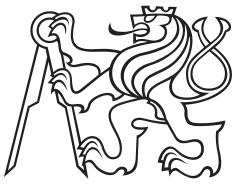
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Návrh mobilní aplikace pro administraci a zobrazování lékařských doporučení

Zdeněk Havelka

Vedoucí: doc. Ing. Daniel Novák Ph.D
Srpen 2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Havelka** Jméno: **Zdeněk** Osobní číslo: **475392**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Software**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Návrh mobilní aplikace pro administraci a zobrazování lékařských doporučení

Název bakalářské práce anglicky:

Mobile application for administration and visualization of medical recommendation

Pokyny pro vypracování:

1) Seznamte se s problematikou odvykání v klinické praxi.
2) Navrhněte aplikaci pro zobrazování tipů pro odvykání, které se budou objevovat v závislosti na environmentálních proměnných (čas, oblíbenost předchozích tipů, vstup od uživatele)
3) Implementujte jednoduchý chatovací systém pro komunikaci s uživatelem v omezené doméně.
4) Navrženou aplikaci otestujte minimálně na 4 uživatelích

Seznam doporučené literatury:

[1] H. Brendryen, P. Kraft, and H. Schaalma, “Looking Inside the Black Box: Using Intervention Mapping to Describe the Development of the Automated Smoking Cessation Intervention ‘Happy Ending’,” The Journal of Smoking Cessation, vol. 5, no. 1, pp. 29–56, Jun. 2010.
[2] H. Brendryen, F. Drozd, and P. Kraft, “A digital smoking cessation program delivered through internet and cell phone without nicotine replacement (happy ending): randomized controlled trial.,” Journal of medical Internet research, vol. 10, no. 5, p. e51, Jan. 2008.
[3] Kulhánek A., Gabrhelík R., Novák D. & Brendren H. (2018). eHealth intervention for smoking cessation for Czech tobacco smokers: Pilot study of user acceptance. Adiktologie, 18(2).

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Daniel Novák, Ph.D., Analýza a interpretace biomedicínských dat FEL

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

Datum zadání bakalářské práce: **12.02.2020** Termín odevzdání bakalářské práce: **14.08.2020**

Platnost zadání bakalářské práce: **30.09.2021**

doc. Ing. Daniel Novák, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. 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

Děkuji doc. Ing. Danielu Novákovi Ph.D a jeho kolegům Ing. Jindřichovi Prokopovi a Ing. Václavovi Burdovi za spolupráci, přínosné rady a možnost být součástí tak zajímavého projektu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 10. srpna 2020

Abstrakt

Práce se zabývá odvykáním kouření prostřednictvím mobilní aplikace. Dále pak implementací aplikace pro administraci klinických doporučení a blog a implementací jednoduchého chatovacího systému.

Klíčová slova: odvykání kouření, Python, Django, rozhraní, chat

Vedoucí: doc. Ing. Daniel Novák Ph.D

Abstract

Thesis deals with the problem of smoking cessation via mobile application. Further describes implementation of application for administration of clinical recommendations, blog and implementation of simple chat service.

Keywords: smoking cessation, Python, Django, interface, chat

Title translation: Mobile application for administration and visualization of medical recommendation

Obsah

Obrázky

Tabulky

Kapitola 1
Odvykání kouření pomocí mobilní aplikace

Kouření tabákových produktů patří mezi nejrozšířenější faktory ohrožující veřejné zdraví. [1, 2] I přes dobře známá zdravotní rizika, která kouření způsobuje jak uživateli, tak osobám kouření vystaveným, se nenechává značná část populace odradit a začíná s kouřením převážně v mladém věku. Velký problém nastává v situaci, kdy se ze zlozvyku stává nutnost a uživatel si na tabákových produktech vytváří závislost. Závislost na tabáku je jednou z nejsilnějších a pro kuřáka je nesmírně těžké při odvykání bez cigarety vydržet. Až 80% kuřáků je závislých a téměř stejná část (70%) by se kouření nejraději zbavila. Většina odvykacích pokusů, bez odborné asistence, končí neúspěchem. [3]

1.1 Princip

Rozvíjející se technologie přinášejí na pole medicíny a adiktologie nové možnosti. Jednou z nich je využití informačních technologií k terapeutickým účelům a intervenci právě při odvykání kouření. Zde je hlavní myšlenkou simulace poradenských sezení, která jsou svým obsahem uživateli ušita na míru. Ta mohou být realizována prostřednictvím mailové komunikace, webových stránek, nebo v rámci mobilní aplikace. Sezení sestávají z textu koncipovaného tak, jako by k uživateli promlouval poradce, a často od uživatele požadují zpětnou vazbu opět v textové podobě nebo formou zaškrtávání odpovědí. Tato nová forma intervence typicky začíná přípravnou fází, trvající zhruba dva týdny. Během ní se uživatel seznamuje s problematikou odvykání a připravuje se na den své poslední cigarety. Následuje přibližně měsíční, intenzivní a aktivní fáze odvykání. Po skončení se uživatel přesouvá do finální, méně intenzivní fáze. [4] Jednotlivá sezení se snaží nejen posílit uživatelovu motivaci, odhodlání a sebedůvěru, ale i jej seznámit s možnými kognitivními a

behaviorálními reakcemi nastávajícími při zapálení cigarety během odvykání. [5] Ukazuje se, že tato forma intervence je mezi pacienty přijímána kladně a někteří dokonce preferují tento způsob před vyhledáním jiné odborné pomoci. [2] Mimo text a formuláře je možné obohatit odvykací program například o chatovací službu, komunikaci prostřednictvím sms nebo o nahrané interaktivní zvukové odpovědi. [5]

1.2 Výhody

Kromě nové formy terapeutické intervence, přinášejí takto orientované aplikace celou řadu benefitů. V dnešní době, kdy značná část populace vlastní mobilní zařízení a je vybavena připojením k internetu, se stávají mobilní a webové aplikace snadno dostupným sortimentem s velkým potenciálem oslovit významné množství lidí. [1] Dalším faktorem je možnost využít aplikaci okamžitě a jednoduše. Například, když kuřák dostane nutkavou chuť na cigaretu. Zatímco existuje mnoho řešení, která dokáží potlačit fyzické symptomy abstinence, může to být právě odvykací aplikace, jež odvede pozornost na jiné téma, připomene důvody, proč se uživatel rozhodl přestat kouřit, nebo jen motivuje. [6] Bonusem je i možnost využívat aplikaci po celý den.

Kapitola 2

Administrace klinických doporučení a blogu

Klinická doporučení jsou v aplikaci ve formě rad nebo tipů, které mají za úkol kuřákovi pomoci zbavit se akutní chuti na cigaretu. Uživatel je vyvolá stisknutím SOS tlačítka. Tato klinická doporučení jsou sadou nejrůznějších aktivit, jež mohou pomoci změnit aktuální činnost a překonat tak ty nejkrizovější momenty.

