Aplikace pro podporu seniorů

Petr Buček

Bakalářská práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Petr Buček**
Osobní číslo:	**A19681**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Aplikace pro podporu seniorů**
Téma práce anglicky:	**Application to Support Seniors**

Zásady pro vypracování

1. Seznamte se s problematikou a vypracujte literární rešerši na dané téma.
2. Popište technologie, které budou v práci použité.
3. Proveďte analýzu požadavků pro podporu seniorů.
4. Navrhněte a realizujte aplikaci usnadňující zvládnutí problematických situací.
5. Zajistěte zabezpečení aplikace.
6. Proveďte testování aplikace a zhodnoťte provedené řešení.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. MALL, Rajib. *Fundamentals of software engineering*. PHI Learning Pvt. Ltd., 2018. ISBN 978-93-88028-03-5.
2. RAJLICH, Vaclav. *Software engineering: The current practice*. Chapman and Hall/CRC, 2019. ISBN 978-1-4665-1035-7.
3. DATE, Chris J. *Database design and relational theory: normal forms and all that jazz*. Apress, 2019. ISBN 978-1-4842-5540-7.
4. HARRINGTON, Jan L. *Relational database design and implementation*. Morgan Kaufmann, 2016. ISBN 978-0-12-804399-8.
5. ATER, Tal. *Building progressive web apps : bringing the power of native to the browser*. Sevastopol, CA. O'Reilly Media, 2017. ISBN 978-1-491-96165-0.
6. MUKHERJEA, Sougata. *Mobile application development, usability, and security*. Hershey, Pennsylvania. IGI Global, 2017. ISBN 978-1-522509462.

Vedoucí bakalářské práce: **doc. Ing. Zdenka Prokopová, CSc.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **3. prosince 2021**
Termín odevzdání bakalářské práce: **23. května 2022**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
T.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 24. ledna 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

Petr Buček, v.r.
podpis studenta

ABSTRAKT

Cílem této bakalářské práce bylo navrhnout a realizovat aplikaci na podporu seniorů. Aplikace je vytvořena jako progresivní webová aplikace, která je přístupná ze všech zařízení. Aplikace byla vytvořena pomocí Ionic a Angular Frameworků pro front-end a Google Firebase pro back-end aplikace. Teoretická část práce se zaměřuje na problematiku stáří a navrhuje, jak mohou aplikace pomoci. Dále jsou zkoumány existující aplikace, které pomáhají seniorům, následně jsou popsány technologie použité při vývoji. Praktická část pak obsahuje návrh a realizaci aplikace. Poslední kapitola se pak zabývá testováním hotové aplikace.

Klíčová slova: Aplikace pro podporu seniorů, PWA, Ionic Framework, Angular Framework, Firebase

ABSTRACT

Aim of this bachelor thesis was to design and implement an application to support seniors. Application is made as progressive web application, which is accessible from all devices. Application is made with Ionic and Angular Frameworks for front-end and Google Firebase for back-end of application. The theoretical part of the thesis focusses on issues of old age and suggests how application can help. Further there are already existing applications that helps seniors researched, then the technologies used for development are described. The practical part contains design and implementation of the application. Last chapter deals with the testing of finished application.

Keywords: Application to support seniors, PWA, Ionic Framework, Angular Framework, Firebase

OBSAH

ÚVOD

Čas je neúprosný a nikdo před ním neuteče. Každý člověk jednou bude starý. Dříve lehké úkony jsou čím dál tím těžší, tělo chřadne a mysl s ním. To vše s sebou přináší proces stárnutí. V této bakalářské práci se zaměříme na to, jak mohou aplikace pomoci starým lidem při vykonávání těchto různých úkonů a poté i takovou aplikace realizovat.

V teoretické části si rozebereme tuto problematiku a popíšeme si způsoby, jak je možné aplikacemi pomoci seniorům. Dále si projdeme již dostupná řešení a zhodnotíme je. Následně proběhne sběr a analýza požadavků na aplikaci, podle kterých bude v praktické části vytvořen návrh aplikace, který bude sestávat z případů užití a entitně relačního diagramu. V praktické části bude zmiňovaný návrh aplikace, popsaný samotný vývoj aplikace a popis zabezpečení aplikace. Praktická část bude zakončena testováním hotové aplikace.

Aplikaci jsem se rozhodl vytvořit za pomoci Ionic a Angular frameworků jakožto progresivní webovou aplikaci, kvůli dostupnosti ze všech zařízení. Aplikace bude uživateli umožňovat zapisovat své léky a události do kalendáře. Uživatel se přihlásí se svým emailem a heslem a bude mít vlastní záznamy v databázi. Aplikace pro databázový a autentizační back-end bude používat službu Firebase od společnosti Google. Závěrem práce bude popsáno zhodnocení řešení.

I. TEORETICKÁ ČÁST

1 STÁŘÍ

Stárnutí je nevyhnutelný proces, který postihuje všechny živé organismy bez výjimek. Tělo ztrácí na vitalitě a k tomu se přidávají nepříznivé psychické změny a deteriorace. Na stárnutí zatím léku není, existuje však několik způsobů, jak proti stárnutí bojovat a zmírnit tak jeho nepříznivé vlivy na člověka. V této rešerši se zaměříme, jakým způsobem by v téhle problematice mohli pomoci technologie, zejména tak, aplikace.

1.1 Technologie a senioři

Moderní technologie jako například chytré telefony jsou tu s námi teprve poslední dekády. Integrace těchto technologií do našich každodenních životů je spíše úspěšnější u mladších generací než u těch starších. V posledních letech se ale tato propast začíná zužovat. Podle výzkumu dostupného ze zdroje [1], je patrné, že zhruba každý pátý senior nad 65 let v České republice vlastní chytrý telefon. Výzkum je již tři roky starý a lze předpokládat, že se tento trend bude spíše rozmáhat než utlumovat. Jelikož data nejsou aktuální, nemusí odpovídat dnešní situaci, ale pomohou nám nastínit bod reference.

Tab. C1 Osoby v Česku používající mobilní telefon; 2019

%

	Celkem	chytrý telefon (smartphone)	tlačítkový telefon
Celkem (starší 16 let)	**96,9**	**69,6**	**31,3**
Celkem (16–74 let)	97,8	75,8	26,3
Pohlaví (starší 16 let)			
muži	96,6	71,3	29,4
ženy	97,1	68,0	33,0
Věk			
16–24	99,2	98,6	4,7
25–34	98,4	94,8	10,2
35–44	98,3	91,2	11,7
45–54	98,8	81,5	22,0
55–64	97,7	60,4	40,5
65+	91,6	19,8	73,7
Vzdělání (25–64 let)			
základní	87,8	53,6	34,5
střední bez maturity	98,8	73,6	29,3
střední s maturitou	98,7	88,3	16,0
vysokoškolské	99,8	95,5	9,8

podíl z celkového počtu osob v dané skupině

Obrázek 1. Výzkum prováděný Českým statistickým úřadem ze zdroje [1]

1.2 Problematické úkony ve stáří a jejich řešení pomocí aplikací

Stárnutí s sebou přináší mnohé komplikace a zvládnutí některých úkonů je pro stárnoucího člověka čím dál tím těžší záležitostí. Podle zdroje [2] je jeden z takových problémů pohybová obtíž. Podle profesorky paní MUDr. Eva Topinkové je *stárnutí doprovázeno postupným snižováním fyzické schopnosti a výkonnosti*. Co se týče aplikací, které s tímto problémem mohou pomoci, tak jen v tom rámci, že mohou seniorům poskytnout sadu cviků, které pomohou seniorům udržet fyzickou kondici, jak paní Eva Topinková ve svém článku navrhuje, aby senior měl dostatečnou fyzickou činnost.

Dalším z častých projevů stárnutí je deteriorace kognitivních funkcí. Podle pana Ing. Ladislava Martinka je *snížení paměti jednou z nejčastěji uváděných psychických změn u stárnoucích lidí* [3]. Proti tomuto problému se dá pomocí technologie bojovat hned několika způsoby. Jeden z nich je paměť trénovat pomocí her nebo nástrojů, při kterých paměť zapojujeme [4]. Další způsob je kupříkladu aplikace, která si dané věci pamatuje za nás a dává nám upozornění o událostech, které mají nastat.

Dalším projevem je zhoršení zrakového vnímaní, to se na aplikacích zaměřených na starší věkovou skupinu odráží v designu aplikace. Tyto aplikace z pravidla mají přehledné uživatelské prostředí, velké a přehledné písmo, větší tlačítka a obecně přehlednost aplikace je na prvním místě, co se týče návrhu designu.

2 DOSTUPNÁ ŘEŠENÍ

Aplikací, které mohou pomoci seniorům při zvládnutí mnoha obtíží, které stáří s sebou přináší je mnoho. Většina z nich spadá do těchto kategorií:

- Zjednodušení UI a lehčí spouštění nebo ovládání aplikací
- Připomínky kdy brát medikaci
- Zápis a analýza zdravotních údajů (např.: krevního tlaku)
- Rozvíjení a udržování kognitivních funkcí pomocí her nebo jiných nástrojů
- Předčítání textu nebo funkce lupy
- Informace o cvicích na udržení kondice

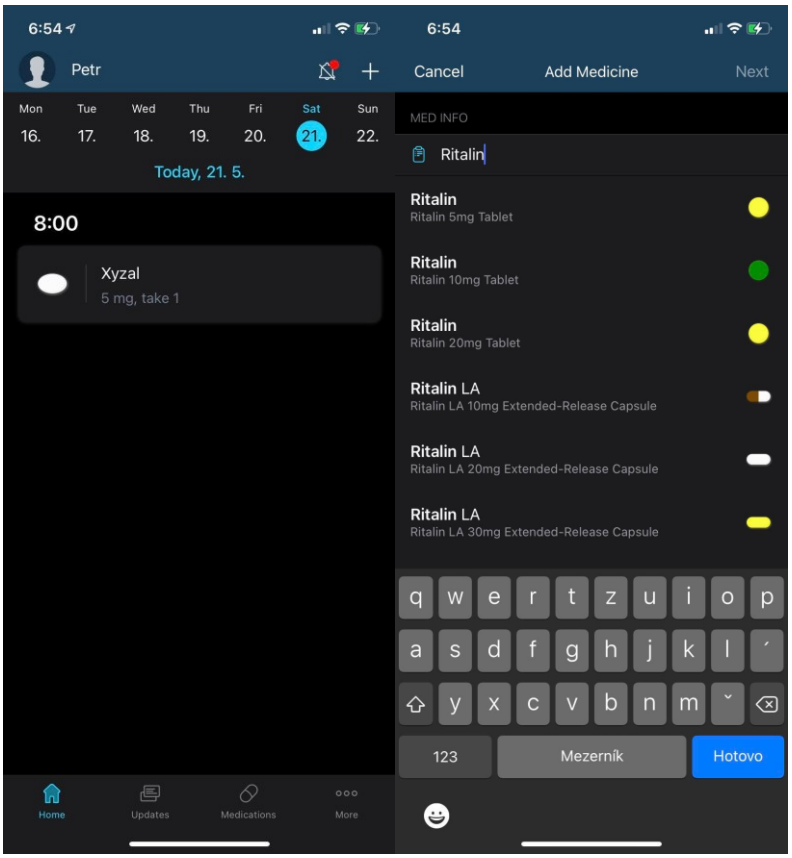
Jen hrstka z aplikací dostupných na trhu patří do více z těchto kategorií naráz, většinou se jedná o aplikace zaměřené na jeden typ problému.

