
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ**
**DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA**


# JEDNODUCHÁ MOBILNÍ A WEBOVÁ APLIKACE PRO ORGANIZOVÁNÍ SCHŮZEK

**THESIS TITLE**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** **ERIKA FAŠÁNKOVÁ**
**AUTHOR**

**VEDOUCÍ PRÁCE** **prof. Ing. ADAM HEROUT, Ph.D.**
**SUPERVISOR**



**BRNO 2021**

**Vysoké učení technické v Brně**
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

# Zadání bakalářské práce


Studentka: **Fašánková Erika**
Program: Informační technologie
Název: **Jednoduchá mobilní a webová aplikace pro organizování schůzek**
**Simple Mobile and Web App for Organizing Meetings**
Kategorie: Uživatelská rozhraní

Zadání:

1. Seznamte se s problematikou vývoje webových aplikací s komunikací v reálném čase. Zohledněte i prezentaci na mobilním telefonu (responzivní design apod.).
2. Diskutujte případy užití aplikace umožňující vytvoření jednoduché virtuální fronty pro snadnou organizaci schůzek.
3. Prototypujte navrhovaný systém a testujte s uživateli.
4. Vytvořte implementaci webové aplikace, cílené na desktop i mobil.
5. Testujte vytvořené řešení s uživateli a iterativně je vylepšujte.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN: 978-0321965516
- Steve Krug: Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability, ISBN: 978-0321657299
- Joel Marsh: UX for Beginners: A Crash Course in 100 Short Lessons, O'Reilly 2016
- Susan M. Weinschenk: 100 věcí, které by měl každý designér vědět o lidech, Computer Press, Brno 2012
- Android Developers: https://developer.android.com/index.html

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3, značné rozpracování bodů 4 a 5.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 21. dubna 2021

# Abstrakt

Cieľom tejto práce je vytvorenie webovej aplikácie, ktorej účelom bude riadiť vyvolávanie ľudí zapísaných v poradovníku. Aplikácia tak pomôže lepšie zorganizovať ľudí čakajúcich na poradie a zároveň im ušetrí množstvo času, nakoľko sa do poradia môžu zapísať ešte pred príchodom na dané miesto. Výsledná aplikácia vyvíjaná pomocou frameworku *Django* s využitím *PostgreSQL* databázy a ďalších technológií, je umiestnená na *Heroku*. Webová aplikácia je responzívna a prispôsobená tak aj mobilným zariadeniam či tabletu. Používateľ vytvorí poradovník – „semafor“ a na jeho spravovacej stránke vidí zoznam ľudí zapísaných v poradovníku. Môže prihlasovanie do poradia ukončiť, otvoriť alebo zavolať ďalšieho čakajúceho v poradí. Ľudia, ktorí sa chcú do poradia zapísať, už vopred vedia, koľko ľudí pred nimi čaká na svoje poradie. Po zapísaní do poradia majú možnosť sledovať svoje poradie, ktoré sa každou zmenou aktualizuje bez nutnosti obnovy stránky. Obsahom práce je oboznámenie s princípom vývoja webových aplikácií, existujúcimi riešeniami podobného problému, návrhom aplikácie *Organize Semaphore*, použitými technológiami a implementáciou aplikácie.



# Abstract

The aim of this work is to create a web application, the purpose of which will be to manage the queue of people. The application will help to better organize the people waiting for their turn and save them a lot of time, as they can enroll to the queue before arriving at the place. The application, developed using the *Django* framework and the *PostgreSQL* database and other technologies, is located on *Heroku*. The web application is responsive and thus adapted to mobile devices or tablets. The user creates the queue - „traffic light“ and sees a list of people enrolled in the queue on his administration page. You can end or open enrolling to queue, or call next person waiting in queue. People who want to join the queue already know, how many people are waiting for their turn. After join, they have the ability to control their position, which is updated without having to refresh the page. The content of the bachelor thesis is acquaintance with the principle of web application development, existing solutions to a similar problem, the design of the application *Organize Semaphore*, the technologies used and the implementation of the application.



# Kľúčové slová

webová aplikácia, Python, Django, PyCharm, PostgreSQL, riadenie poradia, responzivita

# Keywords

web app, Python, Django, PyCharm, PostgreSQL, queue management, responsivity



# Citácia

FAŠÁNKOVÁ, Erika. *Jednoduchá mobilní a webová aplikace pro organizování schůzek*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

# Jednoduchá mobilní a webová aplikace pro organizování schůzek

## Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením pána prof. Ing. Adama Herouta, Ph.D. Uviedla som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpala.

.....
Erika Fašánková
18. mája 2021


## Poďakovanie

Veľké poďakovanie patrí vedúcemu práce prof. Ing. Adamovi Heroutovi, Ph. D., za pomoc pri výbere témy a ujasnení jej prínosu, venovaný čas, cenné rady a najmä trpezlivosť, ktorú so mnou mal počas celého obdobia písania práce. Poďakovanie tiež patrí mojej rodine a všetkým priateľom, ktorí ma podporovali v priebehu celého štúdia.

# Obsah

# Kapitola 1

# Úvod

Cieľom mojej bakalárskej práce je vytvoriť jednoduchú webovú aplikáciu, prostredníctvom ktorej si používateľ bude môcť vytvoriť poradovník na vyvolávanie ľudí v ňom zapísaných. Princíp vyvolávacieho systému je doplnený o obrázok dopravného semaforu, ktorý svojimi farbami upozorňuje ľudí čakajúcich v poradí, aby ešte počkali, pripravili sa, a že už sú na rade.

K vytvoreniu aplikácie ma viedli vlastné skúsenosti. Organizácia času je problém, ktorý ľudia riešia každý deň. Chodia do práce, starajú sa o rodinu, študujú, ale zároveň potrebujú ísť k lekárovi, na študijné oddelenie, na úrad, do poisťovne a na ďalšie rôzne miesta, kde musia počítať s tým, že strávia niekoľko minút, možno až hodiny cestovaním a čakaním v rade na svoje poradie. Účelom aplikácie je uľahčiť ľuďom plánovanie času tým, že sa môžu zaradiť do poradia z pohodlia domova a ušetriť si tak čas strávený v čakárni, ktorý môžu využiť omnoho lepšie. V poradovníku môžu byť zapísaný už počas cesty na dané miesto. Pre lekárov, úradníkov, pracovníkov na študijnom oddelení či iných podobných miestach, predstavuje aplikácia alternatívu už k existujúcim spôsobom riešenia vyvolávacieho systému. Aplikácia *Organize Semaphore*[1] je ekonomicky aj ekologicky menej náročná, ako napríklad panely na tlačenie papierových lístkov.

Používanie webovej aplikácie je prispôsobené tak, aby bolo čo najjednoduchšie. Správca poradia sa zaregistruje a po prihlásení má možnosť vytvoriť si „dopravný semafor“, ktorým bude riadiť poradie čakajúcich. Vytvorením „semaforu“ sa vygeneruje URL, na ktorom bude dostupné prihlásenie do daného „semaforu“. Toto URL si môže používateľ umiestniť na svoje webové stránky, rozposlať svojim klientom mailom alebo akýmkoľvek iným spôsobom. Ľudia, ktorí sa budú chcieť zapísať do poradia sa nepotrebujú registrovať. Stačí napísať svoje meno a jedným kliknutím na tlačítko sa zapíšu. Následne môžu sledovať svoje poradie.

Webová aplikácia *Organize Semaphore* je vytvorená v jazyku *Python* pomocou frameworku *Django*. Je prispôsobená pre obrazovky stolných počítačov, tabletu aj mobilného telefónu. V nasledujúcich kapitolách tejto práce sú opísané už existujúce riešenia podobného problému (kap. 2.3), technológie použité pri vývoji aplikácie (kap. 4), návrh aplikácie *Organize Semaphore* (kap. 5) a implementácia jednotlivých častí aplikácie (kap. 6).

[1] Úvodná stránka aplikácie – https://organize-semaphore.herokuapp.com/

# Kapitola 2

# Webová aplikácia na riadenie a sledovanie poradia čakajúcich

Obsah tejto kapitoly je rozdelený do troch častí. Prvá časť objasňuje dôvody k vytvoreniu webovej aplikácie. V druhej časti je vysvetlený princíp a využitie vytvorenej aplikácie a posledná časť obsahuje popis existujúcich riešení podobného problému, ich výhody a nevýhody.

## 2.1 Motivácia pre vytvorenie aplikácie

Motiváciou pre vytvorenie aplikácie *Organize Semaphore* boli moje vlastné skúsenosti. Keď som išla na úrad vyzdvihnúť si doklady alebo v škole na študijné oddelenie, nevedela som koľko času mi to zaberie. Až keď som prišla na dané miesto, zistila som, že v rade predo mnou stojí určitý počet ľudí. Veľakrát som sa dostala do situácie, že som na dané miesto cestovala hodinu a po príchode som zistila, že počet ľudí predo mnou je príliš veľký. Musela som sa rozhodnúť či ostanem čakať v rade aj za cenu toho, že už nestihnem to, čo som mala naplánované ďalej počas dňa alebo či odídem, čo spôsobí stratu poradia aj hodinu času, ktorú som strávila cestovaním. V takýchto chvíľach som si priala vedieť dopredu, koľko ľudí sa momentálne v rade nachádza a mať možnosť sa do neho zaradiť ešte počas cesty na dané miesto.

Na trhu sa už nejaké riešenia vyvolávacieho systému vyskytujú a sú popísané v kapitole 2.3. Ja som sa pri vývoji aplikácie *Organize Semaphore* rozhodla klásť dôraz na jednoduchosť a komunikáciu v reálnom čase. Nakoľko ide o webovú aplikáciu, nie je potrebná žiadna inštalácia. Je tak dostupná pre každého, kto je pripojený k internetu.

## 2.2 Princíp a využitie aplikácie

Aplikácia *Organize Semaphore* funguje na princípe vyvolávacieho systému a grafického znázornenia pomocou dopravného semaforu. Používatelia, ktorý sa zaradia do poradia, môžu sledovať svoji pozíciu na základe informácií o ich poradí, predpokladanom čase čakania a tiež farby „semaforu“.

Na vytvorenie „semaforu“, ktorý bude riadiť poradie, je potrebná registrácia a následné prihlásenie používateľa. Používateľ po prihlásení vidí nástenku so svojimi „semaformi“, kde má tiež možnosť upraviť svoje profilové údaje alebo účet zrušiť. Ak zatiaľ žiadny „semafor“

vytvorený nemá, môže si ho jednoducho vytvoriť zadaním názvu a odhadovaného počtu minút na jedného klienta.

Každý „semafor“ má svoju spravovaciu stránku. Správca na nej môže otvoriť alebo uzavrieť prihlasovanie do poradia a zavolať ďalšieho čakajúceho v poradí. Táto stránka tiež obsahuje URL, na ktorom je dostupná verzia pre ľudí, ktorí sa chcú do poradia daného „semaforu“ zaradiť. Ľudia tu môžu vidieť aktuálny počet čakajúcich ľudí a podľa toho sa rozhodnúť, či sa do poradia zaradia. Následne po zaradení môžu sledovať svoj stav.

Webovú aplikáciu je možné využiť kdekoľvek, kde je potrebný vyvolávací systém, napríklad už v spomínanej škole, na študijnom oddelení, na úradoch či u lekára. Je to lacnejšia, ekologickejšia aj užívateľsky príjemnejšia cesta ako nainštalovanie panelu na tlačenie vyvolávacích lístkov, ručné zapisovanie či čakanie v rade. Porovnanie existujúcich riešení je podrobnejšie rozobrané v nasledujúcej kapitole 2.3.

## 2.3 Súčasné existujúce riešenia vyvolávacieho systému

Pred samotným vývojom aplikácie je dôležitý prieskum už existujúcich riešení daného problému. Dnes už takmer každý človek má skúsenosť s nejakým druhom vyvolávacieho systému. V minulosti ľudia často stáli fyzicky v rade a čakali tak na svoje poradie. Postupne sa začali ľudia do poradia zapisovať. Najskôr ručne na papier, napríklad u lekára. Dnes sa v takýchto zariadeniach často využívajú rôzne panely na tlačenie lístkov s poradovým číslom. Predchádzajúce spôsoby úplne nezanikli. Aj v súčasnosti sa môžeme ešte v niektorých zariadeniach stretnúť s čakaním v rade alebo zapísaním sa na papier. Tieto spôsoby však nie sú veľmi efektívne. Čas, ktorý človek strávi čakaním v rade, sa dá využiť často omnoho lepšie.

### Panely na tlačenie lístkov a čítačky

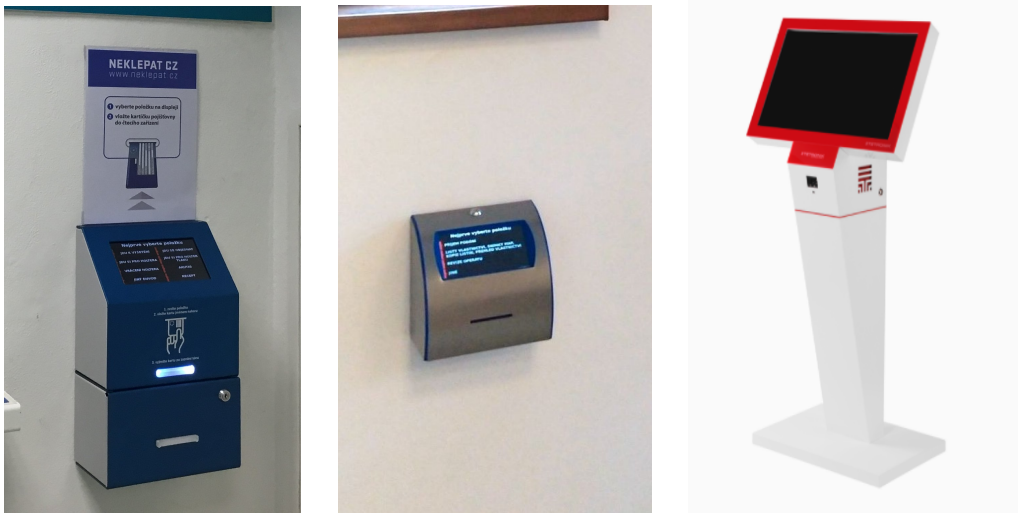
V dnešnej dobe je snaha o digitalizáciu a upustenie od papierovania. Aj preto vznikli rôzne systémy a aplikácie, ktoré sa o toto snažia. Panely na tlačenie lístkov sú dnes asi najbežnejšou formou organizácie poradia čakajúcich. Takéto zariadenia je možné nájsť na rôznych úradoch, v bankách, poisťovniach alebo na pošte.

V rámci prieskumu boli skúmané panely od spoločnosti *Neklepat CZ*[1], *Vyvolávák CZ*[2] a *Tetronik*[3] (obrázok 2.1). Výhodou panelov je, že sú už zaužívané, ľudia ich poznajú a v rámci jedného zariadenia majú klienti výber dôvodu návštevy. Nemusia tak čakať v rade so všetkými ostatnými čakajúcimi, ale môžu sa zaradiť do poradia priamo k danej osobe, ktorá sa ich problémom zaoberá. Veľkou nevýhodou u týchto panelových zariadení je ich cena. Minimálna cena zariadenia sa pohybuje od 20 000,00 CZK. Okrem toho sa často platí k tomu aj pravidelný mesačný poplatok za technickú podporu.