2.1 Kategorizace

Pro efektivní zobrazování doporučení je nutné, aby co nejvíce odpovídala situaci, ve které se uživatel může nacházet. Z tohoto důvodu je vhodné doporučení kategorizovat podle různých kritérií a v závislosti na vstupu od uživatele adekvátně reagovat. Jako hlavní kritéria uvažujeme čas a místo. Kritérium času představuje denní dobu. Kritérium místa sleduje, zda-li je uživatel doma, v přírodě, nebo například v práci. Každému tipu přiřadíme optimální denní dobu a místo. Již tato dvě hlavní kritéria mohou vést k velkému zlepšení. Uvedu pár příkladů. Pokud by měl uživatel obdržet tip, který jej vybízí, aby zavolal svému nejlepšímu příteli, pak v nočních hodinách pro to nebude vhodná doba. V pracovní době si jen těžko začne kuřák číst novou knihu. Jako další možná kritéria pro zobrazení správného tipu se jeví uživatelova nálada nebo například zdravotní stav. Právě zobrazování korektních doporučení může být hlavním impulzem pro zvýšení jejich úspěšnosti.

2.2 Sledování úspěšnosti

Další faktor, který může pomoci při poskytování co nejlepších rad, je sledování jejich úspěšnosti. Po zobrazení tipu má uživatel možnost poskytnout zpětnou vazbu, zda-li mu dané doporučení pomohlo, či nikoliv. Atribut úspěšnosti

napomáhá vzniku nových tipů nebo vede k přepracování některých stávajících. Dále by bylo dobré uživatelům zobrazovat tipy, jejichž úspěšnost je již ostatními prověřená. Tipy, které konkrétnímu uživateli opakovaně nepomohly, v budoucnu nezobrazujeme.

2.3 Sledování zobrazení

Pod slovním spojením sledování zobrazení se skrývá relace mezi uživatelem a konkrétním doporučením. Díky této informaci víme, kdy a která doporučení byla uživateli zobrazena, a jak tato doporučení uživatel ohodnotil. Tuto informaci zohledníme při příštím zobrazení tipů uživateli tak, aby se dokola neopakovaly, a uživatel nenabýval z aplikace špatného dojmu.

2.4 Publikace doporučení

K rozšíření a případnému zkvalitnění kolekce doporučení, je přínosné inspirovat se přímo od uživatelů. Uživatel tak dostává možnost aktivně se zapojit a pomoci ostatním. Tipy od uživatele nelze zařadit do kolekce okamžitě. Mohou obsahovat špatnou diakritiku, gramatické chyby, chyby v interpunkci nebo být obsahem kompletně mimo tématiku odvykání. Proto je nutné, aby publikovaná doporučení byla před zařazením zkontrolována manuálně. Dalším problémem k řešení v této sekci je otázka vzniku potencionálních duplicit. S přibývajícími doporučeními je pravděpodobné, že si některá budou tématicky velice podobná, jen popsána trochu jinými slovy. Detekce duplicit bude více rozebrána v implementační části.

2.5 Podpora jazyků

Pro zvýšení dostupnosti aplikace je nutné uchovávat texty klinických doporučení a jednotlivých článků blogu ve více jazycích.

2.6 Blog

Zatímco klinická doporučení hrají roli jisté první pomoci, blog plní funkci obsáhlejších článků s cílem hlouběji seznámit uživatele s problematikou odvykání, lépe jej připravit na tento proces a sdělit cenné informace.

Kapitola 3

Chatovací systém

Chatovací systém, jako možnost okamžité online komunikace, využíváme pravidelně téměř všichni. Mezi rozšířené chatovací mobilní aplikace patří například Messenger, WhatsApp nebo Slack. Takovou formu komunikace označujeme za Instant Messaging (IM).

3.1 Využití

Zakomponování chatovací platformy do aplikace pomáhající s odvykáním kouření přináší mnoho výhod. Ať už jde o zprostředkování komunikace s vyškoleným odborníkem, nebo o možnost prostého sdělení svých úspěchů kamarádovi, který je při odvykání oporou. V neposlední řadě se otevírají dveře pro integraci vlastního chatbota, schopného v závislosti na uživatelově vstupu poskytnout některá z klinických doporučení, či odpovědět na nejrůznější otázky s tématikou odvykání. Ukazuje se, že využití IM služeb k intervenci pro odvykání kouření, je uživateli kladně přijímáno, převážně díky schopnosti poskytnout více osobní a adaptabilní podporu. [7] Zapojení IM služeb má potenciál ještě více prohloubit motivaci u uživatelů, kteří ji zpočátku mohou mít méně něž ostatní. [8] Vyžití chatovacího systému může být prospěšné k propagaci doprovodných rad v raných fázích odvykání, kdy uživatel potřebuje do začátku co nejvíce informací, a také jako forma skupinových sezení nabízejících různé příběhy a zkušenosti. [9]

3.2 Výběr Instant Messaging platformy

Pro implementaci chatovacího systému do vlastní aplikace se jako první rozumný krok jeví využití již existujícího systému a jeho integrace. Tyto

chatovací platformy zpravidla disponují knihovnami či SDK (Software Development Kit) dostupnými v několika programovacích jazycích a bohatou dokumentací pro jejich používání. Velká část z možných řešení, jimiž jsem se zabýval, jsou orientována primárně na CRM (Customer Relationship Management) a mimo funkcionalitu prostého chatu nabízejí mnoho dalších marketingově zaměřených vylepšení. Společný je pro většinu také cenový model, kdy výše ceny je určena počtem uživatelů, velikostí přenesených dat a případně dalšími parametry.

3.2.1 Intercom

Intercom se jeví jako největší hráč na trhu. Komunikace pomocí Intercomu je využita například v mobilní aplikaci pro odvykání kouření Alex. Základní verze konverzační služby začíná na 49$/měsíc, pokud aplikaci používá maximálně 200 uživatelů.

3.2.2 Sendbird

Dalším z kandidátů je platforma Sendbird. Ta nabízí SDK pro iOS, Android i Javascript a cena začíná tabulkově na zhruba 400$/měsíc. Nabízí se zde i varianta zcela zdarma, která je omezena na celkově 1000 uživatelů (z toho pouze 25 aktivních v jeden čas), historii zpráv pouze na 3 měsíce. Překročení limitů znamená povýšit na placenou verzi do 3 dnů.

3.2.3 Twilio

Podobně jako Sendbird i Twilio disponuje SDK pro iOs, Android a Web. Jedná se o cloudovou komunikaci s pay-as-you-go cenovým modelem, kdy zákazník platí pouze za aktivní uživatele a využité místo (storage).

3.2.4 Závěr

Cílem výběru chatovací platformy bylo najít nejvhodnějšího kandidáta s přihlédnutím k ceně a případné možnosti uchovávat zprávy mimo úložiště provozovatele platformy, jelikož se může jednat o citlivé klinické informace. Vzhledem k faktu, že žádný z výše uvedených kandidátů toto neumožňuje a rovněž i k poměrně vysokým cenám těchto služeb, jsem se rozhodl pro implementaci vlastního řešení.

Kapitola 4
Typ aplikace

Kapitola pojednává o typech implementovaných aplikací.

4.1 Server

Serverová aplikace (často označována jako backend) je taková aplikace, která skrze vystavená rozhraní poskytuje přístup ke zdrojům, nebo nejrůznějším službám. Aplikace pro administraci doporučení s chatovacím systémem je právě takovou aplikací. Existuje celá řada frameworků a technologií pro implementaci serverové aplikace. Mezi nejznámější patří například Django, Ruby On Rails, ASP.NET, nebo Spring Boot. [10] Použitým frameworkem je Django pro jazyk Python díky jeho bohaté funkcionalitě, čitelnosti a udržitelnosti.