Dále se budeme zaměřovat na aplikace, které evidují kdy a jaké léky uživatel bere. Tímto směrem jsem se rozhodl pojmout i tuto práci a aplikaci vyvíjet v podobném duchu.

Postupně si ukážeme některé z těchto aplikací a popíšeme si je.

2.1 Medisafe Pill Reminder [10]

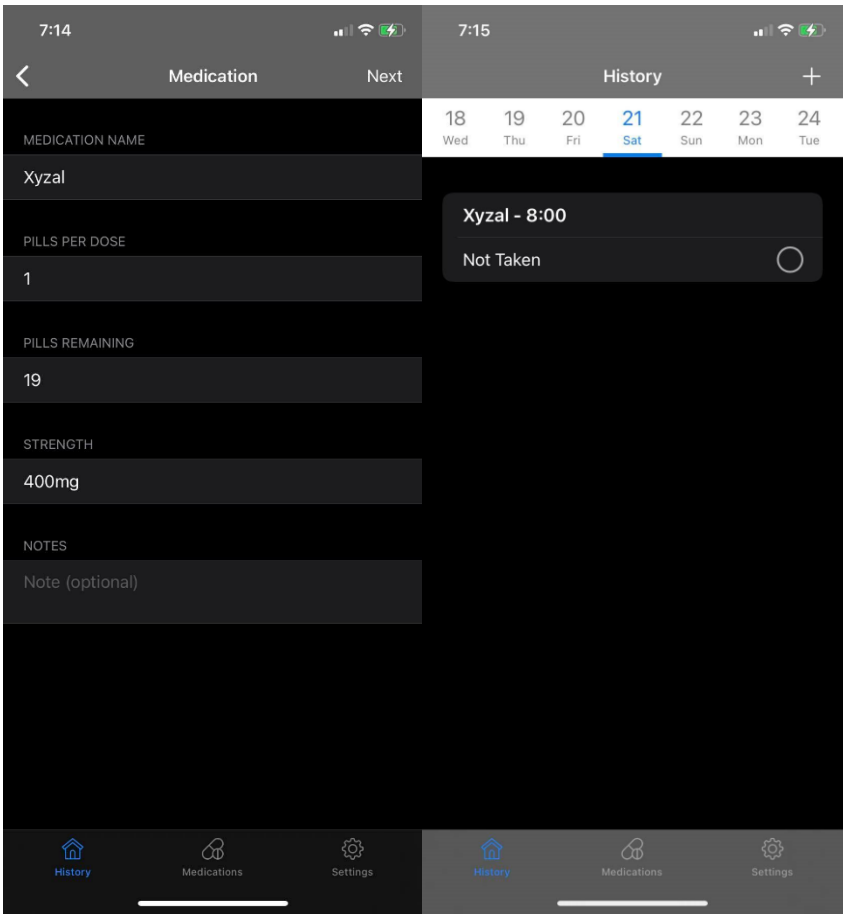
První aplikací dostupnou na platformách Android a iOS je Medisafe. Tato aplikace nabízí možnost vyhledání léku a přidání léku do kalendáře. U léku si uživatel může nastavit jakou dávku přijímá. Aplikace následně posílá uživateli oznámení o nadcházejícím léku. Navíc aplikace do aplikace může uživatel zadávat různé údaje jako například váha, krevní cukr, krevní tlak atd. Dále aplikace dovoluje uživateli, aby si vedl poznámky o svém zdraví ve formě diáře. Aplikace má přehledné uživatelské prostředí, ovšem písmo by pro staršího uživatele mohlo být špatně čitelné. Na platformě iOS má aplikace hodnocení 4.7 z 5 a velmi kladné slovní hodnocení.



Obrázek 2. Aplikace Medisafe Pill Reminder [10]

2.2 Pill Tracker+ [11]

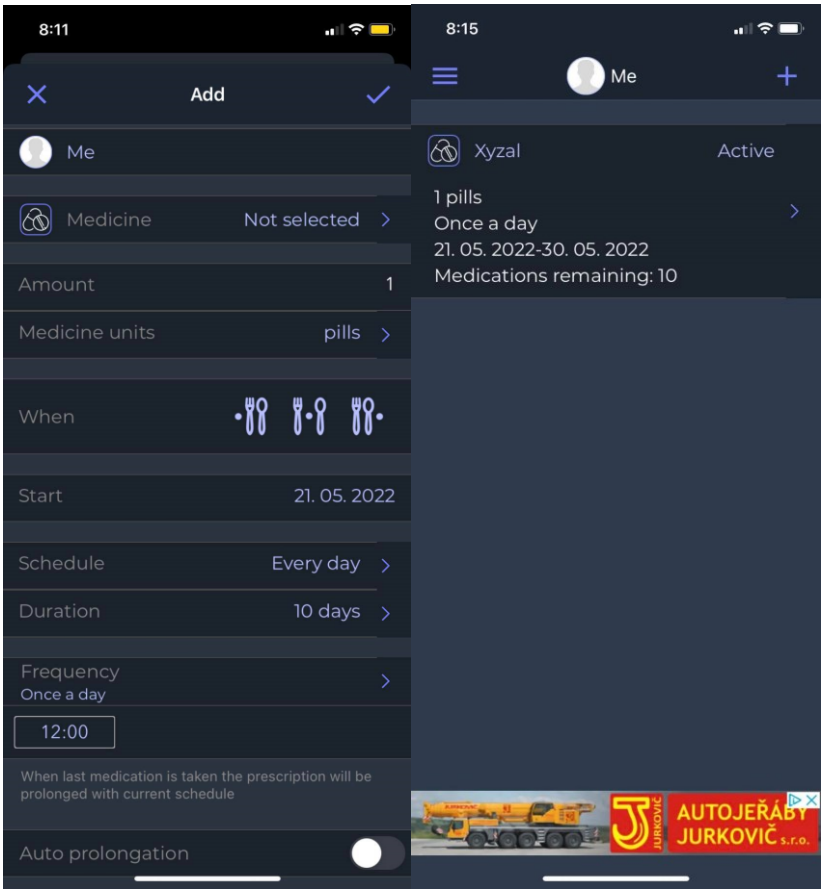
Další aplikace je dostupná pouze na zařízeních iOS. Tato aplikace oproti té předchozí má funkcionalitu pouze v evidenci léků. Při přidávání léku uživatel vyplní název, kolik prášků v jedné dávce uživatel bere, kolik léku zbývá v balení, dávku a popis. Léky lze editovat a mazat. Aplikace za uživatele počítá, kolik prášků v balení zbývá. Přehlednost aplikace je dobrá, ale jako v předchozím případě by písmo mohlo být větší. Na platformě iOS má aplikace hodnocení 3.8 z 5 a kladné slovní hodnocení.



Obrázek 3. Aplikace Pill Tracker+ [11]

2.3 Mr. Pillster [12]

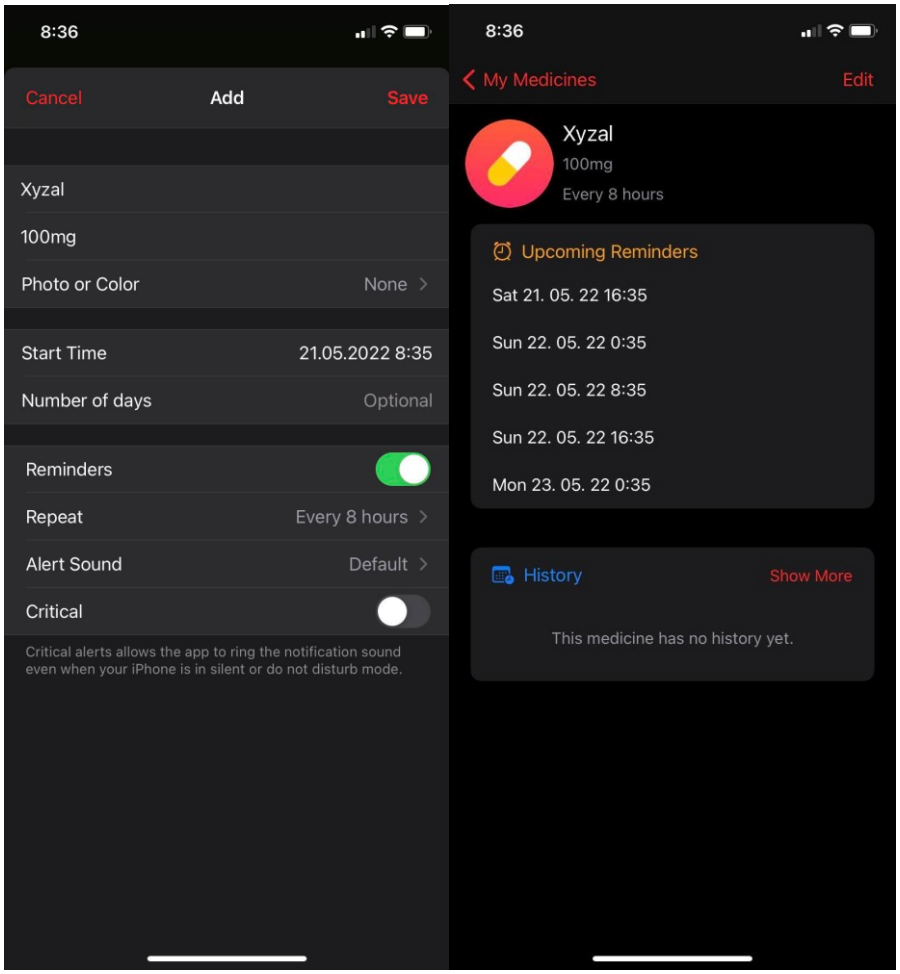
Další zkoumanou aplikací je aplikace Mr. Pillster dostupná na platformách Android a iOS. Aplikace při zadávání léků umožňuje zadat více informací než u předchozích aplikací. Uživatel zadá název léku a může přiložit i obrázek prášku, množství, typ léku, jestli se musí před požitím jíst a dále datum a čas opakovaní. Léky lze editovat. V aplikaci je možnost historie, kde je evidováno kdy, jaký lék byl podán. Přehled evidovaných léku je přehledný a obsahuje důležité informace. Přehlednost aplikace je na tom o něco hůře než u ostatních aplikací, zvláště pro mnoho informací na jednom místě, ovšem velikost písma je o něco větší a přispívá k celkové čitelnosti údajů na obrazovce. Na platformě iOS má aplikace hodnocení 4.8 z 5 a kladné slovní hodnocení.



