2: Ano, povinné pole jsou označené hvězdou.

Q3: Ano aplikace zobrazuje vyplněné pole.

Krok 5. Potvrdit

Q1: Ano, akce by měla být jasná uživateli.

Q2: Ano, ikonka pro potvrzení by měla být viditelná a srozumitelná.

Q3: Ano, aplikace zobrazí stránku s detailem vytvořeného Case.

Shrnutí

V kroku 1 není ikonka dostatečně viditelná. Kvůli této chybě může být aplikace matoucí pro nové uživatele aplikace.

4.3.2 UC-5 Zobrazit Task

Aktér	Operativní zaměstnanec
Počáteční stav	Aktér dostal notifikaci o přiřazení Task.
Konečný stav	Aktér zobrazí Task, který mu byl přiřazen.
Prerekvizity	Aktér je přihlášen a nachází se na hlavní obrazovce aplikace, kde je přehled Case, které jsou relevantní pro aktéra (aktér je řešitel, zakladatel nebo odběratel Case).
Metoda	Heuristická analýza [25]

TS-51 Zobrazit Task po přijetí notifikace

Zobrazit detail Task, od kterého uživatel dostal notifikaci, že mu byl Task přiřazen k řešení. Až management naplánuje Task pro Case, přiřadí tyto Task k řešení operativním zaměstnancům, kteří dostanou notifikaci o jejich přiřazení. Operativní zaměstnanci by měli mít možnost jednoduše zobrazit detail Task po obdržení notifikace.

1. Zvolení notifikace o Task nezobrazí jeho detail, ale zobrazí Case, u kterého je Task přiřazen. **Porušení heuristiky: Konzistence a dodržení standardů.**
2. Na Case v záložce Task není žádný indikátor o tom, který Task byl uveden v notifikaci. **Porušení heuristiky: Rozpoznání místo vzpomínání**

Shrnutí

Byly nalezeny 2 chyby. První závažnější chyba je, že notifikace nepřesměruje uživatele na daný Task, ale na Case. Druhá chyba je, že Task z notifikace není nijak indikovaný na Case v záložce Task.

4.3.3 UC-6 Dokončit úkol

Aktér	Operativní zaměstnanec
Počáteční stav	Aktér potřebuje dát vědět managementu, že na Task již nejde dále pracovat. Důvodem může být mimo jiné dokončení Task.
Konečný stav	Aktér dokončil Task a specifikoval stav dokončení.
Prerekvizity	Aktér je přihlášen a nachází se na hlavní obrazovce aplikace, kde je přehled Case, které jsou relevantní pro aktéra (aktér je řešitel, zakladatel nebo odběratel Case). Aktérovi jsou přiřazené Task.
Metoda	Heuristická analýza [25]

TS-61 Dokončit všechny úkoly uživatele

Dokončit všechny úkoly přiřazené k danému uživateli. Jelikož má společnost *Listing* po celé Praze a *Booking* v různé časy, je důležité aby operativní zaměstnanci měli dobrý přehled o jejich úkolech s dostatečným množstvím detailu jako jsou datum a čas uzávěrky, lokality apod. Po dokončení úkolů je třeba, aby měl operativní zaměstnanec možnost dodat informace o stavu *Task* managementu.

1. Datové skupiny by měly být řazeny vzestupně, aby nejstarší úkoly byly na prvním místě. **Porušení heuristiky: Konzistence a dodržení standardů.**
2. Nelze dokončit více úkolů najednou. **Porušení heuristiky: Flexibilita a efektivnost užívání.**
3. Dialog pro výběr stavu dokončení nezobrazuje, o který Task se jedná. **Porušení heuristiky: Rozpoznání místo vzpomínání.**
4. Když nastane chyba na serveru, uživatel není upozorněn na tuto skutečnost aplikací. **Porušení heuristiky: Pomoct uživatelům rozpoznat, pochopit a vzpamatovat se z chyby.**

Shrnutí

Byly nalezeny 3 chyby, které se dají poměrně snadno opravit. Největší chyba byla čtvrtá, kdy aplikace neupozorní na chybu ze strany serveru. Uživatel takto v domnění, že v pořádku dokončil Task, může opustit proces.

4.3.4 UC-10 Přihlásit odběr ke Case

Přihlásit odběr ke Case, aby mohl uživatel dostávat upozornění o aktuálním dění na Case. Můžou být takové Case, ke kterým uživatel není řešitelem, ale potřebuje vědět o jejich aktuálním stavu, aby mohl pokračovat v řešení vlastního Case. Například zaměstnanec má za úkol nakoupit a postavit nový nábytek do bytu, ale na daném bytu jsou štěnice, což má za úkol řešit jiný zaměstnanec. Zaměstnanec tak může nábytek nakoupit, ale ještě ne postavit, dokud se štěnice nevyhubí.

Aktér	Operativní zaměstnanec
Počáteční stav	Aktér nemá přihlášen odběr u Case a potřebuje dostávat budoucí notifikace od Case.
Konečný stav	Aktér má přihlášen odběr u Case a dostává notifikace o změnách na Case.
Prerekvizity	Aktér je přihlášen a nachází se na hlavní obrazovce aplikace, kde je přehled Case, které jsou relevantní pro aktéra (aktér je řešitel, zakladatel nebo odběratel Case).
Metoda	Kognitivní průchod [24]]

TS-101 Hlavní scénář

1. Vyfiltrovat všechny Case
2. Vyhledat Case podle jména
3. Zobrazit Case
4. Zvolit akci pro odběr Case

Q0: Aktér chce dostávat notifikace o změnách na daném Case.

Krok 1. Vyfiltrovat všechny Case

Q1: Ne, akce nemusí být zřejmá, jelikož textové pole pro vyhledávání nespecifikuje filtrovanou skupinu, kterou prohledává.

Q2: Ano, záložka All je viditelná a jasně pochopitelná.

Q3: Ano, aplikace zvýrazní filtrovanou záložku.

Krok 2. Vyhledat Case podle jména

Q1: Ano, akce by měla být jasná uživateli cílové skupiny.

Q2: Ano, vyhledávací pole je viditelné a popsané.

Q3: Ano, aplikace zobrazí vyhledávané výsledky.

Krok 3. Zobrazit Case

Q1: Ano, akce by měla být jasná uživateli cílové skupiny.

Q2: Ano, vyhledané položky jsou jasné popsané.

Q3: Ano, aplikace zobrazí vybraný Case.

Krok 4. Zvolit akci pro odběr Case

Q1: Ano, akce by měla být jasná uživateli.

Q2: Ano, tlačítko pro odběr je viditelné a jasně pochopitelné.

