FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Rodinný organizér
Student:	Pavel Lipenský
Vedoucí:	Ing. Jan Blizničenko
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem této práce je vytvořit progresivní webovou aplikaci (PWA) - rodinný organizátor pro plánování a komunikaci v rodinném kruhu. Každý uživatel, který se registruje a přihlásí, si může vytvořit svůj rodinný kruh. Do svého rodinného kruhu může uživatel pozvat další registrované uživatele. Může se také přidat do existujícího rodinného kruhu. Členové rodinného kruhu mohou:
- komunikovat prostřednictvím společného chatu,
- sdílet a organizovat rodinné události v kalendáři,
- vytvářet nákupní seznamy.
- vytvářet seznamy úkolů.
- vzájemně evidovat neproplacené finanční prostředky.
- sdílet svoji aktuální polohu s ostatními členy.

Uživatelé jsou informováni o důležitých událostech push notifikacemi.

- Proveďte rešerši podobných nástrojů.
- Proveďte rešerši relevantních technologií, nástrojů, frameworků a knihoven.
- Vytvořte návrh aplikace a jejích součástí.
- Proveďte implementaci aplikace.
- Otestujte a zdokumentujte své řešení, vč. dokumentace nasazení a uživatelské dokumentace.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 14. července 2021 v Praze.

Bakalářská práce

RODINNÝ ORGANIZÉR

Pavel Lipenský

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Jan Blizničenko
4. ledna 2022

Obsah

Seznam obrázků

Seznam výpisů kódu

Chtěl bych poděkovat vedoucímu mé bakalářské práce, inženýru Janu Blizničenkovi, za ochotu, pomoc a cenné rady. Největší dík patří mým rodičům, kteří mě podporovali po celou dobu mého studia na vysoké škole, a přítelkyni Martině, která mi pomohla s korekcí textů.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 4. ledna 2022 .....

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací rodinného organizéru. V práci je provedeno porovnání s konkurenčními řešeními, analýza použitých technologií a návrh uživatelského rozhraní. Dále je popsán celý postup vývoje aplikace od návrhu až po její nasazení. Funkčnost a správný návrh aplikace byl ověřen uživatelským testováním.

Aplikace je postavena na balíku služeb od Firebase od společnosti Google, kde využívá databázi Firestore a Firebase Authentication pro ověřování identity uživatele. Pro frontend je zvolený jazyk Java Script a framework React.js a pro styly je využito frameworku Tailwind CSS.

Výsledkem práce je PWA aplikace, která je spustitelná na jakémkoli počítači nebo mobilním zařízení s internetovým připojením a webovým prohlížečem.

Klíčová slova Progresivní webová aplikace, PWA, rodinný organizér, React, Firebase, Tailwind CSS, seznam úkolů, sdílení polohy, chat, kalendář

Abstract

This bachelor thesis deals with the design and implementation of a family organizer. The thesis includes a comparison with competing solutions, an analysis of the technologies used and the design of the user interface. Furthermore, the whole application development process from design to deployment is described. The functionality and correct design of the application has been verified by user testing.

The application is built on Google's Firebase suite of services, where it uses the Firestore database and Firebase Authentication to verify the user's identity. The language chosen for the frontend is JavaScript and the React.js framework, and the Tailwind CSS framework is used for styles.

The result of the work is a PWA application that is executable on any computer or mobile device with an internet connection and a web browser.

Keywords Progressive web application, PWA, Family organizer, React, Firebase, Tailwind CSS, To-do list, Location sharing, Chat, Calendar

Seznam zkratek

API	Application Programming Interface
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
DOM	Document Object Model
FCM	Firebase Cloud Messaging
GPS	Global Positioning System
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
JS	JavaScript
MVC	Model-View-Controller
NoSQL	Not Only SQL
NPM	Node Package Manager
OOP	Object-oriented programming
SQL	Structured Query Language
UI	User Interface
URL	Uniform Resource Locator

Slovník

Backend	Část systému, která se stará o logiku a práci s daty.
Breakpoint	Bod zlomu, ve kterém dojde ke změně rozložení stránky.
Framework	Softwarová struktura, která slouží jako podpora při programování a vývoji.
Frontend	Část systému, kterou vidí návštěvník stránek.
Google play	Distribuční služba Googlu pro digitální obsah, nejčastěji mobilní aplikace pro operační systém Android.
Chat	Komunikace formou psaného textu dvou nebo více lidí prostřednictvím komunikační sítě.
Key-Value	Způsob ukládání dat, kde má každá hodnota svůj klíč.
Konverzní akce	Konverzní akce je taková akce, která má přínos pro provozovatele webu. Příkladem může být nákup zboží, služeb, přihlášení odběru nebo třeba odeslání poptávky.
Layout	Grafické rozvržení nebo struktura elektronické stránky.
Mobile first	Návrh UI nejdříve pro mobilní telefony, až následně pro desktop.
Open-Source	Počítačový software s otevřeným zdrojovým kódem.
Rozhraní	Slouží k interakci člověka s aplikací.
Uživatel	Člověk vlastnící zařízení, na kterém používá aplikaci.

Kapitola 1
Úvod

V dnešní době je spousta lidí často mimo domov, například v práci nebo na studentských kolejích. Vzhledem k tomu, že členové jednotlivých rodin nejsou spolu nikdy tolik času, kolik by si přáli, hledají způsoby komunikace mezi sebou. Každý z nás má svůj okruh nejbližších lidí, se kterými chce sdílet společné události, narozeniny, oslavy, nebo třeba povinnosti, které se týkají všech členů rodiny.

K organizování nebo datování významných událostí slouží již několik staletí papírový kalendář nebo obyčejný papír, na který se dá zapsat úplně cokoli. Může posloužit od psaní poznámek a seznamů, přes vedení účetních informací, až po napsání dopisu a odeslání jej poštou adresátovi. Toto je již v dnešní době s moderními technologiemi vylepšeno a není nutné předávat nákupní seznam napsaný na papíře, nebo posílat dopis poštou.

Výsledek této práce, jímž bude aplikace Rodinný organizér, je určen pro rodiny nebo okruh lidí, kteří se nevídají a chtějí být více v kontaktu. Sám jsem přemýšlel nad tím, jak si v naší rodině dost často složitě předáváme informace a komunikujeme spolu na několika různých platformách. Toto téma jsem si zvolil, protože jsem sám hledal řešení, kterým bychom se sjednotili na jedné platformě nebo aplikaci. Hotových řešení existuje překvapivě několik, ale po podrobnějším prozkoumání žádné nemělo funkcionality, které jsou pro komunikaci v rodině důležité, nebo byla část funkcionalit zpoplatněna.

Požadavky, které budou na rodinný organizér kladeny, jsou podrobně popsány v kapitole Cíl práce. Pro seznámení a získání kontextu je v kapitole Analýza stávajících řešení provedena analýza již existujících podobných řešení. Pro vhodnou implementaci je potřeba provést návrh řešení a výběr vhodných technologií, což je popsáno v kapitole Návrh. Popis a postup samotné implementace včetně nasazení je v kapitole Realizace. Testování je důležitou součástí každého vývoje aplikace. Postupy testování jsou vysvětleny ve stejnojmenné kapitole. V závěru práce jsou shrnuty poznatky získané při vývoji, zhodnocení vhodnosti použitých technologií a v neposlední řadě je v této kapitole návrh možného funkcionálního rozšíření do budoucna.

Kapitola 2
Cíl práce

Cílem bakalářské práce je navrhnout a implementovat rodinný organizér jako progresivní webovou aplikaci, která usnadní plánování a komunikaci v rodinném kruhu.

Hlavním cílem rešerše je zanalyzovat podobné nástroje a aplikace. Dále zde získáváme přehled o funkcionalitách konkurenčních aplikací a seznámíme se se základními principy, na kterých tyto aplikace fungují. Na konci rešerše se nachází vyhodnocení jednotlivých konkurenčních aplikací z hlediska jejich funkcionalit.

V praktické části je cílem implementace na základě provedené rešerše. Výsledkem bude organizér, který bude možné spustit na jakémkoli počítači s internetovým prohlížečem. Jelikož půjde o progresivní webovou aplikaci, je možné ji umístit na domovskou obrazovku telefonu, tabletu nebo desktopu stejně jako nativní aplikaci.

Všichni uživatelé, kteří se zaregistrují a přihlásí, si budou moci vytvořit svůj rodinný kruh, do kterého si přidají další registrované uživatele. Další možností je, že se uživatelé přidají k rodinnému kruhu, který již existuje, a to na základě přijetí pozvánky. Členové rodinného kruhu budou mít k dispozici několik nástrojů. Budou moci komunikovat prostřednictvím společného chatu, sdílet a organizovat rodinné události v kalendáři, vytvářet nákupní seznamy nebo seznamy úkolů. Všichni uživatelé budou mít také možnost evidovat neproplacené finanční prostředky nebo sdílet svoji aktuální polohu s ostatními členy kruhu. O důležitých událostech budou uživatelé informováni prostřednictvím *push* notifikací.

Pro tento organizér využiji vhodné softwarové návrhové vzory a technologie. Navrhnu moderní a funkční uživatelské prostředí, které bude vhodné pro cílovou skupinu, kterou je v tomto případě převážně rodina a její členové. Kód bude konzistentně strukturován a formátován, aby byla v budoucnu možná rozšiřitelnost o další funkcionality a moduly.

Kapitola 3
Analýza

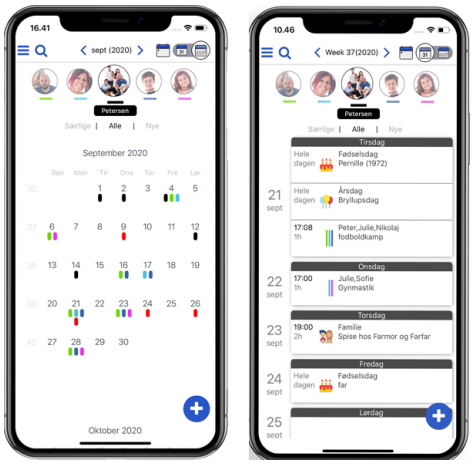
3.1 Analýza stávajících řešení

Konkurenční aplikace a řešení pro prototyp této práce lze rozdělit do dvou skupin. První z nich jsou aplikace zaměřené přímo na rodinu, rodinný kruh nebo motivaci dětí k učení, případně plnění domácích povinností. Tato skupina aplikací má uzpůsobené funkcionality a své uživatelské rozhraní k tomu, aby ovládání a pohyb v aplikaci byl příjemný a intuitivní. Tím oslovuje co nejširší cílovou skupinu uživatelů. Do této skupiny spadá i mé řešení.

Do druhé skupiny zařazuji aplikace, které se zaměřují typicky jen na jednu z funkcionalit, kterou však mají propracovanou velmi do hloubky. Nejsou přímo určené pro typickou cílovou skupinu mé bakalářské práce, a tak může být jejich používání pro běžné uživatele složitější. Kombinace těchto aplikací může sloužit jako ekvivalent, který pokryje většinu podstatných funkcionalit nabízených aplikacemi v první popsané skupině.

3.1.1 Daysi Family App

Daysi je mobilní aplikace, která nemá webové rozhraní. Po registraci a vytvoření rodiny v aplikaci je třeba přidat a registrovat další členy rodiny. Tuto registraci provádí vždy zakládající člen rodiny a přihlašovací údaje, včetně hesla, pak sdělí novému členovi [1].



Obrázek 3.1 Aplikace Daysi Family App [1]

Středem aplikace je sdílený kalendář, do kterého mohou uživatelé přidávat události, ke kterým přiřazují jednotlivé členy rodiny. Uživatelé, v tomto případě hlavně děti, mohou plnit úkoly, které si vytvoří samy, nebo jim je vytvoří jiný uživatel, nejčastěji rodič. Pokud dítě úkol splní a rodič jej schválí, je mu přičtena virtuální platba jako kapesné. Aplikace disponuje také seznamem úkolů, který je ale přístupný až v placené verzi [1].

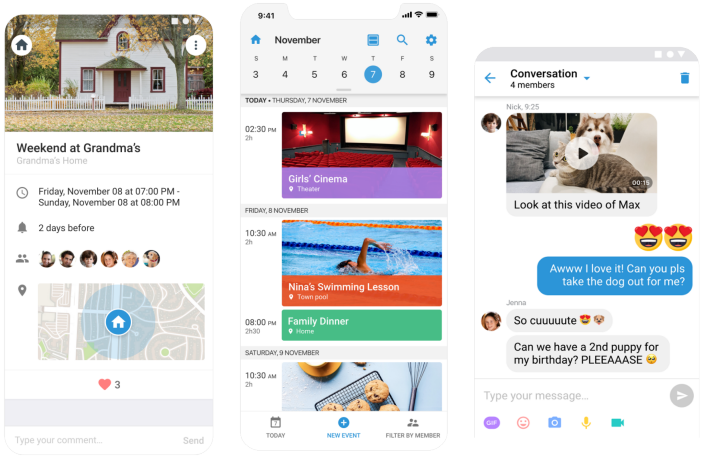
Hlavní nevýhoda aplikace spočívá v jejím pomalém běhu a na jisté interakce má velmi rozdílný čas odezvy než na jiné. Uživatel proto může být velmi zmatený a frustrovaný. Po provedení jakékoli akce neví, zda se aplikace načítá, nebo se jen netrefil na tlačítko. Prostředí je velmi neintuitivní až zmatené. Tlačítka jsou velmi blízko u sebe a některá mají velmi podobné ikonky. Graficky aplikace působí zastarale a nelíbivě. Při prozkoumávání aplikace jsem narazil i na texty, které nebyly přeloženy do angličtiny, přestože se aplikace v angličtině prezentuje [1]. Hodnocení na Google play dosahuje 2,5 hvězdiček a převládá nejnižší hodnocení [2].

3.1.2 FamilyWall

FamilyWall je rodinný asistent, který má mobilní i webovou aplikaci a nabízí širokou paletu funkcí. Po přihlášení se na hlavní stránce objevuje zeď aktivit, která nabízí podobnou funkci jako zdi známé z populárních sociálních sítí [3].

Disponuje propracovaným sdíleným kalendářem, který je možné synchronizovat s Google kalendářem. Členové rodinného kruhu mohou vytvářet nákupní seznamy a seznamy úkolů, které mohou sdílet s ostatními členy rodinného kruhu. Aplikace obsahuje speciální sekci na sdílení receptů mezi členy kruhu. Uživatelé mohou sdílet svoji polohu a zároveň vidět poslední sdílenou polohu ostatních členů rodinného kruhu. Aplikace dále disponuje moderním chatem, kde si uživatelé mohou posílat zprávy mezi sebou, ve skupinách nebo se všemi členy rodinného kruhu. FamilyWall poskytuje také sdílenou galerii, kde mohou všichni nahrávat společné fotografie. Galerie je ale omezena prostorem 100 MB na jeden rodinný kruh a fotografie jsou ukládány ve velmi nízké kvalitě [3].

FamilyWall má oproti ostatním aplikacím propracovaný, líbivý a hlavně funkční design. Nabízí mnoho funkcí, které je možné ještě rozšířit v placené verzi. Nenabízí však žádnou evidenci neproplacených finančních prostředků a mnoho funkcí je zpoplatněno [3]. Na Google play se aplikace pyšní hodnocením 4,6 hvězdiček [4].



Obrázek 3.2 Aplikace FamilyWall [3]

3.1.3 Cozi

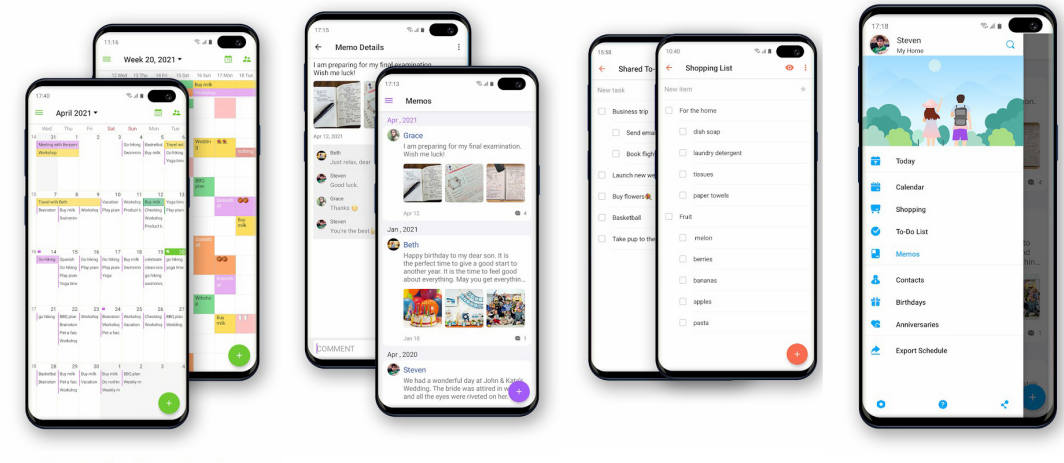
Cozi je mobilní i webová aplikace, která je rodinným organizérem. Nabízí sdílený kalendář pro všechny členy rodiny. Uživatel může vytvářet seznamy úkolů, které může přidělit sobě nebo jinému členovi rodiny. V Cozi je možné vytvářet také nákupní seznamy a stejně jako seznamy úkolů je možné je přiřadit jakémukoliv členovi rodiny. Poslední funkcí, kterou Cozi nabízí, jsou sdílené recepty, které je možné plánovat do kalendáře podle toho, kdy se které jídlo bude připravovat [5].

Cozi je již starší aplikace, což lze poznat jak na vzhledu a designu mobilní aplikace, tak ve webovém rozhraní. Některé ovládací prvky jsou méně intuitivní a rozdílně řešené v mobilní a ve webové aplikaci. Další rozšiřující funkce, jako jsou například narozeniny členů rodiny nebo vyhledávání v aplikaci, jsou až v placené verzi. Vše ale funguje dobře a tomu odpovídá i dobré hodnocení aplikace v Google play, kde aplikace získala 3,9 hvězdiček [6].

3.1.4 FamCal

FamCal je mobilní aplikace, jejíž název vychází ze spojení slov „Family“ a „Calendar“. Jak tedy název napovídá, jedná se o sdílený rodinný kalendář s několika funkcemi navíc. Kalendář nabízí vše podstatné a přehledně zobrazuje události. Aplikace nabízí vytváření seznamů úkolů a nákupních seznamů. Poslední funkcí, kterou FamCal nabízí zdarma, jsou momenty. Momenty obsahují text s obrázky a slouží k uchování rodinných vzpomínek. Vzpomínky mohou ostatní členové rodiny komentovat. V placené verzi FamCal navíc nabídne sdílené kontakty, upozorňování na narozeniny a výroční události, emailové notifikace a tisk plánu [7].

Aplikace funguje dobře, nabízí líbivé a funkčně zaměřené prostředí. Aplikace působí moderně, je intuitivní a rychlá. Kromě momentů ale nenabízí žádný jiný způsob komunikace, nebo interakce. Mnoho funkcionalit je zpoplatněno a při používání bezplatné verze je rušivé a obtěžující větší množství reklam v aplikaci. FamCal má na Google play hodnocení 4,5 hvězdiček [7].