Obrázek 4. Aplikace Mr. Pillster [12]

2.4 Pill Reminder [13]

Poslední zkoumanou aplikací bude Pill Reminder dostupný na platformě iOS. Tato aplikace nabízí v podstatě stejnou základní funkcionalitu jako předchozí aplikace. Lze si zapsat o jaký lék jde, možnost přidat fotku léku, možnost odlišit události s léky pomocí barev. Čím se aplikace odlišuje je možnost zaškrtnutí kritického léku, tato možnost umožní oznámením na zařízení, aby byly doprovázeny zvukovým upozorněním i když je zařízení ztišeno. Přehlednost aplikace je dostačující, až na písmo, které by mohlo být větší. Na platformě iOS má aplikace hodnocení 4.6 z 5.



Obrázek 5. Aplikace Pill Reminder [13]

3 SBĚR A ANALÝZA POŽADAVKŮ

Před samotným návrhem aplikace je kritický sběr požadavků na funkcionalitu aplikace. Tento sběr požadavků jsem realizoval rozhovory se svými prarodiči, kteří vlastní chytrý telefon a orientují se na něm, dále jsem prozkoumal již existující řešení dostupné na internetu.

Co se týče rozhovorů s prarodiči, nejčastějším problémem, se kterým se setkávali bylo zapomínání. Jednalo se o různé schůzky s doktory nebo zapomínání braní léků. Proto jsem se rozhodl pro aplikaci, která bude evidovat jaké léky senior bere. Jeden z hlavních požadavků na aplikaci byla přehlednost, dalším bylo, aby aplikace obsahovala pouze nutné funkce, v tom smyslu, aby seniora nemátla zbytečnými funkcemi navíc.

Dále popsané funkční a nefunkční požadavky jsou sepsány na základě sběru požadavků.

3.1 Funkční požadavky

- Uživatel se bude moci přihlásit/registrovat pomocí emailu a hesla
- Uživatel si bude schopen změnit heslo
- Uživatel bude mít profil se svými údaji, které bude moci měnit
- Uživatel bude moci přidávat léky a události do kalendáře
- Uživatel bude moci přidané léky a události měnit či mazat
-

3.2 Nefunkční požadavky

- Aplikace bude implementována jako PWA a bude přístupná ze všech zařízení
- Aplikace bude přehledná a intuitivní
- Aplikace bude využívat autentizaci uživatele
- Aplikace bude v českém jazyce

4 POPIS POUŽITÝCH TECHNOLOGIÍ

Účelem této kapitoly je seznámit čtenáře s použitými technologiemi a poskytnout kontext vývoje této aplikace. Kapitola se bude zabývat TypeScriptem, Angular frameworkem, který je založen na zmiňovaném TypeScriptu dále Ionic Frameworkem a Firebase od společnosti Google.

4.1 TypeScript

TypeScript je objektově orientovaný open-source programovací jazyk vyvíjený společností Microsoft. TypeScript je pouze nadstavbou jazyka JavaScript, tudíž pouze přidává určitou funkcionalitu [5]. To ve výsledku znamená, že jakýkoliv JavaScriptový kód je validním a spustitelným TypeScript kódem. Při spouštění se TypeScript kompiluje do klasického JavaScriptu. TypeScript podporuje jak dynamické, tak statické typování. Oproti JavaScriptu přivádí funkcionalitu rozhraní, generiky, statické či silné typování. Nevýhodou je mu ale však samotná kompilace do JavaScriptu [6].

4.2 Angular Framework

Angular je open-source framework webových aplikací založený na TypeScriptu. Je vyvíjen týmem Angular Team patřící pod společnost Google. V dnešní době se jedná o oblíbený nástroj vývojářů, kteří pomocí jej tvoří uživatelské rozhraní. Angular je komponentově zaměřený framework pomocí kterého se dají vytvářet škálovatelné webové aplikace [7].

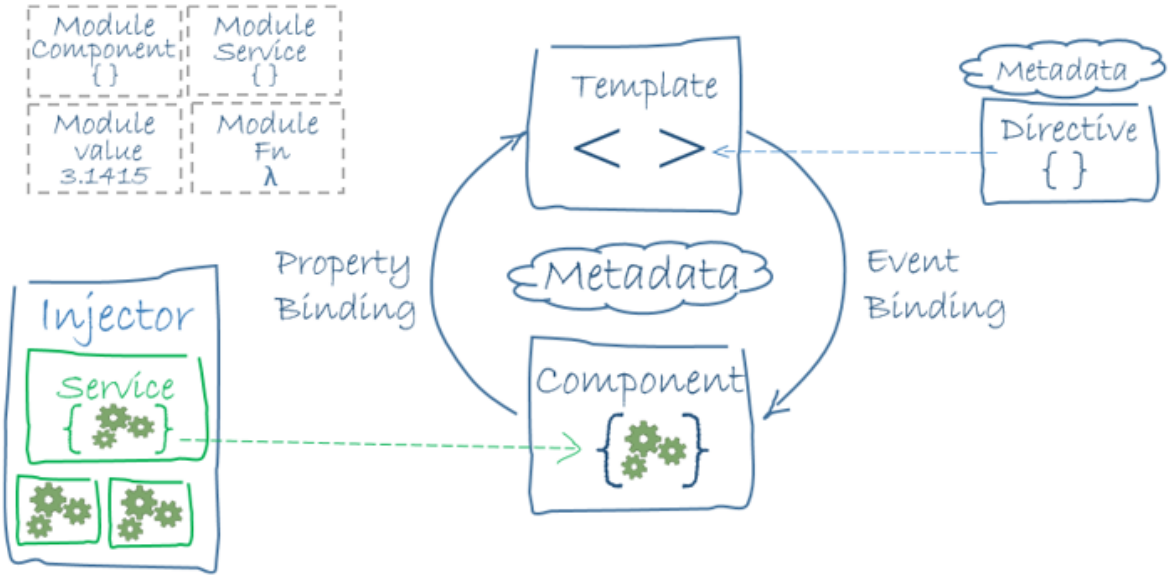
4.2.1 Historie Angularu [8]

Historie Angularu začíná u vývojáře společnosti Google Miško Heveryho, který pracoval na svém osobním projektu, který mu měl pomoci vytvářet webové aplikace pro jiné projekty. Z tohoto projektu se poté stal AngularJS, což je zárodek Angular frameworku. AngularJS používá HTML jako šablonu, ve které jsou vloženy vlastní HTML atributy, které poté AngularJS navazuje na model reprezentován proměnnými JavaScriptu. Po vydání v roce 2010 se AngularJS stal oblíbeným nástrojem pro vytváření dynamických webových aplikací. Po letech od prvního vydání, se standarty JavaScriptu posunuly a

AngularJS začínal narážet na své hranice. Z tohoto důvodu se tým společnosti Google rozhodl celý AngularJS přepsat a postavit plnohodnotný framework, který bude schopen umožňovat vytvářet škálovatelné mezi-platformové aplikace. Angular byl přepsán v TypeScriptu a nyní podporoval vývoj na mobilní zařízení. Architektonický styl nového Angularu je založen na komponentech a dostal integrovaný kompilátor, kvůli TypeScriptu, který musí být kompilován na JavaScript.

4.2.2 Architektura Angular frameworku [9]

Základními stavebními prvky Angular framewroku jsou komponenty NgModuly. Aplikace má vždy základní *root module*, který umožňuje bootstrapping, což je technika kdy jednodušší program aktivuje složitější systém programů. Komponenty definují *views*, obrazové prvky aplikace, které se mění podle programové logiky a dat aplikace. Komponenty dále používají *services*, které poskytují funkcionalitu, která není napřímo související s views. Komponenty aplikace většinou definují několik *views*, které jsou hierarchicky seřazeny. Jedna ze *services* Angularu je Router, který definuje navigační cesty mezi různými *views*. Komponenty používají `@Component()` dekorátor identifikující třídu pod sebou jako komponentu. Následující obrázek ukazuje, jak jsou stavební prvky provázány.



Obrázek 6. Diagram ukazující strukturu Angularu ze zdroje [9]

4.2.3 Šablony a direktivy

Šablona kombinuje HTML s označením Angularu, které dokáže měnit HTML prvky ještě předtím, než jsou zobrazeny. Direktivy šablony zajišťují programovou logiku a označovací vazby propojují data aplikace s DOM. Existují dva typy datové vazby: Event binding, který zajišťuje reagování aplikace na vstup uživatele a Property binding, který převádí hodnoty, které jsou vypočítávány v aplikaci na data do HTML. Předtím než je *view* zobrazen Angular vyhodnotí direktivy a vyřeší vázací syntaxi v šabloně, aby mohl pozměnit HTML prvky podle programové logiky aplikace. Angular také podporuje dvoj směrovou vazbu dat, což znamená, že změny v DOM, kupříkladu vstup uživatele, se projevují v datech programu [9].

4.2.4 Služby a dependency Injection

Aby logika či data byly přivázány k danému *view*, musí být sdíleny napříč komponentami, to zajišťuje třída služeb, tu předchází dekorátor `@Injectable()`. Tento dekorátor zajišťuje, aby mohli být jiné služby injektovány jako závislosti do jiných tříd [9].

4.2.5 Routing

NgModul Router Angularu dovoluje definovat navigační cestu napříč různými stavy či *views* aplikace a definuje hierarchii aplikace. Router modul je modelován podle konvencí navigací v prohlížeči, to znamená: zadání URL adresy do lišty prohlížeče, rozkliknutí odkazů na stránce nebo tlačítka zpět a dopředu prohlížeče. Když například uživatel klikne na tlačítko zpět Router modul zakročí a namísto obvyklého chování prohlížeče akorát zobrazí nebo schová dané hierarchie *views* [9].

4.3 Ionic Framework

Ionic Framework je rámec uživatelského prostředí používán k vývoji hybridních mobilních aplikací. Využívá technologie HTML, CSS a Javascript s integrací frameworků Angularu, Reactu nebo Vue. Jedná se tedy o kombinaci několika technologií, kde Ionic Framework leží ve vrchní vrstvě a slouží jako vrstva uživatelského prostředí [14].

4.3.1 Hybridní aplikace

Jak již bylo zmíněno, pomocí Ionicu jsou vytvářeny takzvané hybridní mobilní aplikace. Tyto aplikace používají WebView, což je objekt, který umožňuje zobrazit obsah webu jako část rozložení [15]. Ve výsledku může být aplikace vydána na více platformách zaráz. Aplikace, ovšem může mít horší výkon u starších zařízení, kvůli jejich zastaralým verzím webových prohlížečů [14]. Oproti nativním aplikacím mají hybridní aplikace většinou rychlejší sestavení. Ovšem jejich vývoj může být náročnější kvůli nutnosti testování aplikace na různých platformách, to stejné platí o designu aplikace, kde se design na různých platformách může lišit [16].