Skúmané spoločnosti ponúkajú okrem panelov aj špecifické zariadenia určené do čakárne u lekára. Rozdiel oproti panelom tlačiacich lístky spočíva v tom, že tieto zariadenia fungujú ako čítačky. Pacient do nich vloží svoju kartičku poistenia. Zariadenie načíta údaje o pacientovi z kartičky a prenesie ich do programu lekára. Vzhľad zariadenia jednotlivých spoločností je veľmi podobný a je možné ho vidieť na obrázku 2.2. Výhodou čítačky je prepojenie so systémom lekára a automatické vyplnenie údajov o pacientovi. Lekár tak dokáže zavolať pacientov priamo z ambulancie. Postačí mu v programe kliknúť na tlačítko, nemusí vchádzať do čakárne. Meno vyzvaného pacienta sa zobrazí v čakárni prostredníctvom

[1] Produkty spoločnosti Neklepat CZ – https://neklepat.cz/nase-produkty
[2] Produkty spoločnosti Vyvolávak CZ – https://vyvolavak.cz/nase-produkty/
[3] Produkty spoločnosti Tetronik – https://tetronik.cz/produkty/

televízora, monitoru alebo inej podobnej techniky. Nevýhodou je, rovnako ako u panelov na tlačenie lístkov, ich vysoká nákupná cena a tiež ďalšie mesačné výdavky na technickú podporu.



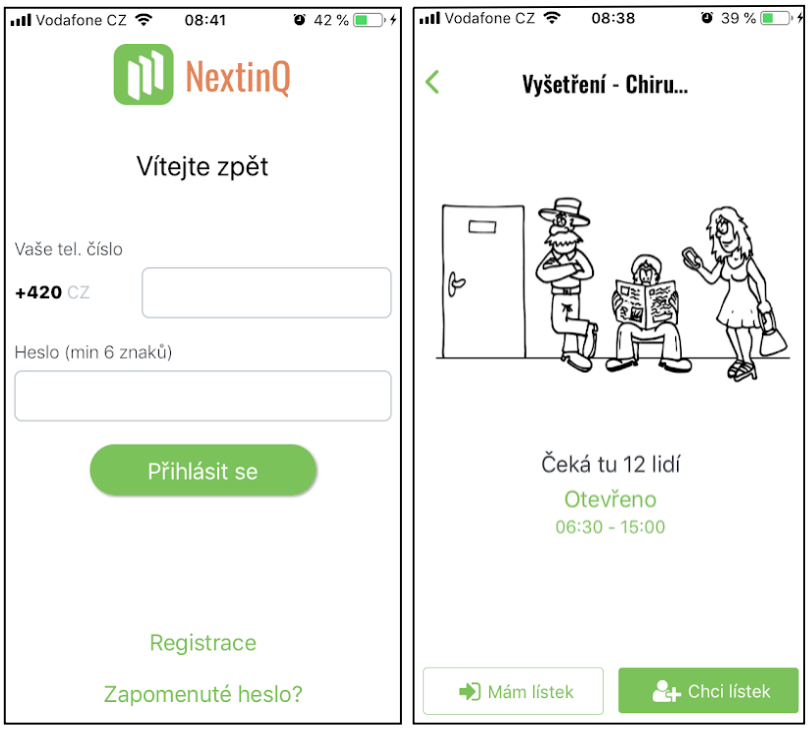

Obr. 2.1: **Fotografie zariadení na tlačenie poradových lístkov.** Na obrázku sú fotografie zariadení od spoločností Neklepat CZ, Vyvolávák CZ a Tetronik, ktoré boli prevzaté z ich oficiálnych stránok. Princíp týchto zariadení spočíva vo výbere dôvodu návštevy na dotykovom displeji a následnom vytlačení papierového lístka s poradovým číslom. K daným panelom bývajú často nainštalované aj monitory, televízie alebo iné zobrazovacie prostriedky, na ktorých sa vždy vyobrazí poradové číslo nasledujúceho návštevníka.


Obr. 2.2: **Zariadenie na snímanie kartičky zdravotného poistenia.** Obrázok bol prevzatý z oficiálnych stránok spoločnosti *Neklepat CZ* a ich propagačného videa. Čítačky prinášajú zjednodušenie práce najmä pre lekárov a zdravotných sestry. Po načítaní kartičky zdravotného poistenia dokážu pár klikmi vyplniť údaje o pacientovi, sú informovaní o dôvode ich návštevy a dokážu si podľa toho pacientov volať tak, aby využili svoj pracovný čas čo najefektívnejšie. Samotným pacientom však neprináša takmer žiadnu informovanosť o svojom poradí.

### Mobilné aplikácie a webové aplikácie

Veľa mobilných aplikácií na trhu, zaoberajúcich sa vyvolávaním ľudí z poradovníka, je viazaných na fyzické zariadenie. Používateľ si pomocou panelu vytlačí lístok s poradovým číslom, ktorý obsahuje QR kód alebo číselný kód. Prostredníctvom mobilného telefónu môže používateľ načítať QR kód, ktorý ho presmeruje na mobilnú aplikáciu. V nej môže sledovať svoje poradie. Rovnako to funguje aj s číselným kódom. Po jeho zadaní do danej aplikácie sa návštevníkovi zobrazia informácie o jeho poradí.

Prvá skúmaná aplikácia bola *NextinQ – inteligentní čekárna*[4]. Aplikácia je dostupná v obchode *Google Play*[5] aj *App Store*[6]. Táto aplikácia funguje na rovnakom princípe. Zákazník si v aplikácii zaregistruje prevádzkovateľa na základe PID kódu uvedeného na vytlačenom lístku a následne pomocou PIN kódu z lístku sleduje svoje poradie.



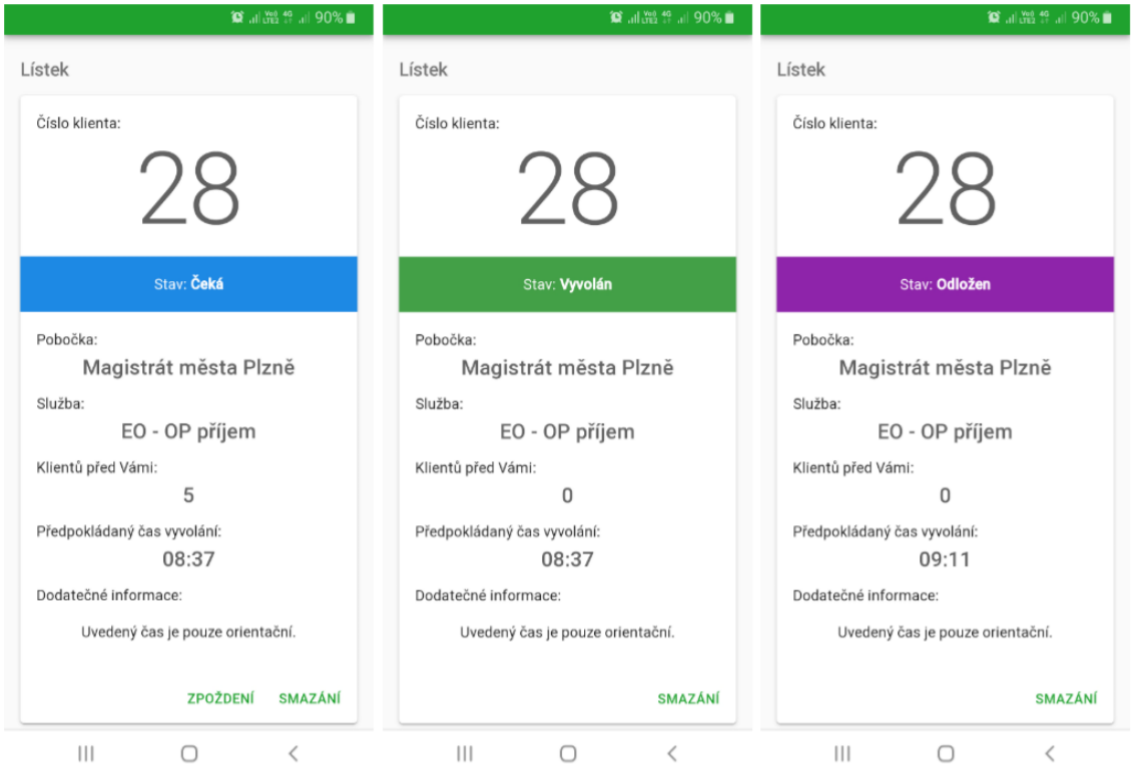

Obr. 2.3: **Ukážka mobilnej aplikácie *NextinQ – inteligentní čekárna*.** Výhodou tejto aplikácie je, že umožňuje klientom prihlásiť sa do poradia aj bez vytlačenia papierového lístku. Môže tak urobiť z pohodlia domova a tiež to, že návštevník je informovaný upozornením na zmenu svojho poradia. Veľkou nevýhodou však je skutočnosť, že na to, aby lekár alebo iné zariadenie mohlo túto aplikáciu poskytovať svojim pacientom a zákazníkom, musí mať nainštalované aj zariadenie na tlač papierových lístkov. Je teda nutné počítať opäť s vysokou cenou na poskytovanie tejto služby. Taktiež návštevník, ktorý sa chce zaradiť do poradia sa musí najskôr zaregistrovať prostredníctvom svojho telefónneho číslo, čo môže veľa ľudí odradiť.

[4] Webová stránka NextinQ – https://www.nextinq.com/end-user

[5] https://play.google.com/store/apps/details?id=com.nextinq&pcampaignid=MKT-Other-global-all-co-prtnr-py-PartBadge-Mar2515-1

[6] https://apps.apple.com/cz/app/nextinq/id1370884582

V štatutárnom meste Ústí nad Labem v Českej republike nedávno zaviedli nový systém, ktorý uľahčí obyvateľom mesta zaradiť sa do poradia na magistráte a kontrolovať ho. Webová aplikácia *Virtuálny Kiosek*[7], od už spomínanej spoločnosti *Tetronik*, ktorá sa zaoberá aj panelmi na tlačenie lístkov, umožňuje výber požadovanej služby prostredníctvom webového prehliadača. Po výbere služby obdrží návštevník *On-line lístok*[8] – poradové číslo (obrázok 2.4). V aplikácii má návštevník možnosť lístok „vytlačiť“ on-line a zaradiť sa do poradia cez internet, alebo načítať QR kód z papierového lístku vytlačeného z panelu.




Obr. 2.4: **Grafická ukážka *On-line lístku.*** Klient môže sledovať svoje poradie, počet ľudí pred ním a predpokladaný čas vyvolania. Výhodou aplikácie je tiež funkcia, ktorou má klient možnosť odloženia svojho poradia, v prípade že to z nejakých dôvodov potrebuje. Vyvolanie klienta tak nastane najskôr až po uplynutí času, ktorý zadal ako oneskorenie. Obrázok a informácie o *On-line lístku* pochádzajú z informačného letáku.

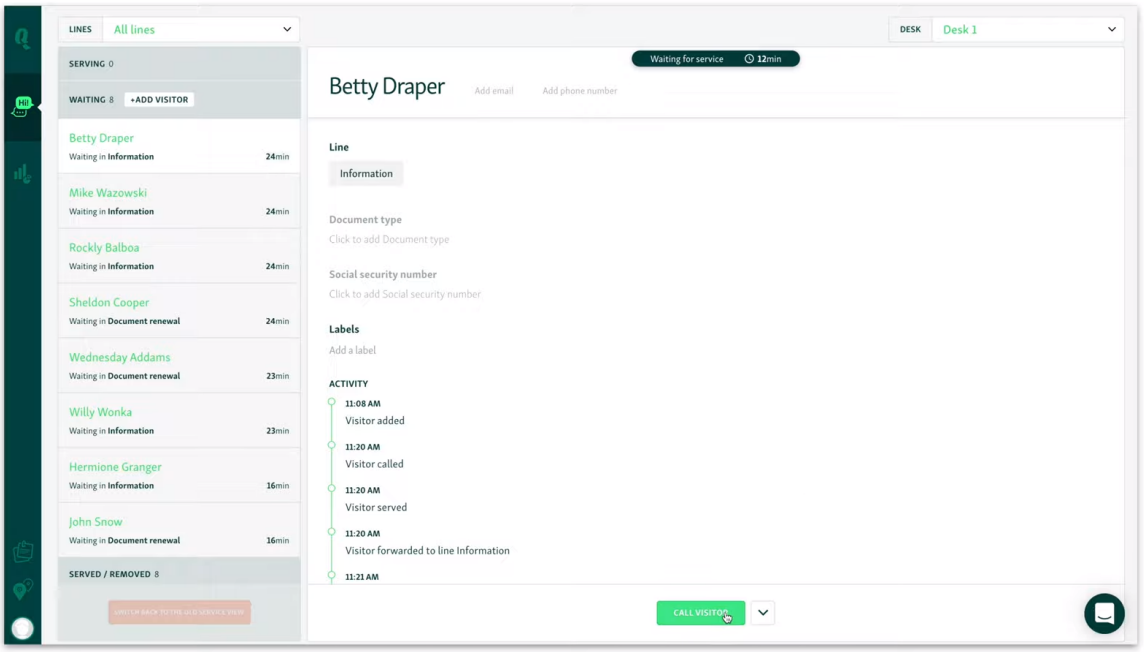
Existuje tiež veľa aplikácií s obrovským množstvom rôznych funkcií, ako je napríklad prihlásenie sa do poradovníka niekoľko dní dopredu na presne stanovený čas. Medzi takéto aplikácie patrí aj *Qminder*[9] a *TimeTrade*[10]. Všetky tieto funkcie môžu byť výhodami, ale veľké množstvo funkcií prináša aj vyššiu komplikovanosť aplikácie. Pre prípady použitia, na ktoré som sa zamerala pri vývoji aplikácie *Organize Semaphore*, tieto funkcie nie sú potrebné a zbytočne komplikujú celkové používanie aplikácie.

[7] Manuál Virtuálneho kiosku – https://www.usti-nad-labem.cz/files/cz/uredni-portal/obcan/objednejte-se-online/spo_webkiosek-manual.pdf

[8] Informačný leták On-line lístku – https://www.usti-nad-labem.cz/files/cz/uredni-portal/obcan/objednejte-se-online/spo_online-listek.pdf

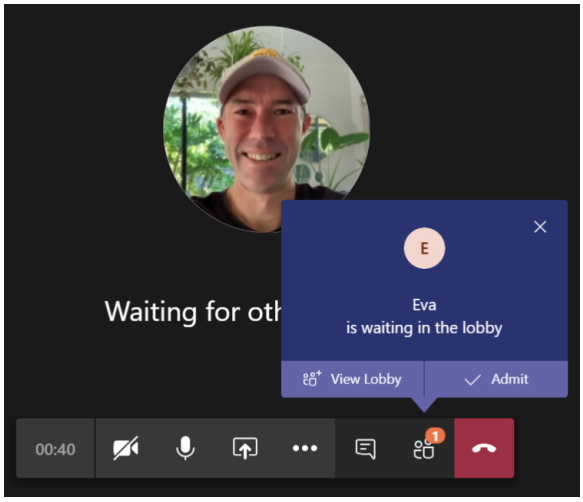
[9] Webová stránka aplikácie Qminder – https://www.qminder.com/

[10] Webová stránka aplikácie TimeTrade – https://www.timetrade.com/




Obr. 2.5: **Ukážka aplikácie *Qminder*.** Aplikácia okrem zoznamu čakajúcich ponúka aj zoznam obslúžených klientov. Ku každému klientovi je možné priradiť rôzne štítky a tiež vidieť celkovú aktivitu klienta znázornenú prostredníctvom časovej osi – od pridania do poradia, cez zavolanie klienta až po jeho obslúženie.

Niektoré aplikácie zamerané na komunikáciu cez Internet (napríklad *Microsoft Teams* [2]) obsahujú spôsob, ktorým môžu vpúšťať ďalších ľudí do konverzácie z takzvanej „predsiene“. Ide taktiež o riešenie zaoberajúce sa poradím ľudí čakajúcich na hovor. Osoba, ktorá sa pripája k hovoru klikne na tlačítko `Join`, ktorým sa pripojí do konverzácie. Dostane sa do „predsiene“, kde sa im zobrazí oznámenie o tom, aby počkali, že im o chvíľu niekto potvrdí hovor. Ako to vyzerá na strane prijímajúceho ukazuje obrázok 2.6.