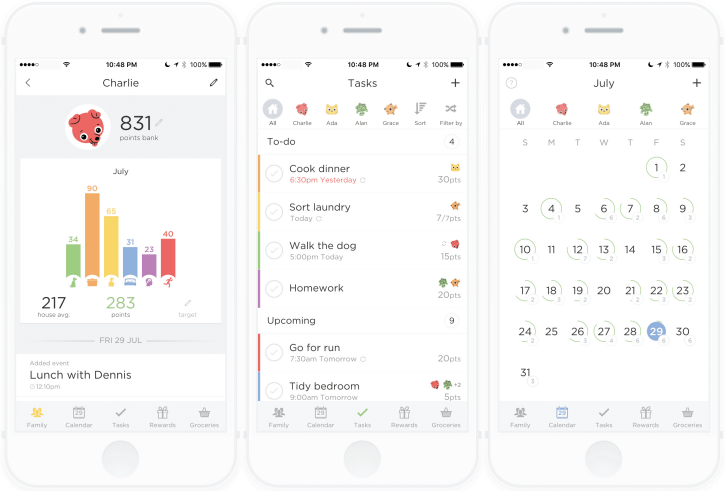
Obrázek 3.3 Aplikace FamCal [7]

3.1.5 OurHome

Aplikace OurHome je aplikace jak mobilní, tak webová. Od ostatních je odlišná tím, že je zaměřena primárně na motivaci dětí k plnění domácích nebo jakýchkoliv jiných povinností. Za splnění nějakého úkolu je uživateli přidělen určitý počet bodů. Za body může uživatel (v tomto

případě dítě) dostat nějakou předem domluvenou odměnu. Aplikace obsahuje kalendář, ve kterém lze vytvářet klasické události, nebo bodované úkoly. Nákupní seznam je také propojen s bodovým systémem [8].

Aplikace je velmi povedená a pro děti je dle mého názoru atraktivní. Informace zobrazuje přehledně a intuitivně. Vše je zabaleno v hezkém grafickém kabátu a doprovázeno milými ilustracemi. Cílem této aplikace je děti motivovat, učit je odpovědnosti a dosahování vlastních cílů. Přestože má OurHome podobné funkcionality jako moje aplikace, její cíl je úplně jiný. Aplikace má v Google play 3,5 hvězdiček [9].



Obrázek 3.4 Aplikace OurHome [8]

3.1.6 Trello, Google kalendář, Evernote

Tyto tři aplikace by se daly považovat za alternativu ke konceptu rodinného organizéru. Každá z nich pokrývá jednu nebo více funkcionalit. Díky tomu, že mají užší zaměření, pokrývají svou problematiku velmi dobře. Všechny tyto aplikace jsou multiplatformní, díky čemuž je uživatelé mohou používat na všech zařízeních.

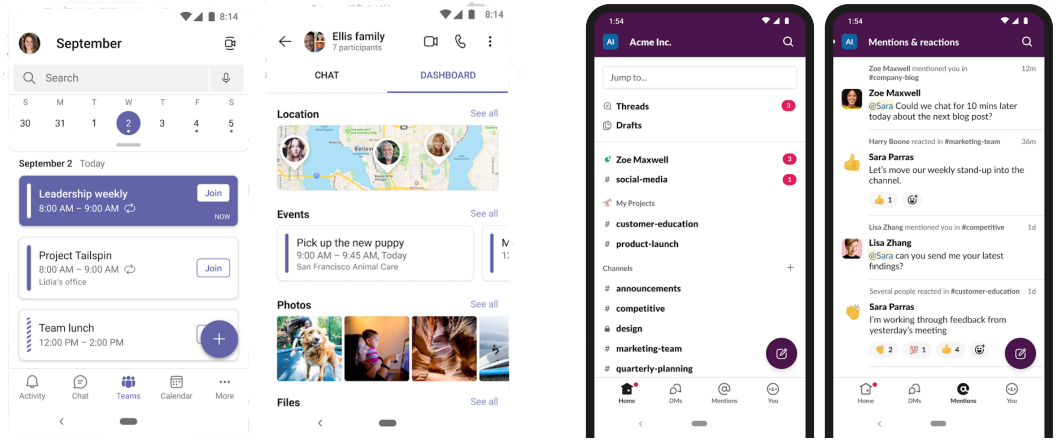
Trello je bezplatná webová aplikace pro týmovou správu projektů. Obsahuje nástěnky se seznamy úkolů. K těmto úkolům lze přiřazovat uživatele, soubory nebo třeba časové termíny. Nástěnky se dají přehledně organizovat a řadit [10].

Google kalendář je jednou z nejpoužívanějších aplikací svého druhu vůbec. Jde o internetový kalendář, který umožňuje kompletní správu času uživatele, úkolů, setkání a výročí. Největší výhodou je propojenost s ekosystémem Google, tedy například s aplikacemi Gmail nebo Disk Google [11].

Evernote je aplikace pro pořizování poznámek a úkolů s jejich následnou archivací. Poznámka může mít podobu čistého textu, obrázku nebo mluveného slova. K poznámkám lze přikládat soubory a lze je sdílet s ostatními uživateli [12].

3.1.7 MS Teams, Slack

Tyto aplikace jsou zaměřeny na firemní použití a mohou být jistou alternativou k již zmíněnému konceptu. Obě platformy slouží k organizaci a komunikaci větší skupiny lidí, typicky firem. Podporují textovou komunikaci, video hovory, ukládání a sdílení souborů. Obě také umožňují integrovat další rozšiřující aplikace. Správa těchto platforem není pro běžného uživatele snadná a díky placeným verzím nejsou pro každého dostupné [13], [14].



Obrázek 3.5 MS Teams a Slack [15], [16]

3.1.8 Papírový kalendář

Fyzický papírový kalendář je také určitou alternativou. První kalendáře vznikaly již před naším letopočtem pro uspořádání svátků, jelikož se podle nich původně provádělo datování událostí. Později se do kalendářů začaly přidávat i základní astronomické údaje a s rozšířením tisku se dostaly mezi širší skupinu lidí. Několik let nazpět byl tištěný kalendář obvyklou součástí každé rodiny. Zapisovaly se do něj narozeniny blízkých, pravidelné lékařské prohlídky, výlety, dovolené a mnoho dalšího. Tento způsob organizování byl a stále je velmi populární [17].

3.1.9 Závěr

I přes velkou konkurenci v této oblasti neexistuje aplikace, která by bezplatně nabízela funkcionality popsané v kapitole Funkční požadavky. Rodina, která chce tyto funkce v aplikaci, si buď musí aplikaci předplácet, nebo se vzdát některé důležité funkcionality a nahradit ji jinou aplikací.

Konkurenční aplikace nabízí mnoho funkčností, ale ve většině případů jsou tyto aplikace nedoladěné a mají nedostatky. Prototyp mé aplikace uskupí různé funkce jednotlivých aplikací, které jsou podle mě důležité pro cílovou skupinu, kterou je rodina, a zkombinuje je. Výsledná aplikace bude ve formě PWA, tedy progresivní webové aplikace, díky čemuž bude multiplatformní a bude fungovat na jakémkoliv zařízení s webovým prohlížečem.

3.2 Analýza relevantních technologií

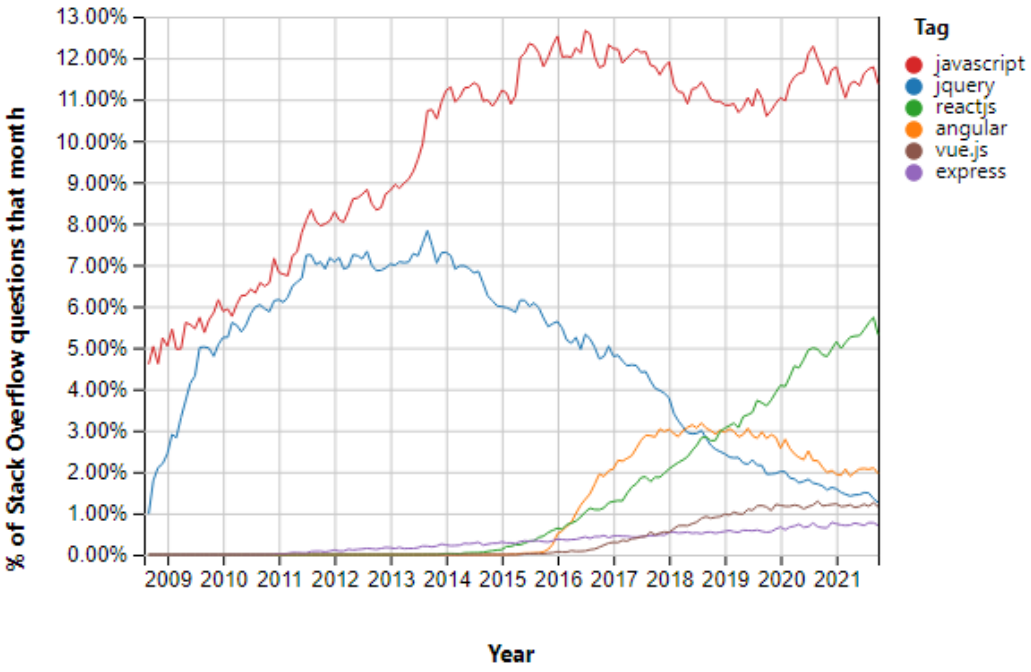
V této kapitole jsou analyzovány technologie, které jsou vhodné pro vývoj progresivní webové aplikace a konkrétně ty, které jsou relevantní pro mou aplikaci.

Pro Rodinný organizér byl vybrán jazyk JavaScript. Jde o interpretovaný programovací jazyk, který je jedním z nejpoužívanějších jazyků pro vývoj webových stránek a aplikací. Jde o dynamicky typovaný jazyk s prvky funkcionálního a objektově orientovaného programování. Umožňuje vytvářet webové stránky s dynamickým obsahem, a protože typicky figuruje na straně klienta v prohlížeči, je velmi rychlý. Díky jeho popularitě existuje mnoho knihoven a frameworků, které rozšiřují a zjednodušují jeho možnosti [18].

3.2.1 Framework

„*Framework je struktura, na které můžeme stavět náš software. Slouží jako základ, takže nezačínáme úplně od nuly.*“ Používáním frameworku se snižuje počet chyb, kód je bezpečnější a píše se díky němu čistší a čitelnější kód [19].

Pro JavaScript existuje mnoho frameworků. Mezi nejpoužívanější a nejrozšířenější patří trojice Angular.js, Vue.js a React.js. Poslední zmíněný jsem si vybral pro svou práci. Tato trojice frameworků je podrobněji rozebraná v následujících kapitolách. Podle statistik na webu Stack Overflow získávají JavaScriptové frameworky větší a větší popularitu, jak je vidět na obrázku 3.6.

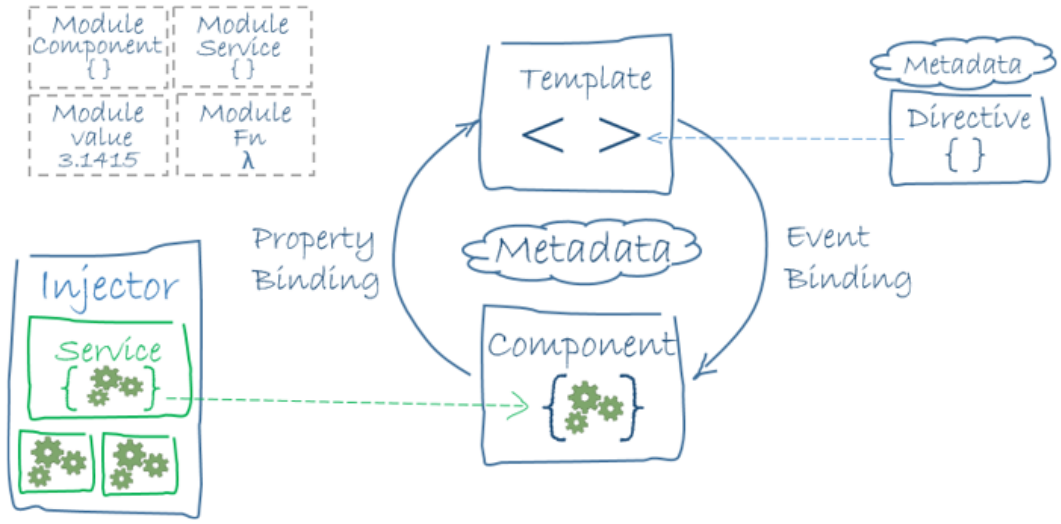


Obrázek 3.6 Četnost otázek na jednotlivé frameworky [20]

3.2.1.1 Angular

Angular je open-source frontendový framework, který slouží pro tvorbu webových aplikací. Využívá k tomu komponentovou architekturu se službami (anglicky services) a místo čistého JavaScriptu používá jazyk TypeScript. Je dobré zmínit, že existuje Angular.js, který je předchůdcem frameworku Angular, a přestože spolu úzce souvisí, jsou spolu nekompatibilní [21].

Vše v Angularu je postaveno na komponentách, které staví na objektově orientovaných třídách (OOP) a vychází z modelu MVC (Model-View-Controller). V těchto třídách jsou pak vázany HTML šablony a CSS styly. Každá webová stránka je složena z vícero komponent, kde má každá komponenta své izolované funkcionality, jako tomu je v OOP. Systém komponent se využívá hlavně za účelem znovupoužitelnosti. O logiku aplikace se starají tzv. služby, které zajišťují předávání dat a komunikaci mezi Angularem a serverovým API. Komponenty a služby se dají rozdělit do logických modulů, jak je ukázáno na obrázku 3.7 [22]. Angular využívají například služby Microsoft Office, Gmail nebo třeba PayPal [23].



Obrázek 3.7 Diagram základních částí Angularu [24]

3.2.1.2 React.js

React je JavaScriptová knihovna od společnosti Meta (dříve známá jako Facebook), která je určená pro tvorbu uživatelského rozhraní. React se oproti frameworku Angular zaměřuje například pouze na jednu specifickou oblast MVC modelu, a to na vrstvu view (vrstvu pohledu). Tato vrstva se zaměřuje na vykreslení jednotlivých komponent včetně dat uživateli [25].

Podobně jako Angular staví React na komponentách se zapouzdřenými funkcionalitami. Skládáním a znovupoužíváním těchto komponent vzniká celé UI. Komponenty mají své *props* (vlastnosti), které spravují své vnitřní *states* (stavy). Tento způsob práce s daty umožňuje snadno předvídatelné chování a snadnější ladění aplikace [25]. Jelikož je React od společnosti Meta, která vyvíjí nejen Facebook, je logické, že má své služby a aplikace postavené na svém frameworku. React je využíván aplikacemi Facebook, Instagram, WhatsApp nebo třeba Netflix a New York Times [26].

3.2.1.3 Vue.js

„*Vue je progresivní framework pro vytváření uživatelských rozhraní.*“ Stejně jako předchozí dva frameworky je Vue založený na komponentách. Tato komponenta se skládá ze tří částí. První z nich je šablona složená z HTML tagů. Druhou částí je JavaScriptový kód, který se stará o obsluhu dat. Poslední jsou CSS styly [27].

Vue používá architektonický vzor model-view-viewmodel (MVVM). Tento vzor odděluje grafické uživatelské rozhraní (UI) neboli *view* od business logiky aplikace. Vrstva viewmodel je konvertorové médium, které synchronizuje data [27]. Framework Vue využívají weby Adobe, Gitlab nebo třeba Apple [28].

3.2.2 Databáze

Mezi nejčastěji využívané databáze patří relační a NoSQL databáze. Existuje ale mnohem více typů databází: cloudové databáze, objektově orientované databáze nebo key-value databáze. V několika následujících podkapitolách jsou rozebráni zástupci několika typů databází [29].

3.2.2.1 MySQL

„MySQL je nejoblíbenější open source SQL databáze, která je vyvíjena, distribuována a podporována společností Oracle Corporation. Relační databáze ukládá data do samostatných tabulek místo toho, aby všechna data vkládala do jednoho velkého úložiště. Struktury databáze jsou organizovány do fyzických souborů optimalizovaných pro rychlost.“ [30] (překlad autora práce)

Výhodou relačních databází je přesná struktura dat a díky tomu je možné definovat relace mezi jednotlivými tabulkami. Tento přístup může být i nevýhodou, a to v případě, že předem neznáme přesnou strukturu dat.

3.2.2.2 MongoDB

MongoDB se řadí mezi NoSQL databáze. Namísto řádků v tabulkách používá MongoDB kolekce a dokumenty. Dokumenty jsou tvořeny dvojicemi klíč a hodnota, které jsou základní jednotkou dat. Jeden z důvodů populárnosti MongoDB je flexibilní formát dat podobný formátu JSON, který je vnitřně ukládán binárně (binary JSON - BSON). Díky tomu lze data z databáze číst a ukládat jako JavaScriptové objekty [31].

3.2.2.3 Firebase

Firebase je tzv. Backend-as-a-Service (Baas). Nejde tedy jen o databázi, ale nabízí řadu připravených nástrojů a služeb pro vývováře, které pomáhají vyvíjet kvalitní aplikace. Jde o službu od společnosti Google. Firebase nabízí dva typy databází: Realtime Database a Firestore Database. Obě se kategorizují jako NoSQL databáze, které ukládájí data do JSON-like dokumentů. Tyto dvě databáze si jsou velmi podobné, ale jak název napovídá, každá je specializovaná trochu jiným směrem. Zatím co Realtime Database je jeden velký JSON strom, Firestore je více strukturovaný na kolekce a dokumenty. Díky tomu je Firestore také lépe škálovatelný a vyhledávací dotazy nad databází jsou rychlé bez ohledu na velikost datasetu. Pro většinu aplikací je vhodnější Firestore, ale pokud aplikace vyžaduje co nejnižší latenci a velký počet jednoduchých dotazů, bude vhodnější Realtime databáze [32].

Firebase nabízí kromě dvou databází také další služby. Jednou z nich je autentizace uživatelů pomocí hesla a emailu, telefonního čísla, Google účtu, Facebookového účtu a dalších podobných služeb. Přihlášení do aplikace pomocí třetí strany je pro uživatele mnohem pohodlnější a rychlejší. Zároveň se díky tomu, že vývojář pracuje jen s Firebase Auth API, vylučuje vznik bezpečnostních chyb při vývoji [33].

Dále také poskytuje hosting pro webové aplikace a Firebase Cloud Messaging (FCM). Jde o službu, která nabízí notifikace napříč platformami [32].

Pro aplikaci Rodinného organizéru byla vybrána technologie Firebase. Na základě funkčních požadavků popsaných v kapitole 4.1 se služby poskytované Firebase velmi dobře hodí pro aplikaci, kterou tato práce popisuje.

3.2.3 CSS framework

Psaní CSS stylů je zdlouhavé a může se velmi rychle stát nepřehledným, pokud se nedodržuje striktní struktura CSS souborů. Existuje více způsobů, jak toto částečně eliminovat [34].

Existují CSS preprocesory, které zpřehledňují kód a zároveň rozšiřují jazyk CSS o nové možnosti. Mezi nejpopulárnější patří LESS a SASS. Jelikož jde o preprocesory, kód se kompiluje na klasické CSS. Řeší sice některé problémy a přidávají nové možnosti pro stylování, ale nevýhodou může být ladění a debugování kódu kvůli kompilaci [35].