4.4 Firebase

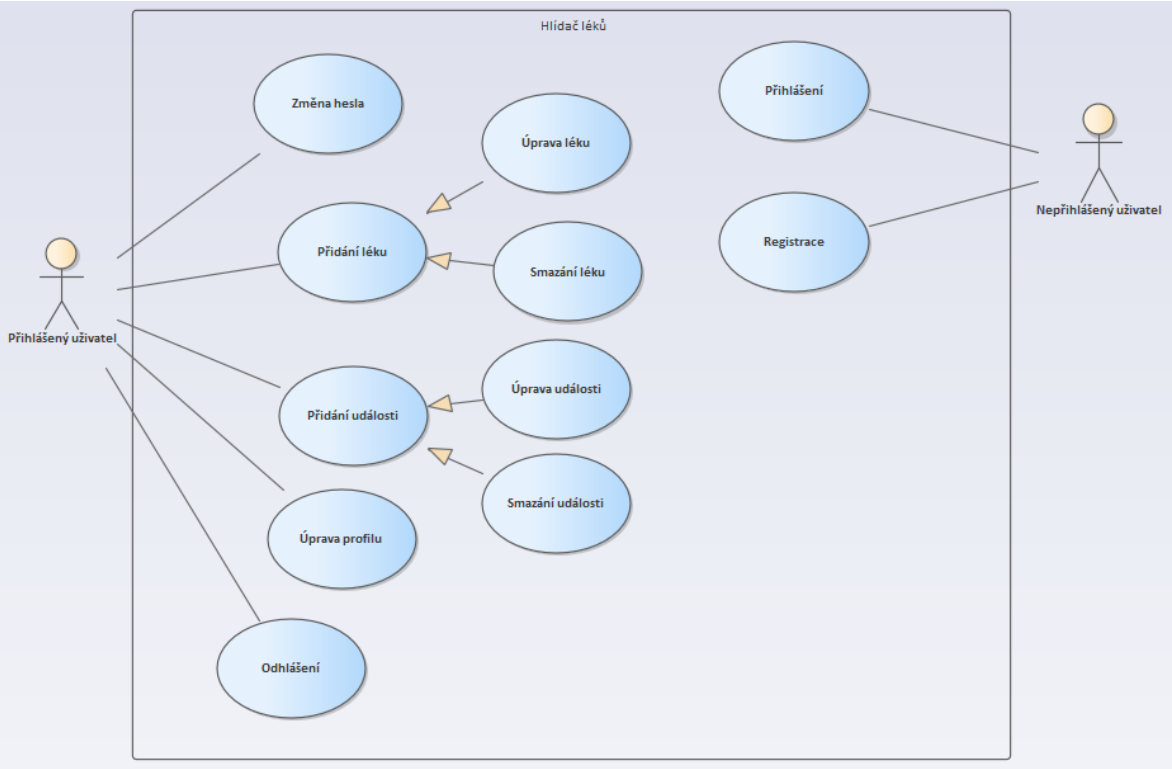
Firebase od společnosti Google je back-endová služba, která umožňuje funckionalitu pomocí několika nástrojů pro bezpečnou komunikaci front-endu s back-end službou. *Firebase je klasifikován jako NoSQL databázový program, který ukládá data v JSON dokumentech* [17]. Mezi služby, které firebase nabízí patří: Autentizace pomocí emailu a hesla, Googlu, Facebooku nebo například telefonního čísla, dále databáze, hosting a notifikace a další [17]. Tato služba se dá propojit se širokou škálou projektů kde zajišťuje již zmíněné funkcionality.

II. PRAKTICKÁ ČÁST

5 NÁVRH APLIKACE

Návrh aplikace proběhne na základě sběru požadavků. Diagram případů užití je sestaven na základě funkčních a nefunkčních požadavků. Dále bude sestaven ERD databázový model pro aplikaci.

5.1 Případy užití



Obrázek 7. Use case model

Diagram případů užití na obrázku č.7 má dva aktéry: přihlášeného a nepřihlášeného uživatele. Nepřihlášený uživatel má pouze dva případy užití, a to buď se přihlásit nebo registrovat. Přihlášený uživatel může změnit své heslo, upravit profil, přidat, měnit a mazat události a léky v kalendáři a odhlásit se.

5.2 Scénáře případů užití

Název: Registrace		
ID: UC001		
Charakteristika: Uživatel se registruje pomocí emailu a hesla		
Aktér: Nepřihlášený uživatel		
Vstupní podmínky: Nejsou		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Aktér	Vyplní email a heslo
2	Aktér	Stiskne tlačítko registrovat
3	Aktér	Je registrován a přesměrován na domovskou stránku
Alternativní scénáře:		
1	Aktér	Vyplní nesprávné údaje
2	Systém	Zobrazí chybový dialog

Tabulka 1. Scénář případů užití UC001

Název: Registrace		
ID: UC002		
Charakteristika: Uživatel se přihlásí pomocí emailu a hesla		
Aktér: Nepřihlášený uživatel		
Vstupní podmínky: Nejsou		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Aktér	Vyplní email a heslo
2	Aktér	Stiskne tlačítko přihlásit
3	Aktér	Je přihlášen a přesměrován na domovskou stránku
Alternativní scénáře:		
1	Aktér	Vyplní nesprávné údaje
2	Systém	Zobrazí chybový dialog

Tabulka 2. Scénář případů užití UC002

Název: Změna hesla		
ID: UC003		
Charakteristika: Uživatel si změní přihlašovací heslo		
Aktér: Přihlášený uživatel		
Vstupní podmínky: Uživatel je přihlášen		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Aktér	Vyplní nové a staré heslo
2	Aktér	Stiskne tlačítko změnit
3	Systém	Změní heslo a zobrazí dialog o úspěšné operaci
Alternativní scénáře:		
1	Aktér	Vyplní nesprávné údaje
2	Systém	Zobrazí chybový dialog

Tabulka 3. Scénář případů užití UC003

Název: Úprava profilu		
ID: UC004		
Charakteristika: Uživatel si změní údaje o profilu		
Aktér: Přihlášený uživatel		
Vstupní podmínky: Uživatel je přihlášen		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Aktér	Vyplní požadované údaje
2	Aktér	Stiskne tlačítko změnit profil
3	Systém	Zapíše nové údaje do databáze
Alternativní scénáře:		

Tabulka 4. Scénář případů užití UC004

Název: Přidání léku		
ID: UC005		
Charakteristika: Uživatel přidá lék do kalendáře		
Aktér: Přihlášený uživatel		
Vstupní podmínky: Uživatel je přihlášen		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Aktér	Stiskne tlačítko přidat lék
2	Aktér	Vyplní údaje o léku
3	Aktér	Stiskne potvrzovací tlačítko
4	Systém	Zapíše lék do databáze a aktualizuje kalendář
Alternativní scénáře:		

Tabulka 5. Scénář případů užití UC005

Název: Úprava léku		
ID: UC006		
Charakteristika: Uživatel upraví lék v kalendáři		
Aktér: Přihlášený uživatel		
Vstupní podmínky: Uživatel je přihlášen		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Aktér	Stiskne tlačítko upravit lék
2	Aktér	Vyplní nové údaje o léku
3	Aktér	Stiskne potvrzovací tlačítko
4	Systém	Zapíše lék do databáze a aktualizuje kalendář
Alternativní scénáře:		

Tabulka 6. Scénář případů užití UC006

Název: Smazání léku		
ID: UC007		
Charakteristika: Uživatel smaže lék z kalendáře		
Aktér: Přihlášený uživatel		
Vstupní podmínky: Uživatel je přihlášen		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Aktér	Naviguje na stránku s odebráním léku
2	Aktér	Vybere lék
3	Aktér	Stiskne potvrzovací tlačítko
4	Systém	Odebere lék z databáze a aktualizuje kalendář
Alternativní scénáře:		

Tabulka 7. Scénář případů užití UC007

Název: Přidání události		
ID: UC008		
Charakteristika: Uživatel přidá událost do kalendáře		
Aktér: Přihlášený uživatel		
Vstupní podmínky: Uživatel je přihlášen		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Aktér	Stiskne tlačítko přidat událost
2	Aktér	Vyplní údaje o události
3	Aktér	Stiskne potvrzovací tlačítko
4	Systém	Zapíše událost do databáze a aktualizuje kalendář
Alternativní scénáře:		

Tabulka 8. Scénář případů užití UC008

Název: Úprava události		
ID: UC009		
Charakteristika: Uživatel upraví událost v kalendáři		
Aktér: Přihlášený uživatel		
Vstupní podmínky: Uživatel je přihlášen		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Aktér	Stiskne tlačítko upravit událost
2	Aktér	Vyplní nové údaje o události
3	Aktér	Stiskne potvrzovací tlačítko
4	Systém	Zapíše událost do databáze a aktualizuje kalendář
Alternativní scénáře:		

Tabulka 9. Scénář případů užití UC009

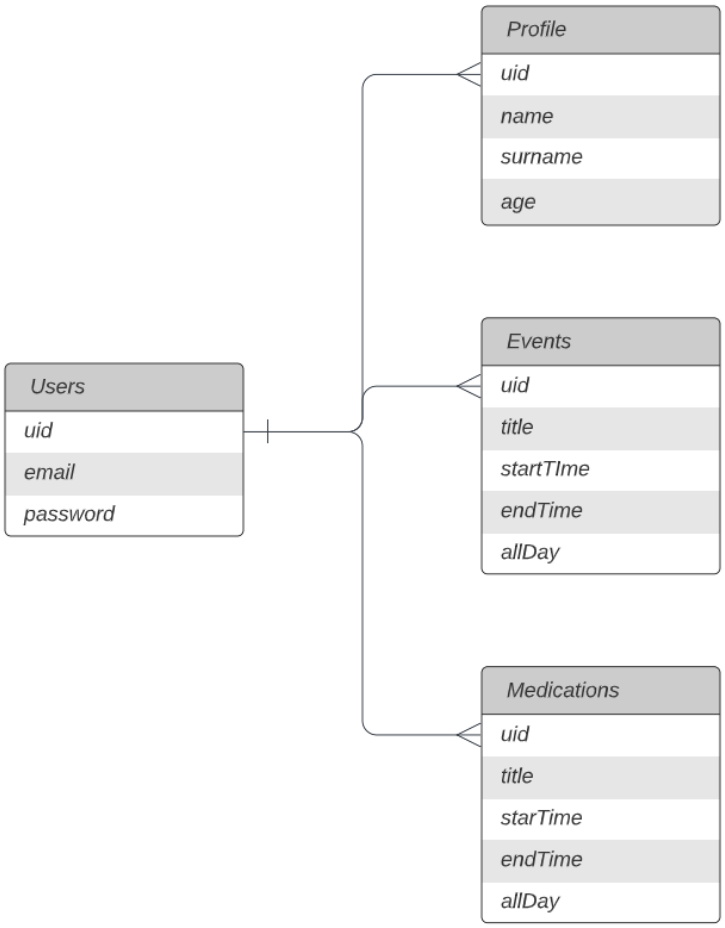
Název: Smazání události		
ID: UC010		
Charakteristika: Uživatel smaže událost z kalendáře		
Aktér: Přihlášený uživatel		
Vstupní podmínky: Uživatel je přihlášen		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Aktér	Naviguje na stránku s odebráním události
2	Aktér	Vybere událost
3	Aktér	Stiskne potvrzovací tlačítko
4	Systém	Odebere událost z databáze a aktualizuje kalendář
Alternativní scénáře:		

Tabulka 10. Scénář případů užití UC010

Název: Odhlášení		
ID: UC011		
Charakteristika: Uživatel se odhlásí		
Aktér: Přihlášený uživatel		
Vstupní podmínky: Uživatel je přihlášen		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Aktér	Stiskne tlačítko odhlášení
2	Systém	Odhlásí uživatele a přesměruje na úvodní stránku
Alternativní scénáře:		