Obr. 2.6: **Ukážka aplikácie *Microsoft Teams.*** Prijímajúcemu účastníkovi hovoru sa zobrazí upozornenie o čakajúcej osobe v „predsieni“. Má možnosť prijať čakajúceho alebo zamietnuť jeho žiadosť.

# Kapitola 3

# Princípy vývoja webových aplikácií

V prvej podkapitole je popísaný model MVC, ktorý sa pri vývoji webových aplikácií v dnešnej dobe využíva najčastejšie. V ďalšej časti kapitoly je vysvetlený rozdiel medzi *User Experience* (UX) a *User Interface* (UI). Kapitola 3.3 pojednáva o dôležitosti responzívneho vzhľadu webových aplikácií a v závere kapitoly sú objasnené typy testovania.

## 3.1 Architektúra Model-View-Controller

*Model-View-Controller* [15] (skrátene MVC) je architektonický vzor, ktorý rozdeľuje výsledný sofvér na menšie komponenty. Vznikol skôr ako webové prehliadače, preto sa pôvodne používal pri tvorbe desktopových grafických užívateľských rozhraní. Jeho použitie sa postupne rozšírilo aj na vývoj webových či mobilných aplikácií.

Hlavná myšlienka architektúry *Model-View-Controller* je oddelenie logiky a výstupu. Aplikácia sa tak stáva prehľadnejšou a jednoduchšie sa udržuje. Jednotlivé komponenty môžu byť opakovane použité, nasadené a testované nezávisle. Architektúra *Model-View-Controller* pozostáva z troch komponentov: *Model*, *View* a *Controller*.

Na architektúre MVC je založených niekoľko frameworkov pre tvorbu webových aplikácií v rôznych programovacích jazykoch. Medzi takéto frameworky patrí napríklad framework *Symfony*[1], napísaný v jazyku *PHP*[2] alebo *Ruby on Rails*[3], ktorý vychádza z jazyka *Ruby*[4]. Patrí medzi ne aj framework *Django*, napísaný v jazyku *Python*, ktorého podrobnejší popis obsahuje kapitola 4.1.

### Model

*Model* sa zaoberá logikou aplikácie a je pripojený k databáze. Jeho úlohou sú výpočty, spracovanie databázových požiadavkov či validácia. Reaguje na požiadavky *Controlleru*, komunikuje s databázou a potrebné dáta vracia naspäť *Controlleru*. Nezaoberá sa tým, ako budú výsledné dáta prezentované užívateľovi.

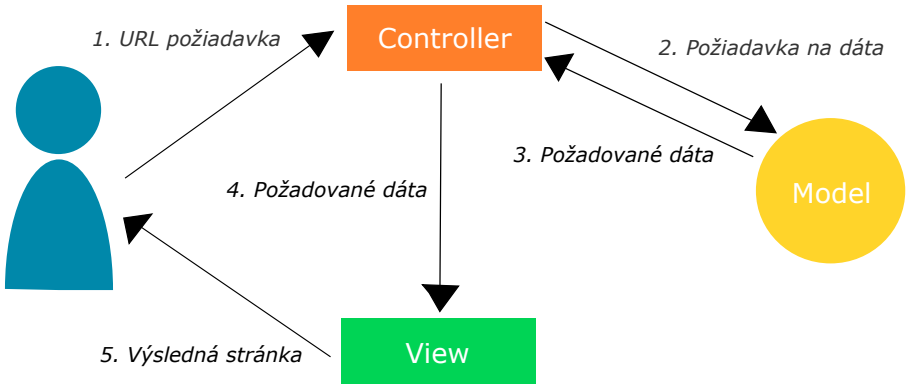
[1] Oficiálna stránka frameworku Symfony – https://symfony.com/
[2] Oficiálna stránka jazyka PHP – https://www.php.net/
[3] Oficiálna stránka frameworku Ruby on Rails – https://rubyonrails.org/
[4] Oficiálna stránka jazyka Ruby – https://www.ruby-lang.org/en/

### View

Úlohou *View* je zobrazenie výstupných dát užívateľovi. Pri tvorbe webových aplikácií obsahuje najčastejšie HTML šablónu, do ktorej je možné vložiť premenné, podmienky či vykonávať cykly.

### Controller

*Controller* prepája *Model* a *View*. Slúži ako prostredník, s ktorým komunikujú ostatné komponenty aj samotný užívateľ. Od užívateľa prijme vstupné dáta a posiela ich príslušnému *Modelu*. Spracuje dáta prijaté od *Modelu*, ktoré následne odošle *View*.




Obr. 3.1: **Životný cyklus webovej stránky** [16]. Cyklus začína, keď užívateľ zadá požadované URL stránky do webového prehliadača. Túto požiadavku zachytí smerovač. Na základe parametrov zadaných užívateľom, smerovač rozpozná *Controller*, ktorý užívateľ volá. *Controller* zavolá model s požiadavkom na dané dáta. *Model* požadované dáta nájde v databáze a vracia ich *Controlleru*. Následne *Controller* dáta predáva *View*. *View* tieto dáta prijíma, vkladá ich do šablóny a výslednú stránku zobrazí používateľovi.

## 3.2 Rozdiel medzi UX a UI

Pojmy *User Experience* (skrátene UX) [10] a *User Interface* (skrátene UI) bývajú často zameňované. Každé zariadenie má svoje používateľské rozhranie. Rovnako aj webové aplikácie majú svoje rozhranie. Sú to tlačidlá, formuláre, informácie, všetko čo používateľ vidí a s ktorými pracuje. Takmer každá webová aplikácia má nejaký formulár na prihlásenie, registráciu alebo kontaktovanie autora. To ako pôsobí na používateľa je úlohou UX.

Kvalitné UX predstavuje nekonečný proces testovania a iteratívneho vylepšovania. Veľmi dôležité je porozumieť koncovým používateľom, ich hodnotám a motívom. Na získanie potrebných informácií o používateľoch a ich správaní sa využíva prieskum v podobe dotazníka či osobného rozhovoru. Často sa využívajú nákresy výslednej webovej aplikácie, prostredníctvom ktorých sa na skupine ľudí testuje, ako používatelia s jednotlivými prvkami pracujú.

Návrh používateľského rozhrania je súčasťou procesu návrhu UX. Zahŕňa návrh navigačných prvkov, ponuky, farieb, tlačidiel či typ písma a tok textu tak, aby bolo zabezpečené, že s výsledným návrhom dosiahne používateľ svoje ciele.

## 3.3 Responzívny dizajn webových aplikácií

Prístup k návrhu grafického používateľského rozhrania (GUI) používaný na tvorbu obsahu, ktorý sa plynulo prispôsobuje veľkosti displeja označujeme responzívny dizajn [3]. Používatelia k webovým aplikáciám častejšie pristupujú z mobilných zariadení ako prostredníctvom stolných počítačov. To je hlavný dôvod toho, prečo je responzívny dizajn populárny a zároveň dôležitý pri tvorbe webových aplikácií.

Hlavnou výhodou responzívnej webovej aplikácie je, že nepotrebujeme viac verzií jednej webovej stránky. Používateľ sa dokáže rýchlo a jednoducho zorientovať bez zbytočného posúvania sa po stránke či akéhokoľvek približovania obrazovky.

Responzívny dizajn sa opiera o tri základné princípy [7]:

- **Systém Fluid Grid** – jednotlivý prvky zaberajú vždy rovnaký percentuálny podiel priestoru bez ohľadu na veľkosť displeja.
- **Použitie Fluid Image** – obrázky majú prednastavenú rovnakú veľkosť a konfiguráciu z jedného zariadenia na iné, čo môže spôsobiť, že obrázky sa neprispôsobia veľkosti obrazovky a budú pôsobiť nekonzistentne (pre zaistenie zmenšenia obrázku na menších displejoch sa používa príkaz CSS: `img{max-width:100%;}`).
- **Media Queries** – filtre, pomocou ktorých sa zisťuje veľkosť obrazovky používateľa a na základe zistenej veľkosti sa mu zobrazí príslušné rozloženie obsahu stránky.

Na tvorbu responzívneho dizajnu existuje niekoľko nástrojov, ako napríklad *Bootstrap* (kap. 4.6), *H5P*[5], *Gomo*[6] či *Elucidat*[7].

## 3.4 Testovanie webových aplikácií

K jednej webovej aplikácií môžu pristupovať používatelia z rôznych webových prehliadačov. Každý webový prehliadač má rôzne schopnosti vykreslovania webového obsahu a je dôležité, aby bolo zaistené očakávané správanie v každom z nich. Testovanie má preto pri vývoji webových aplikácií dôležitú rolu. Existuje niekoľko typov testovania [11] webových aplikácií: testovanie obsahu, testovanie užívateľského rozhrania, testovanie použiteľnosti, testovanie kompatibility, testovanie úrovne komponentov, testovanie navigácie, testovanie konfigurácie a testovanie zabezpečenia.

### Testovanie obsahu

Chyby, ktoré sa snaží testovanie obsahu odhaliť sú najmä nesprávne informácie, preklepy alebo typografické chyby. Často sa využíva automatická kontrola pravopisu a gramatiky. Niektoré syntaktické chyby však odhalí až sám človek.

Dynamické webové aplikácie sú vytvárané v reálnom čase pomocou dát získaných z databázy. Je potrebné testovať či dáta predané medzi klientom a serverom sú správne. Ďalej sa testuje či aplikácia správne spracováva skripty a extrahuje, prípadne formátuje užívateľské dáta.

[5] https://h5p.org/
[6] https://www.gomolearning.com/
[7] https://www.elucidat.com/

### Testovanie používateľského rozhrania

Úlohou testovania používateľského rozhrania je odhaliť chyby v spôsobe, akým rozhranie implementuje sémantiku funkčnosti webovej aplikácie či zobrazenie obsahu. Testuje sa vizuálne spracovanie aplikácie a funkčnosti jednotlivých prvkov.

### Testovanie použiteľnosti

Testovanie použiteľnosti je veľmi podobné testovaniu používateľského rozhrania. Z tohto hľadiska testujeme animácie, tlačítka, ovládanie, grafiku, formuláre, odkazy, navigáciu či selektory. Výsledkom testovania by malo byť dosiahnutie týchto cieľov:

- **Interaktivita** – význam prvkov webovej aplikácie (tlačítko, ukazatele, rozbalovacia ponuka) by mal byť zrejmý a jednoduco pochopiteľný.
- **Rozloženie** – užívateľ rýchlo a bez premýšľania nájde jednotlivé prvky na obrazovke.
- **Čitateľnost** – obsah textu stránky a jeho grafické prevedenie by malo byť jednoduché na pochopenie.
- **Estetika** – dôležitý je aj výber farieb, typ a štýl písma, aby sa používateľ cítil príjemne pri používaní danej webovej aplikácie.
- **Vlastnosti displeja** – webová aplikácia by sa mala prispôsobiť veľkosti displeja (kap. 3.3).
- **Personalizácia** – prispôsobenie aplikácie potrebám daného používateľa.
- **Prístupnosť** – aj používatelia so zdravotnými obmedzeniami by mali byť schopní aplikáciu plnohodnotne využívať.

### Testovanie kompatibility

Webová aplikácia by mala fungovať správne v rôznych webových prehliadačoch na rôznych operačných systémoch. Problémy s kompatibilitou často vyplývajú zo zanedbaného testovania použiteľnosti (napr.: obsah neresponzívnej stránky môže byť skrytý a nepoužiteľný). Testovanie kompatibility pomáha tieto chyby odhaliť skôr, ako sa aplikácia dostane k širokej verejnosti.

### Testovanie na úrovni komponentov

Tento typ testovania sa tiež označuje ako testovanie funkcií. Jeho úlohou je odhaliť chyby vo funkciách aplikácie. Testovacie prípady na úrovni komponentov sú často riadené vstupom na úrovni formulárov. Napríklad pri zadávaní adresy v internetovom obchode používateľ zadáva PSČ. Testovanie na úrovni komponentov pomáha odhaliť chyby spôsobené zadaním nesprávneho alebo neúplného PSČ.

### Testovanie navigácie

Aby sa používateľ dostal k tomu, čo hľadá, môže na to využiť niekoľko rôznych spôsobov. Každý z týchto navigačných mechanizmov by mal byť otestovaný:

- **Navigačné odkazy** – patria tu interné odkazy webovej aplikácie, externé odkazy na iné aplikácie alebo kotvy v rámci webovej stránky.
- **Presmerovanie** – výsledkom testovania je zamedzenie výskytu chýb, ako je presmerovanie na neexistujúce alebo odstránené URL.
- **Mapa stránky** – po kliknutí na odkaz by používateľovi sa zobrazí správny obsah.
- **Interný vyhľadávač** – testovanie overuje presnosť, úplnosť a pokročilé funkcie vyhľadávania v rámci aplikácie.

### Testovanie zabezpečenia

Testovanie zabezpečenia sa zameriava na neoprávnený prístup k obsahu a funkciám webovej aplikácie. Jednotlivé testy preverujú sieťovú komunikáciu, v ktorej dochádza k prenosu dát z klienta na server alebo zo serveru na klienta.

Na strane klienta často sledujeme už existujúce chyby zabezpečenia vo webových prehliadačoch. Medzi typické bezpečnostné otvory patrí pretečenie vyrovnávacej pamäte alebo neoprávnený prístup k súborom cookie. Na strane serveru medzi najčastejšie chyby zabezpečenia patria škodlivé skripty, ktoré sa môžu preniesť na klienta. Aby sa znížilo riziko výskytu týchto chýb zabezpečenia, je nutné implementovať jeden alebo viac z nasledujúcich prvkov zabezpečenia:

- **Firewall** – overuje, že informácie pochádzajú z legitímneho zdroja,
- **Autentifikácia** – verifikačný mechanizmus overujúci identitu klientov a serverov,
- **Šifrovanie** – mechanizmus kódovania, ktorý chráni citlivé dáta,
- **Autorizácia** – prístup do prostredia klienta či serveru je umožnený len jednotlivnom s autorizačnými kódmi.

# Kapitola 4

# Použité vývojové prostredie a technológie

Táto kapitola popisuje technológie a prostredia využité na vytvorenie aplikácie *Organize Semaphore*. Jednotlivé podkapitoly okrem popisu tiež uvádzajú výhody, kvôli ktorým boli dané technológie zvolené pri vývoji.

## 4.1 Programovací jazyk Python a framework Django

*Python*[1] je vysokoúrovňový programovací jazyk, ktorý vytvoril Guido Van Rossum. V roku 1991 sa stal dostupným pre verejnosť a odvtedy sa vyvinul na jeden z popredných programovacích jazykov súčasnosti. Jazyk *Python* sa vyznačuje nasledujúcimi vlastnosťami: [6]

- **Prenosnosť** – kód napísaný v jazyku *Python* je jednoducho prenositeľný z jedného operačného systému na iný.
- **Produktivita vývoja** – dĺžka kódu väčšinou odpovedá 1/3 až 1/5 ekvivalentného kódu napísaného v jazyku Java alebo C++. Z toho vyplýva, že projekt vyvíjaný v *Pythone* je dokončený rýchlejšie. Ďalšou výhodou je aj absencia zdĺhavej kompilácie a linkovania.
- **Rozsiahla knižnica** – základná knižnica je veľmi rozsiahla a knižnice tretích strán je možné získať prostredníctvom *Python Package Index (PyPI)*[2].
- **Kvalita softvéru** – *Python* sa zameriava na vysokú úroveň čitateľnosti, súdržnosť a kvalitu. Výhodou je aj jeho multiparadigmatická povaha. Jeho použitie je ako skriptovací jazyk, ale je možné využiť aj jeho objektovo-orientované, imperatívne a funkcionálne štýly programovania.
- **Softvérová integrácia** – ďalšou výhodou je, že *Python* je možné rozšíriť a interagovať s inými jazykmi a pôsobiť ako spojovací prostriedok medzi zložitými aplikáciami.