Další možností, jak usnadnit a urychlit vývoj, jsou CSS frameworky. Protože jde o knihovnu kódu, která obsahuje běžné webové návrhy, z velké části se eliminuje psaní vlastních kaskádových stylů a využívá se předpřipravených tříd. Pro vývoj to znamená, že styly jsou součástí samotného

HTML nebo šablony pro generování HTML. To urychluje vývoj, prosazuje dobré návyky webdesignu a vytváří čisté a konzistentní stránky [34].

3.2.3.1 Bootstrap

Bootstrap je asi nejznámějším CSS frameworkem. Je postavený v HTML, SASS a JavaScriptu. Zaměřuje se hlavně na responzivní a mobilní frontendový vývoj. Nabízí mnoho předpřipravených rozložení stránek. Díky snadnému použití a velmi návodné dokumentaci je vhodný pro začátečníky v CSS nebo backend vývojáře, kterým umožní vytvořit funkční a pěkné UI bez hlubší znalosti CSS. Minifikovná verze Bootstrapu má velikost 58kB [36].

3.2.3.2 Materialize CSS

Materialize CSS je frontendový framework založený na material designu navržený společností Google. Obsahuje kolekce kompomponent pro uživatelské rozhraní a je kompletně responzivní pro všechna zařízení. Ve frameworku je připraveno mnoho komponent od struktury stránek, přes formuláře až po tlačítka a ikony. Material design kombinuje klasické principy úspěšného designu a technologie. Jeho minifikovaná verze má velikost 175kB [37].

3.2.3.3 Tailwind CSS

Tailwind CSS je mnohem více nízkoúrovňový než předchozí dva frameworky. Nemá tedy předdefinované žádné komponenty. Poskytuje ale všechny potřebné stavební kameny, které jsou potřeba k vytváření návrhu na míru. Tailwind CSS tedy není sám od sebe responzivní, ale má předdefinované breakpointy, kterým vývojář nastaví styly. Jeho minifikovaná verze má velikost jen 1,7kB, což je řádově méně než předchozí dva frameworky [38].

Pro svou aplikaci jsem se rozhodl využít Tailwind CSS, protože umožňuje jednoduše definovat vlastní design, je dostatečně malý a zároveň má výhody výšeúrovňových frameworků.

3.2.4 Progresivní webová aplikace

Progresivní webová aplikace (PWA) je speciální typ webové aplikace využívající technologie moderního webu a moderních webových prohlížečů. Hlavním cílem PWA je přiblížit se nativním aplikacím a poskytnout lepší zážitek, než který poskytuje prohlížení běžného responzivního webu. PWA jsou vyvíjeny již zaběhnutými technologiemi. Nevzniká tak bariéra pro vývojáře, kteří vyvíjí klasické webové aplikace [39].

Vlastnosti, kterými se PWA vyrovnávají nativním aplikacím, jsou především rychlé spuštění, běh aplikace bez přístupu k internetu a podpora notifikací. Uživatel přijde na web, kde aplikace běží. Aplikaci si může v prohlížeči vyzkoušet a poté si ji přidat na plochu zařízení, a to pomocí dialogového okna nebo v kontextové nabídce prohlížeče [40].

Podle [39] a [40] jsou v několika následujících bodech shrnuty podstatné výhody a nevýhody progresivních webových aplikací:

- Výhody:
  - bezpečnost - díky nutnosti HTTPS
  - multiplatformní pužití
  - offline funkčnost
  - levný vývoj
  - rychlost aplikace - díky cachování dat
  - snadná instalace

  - šetří množství stažených dat
  - okamžitý update
- Nevýhody:
  - omezené využití hardwarových prostředků zařízení
  - limitovaná podpora prohlížeče na různých zařízeních
  - nelze aplikaci umístit do AppStore nebo Obchodu play

Podle [39] Google v roce 2015 poprvé použil termín „Progressive Web Apps“ a popsal vlastnosti, které mají mít PWA:

- Progresivní: Fungují všem uživatelům nezávisle na volbě prohlížeče, protože je aplikace vytvořena se zásadou postupného načítání obsahu.
- Responzivní: Zobrazení stránky je optimalizováno pro všechny druhy nejrůznějších zařízení (mobily, notebooky, netbooky, tablety atd.).
- Nezávislé na konektivitě: Díky technologii service workers umožňují používat aplikaci offline nebo na špatném internetovém připojení.
- App-like: Uživatel má při interakci a navigaci pocit, jako by používal nativní mobilní aplikaci.
- Fresh: Obsahují vždy aktuální data díky procesu update technologie service workers.
- Zabezpečené: Obsluha pouze po HTTPS pro zabránění odposlouchávání a změně obsahu načítaných dat.
- Naleznutelné: Díky W3C manifestům jsou identifikovány jako „aplikace“ a registrace service workers napomáhá vyhledávačům k jejich nalezení.
- Znovuzapojení uživatele: Usnadňují znovuzapojení díky funkcím, jako jsou push notifikace.
- Instalovatelné: Umožňují uživatelům „uchovat“ aplikace na jejich domovské obrazovce bez nepříjemného a zdlouhavého procesu instalace z App Store.
- Odkazovatelné: Jednoduše sdílené pomocí URL bez nutnosti složité instalace.

Tento seznam je v několika bodech špatně ověřitelný. Proto vydal Google benchmark Lighthouse, který provede otestování ověřitelných parametrů. Výsledkem je zpráva, která popisuje splněná kritéria a případná doporučení.

3.2.5 Geolokace

Geolocation API se používá k načtení polohy v podobě souřadnic ze zařízení uživatele zavoláním připravené metody `navigator.geolocation.getCurrentPosition()`. Před získáním souřadnic se metoda nejprve dotáže uživatele na povolení sdělení polohy pomocí dialogového okna, až poté vrátí objekt s GPS souřadnicemi [41].

Kapitola 4
Návrh

Začátek této kapitoly je zaměřen na definování funkčních a nefunkčních požadavků na aplikaci. Následuje návrh uživatelského rozhraní a architektury systému, a to na základě analýzy technologií a funkčních požadavků.

4.1 Funkční požadavky

F1 Autentizace — Aplikaci budou moci využívat jen registrovaní a přihlášení uživatelé. Na základě autentizace uživatele bude uživateli umožněn přístup do rodinných kruhů, kterých je členem.

F2 Rozdělení do rodin (rodinných kruhů) — Aby mohl uživatel aplikaci využívat, musí vytvořit rodinný kruh, nebo být do nějakého rodinného kruhu přizván. Rodinným kruhem je myšlena skupina lidí, kteří byli do tohoto kruhu pozváni. Uživatel může být součástí více rodinných kruhů a bude mezi nimi moci přepínat.

F3 Komunikace prostřednictvím chatu — Aplikace bude umožňovat společnou textovou komunikaci členů v jednotlivých rodinných kruzích. Tato komunikace bude viditelná pro všechny členy rodinného kruhu a každý rodinný kruh bude mít svůj chat.

F4 Kalendář — V aplikaci bude umožněno vytvářet události s dnem a časem konání a bude sdílený pro všechny členy rodinného kruhu.

F5 Seznamy úkolů — Uživatelé budou moci v aplikaci vytvářet seznamy úkolů a následně je sdílet s ostatními. Aplikace bude také obsahovat možnost vytvořit nákupní seznam, který bude funkcionálně odvozený od seznamu úkolů. Uživatel bude mít navíc možnost ke každému řádku zadat cenu.

F6 Sdílení polohy — Aplikace bude umožňovat rychlé sdílení polohy ostatním členům rodinného kruhu.

F7 Evidence neproplacených finančních prostředků — Aplikace bude umožňovat uživateli vytvoření požadavku na proplacení částky od jiného člena rodinného kruhu a evidenci předchozích požadavků na proplacení dlužné částky.

4.2 Nefunkční požadavky

N1 Responzivní design aplikace. Uživatelské rozhraní se bude přizpůsobovat velikosti používaného zařízení, aby bylo možné aplikaci používat jak na mobilním zařízení, tak na velkém monitoru.

N2 Webová aplikace bude multiplatformní a spustitelná na jakémkoli zařízení s webovým prohlížečem (OS Microsoft Windows, Linux, MacOS, IOS, Android, ...). Přidání aplikace na plochu (PWA) bude možné na zařízeních, které tuto funkci podporují.

N3 Aplikace bude uživatelsky bezpečná. Hesla budou ukládána pomocí hashovací funkce.

4.3 Návrh uživatelského rozhraní

„*Uživatelské rozhraní (anglicky User Interface, známé pod zkratkou UI) je souhrn způsobů, jakými uživatelé ovlivňují chování systémů.*“ [42] Existuje několik typů uživatelského rozhraní od příkazového řádku, přes grafické uživatelské rozhraní, až po hlasové ovládání nebo virtuální realitu. Vzhled, funkčnost, přehlednost nebo i chaos ovlivňují, zda se uživatel příště vrátí, nebo ne. Návrh, struktura a implementace jsou důležitým faktorem při vzniku jakéhokoli informačního systému. Dle [42], [43] následující výčet popisuje, jaké vlastnosti by mělo mít správně navržené uživatelské rozhraní:

- Srozumitelné – Mělo by uživateli dostatečně napovídat, co která akce způsobí.
- Stručné – Platí „méně je více“, pokud lze použít méně jednodušších prvků, je vhodné to udělat. Mnoho věcí může místo slov říci grafika.
- Povědomé – Uživatelé mají obvykle zafixovány scénáře ovládání různých typů aplikací. Je tedy dobré respektovat zažité principy a postupy. Vymýšlení nových postupů je dobré se vyhnout, nebo s nimi být střídmý.
- Reagující – Uživatelské rozhraní by mělo reagovat na každou akci uživatele. Pokud uživatel klikne na tlačítko, měl by vidět efekt zmáčknutí. Zároveň by rozhraní mělo být rychlé, aby uživatel hned věděl, co se děje.
- Konzistence – Jednotlivé aplikace stejné řady/druhu nebo od stejné firmy využívají podobné uživatelské rozhraní. To pomáhá uživatelům předvídat, jak se bude chovat určitý nástroj v každé aplikaci stejné řady.
- Atraktivní – Uživatelské rozhraní by mělo být přijemné a neobtěžovat. Pokud bude rozhraní ošklivé, je pravděpodobné, že bude běžný uživatel hledat jiné a hezčí. Zároveň ale platí, že obsah a správná funkčnost je důležitější než vzhled.
- Efektivní – Důležitý je odhad, co bude uživatel chtít udělat. Přizpůsobení rozhraní může uživateli nabídnout účinnou zkratku k dosažení požadovaného výsledku rychleji.
- Odpouštějící – Nikdo není neomylný. Občas uživatel udělá něco, co nechtěl a musí se vrátit. Uživatelské rozhraní by mělo nabídnout možnost vrátit chybné kroky zpět.

Podle [44] můžeme webové stránky rozdělit do tří základních skupin podle „výkonu“. Výkonem je myšleno množství potřebných interakcí, které uživatel na webové stránce dělá:

- Webová prezentace – Má za cíl ovlivnit či změnit chování určité skupiny lidí (cílová skupina) a prezentuje určitý produkt, službu a může být spojením pro prodej.
- E-shop – Prodává produkty, nebo služby přímo online. Cílem e-shopu není jen prezentace produktů, ale také jejich přímý prodej zákazníkovi.

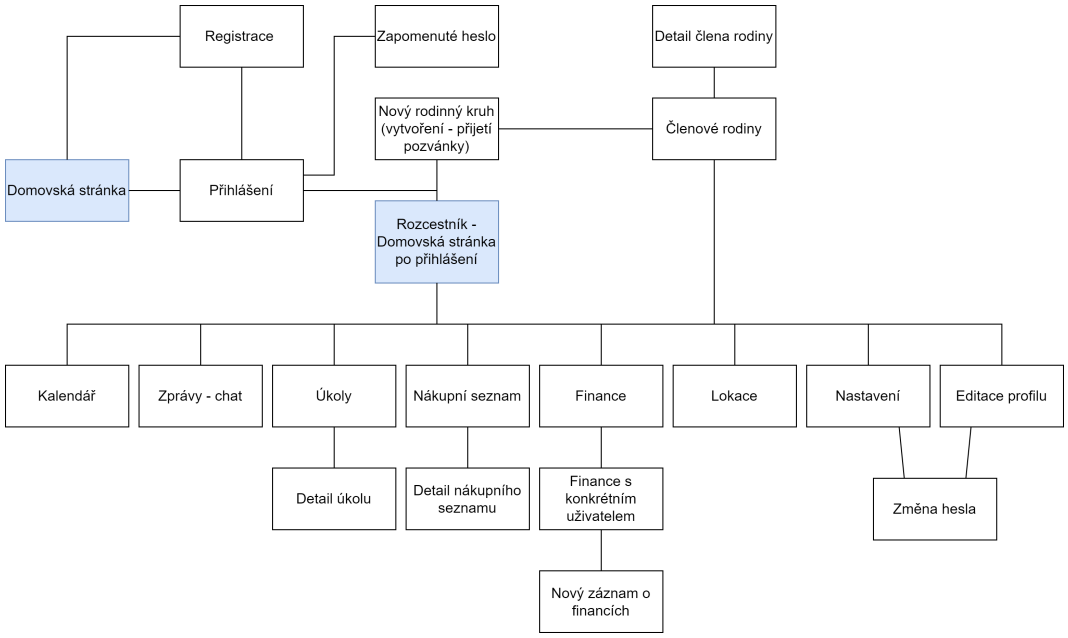
- Webová aplikace – Řeší určitý problém uživatelů prostřednictvím sebe sama. Není kanálem pro prodej, neprodává žádný produkt, ale sama je přímo produktem.

U webových prezentací je nejdůležitější obsah. Právě kvůli němu návštěvníci na webovou prezentaci chodí, a díky tomu se rozhodují, zda uskuteční konverzní akci. Konverzní akce je taková akce, která má přínos pro provozovatele stránky. U webových aplikací jsou klíčové dobře navržené interakce, snadné používání a schopnost podpořit konkrétní procesy, pro které je aplikace navržena. E-shopy jsou z pohledu tohoto rozdělení uprostřed, ale blíže mají k webovým prezentacím [44].

4.3.1 Informační architektura

Na základě těchto informací byla sestavena informační architektura webové aplikace ve formě diagramu. Zmapování informační architektury vytváří „můstek“ mezi grafickou podobou webové stránky a jejími plánovanými funkcionalitami. Pomáhá také v ujasnění výsledné podoby webové stránky a vede k lepší představě o její struktuře [45].

Veškeré funkcionality aplikace jsou uživateli přístupné po přihlášení (případně po registraci). Na diagramu 4.1 je znázorněna struktura navigace v aplikaci. Zvýrazněné části jsou dva hlavní rozcestníky. První z nich je pro všechny uživatele, druhý rozcestník je pro autentizované uživatele. Všechny relace v diagramu jsou oboustranné.



Obrázek 4.1 Diagram informační architektury

4.3.2 Grafický návrh

Tato část se věnuje návrhu obrazovek a vizuální identitě aplikace. Každá jednotlivá obrazovka představuje určitou funkci aplikace podle stanovených funkčních požadavků.

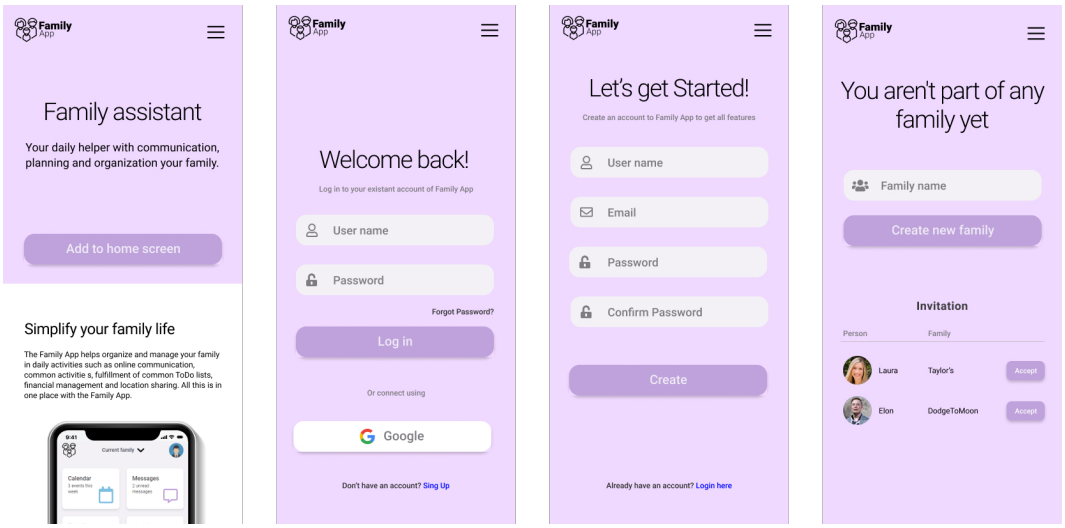
Návrhy obrazovek se liší podle detailnosti zobrazení. Nejméně detailní je takzvaný *wireframe* (v překladu drátěný model), který zobrazuje hlavní části a strukturu obsahu jako obrysy a ukazuje nejzákladnější uživatelské rozhraní. S tímto návrhem se obvykle začíná při návrhu webu nebo aplikace. Jeho největší výhodou je, že se nezabývá estetikou a vizuálem. Jeho tvorba a úpravy

jsou díky tomu velmi rychlé. Další je grafický návrh, který na rozdíl od wireframu ukazuje, jak bude výsledný produkt vypadat. Hlavním cílem grafického návrhu je určit vizuální styl, barevné schéma a typografii produktu. S grafickým návrhem se snadněji experimentuje a provádějí se změny na základě uživatelského testování. Posledním návrhem je prototyp, který má již všechny podstatné rysy výsledného produktu. Na rozdíl od předchozích dvou je prototyp prolinkovaný a dovoluje uživateli interagovat v rozhraní. Zásadní rozdíl mezi prototypem a výsledným produktem je v tom, že rozhraní a backend nejsou svázány. V prototypu jsou tedy jen statická data. Celý tento postup návrhu od wireframu po prototyp se provádí za účelem snížení nákladů na vývoj [46].

U své aplikace jsem se rozhodl přeskočit wireframe a vytvořit rovnou grafický návrh. K tomuto kroku jsem se rozhodl, protože vzhledem k relativně nízkému počtu provedených uživatelských testování by mi tvorba wireframu nepřinesla žádné benefity.

Jelikož aplikace primárně míří na mobilní zařízení, je strategie návrhu mobile first nejlepší volbou. Tato strategie neupřednostňuje mobilní zařízení, ale vyrovnává důležitost všech zařízení na stejnou úroveň. Název je trochu zavádějící, protože v této strategii jde o design nejen pro mobilní telefony, ale i pro všechna ostatní zařízení. Pokud navíc začneme designem pro mobilní zařízení, dostaneme i lepší design pro desktop. Je to zapříčiněno tím, že mobilní zařízení kladou větší nároky na design rozhraní. Mají technická omezení, jako je velikost displeje, rychlost datového připojení nebo objem stažených dat. Mnohem důležitější je ale aspekt uživatelského chování na těchto zařízeních. Uživatel má na mobilním zařízení typicky méně času, nechce moc klikat, nechce vyplňovat zbytečné formuláře a je na mobilním zařízení méně trpělivý [47].