Tabulka 11. Scénář případů užití UC011

5.3 Databázový model



Obrázek 8. ERD databázový model

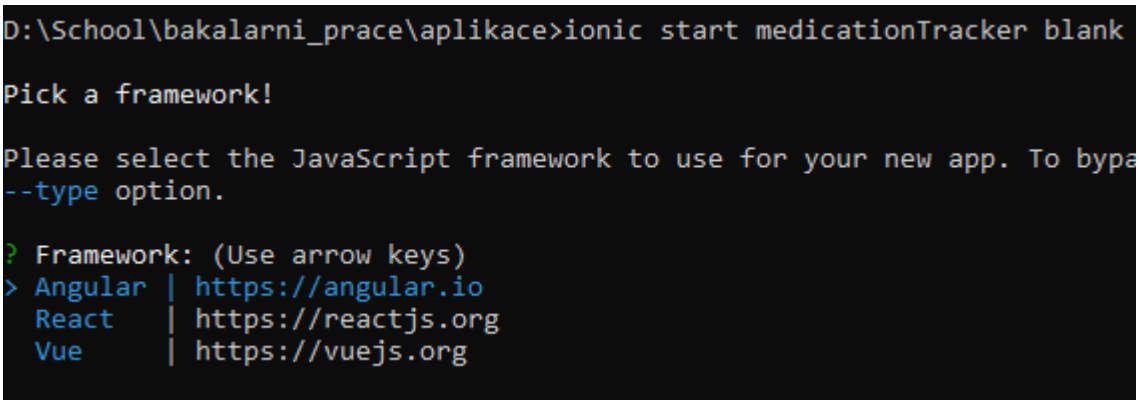
V databázovém modelu jsou 4 tabulky, jedna pro uživatele, která drží přihlašovací údaje. Další tabulky jsou na tabulku s uživateli napojeny pomocí ID uživatele. Tabulka Profile drží informace o uživateli, tabulka Events drží informace o událostech a poslední tabulka Medications drží informace o lécích, které jsou vypisovány do kalendáře.

6 REALIZACE APLIKACE

V této kapitole bude popsán postup tvorby aplikace. Aplikace byla vyvíjena pomocí Ionic a Angular frameworku.

6.1 Konfigurace projektu

Pomocí příkazu `$ ionic start medicationTracker blank` se inicializoval projekt. Při tázání, jaký JavaScriptový framework byl vybrán Angular.



Obrázek 9. Inicializace Ionic projektu

Po této volbě se v dané složce vytvořila složka medicationTracker, ve které je obsažena kostra aplikace.

Na příloze P I. můžeme vidět souborovou strukturu Angular projektu. Ve složce src sídlí většina aplikace, tato složka obsahuje téměř všechny hlavní soubory s kódy aplikace. Složka app obsahuje soubory, které obsahují komponenty aplikace. Ve složce home se nachází soubory k příslušné stránce aplikace. Tato složka obsahuje hlavní html soubor, který je šablonou, scss je soubor se styly. Soubor app.component.ts obsahuje komponenty, jedná se o nejdůležitější soubor s logikou komponentů, dále app.module.ts obsahuje všechny závislosti, definuje jaké moduly mají být načteny. Soubor app-routing.module.ts modul, zajišťuje routing mezi jednotlivými stránky aplikace. Složka e2e obsahuje end-to-end testovací soubory. Angular.json soubor je konfigurační soubor, který definuje strukturu aplikace, specifikují se zde prostředí. Složka assets drží například obrázky a další zdrojové soubory aplikace. Složka environments obsahuje konfigurační konstanty prostředí, se kterými je aplikace tvořena. Ty jsou definovány ve 2 souborech: environments.ts a environment.prod.ts, kde jsou tyto konstanty užity v souboru angular.json.

6.2 Kalendář s událostmi

V této podkapitole si popíšeme tvorbu kalendáře s událostmi. Pro moji implementaci jsem použil komponentu ionicu ionic2-calendar. Instalace tohoto modulu proběhla pomocí příkazu `$ npm install ionic2-calendar.` Dále bylo potřeba v home.module.ts definovat, že bude tato komponenta použita pomocí kódu `import { NgCalendarModule } from 'ionic2-calendar'.`

6.2.1 Vzhled a definování kalendáře

```
<ng-template #myTemplate let-view="view" let-row="row" let-col="col">
  <div class="ion-margin-md" [class.with-event]="view.dates[row*7+col].events.length">
    {{view.dates[row*7+col].label}}
    <div class="indicator-container">
      <div class=" indicator" *ngFor="let e of view.dates[row*7+col].events"></div>
    </div>
  </div>
</ng-template>

<calendar [monthviewDisplayEventTemplate]="myTemplate" [eventSource]="eventSource" [startingDayMonth]="1"
  [currentDate]="calendar.currentDate" (onTitleChanged)="onViewTitleChanged($event)">
</calendar>
```

Obrázek 10. Ukázka kódu kalendáře

V samotném home.page.html souboru, který je šablonou pro hlavní stránku aplikace bylo zapotřebí definovat pomocí tagů `<calendar></calendar>` kalendář. Parametry u tohoto tagu definují, že kalendář bude začínat od pondělí, nastavují šablonu, podle které bude kalendář vykreslen, současné datum, zdroj událostí a funkci, která se zavolá, když se změní název měsíce.

Samotná šablona uvedená tagem výše nám definuje, že bude vykresleno několik řad s daty po 7 a u každého data bude indikátor kolik událostí u daného dne je.

6.2.2 Zápis událostí do kalendáře

Do kalendáře budeme ukládat dva typy událostí: léky, které se budou v daný čas periodicky opakovat ve vybraných dnech a události, které budou jednorázové. V kódu je proměnná `eventSource,` do které se budou zapisovat všechny události, tato proměnná je vázána na kalendář definovaný na úvodní html stránce a drží všechny události, které jsou následně v kalendáři zobrazeny.

```
addMedication(title: string, hour: number, min: number, day: number) {
  var date = new Date()
  date.setDate(date.getDate() + ((7 - date.getDay()) % 7 + day) % 7)

  for (var i = 0; i < 365; i += 7) {
    var time = new Date(
      date.getFullYear(),
      date.getMonth(),
      date.getDate() + i,
      hour,
      min
    )
    var event = ({
      title: 'Lék: ' + title,
      startTime: time,
      endTime: time,
      allDay: false,
    })
    this.db.collection(`users/${this.getUserId()}/medications`).add(event)
  }
}

addEvent(title: string, date: Date) {
  var event = ({
    title: 'Událost: ' + title,
    startTime: date,
    endTime: date,
    allDay: false
  })
  this.db.collection(`users/${this.getUserId()}/events`).add(event)
}
```

Obrázek 11. Ukázka kódu zápisu událostí do databáze

Na obrázku č.11 máme dvě funkce, které přidávají události do databáze. Čtení z databáze a následné vkládání do proměnné `eventSource` si ukážeme v následující kapitole. První funkce, která vytváří opakovanou událost s léky přijímá název, hodinu a minutu kdy má událost léku probíhat, vstupní proměnná `day` slouží k určení, ve který den se má událost opakovat, může nabývat hodnot od 1 do 7 a vždy koreluje s daným dnem v týdnu, kdy pondělí je 1 a neděle 7. Funkce `date.setDate(date.getDate() + ((7 - date.getDay()) % 7 + day) % 7)` určuje počáteční den podle vstupní proměnné `day`. V cyklu, který simuluje opakování po 7 dnech po jeden celý rok se poté vytváří proměnná s datem události a ta je následně použita při vytvoření proměnné samotné události. Nakonec je každá z vytvořených událostí zapsána do databáze.

Druhá funkce je jednodušší a pouze používá vstupní proměnné, aby vytvořila událost a tu přímo vloží do databáze.

6.2.3 Modální dialogy

Pro uživatelský vstup na vytvoření události aplikace využívá modálních dialogů. Tyto modální dialogy se vykreslí před stránkou, na které se uživatel zrovna nachází a po zmizení dokáží předat změnu nebo instanci dat, která v nich vznikla.

```
async openEventModal() {
  const modal = await this.modalContrl.create({
    component: EventModalPage,
    backdropDismiss: false
  });

  await modal.present()

  modal.onDidDismiss().then((result) => {
    var date

    if (result.data) {
      if (!result.data.date) {
        date = new Date
      } else {
        date = new Date(
          Number(format(parseISO(result.data.date), 'yyyy')),
          Number(format(parseISO(result.data.date), 'M')) - 1,
          Number(format(parseISO(result.data.date), 'd')),
          Number(format(parseISO(result.data.date), 'HH')),
          Number(format(parseISO(result.data.date), 'mm'))
        )

      }
      this.addEvent(result.data.title, date)
    }
  })
}
```

Obrázek 12. Ukázka kódu na tvorbu modálního dialogu

Na obrázku č.12 máme příklad funkce, která vytváří modální dialog pomocí ModalController komponenty. Funkce musí být asynchronní, protože se čeká na obsluhu dialogu uživatelem. Šipková funkce `modal.onDidDismiss().then((result) => {})` po zavření dialogu v proměnné `result` nese data, které uživatel zadal. Obrázek č.12 obsahuje příklad logiky modálního dialogu, kde máme název a datum, které po zavření předáváme.

```
title = ''
date

close() {
  this.modalContrl.dismiss()
}

save() {
  this.modalContrl.dismiss({ title: this.title, date: this.date })
}
```

Obrázek 13. Ukázka kódu logiky modálního dialogu

6.2.4 Editace událostí v kalendáři

Pro funkcionalitu editace událostí v kalendáři jsou vytvořeny dvě samostatné modální dialogy, jeden pro léky a jeden pro události.

```
deleteMed(title: string) {
  this.dbService.deleteMedByTitle(title)
  this.close()
}

editMed(title: string) {
  this.openCalendarModal(title)
}

async openCalendarModal(title: string) {
  const modal = await this.modalContrl.create({
    component: CalendarModalPage,
    backdropDismiss: false
  })

  await modal.present()

  modal.onDidDismiss().then((result) => {
    if (result.data) {
      var hour, min
      if (!result.data.time) {
        var date = new Date
        hour = date.getHours()
        min = date.getMinutes()
      }
      else {
        hour = format(parseISO(result.data.time), 'HH')
        min = format(parseISO(result.data.time), 'mm')
      }
      this.dbService.deleteMedByTitle(title)
      this.modalContrl.dismiss({ hour: hour, min: min, event: result.data.event })
    }
  })
```

Obrázek 14. Ukázka kódu pro editaci léku v kalendáři

Na obrázku č.14 je příklad kódu pro editaci léku v kalendáři. Editace je řešena způsobem, že u daného léku má uživatel dvě možnosti, buď jej smazat nebo jej editovat. Pokud se rozhodne uživatel údaj odstranit, zavolá se funkce na smazání léku podle názvu předávaného v argumentu funkce a modální dialog se zavře. Pokud se uživatel rozhodne údaj editovat, vytvoří se nový modální dialog, který je totožný s vytvořením události s lékem. Staré záznamy se z databáze vymažou a nahradí je údaje nové, které jsou specifikovány uživatelem z modálního dialogu.

