Mobilní aplikace pro podporu prodeje

Bc. Patrik Valuch

Diplomová práce
2021

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

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

Akademický rok: 2020/2021

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Patrik Valuch**
Osobní číslo:	**A19370**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**Kombinovaná**
Téma práce:	**Mobilní aplikace pro podporu prodeje**
Téma práce anglicky:	**A Sales Support Mobile Application**

Zásady pro vypracování

1. Definujte základní metody vývoje mobilních aplikací z oblasti nativní i hybridní.
2. Popište vybrané aktuálně dostupné frameworky pro vývoj mobilních aplikací.
3. Definujte funkční a nefunkční požadavky mobilní aplikace pro podporu prodeje implementované v rámci praktické části.
4. Navrhněte jednotlivé obrazovky aplikace pomocí programu Adobe XD.
5. V textu praktické části shrňte nejdůležitější části implementace.
6. Za použití Ionic frameworku implementujte prototyp mobilní aplikaci dle požadavků.

Forma zpracování diplomové práce: **Tištěná/elektronická**

Seznam doporučené literatury:

1. Ionic Framework. [online]. [2020].Dostupné na internetu: https://ionicframework.com/docs.
2. SINGH, Indermohan. Ionic Cookbook: Recipes to create cutting-edge, real-time hybrid mobile apps with Ionic. 3. Birmingham, Spojené království: Packt Publishing, [2018]. ISBN 978-1788623230.
3. YUSUF, Sani. Ionic Framework By Example. Birmingham, Spojené království: Packt Publishing, [2016]. ISBN 1782175415.
4. PHAN, Hoc. Ionic Framework Cookbook. Birmingham, Spojené království: Packt Publishing, [2015]. ISBN 1785287974.
5. KHANNA, Rahat. Ionic: Hybrid Mobile App Development. Birmingham, Spojené království: Packt Publishing, [2017]. ISBN 1788297814.
6. CASTRO, Elizabeth a Bruce HYSLOP. HTML5 a CSS3. Brno: Computer Press, Albatros Media, [2017]. ISBN 8025144658.
7. GAURAV, Saini. Hybrid Mobile Development with Ionic. Birmingham, Spojené království: Packt Publishing, [2017]. ISBN 1785280287.
8. GRIFFITH, Chris W. Mobile app development with Ionic: cross-platform apps with Ionic, Angular, and Cordova. Revised edition. Sebastopol, CA: O'Reilly Media, [2017]. ISBN 9781491998120.

Vedoucí diplomové práce: **Ing. Radek Vala, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **15. ledna 2021**
Termín odevzdání diplomové práce: **17. května 2021**

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

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
E.S.

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

Ve Zlíně dne 15. ledna 2021

Prohlašuji, že

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

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že elektronická verze práce, nahraná to IS/STAG, je shodná s verzí tištěnou.

Ve Zlíně, dne 25.4.2021 Patrik Valuch, v.r.

ABSTRAKT

Cílem této práce je návrh a následná realizace multiplatformní mobilní aplikace pro podporu prodeje. V rámci praktické části bude proveden kompletní návrh dané aplikace a závěr této části bude věnován implementačně zajímavým pasážím. Včetně realizované aplikace je výstupem i teoretická část, která obsahuje výčet možných technik při vývoji mobilních aplikací, možnosti v rámci vývojových nástrojů a obsáhlý popis použitého frameworku.

Klíčová slova: Mobilní aplikace, multiplatformní aplikace, Ionic framework, Angular, podpora prodeje

ABSTRACT

The aim of this thesis is design and implement the multiplatform mobile application for sales promotion. In the practical part will be performed a complete design of the application and the conclusion of this part will be devoted to interesting passages of the implementation. In addition to the implemented application, output is also theoretical part, which contains a list of possible techniques in the development of mobile applications, options within the development tools and a comprehensive description of the used framework.

Keywords: Mobile application, multiplatform application, Ionic framework, Angular, sales promotion

Chtěl bych poděkovat Ing. Radku Valovi, Ph.D. za odborné vedení mé diplomové práce, odborný dohled a cenné rady, které mi pomohly tuto práci dokončit. Dále bych chtěl poděkovat doc. Ing. Radku Šilhavému, Ph.D. za drahocenná doporučení v oblasti analýzy požadavků mobilní aplikace.

Poděkování také patří společnosti Czech Premium Product s.r.o. a vedoucímu vývojového týmu Ing. Martinu Vláčilovi za možnost pracovat na tomto zadání, svěřenou důvěru a odborné konzultace při zpracování tohoto zadání v rámci mé diplomové práce.

Velké díky patří mým rodičům za trpělivost a podporu po celou dobu mého studia.

Prohlašuji, že elektronická verze práce, nahraná to IS/STAG, je shodná s verzí tištěnou.

OBSAH

ÚVOD

Hlavním cílem této práce je vytvořit multiplatformní aplikaci pro podporu prodeje jednotlivých firem. Aplikace není vyvíjena pro jednu konkrétní firmu, ale jedná se o univerzální řešení za účelem jejího pronájmu firmám ve finančním sektoru. Výstupem je jedna aplikace, kde je přístup do jednotlivých firem řešen při autentizaci do této aplikace.

Téma mobilní aplikace bylo zvoleno z důvodu aktuálního vývoje webové aplikace, která by měla být dokončena v prvním kvartálu příštího roku. Mobilní aplikace je v tomto ohledu skvělé doplňující rozšíření, jelikož zákazníci ve finančním sektoru vyžadují rychlý přístup k aktuálním datům své firmy, čehož lze jednoduše docílit skrze mobilní aplikaci. Samotný Ionic framework byl zvolen z důvodu, že v první řadě splňoval podmínku multiplatformosti. Dalším z důvodů bylo, že je tento framework založen na webových technologiích, které jsou standardně používány při vývoji v rámci firmy, pro kterou je tato aplikace vyvíjená, tudíž její udržování a případné budoucí rozšiřovaní bude možné ze strany jakéhokoliv vývojáře z týmu.

Autentizace je jedním z důležitých bodů této aplikace. Aplikace umožňuje kromě klasických metod autentizaci pomocí otisku prstu, QR kódu a v případě iOS pomocí face ID. Další hlavní funkcionalitou je správné zobrazení a případná úprava dat zákazníka. Aplikace dále obsahuje systém interních zpráv pro přehledné informování o dění v aplikaci. Celý vývoj byl směřován tak, aby se v co největší míře jednalo o API based RESTful aplikaci.

Teoretická část této práce se zpočátku zabývá rešerší jednotlivých možností při vývoji mobilních aplikací. Jsou zde popsány metodiky vývoje nativních, hybridních a webových mobilních aplikací, jejich výhody a nevýhody a jejich vzájemné srovnání. Další kapitola je věnována popisu čtyř zvolených nástrojů pro vývoj mobilních aplikací. Tato kapitola obsahuje jejich stručný popis společně s jejich zhodnocením. Zbytek teoretické části je věnován podrobnému seznámení s frameworkem Ionic.

Praktická část začíná analýzou požadavků mobilní aplikace. Jsou zde vydefinovány funkční a nefunkční požadavky, use case model s jednotlivými scénáři. Další kapitola se věnuje návrhu uživatelského rozhraní pomocí aplikace Adobe XD. Závěr praktické části je věnován samotnému vývoji mobilní aplikace, kde jsou popsány zvolené metodiky a implementačně zajímavé části aplikace.

I. TEORETICKÁ ČÁST

1 MOŽNOSTI VÝVOJE MOBILNÍCH APLIKACÍ

Tato kapitola je věnována možným metodikám při vývoji mobilních aplikací. Jednotlivé podkapitoly popisují tři základní způsoby vývoje mobilních aplikací. Jedná se o aplikace hybridní, nativní a webové.