Q3: Ano, tlačítko pro odběr se po akci přemění z "FOLLOW"na "UNFOLLOW"

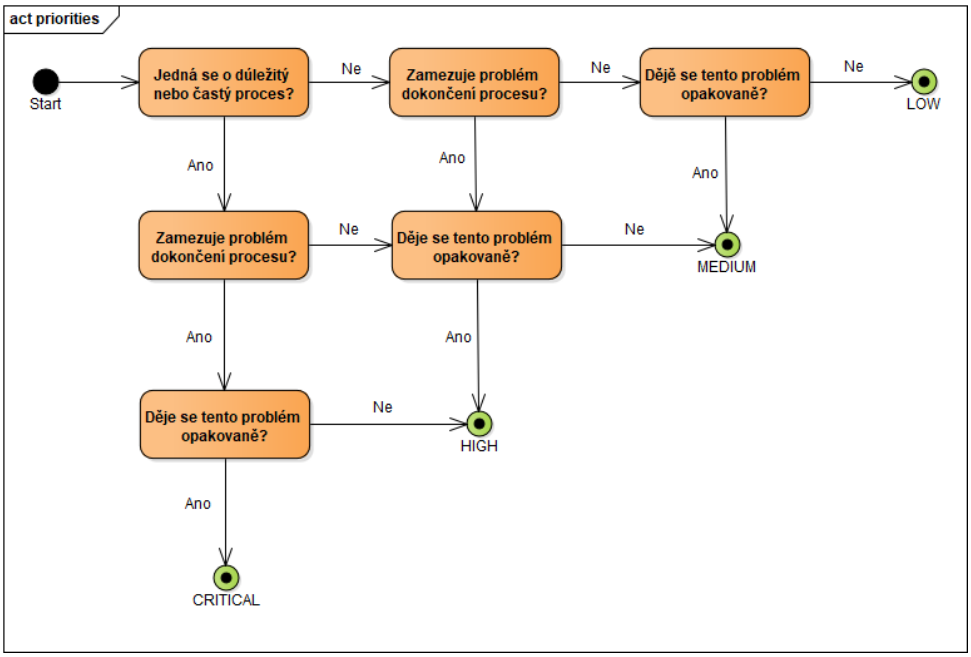
Shrnutí

Vyhledávací pole na Case přehledu může být matoucí, jelikož uživatel si může myslet, že na záložce My Cases může stále vyhledávat přes všechny Case. Tím pádem je možné, že krok 1 úplně přeskočí a zůstane na záložce My Cases.

4.4 Vyhodnocení testů

Ke každému nálezu přiřadíme prioritu na základě tří otázek [26]:

- **Jedná se o důležitý nebo častý proces?**
- **Zamezuje problém dokončení procesu?**
- **Děje se tento problém opakovaně?**



Obrázek 4.1: Digram určování priorit nálezů

Podle odpovědí na tyto otázky určíme prioritu [26] problému viz *Obrázek 4.1*

4.4.1 Priority

Critical — Tento problém znemožní uživateli úkol dokončit. Oprava je naléhavá.

High — Tento problém výrazně zpomalí dokončení procesu a může donutit uživatele aby našel jiný způsob řešení úkolu. Je třeba co nejdříve opravit.

Medium — Tento problém bude frustrovat nebo iritovat některé uživatele, ale nezamezí dokončení procesu. Opravit v další aktualizaci aplikace.

Low — Jedná se o kvalitativní problém například kosmetická nebo gramatická chyba.

4.4.2 Nálezy

Priorita	Scénář	Problém	Návrh
High	Vytvořit case	Ikonka pro přidání málo viditelná	Barevně odlišit nebo přidat ohraničen
High	Zobrazit Task po přijetí notifikace	Po kliknutí na notifikaci aplikace zobrazí Case, ke kterému je Task přiřazen	Po kliknutí zobrazit samotný Task
High	Zobrazit Task po přijetí notifikace	Na Case stránce v záložce Task není jasné, který Task byl už zobrazen	Barevně odlišit již zobrazené Task
High	Dokončit všechny úkoly uživatele	Datové skupiny nejsou seřazeny	Seřadit vzestupně podle data vytvoření
High	Dokončit všechny úkoly uživatele	Nelze dokončit více úkolů najednou	Přidat možnost výběru více úkolů a akci dokončit vybrané
High	Dokončit všechny úkoly uživatele	Dialog pro výběr stavu dokončení nezobrazuje žádný identifikátor Task	Přidat do dialogu název Task s id
High	Dokončit všechny úkoly uživatele	Při chybě na serverové straně není skutečnost oznámena uživateli	Přidat zpětnou vazbu s chybovou hláškou
Medium	Přihlásit odběr ke Case	Vyhledávací pole nespecifikuje filtrovanou skupinu	Do nápovědy přidat popis filtrované skupiny

Tabulka 4.1: Nálezy testování

Aplikace byla testována na vybraných scénářích metodami: kognitivní průchod [24] a heuristická analýza [25]. Při testování jsme nalezli 7 problémů s prioritou High a 1 problém s prioritou Medium. Tyto priority jsme přiřazovali rozhodovacího diagramu viz *Obrázek 4.1*. Žádná z těchto chyb nezamezí dokončení daných procesů, ale může je zpomalit a taky výrazně znepříjemnit užívání samotné aplikace. Tyto problémy nejsou implementačně náročné na opravu a měly by se řešit co nejdříve.

Kapitola 5

Závěr

V této práci, která se zabývá analýzou, implementací a testováním aplikace pro správu úkolů ve společnosti Curso s.r.o., jsem provedl rešerši již existujících aplikací, přičemž jsem detailně analyzoval dvě z nich. Došel jsem k závěru, že vlastní řešení bude lepší volba pro požadavky společnosti, jelikož řešení požaduje integraci s datovými strukturami, které jsou používané ve stávajícím systému a taky protože vlastní řešení nabízí mnohem větší volnost s přizpůsobením a následným přidáváním funkcí. Navíc podle odhadů by integrace těchto aplikací byla náročnější než implementace vlastního řešení.

V rámci analýzy jsem vybral technologie pro implementaci na základě používaných technologií ve stávajícím systému a taky na základě vlastních zkušeností. Dále jsem vytvořil datový model, katalog požadavků a zpracoval jsem případy užití. Analýza sloužila jako podklad pro implementaci a následné testování.

Pro implementaci byla zvolena třívrstvá architektura, kvůli nezávislosti jednotlivých vrstev a jednoduchosti přidávání dalších modulů. Implementace byla dále rozdělena podle vrstev této architektury na datovou, aplikační a prezentační. Díky mým předchozím zkušenostem s touto architekturou proběhla implementace bez větších problémů.

Při testování aplikace, jsme se soustředili především na testování uživatelského rozhraní. Během testů byly nalezeny problémy, které nezamezí uživatelům dokončení úkolů, ale mohou je výrazně zpomalit nebo daným uživatelům znepříjemnit užívání aplikace. Tyto nálezy budou řešeny a opraveny v nejbližších aktualizacích aplikace.

Aplikace je do dnes každodenně využívána zaměstnanci společnosti Curso s.r.o. a dále se pracuje na její optimalizaci a vylepšení.