4.2 Chatovací klient

Při tvorbě klienta bylo důležité se nejdříve rozhodnout, o jaký druh aplikace se bude jednat. Co by měl chatovací klient jistě splňovat je multiplatformnost. Tedy schopnost použití aplikace na telefonu, tabletu i klasickém počítači nehledě na platformu. Pro využití klienta vyškoleným odborníkem je dobré, aby aplikace mohla fungovat na klasickém počítači a jeho práce tak byla pohodlnější díky dostupné klávesnici a monitoru. Pro využití ostatními uživateli je potřeba vzít na vědomí, že se uživatel může nacházet v různých situacích bez přístupu k počítači. Vzhledem k faktu, že chat je úzce spjatý s nutností připojení k internetu, se jeví webová aplikace jako nejlepší kandidát. Webovou aplikaci je možné využít na jakémkoli zařízení, které disponuje webovým prohlížečem, bez nutnosti ji instalovat, či aktualizovat. Díky tomu je také dostupná v podstatě kdykoli. Z těchto důvodů jsem se rozhodl pro

implementaci webové aplikace. Pro implementaci chatovacího klienta jsem vybral framework Angular z důvodu osobních preferencí.

Kapitola 5
Vybrané pojmy a technologie

V této kapitole vysvětlím vybrané pojmy a představím použité technologie, klíčové pro další kapitolu pojednávající o implementaci.

5.1 Rozhraní

Co si představit pod pojmem webového rozhraní? Webové rozhraní, někdy také Web API (z anglického Application Programming Interface), na straně serveru představuje jeden čí více přístupových bodů (endpointů), a k nim definovanou komunikaci typu požadavek-odpověď (klient-server architektura). Ve většině případů je komunikace na server iniciována ze strany klienta. Zaslaný požadavek pak server zpracuje a posílá zpět odpověď. Komunikace je často realizována pomocí protokolu HTTP a řídí se principy REST architektury.

5.2 HTTP

HTTP (HyperText Transfer Protocol) je internetový protokol určený ke komunikaci, respektive výměně hypertextových dat, mezi klientem a serverem. Data mohou být například ve formátu HTML, XML, nebo JSON. Komunikace probíhá za použití několika definovaných HTTP metod (typů požadavků). [11] Mezi nejčastější patří požadavky GET, POST, PUT a DELETE. Pro kapitolu Implementace budou klíčové metody GET a POST. Metoda GET slouží k vyžádání dat ze zdroje. Požadavek může být rozšířen o sérii atributů zadaných přímo do URL zdroje. Atributy přidáváme za symbol '?' a oddělujeme pomocí znaku '&'. Metoda POST data v jednom z výše zmíněných formátů na specifikovanou URL odesílá. Odpověď v sobě skrývá kód, který indikuje, zda vše proběhlo v pořádku, došlo k přesměrování, či chybě na straně serveru, nebo chybě v požadavku.

5.3 REST

REST (REpresentational State Transfer) je architektonický styl navržený pro přístup a manipulaci s daty po síti. Je definován několika principy. [12]

1. Klient-server architektura
2. Bezstavovost - veškeré potřebné informace jsou obsaženy v požadavku
3. Možnost ukládání odpovědí serveru do cache paměti
4. Vrstevnatý systém - přidání jakékoli vrstvy mezi klienta a koncový server nenarušuje běh komunikace
5. Možnost přenosu spustitelného kódu za účelem rozšíření funkcionality klienta
6. Jednotné rozhraní

Byť je REST jako architektura nezávislá na protokolu, či použitých technologiích, je často spojována s formou komunikace pomocí protokolu HTTP.

5.4 Serializace dat

Serializace dat je proces, kdy jakýkoli objekt převedeme na sekvenci bytů tak, abychom jej mohli například zaslat po síti. Data se standardně serializují do formátů XML, nebo JSON. Při práci na webovém rozhraní jsem zvolil formát JSON pro jeho dobrou čitelnosti a jednoduchou práci s ním v jazyce Python a Javascript. Data ve formátu JSON jsou uskupena do objektů tvořených páry klíč-hodnota. Jednoduchý objekt ve formátu JSON může vypadat následovně.

```
{
  "name": "Zdenek",
  "surname": "Havelka",
  "age": 22
}
```

5.5 Django

Serverová část aplikace je napsána v jazyce Python za použití webového frameworku Django. Slovo backend představuje to, co je v pozadí, tedy typicky serverovou aplikaci napojenou na databázi, schopnou komunikace s

uživatelskou aplikací. Díky použití jazyku Python je kód dobře čitelný. Django je open-source framework, jehož velkou výhodou je schopnost automaticky vytvořit administrátorskou sekci v závislosti na datovém modelu, tedy poskytnout základní webové uživatelské rozhraní pro přehled a CRUD operace nad modelovými entitami. Dále je Django schopen objektově relačního mapování a pomocí migrací tak umožňuje vytvářet či upravovat databázové tabulky v závislosti na datovém modelu bez nutnosti zasahovat přímo do databáze. Framework Django vyniká svojí bohatostí a množstvím funkcionalit, bezpečností a dobrým škálováním. [13] Díky tomu je ideálním frameworkem pro velice svižný vývoj.

5.6 PostgreSQL

PostgreSQL je open-source relační databázový systém. Je jedním z databázových systémů přímo podporovaných frameworkem Django. Databázový systém slouží k uložení dat, s nimiž poté můžeme manipulovat prostřednictvím naší aplikace. K přímé práci s uloženými daty používáme dotazovací jazyk SQL (Structured Query Language), jež je standardem pro většinu relačních databází. Relační databáze uchovávají data v tabulkách, kde jednotlivé sloupce reprezentují atributy a řádky vlastní data.

5.7 Technologie WebSocket

Od chatovacího systému primárně požadujeme, aby byla komunikace okamžitá. Vyžití standardní klient-server komunikace za účelem výměny zpráv mezi jednotlivými uživateli nese jistá úskalí. Cyklickým a nepřetržitým dotazováním se serveru, zda-li uživateli nedorazila nová zpráva, bychom server jistě při velkém počtu aktivních uživatelů zatížili do neúnosné míry. Proto zde není možné využít standardní HTTP komunikaci. Technologie WebSocket je protokol pro oboustrannou komunikaci mezi klientskou aplikací a serverem. WebSocket komunikace je iniciována ze strany klienta speciálním HTTP požadavkem, který informuje server o změně komunikačního protokolu. Takto oboustranná komunikace je ideální pro realizaci například chatovacího systému.[14]

5.8 Django Channels

Django Channels je Python framework rozšiřující základní funkcionalitu Djanga o práci s různými komunikační protokoly mimo HTTP. Zejména

pak o komunikaci prostřednictvím WebSocket. [15] Klíčovým aspektem je *channel*, který svým chováním představuje FIFO frontu jednotlivých zaslaných zpráv. Zprávy z této fronty asynchronně vybírají a zpracovávají příslušní posluchači. K zaslání odpovědi slouží *reply_channel*.[16] Pro zaslání zprávy do více *reply_channel* jsou využity *Groups*. Ty poskytují možnost svázání více *reply_channel* a simulují tak "broadcast". [17]

5.9 Angular

Angular je populární open-source framework pro tvorbu single-page webových aplikací vyvíjený společností Google. Za single-page aplikace považujeme takové aplikace, které dynamicky mění a překreslují svůj obsah, bez nutnosti načítání nových stránkek. Za tímto účelem doplňuje Angular HTML o vlastní šablonovací systém umožňující měnit obsah jednotlivých elementů před jejich vykreslením a interní systém navigace mezi jednotlivými "stránkami". Angular stojí na hierarchickém principu komponent, kde každá komponenta definuje logiku a data zobrazená v přiřazené HTML šabloně. [18]Angular je postaven na programovacím jazyce Typescript. Typescript je nadmnožinou Javascriptu, což znamená, že jakýkoli kód v jazyce Javascript je validním kódem v jazyce Typescript. Typescript je také ve finále do Javascriptu kompilován.