Pro vizuální identitu jsem zvolil font Roboto, který vyvinul Google pro svůj operační systém Android [48]. Dále využívám dva odstíny fialové barvy, tři odstíny šedé a zvolil jsem zaoblený design tlačítek, karet a boxů. Tuto paletu dodržuji napříč celým návrhem pro zachování konzistence. Layout pro desktop bude vycházet z rozložení pro mobilní telefony. Prvky budou na obrazovce rozmístěny tak, aby využily větší dostupnou plochu. Rozhraní pro desktop bude vytvořeno během vývoje.

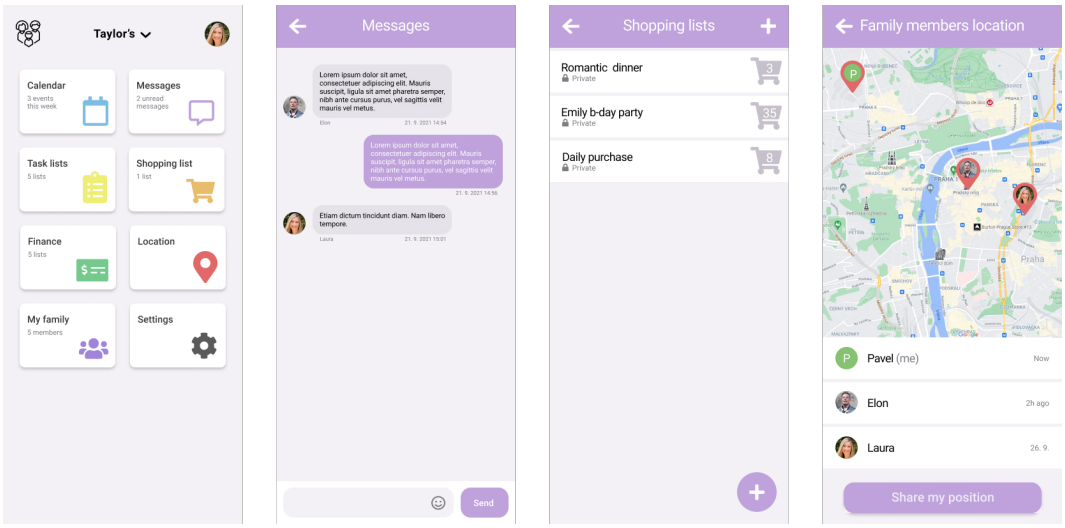


Obrázek 4.2 Grafický návrh obrazovek

Na domovské stránce se nachází krátký popis aplikace a několika funkcionalit, kterými aplikace disponuje. Stránka přihlášení má klasické rozložení a nabízí přihlášení buď emailem a heslem, nebo přes účet Google. Uživatel má hned možnost obnovit heslo, pokud ho zapomněl. Dalším odkazem je nová registrace uživatele. Na stránce s registračním formulářem uživatel vyplní jméno, email a heslo, které musí potvrdit a následně se může registrovat. Poslední z obrazovek na obrázku

4.2 se zobrazí novým uživatelům nebo těm, kteří ještě nejsou součástí žádné rodiny. Uživatel má možnost zde vytvořit novou rodinu, nebo přijmout pozvání do rodiny od jiného uživatele.

Po autorizaci je uživatel přesměrován na stránku s navigací pro jednotlivé funkcionality aplikace viz obrázek 4.3. V horní části je navigace, kde se kliknutím na jméno rodiny rozbalí výběr rodin, kterých je přihlášený uživatel členem. Přes tento výběr je možné přepínat mezi rodinami. V pravém horním rohu je profilová fotka uživatele, která vede na nastavení účtu uživatele. Dále je na této obrazovce osm dlaždic, které vedou na obrazovky s dalšími funkcionalitami. Druhá obrazovka na obrázku 4.3 je chat, který kopíruje zaběhnutý layout populárních chatovacích aplikací. Na další obrazovce je výpis nákupních seznamů. Boxy s názvem, viditelností pro ostatní a počtem položek vedou na detail nákupního seznamu. Po rozkliknutí vidíme jednotlivé položky nákupního seznamu a zaškrtávací políčka. V pravé dolní části je pak tlačítko pro vytvoření nového nákupního seznamu. Poslední obrazovka ukazuje na mapě, kde členové rodiny naposledy sdíleli svoji polohu. Pod mapou je výpis členů rodiny s údajem, kolik času uplynulo od posledního sdílení polohy. Po kliknutí na položku tohoto výpisu se na mapě zaměří a přiblíží poloha na daného uživatele. Tlačítkem „Share my position“ dojde ke sdílení aktuální polohy uživatele.



Obrázek 4.3 Grafický návrh obrazovek 2

Více návrhů obrazovek je přiloženo v příloze A. Obrazovky výsledné aplikace jsou přiloženy v příloze B.

Kapitola 5

Realizace

V této kapitole jsou shrnuty praktické postupy použité při implementaci a použití technologií rozebraných v kapitole 3. Dále je zde popsána realizace jednotlivých částí aplikace, struktura projektu a problematika PWA.

5.1 Struktura projektu

Adresářová struktura projektu je znázorněna na diagramu 5.1. Tato struktura vychází z výchozí stromové struktury React aplikace vytvořené pomocí `npm` příkazem `npx create-react-app my-app`.

Složka `build` obsahuje vygenerované soubory pro nasazení na server. Tyto soubory jsou minifikované a optimalizované tak, aby neobsahovaly žádný zbytečný kód.

Adresář `node_modules` obsahuje nainstalované závislosti projektu včetně jejich zdrojových souborů.

V adresáři `public` je umístěný základní `index.html`, který je prohlížeči dostupný a pomocí kterého framework manipuluje s DOM. Tento HTML dokument se také uživateli zobrazí, pokud bude mít vypnutý JavaScript na klientské straně. Ve složce jsou dále statické soubory, jako jsou obrázky, ikony a favicony. Mezi tyto statické soubory potří také service worker (`firebase-messaging-sw.js`) pro správu příchozích notifikací a soubor `manifest.json`, který poskytuje webovému prohlížeči informace o aplikaci.

Ve složce `src` se nachází všechny zdrojové soubory, ze kterých je sestaven build aplikace. Hlavní komponenta je uložena v souboru `App.js`, ve které je router, pomocí kterého jsou jednotlivým URL adresám přiřazena daná *views*. Tato *views* jsou uložena ve složce `webpages` a mohou se skládat z více komponent. Tyto komponenty jsou uloženy v adresáři `components`, nebo v jeho jednotlivých podadresářích. Soubor `index.js` obsahuje základní inicializaci React aplikace a hlavní komponenty `App.js`. V souboru `firebase.js` je inicializace firebase a notifikací. O odesílání notifikací se starají JS funkce v souboru `Notification.js`. Sestavení service workeru pomocí rozšíření Workbox se nachází v souboru `sw.js`.

Soubor `.env` slouží pro nastavení konfiguračních informací pro aplikaci a pro Firebase. V souboru `firebase.rules` jsou nastavena pravidla pro přístup do Firestore. V `package.json` jsou obsaženy závislosti aplikace. Poslední dva soubory, `tailwind.config.js` a `workbox-config.js`, jsou konfigurační pro Tailwind CSS a Workbox.

```
/
├── build ..... adresář se sestaveným produkčním kódem aplikace
├── node_modules
├── public ..... adresář se soubory a obrázky
│   ├── images ..... adresář se statickými obrázky
│   ├── firebase-messaging-sw.js ..... service worker pro push notifikace
│   ├── index.html ..... adresář se statickými obrázky
│   └── manifest.json ..... json soubor s informacemi o aplikaci
├── src ..... zdrojové kódy implementace
│   ├── auth ..... adresář se soubory pro autentizaci uživatele
│   ├── components ..... adresář se soubory jednotlivých React komponent
│   ├── database ..... adresář s funkcemi pro komunikaci s Firebase
│   ├── firestore-test ..... adresář s testy
│   ├── hooks ..... adresář s React hooks
│   ├── images ..... adresář s obrázky a ikonami
│   ├── style ..... adresář s kaskádovými styly
│   ├── validation ..... adresář s funkcemi pro validaci vstupů od uživatele
│   ├── webpages ..... adresář s komponentami pro celé stránky
│   ├── App.js ..... hlavní ReactJS komponenta
│   ├── index.js ..... inicializace ReactJS aplikace
│   ├── firebase.js ..... inicializace Firebase
│   ├── Notification.js ..... odesílání požadavku na notifikaci
│   ├── sw.js ..... inicializace Workbox pro service worker
│   └── webpages ..... adresář s komponentami pro celé stránky
├── .env ..... soubor pro nastavení parametrů aplikace
├── firebase.json ..... konfigurační soubor pro Firebase
├── firestore.rules ..... soubor s pravidly pro data ve Firestore
├── firestore.indexes.json ..... soubor s konfiguracemi pro indexování
├── package.json ..... závislosti
├── tailwind.config.js ..... konfigurační soubor pro tailwind
├── workbox-config.js ..... konfigurační soubor pro Workbox
└── README.md
```

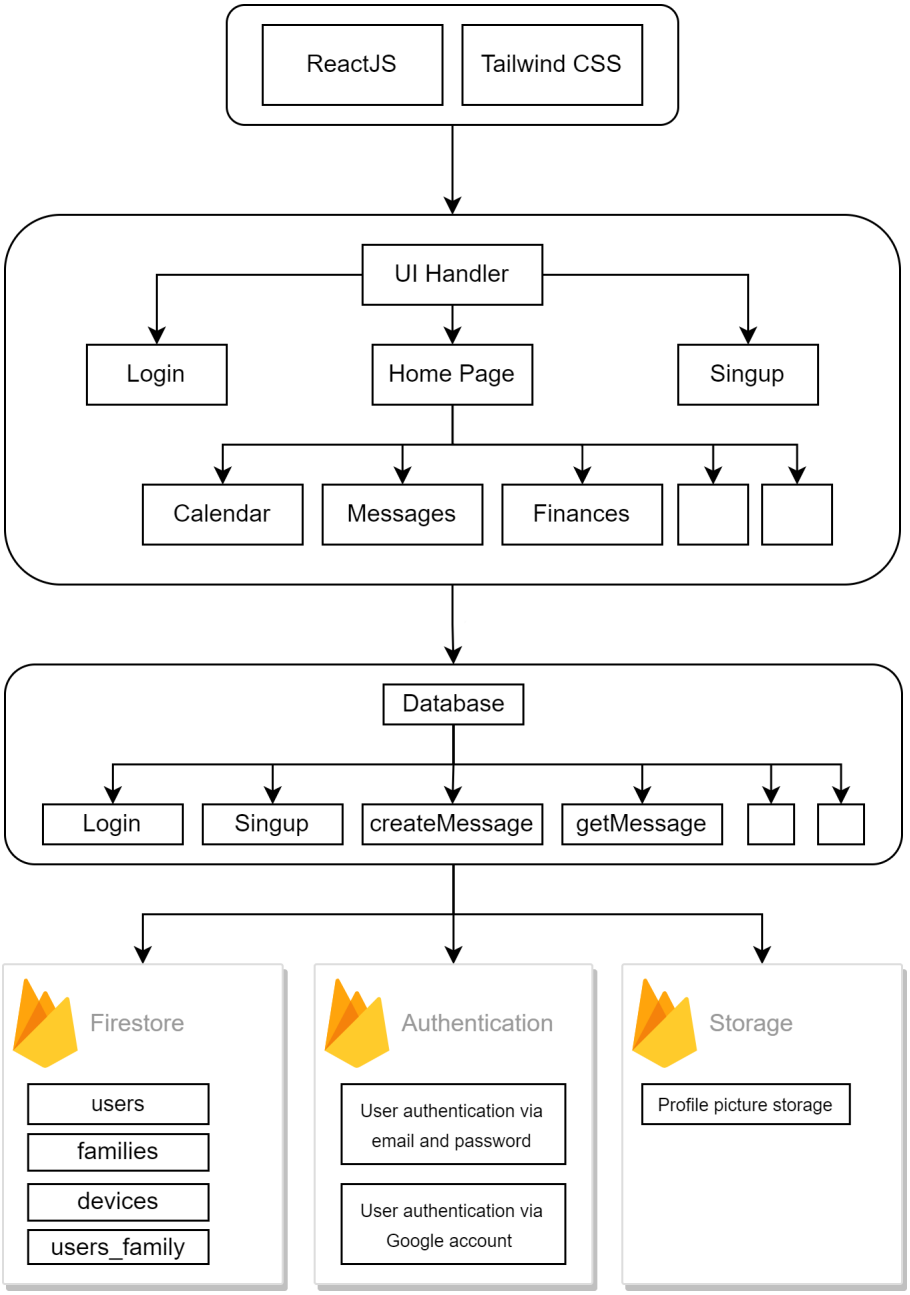
Obrázek 5.1 Adresářová struktura

5.2 Architektura

Na diagramu 5.2 je architektura celé aplikace. Architektura je rozdělena do tří částí. První část generuje uživatelské rozhraní pomocí JavaScriptového frameworku ReactJS a CSS frameworku Tailwind CSS. Druhá část se stará o komunikaci s balíkem služeb Firebase. Poslední částí je služba Firebase, která zde slouží jako backend.

Uživatelské rozhraní a jednotlivé stránky jsou složeny z komponent frameworku ReactJS. Pro komunikaci s Firebase slouží vrstva v adresáři `database`, ve které jsou připravené JavaScriptové funkce.

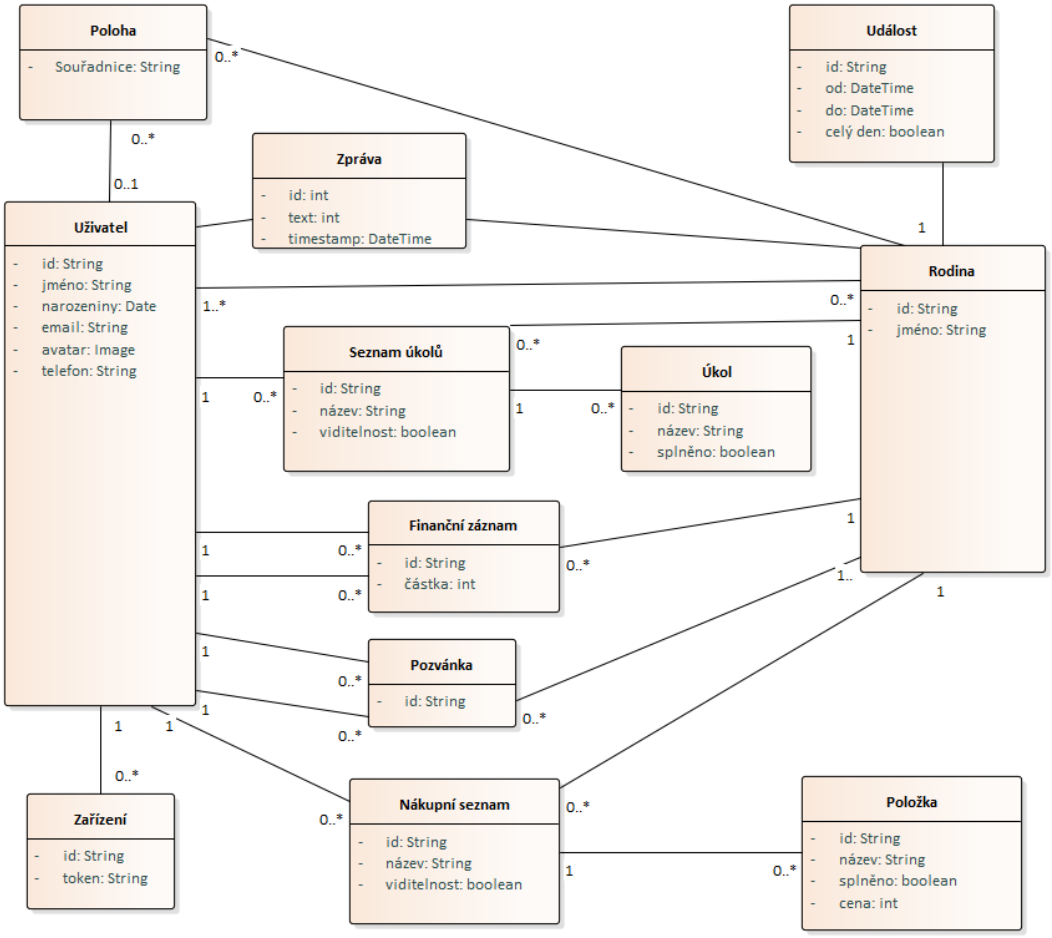
Z balíku služeb je využívána Firestore databáze, ve které jsou ukládána data. K autentizaci je také využívána Firebase, uživatel se může registrovat/přihlásit emailem a heslem, nebo pomocí již existujícího Google účtu. Poslední službou je Storage, která se využívá k ukládání profilových obrázků registrovaných uživatelů.



Obrázek 5.2 Diagram architektury aplikace

5.3 Doménový model

Na doménovém modelu 5.3 jsou znázorněny jednotlivé komponenty. Tyto komponenty jsou znázorněny jako diagram tříd, které reprezentují data v aplikaci a jejich vzájemné vazby.



Obrázek 5.3 Doménový model aplikace

5.4 Rozšíření frameworku

Jelikož existuje mnoho již hotových a otestovaných dílčích řešení, jsou některá využita v této aplikaci. V podkapitolách jsou popsány významné závislosti a hotové balíčky třetích stran použité při vývoji aplikace.

5.4.1 Formik

Jde o jednu z nejoblíbenějších knihoven pro tvorbu formulářů v ReactJS. Formik poskytuje několik komponent, které šetří čas při vývoji tím, že redukují duplicitní kód a dělají kód více konzistentní a organizovaný.

Formik je v aplikaci využit pro každý vstup od uživatele, kde obstarává načítání hodnot, validuje data již při psaní do formuláře a vyřizuje odeslání dat z formuláře.

5.4.2 MomentJS

JavaScript nabízí velmi jednoduché a omezené možnosti práce s časem a datumy. Existuje mnoho knihoven s pokročilými funkcemi, mezi nejznámější patří MomentJS.

MomentJS je v aplikaci využit pro syntaktickou analýzu datumu a času ze vstupu formuláře a také formátování datumu s časem pro uložení do databáze. Další funkční vlastnotí, která je v aplikaci využita, je doba trvání od posledního sdílení polohy.

5.4.3 Big calendar

Big calendar je komponenta frameworku ReactJS, která poskytuje kalendář s událostmi. Tato kalendářová komponenta vyžaduje jednu z těchto knihoven: Moment.js, Globalize.js nebo date-fns pro správnou lokalizaci.

Komponenta umožňuje pomocí parametrů nastavení širokého spektra možností pro customizaci vykreslení a funkcionalit kalendáře. Jedním z mnoha parametrů je pole událostí, kde položky obsahují informace o začátku a konci události. Dále obsahuje informaci, zda jde o celodenní událost, název události, autora a barvu dané události.