Příloha A

Literatura

[1] *"Správa Krátkodobých Pronájmů." Blahobyty, https://blahobyty.cz/.*

[2] *Arlow, J., Neustadt, I. (2007). Uml 2 a unifikovaný proces vývoje aplikací. Brno: Computer Press.*

[3] *"Hypertext Markup Language." Wikipedia, Wikimedia Foundation, 12. listopadu 2019, https://cs.wikipedia.org/wiki/Hypertext_Markup_Language.*

[4] *Adaptic, s.r.o. - Internetová řešení podle vašich potřeb; www.adaptic.cz. "CSS." Adaptic, http://www.adaptic.cz/znalosti/slovnicek/css/.*

[5] *"JavaScript." MDN Web Docs, https://developer.mozilla.org/en-US/docs/Web/JavaScript.*

[6] *Sridhar, Jay. "What Is ES6 and What Javascript Programmers Need to Know." MakeUseOf, 23. října 2017, https://www.makeuseof.com/tag/es6-javascript-programmers-need-know/.*

[7] *Banks, A., Porcello, E. (2017). Learning react: functional web development with React and Redux. Beijing: OReilly.*

[8] *"A Javascript Module Loader." RequireJS, https://requirejs.org/.*

[9] *freeCodeCamp.org. "Meet Material-UI - Your New Favorite User Interface Library." FreeCodeCamp.org, FreeCodeCamp.org, 15. dubna 2018, https://www.freecodecamp.org/news/meet-your-material-ui-your-new-favorite-user-interface-library-6349a1c88a8c/.*

[10] *"Java (Programovací Jazyk)." Wikipedia, Wikimedia Foundation, 27. listopadu 2019, https://cs.wikipedia.org/wiki/Java_(programovací_jazyk).*

[11] *Rouse, Margaret a kolektiv. "What Is Java Platform, Enterprise Edition (Java EE)? - Definition from WhatIs.com." TheServerSide.com, https://www.theserverside.com/definition/J2EE-Java-2-Platform-Enterprise-Edition.*

[12] *Janssen, Thorben a kolektiv. "What's the Difference between JPA, Hibernate and EclipseLink." Thoughts on Java, 16. ledna 2019, https://thoughts-on-java.org/difference-jpa-hibernate-eclipselink/.*

[13] *"Understanding JTA - The Java Transaction API." Progress.com, http://www.progress.com/tutorials/jdbc/understanding-jta.*

[14] *Mandliya, Arpit. "JAXB Tutorial - Getting Started." Java Code Geeks, 5. ledna 2015, http://www.javacodegeeks.com/2013/02/jaxb-tutorial-getting-started.html.*

[15] *"Introduction to XML." XML Introduction, http://www.w3schools.com/xml/xml_whatis.asp.*

[16] *Andronache, Mihai. "JAX-RS Is Just an API!" Baeldung, 9. července 2019, http://www.baeldung.com/jax-rs-spec-and-implementations.*

[17] *Rouse, Margaret a Margaret Rouse. "What Is PostgreSQL? - Definition from WhatIs.com." WhatIs.com, https://whatis.techtarget.com/definition/PostgreSQL.*

[18] *"Domain Model." Wikipedia, Wikimedia Foundation, 6. listopadu 2019, https://en.wikipedia.org/wiki/Domain_model.*

[19] *"What Is CRUD?" Codecademy, https://www.codecademy.com/articles/what-is-crud.*

[20] *Moldovan, Alex. "Scaling Your Redux App with Ducks." FreeCodeCamp.org, FreeCodeCamp.org, 15. června 2019, https://www.freecodecamp.org/news/scaling-your-redux-app-with-ducks-6115955638be/.*

[21] *Baeldung. "The DAO Pattern in Java." Baeldung, 9. října 2019, https://www.baeldung.com/java-dao-pattern.*

[22] *ManagementMania. "Třívrstvá Architektura (Three-Tier Architecture)." ManagementMania.com, ManagementMania.com, 5. prosince 2015, https://managementmania.com/cs/trivrstva-architektura-three-tier-architecture.*

[23] *"JSON." JSON, http://www.json.org/.*

[24] *"How to Conduct a Cognitive Walkthrough." The Interaction Design Foundation, https://www.interaction-design.org/literature/article/how-to-conduct-a-cognitive-walkthrough.*

[25] *"What Is Heuristic Evaluation?" The Interaction Design Foundation, https://www.interaction-design.org/literature/topics/heuristic-evaluation.*

[26] *Travis, David. "How to Prioritise Usability Problems." UserFocus, https://www.userfocus.co.uk/articles/prioritise.html.*

Příloha B

Návod na lokální spuštění

Pro lokální spuštění je třeba mít nainstalovaný PostgreSQL. Stažení PostgreSQL. Instalace Mavenu a JDK.

1. Vytvoření lokální databáze pomocí psql skriptu:

```
CREATE USER "curso-dev" WITH PASSWORD 'xxx';
CREATE DATABASE "curso-dev" WITH ENCODING='UTF8' OWNER="
    curso-dev" CONNECTION LIMIT=-1;
```

2. Stažení aplikačního serveru Payara https://www.payara.fish/software/downloads/

3. Ve složce staženáPayara/glassfish/config/domain1/config je soubor domain.xml, který upravíte.

   K node security-service přidejte následující:

```
<auth-realm classname="com.sun.enterprise.security.auth.realm.jdbc.
    JDBCRealm" name="curso-dev">
 <property name="jaas-context" value="jdbcRealm"></property>
 <property name="charset" value="UTF-8"></property>
 <property name="encoding" value="Hex"></property>
 <property name="digest-algorithm" value="MD5"></property>
 <property name="digestrealm-password-enc-algorithm" value="MD5
    "></property>
 <property name="datasource-jndi" value="jdbc/dev/curso"></property
    >
 <property name="user-table" value="sysuser"></property>
 <property name="group-table" value="user_group"></property>
 <property name="user-name-column" value="username"></property>
 <property name="password-column" value="password"></property>
 <property name="group-name-column" value="groupname"></
    property>
 <property name="group-table-user-name-column" value="username
    "></property>
```

```
</auth-realm>
```

K node resource přidejte následující:

```
<jdbc-connection-pool datasource-classname="org.postgresql.ds.
      PGConnectionPoolDataSource" name="pg-curso-dev" res-type="
      javax.sql.ConnectionPoolDataSource">
 <property name="PortNumber" value="5432"></property>
 <property name="Password" value="xxx"></property>
 <property name="ServerName" value="localhost"></property>
 <property name="DatabaseName" value="curso-dev"></property>
 <property name="User" value="curso-dev"></property>
</jdbc-connection-pool>
<jdbc-resource pool-name="pg-curso-dev" jndi-name="jdbc/dev/
      curso"></jdbc-resource>
```

K node server přidejte následující:

```
<resource-ref ref="jdbc/dev/curso"></resource-ref>
```

4. Stáhnout projekt curso z github repozitáře https://github.com/nguyeda1/curso.
5. Ve složce curso spusťte příkazový řádek a v něm příkaz:

```
mvn clean install
```