[1] Oficiálna stránka jazyka Python – https://www.python.org/
[2] Oficiálna stránka PyPI – https://pypi.org/

Okrem iného je možné *Python* použiť aj na tvorbu webových aplikácií na strane serveru. Medzi najznámejšie webové frameworky patrí *Flask*[3] alebo *Django*[4], ktorý bol aj použitý na vývoj aplikácie *Organize Semaphore*.

Cieľom *Djanga* [13] je vytvárať rýchle dynamické webové aplikácie. Framework využíva model MVC, ktorého princíp je popísaný v kapitole 3.1. Beží pod licenciou BSD, ktorá zaručuje, že webové aplikácie môžu byť voľne používané a modifikované.

Po vytvorení projektu príkazom `py -m django startproject` *`nazovProjektu`* Django automaticky vytvorí štruktúru projektu. Súbor `manage.py` umožňuje spúšťať príkazy na vytvorenie aplikácie, migrácie databázy, zber statických súborov či spustenie aplikácie. Novo vzniknutý adresár obsahuje niekoľko súborov:

- `settings.py` – obsahuje konfiguráciu projektu,
- `urls.py` – zahŕňa spôsob spracovania URL adresy a definuje, ktoré adresy smerujú do ktorej časti aplikácie,
- `wsgi.py` – obsahuje konfigurácia rozhrania pre nasadenie projektu na server.

Projekt môže pozostávať z viac aplikácií. Aplikácia vznikne príkazom `py manage.py startapp` *`nazovAplikacie`*. Adresárovú štruktúru aplikácie tvoria tieto súbory, ktoré *Django* opäť vytvorí automaticky:

- `admin.py` – administrácia aplikácie,
- `apps.py` – konfigurácia aplikácie,
- `models.py` – správa modelov databázy,
- `tests.py` – testy aplikácie.

Manuálne musíme doplniť súbor `urls.py`, ktorého obsahom budú odkazy na skripty spracujúce požiadavky zaslané cez dané URL.

## 4.2 Vývojové prostredie PyCharm

Webová aplikácia bola vyvíjaná v prostredí *PyCharm* od spoločnosti *JetBrains*[5] vo verzii Professional, nakoľko táto verzia podporuje webový framework Django. Verzia Professional je platená, ale pre študentov niektorých vysokých škôl je k dispozícii zdarma.

## 4.3 Databáza PostgreSQL

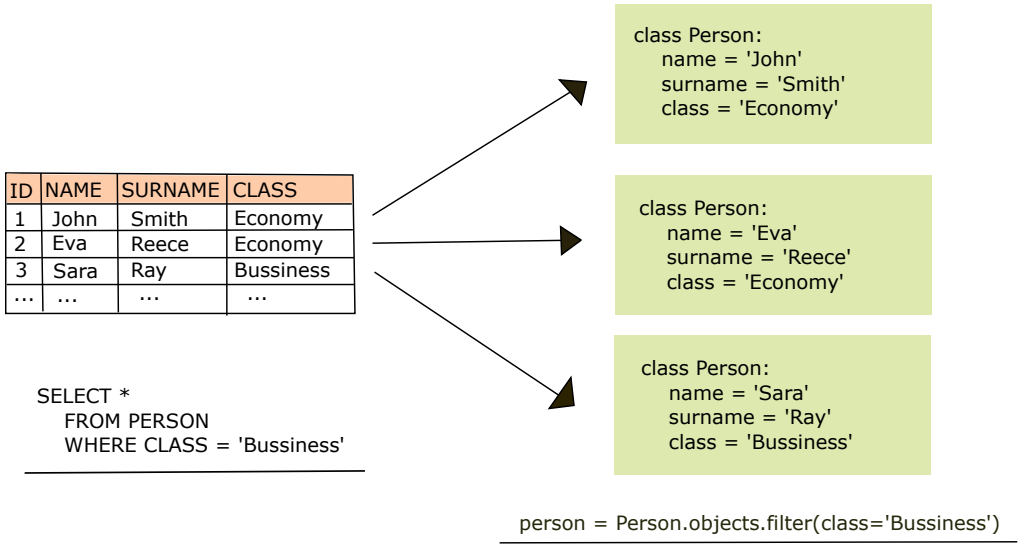
*PostgreSQL* [12] je voľne šíriteľný objektovo relačný databázový systém jednoduchý na naučenie. Získal si silnú reputáciu vďaka osvedčenej architektúre, spoľahlivosti, integrite dát či robustnej množine funkcií. Funguje na všetkých operačných systémoch a je v súlade s ACID. Primárne sa používa ako úložisko dát alebo dátový sklad pre mnoho webových, mobilných a analytických aplikácií. Podporuje rôzne programovacie jazyky ako *Java*, *Ruby*, *.NET*, *Perl*, *C/C++* alebo *Python*.

[3] Dokumentácia frameworku Flask – https://flask.palletsprojects.com/en/1.1.x/#
[4] Oficiálna stránka frameworku Django – https://www.djangoproject.com/
[5] Oficiálna stránka spoločnosti JetBrains – https://www.jetbrains.com/

*PostgreSQL* som si vybrala pri vývoji webovej aplikácie preto, že pre vývojárov v jazyku *Python* je voľba *PostreSQL* často štandardom vďaka tomu, že ovládače a ukážkový kód je dostatočne zdokumentovaný. *PostgreSQL* tiež dobre spolupracuje s frameworkom *Django* (kap. 4.1) a je podporované platformou *Heroku*[6]. Pre prácu s relačnými databázami v jazyku *Python* je potrebné použiť ovládač, často označovaný ako databázový konektor. Najbežnejšiou knižnicou ovládačov pre *PostreSQL* je `psycopg2`[7]. [5]

Na abstrahovanie spojenia medzi tabuľkami a objektami sa často používa objektovo-relačný mapovač (skrátene ORM) [4] na premenu relačných údajov z *PostgreSQL* na objekty, ktoré je možné využiť v *Python* aplikácií. ORM poskytuje vývojárovi možnosť písať kód priamo v jazyku *Python* namiesto *SQL* a tak vytvárať, čítať, aktualizovať či zmazať dáta v databáze. Takýto prístup prispieva k úrýchleniu vývoja webovej aplikácie.

Webový framework *Django* obsahuje vlastný vstavaný modul objektovo-relačného mapovania *Django ORM*, ktorý je vhodný pre jednoduché až stredne zložité databázové operácie.




Obr. 4.1: **Grafické znázornenie princípu ORM** [4]. Tabuľka *Person* zahŕňa záznamy ľudí, ich meno, priezvisko a informáciu o tom, či cestovali triedou *Economy* alebo *Bussiness*. Jednoduchým príkazom `SELECT` dokážeme z tabuľky vyňať iba záznamy o cestujúcich *Bussiness* triedou. ORM poskytuje prepojenie medzi tabuľkami relačných databáz, vzťahmi, poľami a *Python* objektmi. Rovnaký `SELECT` je tak možné zapísať priamo v jazyku *Python* ako je uvedené na obrázku.

## 4.4 WebSockets

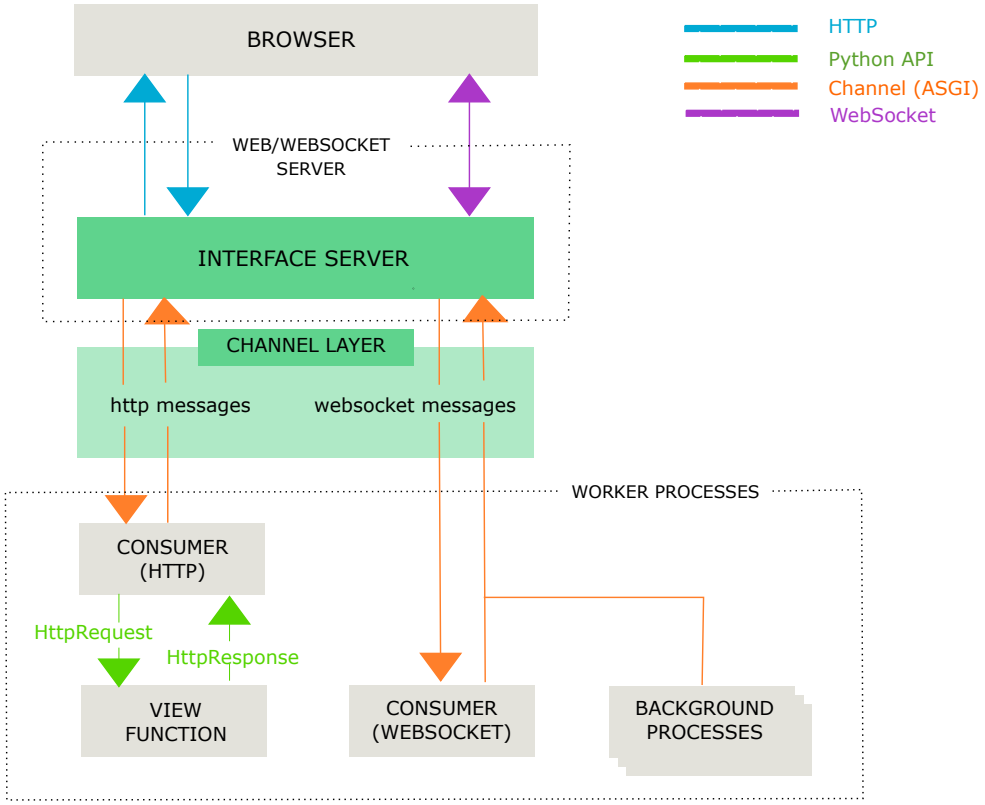
Protokol *WebSocket*, popísaný v špecifikácií RFC 6455[8], umožňuje obojsmernú komunikáciu medzi klientom a vzdialeným hostiteľom. Technológia *WebSocket* umožňuje nepretržité spojenie a výmenu dát medzi serverom a klientom.

[6] Oficiálna stránka Heroku – https://www.heroku.com/
[7] Popis a inštalácia psycopg2 – https://pypi.org/project/psycopg2/
[8] https://tools.ietf.org/html/rfc6455

### WebSockets na strane serveru

*Django* je postavené na jednoduchom koncepte požiadavkov a odpovedí, kde prehliadač zadá požiadavku [9], *Django* zavolá *View*, ktoré vráti odpoveď a tá je následne odoslaná späť do prehliadača. S technológiou *WebSocket* to ale takto jednoduché nie je. *View* je k dispozícii iba na dobu životnosti jednej žiadosti a neexistuje mechanizmus, ktorý by udržal otvorené pripojenie alebo odosielal dáta klientovi bez súvisiacej žiadosti. Pre použitie *WebSocketov* je nutná inštalácia *Django Channels*.

*Django Channels* nahrádzajú cyklus požiadaviek a odpovedí správami, ktoré sa posielajú cez kanály. Kanály umožňujú, aby úlohy bežali na pozadí, ktoré bežia na rovnakých serveroch ako zvyšok *Django*. HTTP žiadosti sa správajú rovnako ako pred inštaláciou *Django Channels*, ale teraz ich smerovanie prebieha tiež cez kanály.




Obr. 4.2: **Grafické znázornenie princípu Django Channels** [9]. Správy sa dostanú do kanálu od *producers*. Následne sa posunú ďalej jednému z *consumers*, ktorí počúvajú na danom kanáli. *Django Channels* fungujú naprieč celou sieťou a vďaka tomu môžu *consumers* bežať transparentne na viacerých zariadeniach.

### WebSockets na strane klienta

Do HTML šablóny je nutné napísať kód v jazyku *JavaScript* [1]. Otvorenie *WebSocket* spojenia vznikne vytvorením `new WebSocket`, použitím špeciálneho protokolu *ws* v URL – `let socket = new WebSocket("ws://javascript.info");`. Preferuje sa však použitie `wss://`, pretože je šifrovaný a spoľahlivejší. Údaje v `ws://` šifrované nie sú a tým pádom sú viditeľné pre všetkých sprostredkovateľov. Staré proxy servery nepoznajú *WebSockets*

a preto môžu vidieť „zvlášte“ hlavičky, čo má za následok prerušenie pripojenia. Na druhej strane `wss://` je *WebSocket* cez TLS, rovnako ako HTTPS je HTTP cez TLS. Transportná bezpečnostná vrstva šifruje dáta u odosielateľa a dešifruje u prijímateľa. Datové pakety sa tak prenášajú šifrované prostredníctvom proxy serverov, ktoré nemôžu vidieť čo je vo vnútri a nechajú ich prejsť ďalej.

Komunikácia *WebSocket* pozostáva z „rámcov“ – dátových fragmentov, ktoré je možné posielať z ktorejkoľvek strany a môžu mať rôzne podoby:

- **Textové rámce** – obsahujú textové údaje,
- **Binárne datové rámce** – obsahujú binárne údaje,
- **Ping/Pong rámce** – používajú sa na kontrolu spojenia,
- ***Connection close* rámce**.

Priamo vo webovom prehliadači sa pracuje len s textovými alebo binárnymi rámcami. *WebSocket* metóda `.send` umožňuje tieto rámce odosielať. Prostredníctvom metódy `.onmessage` môže prehliadač obdržať dáta od serveru. Text je vždy v podobe textového reťazca (`string`) a binárne dáta majú podobu `Blob` alebo `ArrayBuffer`. Závisí to od nastavenia vlastnosti `socket.binaryType` (v predvolenom nastavení je `Blob`).

Akonáhle chce jedna zo strán ukončiť vzájomné spojenie, pošle rámec *connection close* a numerickým kódom a textovým zdôvodnením – `socket.close([code], [reason])`.

## 4.5 AJAX

AJAX [8] je kolekcia technológií, ktoré existovali už niekoľko rokov pred jeho vznikom. Jednotlivé technológie boli vyvinuté z rôznych dôvodov. Webový vývojári prišli postupne na to, že ich používanie spoločne poskytuje výkonné prostredie pre tvorbu a beh webových aplikácií. Medzi tieto technológie, ktoré tvoria AJAX patrí:

- **JavaScript**[9] – programovací jazyk používaný na vývoj AJAX aplikácií, ktorý prepája interakcie všetkých ostatných AJAX technológií,
- **XML** – poskytuje prostriedky na výmenu štruktúrovaných údajov medzi webovým serverom a klientom,
- **XMLHttpRequest** – poskytuje možnosť asynchrónnej výmeny údajov medzi webovými prehliadačmi a webovým serverom,
- **HTML a CSS** – umožňuje označiť a upraviť štýl zobrazenia textu webovej stránky,
- **Document Object Model (DOM)** – poskytuje schopnosť dynamicky interagovať s rozloženým a obsahom stránky, prípadne ho meniť.

Pred niekoľkými rokmi webové aplikácie zaostávali za dektopovými aplikáciami vzhľadom na ich vzhľad a problémy s internetovým pripojením. Aj vďaka technológiám AJAX dokážu dnešné webové aplikácie poskytnúť vysoký výkon a vývojári môžu vytvárať aplikácie, ktoré sú schopné dynamicky interagovať s používateľmi a pracovať na pozadí s webovými servermi. Údaje načítané zo serveru na zobrazenie nepotrebujú obnovu stránky. Webové

[9] Oficiálna stránka jazyka JavaScript – https://www.javascript.com/

servery negenerujú a nevracajú celé webové stránky ako odpoveď na každú žiadosť používateľa. Namiesto toho sa prostredníctvom asynchrónnej komunikácie vymieňa iba malé množstvo dát. AJAX aplikácie tak môžu posielať požiadavky na server bez nutnosti pozastaviť vykonávanie aplikácie. Údaje sa neposielajú na server pomocou formulára, ale ako objekt *XMLHttpRequest*, ktorá zaručuje asynchrónnu komunikáciu.