Kalendář je díky flexbox stylům plně responzivní na všech potřebných zařízeních. Pro kalendář jsou nastaveny tři možnosti zobrazení: měsíční, týdenní a denní. Zadávání a úprava událostí je řešena pomocí modálu. Jakmile je událost vytvořena, zobrazí se v kalendáři všem členům rodiny. Všichni uživatelé mohou události upravovat a vytvářet, ale jen autor dané události ji může smazat.

5.4.4 Toastify

Toastify je v aplikaci využíván pro zobrazení notifikací přímo v aplikaci. Pomocí komponenty `ToastContainer` se nastaví atributy, vzhled a umístění, ve které komponentě se budou notifikace zobrazovat. K zobrazení notifikací pak jen stačí zavolat funkci `toast("Text notifikace");`.

5.5 Firebase

V následujících podkapitolách jsou rozebrány služby Firebase využité v této aplikaci. Jde o služby Firestore pro ukládání dat, Firebase Authentication pro ověřování identity uživatelů a Firebase Storage pro ukládání profilových obrázků uživatelů.

5.5.1 Firestore

Při implementaci je důležité rozvrhnout si strukturu kolekcí v databázi. Schéma může být vytvořeno striktně, kdy pomocí pravidel nastavíme přesnou strukturu a typ dat, nebo můžeme do databáze ukládat různě strukturovaná data.

Pro svou implementaci jsem si vybral striktní strukturu, a to z důvodu jednodušší správy dat a validace na straně Firestore. V projektu je pět hlavních kolekcí:

- *devices*
- *users*
- *users_family*
- *invitations*
- *families*

V kolekci *devices* jsou ukládány tokeny, které slouží k identifikaci a odesílání notifikací do zařízení. Tyto tokeny jsou zároveň identifikátorem záznamu v databázi, tím je zajištěna unikátnost záznamu s daným tokenem. Společně s tokenem je uloženo ID uživatele.

Kolekce *users* obsahuje povinné a nepovinné informace o registrovaných uživatelích. Mezi povinné patří *email*, *name*, *uid* a *authProvider*. Mezi nepovinné patří *avatar*, *birthday*, *current_family* a *tel*.

Kolekce *users_family* reprezentuje vztah mezi jednotlivými uživateli a rodinami. Má tři atributy *family_id*, *family_name* a *user_id*.

Pozvánky do rodin jsou ukládány v kolekci *invitations*. Kolekce obsahuje informace *creator_user_avatar*, *creator_user_id*, *creator_user_name*, *family_id*, *family_name* a *invited_user_id*.

Poslední hlavní kolekcí je *families*. V této kolekci jsou uchovávány informace *id*, *name* a *owner*. Zároveň tato kolekce obasahuje šest subkolekcí:

- *events*
- *finances*
- *locations*
- *messages*
- *shoppings*
- *tasks*

V těchto subkolekcích jsou uložena data k jednotlivým funkcionalitám aplikace. Kolekce *events* reprezentuje události v kalendáři. V kolekci *finances* jsou uloženy informace o financích mezi uživateli. Poloha je ukládána v kolekci *locations*, jelikož se uchovává jen poslední sdílená poloha, je identifikátorem záznamu ID uživatele. V kolekci *messages* jsou uloženy zprávy chatu v rodině. Kolekce *shoppings* a *tasks* jsou podobné. Obě obsahují jednu subkolekci *items*, která reprezentuje položky seznamu. Liší se v ukládaných informacích.

Firestore nabízí dvě možnosti, jakým způsobem lze načíst uložená data. První možností je klasický `get()`. Ten načte a vrátí obsah dokumentu pouze jednou. Pokud později dojde ke změně v záznamu v databázi, zůstane výsledek beze změny. Pro získání nových dat je potřeba znovu zavolat `get()`. Druhou možností je metoda `onSnapshot()`. Tato metoda po zavolání vrátí aktuální data. Vždy když dojde ke změně v datech, automaticky aktualizuje dříve načtená data. Díky tomu je zajištěna automatická okamžitá synchronizace dat pro všechny uživatele bez nutnosti přenačtení stránky.

Firestore není vhodný pro ukládání dat, která jsou mnohočetně provázaná. Vyhledávání a filtrování v této databázi je omezené a nedovoluje skládat složitější dotazy nad daty. Navíc většina dotazů, která není jednoduchý `get()`, vyžaduje dodatečné idexování. Pokud je pro dotaz toto indexování vyžadováno, vypíše se v konzoli prohlížeče chybová hláška a odkaz s předvyplněným požadovaným indexováním, jako je na obrázku 5.4.

Obrázek 5.4 Chybová hláška chybějící indexace ve Firebase

5.5.2 Autentizace

Firebase nabízí snadnou a bezpečnou autentizaci uživatelů. Podporuje více možností autentizace, jako je telefonní číslo, email a heslo, Google účet, Facebook, Twitter a další.

Přihlašování prostřednictvím třetích stran je pro uživatele pohodlnější a rychlejší. Proto je v aplikaci implementována registrace a přihlašování nejen pomocí emailu a hesla, ale i přes Google účet.

Pokud si uživatel vybere autentizaci pomocí emailu a hesla, aplikace vyžaduje ověření emailu. Uživateli přijde email s odkazem, prostřednictvím kterého se provede ověření emailu.

Na výpisu kódu 5.1 je ukázána funkce pro přihlášení Google účtem. Pro autentizaci třetí stranou není rozlišováno mezi přihlášením a registrací. Pokud jde o nového uživatele, vytvoří se v databázi záznam s novým uživatelem a dojde k přihlášení uživatele.

Výpis kódu 5.1 Funkce pro přihlášení prostřednicvím Google účtu

```
function logInWithGoogle() {
    let provider = new firebase.auth.GoogleAuthProvider();
    firebase.auth().signInWithPopup(provider).then((cred) => {
            if(cred.additionalUserInfo.isNewUser){
                firebase.firestore().collection('users')
                .doc(cred.user.uid).set({
                    uid: cred.user.uid,
                    name: cred.additionalUserInfo.profile.given_name,
                    avatar: cred.additionalUserInfo.profile.picture,
                    email: cred.user.email,
                    birthday: "",
                    tel: "",
                    authProvider: "google.com"
                }).then()
            }
            getToken().then((token) => {
                saveToken(cred.user, token);
            });
        }).catch((err) => {
            console.error(err);
        });
}
```

5.5.3 Ukládání souborů

Firebase poskytuje také službu Cloud Storage, která je v aplikaci využívána jako úložiště pro profilové obrázky uživatelů. Po nahrání profilového obrázku je k danému uživateli uložena URL adresa do databáze.

5.6 PWA

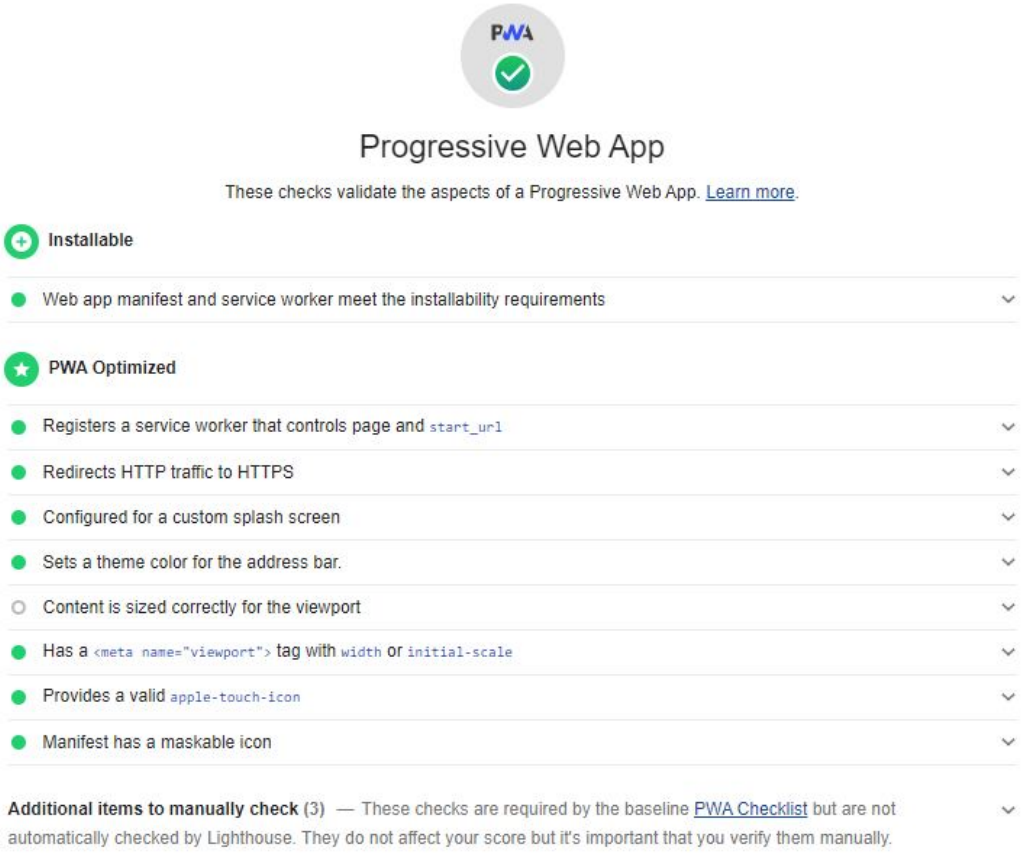
Podle informací z podkapitoly 3.2.4 patří mezi hlavní vlastnosti PWA instalovatelnost a funkčnost offline.

Aby prohlížeče nabídly uživateli možnost aplikaci nainstalovat, musí splňovat několik následujících požadavků. Webová stránka musí být dostupná přes protokol HTTPS, pro webobové prohlížeče musí být dostupný `service worker`, který zajistí offline funkčnost. Posledním požadavkem je přítomnost souboru `manifest.json`, ve kterém jsou popsány informace o aplikaci. Jednou z povinných informací je cesta k ikoně aplikace. Kód souboru manifest.json pro tuto aplikaci je ukázán na výpisu kódu 5.2

Výpis kódu 5.2 manifest.json použitý v aplikaci

```
{
    "short_name": "Family app",
    "name": "Family organizer app",
    "description": "Family assistant - Your daily helper
    with communication, planning and organization your family.",
    "icons": [
      {
        "src": "images/icon-512x512.png",
        "sizes": "512x512",
        "type": "image/png"
      }
    ],
    "start_url": "/dashboard",
    "display": "standalone",
    "theme_color": "#BFA2DB",
    "background_color": "#BFA2DB"
  }
```

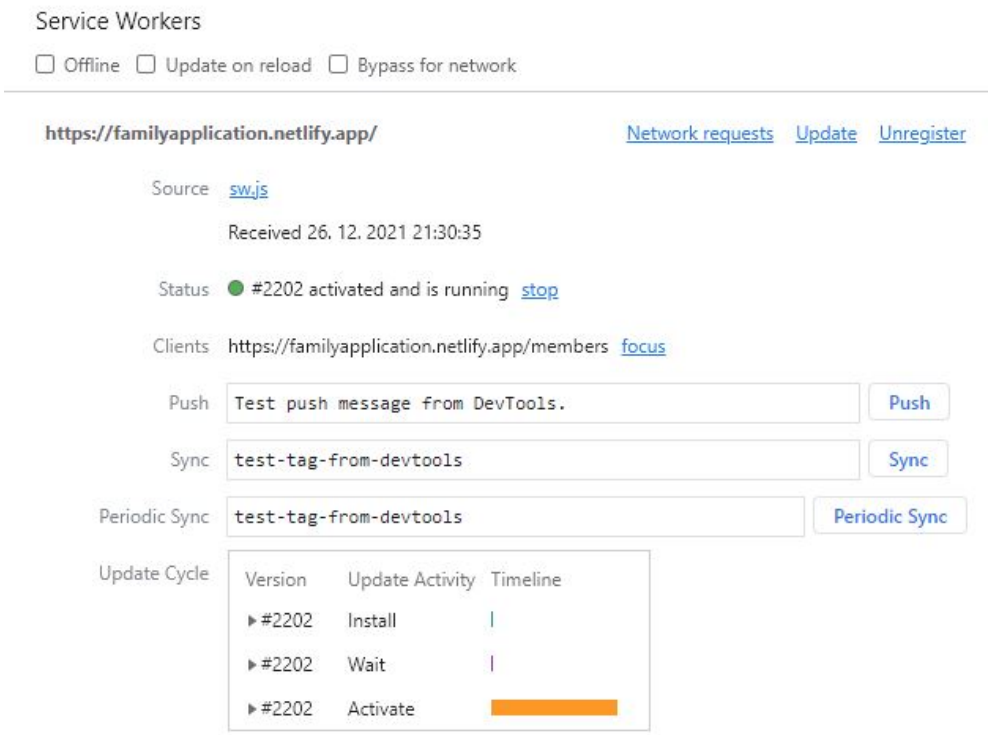
Aplikace byla otestována nástrojem Lighthouse. Z výsledého reportu dle obrázku 5.5 byla aplikace klasifikována jako PWA ve všech možných testovaných bodech.



Obrázek 5.5 Lighthouse report - klasifikace jako PWA

5.6.1 Service worker

Pro PWA aplikaci je důležité, aby aplikace fungovala i offline bez přístupu k internetu. Po prvotním načtení stránky dojde k uložení všech potřebných souborů a zdrojů do takzvané `cache storage`. O toto uložení se stará service worker, který se zaregistruje v prohlížeči viz obrázek 5.6.



Obrázek 5.6 Zaregistrovaný service worker

Následně při načítání webové stránky zajišťuje, aby se načetly tyto uložené zdroje. Existuje více strategií správy `cache storage`:

- Stale-While-Revalidate
- Cache First (Cache Falling Back to Network)
- Network First (Network Falling Back to Cache)
- Network Only
- Cache Only

V aplikaci je využita strategie `Cache First`. V této strategii se nejprve využívají data uložená v `cache storage` a pokud nejsou k dispozici, nebo vypršela jejich platnost, využije se načtení ze serveru. Tato strategie načte uložené zdroje nejen pokud je uživatel offline, ale i když je jeho internetové připojení pomalé. Stránka se vždy načte korektně a rychle.

Pro tuto aplikaci je využito rozšíření Workbox, které usnadňuje správu service wokeru a ukládání do `cache storage`. V konfiguračním souboru se nastaví parametry a Workbox vygeneruje service worker. Konfigurační soubor Workboxu použitý pro tuto aplikaci je v ukázce kódu 5.3.

Výpis kódu 5.3 Konfigurační soubor Workbox

```
module.exports = {
    globDirectory: 'build/',
    globPatterns: [
        '**/*.{json,svg,jpg,png,html,css,js,txt}'
    ],
    maximumFileSizeToCacheInBytes: 5000000,
    swDest: 'build/sw.js',
    swSrc: "src/sw.js"
};
```

5.6.2 Data z databáze offline

Service worker v tomto nastavení ale neukládá data z databáze. Veškerý obsah, který je dostupný z Firebase Firestore, je potřeba také ukládat na klientově straně. K tomu slouží vestavěná funkce `enablePersistence()` dostupná z knihovny Firebase viz výpis kódu 5.4.

Výpis kódu 5.4 Firebase persistence

```
firebase.initializeApp(firebaseConfig);
const fireDb = firebase.firestore();
const messaging = firebase.messaging();

fireDb.enablePersistence().catch((err) => {
    if (err.code === 'failed-precondition') {
        console.log("Multiple tabs open, persistence can only be
        enabled in one tab at a time");
    } else if (err.code === 'unimplemented') {
        console.log("The current browser does not support
        all of the features required to enable persistence");
    }
});
```

Pokud je tato funkce zavolána po inicializaci Firebase, uloží odpovědi na požadavky API Firebase do `IndexedDB`. Díky této funkci může uživatel aplikaci využívat bez připojení k internetu stejně, jako kdyby byl online. Veškeré interakce s databází jsou uloženy lokálně a po připojení k internetu dojde k synchronizaci dat se serverem. Jestliže je tedy uživatel offline a přidá například událost do kalendře, událost se zobrazí v kalendáři stejně, jako by byl online. Může ji upravit nebo i smazat. Po připojení k internetu se tato událost zobrazí ostatním uživatelům.

5.6.3 Lighthouse

Pro otestování splnění kritérií aplikace PWA slouží nástroj Lighthouse. Nástroj je dostupný přímo ve webovém prohlížeči Google Chrome, případně je možné ho nainstalovat příkazem `npm install -g lighthouse`. Aplikace podle tohoto testu splňuje podmínky PWA aplikace.

5.7 Notifikace

Notifikace jsou přímo v aplikaci zobrazovány pomocí knihovny Toastify viz kapitola 5.4.4. Pokud ale není aplikace aktivní, notifikace musí přijít přímo do zařízení uživatele. Příjem notifikace je tedy možný dvěma způsoby.

Pro příjem notifikace bez aktivní aplikace je nutné mít ve webovém prohlížeči registrovaný service worker, který notifikaci přijme. Kód service workeru použitý v této aplikaci je ve výpisu

kódu 5.5. Tento service worker využívá knihovnu Firebase Cloud Messaging (FCM). Proměnná `firebaseConfig` obsahuje přístupové tokeny, kterými se inicializuje Firebase. O zobrazení notifikace se stará metoda `onBackgroundMessage`, ve které se nakonfiguruje tělo a ikona notifikace.

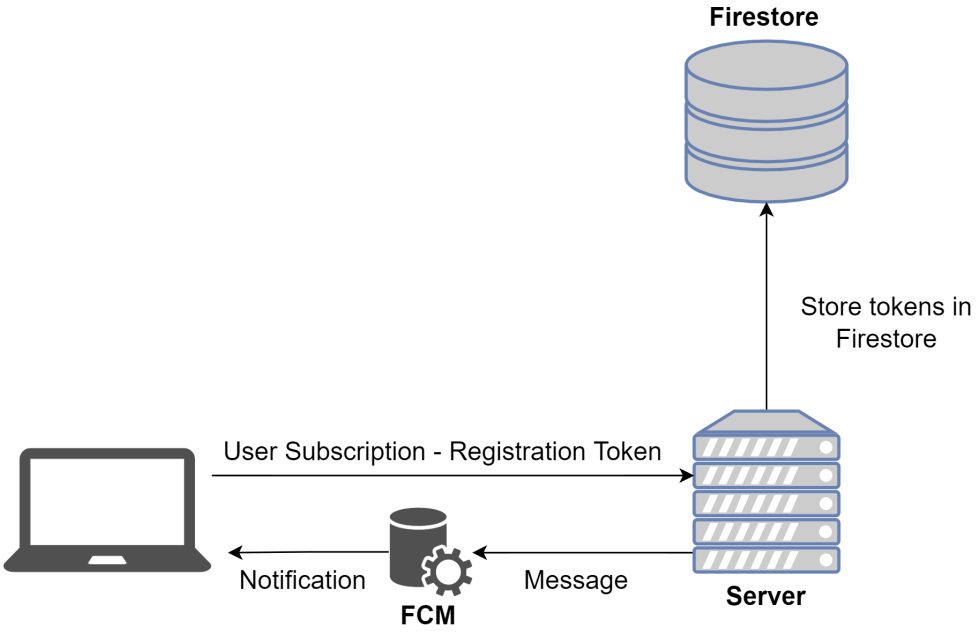
Výpis kódu 5.5 Service worker pro příjem notifikací na pozadí

```
var firebaseConfig = {
    ...
};
firebase.initializeApp(firebaseConfig);
const messaging = firebase.messaging();

messaging.onBackgroundMessage(function(payload) {
    console.log('Received background message ', payload);

    const notificationTitle = payload.notification.title;
    const notificationOptions = {
        body: payload.notification.body,
        icon: "./images/Logo-app.png"
    };
    return self.registration.showNotification(
        notificationTitle,
        notificationOptions);
});
```

Pro odeslání notifikace je potřeba token, který je unikátní pro každé zařízení. Tento token je nejprve nutné získat. Získává se hned po přihlášení uživatele, jak je vidět ve výpisu kódu 5.1 funkcí `getToken`. Token zařízení je vždy unikátní a je použitý i jako identifikátor záznamu v databázi. Společně s tokenem zařízení je ukládáno ID uživatele. Tento přístup eliminuje příjem duplicitních notifikací, pokud by se na jednom zařízení vystřídalo více unikátních uživatelů. Proces je znázorněn na diagramu 5.7.