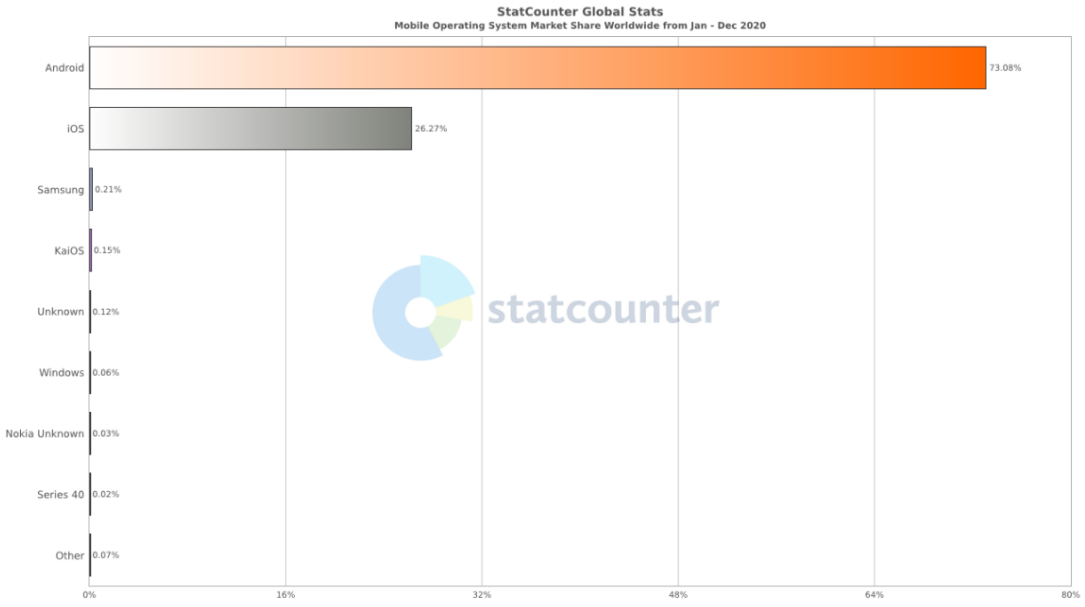
Obrázek 1: Možnosti při vývoji mobilních aplikací [1]

Při rozhodování, kterou z výše zmíněných metod vývoje zvolit pro vývoj své aplikace, musíme najít kompromis mezi možnostmi daného přístupu a našimi požadavky na aplikaci. Může se jednat o multiplatformnost aplikace, metodu distribuce nebo například prostředky pro samotný vývoj, ať už časové či finanční. Následující body obsahují parametry, které je nutné brát v potaz při tomto výběru [2].

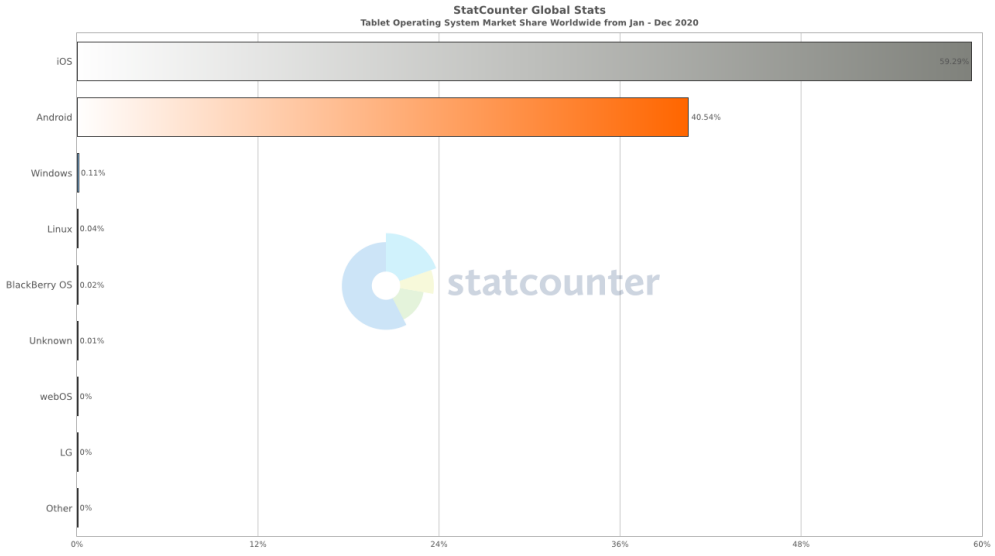
- Čas na vývoj
- Rozpočet pro vývoj
- Výkon
- Náročnost implementace
- Dostupnost pro různé platformy
- Přístup k *Application Programming Interface* (API)
- Metoda distribuce

Už zde byl několikrát použit pojem multiplatformnost. V současné době jsou mezi mobilními zařízeními, popřípadě tablety nejvíce používané operační systémy Android a iOS. Systém Android dle dostupných dat používá přibližně 73 % uživatelů mobilních zařízení, u iOS zdroje uvádí přibližně 26 %. U tabletů se pak rozdělení pohybuje okolo 59 % pro systém iOS a 40 % pro Android. Je nutné dodat, že se jedná o celosvětové statistiky, protože

v případě potencionálního vývoje aplikace pro severoamerický trh by situace byla úplně jiná, jelikož konkrétně v Severní Americe je podíl systému iOS daleko větší než například v Evropě, kde je poměr spíše opačný.



Obrázek 2: Graf podílu operačních systémů mobilních zařízení pro rok 2020 [3]



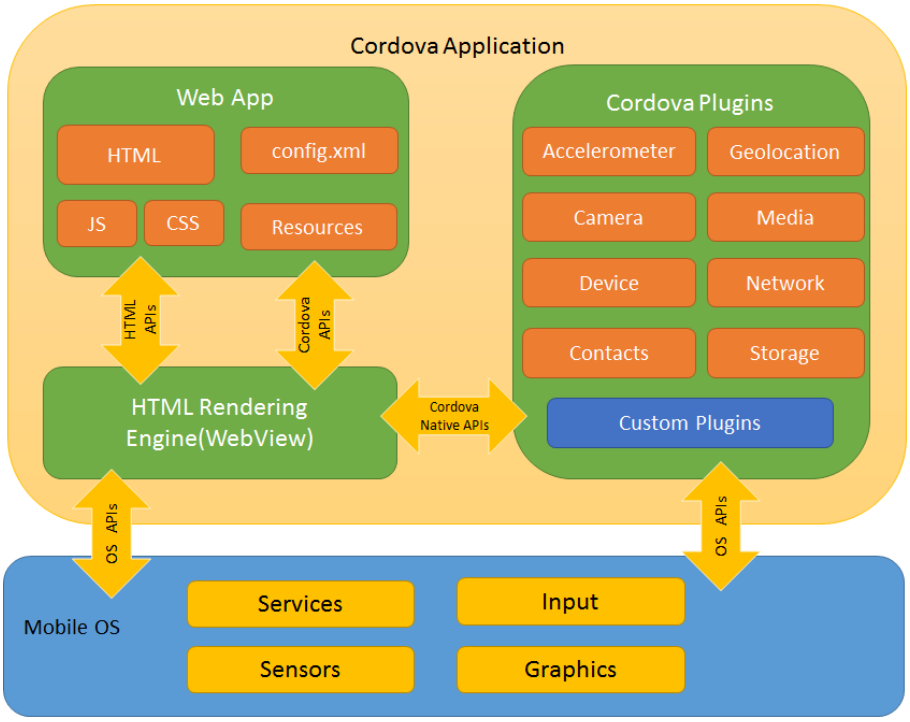
Obrázek 3: Graf podílu operačních systémů tabletů pro rok 2020 [4]

1.1 Hybridní mobilní aplikace

Webové hybridní mobilní aplikace lze definovat jako kombinaci nativní a webové aplikace. Webová hybridní aplikace je vyvíjena pomocí technologií *Hypertext Markup Language* (HTML), *Cascading Style Sheets* (CSS) a *JavaScript* (JS), stejně je tomu u aplikací

webových. Samotná hybridní aplikace poté běží v takzvaném WebView režimu, kde připojené pluginy umožňují využití nativních funkcí zařízení. Nativní pluginy aplikacím umožňují využívat funkce původních mobilních aplikací a jejich chování se značně podobá nativním aplikacím. Nesporným benefitem je vývoj v jednom programovacím jazyku, což znamená, že samotný vývoj probíhá pouze jednou s následným konvertováním pro všechny cílené platformy [5].

K vývoji hybridních aplikací se používají technologie jako je například Ionic nebo React Native. U frameworku Ionic dochází pomocí webové platformy, která je založená na standardech napříč platformami, k implementaci nativních vzorů uživatelského rozhraní iOS a Android. Naproti tomu u frameworku React Native dochází k přímému řízení ovládacích prvků uživatelského rozhraní platformy pomocí Reactu, který poskytuje potřebnou abstrakci [6]. Výše zmíněné frameworky jsou podrobněji popsány v nadcházejících kapitolách.