6. Ve složce staženáPayara/bin spusťte příkazový řádek a v něm příkazy (jeden po druhém):

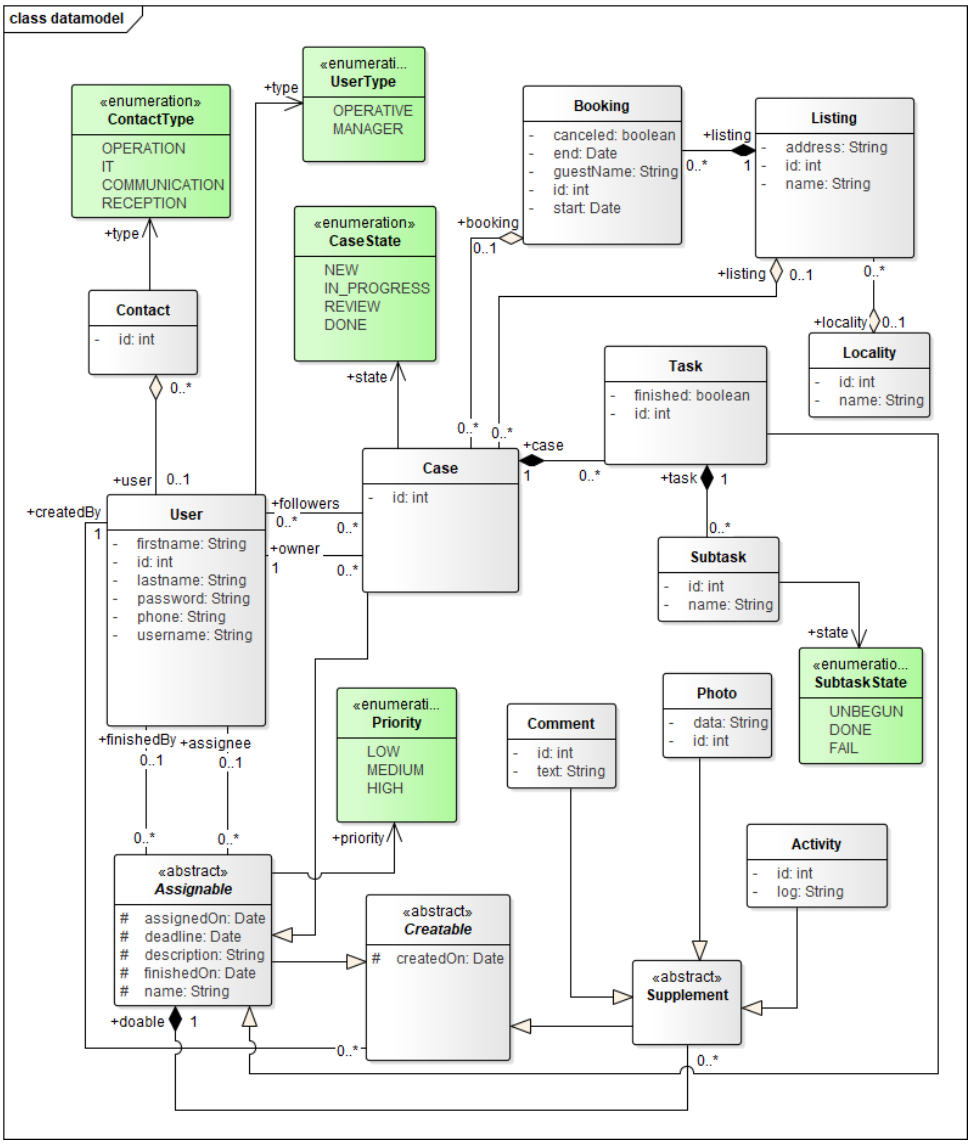
```
./asadmin start-domain
./asadmin deploy curso/curso-ear/target/curso-dev-ear-1.0-
      SNAPSHOT.ear
```

Mobilní aplikace by měla být dostupná z prohlížeče na http://localhost:8080/tasks/curso-dev/#/.
Pro optimální zobrazení spustit v mobilním prohlížeči nebo na normálním prohlížeči pomocí vývojářských nástrojů s možností zobrazení pro mobilní zařízení.
Username: operative, Password: 12345.

Manažerská aplikace by měla být dostupná z prohlížeče na http://localhost:8080/curso-dev/app. **Username: admin, Password: 12345**.

Příloha C

Datový model



Obrázek C.1: Diagram tříd

Creatable

Typ: Abstraktní třída.

Popis: Třídy odvozené z Creatable jsou ty, které mají evidovat datum a čas vytvoření instance a *User*, který ji vytvořil.

Atributy: **createdOn** je datum vytvoření

Vztahy: **createdBy** je *User*, který instanci vytvořil

User

Typ: Třída.

Popis: Uživatel aplikace.

Atributy: **id** je identifikátor
firstname je jméno
lastname je příjmení
username je přihlašovací jméno
password je heslo
phone je telefonní číslo

Vztahy: **type** je *UserType* daného uživatele

UserType

Typ: Výčtový typ

Popis: Jedná se typ *User*, který popisuje jeho roli/pozici ve společnosti.

Hodnoty: **EMPLOYEE** je terénní zaměstnanec firmy využívající Mobile aplikaci
MANAGER je manager společnosti využívající Internal aplikaci

Contact

Typ: Třída.

Popis: Kontakt na osobu, která aktuálně zodpovídá za telefonické hovory pro dané oddělení

Atributy: **id** je identifikátor

Vztahy: **user** je *User*, který je vázaný na Contact
type je *ContactType* daného Contact

ContactType

Typ: Výčtový typ

Popis: Typ/oddělení, ke kterému se *Contact* váže

Hodnoty:
IT je IT oddělení
OPERATION je oddělení terénním zaměstnanců
RECEPTION je recepce
COMMUNICATION je oddělení komunikace s majiteli bytů

Assignable

Typ: Abstraktní třída.

Dědičnost: Odvozená z *Creatable*

Popis: Třídy odvozené z Assignable jsou činnosti, které jsou přiřaditelné ke splnění určitému *User*, mají deadline a evidují datum splnění.

Atributy:
assignedOn je datum přiřazení činnost určitému *User*
deadline je nejpozdější datum, do kterého je třeba činnost splnit
finishedOn je datum, kdy byla činnost splněna
name je název činnosti
description je popis činnosti
createdOn

Vztahy:
assignee je *User*, kterému byla činnost přidělena
finishedBy je *User*, kterým byla činnost splněna
priority je *Priority* činnosti
createdBy

Priority

Typ: Výčtový typ

Popis: Priorita daného *Task* nebo *Task*

Hodnoty:
LOW značí nízkou prioritu
MEDIUM značí střední prioritu
HIGH značí vysokou prioritu

Case

Typ: Třída.

Dědičnost: Odvozená ze třídy *Assignable*

Popis: Je událost, kterou je třeba vyřešit ke správnému chodu správy bytu. Tato událost může mít několik *Task*, které je třeba splnit.

Atributy:
id je identifikátor
createdOn
assignedOn
deadline
finishedOn
name
description

Vztahy:
state je *CaseState*, ve kterém se Case nachází
followers je seznam *User*, kterým přichází notifikace o změnách na daném Case
booking je *Booking*, ke kterému se Case vztahuje
listing je *Listing*, ke kterému se Case vztahuje
createdBy
assignee
finishedBy
priority

CaseState

Typ: Výčtový typ

Popis: Popisuje stav daného *Task*.

Hodnoty:
NEW značí nový *Task*
IN_PROGRESS značí *Task*, na kterém se aktuálně pracuje
REVIEW značí *Task*, který by měl být zkontrolován managementem
DONE značí *Task*, který byl dokončen

Task

Typ: Třída.

Dědičnost: Odvozená ze třídy *Assignable*

Popis: Úkol týkající se správy bytu zákazníka společnosti, kterou je třeba vykonat. Tento úkol může být složen z několika *Subtask*.

Atributy:
id je identifikátor
finished je ukazatel, jestli je úkol dokončen nebo ne
createdOn
assignedOn
deadline
finishedOn
name
description

Vztahy:
task je *Task*, ke kterému je Task přiřazený
createdBy
assignee
finishedBy
priority

Subtask

Typ: Třída.