Obrázek 5.7 Diagram Firebase Cloud Messaging

Pokud někdo rodinu opustí, je přijat nový člen, nebo je rodina zrušena, notifikaci obdrží všichni členové rodinného kruhu. V případě vyrovnání financí mezi dvěma uživateli přijde notifikace tomu z uživatelů, který neprovedl akci pro jejich vyrovnání.

Odeslání samotné notifikace je prováděno prostřednictvím Rest API, které poskytuje FCM. Pokud je provedena akce, která má odeslat notifikaci, je odeslán požadavek typu POST na URL `https://fcm.googleapis.com/fcm/send`. V hlavičce je autorizační token a v těle je token určující zařízení, kterému je notifikace určena a nadpis s tělem notifikace.

5.8 Sdílení polohy

Pro sdílení polohy je použito Geolocation API, které je popsáno v kapitole 3.2.5. Získaná poloha je uložena do databáze a vždy se uchovává pouze poslední sdílená poloha. Pro zobrazení polohy na mapě bylo využito mapových podkladů Mapbox, protože nejpoužívanější mapové podklady od Googlu (Google Maps API) nejsou od roku 2018 poskytovány zdarma..

Pro vykreslení mapy je využito hnihovny React Map GL, která podporuje všechny potřebné funkce. Na mapě se zobrazí aktuální poloha, pokud uživatel povolí přístup k poloze ve svém prohlížeči. Mapa je plně interaktivní a podporuje všechny známé ovládací prvky (zoom, posouvání, tlačítko pro přesun na aktuální polohu) napříč zařízeními.

React Map GL disponuje komponentou `Marker`, která vykreslí dané HTML elementy na zadaných souřadnicích. V tomto případě se vykresluje obrázek (bod), který znázorňuje pozici na mapě.

Přechod mezi pozicemi ostatních členů je realizován pomocí seznamu pod mapou. Po kliknutí na položku seznamu (člena rodiny) dojde k přenastavení State Hooku, který nese informaci o rozměrech mapy, souřadnicích a přiblížení. Při změně je možné nastavit funkci, která provede plynulý přechod mezi dvěma souřadnicemi. Přechodové funkci se nastavuje také doba přechodu v milisekundách.

5.9 Spuštění aplikace

Výsledné produkční sestavení aplikace se nachází na přiloženém mediu B v adresáři `/build`. Pro spuštění je potřeba obsah adresáře nahrát na webový server. Pro správné fungování je také nutné mít vytvořenou službu Firebase a v souboru `/.env` nastavit přístupové tokeny a údaje. Tyto údaje se musí vyplnit i v souboru `/public/firebase-messaging-sw.js`, který slouží pro příjem notifikací.

Pokud dojde ke změně konfigurace, je zapotřebí znovu sestavit produkční aplikaci příkazem `npm run build`. Jestliže je nastavena nová konfigurace, je nezbytné nahrát do prostředí Firebase Rules soubor `firestore.rules`, ve kterém jsou definována pravidla pro správu a validaci dat ve Firestore. Jako poslední je třeba nahrát příkazem (je nutné mít nainstalovaný balíček Firebase CLI) `firebase deploy --only firestore:indexes` seznam indexů nutný pro správné fungování dotazů nad databází Firestore. Případně lze tyto indexy nastavit ve webovém prostředí Firebase Firestore Indexes podle přiloženého souboru `firestore.indexes.json`. Aplikace je dostupná také na adrese `https://familyapp.online/`.

Kapitola 6
Testování

Pojmem testování se rozumí více různých činností. V tomto případě půjde o testování webové aplikace. Tyto činnosti se dají rozdělit mnoha možnými způsoby. Nejvíce obecné dělení je na automatické testy prováděné strojově a uživatelská testování, která se zabývají nejen chybami v programu, ale i nevhodným návrhem uživatelského rozhraní. Testování je často opomíjenou, ale velmi důležitou součástí procesu vývoje jakéhokoliv softwaru. Oba tyto přístupy se navzájem nevylučují, naopak se doporučuje provádět oba tyto druhy testování v průběhu celého vývojového procesu. Pokud testování provádíme správně a dostatečně často, energie vložená do psaní automatických testů i čas strávený uživatelským testováním se nám do konce vývoje vrátí. Díky těmto testům můžeme odhalit chyby v kódu nebo chybný návrh uživatelského rozhraní hned v počátku. V následujících sekcích této kapitoly lze nalézt bližší popis a postupy jednotlivých testování [49].

6.1 Uživatelské testování

Základem uživatelského testování jsou lidé, kteří se nepodílejí na vývoji aplikace, ani na jejím návrhu. Člověk (dále „respondent“), se kterým provádíme testování, uvidí samotné uživatelské rozhraní v ideálním případě „poprvé v životě“ až při samotném testování. Tím se předejde zkreslení, které vznikne, pokud by respondent uživatelské rozhraní nebo aplikaci předem znal. Typickým příkladem nevhodného respondenta je vývojář aplikace, který může, třeba i nevědomě, bránit „svůj výtvor“ a přehlížet chyby nebo nelogičnosti. Dalším nevhodným příkladem je respondent, který provádí stejné testování opakovaně. V tomto případě je zkreslení způsobeno znalostí procesů aplikace. Ideálními příklady respondentů jsou lidé napříč celou skupinou, na kterou cílí moje vyvíjená aplikace. Adekvátní počet respondentů a iterací je vzhledem k rozsáhlosti uživatelského rozhraní a množství funkcionalit velmi individuální. Většinou se vychází z nepsaného pravidla, které říká, že už od tří respondentů jsou výsledky testování relevantní [49].

6.1.1 Příprava na testování s uživatelem

Dle [50] stačí na základní testování 10 až 20 hodin času a jsou potřeba čtyři věci:

- uživatel z cílové skupiny
- prostory (ale velmi dobře to jde i online)
- scénář uživatelského testování
- interaktivní („klikatelný“) prototyp uživatelského rozhraní, nebo již plně funkční aplikace

Nemůžeme testovat uživatelské rozhraní pro výukový program dětí na prvním stupni na studentech konzervatoře. Respektive můžeme, ale velmi pravděpodobně nezjistíme to, co bychom potřebovali. Výsledky testování jsou přímo závislé na výběru respondentů, proto je vhodné volit respondenty z cílové skupiny, a pokud je různorodá, je dobré vybrat respondenty napříč celým jejím spektrem. Různí respondenti mají odlišné zkušenosti a nápady [50].

Testovat lze v podstatě kdekoli – kancelář, kavárna i online prostředí. Ideálním případem jsou speciální místnosti pro uživatelské testování, které jsou přizpůsobené k tomu, aby z testování vzešlo co nejvíce relevantních dat. V těchto místnostech bývají instalovány kamery, které zabírají respondenta v průběhu testování z více úhlů a vyhodnocuje se každý jeho pohyb. Pro běžné testování stačí počítač s programem, který zaznamenává dění na obrazovce a místnost, kde není průběh testování rušen vnějšími vlivy. Pro testování online je potřeba nástroj, který umožňuje sdílet pracovní plochu a nahrávat záznam [49].

Při uživatelském testování je klíčová příprava a ujasnění si, co potřebujeme testováním zjistit. Pro respondenty je potřeba připravit si určité scénáře a úkoly, které se pokusí splnit jeden po druhém. Scénář by měl aspoň částečně kopírovat případy užití definované při návrhu aplikace. Pokud diagram případů užití (nebo jen případy užití) není z jakéhokoli důvodu k dispozici, je nutné vytvořit si scénář od základu. Jelikož se jedná o testy zaměřené na správný návrh rozhraní a schopnost uživatelů orientovat se v tomto rozhraní, neměl by být scénář příliš popisný a návodný. Respondent by tedy neměl dostávat konkrétní instrukce jako „klikněte vlevo nahoře na logo“ nebo „přesuňte se dolů a přejeďte přes obrázek“ atp. Vhodnější jsou scénáře, které mají obecný cíl. V průběhu plnění respondentem se sleduje, jak si počíná, na jaké problémy narazil, případně se respondenta můžeme doptávat na detaily. Pokud respondent nebude vědět, jak pokračovat v plnění scénáře, je důležité nenechat ho dlouho tápat. Nesplnění se samozřejmě eviduje a později je nutné vyhodnotit, proč respondent nevěděl, jak pokračovat. V průběhu testování jsou nejdůležitější tzv. „AHA momenty“, kdy respondent typicky řekne „AHA“. To jsou nejcennější informace, které si z testování můžeme odnést. Znamenají, že uživatel čekal něco jiného, než se stalo. Respondenta se doptáme, co očekával a následně zhodnotíme získané poznatky. Po splnění (nebo nesplnění) všech scénářů je ještě respondent dotázán na problémy, které měl, co očekával, že se stane, případně další věci. Po skončení samotného testování je dobré s respondentem dále chvíli diskutovat o průběhu a problémech a zeptat se na jeho názor. To mohou být další cenné informace získané testováním [50].

S použitím těchto pravidel jsem vytvořil následující scénář pro testování:

- O čem webová stránka je?
- Řekněme, že vás aplikace zaujala, začněte ji používat. Od teď prosím komentujte vše, co vás napadne. (registrace a přihlášení)
- Co se po vás na této stránce chce? Vytvořte novou rodinu. (Po registraci není uživatel součástí žádné rodiny a musí buď potvrdit dříve zaslanou žádost, nebo založit novou rodinu.)
- Kde se v aplikaci nacházíme? (dashboard)
- Popište jednotlivé části. (Kalendář, Messenger, Seznam úkolů, Nákupní seznamy, Finance, Lokace, Moje rodina, Nastavení)
- Pozvěte dalšího uživatele do vaší rodiny. (Moje rodina)
- Napište mu pozdrav. (Messenger)
- Co myslíte, že bude na stránce Lokace? Přejděme tam. Je zde to, co jste očekával?
- Co myslíte, že bude na stránce Finance? Přejděme tam. Je zde to, co jste očekával?
- Podívejte se na finance s nově pozvaným uživatelem. Co vidíte?

- Vytvořte nový záznam. Řekněme, že jste za něj včera platil večeři a on vám dluží 400 korun.
- Stalo se, co jste očekával? Jaké informace před sebou vidíte?
- Vraťte se zpět a vytvořte nákupní seznam s nákupem na víkend. Budeme kupovat maso, rýži a víno.
- Změňte vaše heslo na nové.
- Vytvořte událost v kalendáři o nedělním obědě. Vidí událost všichni členové rodiny?
- Co by se v aplikaci podle vás dalo ještě zlepšit?
- Naplnila aplikace vaše očekávání, která jste měl na začátku?

Nejtěžší věcí na uživatelském testovaní je nenapovídání respondentům. I nevinná otázka „Co získáte, když vyplníte formulář?“ je zavádějící. Předpokládá totiž, že si respondent všiml formuláře, že pochopil, že se tam něco vyplňuje a pochopil, proč ho vyplňuje. Ani jeden z těchto faktů respondent před položením otázky nemusí znát a otázkou mu napovíme. Negativně tím ovlivníme výsledek testování. Nejlepší je nechat respondenta samotného popsat, co vidí a k čemu to slouží. Pokud by se respondent sám zeptal na něco k aplikaci, odpovědí by mohlo být: „Já vám to nechci teď říkat, ale zajímá mě, co si myslíte vy.“ Je vhodné se během testování vyvarovat vysvětlování a snažíme se pokládat jen „bezpečné otázky“. Je velmi důležité, aby se respondent necítil sám testovaný, a proto mu připomínáme, že žádná odpověď není správná/špatná. Bude potom více otevřený a upřímný, testování bude tedy relevantnější. Na konci je dobré mu vysvětlit části scénáře, které se během testování ukázaly jako problematické. Také je dobré neformálně probrat, co se mu líbilo a co by se dalo zlepšit [50].

6.1.2 Průběh samotného testu

Testování v této práci proběhlo celkem ve třech fázích vývoje, pokaždé na jednom respondentovi. První probíhalo nad grafickým návrhem, který nebyl interaktivní, skládal se tedy jen z návrhů obrazovek, které jsem popsal v kapitole Návrh uživatelského rozhraní. Jelikož testování probíhalo na nefunkčním grafickém návrhu, byl pro respondenta složitější pohyb v samotném grafickém návrhu a testování bylo obtížnější. Připravil jsem si několik jednoduchých scénářů (nebyly tak podrobné, jako je napsaný výše) a v průběhu se ptal na doplňující otázky. Samotné testování zabralo přibližně 15 minut a nasledující diskuze asi 20 minut. Ve druhém testování jsem již měl hotový frontend a připravená základní testovací data. K testování jsem použil výše popsaný scénář, stejně jako pro poslední testování, které probíhalo na již hotové aplikaci.

Během testů byly objeveny následující problémy s uživatelským rozhraním:

- Respondent při prvním testování nevěděl, co si představit pod jednotlivými položkami na dashboard stránce.
- Příliš veliká šipka „zpět“. Během druhého testování jsem si všiml, že respondent velmi rychle směřoval myší k šipce „zpět“.
- Respondent při pokynu ke změně hesla očekával tuto funkcionalitu v editaci profilu namísto v nastavení aplikace.
- Zaškrtávací pole u seznamů nejsou celé „klikatelné“.
- Respondent očekával, že po vytvoření nového seznamu dojde k přesměrování na předchozí stránku.
- Během testování kalendáře respondent očekával, že vytvoření nové události provede i přes kliknutí na konkrétní den.

Dle provedeného testování a nalezených problémů byly provedeny úpravy v aplikaci tak, aby se tyto problémy již nevyskytovaly. Úpravy jsou následující:

- Po testování jsme diskutovali s respondentem a došli jsme k závěru, že by se v dashboardu mohly k jednotlivým odkazům na stránky přidat ikony pro snadnější orientaci a pochopení, co se pod odkazy nachází.
- Po skončení testovaní a rozhovorem s respondentem jsem usoudil, že šipka „zpět“ je zbytečně dominantní. Zmenšil jsem proto tuto šipku přibližně o čtvrtinu.
- Změnu hesla jsem přesunul do editace profilu a zároveň ji ponechal i v nastavení. Pokud by v budoucím testování vyšlo najevo, že je pro respondenty matoucí mít tu samou funkcionalitu na dvou místech, musel bych přijít s jiným řešením.
- Byla rozšířena oblast pro zaškrtnutí položky. Po diskuzi s respondentem jsme došli k závěru, že by bylo dobré mít možnost měnit pořadí položek. Toto rozšíření by mohlo být zpracováno v budoucím rozšíření aplikace.
- Bylo přidáno přesměrování po vytvoření nového seznamu.
- Z důvodu omezených možností použitého kalendáře třetí strany není možné doimplementovat přidání nové události po kliknutí na konkrétní den. Toto zjištění může být podnět pro budoucí zdokonalení aplikace.

6.2 Automatické testování

Řádné a metodické testování má smysl při větším počtu programátorů u větších projektů. Důkladné a řádné testování znamená testovat od nejmenších oddělených částí. Následně jdeme o stupeň výše až po testování aplikace jako celku [51].

„Automatickým testováním softwaru rozumíme kód (tedy další software), který nějakým způsobem sám otestuje, zda naše aplikace funguje.“ Největším rozdílem mezi manuálním a automatickým testováním je nutnost kontroly výsledků. Automatický test nám řekne „prošel“, nebo „neprošel“ [51].

Rozdělení automatických testů jde kategorizovat podle různých kritérií. Jak je výše zmíněno, testy lze rozdělit na manuální a automatické. Jde tedy o rozdělení dle způsobu vyhodnocování. Další dělení je podle toho, zda známe zdrojový kód či nikoliv. Pokud kód známe, jde o *white box* testování. Pokud kód neznáme, testování je označováno za *black box*. Automatické testy se však nejčastěji dělí dle rozsahu testovaného celku. Konkrétně jde o testy jednotkové, testy komponent, integrační testy a systémové testy. Jednotkovými testy nazýváme takové testy, které se zabývají správností funkcí metod a tříd. Většími celky, konkrétně komunikací mezi třídami, se zabývají testy komponent. Tyto dva typy jsou nejčastěji a také nejjednodušeji automatizovatelné. Následují integrační testy, které již testují spolupráci komponent a simulují reálné prostředí. Zabývají se například testem připojení a komunikace s databází. Poslední jsou systémové testy, které testují největší celek aplikace, kterým je celý systém. Systémové testy se typicky provádí manuálně. Postupuje se při nich dle zadaného scénáře. Většinou jde o scénáře z případů užití [51].

Pro architekturu aplikace a technologie, které jsou zvolené, byl na automatické testování vybrán framework JestJS Ten je zaměřený na jednotkové testy v programovacím jazyce JavaScript.

Jelikož největší bezpečnostní chyby mohou vznikat na nedostatečně zabezpečeném backendu, v tomto případě Firebase, jsou testována pravidla pro CRUD operace ve Firestore.

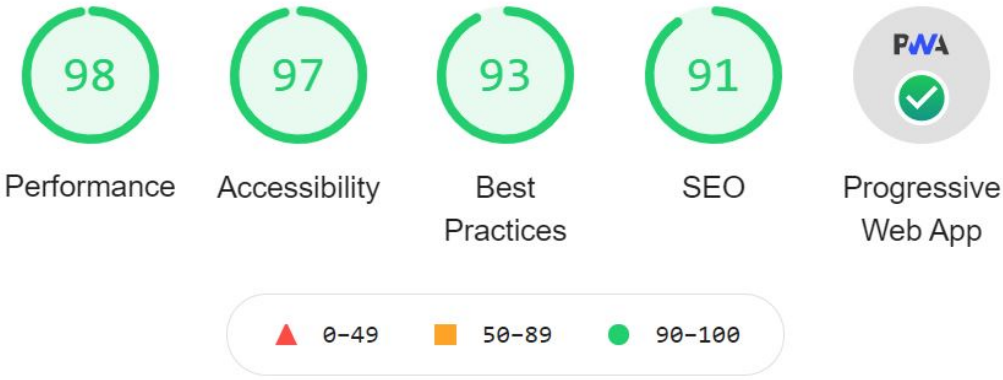
Pro testování pravidel Firestore je nutné mít lokálně spuštěný firebase emulátor. Tento emulátor lokálně simuluje Firebase služby, konkrétně je pro testování potřeba služba Firestore.