Obrázek 4: Struktura hybridní aplikace frameworku Ionic – Apache Cordova [7]

1.1.1 Výhody

Vzrůstající popularita hybridních aplikací jde souběžně s jejich technologickou vyspělostí. Popularita může vycházet z daleko levnějšího vývoje, ať už z důvodu kratšího času vývoje, či potřeby nižších znalostní samotných tvůrců, jelikož odpadá nutnost učení se nativních jazyků pro každou platformu zvlášť. Oproti webovým aplikacím, hybridní aplikace

umožnují interakci s nativními API daného zařízení a díky distribuci skrze obchody AppStore a Google Play jsou schopny oslovit větší množství potencionálních uživatelů. Následující body obsahují stručný seznam výhod hybridních aplikací [5].

- Multiplatformnost
- Globální úpravy, rozsáhlé možnosti úpravy vzhledu
- Jeden vývojový tým
- Jednodušší testování a následná údržba
- Stejná funkcionalita pro různé platformy
- Nižší náklady na vývoj
- Snadno škálovatelné pro jinou platformu
- Mohou pracovat bez připojení k internetu

1.1.2 Nevýhody

Největší nevýhodou, která může mít nesporně největší vliv na vývoj, je absence přímého přístupu k hardwaru zařízení. Pod tímto hardwarem si lze představit běžné věci, jako je fotoaparát nebo například čtečka otisku prstů. V případě chybějícího pluginu nebude možné s tímto hardwarem navázat spojení, pokud nedojde k naprogramování tohoto pluginu včetně jeho nativní části [5].

- Nižší výkon z důvodu provádění aplikačního kódu v běhovém prostředí JS
- Absence přímého přístupu k hardwaru – závislost na pluginech
- Nutnost kompatibility frameworků a knihoven s aktuální verzí operačního systému
- Zážitek z *User Experience* (UX) není plnohodnotný jako u nativních aplikací

1.2 Nativní mobilní aplikace

Nativní mobilní aplikace jsou aplikace vytvořené v konkrétním programovacím jazyce pro konkrétní platformu [2]. I v případě, že je funkcionalita aplikace pro všechny platformy stejná, vždy se jedná o jinou aplikaci [8]. V současné době se jedná především o operační systémy iOS a Android. Oproti například webovým aplikacím, které bývají převážně psány v jazyce Javascript, nativní aplikace jsou psány v jazycích pro danou platformu, v jazyce, který je na dané platformě využíván. U iOS se jedná například o programovací jazyk Swift nebo Objective-C, u systému Android je to pak programovací jazyk Java nebo Kotlin. Jak

společnost Apple, tak společnost Google nabízí vývojářům mobilních aplikací vlastní vývojové nástroje, prvky rozhraní nebo standardizovanou sadu *Software development kit* (SDK) [2].

Obrázek 5: Nativní mobilní aplikace [9]

1.2.1 Výhody

Výhodou výběru nativní aplikace je, že je nejrychlejší a nejspolehlivější, pokud jde o uživatelskou zkušenost. Nativní aplikace mohou také interagovat se všemi funkcemi operačního systému zařízení, jako je mikrofon, kamera, *Global Positioning System* (GPS) nebo seznam kontaktů [10].

Nativní aplikace je také pomocí vykreslovacího jádra zařízení spuštěna a vykreslena bez jiných mezivrstev kódu, které by mohly mít negativní vliv na výkon aplikace [12]. To lze považovat za pozitivum v případě, že aplikace potřebuje náročné zpracování pomocí *Central processing unit* (CPU) a *Graphics processing unit* (GPU), které vyžaduje spoustu paměti [13].

- Nejlepší výkon ze všech tří možností vývoje mobilních aplikací
- Plný přístup k celé sadě funkcí vybraného operačního systému
- Plnohodnotný zážitek z UX
- Distribuce skrze obchody AppStore a Google Play, lepší objevitelnost
- Vyšší míra zabezpečení
- Mohou pracovat bez připojení k internetu

1.2.2 Nevýhody

Při vývoji nativních mobilních aplikací je zapotřebí daleko větší rozpočet v případě vývoje pro více platforem. Pokud dojde například k rozhodnutí vyvíjet jak pro iOS, tak Android, bude zapotřebí mít k dispozici dva vývojářské týmy a je nutné počítat i s vyššími náklady na testování a na následný provoz a vývoj aktualizací pro danou aplikaci [10].

- Daleko vyšší náklady na vývoje ve srovnání s aplikacemi hybridními a webovými
- Složitější programovací jazyky, nutnost zkušeného vývojářského týmu
- Vyšší náklady na údržbu a aktualizace
- Nepříliš vhodné pro jednoduché aplikace
- Osloví pouze „publikum“ na jedné platformě a vylučují druhou
- Je obtížné docílit stejné uživatelské zkušenosti pomocí dvou různých aplikací na dvou různých platformách

1.3 Webové mobilní aplikace (PWA)

Webové mobilních aplikace nelze považovat za plnohodnotné mobilní aplikace, ale spíše webové stránky, které se snaží vypadat jako nativní aplikace. Tyto aplikace jsou ve velké míře naprogramovány pomocí technologie HTML5 a jsou spouštěny pomocí webového prohlížeče. Uživatel nejprve přistupuje na aplikaci skrze speciální *Uniform Resource Locator* (URL) adresu, která jej poté vyzve k nainstalovaní dané aplikace. Po odsouhlasení dochází k vytvoření záložky na tuto aplikaci na domovské obrazovce mobilního zařízení [14].

Největší popularita webových aplikací přišla právě s příchodem HTML5, kdy se ukázalo, že v prohlížeči lze získat podobné funkce jako u aplikací nativních. Dnes, kdy je technologie HTML5 široce rozšířená mezi klasickými webovými stránkami, jsou rozdíly mezi webovými aplikace a těmito weby minimální [14].

Nejzajímavější technologií z oblasti webových aplikací jsou aktuálně určitě *Progressive Web Apps* (PWA). PWA jsou nový typ webových aplikací a lze je označit za hybrid aplikací webových a nativních, kdy došlo ke spojení benefitů jednotlivých typů. PWA jsou stále spouštěny jako klasické webové stránky s rozdílem, že jejich vhled a uživatelský zážitek je srovnatelný s aplikacemi nativními. Díky absenci potřeby měnit způsob nasazení a balíčkování aplikace podle typu daného zařízení se přenositelnost aplikace stává hlavní

výhodou [15]. PWA rovněž webovým aplikacím zpřístupnili funkce jako odesílaní push notifikací, dotyková gesta nebo přístup k hardwaru zařízení, jako jsou například vibrace [16]. I přes tyto vylepšení je vcelku problematické vyvíjet PWA pro obě mobilní platformy. U systému iOS bohužel tyto aplikace nemají kompletní přístup ke všem funkcionalitám zařízení, což může být u některých typů aplikací problematické [16].

Obrázek 6: Přístup k funkcionalitám zařízení pro aplikace PWA [17]

1.3.1 Výhody

- Nízké náklady na vývoj
- Snadná údržba aplikace díky společné kódové základně
- Webové aplikace nemusí být schvalovány vlastníky obchodů
- Webové aplikace nevyžadují aktualizace po uživateli
- Ve srovnání s nativní nebo hybridní aplikací není nutná žádná instalace

1.3.2 Nevýhody

- Webové aplikace mají omezený přístup k funkcím daného zařízení
- Webové aplikace nejsou distribuovány skrze obchody, snížení objevitelnosti
- Nižší míra zabezpečení
- Webové aplikace vyžadují připojení k internetu
- Méně intuitivní a interaktivní, nižší provozní rychlost

2 NÁSTROJE PRO VÝVOJ MOBILNÍCH APLIKACÍ

Z předchozí kapitoly je zřejmé, že při vývoji mobilních aplikací mohou být zvoleny různé způsoby vývoje podle mnoha kritérií. Stejně tomu je v případě nástrojů pro vývoj mobilních aplikací. Tato kapitola se věnuje čtveřici vybraných nástrojů: React Native, Xamarin, Flutter a NativeScript. Následující podkapitoly obsahují obecný popis, popis architektury a specifické vlastnosti výše zmíněných nástrojů. Zvolený framework Ionic bude podrobněji popsán v samostatné kapitole.