Kapitola 6

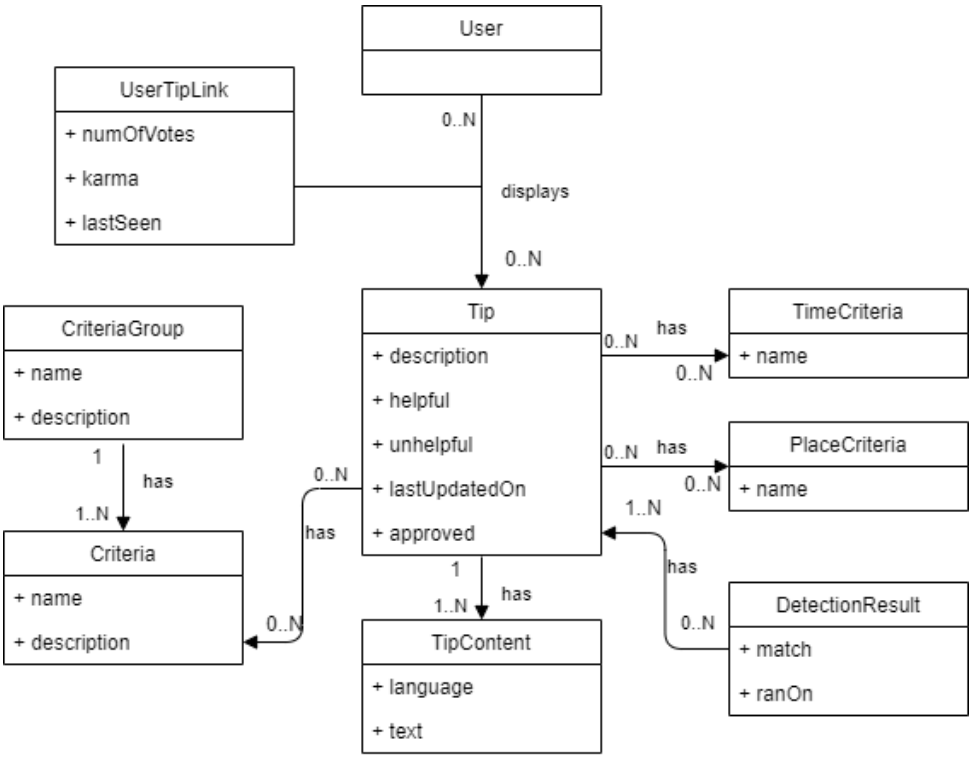
Implementace rozhraní pro administraci doporučení a blogu

V této kapitole se zaměřím na postupy a algoritmy použité při implementaci webového rozhraní pro administraci klinických doporučení a blogu ve výše představeném frameworku Django.

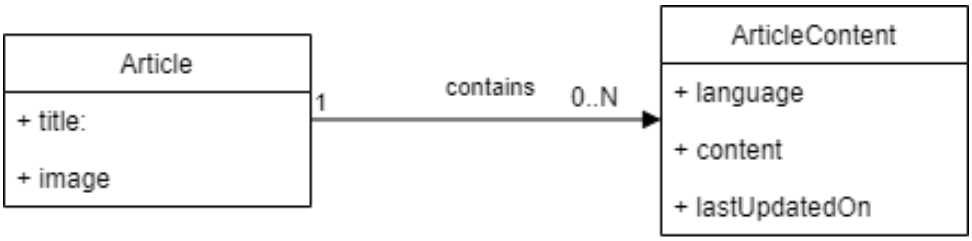
6.1 Datový model

Třídy klientských doporučení

- **Recommendation** představuje entitu samotného doporučení. U této třídy evidujeme počet hlasů, jež ji shledaly užitečnou, neužitečnou, kdy byla naposledy upravena, zda byla schválena (je ji možné zobrazit uživatelům) a její stručný popis.
- **Recommendation Content** tvoří obsah doporučení. Sestává z vlastního textu a jazyka, ve kterém je napsán.
- **Criteria, Place Criteria a Time Criteria** jsou napojené jednotlivé atributy pro filtrování doporučení.
- **Criteria Group** zastřešuje kolekci podobných kritérií pro lepší a pohodlnější správu v administrátorské sekci.
- **User Tip Link** nám dává informaci o tom, který uživatel si naposledy zobrazil jaký tip, a kdy. Atribut Karma je číselný indikátor oblíbenosti tipu daným uživatelem.



Obrázek 6.1: Datový model klientských doporučení



Obrázek 6.2: Datový model blogu

Třídy pro blog

Pro vytvoření modelu pro blog postupujeme obdobně. Stačí nám pouze dvě třídy.

- **Article** s atributy obsahujícími titulek článku a obrázek
- **Article Content** podobně jako RecommendationContent nese vlastní

text článku a jazyk.

6.2 Přenesení modelu do Djanga

Po vytvoření databázového modelu je potřeba tento model přenést do naší aplikace. V souboru models.py vytvoříme ke každé třídě ze schématu odpovídající třídu s příslušnými atributy v jazyce Python, která rozšiřuje třídu Model z knihoven Djanga. Typy a vlastnosti jednotlivých atributů specifikujeme pomocí instancí třídy Field.

```
1 class RecommendationContent(models.Model):
2         content = models.TextField(default=False)
3         language = models.CharField(max_length=50,
             choices=LANGUAGES)
4         recommendation = models.ForeignKey(
             Recommendation, on_delete=models.CASCADE)
```

Ačkoli je v Djangu možné specifikovat many-to-many vazbu pomocí models.ManyToMany(), vazby one-to-many (případně many-to-one) realizujeme pomocí ForeignKey(), čímž na objektu, na který je pomocí cizího klíče ukazováno, ztrácíme referenci na tuto kolekci. Pro tyto účely je možné zadefinovat vlastní metodu anotovanou značkou @property a seznam objektů extrahovat explicitně. Nevýhodou je, že se pro každou entitu vyhodnotí dotaz do databáze, tedy v případě velkého množství dotazů může dojít ke zpomalení.

6.3 Migrace

S úpravou či vytvořením nových modelových tříd přestává odpovídat databázové schéma. K tomuto účelu slouží migrace. Jedná se o způsob propagace změn v aplikaci do databáze bez nutnosti volání vlastních sql skriptů. Po spuštění migrace se údaj o proběhnutí uloží do databázové tabulky. Po provedení změn v souboru models.py, které se mají zpropagovat, stačí spustit následující skripty z prostředí terminálu.
python manage.py makemigrations <jméno_modulu>
python manage.py migrate <jméno_modulu>

6.4 Serializace

Před popisem implementace hlavních funkcionalit rozhraní je nutné se podívat na problém serializace. V souboru serializers.py vytváříme serializační třídu pro každou z našich modelových tříd, kterou chceme po síti nejen odeslat, ale také ve specifikovaném formátu přijmout, deserializovat a uložit. Jedná se opět o třídu v jazyce Python. Ta rozšiřuje serializers.ModelSerializer z knihoven Djanga. V této třídě primárně uvádíme, k jakému modelovému objektu se vztahuje a jaké atributy se mají serializovat. Dále je možné implementovat metody create() a update(). To zajistí, že při deserializaci vnořených objektů se tyto objekty buď vytvoří, aktualizují, nebo pouze vrátíme referenci na již vytvořený objekt. Pokud by došlo k deserializaci objektu Recommendation, v jehož JSON formátu jsou uvedeny názvy přiřazených kritérií, nebude dobré tato kritéria opět ukládat do databáze a vytvářet tak duplicity. Lepší variantou je se při procesu deserializace podívat, zda kritérium stejného jména v databázi již existuje, a případně vrátit odkaz na konkrétní entitu.