Mohlo by se zdát, že jde o integrační testování. Testují se ale jen pravidla pro dazabázi jako malé celky, jde tedy o jednotkové testování.

Emulátor vyžaduje globálně nainstalovaný Firebase CLI. Příkazem `firebase emulators:start` se spustí lokální server (pokud není nainstalovaný, provede se nejprve instalace) s databází. Pro spuštění testů pak stačí provést příkaz `npm run test`.

JavaScriptové soubory s testy jsou umístěny v adresáři `src/firestore-test`. V adresáři je soubor `helpers.js`, který obsahuje základní nstavení databáze pro testování. Zároveň je funkce `expect()` rozšířena o dvě další testovací funkce (`toAllow()` a `toDeny()`). Jde o fukce, které testují přijaté operace v atributu. Následně jsou využívány pro testování pravidel pro jednotlivé kolekce.

Výsledná implementace byla podrobena testovacímu nástroji Lighthouse. Na obrázku 6.1 je zobrazen výsledek testu. Dobrého skóre bylo dosaženo ve všech testovaných oblastech. Ve výsledku je rychlost aplikace ještě umocněna ukládáním assetů do Cache Storage a dat z databáze do IndexedDB.



Obrázek 6.1 Lighthouse benchmark report

Tato aplikace je spíše prototypem. Před produkčním nasazením by bylo nutné vytvořit mnohem komplexnější testy, které by pokryly funkcionality jednotlivých komponent.

Kapitola 7
Závěr

Cílem práce bylo navrhnout a implementovat rodinný organizér formou progresivní webové aplikace, která usnadní plánování a komunikaci v rodinném kruhu. Součástí této aplikace měl být sdílený kalendář mezi členy rodiny, společný chat, sdílené seznamy úkolů a nákupní seznamy, evidence neproplacených financí a sdílení polohy. Aplikace měla být spustitelná v jakémkoliv moderním webovém prohlížeči.

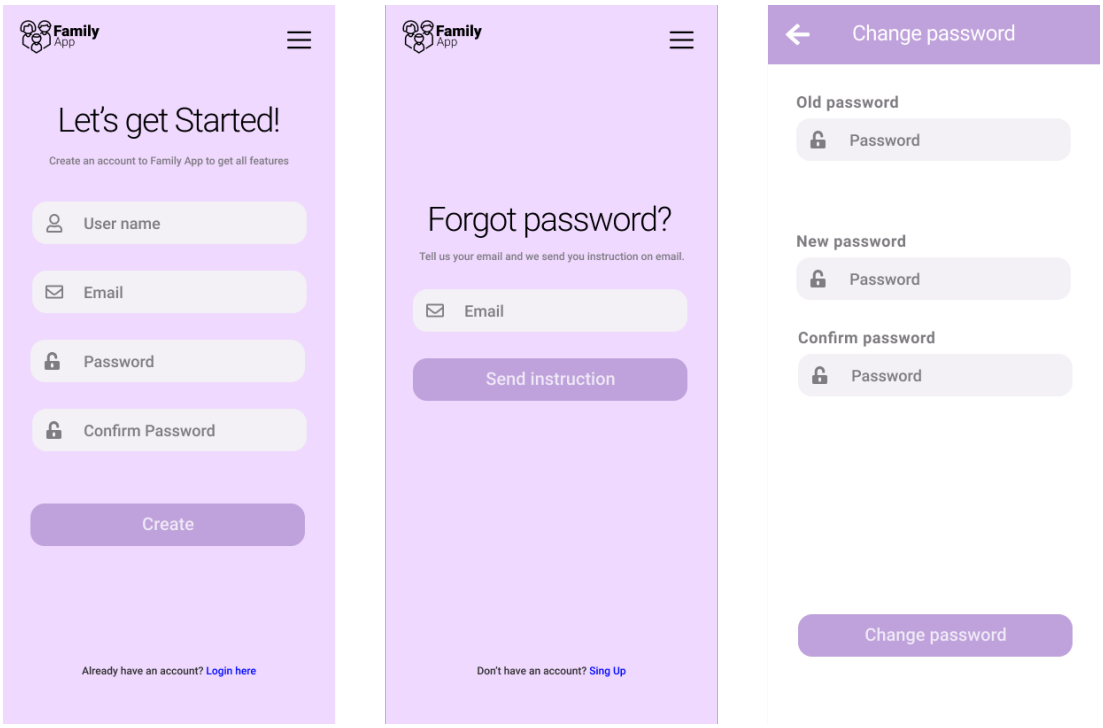
Z počátku bylo nutné provést analýzu konkurenčních řešení, která mají podobné případy užití. Na základě této analýzy byly sestaveny funkční a nefunkční požadavky aplikace. Následně byla provedena analýza relevantních technologií, kde došlo k výběru technologií, které byly následně využity při implementaci. V části návrhu byl na základě předchozích analýz a funkčních požadavků vytvořen grafický návrh aplikace, který obsahuje všechny důležité obrazovky aplikace. V implementační části je navržena struktura a architektura aplikace. Výsledkem této části je prototyp progresivní webové aplikace, kterou si uživatelé mohou nainstalovat na své zařízení a používat ji i bez připojení k internetu.

Během celého procesu vývoje byla provedena tři uživatelská testování. První z nich na grafickém návrhu aplikace, druhé na prototypu bez implementované logiky a poslední na již hotovém prototypu aplikace. Většina poznatků, které toto testování přineslo, byla zapracována během implementace aplikace.

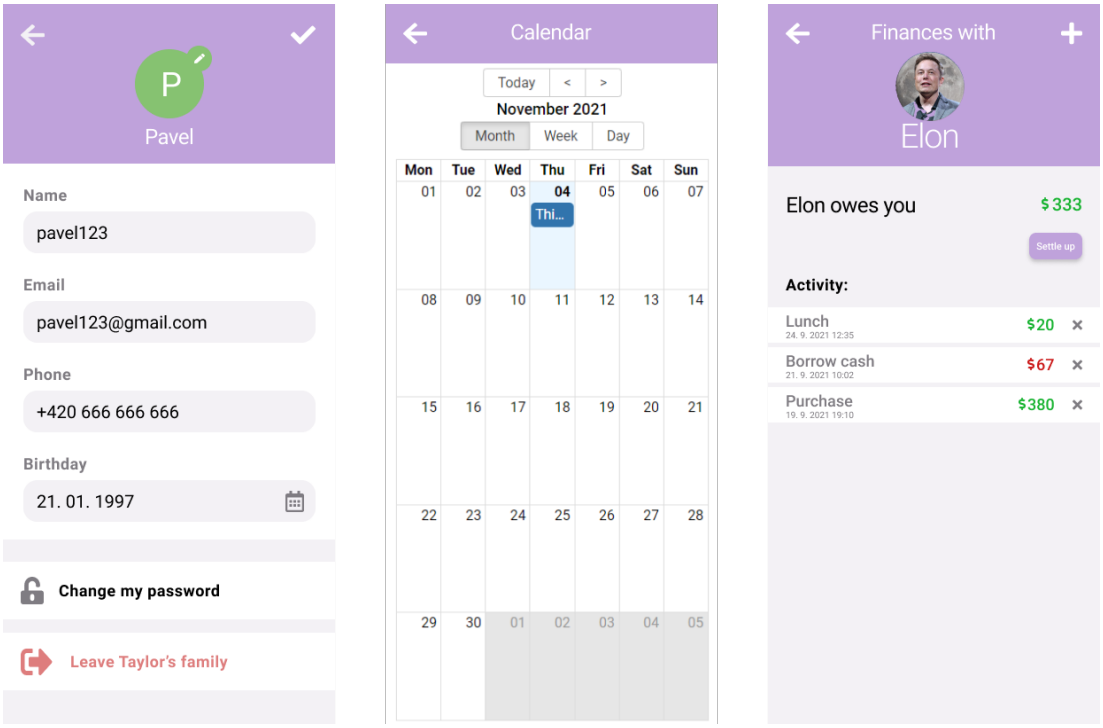
V budoucnosti by bylo možné aplikaci rozšířit o další funkcionality, jako například sdílení událostí do jiných kalendářů, databázi receptů nebo přidání dětských účtů a sledování polohy dětí. Aplikace má velký potenciál v možnostech rozšíření o další funkcionality. Případná rozšíření by měla usnadnit jednoduchá architektura aplikace, která dodržuje doporučení pro vytváření aplikací ve frameworku React.js.

Příloha A

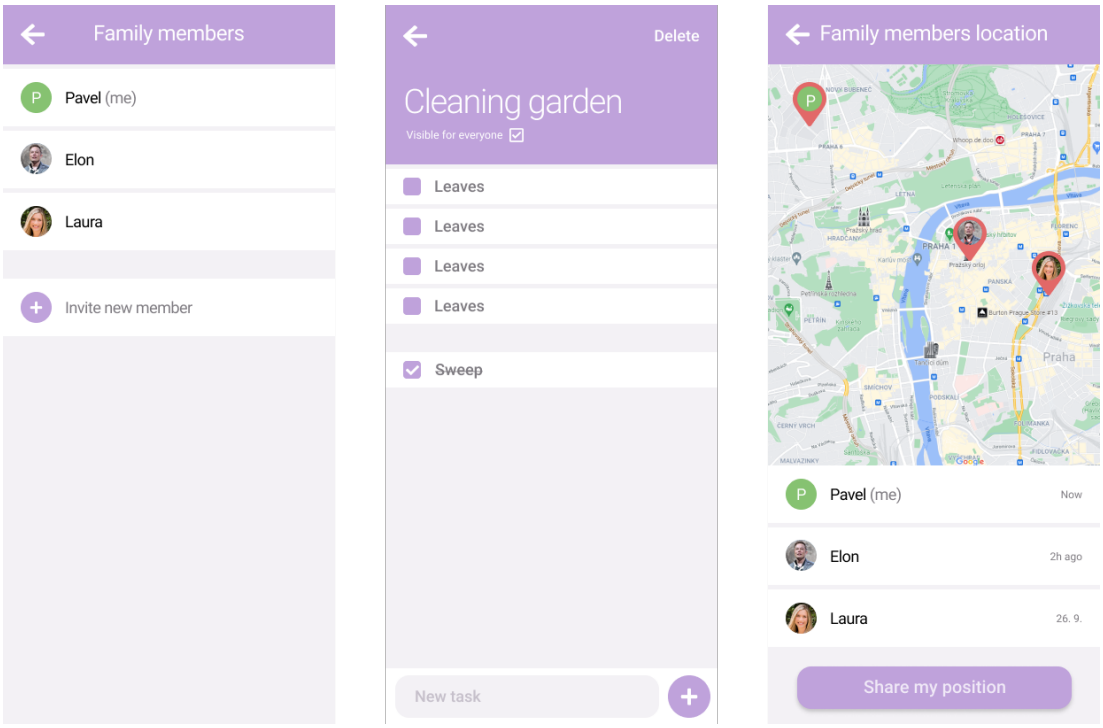
Návrhy obrazovek aplikace



Obrázek A.1 Grafický návrh obrazovek - registrace, zapomenuté heslo, změna hesla



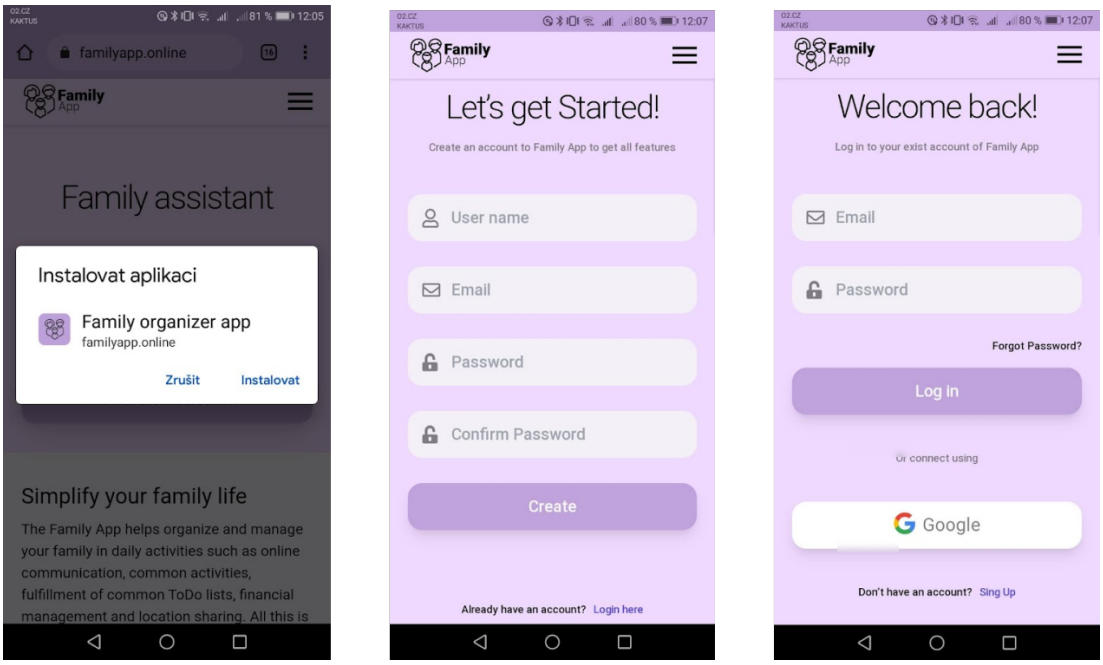
Obrázek A.2 Grafický návrh obrazovek - editace profilu, kalendář, finance



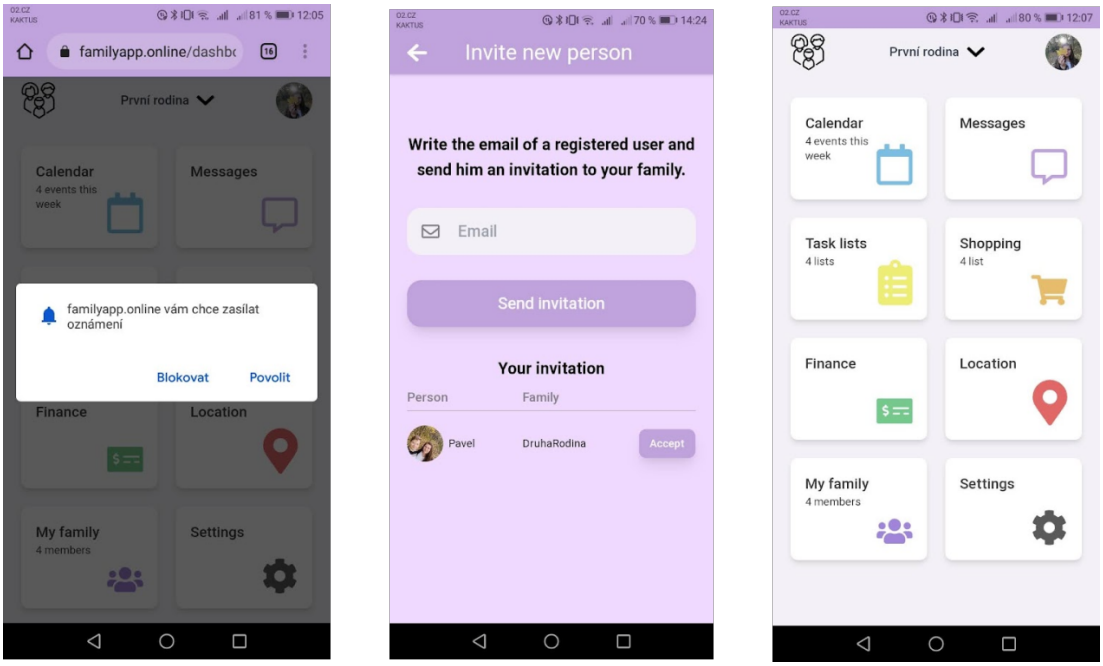
Obrázek A.3 Grafický návrh obrazovek - členové rodiny, seznam úkolů, sdílení polohy

Příloha B

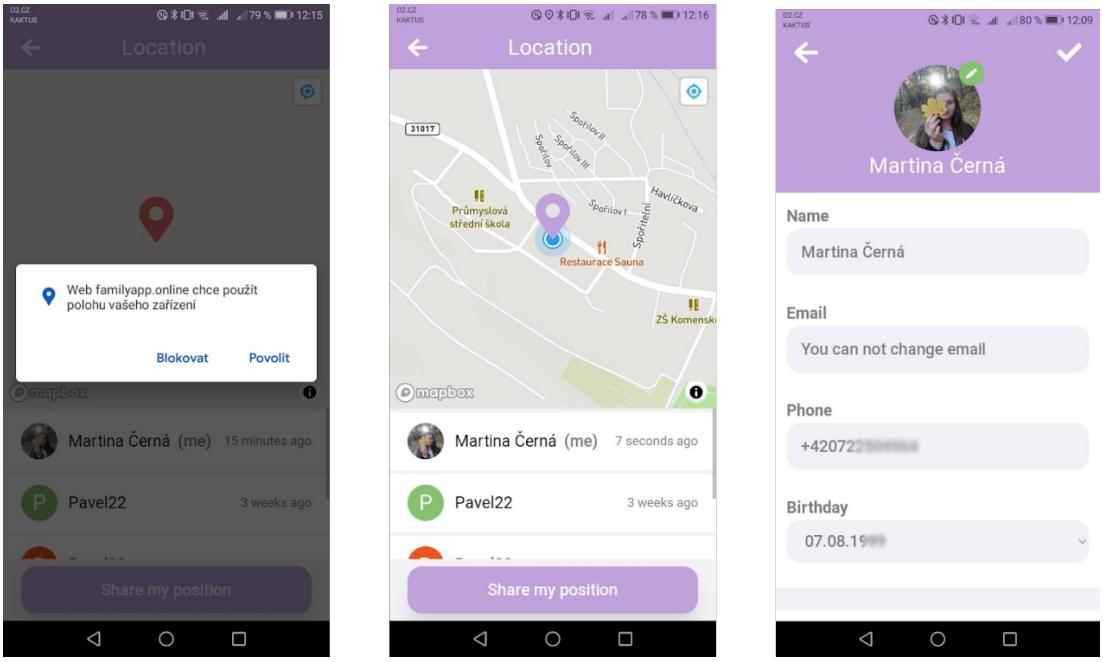
Obrazovky výsledné aplikace



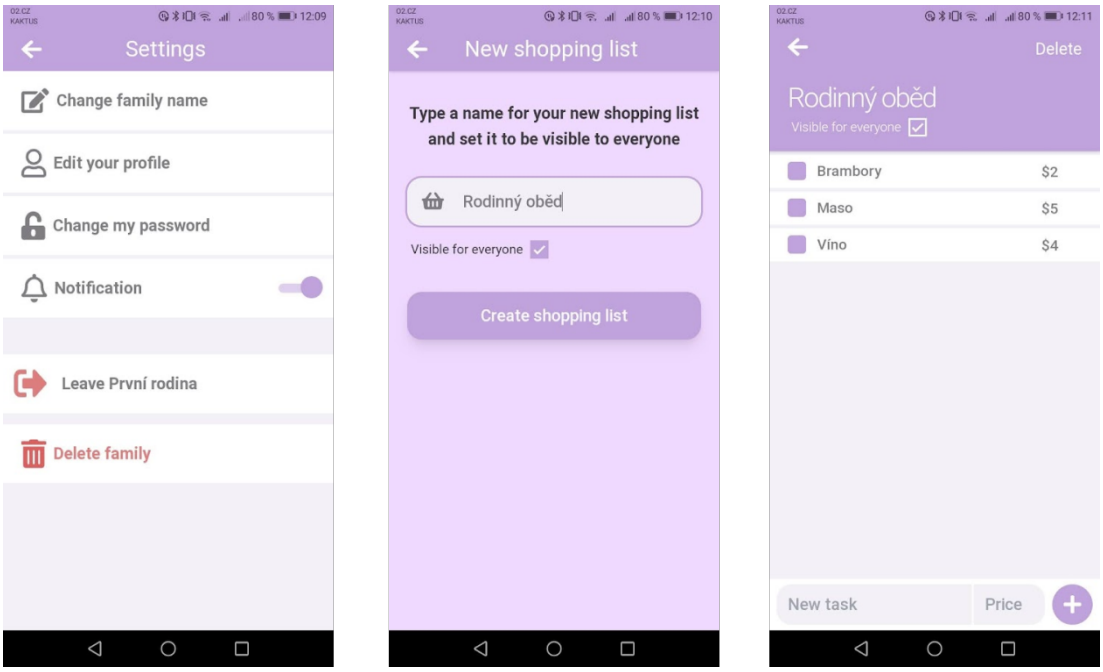
Obrázek B.1 Obrazovky aplikace - instalace aplikace, registrace, přihlášení