Popis: Podúkol, který může *Task* mít

Atributy:
id je identifikátor
name je datum název

Vztahy:
task je *Task*, ke kterému se Subtask vztahuje
sate je *SubtaskState*, ve kterém se Subtask nachází

SubtaskState

Typ: Výčtový typ

Popis: Popisuje stav daného *Subtask*.

Hodnoty:
UNBEGUN značí, že na *Subtask* se ještě nezačalo pracovat
DONE značí, že *Subtask* byl úspěšně dokončen
FAIL značí, že *Subtask* byl neúspěšně dokončen

Locality

Typ: Třída.

Popis: Oblasti města Prahy, kde se vyskytují byty, které společnost spravuje.

Atributy: **id** je identifikátor
name je název oblasti

Listing

Typ: Třída.

Popis: Byt zákazníka společnosti.

Atributy: **id** je identifikátor
address je adresa bytu
name je zobrazený název bytu

Vztahy: **locality** je *Locality*, kde se byt nachází

Booking

Typ: Třída.

Popis: Rezervace nebo uskutečněný pobyt na bytě majitele, který je zákazník společnosti.

Atributy: **id** je identifikátor
start je datum začátku pobytu
end je datum konce pobytu
guestName je jméno hosta
canceled popisuje jestli je nebo byl pobyt zrušen

Vztahy: **listing** je *Listing*, ke kterému se Booking vztahuje

Supplement

Typ: Abstraktní třída.

Dědičnost: Odvozená z *Creatable*.

Popis: Jedná se o doplňující objekty, které se vážou na *Task* nebo *Task*.

Atributy: *createdOn*

Vztahy: **assignable** je *Assignable*, ke kterému se Supplement vztahuje
createdBy

Photo

Typ: Třída.

Dědičnost: Odvozená z *Supplement*.

Popis: Fotka spojená s daným *Task* nebo *Task*.

Atributy: **id** je identifikátor
data jsou samotné data fotky
createdOn

Vztahy: *assignable*
createdBy

Comment

Typ: Třída.

Dědičnost: Odvozená z *Supplement*.

Popis: Komentář spojený s daným *Task* nebo *Task*.

Atributy: **id** je identifikátor
text je textový obsah komentáře
createdOn

Vztahy: *assignable*
createdBy

Activity

Typ: Třída.

Dědičnost: Odvozená z *Supplement*.

Popis: Jedná se o záznamu o provedení změny daného *Task* nebo *Task*.

Atributy: **id** je identifikátor
log je textový záznam změny
createdOn

Vztahy: *assignable*
createdBy

Příloha D

Případy užití

UC-1 Vytvořit Case

Aktéři	*User*
Popis	*User* vytvoří nový *Case* na základě události spojenou s *Listing*.
Spoušť	*User* chce založit *Case* na základě události, kterou je třeba řešit.
Vstup	*User* je přihlášen a nachází se na domovské stránce.
Výstup	*Case* je vytvořen a uložen v databázi. *User* se nachází na stránce *Case* detailu.
Hlavní scénář	1. Uživatel přejde na stránku pro Case vytvoření. 2. Systém zobrazí stránku pro Case vytvoření. 3. Uživatel vyplní formulářové pole 4. Uživatel odešle formulář 5. Systém uloží Case do databáze 6. Systém přesměruje stránku na Case detail
Vedlejší scénáře	5a Pokud uživatel nevyplnil všechny povinné formulářové pole, tak ho systém o to požádá a změny neuloží. Případ užití se vrací na krok 3.

UC-2 Zobrazit Case

Aktéři	*User*
Popis	*User* vyhledá *Case* podle atributů, které zná. Tyto atributy můžou být *id, name* nebo *caseState*.
Spoušť	*User* chce vyhledat *Case*, aby s ním mohl dále pracovat.
Vstup	*User* je přihlášen a nachází se na domovské stránce.

Výstup *User* nalezl *Case* a nachází se na stránce *Case* detailu.

Hlavní scénář

1. *User* vyfiltruje *Case* přehled podle *caseState*.
2. Systém zobrazí *Case* přehled podle vyfiltrovaného *caseState*.
3. *User* zadá do vyhledávání *id* nebo *name* daného *Case*.
4. Systém zobrazí všechny *Case*, které splňují vyhledávací kriteria.
5. *User* přejde na detail hledaného *Case*.
6. Systém zobrazí *Case* detail.

Vedlejší scénáře

3a *User* nezná *id* ani *name* a nalezne hledaný *Case* scrollováním přehledu.

3b *User* nezná *id* ani *name* a dál nehledá. Případ užití tímto končí.

UC-3 Změnit stav od Case

Aktéři *Manager*

Popis *Manager* změní *caseState* u daného *Case*.

Spoušť *Manager* chce změnit *caseState* u daného *Case*.

Vstup Tento *případ užití* rozšiřuje (**UC-4** Case úprava). *Manager* jej iniciuje.

Výstup *Manager* změnil *caseState* u daného *Case*.

UC-4 Upravit Case

Aktéři *User*

Popis *User* upraví některé z *Case* následujících atributů: *name, description, Listing, Booking*.

Spoušť *User* chce upravit atributy u daného *Case*.

Vstup *User* je přihlášen a nachází se na domovské stránce. *User* je *owner* nebo *createdBy* pro daný *Case*.

Výstup *User* upravil vybrané atributy u daného *Case*

Hlavní scénář

1. *User* najde daný *Case* detail (**UC-2**)
2. *User* zvolí možnost úpravy na daném *Case*
3. *User* upraví vybrané atributy a pokud chce tak i *caseState* (**UC-3**)
4. *User* potvrdí změny

5. Systém uloží změny
6. Systém se vrátí na stránku *Case* detail

Vedlejší scénáře

4a *User* se vrátí zpět na *Case* detail. Případ užití se vrací na krok 2.

5a *User* nechal povinné údaje prázdné, tak ho systém vyzve k úpravě chyb. Případ užití se vrací na krok 3.

UC-5 Zobrazit Task

Aktéři *User*

Popis *User* vyhledá *Task* podle atributů, které zná. Tyto atributy můžou být *id, name* nebo *finished*.

Spouští *User* chce vyhledat *Task*, aby s ním mohl dále pracovat.

Vstup *User* je přihlášen a nachází se na domovské stránce. *User* je *assignee* hledaného *Task*.

Výstup *User* nalezl *Task* a nachází se na stránce *Task* detailu.

Hlavní scénář

1. *User* přejde na stránku s *Task* přehledem
2. Systém zobrazí Task přehled.
3. *User* vyfiltruje *Task* přehled podle atributu *finished*.
4. Uživatel pomocí seskupení přehledu podle deadlinu a místa konání nalezne hledaný Task
5. *User* přejde na detail hledaného *Task*.
6. Systém zobrazí *Task* detail.

Vedlejší scénáře

3a *Task* se nachází v přehledu dokončených *Task*, tak se uživatel pokusí najít *Task* pomocí *id* nebo *name*.

3b *Task* se nachází v přehledu dokončených *Task* a uživatel nezná *id* ani *name* tak se pokusí najít Task scrollováním.

3c *Task* se nachází v přehledu dokončených *Task*, uživatel nezná *id* nebo *name* a už dál nehledá. Případ užití tímto končí.