Obrázek 7: Porovnání nástrojů React Native, Xamarin, NativeScript a Flutter [18]

2.1 React Native

React Native je open source, multiplatformní framework pro vývoj mobilních aplikací od společnosti Facebook, která tento framework vyvíjí již od roku 2015 [19]. Jeho vývoj byl zpočátku směřován především pro systém iOS, nyní je dostupný i pro systém Android [20]. Tento framework při vývoji umožňuje použití JavaScriptu s knihovnami React společně s nativními funkcemi dané platformy pomocí nativních API [19].

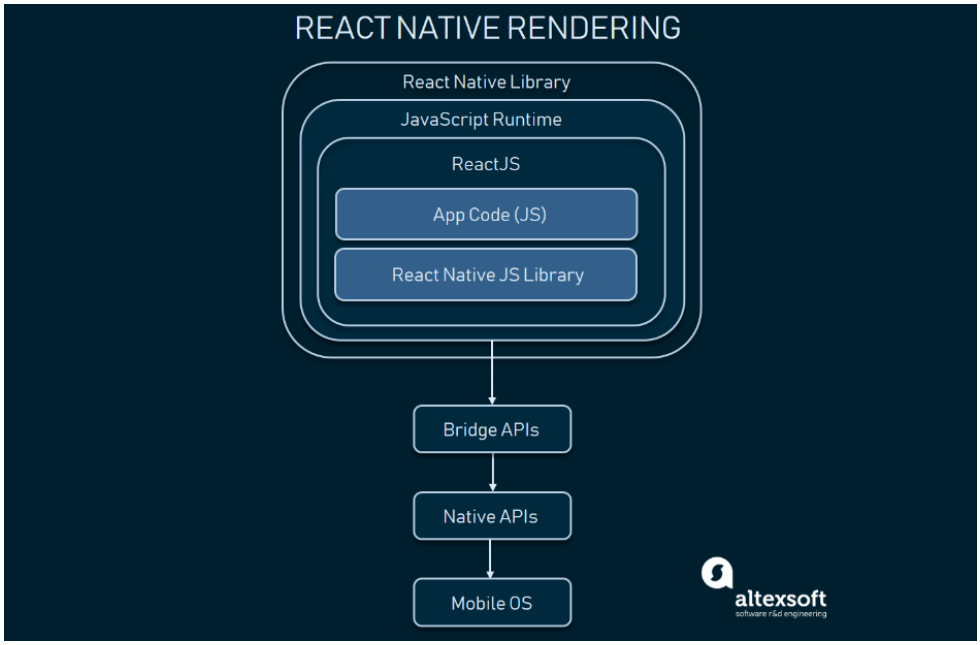
Výkon tohoto frameworku dosahuje výsledků, které by se daly srovnat s těmi nativními díky způsobu vykreslování vizuálních komponent do nativního rozhraní API. Zde je využíváno zobrazování pomocí vykreslovacího jádra webkit. U systému Android je nutné, aby React tento webkit nainstaloval, jelikož není jeho základní součástí na rozdíl od iOS. To může vést k navýšení velikosti aplikace pro tento systém. Použití Reactu nám rovněž dává možnost použít nativní moduly obou systémů, které jsou napsány v nativním jazyce. Připojení k nativním modulům vyžaduje vytvoření takzvaného bridge API pro oba operační systémy [21].

Hlavním principem frameworku je využití dvou vláken, kde jedno z vláken je hlavní a stará se o celkové zobrazení jednotlivých prvků uživatelského rozhraní včetně správy vstupů od uživatele. Druhé vlákno má na starosti samotnou funkcionalitu aplikace spuštěním JavaScript kódu a následným mapováním příslušných funkcionalit na prvky uživatelského rozhraní. Mezi těmito vlákny nedochází k přímé komunikaci ani k vzájemnému ovlivňování. Mezi těmito vlákny se nachází již zmíněný bridge, který tvoří srdce React Native. Samotný bridge umožňuje asynchronní komunikaci, díky které nedochází k vzájemnému blokování vláken a díky serializovatelnosti vlákna rovněž nepracují nebo nesdílí stejná data. Mezi vlákny dochází k výměně serializovaných zpráv, kde se bridge stará o to, aby tento přenos proběhl optimalizovaným způsobem [22].

Jednou z výhod je samozřejmě opětovné použití kódu, které React Native rovněž nabízí, což může vést k výraznému snížení nákladů na vývoj. React Native zajišťuje zjednodušené, nekomplikované a responzivní uživatelské rozhraní, které poskytuje skvělé UX s nízkou dobou načítání. Framework také podporuje doplňky třetích stran, které slouží jako doplnění chybějících komponent React Nativu. Může se například jednat o Google Maps [23].

Hodně zmiňovanou nevýhodou je frekvence aktualizací samotného frameworku. Aktualizace a vylepšení jsou vydávány v poměrně častých intervalech a pokud nebude docházek k pravidelným aktualizacím, může dojít ke ztrátě funkcionality aplikace. Podobným problémům v minulosti čelila populární aplikace Airbnb. Tento framework není příliš doporučován pro aplikace, které chtějí provádět složitější výpočty kvůli špatné správě paměti vycházející z použití JavaScriptu samotného. Obzvlášť float výpočty bývají zpracovány neefektivním způsobem, což má za důsledek špatnou správu a využití pamětí. Díky použití již zmíněného JavaScriptu a taky faktu, že se jedná o open source framework, React Native rovněž čelí problémů se zabezpečením. JavaScript by se dal označit za poměrně křehký programovací jazyk, což může mít za důsledek nízkou úroveň zabezpečení a u finančních nebo bankovních aplikací je nutné implementovat další vrstvy zabezpečení. Z toho důvodu se u tohoto typu aplikací nedoporučuje použití frameworku React Native [23].

Framework React Native byl použit při vývoji mnoha známých a populárních aplikací. Kromě velké skupiny aplikací od společnosti Facebook, jako je například Facebook Analytics nebo Instagram, byly pomocí tohoto frameworku vytvořeny aplikace jako Skype, Tesla, Oculus nebo Pinterest [24].



Obrázek 8: Architektura React Native [21]

2.2 Xamarin

Xamarin je open source, multiplatformní framework pro vývoj mobilních aplikací od společnosti Microsoft. Microsoft tento framework vyvíjí od roku 2011 [20]. Xamarin používá programovací jazyk C# společně s nativními knihovnami, které jsou obaleny do vrstvy .NET. Celková technologie tohoto frameworku je složena ze tří základních částí. Jedná se o Xamarin.Platform, Xamarin.Cloud a Xamarin.Insights [25].

Zřejmě nejdůležitější z těchto prvků je Xamarin.Platform. Tento prvek zprostředkovává například rozhraní API, šifrování, ověřování, ovládací prvky, běhové moduly a mnoho dalších. Oproti tomu Xamarin.Cloud zajišťuje kvalitu výsledné aplikace pomocí automatizované testovací platformy. Pro lepší sledování selhání aplikací nebo sledování výjimek je zde využíván monitorovací nástroj Xamarin.Platform [25]

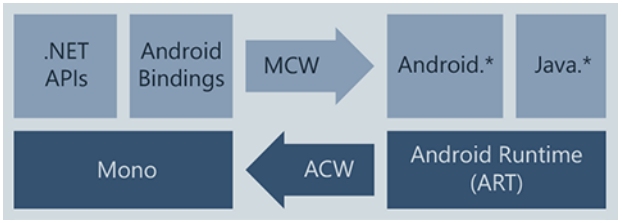
Xamarin při vytváření mobilních aplikací definuje dva přístupy při vývoji [21].

- Xamarin.Android a Xamarin.iOS
- Xamarin.Forms

U Xamarin.Android a Xamarin.iOS se aplikace chovají jako nativní, jelikož jejich schopnosti mezi jednotlivými platformami se soustředí především na sdílení obchodní lo-

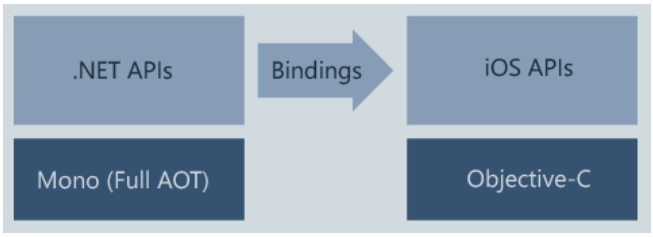
giky než na zdrojový kód sloužící k sestavení. Využití nativních ovládacích prvků uživatelského rozhraní společně s hardwarovou akcelerací pomáhá dosáhnout téměř nativního výkonu, což by nebylo možné při interpretaci kódu za běhu [21].