6.5 Bezpečnost

Aby k dostupným informacím implementovaného API a funkcionalitě chatovacího systému měli přístup pouze registrovaní uživatelé, je nutné zavést jistá bezpečností opatření. Zavedené bezpečnostní opatření začíná podobně jako Basic Authentication, tedy je zaslán požadavek na server obsahující autorizační hlavičku. Obsah autorizační hlavičky tvoří klíčové slovo 'Basic' a poté zakódované jméno:heslo pomocí Base64 kódování. Po přijetí přístupových údajů server vygeneruje token s omezenou platností, který zasílá uživateli zpět. Po dobu platnosti je uživatel schopen se pomocí tokenu identifikovat.

6.6 Funkcionalita rozhraní

Nyní se přesouváme k obsluze příchozích požadavků na server. URL, na které budou požadavky přicházet, zaregistrujeme do souboru zodpovědného za konfiguraci routování společně s třídou implementující logiku obsluhování požadavku. Pozor je třeba dát na autorizaci. Aby API mohl využít pouze registrovaný uživatel, je nutné použít třídy View pro to určené, nebo specifikovat permission_classes atribut.

6.6.1 Zobrazení

Pro zobrazení jednoho, či více doporučení použijeme GET požadavek na specifikovanou URL. Doporučení, které chceme obdržet, můžeme specifikovat pomocí atributů zadaných přímo do URL. Zadané atributy se transformují do filtru odpovídající SQL "where"klauzuli. Všechny podmínky musí být splněny, což v SQL odpovídá spojení jednotlivých logických výrazů slovem "and". Do filtru se automaticky přidává kritérium času a podmínka, že je doporučení schválené. Vrácené doporučení tedy bude vždy korespondovat s denní dobou odeslání požadavku. Z vyfiltrované množiny, seřazené podle míry úspěšnosti se vybere právě to doporučení, které uživatel viděl jako poslední.
Možné atributy specifikující požadavek jsou:

- Id - specifikace jednoho konkrétního doporučení
- Place - seznam kritérií na místo
- Criteria - seznam dalších kritérií pro filtrování
- Date - datum poslední změny
- Language - jazyk, do které musí být tip přeložen

6.6.2 Hodnocení

Doporučení je možné ohodnotit zasláním POST požadavku. Data zaslaná v tomto požadavku obsahují identifikátor cílového doporučení a pravdivostní hodnotu, zda byla zaslaná rada shledána užitečnou. V tento moment se vytváří, či aktualizuje vazba uživate na konrétní doporučení. S přibývajícím počtem zobrazení konkrétního doporučení se uživateli snižuje váha jeho hodnocení.

6.6.3 Publikace

Pokud se uživatel rozhodne podělit se o tipy, jež mu pomáhají zbavit se chuti na cigaretu, je zaslán opět POST požadavek s novým doporučením zapsaným v JSON formátu. Jeho obsah se deserializuje a uloží. Nyní narážíme na problém s možnými duplicitami.

6.7 Kontrola duplicit

Míra duplicity doporučení znamená míru podobnosti jejich textů ve stejném jazyce. Algoritmy určující míru podobnosti se liší podle přístupu k textu

na algoritmy pracující čistě s lexikální stránkou a algoritmy založené na sémantice. V této části vysvětlím, na jaké problémy jsem narazil, jak některé řešit a popíši metriku kosínové podobnosti, kterou jsem zvolil pro určení míry duplicity doporučení.

6.7.1 Problémy

Při kontrole duplicit existuje celá řada faktorů, které je nutné vzít na vědomí. Text publikovaný uživatelem nemusí být vždy příkladný. Při porovnávání textu, či porovnávání jednotlivých slov, je velmi pravděpodobné, že narazíme na následující problémy:

- Gramatické chyby
- Chyby v interpunkci
- Překlepy
- Časování
- Použitý pád

Část těchto problémů odstraníme preprocessingem textu.

6.7.2 Preprocessing

Preprocessing je proces aplikovaný na text před samotným určením míry duplicity. Během tohoto procesu dochází k sekvenční aplikaci několika kroků. [19]

- Převedení celého textu na malá (velká) písmena
- Odstranění interpunkční znamének
- Odstranění takzvaných "white-space"znaků (tabulátor, mezera)
- Odstranění slov jako jsou členy (takzvané "stop words")
- Tokenizace - rozdělení textu na menší části, například na jednotlivá slova

Díky preprocessingu můžeme eliminovat některé ze zmíněných chyb.

Textové řetězce "Ahoj! Jak se máš?"a "Ahoj, jak se máš"by se po provedení tokenizace shodovaly jen v řetězci "se". Po aplikaci celého preprocessingu dostáváme z obou řetězců totožný seznam slov {"ahoj", "jak", "máš"}, uvažujeme-li slovo "se"jako stop-word z bodu 4.

6.7.3 Vektorizace textu

Vektorizace textu nám umožňuje řetězec znaků reprezentovat jako vektor, který dostaneme následovně. Napříč porovnávanými texty vytvoříme seznam všech slov a označme jej B. Složka vektoru v v textu i na pozici j je rovna počtu výskytů j-tého slova z B v textu i.
Pro řetězce v_1 = "Přečtu si knihu", v_2 = "Rozečtu novou knihu"a seznam B = {"přečtu", "si", "knihu", "rozečtu", "novou"} vypadají vektory následovně: $v_1 = (1, 1, 1, 0, 0)$, $v_2 = (0, 0, 1, 1, 1)$ Jak byly vektory vytvořeny vidíme v tabulce níže.

přečtu	si	knihu	rozečtu	novou
1	1	1	0	0
0	0	1	1	1

Jelikož frekventovaně se vyskytující slova zpravidla nejsou ta sémanticky nejvýznamnější, můžeme vektor obsahující počty výskytů jednotlivých slov zaměnit za vektor obashující tf-idf (term frequency–inverse document frequency) hodnoty. Tím docílíme reflektování důležitosti méně frekventovaných slov významných a specifických pro daný dokument. Jednotlivé složky vektoru poté vypočteme následovně.

$$tfidf(d,t) = tf(d,t) \times \log \frac{|D|}{df(t)}$$

Kde $tf(d,t)$ označuje počket výskytů slova t v řetězci d, D je počet všech dokumentů a $df(t)$ je počet dokumentů, ve kterých se slovo t vyskytuje. [20]

6.7.4 Kosínová podobnost

Kosínová podobnost vychází ze známého vzorečku pro výpočet úhlu mezi vektory a je jednou z nejpopulárnějších metrik při porovnávání textů. [20]

$$\cos\phi = \frac{v_1 \cdot v_2}{\|v_1\| \cdot \|v_2\|}$$

Kapitola 7

Implementace chatovacího systému

7.1 Princip

WebSocket komunikace v rámci Django Channels je relativně jednoduchá. Po inicializaci spojení předá klient serveru identifikátor, který je unikátní uživatelům, kteří mezi sebou chtějí komunikovat. Tito uživatelé jsou po navázání spojení přiřazeni do skupiny a veškeré přichozí zprávy na server mohou být zaslány jako broadcast všem ostatním. Pro zahájení komunikace s jiným uživatelem (případně více uživateli) je tedy nutné znát předem "dohodnutý"identifikátor.