UC-6 Dokončit Task

Aktéři *User*

Popis *User* reportuje, že *Task* je hotov tím, že ho v systému jako *finished*.

Spouští *User* chce reportovat dokončení *Task*.

Vstup *User* je přihlášen a nachází se na domovské stránce. *User* je *assignee* daného *Task*, který ještě není dokončený. *Task* má všechny *Subtask* v jiném *SubtaskState* než *UNBEGUN* nebo neobsahuje *Subtask*.

Výstup *User* označil *Task* jako *finished* a nachází se na stránce *Task* detailu. Systém uložil změny do databáze.

Hlavní scénář

1. *User* přejde na stránku s *Task* přehledem
2. Systém zobrazí *Task* přehled.
3. *User* najde daný *Task* detail (**UC-5**)
4. *User* zvolí možnost dokončení *Task*.

UC-7 Nahrát fotky

Aktéři *User*

Popis *User* nahraje *Photo* k danému *Assignable*, aby přiblížil ostatním situaci.

Spouští *User* chce nahrát *Photo* ke *Assignable*.

Vstup *User* je přihlášen a nachází se na stránce s *Assignable* přehledem.

Výstup *User* nahrál *Photo* k danému *Assignable*

Hlavní scénář

1. *User* zvolí možnost nahrání *Photo*
2. *User* vybere jednu nebo více *Photo*
3. *User* potvrdí výběr
4. Systém uloží *Photo*

Vedlejší scénáře

3a *User* nevybral žádnou *Photo*, tak ho systém k tomu vyzve znovu.

3b *User* zruší operaci potvrzení. Případ užití tímto končí.

UC-8 Samazat fotky

Aktéři *User*

Popis *User* smaže *Photo* z daného *Assignable*.

Spouští *User* chce smazat *Photo* z *Assignable*.

Vstup *User* je přihlášen a nachází se na stránce s *Assignable* přehledem. Daná *Photo* byla nahrána *User*.

Výstup *User* smazal *Photo* z daného *Assignable*

Hlavní scénář

1. *User* vybere jednu nebo více *Photo*
2. Systém zobrazí možnost pro smazání vybraných *Photo*
3. *User* zvolí možnost smazání *Photo*
4. Systém zobrazí možnost pro potvrzení smazání vybraných *Photo*
5. *User* potvrdí výběr
6. Systém smaže *Photo*

Vedlejší scénáře

5a *User* zruší operaci potvrzení a může upravit svůj výběr *Photo*. Případ užití se vrací na krok 3.

UC-9 Přidat komentář

Aktéři *User*

Popis *User* okomentuje *Assignable*.

Spoušť *User* chce okomentovat *Assignable*.

Vstup *User* je přihlášen a nachází se na stránce s *Assignable* přehledem.

Výstup *User* okomentoval *Assignable*

Hlavní scénář

1. *User* přejde na seznam *Comment*
2. Systém zobrazí přehled *Comment*
3. *User* napíše komentář do textového pole
4. *User* odešle *Comment*
5. Systém uloží změny
6. Systém zobrazí nový *Comment*

Vedlejší scénáře

4a *User* odesílá prázdné textové pole a systém neprovede požadavek. Případ užití končí

UC-10 Přihlásit odběr ke Case

Aktéři *User*

Popis *Operataive* bude odebírat notifikace o změnách na daném *Case*. *User* se stane jeden z *followers* daného *Case*.

Spoušť *Operataive* chce dostávat *Case* notifikace o změnách na daném *Case*.

Vstup *Operataive* je přihlášen a nachází se na domovské stránce. *User* ještě není odběratelem *Case*.

Výstup	*Operataive* se stal odběratelem daného *Case* a bude dostávat notifikace o jeho změnách.
Hlavní scénář	1. *User* najde daný *Case* detail (**UC-2**) 2. *User* zvolí možnost odběru na daném *Case* 3. Systém zařadí *User* mezi *followers* na daném *case*

UC-11 Zrušit odběr od Case

Aktéři	*Operataive*
Popis	*Operataive* přestane odebírat notifikace o změnách na daném *Case*.
Spouští	*Operataive* chce dostávat *Case* notifikace o změnách na daném *Case*.
Vstup	*Operataive* je přihlášen a nachází se na domovské stránce. *User* je odběratelem *Case*.
Výstup	*Operataive* přestal být odběratelem daného *Case* a nebude dostávat notifikace o jeho změnách.
Hlavní scénář	1. *Operataive* najde daný *Case* detail (**UC-2**) 2. *Operataive* zvolí možnost zrušení odběru na daném *Case* 3. Systém vyřadí *User* z *followers* na daném *case*

UC-12 Nahrát fotku ke Case

Aktéři	*User*
Popis	*User* nahraje *Photo* k danému *Case*, aby přiblížil ostatním situaci.
Spouští	*User* chce nahrát *Photo* ke *Case*.
Vstup	*User* je přihlášen a nachází se na domovské stránce.
Výstup	*User* nahrál *Photo* k danému *Case*
Hlavní scénář	1. *User* najde daný *Case* detail (**UC-2**) 2. *User* nahraje fotku ke *Case* (**UC-7**)

UC-13 Smazat fotky u Case

Aktéři	*User*
Popis	*User* smaže *Photo* z daného *Case*.
Spouští	*User* chce smazat *Photo* z *Case*.
Vstup	*User* je přihlášen a nachází se na domovské stránce. Daná *Photo* byla nahrána *User*.
Výstup	*User* smazal *Photo* z daného *Case*
Hlavní scénář	1. *User* najde daný *Case* detail (**UC-2**) 2. *User* smaže *Photo* z *Case* (**UC-8**)

UC-14 Přidat komentář ke Case

Aktéři	*User*
Popis	*User* okomentuje *Case*.
Spouští	*User* chce okomentovat *Case*.
Vstup	*User* je přihlášen a nachází se na domovské stránce.
Výstup	*User* okomentoval *Case*
Hlavní scénář	1. *User* najde daný *Case* detail (**UC-2**) 2. *User* přidá *Comment* na *Case* (**UC-9**)

UC-15 Nahrát fotky k Task

Aktéři	*User*
Popis	*User* nahraje *Photo* k danému *Task*, aby přiblížil ostatním situaci.
Spouští	*User* chce nahrát *Photo* k *Task*.
Vstup	*User* je přihlášen a nachází se na domovské stránce.
Výstup	*User* nahrál *Photo* k danému *Task*
Hlavní scénář	1. *User* přejde na *Task* přehled 2. *User* najde daný *Task* detail (**UC-5**) 3. *User* nahraje *Photo* na *Task* (**UC-7**)

UC-16 Smazat fotky u Task

Aktéři	*User*
Popis	*User* smaže *Photo* z daného *Task*.
Spouští	*User* chce smazat *Photo* z *Task*.
Vstup	*User* je přihlášen a nachází se na domovské stránce. Daná *Photo* byla nahrána *User*.
task	*User* smazal *Photo* z daného *Task*
Hlavní scénář	1. *User* přejde na *Task* přehled 2. *User* najde daný *Task* detail (**UC-5**) 3. *User* smaže *Photo* z *Task* (**UC-8**)

UC-17 Přidat komentář k Task

Aktéři	*User*
Popis	*User* okomentuje *Task*.
Spouští	*User* chce okomentovat *Task*.
Vstup	*User* je přihlášen a nachází se na domovské stránce.
Výstup	*User* okomentoval *Task*
Hlavní scénář	1. *User* přejde na *Task* přehled 2. *User* najde daný *Task* detail (**UC-5**) 3. *User* přidal *Comment* k *Task* (**UC-9**)

UC-18 Zavolat Contact

Aktéři	*Operataive*
Popis	*Operataive* zavolá danému *Contact*.
Spouští	*Operataive* chce zavolat *Contact*.
Vstup	*Operataive* je přihlášen a nachází se na domovské stránce.
Výstup	*Operataive* volá *Contact*
Hlavní scénář	1. *Operataive* přejde na *Contact* přehled 2. *Operataive* najde daný *ContactType* 3. *User* zvolí možnost volání daného *Contact* 4. Systém vytočí *phone* od *User* daného *Contact*
Vedlejší scénáře	3a Za *ContactType* aktuálně nezodpovídá *User* a *Operataive* nemůže zavolat *Contact*. Případ užitím tímto končí.