## 4.6 Bootstrap 4

*Bootstrap* [14] je framework, ktorý sa používa na tvorbu responzívnych webových stránok. Pozostáva zo širokej škály nástrojov, ktoré sa spoliehajú na opakovanie použiteľný kód, aby vývojári nemuseli od nuly vyvíjať jednotlivé stavebné prvky webových stránok, čo urýchľuje tento proces. Zabezpečuje konzistentnosť a nedochádza tak k problémom medzi jednotlivými prehliadačmi. Vo veľkej miere využíva *jQuery*, bez ktorého by kompatibilita medzi prehliadačmi nebola možná a *JavaScript* by bol príliš komplikovaný.

*Bootstrap* je jednoduchý na naučenie a jeho používanie je pohodlné a dokáže ušetriť množstvo času. Jednou z hlavných výhod je aj to že má v sebe zabudovaný *grid* systém. Vývojár si tak nastaví iba svoje vlastné zarážky pre každý stĺpec podľa svojich potrieb, prípadne sa môže držať už predvolených nastavení. Tvorba responzívnych stránok je tak omnoho jednoduchšia.

Tvorbu responzívnych stránok často doprevádza problém, ktorý vzniká zmenou veľkosti obrázkov. *Bootstrap* obsahuje vlastný kód na automatickú zmenu ich veľkosti pomocou vopred určených pravidiel CSS a novej triedy. Obsahuje tiež prednastavené komponenty ako rozbaľovacie ponuky či navigačné lišty.

# Kapitola 5

# Návrh aplikácie *Organize Semaphore*

Pred samotným návrhom aplikácie bolo nutné naštudovať princípy vývoja webových aplikácií, popísané v kapitole 3 a tiež sa zoznámiť s jednotlivými technológiami, ktoré boli využité pri vývoji aplikácie *Organize Semaphore* (kap. 4). Táto kapitola zahŕňa návrh fungovania webového serveru, spôsob komunikácie, návrh databázy a používateľského rozhrania.

## 5.1 Princíp aplikácie a spôsob komunikácie v aplikácii

Používatelia, ktorí si chcú vytvoriť svoj „semafor“ = poradovník, prostredníctvom ktorého budú vyvolávať zapísaných ľudí v poradí, sa musia najskôr zaregistrovať. Pre registráciu je potrebné vyplniť používateľské meno, e-mail a heslo. Výhodou použitia frameworku *Django* je, že heslo vytvorené používateľom sa automaticky zašifruje. V takejto podobe sa uloží do databázy. *Django* má v sebe zabudované používateľské rozhranie pre admin panel. Superpoužívateľ v ňom má možnosť spravovať dáta uložené v databáze (v rámci aplikácie *Organize Semaphore* ide o používateľov aplikácie, vytvorené semafory a ľudí čakajúcich v poradí). Ani superpoužívateľ nevidí v admin paneli heslo používateľa tak ako ho zadal pri registrácii, ale iba jeho zašifrovanú verziu.

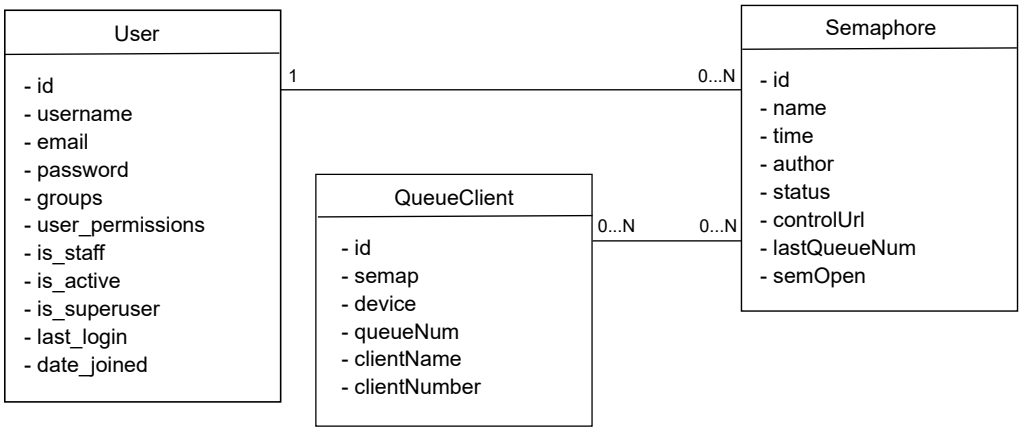
Po vytvorení nového semaforu sa vygeneruje URL, ktoré je dostupné na skopírovanie na správcovskej stránke každého semaforu. Na tejto URL adrese sa môžu ľudia prihlásiť do poradovníka daného semaforu. Ihneď po otvorení stránky, kde sa môže klient prihlásiť do poradia, sa klientovi priradí cookie device. Vytvorený semafor má ukončené prihlasovanie do poradia. Po otvorení prihlasovania sa pošle správa pomocou technológie *WebSocket*. Na základe obdržanej správy sa všetkým klientom daného semaforu otvorí prihlasovanie do poradia. Keď sa do poradia zapíšu, pomocou technológie AJAX sa na pozadí zavolá metóda z `view.py`, ktorá uloží čakajúce do databázy. Poradie čakajúceho je tak viazané na cookie device a číslo, ktoré je priradené každému zapísanému klientovi v poradí.

Proces vyvolávania ľudí v tom poradí, ako sa zapísali do poradovníka semaforu, spočíva v priradení skrytého čísla. Pri vyvolaní ďalšieho klienta sa sa vezme číslo posledného vyvolaného klienta a nájde sa v databáze najbližšie vyššie číslo klienta, ktorý je zapísaný do poradovníka daného semaforu. Poradovník konkrétneho semaforu tak tvoria všetci klienti v databáze, ktorých skryté číslo je väčšie, ako číslo posledného vyvolaného a zároveň patria k danému semaforu.

Akonáhle správca semaforu vyvolá ďalšieho klienta, pošle *WebSocket* správu s informáciou, že práve niekoho vyvolal. Všetci klienti čakajúci v poradí tohto semaforu po obdržaní správy skontrolujú svoju pozíciu a aktualizujú svoje údaje o poradí.

## 5.2 Návrh databázy

Pre fungovanie aplikácie *Organize Semaphore* je nutné uchovávať určité dáta v databáze. Vytvorená *PostgreSQL* databáza (kap. 4.3) využíva AWS RDS (*Amazon Web Services Relation Database Services*)[1]. Amazon RDS uľahčuje nastavenie, prevádzku a škálovanie nasadenia *PostgreSQL* v cloude. Spravuje zložité a časovo náročné administratívne úlohy, ako je inštalácia a upgrady softvéru *PostgreSQL*, správa úložiska, replikácia pre vysokú dostupnosť a priepustnosť čítania, a zálohy na zotavenie po katastrofe. Samotná databáza aplikácie je tvorená 3 tabuľkami: `User`, `Semaphore` a `QueueClient`. Ich grafické znázornenie je možné vidieť na obrázku 5.1.




Obr. 5.1: **Grafické znázornenie databázových tabuliek**. Používateľ môže vytvoriť 0 až N „semaforov“. Vytvorený „semafor“ prináleží však len jednému užívateľovi a to autorovi, ktorý ho vytvoril. Iba on ho môže spravovať. Do poradia daného „semaforu“ sa môže prihlásiť 0 až N klientov. Klienti sa tiež môžu prihlásiť do 0 až N „semaforov“.

### Popis tabuľky `User`

Používatelia, ktorí chcú riadiť poradie, sa musia zaregistrovať. *Django* má v sebe zabudovaný autentifikačný systém[2] s predvolenými nastaveniami, ktoré sú prispôsobené najbežnejším potrebám. Autentifikačný systém poskytuje autentifikáciu aj autorizáciu spoločne, nakoľko ich funkcie sú trochu prepojené.

Jadro autentifikačného systému tvoria objekty `User`. Zvyčajne zastupujú ľudí, ktorí interagujú s aplikáciou. Používajú sa tiež na obmedzenie prístupu či registráciu profilov. V autentifikačnom systéme frameworku *Django* existuje trieda `Superuser`. `Superuser` (tiež administrátor) predstavujú `User` objekty s nastavenými špeciálnymi atribútmi. Takýto používateľ má prístup do grafického používateľského rozhrania admin panelu.

[1] Oficiálna stránka AWS – https://aws.amazon.com/rds/postgresql/
[2] https://docs.djangoproject.com/en/3.2/topics/auth/default/#user-objects

Medzi atribúty[3] prednastaveného objetku `User` patria:

- `id` – primárny kľúč.
- `username` – používateľské meno, ktoré zadáva používateľ pri registrácií a prihlasovaní do aplikácie. Môže pozostávať z alfanumerických znakov a symbolov: `_`, `@`, `+`, `.`, `-`.
- `email` – emailová adresa používateľa, ktorý ju zadáva pri registrácii.
- `password` – povinný atribút, ktorého hodnotu používateľ zadá pri registrácii a naďalej používa pri prihlasovaní do aplikácie. *Django* neukladá heslo tak ako ho používateľ zadal. Ukladá jeho zahashovanú podobu.
- `first_name` – voliteľný atribút, ktorého hodnotou je krstné meno používateľa. Pre funkčnosť aplikácie *Organize Semaphore* nie je potrebný.
- `last_name` – hodnotou atribútu je priezvisko používateľa. Je to voliteľný atribút a pre fungovanie aplikácie *Organize Semaphore* nie je potrebný.
- `groups` – hodnotou atribútu bývajú názvy skupín, ktorých je používateľ účastníkom.
- `user_permissions` – obsahuje povolenia používateľa,
- `is_staff` – atribút, ktorého hodnota je typu *boolean*. Poukazuje na to, či má používateľ prístup do admin panelu.
- `is_active` – hodnota atribútu je typu *boolean* a určuje, či sa daný používateľský účet považuje za aktívny. Často sa pri odstraňovaní účtov doporučuje namiesto fyzického odstránenia z databázy iba nastavenie tohto atribútu na `False`. Ak aplikácia obsahuje cudzie kľúče, ktoré sa viažu na používateľov, mali by tak ostať neporušené.
- `is_superuser` – hodnota atribútu je typu *boolean*. V prípade, že je nastavená na `True`, používateľ vlastní všetky povolenia.
- `last_login` – časový údaj, ktorý označuje posledné prihlásenie používateľa do aplikácie.
- `date_joined` – predstavuje dátum a čas, kedy bol vytvorený používateľský profil.

### Popis tabuľky `Semaphore`

Každý zaregistrovaný používateľ môže vytvoriť „semafor“, pomocou ktorého bude riadiť poradie. O tomto „semafore“ je potrebné uchovávať v databáze tieto informácie:

- `id` – primárny kľúč tabuľky. *Django* ho vytvára automaticky.
- `name` – názov „semaforu“, ktorý používateľ zadá pri jeho vytvorení.
- `time` – predpokladaný počet minút na jedného klienta. Tento časový údaj zadáva používateľ taktiež pri jeho vytvorení.
- `author` – meno používaľa, ktorý „semafor“ vytvoril. Atribút sa vyplní pri vytvorení „semaforu“ na základe toho, ktorý používateľ je momentálne prihlásený..

[3] https://docs.djangoproject.com/en/3.2/ref/contrib/auth/

- `status` – pokiaľ je hodnota atribútu `Ready`, na „semafore“ svieti zelená farba, čo znamená že bol zavolaný ďalší klinet. Hodnota atribútu sa zmení na `Busy` v prípade, že používateľ môže mať síce otvorené prihlasovanie do poradia, ale momentálne má prestávku a nevolá ďalších klientov ,a tak všetkým čakajúcim v poradí svieti červená farba.
- `controlUrl` – každému „semaforu“ sa vygeneruje reťazec znakov, ktorý je použitý v URL stránky, kde sa návštevníci môžu zapísať do poradia daného „semaforu“.
- `lastQueueNum` – skryté číslo posledného zavolaného klienta v danom „semafore“,
- `semOpen` – po vytvorení „semaforu“ je hodnota atribútu `semOpen` nastavená na *False*. To znamená, že prihlasovanie do poradia je uzatvorené. Správca „semaforu“ môže prihlasovanie otvoriť pomocou tlačítka. Hodnota atribútu sa následne zmení na *True*.

### Popis tabuľky `QueueClient`

Po klientoch sa nevyžaduje žiadna registrácia. Počet klientov v poradí môžu skontrolovať kedykoľvek. Akonáhle sa zapíšu do poradia, do databázy sa o danom klientovi uloží niekoľko informácií:

- `id` – primárny kľúč tabuľky, ktorý sa automaticky inkrementuje. Za jeho vytvorenie je zodpovedné *Django*.
- `semap` – cudzí kľúč, pomocou ktorého je prepojený klient zapísaný do poradia s daným „semaforom“..
- `device` – jedinečný textový reťazec získaný zo súboru cookie.
- `queueNum` – skryté číslo klienta. Pri zápise do poradia sa každému klientovi priradí číslo, pomocou ktorého sa zisťuje umiestnenie v poradovníku..
- `clientName` – meno klienta, ktoré vyplní pri zápise do poradia daného „semaforu“.
- `clientNumber` – číslo, ktoré určuje koľko ľudí sa nachádza pred daným klientom.

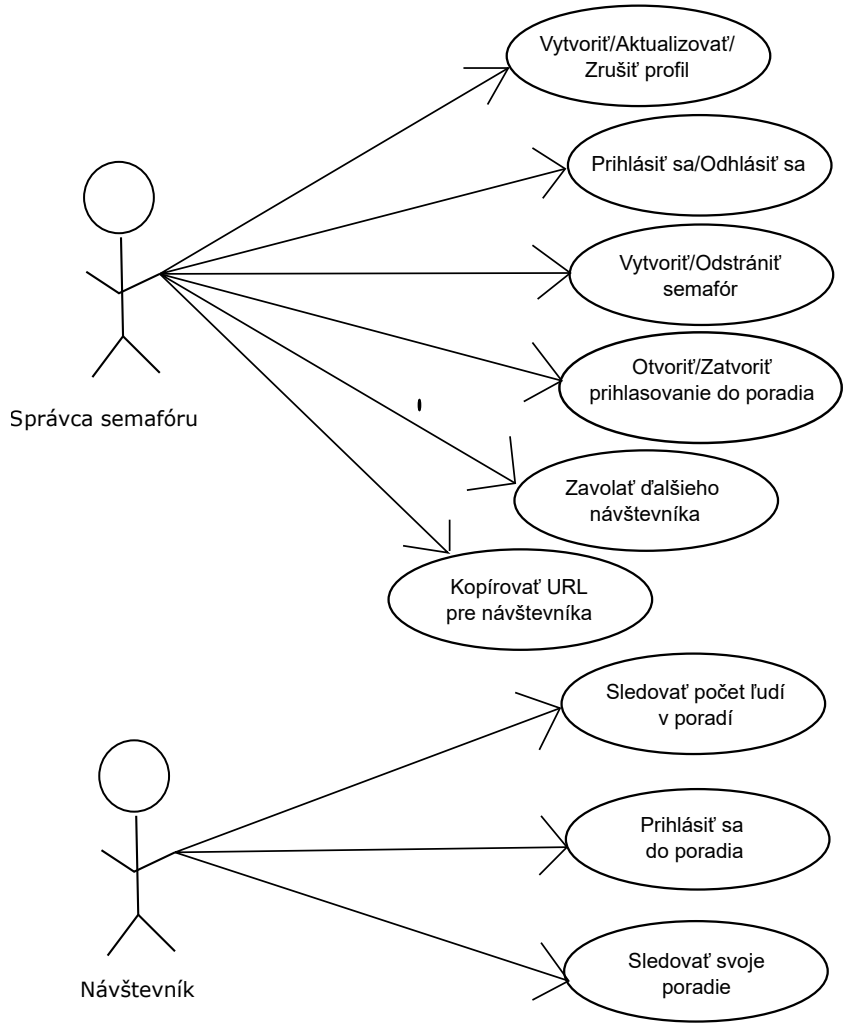
## 5.3 Návrh používateľského rozhrania

Používateľov aplikácie *Organize Semaphore* môžeme rozdeliť do dvoch skupín. Prvú skupinu tvoria používatelia, ktorí si vytvoria „semafor“ a riadia poradie. Druhá skupina používateľov sú tí, ktorí sa do poradia prihlásia a sledujú svoju pozíciu v ňom. Grafické znázornenie prípadov použitie je ukázané na obrázku 5.2

### Konkrétne príklady použitia aplikácie:

- Študent vysokej školy má pol hodinu pauzu medzi prednáškami. Potrebuje ísť na študijné oddelenie, aby si tam vyzdvihol novú revalidačnú známku na ISIC a potvrdenie o návšteve školy. Je začiatok semestra, to znamená že pravdepodobne bude na študijnom oddelení zvýšený počet študentov, tak má obavy, aby si to stihol vybaviť počas prestávky. Študent bez toho, aby musel osobne ísť na študijné oddelenie, kde by videl koľko ľudí stojí pred dverami, si môže situáciu skontrolovať prostredníctvom svojho počítača alebo mobilného telefónu online a prípadne sa zaradiť do poradia.