7.1.1 Zprávy

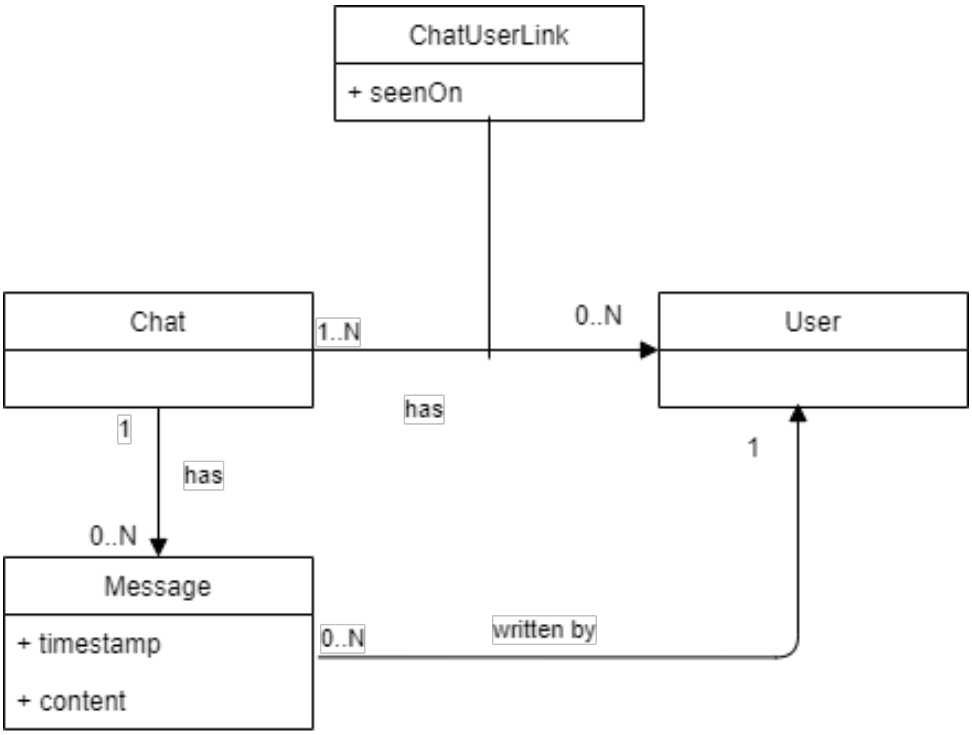
Odesílané zprávy jsou reprezentovány jako řetězec znaků obsahující JSON objekt. Objekt obsahuje 2 klíče.

- **type** - specifikuje druh zprávy (message, get_last_messages)
- **payload** - nese vlastní data zprávy

Typ message určuje, že se jedná o klasickou zprávu konverzace. Get_last_messages s parametry *chatId*, *size*, *offset* v *payload* poli indikuje snahu klienta o vyžádání historie zpráv. *Size* a *offset* slouží k postupnému načítání historie a minimalizují tak přenesená data.

7.2 Doménový model

Za účelem vedení historie zpráv a identifikátorů jednotlivých konverzací, je nutné tyto informace ukládat do databáze. K tomu slouží níže uvedené entity:



Obrázek 7.1: Datový model chatovacího systému

- **Chat**
  Entita konverzace
  id - slouží jako identifikátor konverzace
  participants - vazba na konkrétní uživatele pomocí ChatUserRelation

- **Message**
  Entita zprávy
  chat - konverzace, ke které zpráva patří
  content - obsah zprávy
  user - uživatel, který zprávu odeslal
  timestamp - datum a čas odeslání

- **ChatUserRelation**
  Vazební entita mezi konverzací a uživatelem nesoucí informaci posledního zobrazení konverzace konkrétním uživatelem.

7.3 Consumer

Jako consumer je označován v dokumentaci Django Channels posluchač, jehož úkolem je zpracování požadavku na inicializaci a ukončení WebSocket komunikace a zpracování přijatých zpráv. [21]

- **Otevření spojení**
  Při novém spojení Consumer ověří, zda-li je uživatel součástí systému, přijme identifikátor z URL a přidá uživatele do skupiny. Při novém spojení Consumer získá posledních 10 zpráv z konverzace a zašle je zpět uživateli.
- **Přijetí zprávy**
  Po přijetí zprávy Consumer provede definované akce v závisloti na typu zprávy. Vytvoří a uloží instanci třídy Message, nebo získá specifikované zprávy z historie a odesílá zpět patřičnou odpověď.
- **Odpojení**
  Po odpojení Consumer vyřadí uživatele ze skupiny.

7.4 CORS

Na jeden z problémů, na které jsem narazil při implementaci je CORS. Jelikož je chatovací klient samostatnou webovou aplikací, má jiný origin. Slovem origin je v rámci webových aplikací označována trojice - protokol, doména, port. CORS (Cross Origin Resource Sharing) označuje zásady pro přístup ke zdrojům s odlišným originem. Při pokusu o přístup ke zdrojům z jiného originu je odeslán cross-origin HTTP požadavek, který očekává odpověď s hlavičkou Access-Control-Allow-Origin a doménou, z níž byl požadavek odeslán, případně symbolem "*". [22]

Kapitola 8
Chatovací klient

V této kapitole se zabývám vývojem webové aplikace chatovacího klienta.

8.1 Design uživatelského rozhraní

Cílem designu uživatelského rozhraní byla snaha o splnění několika obecných vývojových zásad, které vedou k příjemnému uživatelskému zážitku. [23]

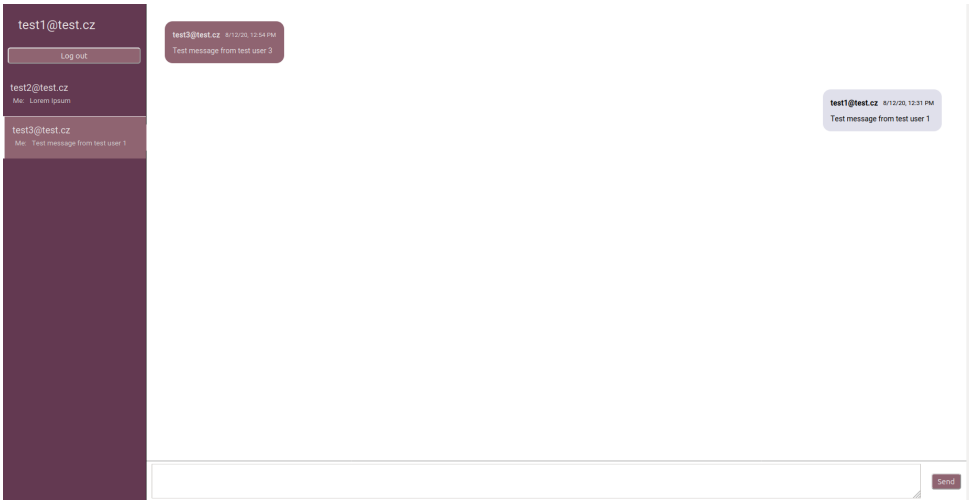
- Líbivý celkový vzhled
- Jednoduchost
- Efektivnost
- Ovladatelnost
- Podobnost stávajícím řešením

Design chatovacího klienta je inspirován běžně používanými aplikacemi. Proto je jednoduché s aplikací pracovat a ovládat ji. Aplikace je tvořena pouze přihlašovací obrazovkou a obrazovkou sloužící k výběru konverzace, zobrazení a odeslání zpráv.