UC-19 Vytvořit Task

Aktéři	*Manager*
Popis	*Manager* vytvoří nový *Task* a přiřadí ho ke *Case*
Spoušť	*Manager* chce založit *Task* ke *Case*
Vstup	*Manager* je přihlášen a nachází se na domovské stránce.
Výstup	*Task* je vytvořen a uložen v databázi. *User* se nachází na stránce *Task* detailu.
Hlavní scénář	1. *Manager* přejde na přehled *Task* 2. *Manager* přejde na stránku pro *Task* vytvoření. 3. Systém zobrazí stránku pro *Task* vytvoření. 4. Uživatel vyplní formulářové pole 5. *Manager* odešle formulář 6. Systém uloží *Task* do databáze 7. Systém přesměruje stránku na *Task* detail
Vedlejší scénáře	6a Pokud uživatel nevyplnil všechny povinné formulářové pole, tak ho systém o to požádá a změny neuloží. Případ užití se vrací na krok 4.

UC-20 Upravit Task

Aktéři	*Manager*
Popis	*Manager* upraví některé z *Task* atributů
Spoušť	*Manager* chce upravit atributy u daného *Task*.
Vstup	*Manager* je přihlášen a nachází se na domovské stránce.
Výstup	*Manager* upravil vybrané atributy u daného *Task*
Hlavní scénář	1. *Manager* přejde na přehled *Task* 2. *Manager* najde daný *Task* detail (**UC-5**) 3. *Manager* zvolí možnost úpravy na daném *Task* 4. *Manager* upraví vybrané atributy 5. *Manager* potvrdí změny 6. Systém uloží změny 7. Systém se vrátí na stránku *Task* detail
Vedlejší scénáře	5a *Manager* se vrátí zpět na *Task* detail. Případ užití se vrací na krok 3. 6a *Manager* nechal povinné údaje prázdné, tak ho systém vyzve k úpravě chyb. Případ užití se vrací na krok 4.

UC-21 Vytvořit Subtask

Aktéři *Manager*

Popis *Manager* vytvoří nový *Subtask* a přiřadí ho ke *Task*

Spouští *Manager* chce založit *Subtask* ke *Task*

Vstup *Manager* je přihlášen a nachází se na domovské stránce.

Výstup *Subtask* je vytvořen a uložen v databázi. *User* se nachází na stránce *Subtask* detailu.

Hlavní scénář

1. *Manager* přejde na přehled *Task*
2. *Manager* najde daný *Task* detail (**UC-5**)
3. *Manager* zvolí možnost pro *Subtask* vytvoření.
4. Systém zobrazí stránku pro *Subtask* vytvoření.
5. Uživatel vyplní formulářové pole
6. *Manager* odešle formulář
7. Systém uloží *Subtask* do databáze
8. Systém přesměruje stránku na *Subtask* detail

Vedlejší scénáře

7a Pokud uživatel nevyplnil všechny povinné formulářové pole, tak ho systém o to požádá a změny neuloží. Případ užití se vrací na krok 5.

UC-22 Upravit Subtask

Aktéři *Manager*

Popis *Manager* upraví některé z *Subtask* atributů.

Spouští *Manager* chce upravit atributy u daného *Subtask*.

Vstup *Manager* je přihlášen a nachází se na domovské stránce.

Výstup *Manager* upravil vybrané atributy u daného *Subtask*

Hlavní scénář

1. *Manager* přejde na přehled *Task*
2. *Manager* najde *Task* detail (**UC-5**)
3. *Manager* najde seznam *Subtask*
4. *Manager* zvolí možnost úpravy daném *Subtask*
5. *Manager* upraví vybrané atributy
6. *Manager* potvrdí změny
7. Systém uloží změny
8. Systém se vrátí na stránku *Subtask* detail

Vedlejší scénáře

6a *Manager* se vrátí zpět na *Subtask* detail. Případ užití se vrací na krok 4.

7a *Manager* nechal povinné údaje prázdné, tak ho systém vyzve k úpravě chyb. Případ užití se vrací na krok 5.

UC-23 Vytvořit Listing

Aktéři *Manager*

Popis *Manager* vytvoří nový *Listing* a přiřadí ho ke *Case*

Spoušť *Manager* chce založit *Listing* ke *Case*

Vstup *Manager* je přihlášen a nachází se na domovské stránce.

Výstup *Listing* je vytvořen a uložen v databázi. *User* se nachází na stránce *Listing* detailu.

Hlavní scénář

1. *Manager* přejde na přehled *Listing*
2. *Manager* přejde na stránku pro *Listing* vytvoření.
3. Systém zobrazí stránku pro *Listing* vytvoření.
4. Uživatel vyplní formulářové pole
5. *Manager* odešle formulář
6. Systém uloží *Listing* do databáze
7. Systém přesměruje stránku na *Listing* detail

Vedlejší scénáře

6a Pokud uživatel nevyplnil všechny povinné formulářové pole, tak ho systém o to požádá a změny neuloží. Případ užití se vrací na krok 4.

UC-24 Upravit Listing

Aktéři *Manager*

Popis *Manager* upraví některé z *Listing* atributů

Spoušť *Manager* chce upravit atributy u daného *Listing*.

Vstup *Manager* je přihlášen a nachází se na domovské stránce.

Výstup *Manager* upravil vybrané atributy u daného *Listing*

Hlavní scénář

1. *Manager* přejde na přehled *Listing*
2. *Manager* najde daný *Listing* detail
3. *Manager* zvolí možnost úpravy na daném *Listing*
4. *Manager* upraví vybrané atributy

5. *Manager* potvrdí změny
6. Systém uloží změny
7. Systém se vrátí na stránku *Listing* detail

Vedlejší scénáře

5a *Manager* se vrátí zpět na *Listing* detail. Případ užití se vrací na krok 3.

6a *Manager* nechal povinné údaje prázdné, tak ho systém vyzve k úpravě chyb. Případ užití se vrací na krok 4.

UC-25 Vytvořit Booking

Aktéři *Manager*

Popis *Manager* vytvoří nový *Booking* a přiřadí ho ke *Case*

Spoušť *Manager* chce založit *Booking* ke *Case*

Vstup *Manager* je přihlášen a nachází se na domovské stránce.