U aplikace Xamarin.Android dochází ke zkompilování z jazyku C# do takzvaného *Intermediate language* (IL), který je při spuštění aplikace zkompilován do nativního sestavení. Běh této aplikace probíhá paralelně s virtuálním strojem *Android runtime (*ART) v běhovém prostředí mono. Framework Xamarin se stará o definování vazeb .NET k oborům názvů Java. * a Android. *. Pro dosažení toho, aby obě prostředí vyvolaly kód v sobě navzájem, je využito volání ze strany prostředí pro spuštění mono, do oborů názvů pomocí *Managed Callable Wrappers (*MCW) [26].



Obrázek 9: Xamarin.Android [26]

U aplikace Xamarin.iOS dochází ke komprimaci *Ahead-of-Time* (AOT) z jazyku C# do nativního kódu. Hlavními prvky vazby jsou zde registrátory a selektory, které poskytují možnost komunikace mezi Objective-C a C# [26].



Obrázek 10: Xamarin.iOS [26]

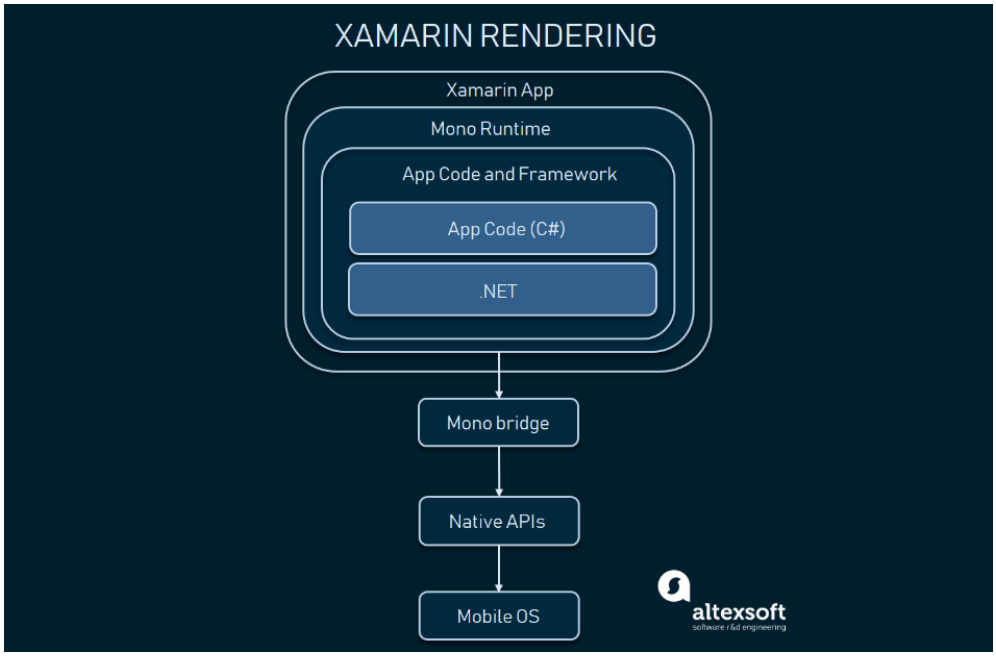
Oproti tomu Xamarin.Forms využívá sdílení kódu s menším důrazem na chování specifické platformy. Tímto přístupem se dá až 96 % zdrojového kódu znovu použít, ovšem za cenu výrazného snížení výkonu ve srovnání s Xamarin.Native nebo celkovým nativním vývojem [21].

Jednou z hlavních výhod frameworku Xamarin je bezesporu výkon srovnatelný s aplikacemi vytvořenými nativním vývojem. C# je v mnoha ohledech velice silný programovací jazyk, který si získal své místo na poli programovacích jazyků používaných pro vývoj mobilních aplikací a z toho důvodu jsou i aplikace vyvíjené pomocí tohoto frameworku schopné

konkurovat aplikacím vytvořených v Objective-C, Swiftu nebo Javě. Vývojáři mají k dispozici taky mnoho API, které dokáží integrovat mnoho nativních hardwarových komponent, díky kterým může dojít k výraznému zlepšení uživatelského prostředí. Další výhodou jsou již zmíněné Xamarin.Forms. Díky této technologii lze kód uživatelského rozhraní sdílet mezi aplikacemi obou operačních systémů. Jednotlivé sady nástrojů s mnoho ovládacími prvky a rozloženími jsou za běhu postupně mapovány na nativní ovládací prvky aplikace. Xamarin rovněž podporuje softwarové architektury *Model-view-controller* (MVC) a *Model–view–viewmodel (*MVVM). Nespornou výhodou je integrovaná podpora offline aplikací. Tato podpora byla dříve dominantou nativních aplikací, avšak synchronizace dat a funkce cloudu umožňují Xamarinu této podpory docílit také [21].

Xamarin bohužel po vývojářích vyžaduje korektní označování kódu mezi jednotlivými operačními systémy a frameworkem .NET. Důsledkem toho je nepříznivé ovlivnění doby nutné pro start aplikace a stažení dat, což může způsobit méně příjemný uživatelský zážitek z používaní aplikace. Framework se také potýká s problémem náročnosti vývoje uživatelského rozhraní. Ačkoliv bylo o technologii v předchozích odstavcích psáno víceméně v superlativech, i přesto jsou zde jisté nevýhody. Mnoho komponent umožnuje již zmíněné sdílení, ale stále jsou zde komponenty, u kterých je nutný separátní vývoj pro každou platformu. Vývoj uživatelského rozhraní je tak náročnější a zvyšuje nám časové náklady na vývoj. Xamarin také značně naráží v oblasti své složitosti a podpory kódu. Pro jeho použití je nutné mít znalosti na poli nativního vývoje, znalosti daných architektur a rozhraní. Tyto dovednostní požadavky mohou přinést problém s hledáním zkušených vývojářů v této oblasti, kteří nám v budoucnu budou danou aplikaci déle rozvíjet, opravovat a aktualizovat. I když Xamarin umí sdílet velké části kódu mezi platformami, je zde pořád nutnost naprogramování nativního kódu pro přizpůsobení pro specifickou platformu. Tato nutnost přináší hned dva problémy. Jedním z problémů může být hledání schopných vývojářů se znalostmi tohoto frameworku, druhým jsou problémy s velikostí souborů, které mohou negativně ovlivnit výkon aplikace [21].

Framework Xamarin byl použit při vývoji mobilních aplikací pro společnosti jako je například UPS, Alaska Airlines, BBC Good Food nebo Allscripts. [24].



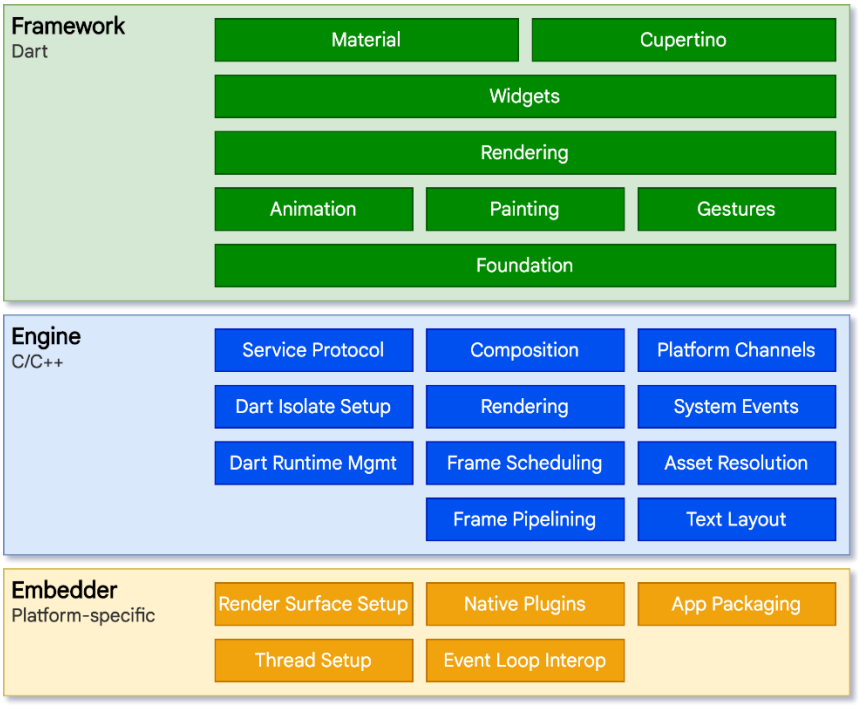
Obrázek 11: Architektura Xamarin [21]

2.3 Flutter

Flutter je open source, multiplatformní framework vytvořený společností Google v roce 2017 [19]. Tento framework byl původně vyvíjen pro operační systém Google Fuchsia, později došlo k přechodu na open source řešení s podporou více platforem. Flutter používá programovací jazyk Dart společně s kolekcemi nativních widgetů, *Command Line Interface* (CLI) nástroji a různými API [20].