6.3 Databáze

Jako databázový back-end aplikace používá službu Firebase od společnosti Google. To je umožněno pomocí sady firebase pluginů pro Angular. V databázi má každý uživatel vlastní kolekci dat pod svým uživatelským ID, to zajišťuje, že se uživatel dostane pouze ke svým uloženým datům.

6.3.1 Čtení dat z databáze

```
this.db.collection(`users/${this.getUserId()}/events`).snapshotChanges().subscribe(snapEvent => {
  this.db.collection(`users/${this.getUserId()}/medications`).snapshotChanges().subscribe(snapMed => {
    this.eventSource = []
    snapEvent.forEach(snapshot => {
      var event: any = snapshot.payload.doc.data()
      event.id = snapshot.payload.doc.id
      event.startTime = event.startTime.toDate()
      event.endTime = event.endTime.toDate()
      this.eventSource.push(event)
      this.tracker.loadEvents()
    })
    snapMed.forEach(snapshot => {
      var med: any = snapshot.payload.doc.data()
      med.id = snapshot.payload.doc.id
      med.startTime = med.startTime.toDate()
      med.endTime = med.endTime.toDate()
      this.eventSource.push(med)
      this.tracker.loadEvents()
    })
  })
})
```

Obrázek 15. Ukázka kódu pro čtení dat z databáze

Na obrázku č.15 máme příklad čtení obou typů událostí z databáze. Jsou zde použity dvě do sebe vnořené funkce, vnoření je důležité, protože na začátku zápisu událostí je potřeba vymazat všechny předchozí události z proměnné `eventSource`, aby nedocházelo k přidání již přidaných událostí, kdyby funkce nebyly vnořeny do sebe vymazání by muselo proběhnout dvakrát a data jedné z funkcí by byly vymazány.

Každá funkce vytvoří snapshot databáze a pomocí šipkové funkce se dostáváme k uloženým datům. V každé šipkové funkci procházíme data pomocí funkce `forEach()` následně vytváříme proměnnou do které předáváme uložená data z databáze, tuto proměnou poté přidáváme do `eventSource`. Kód `this.tracker.loadEvents()` následně aktualizuje kalendář a zobrazí načtené události.

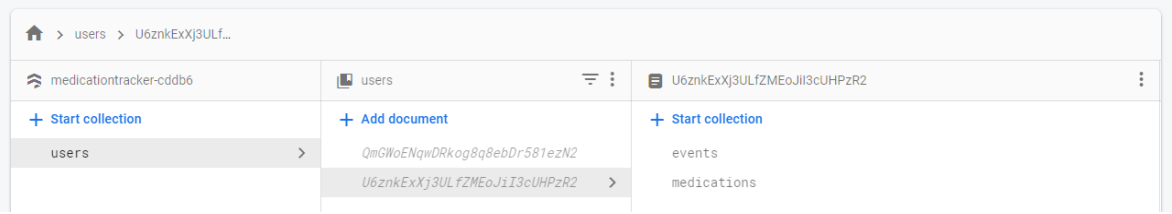
6.3.2 Mazání dat v databázi

```
deleteEventByTitle(title: string) {
  var collection = this.db.collection(`users/${this.getUserId()}/events`)
  this.db.collection(`users/${this.getUserId()}/events`).snapshotChanges().subscribe(snapMed => {
    snapMed.forEach(async snapshot => {
      var event: any = snapshot.payload.doc.data()
      if (event.title == title) {
        await collection.doc(snapshot.payload.doc.id).delete()
      }
    })
  })
}
```

Obrázek 16. Ukázka kódu pro mazání událostí z databáze

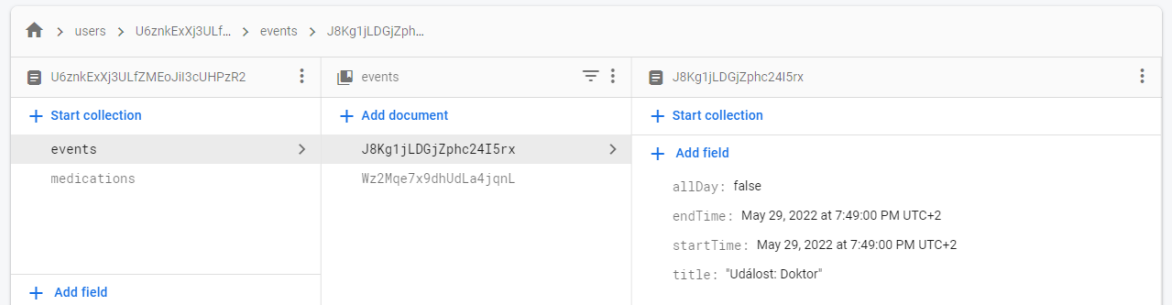
V této ukázce kódu je ukázán příklad mazaní událostí z databáze pomocí názvu události. Postup je obdobný ke čtení dat z databáze. Opět se vytvoří snapshot databáze a prochází se každý údaj zvlášť, následně se porovnává název události v databázi s předaným názvem ve vstupní proměnné funkce. Pokud název je stejný zavolá se funkce na vymazání údaje z databáze, do které předáváme ID údaje, který chceme smazat. Tato šipková funkce musí být asynchronní.

6.3.3 Struktura databáze



Obrázek 17. Ukázka struktury databáze 1

Databáze je strukturována tak, že každý uživatel má svoji kolekci dat pod svým uživatelským ID v kolekci *users* obr.č.17. V každé kolekci uživatele jsou poté kolekce s názvem *events* a *medications*. Ty obsahují data událostí nebo léků pod svým ID obr.č.18.



Obrázek 18. Ukázka struktury databáze 2

6.4 Autentizace uživatele

```
async registration() {
  try {
    const user = await this.authService.register(this.creds.value)
    if (user) {
      this.router.navigateByUrl('/home');
    }
  } catch (error) {
    this.alert('Registrace nebyla úspěšná', 'Zkuste to znovu')
  }
}

async login() {
  try {
    const user = await this.authService.login(this.creds.value)
    if (user) {
      this.router.navigateByUrl('/home');
    }
  } catch (error) {
    this.alert('Přihlášení nebylo úspěšné', 'Zkuste to znovu')
  }
}

async alert(header, message) {
  var alert = await this.alertContrl.create({
    header,
    message,
    buttons: ['Dobře']
  })
  await alert.present()
}
```

Obrázek 19. Ukázka kódu pro autentizaci uživatele

Pro autentizaci uživatele aplikace opět užívá back-endu služby Firebase. Uživatel si vytvoří vlastní účet pomocí emailu a hesla. Aplikace využívá AuthGuard z balíčku pluginů firebase pro Angular, který zajišťuje, že se nepřihlášený uživatel nedostane na jinam než na stránku s přihlášením. Dále je použitý plugin AuthenticationService, který zajišťuje bezpečnou registraci a přihlášení uživatele.

Na obrázku výše jsou funkce pro registraci a přihlášení uživatele. V obou funkcích je blok try catch, kdy je po neúspěšné registraci či přihlášení zobrazeno oznámení s hláškou nezdařené operace. Správnost údajů a vydaření přihlášení řeší plugin AuthenticationService. Pokud je registrace nebo přihlášení úspěšné je uživatel přesměrován na domovskou stránku aplikace.

6.4.1 Změna hesla uživatele

```
async changePassword() {
  if (this.newPass && this.oldPass) {
    if (this.newPass.length >= 6) {

      const user = this.auth.currentUser
      const credential = EmailAuthProvider.credential(
        user.email,
        this.oldPass
      )

      if (await reauthenticateWithCredential(user, credential)) {
        updatePassword(user, this.newPass)

        this.alert('Heslo bylo změněno', '')

        this.modalContrl.dismiss()
      } else {
        this.alert('Staré heslo je nesprávné', 'Zkuste to znovu')
      }

    } else {
      this.alert('Nové heslo musí obsahovat aspoň 6 znaků', 'Zkuste to znovu')
    }
  } else {
    this.alert('Musíte vyplnit obě pole', 'Zkuste to znovu')
  }
}
```

Obrázek 20. Ukázka kódu pro změnu hesla uživatele

Kód na obrázku č.20 obsahuje funkci, která uživateli změní heslo. Proměnné newPass a oldPass jsou převzaty ze vstupů na šabloně HTML, jedná se o proměnné pro staré a nové heslo. Staré heslo musí uživatel zadat, protože funkce pluginu Auth `updatePassword()` vyžaduje čerstvé přihlášení uživatele. To je zajištěno pomocí funkce `reauthenticateWithCredential()`, která znovu autentizuje uživatele, čímž se vytvoří nový token přihlášení. Funkce ošetřuje, aby uživatel zadal obě hesla a navíc, jestli je nové heslo aspoň 6 znaků dlouhé, pokud nejsou podmínky splněny, vypíše se uživateli hláška s chybou a výzvou o nový pokus.

7 ZABEZPEČENÍ APLIKACE

Zabezpečení aplikace řeší pluginy AuthGuard a AuthenticationService v pluginovém balíčku firebase. Plugin AuthGuard se stará, aby nepřihlášený uživatel neměl přístup k daným stránkám aplikace, v tomto případu se jedná o všechny stránky aplikace kromě té přihlašovací. Teprve poté až je uživatel autentizován, je mu povolen přístup na domovskou stránku aplikace.

```
const redirUnauth = () => redirectUnauthorizedTo(['']);
const redirLogged = () => redirectLoggedInTo(['home']);

const routes: Routes = [
  {
    path: '',
    loadChildren: () =>
      import('./login/login.module').then((m) => m.LoginPageModule),
    ...canActivate(redirLogged),
  },
  {
    path: 'home',
    loadChildren: () =>
      import('./home/home.module').then((m) => m.HomePageModule),
    ...canActivate(redirUnauth),
  },
  {
    path: '**',
    redirectTo: '',
    pathMatch: 'full',
  },
```

Obrázek 21. Ukázka kódu pro přesměrovávání aplikace

Na obrázku č.21 je ukázka logiky přesměrovávání aplikace. Kód `…canActivate(redirLogged)` aplikace zajišťuje, aby přesměrování na příslušnou stránku proběhlo jen tehdy, kdy je uživatel přihlášen. Kód `…canActivate(redirUnauth)` zajišťuje přesměrování na přihlašovací stránku aplikace, pokud je uživatel nepřihlášen. Poslední kód se stará o přesměrování, pokud je URL adresa špatně zapsána.

```
async register({ email, password }) {
  try {
    const user = createUserWithEmailAndPassword(this.auth, email, password)
    return user
  } catch (error) {
    return null
  }
}

async login({ email, password }) {
  try {
    const user = signInWithEmailAndPassword(this.auth, email, password)
    return user
  } catch (error) {
    return null
  }
}
```

Obrázek 22. Ukázka kódu pro registraci a přihlášení uživatele

Kód na obrázku č.22 se nachází v souboru authentication.service.ts, který zajišťuje logiku přihlašování a registraci uživatele. Funkce `createUserWithEmailAndPassword()` a `signInWithEmailAndPassword()` jsou zabezpečené vestavěné funkce firebase pluginu.