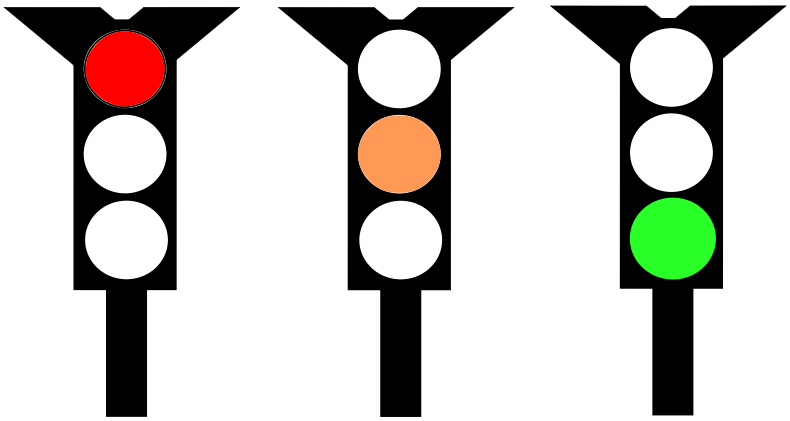
- Niektorí lekári neobjednávajú pacientov na presne daný čas ale iba deň. Ak by lekár používal aplikáciu *Organize Semaphore*, tak pacienti majú prehľad o tom, koľko ľudí sa momentálne nachádza v čakárni. Na základe toho sa môžu rozhodnúť kedy za lekárom vycestujú a môžu sa do poradia prihlásiť už počas cesty. Takýmto spôsobom si výrazne skrátia čakaciu dobu a lekár má k dispozícii okamžite informáciu o tom, že sa nový pacient nachádza v poradovníku. Podobné využitie môže byť aj na úradoch pri výdaji dokladov či na pošte, alebo kdekoľvek je možné využiť vyvolávací systém.
- Študent pracuje na projekte a potreboval by ísť za učiteľom, ktorý projekt zadal a skonzultovať s ním nejaký problém. Učiteľ, ktorý aplikáciu využíva aplikáciu, môže otvoriť prihlasovanie do poradia keď má konzultačné hodiny alebo voľný čas, počas ktorého by sa mohol venovať študentom. Ak nejaký študent jeho pomoc potrebuje, postačí mu tak skontrolovať či je možné sa prihlásiť do jeho poradovníka, teda či má učiteľ momentálne čas. Nemusí tak cestovať do školy, hľadať jeho kanceláriu alebo mu písať mail. Postačí mu skontrolovať jeho „semafor“.




Obr. 5.2: **Diagram prípadov použitia aplikácie *Organize Semaphore*.**

Počiatočný návrh v sebe zahŕňal prvky a funkcie, ktoré som sa nakoniec rozhodla vynechať. Mojím cieľom od začiatku vytvoriť veľmi jednoduchú aplikácie, pri ktorej nebude používateľ musieť premýšľať nad tým, ako sa používa. Na základe toho vznikol aj nápad s použitím dopravného semaforu, ktorý každý pozná z bežného života. Výsledný obrázok semaforu, použitý aj priamo v aplikácii, je možné vidieť na obrázku 5.3.

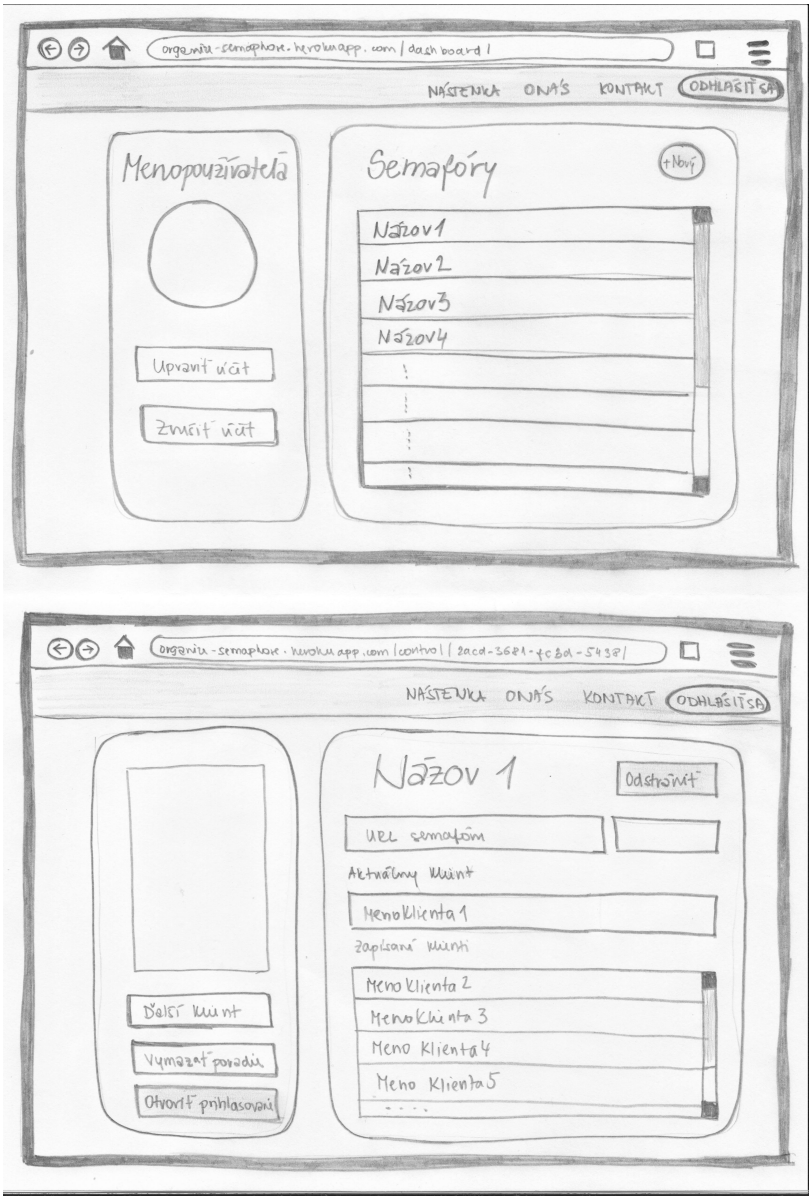
Pôvodný návrh obsahoval iba dve farby dopravného semaforu: červenú a zelenú. Nápad ako zapojiť aj oranžovú farbu prišiel od niekoľkých nezávislých používateľov, ktorý sa podieľali na testovaní aplikácie. Častokrát bola kladená otázka, či semafor bude niekedy svietiť aj na oranžovo. Nakoľko sa táto otázka od väčšiny používateľov bola medzi prvými, rozhodla som sa ju nakoniec pridať do aplikácie.



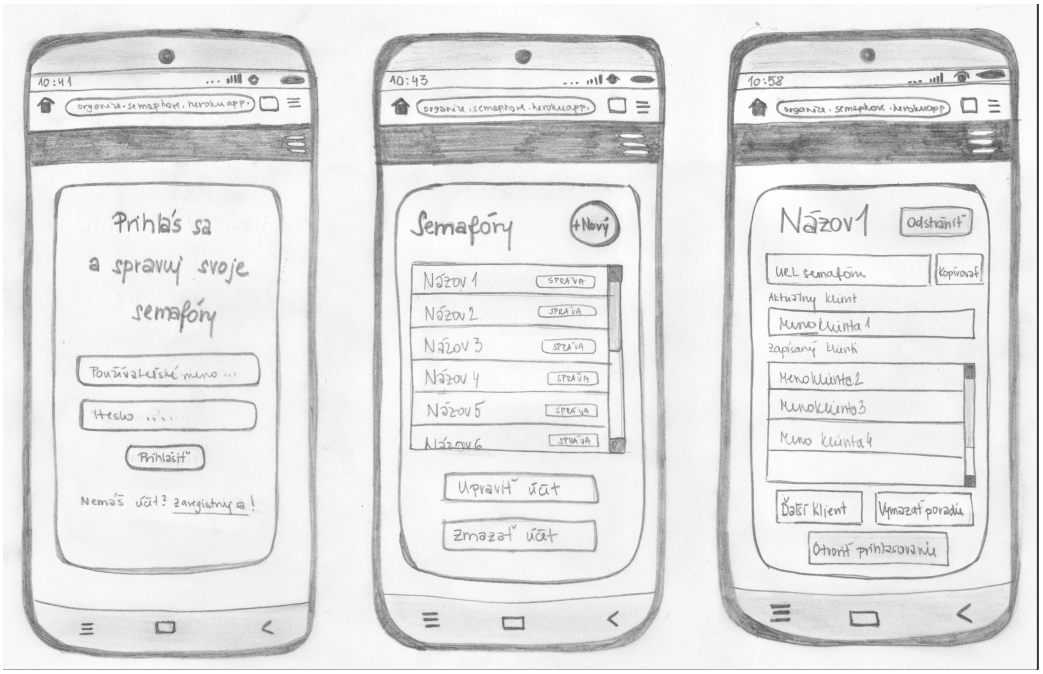
Obr. 5.3: **Obrázok dopravného semaforu použitého v aplikácii *Organize Semaphore***. Keď sa klientovi zobrazí stránka, na ktorej môže sledovať aktuálny počet čakajúcich ľudí, dopravný semafor bude svietiť na červeno. Pokiaľ je prihlasovanie do poradia otvorené, môže sa zapísať do poradia vložením svojho mena. Následne po zapísaní sa klientovi zobrazí na semafore farbe podľa toho, koľko ľudí sa nachádza pred ním. Ak je počet čakajúcich pred ním väčší ako 0, klient „má červenú“ a musí tak čakať. Vo chvíli, keď pred ním v poradí už nikto nebude, rozsvieti sa oranžová farba. Klient tak vie, že je čas pripraviť sa, o chvíľu bude vyvolaný. Keď táto skutočnosť nastane, objaví sa na semafore zelená farba.

Navigačný panel aplikácie sa líši podľa toho, či je používateľ prihlásený alebo nie. Neprihlásený používateľ v navigácii vidí odkaz na hlavnú stránku, na stránku *O Nás*, kde sa nachádza vysvetlenie fungovania aplikácie a záložku *Kontakt.* V tejto sekcii používateľ nájde kontaktné údaje. Prihlásený používateľ odkaz na hlavnú stránku už nevidí. Namiesto toho tam má záložku na svoju nástenku a tlačítko na odhlásenie sa.

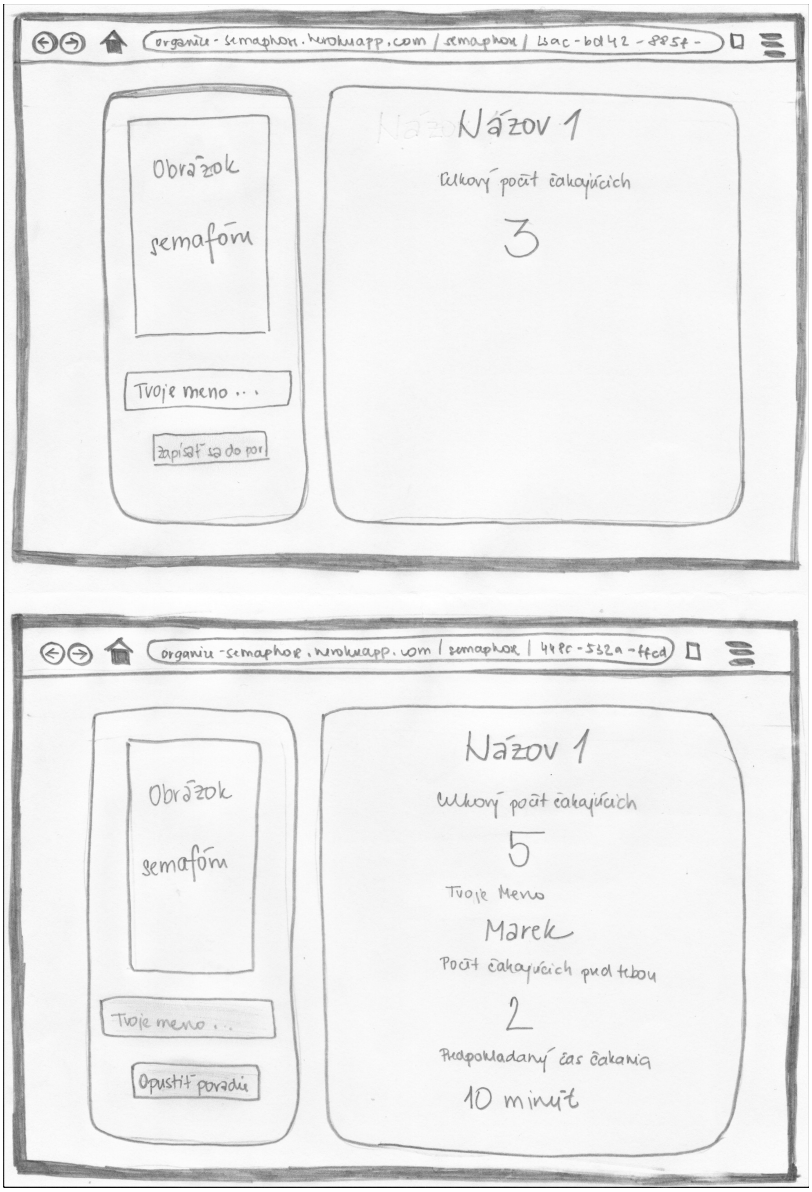
Návrhy používateľského rozhrania webovej aplikácie boli vytvorené pre obrazovku notebooku či monitoru, ale taktiež pre mobilné zariadenia. Nie všetci ľudia vlastnia počítač, ale mobil s prístupom na internet má dnes už väčšina populácie, a je možné ho so sebou vziať takmer kamkoľvek. Vytvorené návrhy mi poslúžili na získanie prvej spätnej väzby od potenciálnych budúcich používateľov a tiež pri samotnom vývoji aplikácie. Nemusela som tak už premýšľať nad dispozíciou jednotlivých prvkov počas programovania. Výsledná aplikácia nie je identická s návrhmi. Líši sa drobnými rozdielmi, ale podstata ostala rovnaká.