8.1.1 Vykreslování zpráv

Z používání aplikací jako je Messenger, nebo Whatsapp jsme zvyklí, že vykreslování zpráv má určitá pravidla. Zprávy uživatele jsou vykresleny do bublin zarovnaných k jedné straně, ostatní zprávy do jinak barevných bublin na straně druhé. Toto se při testování ukázalo jako možný problém. Jelikož obrazovka zobrazující zprávy zabírá téměř celou šíři okna prohlížeče, zarovnáním zpráv k opačným okrajům obrazovky se zhoršila přehlednost

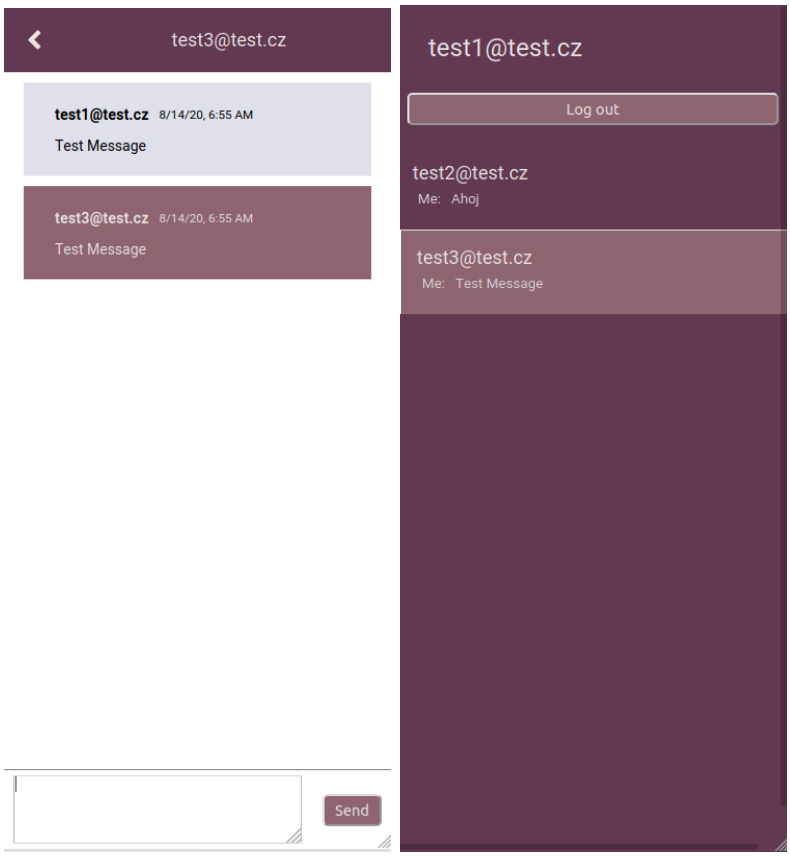
vykreslených zpráv a celkový dojem uživatelů. Tento problém se vyskytoval při testování na 24"full HD monitoru. Jako možné řešení se naskytla varianta nastavení fixní maximální velikosti obrazovky, nebo zarovnání všech zpráv k jedné straně. Varianta zarovnání zpráv k levému okraji se jeví jako nejlepší řešení.



Obrázek 8.1: Chatovací klient

8.1.2 Responzivní design

Aby byla aplikace použitelná na napříč různými zařízeními, je potřeba její uživatelské rozhraní přizpůsobit různé velikosti displejů. K tomu slouží v CSS takzvané *mediaselektory*, jejichž pomocí předefinujeme jednotlivé styly pro různé velikosti displejů. [24] Zatímco na obrazovce tabletu či stolního počítače není problém zobrazit panel pro výběr konverzace a panel chatu vedle sebe, pro displeje mobilních zařízení toto není prakticky možné. Z tohoto důvodu muselo být uživatelské rozhraní pro mobilní zařízení upraveno tak, aby byla vždy aktivní pouze obrazovka panelu pro vybrání chatu, nebo obrazovka se zprávami vybrané konverzace.



(a) : Obrazovka chatu na telefonu **(b) :** Menu pro výběr chatu

Kapitola 9
Testování

Testování je nedílnou součástí softwarového vývoje. V této kapitole představím použité nástroje pro testování implementovaného API a chatovacího systému.

9.1 Testování rozhraní

Pro testování rozhraní administrace klinických doporučení a blogu jsem zvolil nástroj Postman. Postman je ideálním nástrojem pro testování REST API. Postman umožňuje vytvářet kolekce HTTP požadavků, a k nim navázat testovací skripty v jazyce Javascript kontrolující například status odpovědi, či přesně specifikovat očekávanou odpověď. Vytvořené testovací kolekce slouží zároveň jako kvalitní dokumentace vytvořeného rozhraní. [25]

9.2 Uživatelské testování chatovacího klienta

Ačkoli je možné samotnou funkcionalitu aplikace pokrýt *unit a integračními testy*, neodhalíme možné problémy v designu a návrhu aplikace. K tomu slouží uživatelské testování, které má za úkol zjistit, jak dobře se uživatelům s aplikací pracuje, jak na ně aplikace působí nebo například jak rychle dokáže uživatel splnit zadaný úkol. Zpětná vazba uživatelů je důležitým podkladem pro odhalení návrhových chyb a budoucí vývoj v podobě možných vylepšení.

9.2.1 Průběh

Chatovací klient byl testován 6 uživateli z různých věkových kategorií a s různými zkušenostmi s informačními technologiemi. Na úvod byli uživatelé obeznámeni s jakou aplikací budou pracovat. Poté obdrželi uživatelé sadu bodů pro simulaci průchodu aplikací. Testování proběhlo ve 3 kolech s cílem získat

informace o přívětivosti uživatelského rozhraní a jednoduchosti jeho ovládání. Nejdříve byla aplikace testována na notebooku s 15"full HD displejem, poté na 24"full HD monitoru a ve finále došlo k simulaci displeje mobilního zařízení. Během testování byli uživatelé pozorováni. Po ukončení následoval krátký rozhovor za účelem získání zpětné vazby.
Body průchodu aplikací:

1. Příhlásit se do aplikace přiděleným uživatelským jménem a heslem.
2. Najít přidělený kontakt v seznamu.
3. Přepnout do okna vybraného chatu.
4. Odeslat sérii zpráv
5. Odhlásit se z aplikace

Dotazník s možnými odpověďmi:

1. Věk:
   - 20-30
   - 30-40
   - 40-50
   - 50+
2. Pohlaví:
   - Muž
   - Žena
3. Zkušenost s používáním informačních technologií:
   - Začátečník
   - Mírně pokročilý
   - Středně pokročilý
   - Pokročilý
4. Jak často používáte chatovací aplikace:
   - Vůbec
   - Občas (1-2 týdně)
   - Denně

9.2.2 Výsledky a zpětná vazba

Testování se zúčastnili celkem 2 ženy a 4 muži s věkovým rozpětím od 21 do 53 let. Většina uživatelů používá chatovací aplikace denně. Při pozorování nebyly zjištěny žádné překážky, které by bránily průchodu aplikací. Ze zpětné vazby bylo zjištěno, že některým uživatelům chybí možnost zaslat emoji. Dalším odhaleným problémem bylo zobrazování zpráv, kdy vykreslování zpráv na protilehlé strany obrazovky se zdálo uživatelům méně přehledné. Po ukončení testování došlo k přepracování zobrazování zpráv. Uživatelé popsali práci s aplikací jako intuitivní.