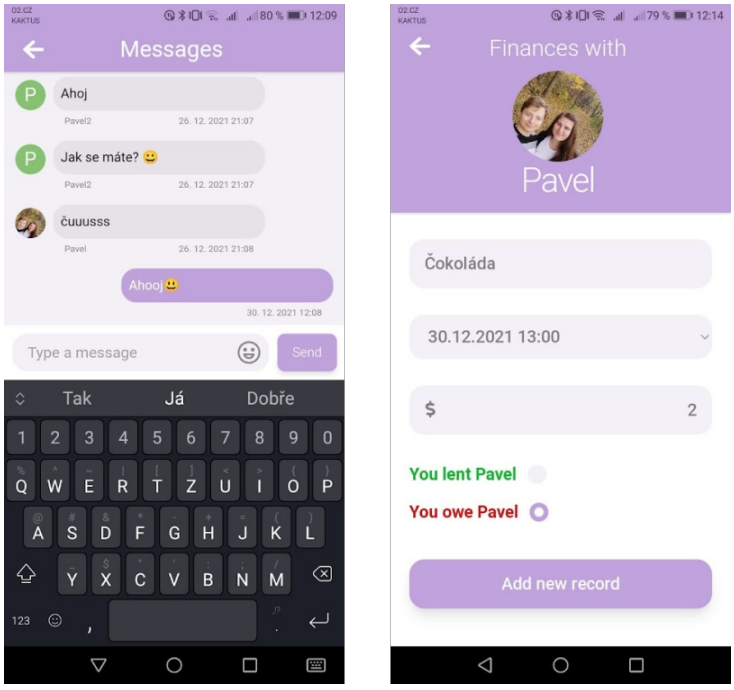
Obrázek B.2 Obrazovky aplikace - povolení oznámení, pozvání nového člena, hlavní rozcestník



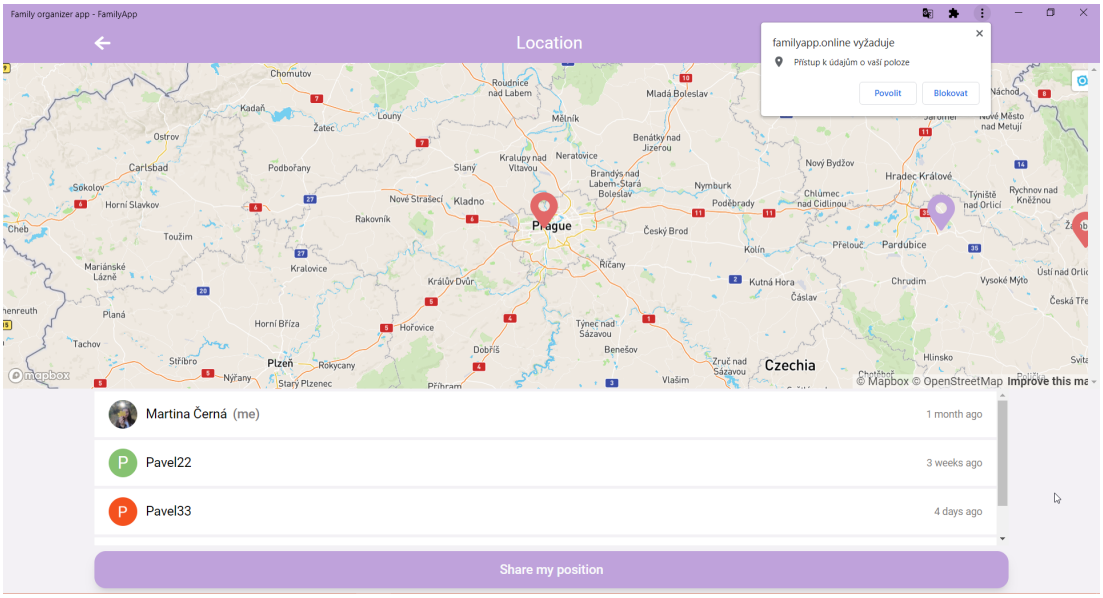
Obrázek B.3 Obrazovky aplikace - povolení sdílení polohy, sdílení polohy, editace profilu



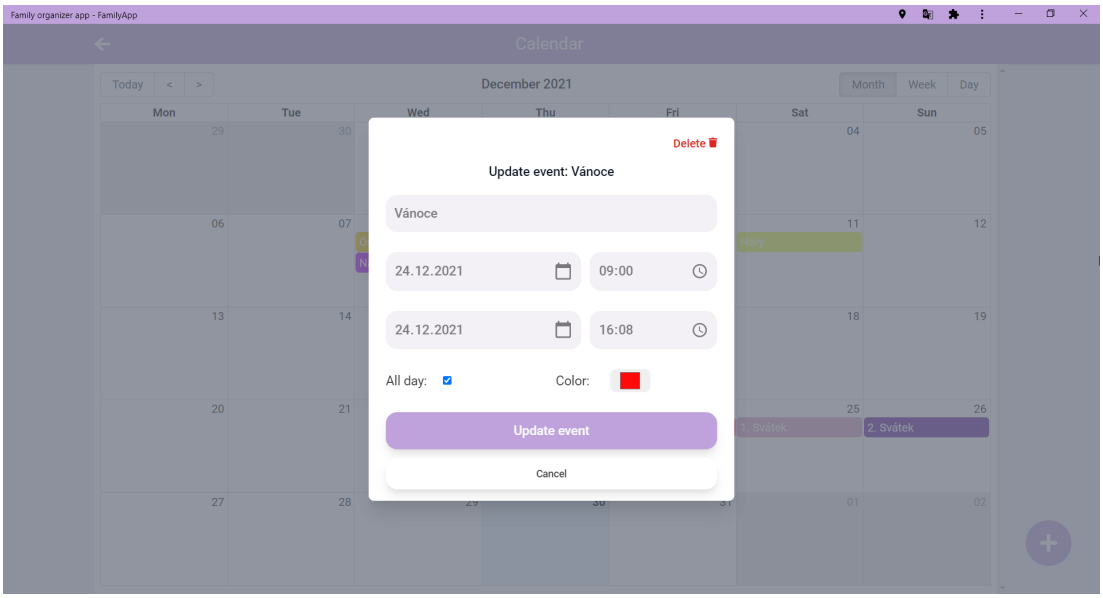
Obrázek B.4 Obrazovky aplikace - nastavení, vytvoření nákupního seznamu, položky seznamu



Obrázek B.5 Obrazovky aplikace - chat, vytvoření finančního záznamu



Obrázek B.6 Obrazovka aplikace - sdílení polohy s dotazem na povolení sdílení



Obrázek B.7 Obrazovka aplikace - vytvoření události v kalendáři

Bibliografie

1. *Daisy Family App* [online]. Daysi Aps [cit. 2021-12-04]. Dostupné z: https://daysiapp.com/.
2. *Google Play: Daysi Familie App (Daysi Family)* [online]. Daysi Aps - Google play [cit. 2021-12-05]. Dostupné z: https://play.google.com/store/apps/details?id=io.trigger.forgeb30bfde2e5d911e5b0e412313b0234c0.
3. *FamilyWall: Happy Organization* [online]. Family & Co [cit. 2021-12-04]. Dostupné z: https://www.familywall.com/index.html.
4. *Google Play: FamilyWall* [online] [cit. 2021-12-05]. Dostupné z: https://play.google.com/store/apps/details?id=com.familywall.
5. *Cozi Family Organizer* [online]. Cozi Inc. [cit. 2021-12-04]. Dostupné z: https://www.cozi.com/.
6. *Google Play: Cozi Family Organizer* [online]. Cozi Inc. [cit. 2021-12-05]. Dostupné z: https://play.google.com/store/apps/details?id=com.cozi.androidfree\&referrer=utm_source\%3DCoziHP\%26utm_medium\%3Ddesktop.
7. *Google Play: Family Shared Calendar* [online]. Beesoft Apps [cit. 2021-12-04]. Dostupné z: https://play.google.com/store/apps/details?id=com.appxy.famcal\&hl=cs\&gl=US.
8. *Our Home* [online]. OurHomeApp [cit. 2021-12-04]. Dostupné z: http://ourhomeapp.com/.
9. *Google Play: OurHome* [online] [cit. 2021-12-05]. Dostupné z: https://play.google.com/store/apps/details?id=com.getfairshare.ourhome.
10. *Trello* [online]. Trello, Inc. [cit. 2021-12-05]. Dostupné z: https://trello.com/.
11. *Gmail* [online]. Google [cit. 2021-12-05]. Dostupné z: https://mail.google.com/.
12. *Evernote* [online]. Evernote Corporation [cit. 2021-12-05]. Dostupné z: https://evernote.com/intl/cs.
13. *MS Teams* [online]. Microsoft [cit. 2021-12-05]. Dostupné z: https://www.microsoft.com/cs-cz/microsoft-teams.
14. *Slack* [online]. Slack Technologies [cit. 2021-12-05]. Dostupné z: https://slack.com/.
15. *Google Play: Microsoft Teams* [online]. Microsoft Corporation [cit. 2021-12-05]. Dostupné z: https://play.google.com/store/apps/details?id=com.microsoft.teams\&hl=cs\&gl=US.
16. *Google Play: Slack* [online]. Slack Technologies Inc. [cit. 2021-12-05]. Dostupné z: https://play.google.com/store/apps/details?id=com.Slack\&hl=cs\&gl=US.

17. LANDSBERGEROVÁ, Alžběta. *Kalendář, historie, proměny a současné trendy tvorby* [online]. Zlín, 2006 [cit. 2021-12-05]. Dostupné z: http://digilib.k.utb.cz/bitstream/handle/10563/1997/landsbergerov\%C3\%A1_2006_bp.pdf?sequence=1. Bakalářská práce. Univerzita Tomáše Bati.
18. HAZAËL-MASSIEUX, Dominique. JavaScript web APIs [online]. [B.r.] [cit. 2021-12-05]. Dostupné z: https://www.w3.org/standards/webdesign/script.
19. What Is a Framework? *Codecademy* [online]. 2021 [cit. 2021-12-05]. Dostupné z: https://www.codecademy.com/resources/blog/what-is-a-framework/.
20. *Stack Overflow Insights: Trends* [online] [cit. 2021-12-05]. Dostupné z: https://insights.stackoverflow.com/trends?tags=jquery\%2Cangular\%2Creactjs\%2Cjavascript\%2Cexpress\%2Cvue.js.
21. MÁCA, Jindřich. Úvod do Angular frameworku. *Itnetwork* [online]. 2020 [cit. 2021-12-05]. Dostupné z: https://www.itnetwork.cz/javascript/angular/zaklady/uvod-do-angular-frameworku.
22. *Introduction to Angular concepts* [online] [cit. 2021-12-05]. Dostupné z: https://angular.io/guide/architecture.
23. *Who Uses Angular in 2022?: 12 Global Websites Built With Angular* [online]. 2021 [cit. 2021-12-05]. Dostupné z: https://trio.dev/blog/companies-use-angular.
24. *Basic pieces in relation* [online] [cit. 2021-12-05]. Dostupné z: https://angular.io/generated/images/guide/architecture/overview2.png.
25. *React* [online] [cit. 2021-12-05]. Dostupné z: https://reactjs.org/.
26. KOSTRZEWSKI, Rafał; WARCHOLINSKI, Matt. 10 Famous Apps Using ReactJS Nowadays [online]. 2021 [cit. 2021-12-05]. Dostupné z: https://brainhub.eu/library/famous-apps-using-reactjs/.
27. YOU, Evan. *Vue.js* [online] [cit. 2021-12-05]. Dostupné z: https://vuejs.org/v2/guide/.
28. BREWSTER, Cordenne. 15 Examples of Global Websites Using Vue.js in 2022. *Trio* [online]. [B.r.] [cit. 2021-12-05]. Dostupné z: https://trio.dev/blog/websites-using-vue.
29. The Types of Databases. *Matillion* [online]. 2020 [cit. 2021-12-05]. Dostupné z: https://www.matillion.com/resources/blog/the-types-of-databases-with-examples.
30. *MySQL Documentation* [online]. Oracle Corporation [cit. 2021-12-05]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html.
31. *MongoDB Documentation* [online]. MongoDB, Inc. [cit. 2021-12-05]. Dostupné z: https://docs.mongodb.com/.
32. *Firebase Documentation* [online]. Google Developers [cit. 2021-12-05]. Dostupné z: https://firebase.google.com/docs.
33. *Firebase Authentication* [online]. Google Developers [cit. 2021-12-05]. Dostupné z: https://firebase.google.com/docs/auth.
34. Best CSS Frameworks in 2021. *Https://dev.to/* [online]. 2020 [cit. 2021-12-05]. Dostupné z: https://dev.to/theme_selection/best-css-frameworks-in-2020-1jjh.
35. MICHÁLEK, Martin. Průvodce CSS preprocesory. *Vzhůru Dolů* [online]. 2014 [cit. 2021-12-05]. Dostupné z: https://www.vzhurudolu.cz/blog/12-css-preprocesory-1.
36. *Bootstrap Documentation* [online] [cit. 2021-12-05]. Dostupné z: https://getbootstrap.com/docs/5.1/about/overview/.
37. *Materialize Documentation: Material Design* [online] [cit. 2021-12-05]. Dostupné z: https://materializecss.com/.

38. *Tailwind CSS Documentation* [online] [cit. 2021-12-05]. Dostupné z: https://tailwindcss.com/docs.

39. GATTERMAYER, Josef. Proč a jak psát progresivní webové aplikace. *Ackee: Blog* [online]. 2017 [cit. 2021-12-05]. Dostupné z: https://www.ackee.cz/blog/proc-a-jak-psat-progresivni-webove-aplikace/.

40. RUSSELL, Alex. Progressive Web Apps: Escaping Tabs Without Losing Our Soul. *Medium* [online]. 2015 [cit. 2021-12-05]. Dostupné z: https://medium.com/@slightlylate/progressive-apps-escaping-tabs-without-losing-our-soul-3b93a8561955.

41. *Using the Geolocation API* [online]. Mozilla MDN [cit. 2021-12-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API/Using_the_Geolocation_API.

42. OBSTOVA, Barbora. *Uživatelské rozhraní* [online] [cit. 2021-12-05]. Dostupné z: https://wikisofia.cz/wiki/U%C5%BEivatelsk%C3%A9_rozhran%C3%AD.

43. FADEYEV, Dmitry. 8 Characteristics Of Successful User Interfaces. *Usabilitypost* [online]. 2009 [cit. 2021-12-05]. Dostupné z: https://usabilitypost.com/2009/04/15/8-characteristics-of-successful-user-interfaces/.

44. ŘEZÁČ, Jan. *Web ostrý jako břitva: návrh fungujícího webu pro webdesignery a zadavatele projektů*. Jihlava: Baroque Partners, 2014. ISBN 978-808-7923-016.

45. RAE, Matt. 5 Examples of Effective Information Architecture. *Xd Ideas* [online]. 2020 [cit. 2021-12-05]. Dostupné z: https://xd.adobe.com/ideas/process/information-architecture/information-architecture-examples/.

46. MKRTCHYAN, Rafayel. Wireframe, Mockup, Prototype: What is What? *UX Planet* [online]. 2018 [cit. 2021-12-05]. Dostupné z: https://uxplanet.org/wireframe-mockup-prototype-what-is-what-8cf2966e5a8b.

47. MICHÁLEK, Martin. Co je to „Mobile First“? Ale doopravdy. *Zdrojak* [online]. 2015 [cit. 2021-12-05]. ISSN 1803-5620. Dostupné z: https://zdrojak.cz/clanky/co-je-to-mobile-first-ale-doopravdy/.

48. TOPOLSKY, Joshua. Exclusive: Matias Duarte on the philosophy of Android, and an in-depth look at Ice Cream Sandwich. *The Verge* [online]. 2011 [cit. 2021-12-05]. Dostupné z: https://www.theverge.com/2011/10/18/exclusive-matias-duarte-ice-cream-sandwich-galaxy-nexus.

49. KOŠUT, Ondřej. *MOBILNÍ LEXIKON ZVÍŘAT ZOO PRAHA* [online]. Praha, 2020 [cit. 2021-12-05]. Dostupné z: https://dspace.cvut.cz/handle/10467/86573. Diplomová práce. České vysoké učení technické v Praze.

50. SOUKUP, Jan. Jak udělat uživatelské testování svépomocí [online]. 2020 [cit. 2021-12-05]. Dostupné z: https://www.honzasoukup.cz/blog/jak-udelat-uzivatelske-testovani-svepomoci.

51. PEKAŘ, Lukáš. Automatické testování softwaru: neboli kód, co kontroluje kód. *Bonsai development* [online]. 2017 [cit. 2022-01-02]. Dostupné z: https://bonsai-development.cz/clanek/automaticke-testovani-softwaru.

Obsah přiloženého média

- `/`
  - `readme.txt` ..... stručný popis obsahu média
  - `screens` ..... adresář s grafickým návrhem
    - `items` ..... adresář s jednotlivými obrazovkami návrhu
    - `graphic_design.pdf` ..... PDF soubor s celým grafickým návrhem
  - `src`
    - `impl` ..... zdrojové kódy implementace
    - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `text` ..... text práce
    - `thesis.pdf` ..... text práce ve formátu PDF
    - `user_documentation.pdf` ..... uživatelská dokementace ve formátu PDF