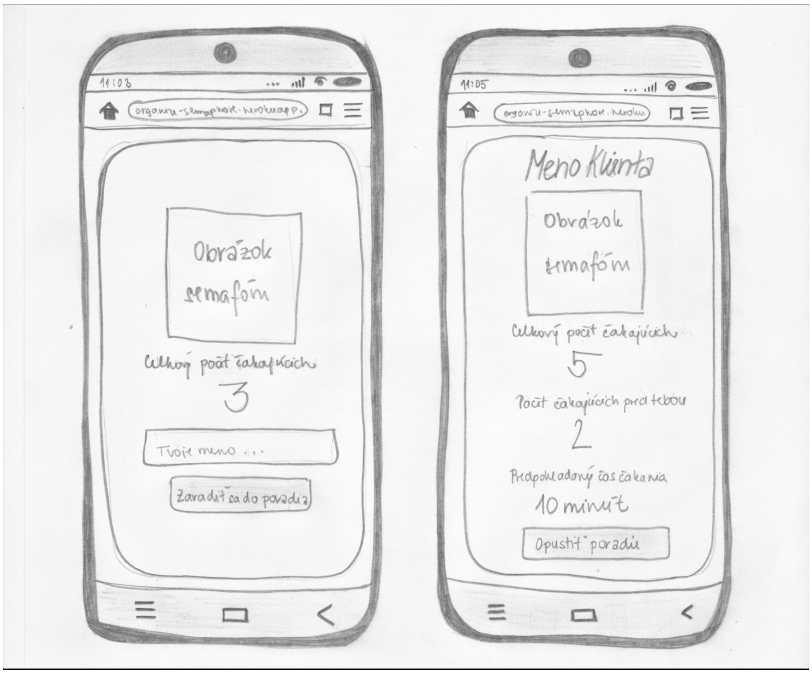

Obr. 5.4: **Webová verzia aplikácie z pohľadu správcu semaforu.** Prvá časť obrázku ponúka náhľad na nástenku, kde používateľ vidí zoznam svojich „semaforov“, ktorá tiež zahŕňa profilovú časť. V nej má používateľ možnosť upraviť alebo zrušiť svoj účet. Po kliknutí na jeden zo semaforov sa dostane na spravovaciu stránku, ktorú je možné vidieť v druhej časti obrázku. Tá zahŕňa všetky interakcie, ktoré môže nad semaforom vykonávať: zavolať ďalšieho klienta, vymazať poradovník, otvoriť/ukončiť prihlasovanie do poradia. Nachádza sa tu aj URL adresa, ktorú môže skopírovať pomocou tlačítka do clipboardu.




Obr. 5.5: **Mobilná verzia aplikácie z pohľadu správcu semaforu.** Rozdiel medzi webovou a mobilnou verziu spočíva v zobrazení informácií. V mobilnej verzii sa už nenachádza samostatná profilová časť. Tlačítka na úpravu a odstránenie účtu boli presunuté pod zoznam semaforov, aby sa všetky funkcie aplikácie zmestili na malú obrazovku. Rovnako aj na stránke spravovania semaforu zmizol obrázok semaforu a tlačítka na riadenie poradia boli presunuté pod zoznam zapísaných klientov.




Obr. 5.6: **Webová verzia aplikácie z pohľadu návštevníka.** Obrázok zahŕňa dve situácie z pohľadu návštevníka. Po vstupe na stránku vidí názov semaforu a aktuálny celkový počet čakajúcich. V ľavej časti je umiestnený obrázok semaforu a priestor na vloženie svojho mena s tlačítkom, pomocou ktorého sa zapíše do poradia. Po zápise sa v pravej časti zobrazia všetky informácie o jeho poradí.




Obr. 5.7: **Mobilná verzia aplikácie z pohľadu návštevníka.** Oproti webovej verzii sú v mobilnej verzii umiestnené všetky informácie v rámci jedného okna. Výsledok bol dosiahnutý zmenšením obrázku dopravného semaforu a tiež veľkosti písma. Informácie však naďalej ostávajú dostatočne čitateľné a prehladné.

# Kapitola 6

# Implementácia jednotlivých častí aplikácie a testovanie

Kapitola obsahuje štruktúru aplikácie *Organize Semaphore* a popis implementácie jednotlivých častí. Nadväzuje na informácie popísané v kapitole 5 zaoberajúcej sa návrhom aplikácie.

## 6.1 Adresárová štruktúra aplikácie *Organize Semaphore*

Adresárová štruktúra aplikácie vychádza zo štruktúry definovanej frameworkom *Django* (kap. 4.1.):

- adresár `core` – obsahuje konfiguráciu projektu. Súčasťou adresára sú súbory: `__init__.py`, `asgi.py`, `urls.py`, `routing.py` a `settings.py`.
- adresár `mysite` – zahŕňa súbory potrebné na fungovanie webovej aplikácie *Organize Semaphore*. Obsahuje dva podadresáre. Podadresár `migrations` obsahuje migrácie databázy a podadresár `templates` obsahuje HTML súbory aplikácie. Okrem podadresárov sa tu nachádzajú aj tieto súbory: `__init__.py`, `admin.py`, `upps.py`, `consumers.py`, `forms.py`, `model.py`, `routing.py`, `tests.py`, `urls.py` a `views.py`.
- adresár `static` – obsahuje CSS súbory a v podadresári `images` sú umiestnené obrázky dopravného semafora.
- súbory potrebné pre službu *Heroku* – súbor `Procfile` definuje spôsob spustenia aplikácie a súbor `requirements.txt` obsahuje názvy a verzie všetkých modulov potrebných, ktoré sa pri nasadení aplikácie na heroku nainštalujú a sú potrebné pre chod aplikácie.
- súbor `manage.py` – skript vygenerovaný frameworkom *Django* určený na vytvorenie aplikácie, migráciu databázy a spustenie aplikácie.,

## 6.2 Uchovávané dáta v databáze

*PostgreSQL* databázu je nutné najskôr nakonfigurovať[1] v súbore `settings.py` (obr. 6.1). O každom zaregistrovanom používateľovi sa v databáze uchovávajú určité dáta. Model `User`, ktorý je súčasťou modulu `django.contrib.auth`, predstavuje zaregistrovaného používateľa.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'semapdb',
        'USER': '[redacted]',
        'PASSWORD': '[redacted]',
        'HOST': 'database-1.cwzmbitogqlu.us-east-2.rds.amazonaws.com',
        'PORT': '5432',
    }
}
```

Obr. 6.1: **Fragment kódu prezentujúci konfiguráciu *PostgreSQL* databázy.** Používateľské meno a heslo k databáze *Organize Semaphore* sú na obrázku zakryté. Príklad používateľského mena môže byť `'mydatabaseuser'` a heslo `'mypassword'`.

Po umiestnení aplikácie na *Heroku* bolo nutné konfiguráciu databázy pozmeniť. V prvom rade je potrebné manuálne nastaviť premennú `DATABASE_URL` na *Heroku*. V záložke *Settings*, v oddelení *Config Vars*, som odstránila už existujúcu premennú a vytvorila novú. Hodnota novej premennej `DATABASE_URL` má podobu `postgres://USERNAME:PASSWORD@database url endpoint:PORT/DB NAME`. Aplikácia využíva knižnicu[2], ktorú je možné nainštalovať príkazom `pip install dj-database-url`. Knižnica nám dovoľuje nastaviť konfiguráciu v súbore setting.py tak, že kedykoľvek nasadíme na *Heroku* novú verziu aplikácie, dovolí nám hľadať premenné na *Heroku*. To znamená, že keď aplikáciu spustíme lokálne na svojom počítači, bude využitá *Djangom* prednastavená *sqlite* databáza, ale pri otvorení aplikácie umiestnenej na *Heroku*, bude aplikácie pristupovať k vytvorenej *PostgreSQL* databáze.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

db_from_env = dj_database_url.config(conn_max_age=600)
DATABASES['default'].update(db_from_env)
```

Obr. 6.2: **Fragment kódu prezentujúci konfiguráciu *PostgreSQL* databázy po umiestnení aplikácie na *Heroku*.** Aby bol kód funkčný, vyžaduje pridať na začiatok súboru riadok `import dj_database_url`.

V súbore `models.py` sú definované tabuľky databázy aplikácie pomocou tried. Aplikácia pre svoje fungovanie využíva 3 triedy: `User`, `Semaphore` a `QueueClient`. Trieda `User` ako je-

[1] https://docs.djangoproject.com/en/3.1/ref/settings/#databases
[2] https://pypi.org/project/dj-database-url/

diná nie je priamo vytvorená v `models.py` ale naimportovaná z `django.contrib.auth.models`. Ostatné triedy modelu je nutné zaregistrovať v súbore `admin.py`. Najskôr sa naimportujú modely a následne príkazom `admin.site.register(`*`nazov_triedy`*`)` sa zaregistrujú.

## 6.3 Registrácia a prihlásenie používateľov

*Django* má v sebe zabudovaný prednastavený formulár na vytvorenie nového používateľa, ktorý stačí naimportovať `UserCreationForm` do `views.py` z `django.contrib.auth.forms`. Prednastavený formulár zahŕňa používateľské meno a heslo. Výhodu použitia už zabudovaného formuláru je automatické zahašovanie hesla, o ktoré sa postará *Django*, a tiež systém kontroly sily hesla zvoleného používateľom. *Django* overuje či zvolené heslo nie je podobné ostatným zvoleným osobným informáciám, pozostáva minimálne z 8 znakov, ktoré zároveň nie sú iba číselné, a že zvolené heslo nepatrí medzi najbežnejšie používané heslá obecne. Tento formulár je možné upravovať podľa potrieb. Aplikácia *Organize Semaphore* vyžaduje po svojich používateľoch zadanie e-mailovej adresy pri registrácii a preto bolo nutné v súbore `form.py` vytvoriť triedu, ktorá dedí funkcionalitu z `UserCreationForms` (obr. 6.3).

```
class CreateUserForm(UserCreationForm):
    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']
```

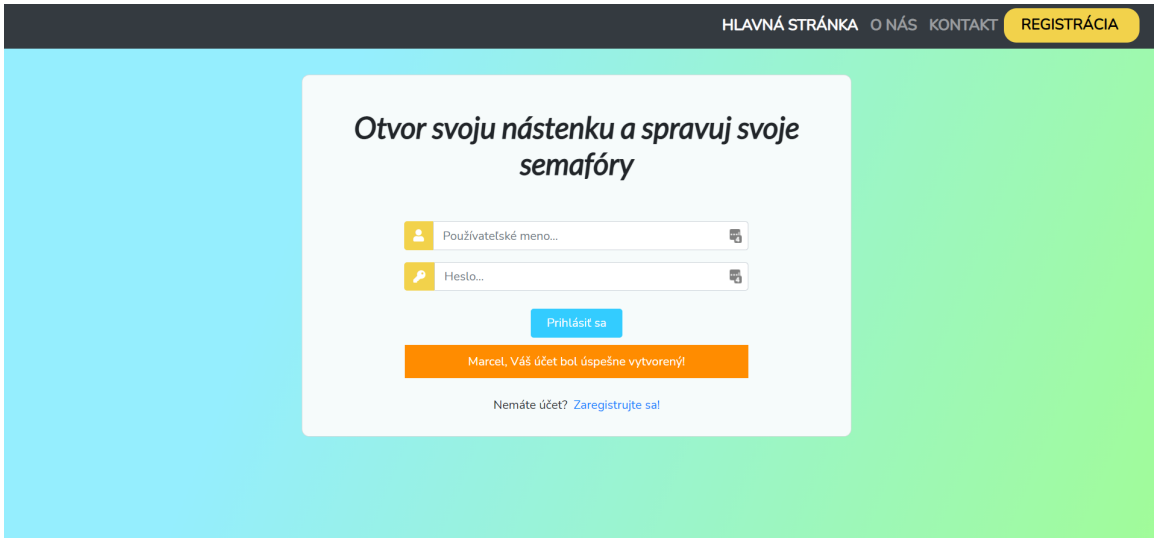
Obr. 6.3: **Fragment kódu prezentujúci triedu `CreateUserForm`**. Táto trieda vyžaduje importovanie triedy `UserCreationForms`, od ktorej dedí funkcionalitu, model `User` a *Django* formuláre. V premennej `fields` sú nastavené všetky polia, ktoré musí používateľ vyplniť pri registrácii.

Na zobrazenie upraveného formulára sa tak namiesto `UserCretionForm` naimportuje vytvorená trieda `CreateUserForm`. V HTML šablóne registračnej stránky sa vloží pred inputy formulára `{%crsf_token%}`, ktorý zabezpečuje bezpečnú cestu posielania dát. Jednotlivé HTML inputy sú nahradené *Django* premennými. Napríklad namiesto inputu na vloženie používateľského mena sa použije `{{ form.username }}`. To isté platí aj pre polia `email`, `password1` a `password2`. Po úspešnej registrácii sa vytvorí flash správa, ktorá po presmerovaní na hlavnú stránku informuje používateľa, že registrácia prebehla úspešne a môže sa prihlásiť (obr. 6.4).

Počas procesu prihlasovania do aplikácie sa overuje identita prihlasujúceho. Využíva sa k tomu funkcia `authenticate`. Funkcia vracia objekt používateľa, prípadne prázdny objekt ak sa overenie nepodarí. Overený používateľ sa môže prihlásiť prostredníctvom metódy `login` a následne odhlásiť pomocou metódy `logout`.

Niektoré stránky ako nástenka, spravovacia stránka, stránka na úpravu profilových údajov či vytvorenie nového semaforu sú dostupné iba pre prihlásených používateľov. Na zabezpečenie, aby sa na tieto stránky nedostal používateľ, ktorý nie je prihlásený, bola použitý dekorátor `login_required(login_url='mainpage')`. Používatelia, ktorí sa pokúsia nejakým spôsobom dostať na tieto stránky, budú presmerovaní na hlavnú stránku, kde sa najskôr musia prihlásiť.

Na aktualizovanie profilových údajov bola vo `views.py` vytvorená metóda *editAccount*. Po vykreslení stránky sa do jednotlivých polí formulára doplnia údaje z inštancie triedy `User`




Obr. 6.4: **Ukážka flash správy po úspešnej registrácii.**

konkrétneho prihláseného používateľa. Používateľ môže zrušiť úpravu, čo ho presmeruje späť na nástenku. Môže aj vykonané zmeny uložiť, ktoré sa ihneď uložia do databázy. V súbore `views.py` sa nachádza aj metóda `deleteAccount`, ktorej účelom je odstránenie používateľa z databázy pomocou `user.delete()`.

## 6.4 Priradenie súboru cookie návštevníkom „semaforu“

Každému návštevníkovi stránky „semaforu“ sa ihneď po jej otvorení pridelí súbor cookie s názvom `device`. Obsahom súboru je náhodný vygenerovaný reťazec (obr. 6.5), ktorým je možné identifikovať každého návštevníka, ktorý sa prihlási do poradia. Tento reťazec sa spolu s ostatnými údajmi o čakajúcom uloží do databázy a je ďalej využívaný pre identifikáciu čakajúcich pri kontrole aktuálneho poradia. Úsek kódu generujúci náhodný reťazec bol prevzatý z kódu inej aplikácie[3].

```
function setDevice() {
    return 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(/[xy]/g, function (c) {
        var r = Math.random() * 16 | 0, v = c == 'x' ? r : (r & 0x3 | 0x8);
        return v.toString(16);
    });
}
```

Obr. 6.5: **Funkcia generujúca náhodný reťazec uložený ako hodnota cookie `device`.**

[3] https://github.com/divanov11/guest_user_shopping_cart/blob/master/store/templates/store/main.html

## 6.5 Komunikácia v reálnom čase medzi serverom a klientom

Aby aplikácia bola pre používateľa čo najjednoduchšia a zároveň užitočná, je potrebná komunikácia v reálnom čase medzi serverom a klientom. Najmä keď sa jedná o aplikáciu, kde používateľ sleduje svoj stav, ktorý sa mení na základe určitých informácií. Zmena by sa mala prejaviť okamžite a používateľ na to, aby zmenu videl, nemusí obnovovať stránku.