Výstup *Booking* je vytvořen a uložen v databázi. *User* se nachází na stránce *Booking* detailu.

Hlavní scénář

1. *Manager* přejde na přehled *Booking*
2. *Manager* přejde na stránku pro *Booking* vytvoření.
3. Systém zobrazí stránku pro *Booking* vytvoření.
4. Uživatel vyplní formulářové pole
5. *Manager* odešle formulář
6. Systém uloží *Booking* do databáze
7. Systém přesměruje stránku na *Booking* detail

Vedlejší scénáře

6a Pokud uživatel nevyplnil všechny povinné formulářové pole, tak ho systém o to požádá a změny neuloží. Případ užití se vrací na krok 4.

UC-26 Upravit Booking

Aktéři *Manager*

Popis *Manager* upraví některé z *Booking* atributů

Spoušť *Manager* chce upravit atributy u daného *Booking*.

Vstup *Manager* je přihlášen a nachází se na domovské stránce.

Výstup *Manager* upravil vybrané atributy u daného *Booking*

Hlavní scénář

1. *Manager* přejde na přehled *Booking*

2. *Manager* najde daný *Booking*
3. *Manager* zvolí možnost úpravy na daném *Booking*
4. *Manager* upraví vybrané atributy
5. *Manager* potvrdí změny
6. Systém uloží změny
7. Systém se vrátí na stránku *Booking* detail

Vedlejší scénáře

5a *Manager* se vrátí zpět na *Booking* detail. Případ užití se vrací na krok 3.

6a *Manager* nechal povinné údaje prázdné, tak ho systém vyzve k úpravě chyb. Případ užití se vrací na krok 4.

UC-27 Vytvořit Locality

Aktéři *Manager*

Popis *Manager* vytvoří nový *Locality* a přiřadí ho ke *Case*

Spouští *Manager* chce založit *Locality* ke *Case*

Vstup *Manager* je přihlášen a nachází se na domovské stránce.

Výstup *Locality* je vytvořen a uložen v databázi. *User* se nachází na stránce *Locality* detailu.

Hlavní scénář

1. *Manager* přejde na přehled *Locality*
2. *Manager* přejde na stránku pro *Locality* vytvoření.
3. Systém zobrazí stránku pro *Locality* vytvoření.
4. Uživatel vyplní formulářové pole
5. *Manager* odešle formulář
6. Systém uloží *Locality* do databáze
7. Systém přesměruje stránku na *Locality* detail

Vedlejší scénáře

6a Pokud uživatel nevyplnil všechny povinné formulářové pole, tak ho systém o to požádá a změny neuloží. Případ užití se vrací na krok 4.

UC-28 Upravit Locality

Aktéři *Manager*

Popis *Manager* upraví některé z *Locality* atributů

Spouští *Manager* chce upravit atributy u daného *Locality*.

Vstup *Manager* je přihlášen a nachází se na domovské stránce.

Výstup *Manager* upravil vybrané atributy u daného *Locality*

Hlavní scénář

1. *Manager* přejde na přehled *Locality*
2. *Manager* najde daný *Locality*
3. *Manager* zvolí možnost úpravy na daném *Locality*
4. *Manager* upraví vybrané atributy
5. *Manager* potvrdí změny
6. Systém uloží změny
7. Systém se vrátí na stránku *Locality* detail

Vedlejší scénáře

5a *Manager* se vrátí zpět na *Locality* detail. Případ užití se vrací na krok 3.

6a *Manager* nechal povinné údaje prázdné, tak ho systém vyzve k úpravě chyb. Případ užití se vrací na krok 4.

UC-29 Vytvořit User

Aktéři *Manager*

Popis *Manager* vytvoří nový *User* a přiřadí ho ke *Case*

Spoušť *Manager* chce založit *User* ke *Case*

Vstup *Manager* je přihlášen a nachází se na domovské stránce.

Výstup *User* je vytvořen a uložen v databázi. *User* se nachází na stránce *User* detailu.

Hlavní scénář

1. *Manager* přejde na přehled *User*
2. *Manager* přejde na stránku pro *User* vytvoření.
3. Systém zobrazí stránku pro *User* vytvoření.
4. Uživatel vyplní formulářové pole
5. *Manager* odešle formulář
6. Systém uloží *User* do databáze
7. Systém přesměruje stránku na *User* detail

Vedlejší scénáře

6a Pokud uživatel nevyplnil všechny povinné formulářové pole, tak ho systém o to požádá a změny neuloží. Případ užití se vrací na krok 4.

UC-30 Upravit User

Aktéři	*Manager*
Popis	*Manager* upraví některé z *User* atributů
Spoušť	*Manager* chce upravit atributy u daného *User*.
Vstup	*Manager* je přihlášen a nachází se na domovské stránce.
Výstup	*Manager* upravil vybrané atributy u daného *User*
Hlavní scénář	1. *Manager* přejde na přehled *User* 2. *Manager* najde daný *User* 3. *Manager* zvolí možnost úpravy na daném *User* 4. *Manager* upraví vybrané atributy 5. *Manager* potvrdí změny 6. Systém uloží změny 7. Systém se vrátí na stránku *User* detail
Vedlejší scénáře	5a *Manager* se vrátí zpět na *User* detail. Případ užití se vrací na krok 3. 6a *Manager* nechal povinné údaje prázdné, tak ho systém vyzve k úpravě chyb. Případ užití se vrací na krok 4.

UC-31 Vytvořit Contact

Aktéři	*Manager*
Popis	*Manager* vytvoří nový *Contact* a přiřadí ho ke *Case*
Spoušť	*Manager* chce založit *Contact* ke *Case*
Vstup	*Manager* je přihlášen a nachází se na domovské stránce.
Výstup	*Contact* je vytvořen a uložen v databázi. *User* se nachází na stránce *Contact* detailu.
Hlavní scénář	1. *Manager* přejde na přehled *Contact* 2. *Manager* přejde na stránku pro *Contact* vytvoření. 3. Systém zobrazí stránku pro *Contact* vytvoření. 4. Uživatel vyplní formulářové pole 5. *Manager* odešle formulář 6. Systém uloží *Contact* do databáze 7. Systém přesměruje stránku na *Contact* detail
Vedlejší scénáře	6a Pokud uživatel nevyplnil všechny povinné formulářové pole, tak ho systém o to požádá a změny neuloží. Případ užití se vrací na krok 4.

UC-32 Upravit Contact

Aktéři	*Manager*
Popis	*Manager* upraví některé z *Contact* atributů
Spouští	*Manager* chce upravit atributy u daného *Contact*.
Vstup	*Manager* je přihlášen a nachází se na domovské stránce.
Výstup	*Manager* upravil vybrané atributy u daného *Contact*
Hlavní scénář	1. *Manager* přejde na přehled *Contact* 2. *Manager* najde daný *Contact* 3. *Manager* zvolí možnost úpravy na daném *Contact* 4. *Manager* upraví vybrané atributy 5. *Manager* potvrdí změny 6. Systém uloží změny 7. Systém se vrátí na stránku *Contact* detail
Vedlejší scénáře	5a *Manager* se vrátí zpět na *Contact* detail. Případ užití se vrací na krok 3. 6a *Manager* nechal povinné údaje prázdné, tak ho systém vyzve k úpravě chyb. Případ užití se vrací na krok 4.