Běh aplikace Flutter během samotného vývoje probíhá na virtuálním stroji, který umožňuje opakované načtení provedených změn bez potřeby provedení plné kompilace. Kompilace je prováděná přímo do zdrojového kódu dle instrukcí ARM nebo Intel x64. V případě cílení na web může být kompilováno do JavaScriptu. Jak bylo již zmíněno, jedná se o open source framework s *Berkeley Software Distribution* (BSD) licencí a podporou balíčků třetích stran, které doplňují základní funkcionalitu [27].

Flutter je navržený jako vrstvený systém s možností rozšiřitelnosti. Základem jsou podkladové vrstvy, na které je pevně navázána série nezávislých knihoven. Žádná vrstva nemá výhradní přístup k níže uvedené vrstvě, jednotlivé úrovňové části rozhraní jsou designované tak, aby se daly volit a nahradit [27].



Obrázek 12: Architektura Flutter [27]

Flutter aplikace je k základnímu operačnímu systému zabalena jako běžná nativní aplikace. Jednotlivé platformy obsahují specifický Embedder, který poskytuje vstupní bod. Jedná se o kooperaci se základním operačním systémem sloužící k přístupu k různým službám. Jedná se o služby jako je vykreslování povrchů nebo služby přístupnosti. Programovací jazyk použit pro tento vkládací program je zvolen podle vhodnosti pro danou platformu. V případě Androidu se jedná o jazyk Java nebo C++, u iOS a macOS se jedná o Objective-C / Objective-C++. Pro operační systémy Windows a Linux je použit jazyk C++. Embedder umožnuje modulovou integraci Flutter kódu do existující aplikace nebo může kód tvořit celý obsah aplikace [27].

Flutter engine je hlavním jádrem Flutteru. Tento engine je ve většině případů napsán v programovacím jazyce C++ a poskytuje podporu primitiv, které jsou nezbytné pro podporu Flutter aplikací. Engine se také stará o rasterizaci složených scén, jejich překreslování a zároveň poskytuje nízko úrovňovou implementaci Flutter API společně s grafikou, rozložením textu, souborových a síťových vstupů/výstupů, podpory přístupů nebo kompilovacích nástrojů [27].

Vystavení enginu do Flutter frameworku probíhá skrze dart:ui, které se postará o zabalení základního kódu v jazyce C++ v jednotlivých Dart třídách. Funkcí této knihovny

je vystavení primitiv nejnižší úrovně, jakou jsou například třídy řídící vstupy, grafiku a subsystém renderování textu [27].

Většina vývojářů obvykle Flutter obsluhuje pomocí frameworku Flutter, který přináší moderní framework v programovacím jazyce Dart s širokou sadou platforem, layoutů a základních knihoven [27]. Jednotlivé části jsou popsány od spodní vrstvy nahoru (viz obrázek 12).

- Základní třídy obsahují služby stavebních bloků. Mezi tyto bloky patří například gesta, animace a malování [27].
- Vykreslovací vrstva označená jako „Rendering“ má na starosti poskytnutí abstrakce pro práci s layoutem. Vrstva dále umožnuje tvorbu stromu vykreslovaných objektů, se kterými lze dynamicky manipulovat, přičemž strom je automaticky aktualizován [27].
- Další je vrstva widgetů, jedná se o tak zvanou kompoziční abstrakci. Jednotlivé objekty ve vykreslovací vrstvě mají ve vrstvě widgetů odpovídající třídu. Díky této vrstvě je také možné definovat kombinace jednotlivých tříd za účelem jejich znovu použití [27].
- Knihovny Material a Cupertino nabízejí komplexní sady ovládacích prvků, které k implementaci návrhových jazyků Material nebo iOS používají primitiva kompozice vrstvy widgetu [27].

Flutter aplikace obecně dosahují dobrých výsledků výkonu díky předem kompilovanému jazyku Dart. Tento programovací jazyk umožňuje přímou komunikaci aplikace s nativní platformou bez nutnosti využívat JavaScript most jako tomu je například u Reactu Nativ. Díky tomuto mohou vývojáři vyvíjet složité aplikace bez negativního vlivu na výkon nebo dobu spuštění. Funkce hot reload poskytuje možnost bezprostředního znovuvytvoření aplikace, jako by se jednalo o webovou stránku. Tento hot reload je stavový a díky tomu není nutné aplikaci při každé změně restartovat a lze pokračovat tam, kde vývojář skončil. Další výhodou frameworku Flutter je obsáhlá sada jedinečných widgetů, díky kterým není nutné spoléhat na prvky uživatelského rozhraní jednotlivých platforem. Framework implementuje widgety Material Design pro Android platformu, v případě platformy iOS jsou použity widgety Cupertino. Každý prvek na obrazovce je ve Flutteru widget. Tímto je velice zjednodušené rozložení aplikace, jelikož je každý widget určený vlastním modelem rozložení namísto použití sady pravidel pro všechny widgety. Flutter rozložení je ve výsledku

poměrně malé, což umožňuje snadnější optimalizaci, a kompletní rozvržení aplikace je tak snadno zpracovatelné. Nativní widgety rovněž přináší výhody na poli kompatibilit. Tyto nativní widgety poskytují aplikacím možnost přetrvat kompatibilní s novými verzemi operačních systémů, jelikož Flutter nezasahuje do nativních widgetů specifických platforem. I když je Flutter jeden z nejmladších frameworků, tak si za krátkou dobu získal velkou oblibu mezi vývojáři a díky tomu je již k dispozici spousta balíčků například pro zpracování obrázků, připojení WebSocket, push notifikací požadavků *Hypertext Transfer Protocol* (HTTP) nebo vestavěných databází [25].

Již zmíněné nedávné vytvoření frameworku lze označit i za jednu z jeho nevýhod. Už v roce 2018 bylo společností Flutter oznámeno, že je tento Framework připraven k produkčnímu využití, ale i nyní je těžké odhadnout, jak to bude s úspěšností tohoto frameworku v budoucnu, tudíž i jak to bude s aplikacemi vyvíjenými Flutterem. I když byla v předchozím odstavci zmíněna působivá sbírka již dostupných knihoven, pořád existují funkcionality, které nejsou úplně dotaženy a bude nějakou dobu trvat, než se tento framework dostane funkčně na úroveň například již zmíněného Reactu Native. Jednou z nevýhod může být velikost aplikace. Z důvodu absence mostu mezi aplikací a zařízením jsou všechny komponenty uživatelského rozhraní obsaženy přímo v aplikaci [25].

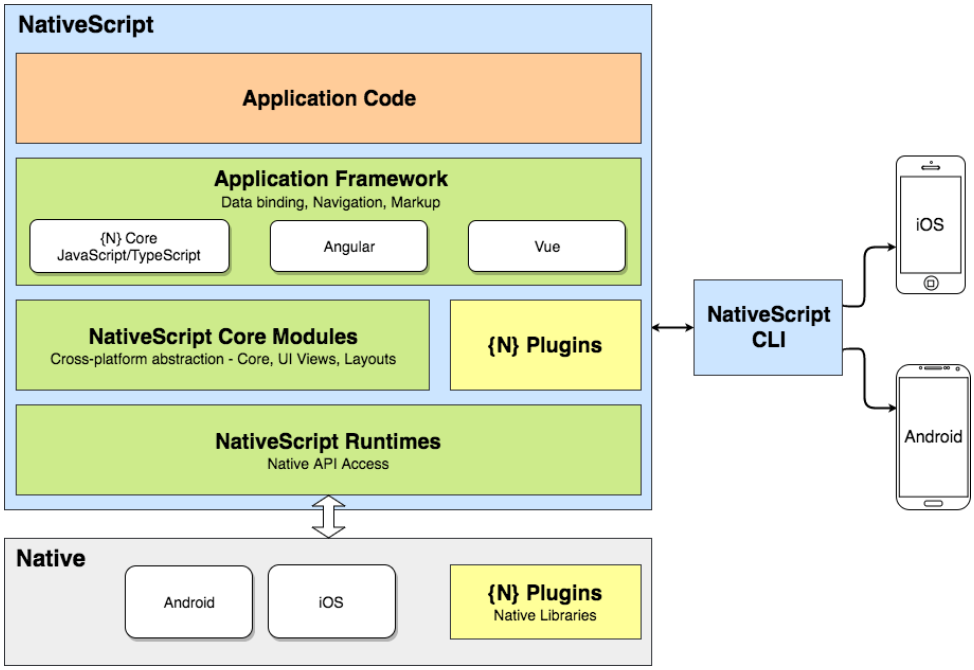
Framework Flutter byl použit při vývoji mobilních aplikací pro společnosti, jako je například BMW, ebay, SONOS nebo The New York Times. [24].