Dále v nastavení databáze ve firebase konzoli jsou pravidla nastavena tak, že pokud není požadavek autentizován, uživatel ji nemůže číst ani do ní zapisovat.

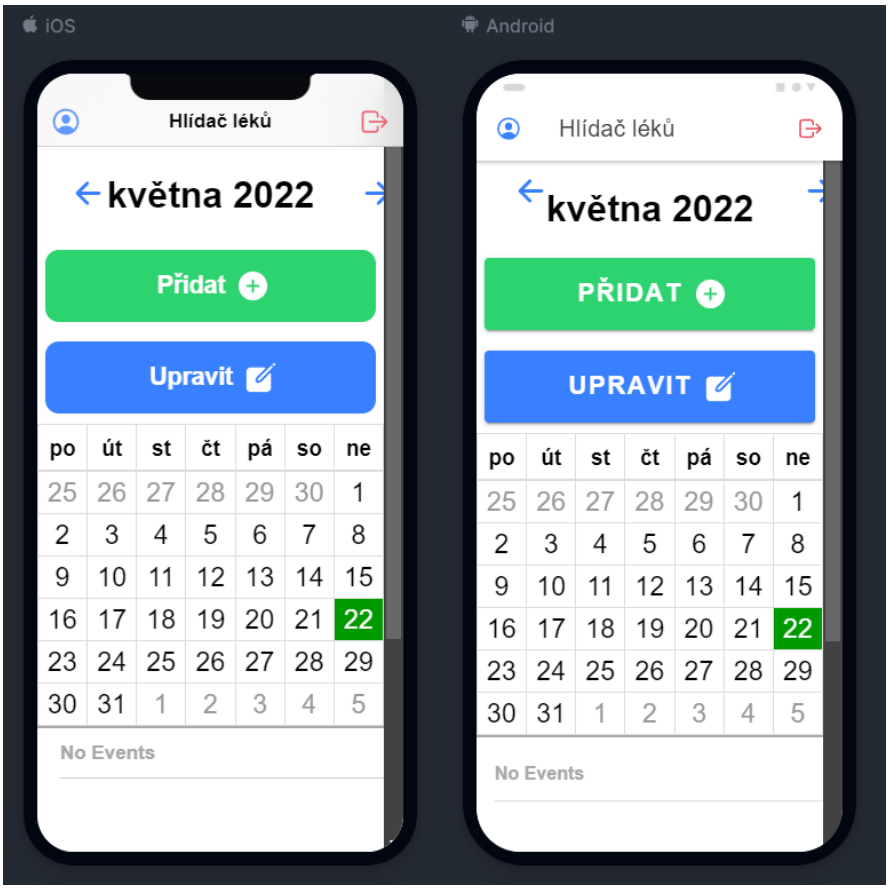
```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

Obrázek 23. Ukázka pravidel pro firestore databázi

8 TESTOVÁNÍ APLIKACE

Hotová aplikace bude testována manuálně. Pro spuštění aplikace je potřeba nainstalovat Ionic CLI pomocí příkazu v příkazovém terminálu pomocí `$ npm install -g @ionic/cli`. Dále v příkazovém terminálu ve zdrojové složce s aplikací spustíme aplikaci příkazem `ionic serve`.

8.1 Test zobrazení

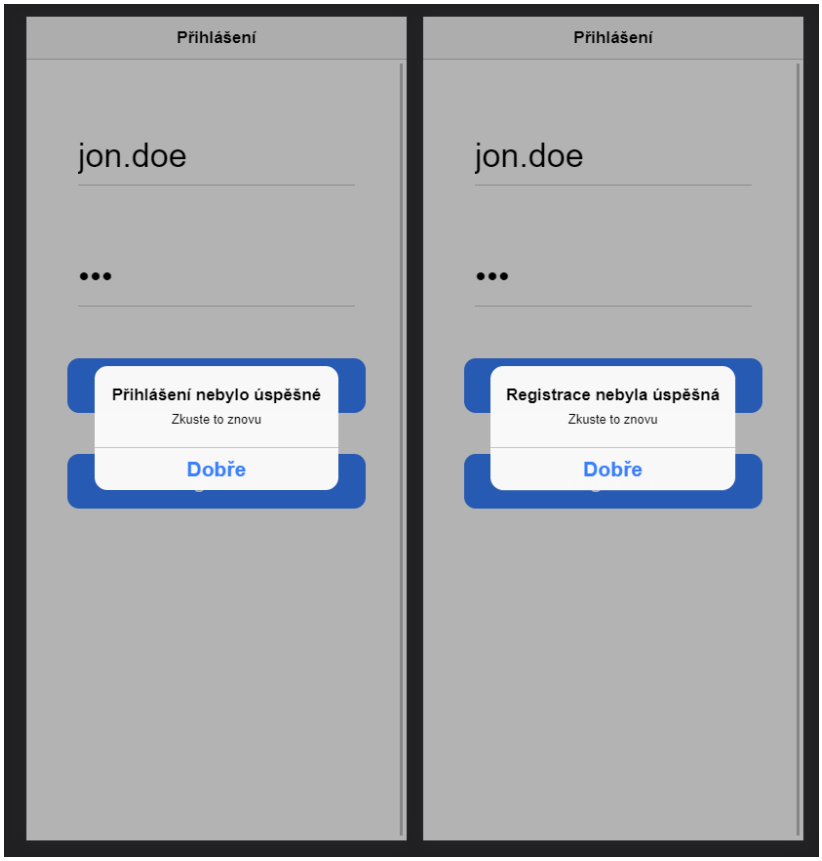


Obrázek 24. Ukázka zobrazení aplikace na platformách iOS a Android

Na obrázku č.24 vidíme, jak se aplikace zobrazuje na zařízeních iOS a Android. Aplikace se zobrazuje správně a je responsivní s měnícím se rozlišením.

8.2 Test autentizace

Dalším testem bude test autentizace uživatele. Test bude spočívat v zadání neplatného emailu a hesla a kontrole, jestli se vykresluje chybový dialog. Následně se otestuje zadání správných údajů a kontrola, jestli je uživatel přihlášen a přesměrován na hlavní stránku.

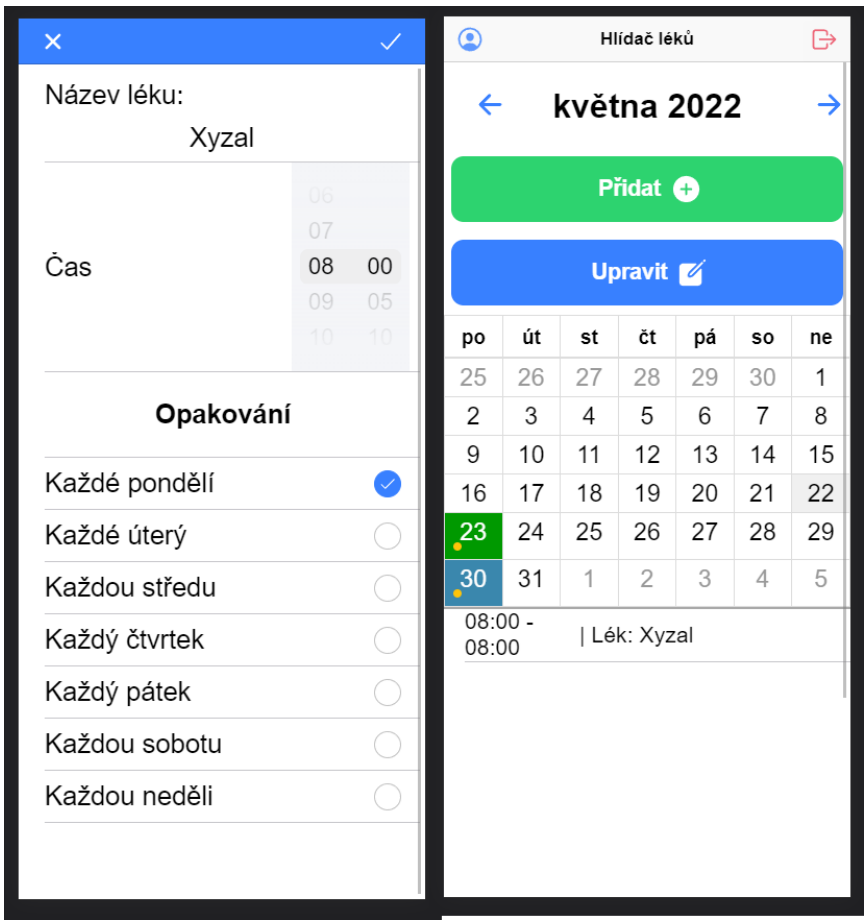


Obrázek 25. Ukázka chybné registrace a přihlášení

Aplikace při chybně zadaných údajích uživateli vykreslí dialog s chybovou hláškou podle očekávání. Při zadání správných údajů je uživatel přihlášen a přesměrován na domovskou stránku podle očekávání.

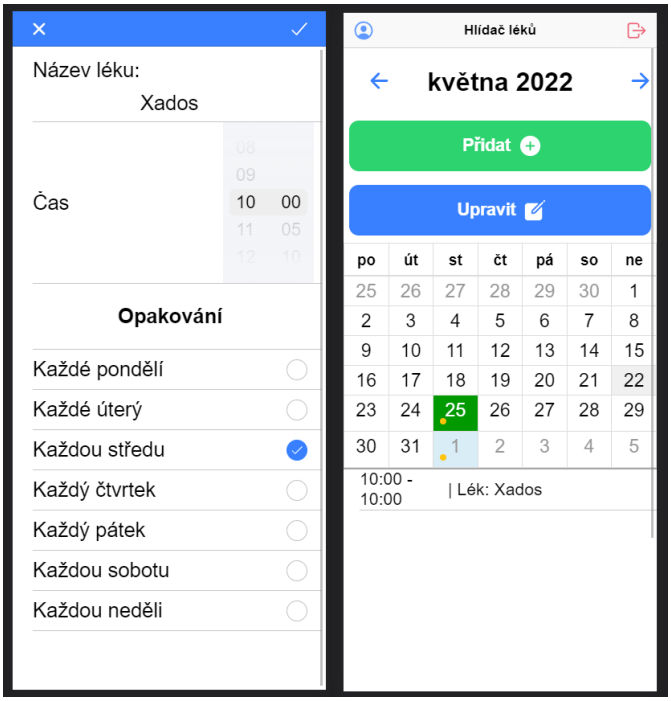
8.3 Test vytvoření, editace a smazání léku a události

Test bude spočívat ve vytvoření události a léku a kontrole, jestli jsou správně zapsány v kalendáři. Poté lék a událost změníme a budeme kontrolovat správnost změny. Nakonec lék i událost vymažeme a zkontrolujeme, jestli byly smazány.



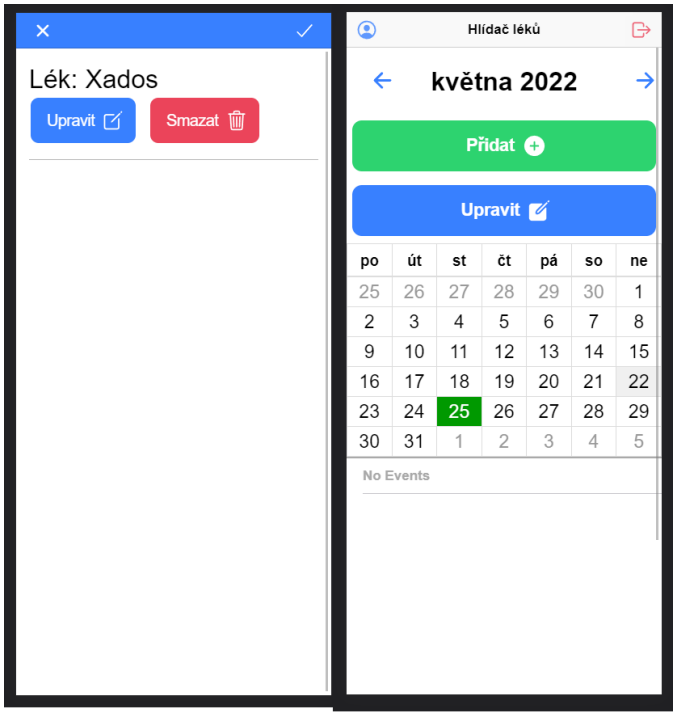
Obrázek 26. Ukázka vytvoření léku v kalendáři

Aplikace při zadání léku i události je úspěšně zapíše do databáze. Data z databáze jsou správně interpretovány v kalendáři přesně podle očekávání.



Obrázek 27. Ukázka změny léku v kalendáři

Při výběru akce upravit lék nebo událost a vyplnění nových údajů aplikace správně změní údaje o léku nebo události a správně je vypíše v kalendáři.

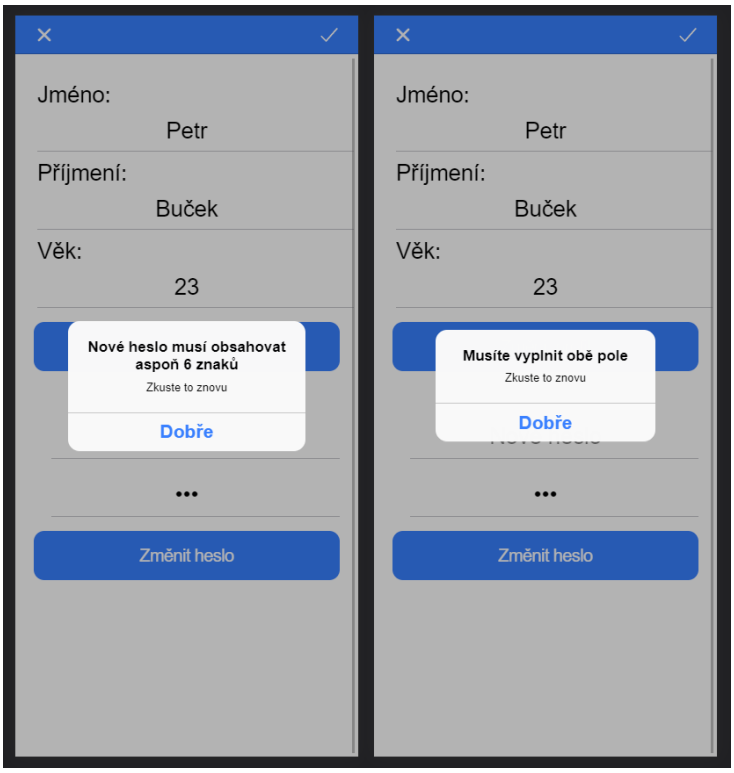


Obrázek 28. Ukázka smazání léku v kalendáři

Při výběru akce smazat lék nebo událost aplikace správně v databázi vymaže požadovaný údaj a správně vykreslí kalendář.

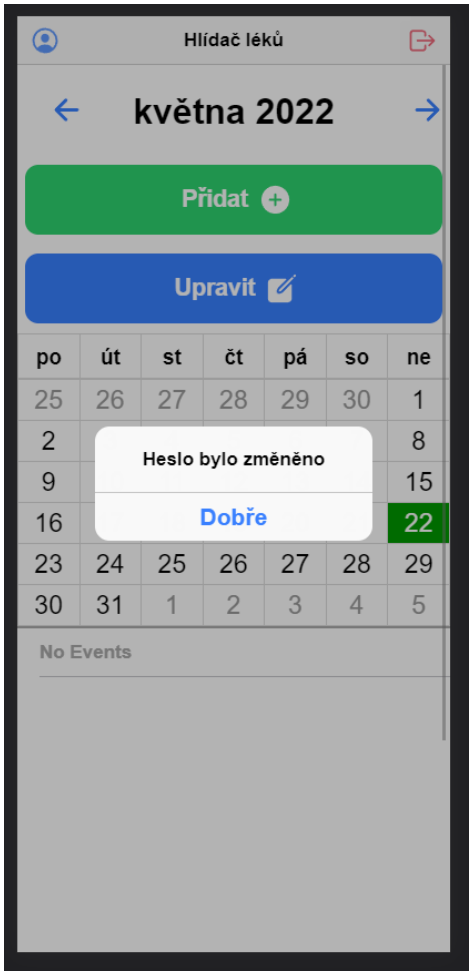
8.4 Test změny hesla uživatele

Zde budeme testovat správnost funkce pro změnu hesla uživatele. Test bude spočívat v tom, že nejprve zadáme chybné údaje a budeme kontrolovat, jestli nám aplikace vypíše chybovou hlášku. Poté vyplníme údaje správné a budeme kontrolovat, jestli se změna hesla zdařila.



Obrázek 29. Ukázka chybových dialogů při změně hesla

Uživatel musí vyplnit nové a staré heslo, pokud jeden z údajů nevyplní, zobrazí se chybový dialog, který upozorňuje na danou chybu. Pokud nové heslo nemá alespoň 6 znaků, je také zobrazen chybový dialog.



Obrázek 30. Ukázka úspěšné změny hesla

Při vyplnění správných údajů je heslo úspěšně změněno a je zobrazen dialog o úspěšné operaci.

8.5 Výsledek

Aplikace prošla všemi manuálními testy. Při testování aplikace v konzoli nevypsala žádnou neočekávanou chybu. Aplikace se na každé své stránce zobrazovala tak jak měla, a to na všech simulovaných zařízeních. Aplikace byla testována v prohlížeči Chrome, který dokáže simulovat zobrazení mobilního zařízení.

ZÁVĚR

Cílem této práce byl návrh a realizace aplikace, která pomůže seniorům při vykonávání obtížných úkonů.

Teoretická část této bakalářské práce se zabývá problematikou stárnutí a navrhuje, jak s touto problematikou mohou pomoci aplikace. Dále jsou zkoumána již dostupná řešení dostupná na internetu a podle nich a rozhovorů se seniory jsou sepsány požadavky na aplikaci.