- Výsledky dotazníku:

1. Věk: 21
   Pohlaví: Žena
   Zkušenost s používáním informačních technologií: Mírně pokročilý
   Jak často používáte chatovací aplikace: Denně
2. Věk: 22
   Pohlaví: Muž
   Zkušenost s používáním informačních technologií: Středně pokročilý
   Jak často používáte chatovací aplikace: Denně
3. Věk: 53
   Pohlaví: Muž
   Zkušenost s používáním informačních technologií: Pokročilý
   Jak často používáte chatovací aplikace: Denně
4. Věk: 49
   Pohlaví: Žena
   Zkušenost s používáním informačních technologií: Mírně pokročilý
   Jak často používáte chatovací aplikace: Občas
5. Věk: 22
   Pohlaví: Muž
   Zkušenost s používáním informačních technologií: Mírně pokročilý
   Jak často používáte chatovací aplikace: Denně

6. Věk: 28
Pohlaví: Muž
Zkušenost s používáním informačních technologií: Středně pokročilý
Jak často používáte chatovací aplikace: Denně

Kapitola 10

Závěr

Hlavním cílem mé práce byl vývoj, implementace a otestování serverové aplikace poskytující rozhraní pro administraci klinických doporučení, blogu a jednoduchého chatovacího systému s klientským rozhraním. Práce mi umožnila seznámit se s problematikou závislosti na kouření a možnostmi zapojení informačních technologií v klinické praxi.
Ve své práci vysvětluji podstatu a význam klinických doporučení v aplikaci pro odvykání kouření a funkcionalitu, kterou by mělo rozhraní pro jejich administraci obsahovat, a představuji možnosti využití chatovacího systému. Dále popisuji důležité pojmy a použité technologie, které jsou klíčové pro implementační část. Během implementace jsem se seznámil se základními principy a limity některých technologií.
Posledním krokem bylo testování. Při testování jsem se zaměřil především na funkcionalitu implementovaného REST API, které jsem testoval pomocí nástroje Postman a chatovacího klienta. Chatovací klient byl testován 6 uživateli. Zpětná vazba týkající se problematiky vykreslování zpráv na 24"monitoru byla zapracována do aplikace. Výsledkem mé práce je funkční serverová a webová aplikace, kdy obě aplikace jsou připravené pro nasazení. Do budoucna bych rád rozšířil chatovací systém o podporu emoji.

Literatura

[1] Nathan K Cobb, Amanda L Graham, M. Justin Byron, Raymond S Niaura, and David B Abrams. Online social networks and smoking cessation: A scientific research agenda. *J Med Internet Res*, 13(4):e119, Dec 2011.

[2] A. KULHÁNEK, R. GABRHELÍK, NOVÁK D., V. BURDA, and H. BRENDRYEN. ehealth intervention for smoking cessation for czech tobacco smokers: Pilot study of user acceptance. *Adiktologie*, pages 81–85, 2019.

[3] Eva Králíková. Jak pomoci pacientům přestat kouřit? *Interní medicína pro praxi*, 13(11):453–454, 2011.

[4] Håvar Brendryen, Pål Kraft, and Herman Schaalma. Looking inside the black box: Using intervention mapping to describe the development of the automated smoking cessation intervention 'happy ending'. *The Journal of Smoking Cessation*, 5:29–56, 06 2010.

[5] Håvar Brendryen, Filip Drozd, and Pål Kraft. A digital smoking cessation program delivered through internet and cell phone without nicotine replacement (happy ending): Randomized controlled trial. *Journal of medical Internet research*, 10:e51, 02 2008.

[6] Ifa Nofalia. A systematic review mobile or web-based intervention for smoking cessation. *Babali Nursing Research*, 1(1):31–38, 2020.

[7] Tzu Tsun Luk, Sze Wing Wong, Jung Jae Lee, Sophia Siu-Chee Chan, Tai Hing Lam, and Man Ping Wang. Exploring community smokers' perspectives for developing a chat-based smoking cessation intervention delivered through mobile instant messaging: Qualitative study. *JMIR Mhealth Uhealth*, 7(1):e11954, Jan 2019.

[8] Man Ping Wang, Tzu Tsun Luk, Yongda Wu, William H Li, Derek Y Cheung, Antonio C Kwong, Vienna Lai, Sophia S Chan, and Tai Hing Lam. Chat-based instant messaging support integrated with brief interventions for smoking cessation: a community-based, pragmatic, cluster-randomised controlled trial, Jul 2019.

[9] Esther Granado-Font, Carme Ferré-Grau, Cristina Rey-Reñones, Mariona Pons-Vigués, Enriqueta Pujol Ribera, Anna Berenguera, Maria Luisa Barrera-Uriarte, Josep Basora, Araceli Valverde-Trillo, Jordi Duch, and Gemma Flores-Mateo. Coping strategies and social support in a mobile phone chat app designed to support smoking cessation: Qualitative analysis. *JMIR Mhealth Uhealth*, 6(12):e11071, Dec 2018.

[10] Server-side web frameworks. `https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks`.

[11] An overview of http. `https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview`.

[12] Roy Thomas Fielding. *REST: Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.

[13] Django overview. `https://www.djangoproject.com/start/overview`.

[14] Ian Fette and Alexey Melnikov. The websocket protocol, 2011.

[15] Django channels. `https://channels.readthedocs.io/en/latest/`.

[16] Django - channel. `https://channels.readthedocs.io/en/1.x/concepts.html#what-is-a-channel`.

[17] Django - groups. `https://channels.readthedocs.io/en/1.x/concepts.html#groups`.

[18] Introduction to angular concepts. `https://angular.io/guide/architecture`.

[19] Matthew James Denny and Arthur Spirling. Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it. *SSRN Electronic Journal*, 2017.

[20] Anna Huang. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student*

conference (NZCSRSC2008), Christchurch, New Zealand, volume 4, pages 9–56, 2008.

[21] Django - consumers. `https://channels.readthedocs.io/en/latest/topics/consumers.html`.

[22] Cross-origin resource sharing (cors). `https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS`.

[23] N Bhaskar, Ravi Babu, S Govindarajulu, Uday Bhaskar, P Naidu, S Ravi, Chandra Babu, and P Govindarajulu. General principles of user interface design and websites. 08 2020.

[24] Bohyun Kim. Responsive web design, discoverability, and mobile challenge. *Library technology reports*, 49(6):29–39, 2013.

[25] Kate Moran. Usability testing 101. `https://www.nngroup.com/articles/usability-testing-101/`, Dec 2019.

Příloha A

Přiložené soubory

- Zdrojové soubory serverové aplikace.
- Zdrojové soubory chatovacího klienta.
- Exportovaná kolekce požadavků pro testování rozhraní pomocí aplikace Postman.