2.4 NativeScript

NativeScript je open source framework pro vývoj mobilních aplikací od společnosti Progress, která začala s vývojem v roce 2014. Stejně jako ReactNative se NativeScript zaměřuje na webové vývojáře, kteří mají zkušenosti s jazykem JavaScript. Pro tvorbu uživatelského rozhraní jsou dostačující znalosti HTML a CSS, pro vytvoření obchodní logiky již zmíněný JavaScript. Kromě JavaScriptu je možné použít Angular, TypeScript nebo Vue.js [28]. Pomocí tohoto frameworku lze vyvíjet aplikace pro Android, iOS nebo univerzální platformu Windows [29].

Technologii NativeScript lze rozdělit na několik hlavních částí. Mezi tyto hlavní části patří Runtime, Core Modules, CLI a Plugins [30]. Tyto jednotlivé části lze vidět na následujícím obrázku č. 13. Kód JavaScriptu je interpretován a spouštěn pomocí virtuálního stroje, následně dochází k převodu volání na API volání, typické pro danou platformu. K tomuto

převodu je využíván modul mostu. NativeScript umožňuje při vývoji ovládat prvky nativní platformy prostřednictvím programovacího jazyku JavaScript namísto Objective-C nebo Java [29].



Obrázek 13: Architektura NativeScript [30]

- Runtimes
  - Díky modelům runtimes lze prostřednictvím JavaScriptu volat API rozhraní v rámci iOS a Androidu. Pro Android je využíván *JavaScript Virtual Machines – V8* od Googlu a jako další webkity *JavaScriptCore* distribuované s iOS 7.0+ [30].
- Core moduly
  - Core moduly slouží k poskytnutí potřebné abstrakce, která je nutná k přístupu k nativním platformám. Jako příklad lze využít modul gest. Tento modul formuje jednotné JavaScript API, které se následně starají o převod kódu aplikace v jazyku JavaScript nebo TypeScript do API volání nativních gest. Tyto moduly se rovněž starají o definování uživatelského rozhraní, datové vazby nebo navigace pomocí *Extensible Markup Language* (XML) [30].
- NativeScript CLI
  - Pomocí tohoto rozhraní příkazového řádku lze vytvářet, dělat build a spouštět aplikace prostřednictvím NativeScriptu. Toto rozhraní je dostupné na operačním systému Windows, macOS i Linuxu [30].

- NativeScript pluginy
    - Jedná se o základní stavební bloky, které se starají o zapouzdření některých funkcionalit a zrychlují vývoj samotných aplikací. Většina z těchto pluginů je vytvořena komunitou vývojářů a jako programovací jazyk je použit TypeScript nebo JavaScript [30].

Výhodou NativeScriptu je jeho kompatibilita napříč různými technologiemi. Lze použít Angular, TypeScript nebo JavaScript, díky kterým je možné komfortně vázat data a opětovně používat komponenty. K vývoji aplikací s přizpůsobenými funkcemi je rovněž využíván značkovací jazyk, jako je například HTML. NativeScript uděluje možnost přímého přístupu ke všem API rozhraním jak u Androidu, tak iOS, což poskytuje příležitost opětovného použití bezplatných pluginů. U Androidu se jedná o samotné SDK, u iOS o CocoaPods. Další výhodou může být přístup k nativním funkcím. Díky nativním komponentám s nativním výkonem je umožněn přístup k nativnímu API zařízení. NativeScript také poskytuje vhodné CLI rozhraní pro vývojáře. CLI poskytuje funkcionalitu, která dovoluje vývojářům například přidávat platformu nebo provést nasazení aplikace na zvolenou platformu či zařízení. Rovněž instalace pluginů nebo ladění aplikací je daleko rychlejší a komfortnější [31].

První nedostatky se objevují u uživatelského rozhraní. Podpora HTML a *Document Object Model* (DOM) není na vysoké úrovni a je nutné použití různých prvků uživatelského rozhraní, což může vést k vyšším časovým nákladům. U pluginů naopak nastává problém s jejich ověřením. Velká část pluginů není ověřená, tudíž zde nejsou záruky na jejich kvalitu zpracování a použití u tohoto frameworku. Dále je nutné, aby měl vývojář znalosti a povědomí o nativních funkcích a API rozhraních u jednotlivých platforem. Bez těchto znalostí je velice těžké přistupovat k hardwaru zařízení nebo k jakýmkoli prvkům specifických pro konkrétní platformy. Jednou z dalších nevýhod může být také pomalé počáteční testování. Přesné testování je možné pouze na emulátoru nebo fyzickém zařízení, což může značně zpomalit rychlost počátečního testování [31].

Framework NativeScript byl použit při vývoji mobilních aplikací, jako je například Puma, Airbnb Design, Sennheiser nebo MDBootstrap. [31].

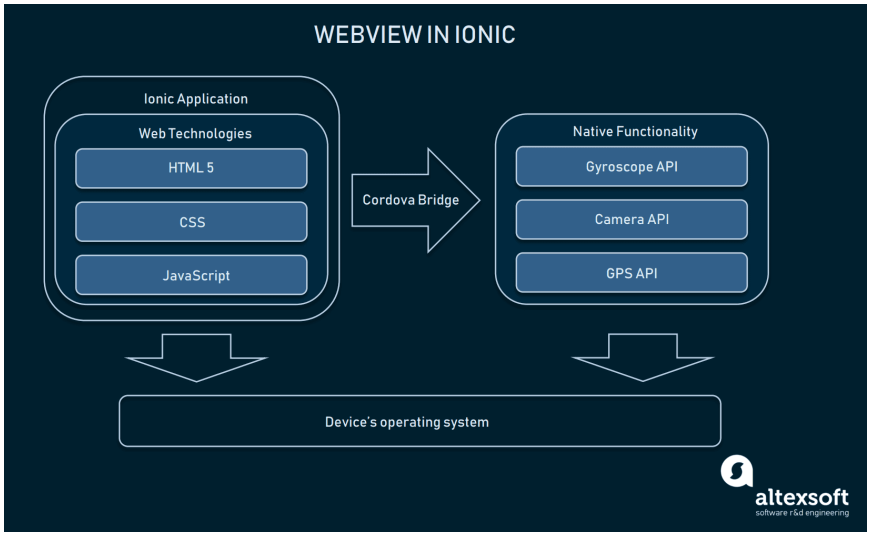
3 IONIC FRAMEWORK

Ionic je open source front-end SDK framework pro vývoj hybridních mobilních aplikací. K vývoji jsou použity webové technologie jako HTML, CSS, JavaScript, Angular a TypeScript [32]. Ionic používá vlastní knihovnu nativních komponent uživatelského rozhraní pro iOS a Android, která poskytuje jednotný a funkční vzhled na obou platformách. Díky tomuto frameworku vzniklo celosvětově přes 5 miliónů mobilních aplikací [33].

Hlavním zaměřením frameworku Ionic je *User Interface* (UI) a UX. Jedná se například o prvky uživatelského rozhraní, jako jsou gesta a animace. Výhodou je snadná integrace s jiným knihovnami a frameworky, jako je Angular, Vue a React, nebo jej použít zcela samostatně bez použití front-end rozhraní. Díky Ionicu lze vytvářet pomocí webových technologií aplikace pro všechny nynější obchody s aplikacemi za použití jednoho kódu [34].