V praktické části je popsán návrh aplikace, poté je popsána již samotná realizace této aplikace. V následujících kapitolách je popsáno zabezpečení aplikace a samotné testování aplikace.

Výsledná aplikace po vzoru sběru požadavků pomůže seniorovi při pamatování si událostí a braní léků. Důraz při vývoji byl kladen na přehlednost a intuitivnost uživatelského prostředí aplikace. Aplikace se od ostatních dostupných aplikací popsaných v druhé kapitole této práce odlišuje tím, že není vázána na jednu platformu, ale je přístupná ze všech. Aplikace plní všechny funkční požadavky popsané v návrhu aplikace.

SEZNAM POUŽITÉ LITERATURY

[1] Informační společnost v číslech [online]. Česká republika: Český statistický úřad, 2020 [cit. 2022-05-09]. Dostupné z: https://www.czso.cz/documents/10180/122362632/06100420c.pdf/01ab7bd8-1baa-4b8d-854d-81d000d0c953?version=1.2

[2] Nejčastější zdravotní potíže seniorů [online]. Česká republika: Vademecum zdraví, 2007 [cit. 2022-05-08]. Dostupné z: http://vademecum-zdravi.cz/nejcastejsi-zdravotni-potize-senioru

[3] MARTINEK, Ladislav. Kognitivní funkce u seniorů. Česká republika, Olomouc, 2015. Magisterská diplomová práce. Univerzita Palackého v Olomouc.

[4] Mind games: Study finds gaming improves memory and physical function in older people [online]. USA: USQ, 2021 [cit. 2022-05-09]. Dostupné z: https://www.usq.edu.au/news/2021/05/gaming-for-older-people

[5] Typescript history [online]. USA: Coderlipi, 2019 [cit. 2022-05-09]. Dostupné z: https://coderlipi.com/typescript/typescript-history

[6] FENTON, Steve. Pro TypeScript: Application-Scale JavaScript Development. Apress, 2014. ISBN 978-1430267911.

[7] Angular - What is angular [online]. USA: Angular, 2022 [cit. 2022-05-09]. Dostupné z: https://angular.io/guide/what-is-angular

[8] Hisotry of Angular [online]. USA: Medium, 2018 [cit. 2022-05-09]. Dostupné z: https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7

[9] Introduction to Angular concepts [online]. USA: Angular, 2022 [cit. 2022-05-09]. Dostupné z: https://angular.io/guide/architecture

[10] Medisafe [online]. USA: App Store, 2022 [cit. 2022-05-10]. Dostupné z: https://apps.apple.com/cz/app/medisafe-medication-management/id573916946?l=cs

[11] Pill Tracker+ [online]. USA: App Store, 2022 [cit. 2022-05-10]. Dostupné z: https://apps.apple.com/cz/app/pill-tracker/id1511818943

[12] Mr. Pillster [online]. USA: App Store, 2022 [cit. 2022-05-10]. Dostupné z: https://apps.apple.com/cz/app/mr-pillster-pill-reminder/id1116960025

[13] Pill Reminder Medication Alarm [online]. USA: App Store, 2022 [cit. 2022-05-10]. Dostupné z: https://apps.apple.com/cz/app/pill-reminder-medication-alarm/id863327251

[14] GRIFFITH, Chris. Mobile App Development with Ionic, Revised Edition. USA: O'Reilly Media, 2017. ISBN 9781491998120.

[15] WebView [online]. USA: Developers, 2022 [cit. 2022-05-11]. Dostupné z: https://developer.android.com/reference/android/webkit/WebView

[16] Hybrid application (hybrid app) [online]. USA: TechTarget, 2019 [cit. 2022-05-11]. Dostupné z: https://www.techtarget.com/searchsoftwarequality/definition/hybrid-application-hybrid-app

[17] What is Firebase [online]. USA: Educative, 2022 [cit. 2022-05-11]. Dostupné z: https://www.educative.io/edpresso/what-is-firebase

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

UI	User Interface
PWA	Progressive Web Application
DOM	Document Object Model
HTML	Hypertext Markup Language
URL	Uniform Resource Locator
ID	IDentification
ERD	Entity Releationship Diagram
JSON	JavaScript Object Notation

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: SOUBOROVÁ STRUKTURA ANGULAR APLIKACE