### Otvorenie a ukončenie prihlasovania do poradia

Po vytvorení nového semaforu, je takýto semafor v stave zatvorený. Otvoriť prihlasovanie do poradia daného semaforu môže správca uskutočniť jednoduchým kliknutím na tlačítko. Na pozadí prebehne prostredníctvom AJAX technológie zmena hodnoty atribútu `semOpen` z *False* v modeli `Semaphore`, na hodnotu *True* Všetkým klientom sa pošle *Websocket* správa s informáciou, že prihlasovanie bolo otvorené. Od tohto momentu sa klienti môžu zapisovať do poradia. Ukončenie prebieha na rovnakom princípe. Klientom sa po obdržaní správy s ukončením prihlasovania zamkne input, do ktorého nebudú môcť napísať svoje meno a zobrazí sa im upozornenie o ukončení prihlasovania.

### Zapísanie klienta do poradia

Po zobrazení stránky sa automaticky na pozadí skontroluje, či sa daný klient už nachádza v poradí alebo nie. Pomocou technológie AJAX je zavolaná metóda `helloQueueUrl` zo súboru `views.py` (obr. 6.6). Každému zariadeniu je pridelené cookie device s náhodným vygenerovaným reťazcom. Kontrola existencie klienta v poradovníka funguje na základe hodnoty cookie device a tiež čísla prideleného klientovi. Ak sa klient v poradovníku nenechádza, má možnosť sa do neho zapísať vložením svojho mena.

### Kontrola pozície a vyvolanie ďalšieho klienta

V prípade, že po otvorení stránky sa zistí, že návštevník už je zapísaný do poradia, pokračuje sa zisťovaním jeho aktuálnej pozície. Kontrola pozície taktiež nastáva vždy, akonáhle správca semafóru vyvolá ďalšieho klienta. Pošle sa správa všetkým klientom použitím protokolu *WebSocket*, ktorá obsahuje text `"GREEN"`. Klienti po obdržaní správy pomocou technológie AJAX zavolajú metódu `checkQueueUrl()`. Metóda vyhodnotí stav jednotlivých klientov a posiela JSON správu klientom o ich stave.

Kontrola pozície spočíva v porovnaní atribútu `lastQueueNum` tabuľky `Semaphore` a atribútu `queueNum` tabuľky `QueueClient`. Po kliknutí na tlačítko *Ďalší klient* sa hodnota atribútu `lastQueueNum` zmení na hodnotu atribútu `queueNum` nasledujúceho klienta v databáze. Nad každým klientom je zavolaná metóda `chceckQueueUrl()`. Metóda overí, či hodnota porovnávaných atribútov je rovnaká. Klientom sa odošle JSON správa `"SAME"`, ak sa hodnoty zhodujú alebo `"DIFFERENT"`, v prípade, že sa hodnoty líšia. Podľa toho, akú správu klient zo serveru obdrží, sa prostredníctvom JavaScriptu zobrazia príslušné informácie a farba semaforu (obr. 6.7). V prípade správy `"SAME"` sa klient vyhodnotí ako ten, kto je momentálne na rade. Správa `"DIFFERENT"` znamená, že ešte nie je na rade a vypočíta si tak počet ľudí čakajúcich pred ním. Všetky zmeny prebiehajú asynchrónne a nie tak potrebná obnova stránky na zobrazenie zmien.

```python
@csrf_exempt
def helloQueueUrl(request, pk_test):
    device = request.COOKIES['device']
    semap = Semaphore.objects.get(controlUrl=pk_test)

    if semap.semOpen == False:
        response = {
            'msg': "Close now"
        }
        return JsonResponse(response)

    # check if the client is in DB
    try:
        client = QueueClient.objects.get(device=device, semap=semap, queueNum__gte=semap.lastQueueNum)
        response = {
            'msg': "You are already in the queue",
        }
    except:
        response = {
            'msg': "Not in queue"
        }

    return JsonResponse(response)
```

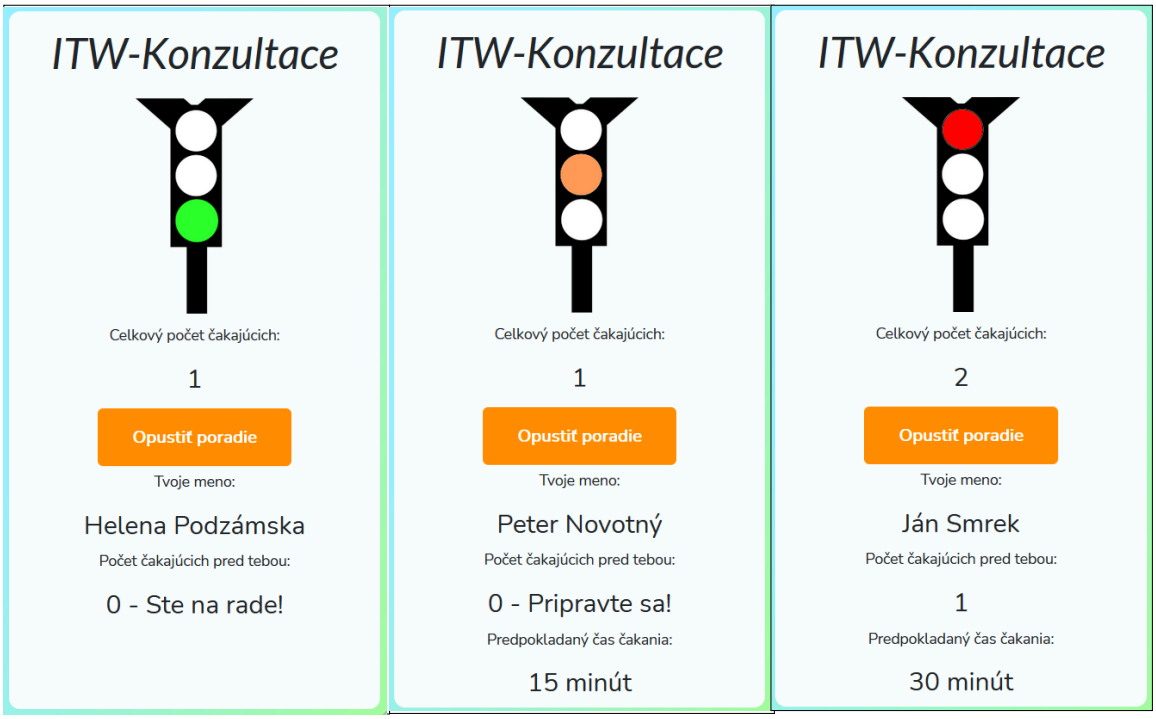
Obr. 6.6: **Ukážka kódu funkcie `helloQueueUrl`**. Funkcia si zistí hodnotu cookie `device` a „semafor“, ktorého stránku si návštevník otvoril. Pokúsi sa nájsť v databáze klienta ktorého atribút `device` je rovnaký nájdená hodnota cookie `device`. Zároveň musí byť skryté číslo klienta, ktoré získa zapísaním do poradia, väčšie alebo rovné, ako číslo posledného vyvolaného klienta. Ak funkcia v databáze takého klienta nájde, znamená to, že návštevník už do poradia zapísaný je. Odošle sa JSON správa klientovi s hodnotou `"You are already in queue"`. Klient tak na základe obdržanej správy pokračuje ďalšou metódou na získanie informácií o jeho poradí. V prípade, že funkcia takého klienta v databáze nenájde, odosiela JSON správu s hodnotou `"Not in queue"`.

## 6.6 Testovanie aplikácie a ďalší vývoj

Testovanie aplikácie prebiehalo po častiach v priebehu celého vývoja aplikácie. Funkcionalita aplikácie bola najskôr otestovaná mnou, pomocou viacerých webových prehliadačov v normálnom režime a v režime inkognito. Následne mi s testovaním pomáhalo niekoľko ďalších ľudí. Časť z nich boli študenti technicky zameraných škôl a časť ľudia bez technického vzdelania, vo veku od 19 do 50 rokov.

Úlohou ľudí napomáhajúcich s testovaním bolo zaregistrovať si účet a vytvoriť nový „semafor“. Následne prebehlo otestovanie nazdieľania URL adresy. S pomocou testujúcich klientov bolo otestované prihlasovanie sa do poradia z rôznych prehliadačov a typov zariadení. Rovnako aj postupné vyvolávanie ľudí z poradia.

Testujúci klienti pomohli pri vývoji aplikácie odhaliť radu chýb a prišli aj so spomínaným nápadom na pridanie oranžovej farby „semaforu“. Od niekoľkých testujúcich klientov prišla aj myšlienka dostávania upozornení o zmene aktuálnej pozície v poradí. Táto funkcia zatiaľ nie je implementovaná v aplikácií, ale je možným predmetom ďalšieho vývoja




Obr. 6.7: **Ukážka sledovania svojho poradie na mobilnom zariadení**. Na prvom obrázku je možné vidieť, že klienta bola už vyvolaná. Svieti jej zelená farba a zobrazuje sa nápis *„Ste na rade!“*. Ďalší v poradí je klient na strednom obrázku. Je možné vidieť jeho čas čakania a tiež informáciu o tom, že je ďalší na rade, nikto pred ním už nečaká, a má sa pripraviť. Znázorňuje to aj oranžová farba na dopravnom semafore. Posledný klient musí na svoje poradie ešte čakať, nakoľko čaká pred ním v poradí 1 osoba a ďalšie jedna osoba sa nachádza vo vnútri. Svieti mu teda červená farba.

vo forme *push notifikácií*. V rámci ďalšieho vývoja by som chcela umožniť používateľom aplikácie výber z niekoľkých rôznych jazykov. Aktuálna verzia aplikácie je k dispozícii len v slovenskom jazyku. Pre lepšiu organizáciu poradia, som tiež premýšľala nad pridaním možnosti zadania časového zdržania. Ak by sa správca poradia pri niektorom klientovi zdržal napríklad o 5 minút, mohol by to zadať do systému. Všetkým čakajúcim klientom by sa tak zmenil ich čas o pripočítanie predpokladaného zdržania.

# Kapitola 7

# Záver

Prvou fázou vývoja aplikácie bolo preskúmanie už existujúcich riešení daného problému a porovnanie ich výhod a nevýhod. Nasledovalo oboznámenie sa s princípmi vývoja webových aplikácií a technológiami, ktoré sa na ich tvorbu využívajú. Ďalšou fázou bolo vytvorenie návrhu aplikácie, ktorý zahŕňal celkový spôsob fungovania aj náčrt užívateľského rozhrania na papier pre počítače aj mobilné zariadenia. Návrh bol predložený ľudom napomáhajúcim s testovaním aplikácie, ktorí sa tak už na začiatku mohli vyjadriť k vzhľadu aplikácie. Implementácia aplikácie tak bola o to jednoduchšia, že som nemusela premýšľať nad funkcionalitou a vzhľadom aplikácie priamo pri písaní kódu, ale vychádzala už z vytvoreného návrhu a postupne len pridávala ďalšie funkcie, prípadne upravovala už existujúce. Nedá sa povedať, že testovanie a umiestnenie na *Heroku* bola posledná fáza, nakoľko aplikácia bola na *Heroku* umiestňovaná po častiach počas celej doby vývoja aplikácie. To isté platí aj pre testovanie aplikácie.

Výsledná aplikácia spĺňa mnou preddefinovaný požiadavok – jednoduchosť a tiež zadanie bakalárskej práce. Používateľ si vytvorí poradovník, do ktorého sa môžu ľudia prihlasovať a on ich jedným kliknutím vyvoláva v tom poradí, ako sa zapísali do poradia. Aplikácia je doplnená o obrázok a princíp fungovania dopravného semaforu. K aplikácii bolo vytvorené video a plagát, ktoré vysvetľujú fungovanie aplikácie *Organize Semaphore*.

Doteraz som mala možnosť programovať iba v rámci školských projektov. Táto bakalárska práca bola pre mňa možnosť zoznámiť sa s technológiami, s ktorými som sa doposiaľ nestretla. Je to môj prvý väčší projekt, ktorého téma ma zaujala hneď na začiatku a plánujem ho v budúcnosti rozvíjať. Pri vývoji webovej aplikácie a taktiež pri písaní bakalárskej práce som sa naučila, ako veľmi vedomostne aj časovo náročný dokáže taký projekt byť.

# Literatúra


[1] *WebSocket* [online]. [cit. 2021-04-29]. Dostupné z: https://javascript.info/websocket#summary.

[2] Anderson, D. *How to use Microsoft Teams for Parent Teacher Interviews* [online]. [cit. 2021-05-10]. Dostupné z: https://www.sharepointfocus.com/microsoft-teams/how-to-use-microsoft-teams-for-parent-teacher-interviews/.

[3] Curinga, M. a Saravanos, A. Mobile First E-Learning. In:. Január 2017. DOI: 10.4018/978-1-5225-0783-3.ch039. ISBN 9781522507840.

[4] Django. *Object-relational Mappers (ORMs)* [online]. [cit. 2021-04-28]. Dostupné z: https://www.fullstackpython.com/object-relational-mappers-orms.html.

[5] Django. *PostgreSQL* [online]. [cit. 2021-04-27]. Dostupné z: https://www.fullstackpython.com/postgresql.html.

[6] Fabrizio Romano, A. R. *Learn Web Development with Python*. Packt Publishing Ltd, 2018. ISBN 978-1-78995-329-9.

[7] Foundation, I. D. *Responsive Design* [online]. [cit. 2021-04-20]. Dostupné z: https://www.interaction-design.org/literature/topics/responsive-design.

[8] Jerry Lee Ford, J. *Ajax Programming for the Absolute Beginner*. Course Technology Cengage Learning, 2009. ISBN 978-1-59863-564-5.

[9] Kaplan Moss, J. *Finally, Real-Time Django Is Here: Get Started with Django Channels* [online]. [cit. 2021-04-28]. Dostupné z: https://blog.heroku.com/in_deep_with_django_channels_the_future_of_real_time_apps_in_django.

[10] Kozlowski, A. *What Is The Difference Between UX and UI Designer and Front-end Developer?* [online]. [cit. 2021-04-20]. Dostupné z: https://www.ideamotive.co/blog/difference-between-ux-and-ui-designer-and-front-end-developer.

[11] Qassim, H. Testing Web Applications. *International Journal of Computer Science and Engineering*. December 2019, zv. 6, s. 1–9. DOI: 10.14445/23488387/IJCSE-V6I12P101.

[12] Salahaldin Juba, A. V. *Learning PostgreSQL*. Packt Publishing Limited, 2015. ISBN 978-1-78398-918-8.

[13] Samuel Dauzon, A. R. *Django: Web Development with Python*. Packt Publishing Ltd, 2016. ISBN 978-1-78712-138-6.

[14] Staff, W. A. *What is Bootstrap? An In-depth Guide of the Framework* [online]. [cit. 2021-04-29]. Dostupné z: https://wpamelia.com/what-is-bootstrap/.

[15] Svirca, Z. *Everything you need to know about MVC architecture* [online]. 2020 [cit. 2021-04-15]. Dostupné z: https://towardsdatascience.com/everything-you-need-to-know-about-mvc-architecture-3c827930b4c1.

[16] Čápka, D. *Lekce 3 - Představení MVC a MVT architektury v Django* [online]. 2018 [cit. 2021-04-15]. Dostupné z: https://www.itnetwork.cz/aktivita/article/python-django.

# Príloha A

# Obsah priloženého pamäťového média

Priložené pamäťové médium obsahuje súbor textový `README.txt` s popisom obsahu pamäťového média. Ďalej obsahuje vytvorený plagát k aplikácii, prezentačné video, technickú správu vo formáte `pdf`, priečinok so zdrojovými súbormi aplikácie a priečinok so zdrojovými súbormi technickej správy.