Funkce adaptivního stylování poskytují aplikacím schopnost zapadnout do ekosystému pro danou platformu, což umožnuje, aby Ionic dobře fungoval na všech současných mobilních zařízeních a platformách. Velkým benefitem jsou předpřipravené komponenty, typografie a základní motivy, které je možné rozšiřovat a jsou přizpůsobovány zvolené platformě [34].

Ionic emuluje pokyny pro uživatelské rozhraní nativních aplikací a používá nativní sady SDK. Díky tomuto přináší standardy uživatelského rozhraní a funkce nativních aplikací spolu s plnou mocí a flexibilitou otevřeného webu. Pro nativní nasazení aplikace Ionic využívá Capacitor nebo Cordovu, v případě běhu v prohlížeči se jedná o progresivní webovou aplikaci [34].



Obrázek 14: Architektura Ionic (Cordova) [33]

3.1 Kompatibilita frameworků

Starší verze Ionicu byly pevně spjaty s frameworkem Angular. Tato věc se změnila od verze 4.x, kdy došlo k přepracování frameworku tak, aby framework fungoval jako samostatná knihovna webových prvků a bylo do něj možné integrovat nejnovější verze frameworků, jako je například již zmiňovaný Angular. Díky tomu je možné Ionic použít ve spolupráci s front-endovými frameworky, jako je React nebo Vue [34].

3.1.1 JavaScript

Přechod Ionic Frameworku na webové komponenty mělo za cíl eliminovat náročné požadavky, které cílily na jediný framework sloužící k hostování komponent. Tento přechod umožnil, aby na webové stránce tyto součásti fungovaly samostatně, pouze za použití script tagu. Ionic může být použit v rámci velkých týmů a větších aplikací, kteří především ocení práci s jednotlivými frameworky, rovněž je zde možnost využití Ionicu jako samostatné knihovny v kontextu single page [34].

3.1.2 Angular

Právě Angular stojí za celkovým úspěchem frameworku Ionic. Zatímco základní komponenty byly napsány tak, aby fungovaly jako samostatná knihovna webových komponent, balíček @ionic/angular se stará o samotnou integraci ekosystému Angular. Tento balíček obsahuje veškerou funkcionalitu, kterou by Angular vývojáři očekávali díky integraci hlavních knihoven Angularu, jako je například Angular router [34].

3.1.3 React

Ionic disponuje oficiální podporou populární knihovny React. Díky této podpoře mohou vývojáři pracující s touto knihovnu využít své dovednosti k vytváření multiplatformních mobilních aplikací nebo desktop aplikací s využitím frameworku Ionic společně s knihovnou React. Pomocí @ionic/react lze používat všechny komponenty obsažené ve frameworku Ionicu způsobem, který je typický pro použití nativních komponent React [34].

3.1.4 Vue

Ionic má nyní oficiální podporu pro populární knihovnu Vue 3. Tato podpora umožnuje, aby Vue vývojáři byli schopní využít své zkušenosti v rámci této knihovny pro vývoj

mobilních aplikací pro iOS a Android nebo aplikace zaměřené na web a desktop ve spolupráci s frameworkem Ionic. Vue vývojářům je tak umožněno využívat základní Ionic komponenty způsobem, na který jsou zvyklí z použití nativních komponent Vue [34].

3.2 Nativní API

Framework Ionic umožňuje přidávání nativních funkcí do zařízení díky nativním API, které obsahují balíčky open source a prémiových pluginů. V současné verzi lze volit mezi použitím Apache Cordova nebo Capacitoru [34].

3.2.1 Apache Cordova

Apache Cordova je open source nativní runtime, prostřednictvím kterého je možné vytvářet nativní mobilní aplikace s použitím technologií, jako je HTML, CSS a JavaScript. Díky tomu, že Cordova stejně jako Capacitor obsahuje vlastní nativní runtime, je vývojářům poskytnut přístup k nativním funkcím zařízení. Může se jednat například o geolokaci, kameru nebo klávesnici. Přístup je zprostředkován pomocí systémových pluginů. Cordova obsahuje potřebné nástroje pro vytváření webových aplikací pro iOS, Android a Windows Phone [34].

3.2.2 Capacitor

Capacitor je multiplatformní, nativní runtime, díky kterému lze pohodlně vytvářet nové webové aplikace s následným nativním během na systémech Android, iOS nebo na webu. Pomocí Capacitoru jsou vytvářeny aplikace Web Native a je poskytnut přístup k nativním kontejnerům. To umožnuje vývojářům stavět nejprve na webu bez nutnosti obětování plného přístupu k SDK [35].

Capacitor přináší sadu API rozhraní, která se zaměřují především na web. Tato sada přispívá k tomu, že Capacitor zůstává velice blízko webovým standardům, ale zároveň přináší přístup k mnoha nativním funkcím na cílených platformách. Nativní funkce jsou přidány díky API pluginu. U zařízení iOS se jedná o plugin pro Swift, u Androidu pro Javu a u webu pro JavaScript [35].

Capacitor je koncipován jako nástupce Apache Cordova a Adobe PhoneGap s inspirováním z populárních nástrojů, jako je React Native nebo Turbolinks. V případě Apache

Cordova je Capacitor zpětně kompatibilní s mnoha již vytvořenými pluginy. Hlavním záměrem je, aby moderní webové aplikace snadno a spolehlivě fungovaly na všech hlavních platformách [35].

3.3 Ionic CLI

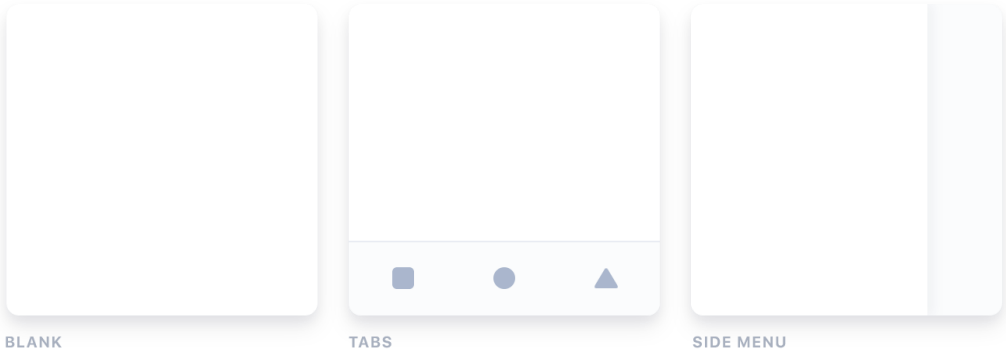
Ionic CLI nebo rozhraní příkazového řádku je oficiální nástroj, který poskytuje vývojářům mnoho užitečných příkazů, které lze použít při vývoji aplikace. Kromě toho, že se tento nástroj využívá pro instalace, aktualizace nebo vytváření nových projektů, Ionic CLI přináší integrovaný vývojový server nebo nástroje pro sestavování a ladění aplikace. Ionic CLI je upřednostňovaný způsob instalace, protože nabízí širokou škálu nástrojů pro vývojáře a možnosti nápovědy. Je také hlavním nástrojem, pomocí kterého lze aplikaci spouštět a připojovat k dalším službám, například Appflow. V případě, že je vývojář člen Ionic Appflow, je možné pomocí tohoto CLI provádět cloudové sestavení, nasazení nebo ho využívat ke správě vlastního účtu [34].

3.3.1 Instalace a vytvoření aplikace

Před instalací samotného Ionicu je nutné si ověřit, zda je v počítači instalován Node.js, protože bez něj nebude možné Ionic CLI nainstalovat. Pro instalaci stačí zadat do příkazového řádku příkaz *npm install -g @ionic/cli* [34]. Po provedení tohoto příkazu je vše připraveno a lze přistoupit k prvotnímu vytvoření naší aplikace.

Pro vytvoření lze zvolit dvě cesty. V mnoha návodech najdeme příkaz *ionic start name template.* Kde name označuje název naší aplikace a template název použité šablony v naší aplikaci. Nejčastěji jsou uváděny tyto šablony:

- Tabs: Rozložení založené na kartách
- Sidemenu: Rozložení založené na vedlejší nabídce
- Blank: Prázdný projekt s jedinou stránkou



Obrázek 15: Šablony při vytvoření aplikace Ionic [34]

Po zvolení názvu naší aplikace a šablony, kterou jsme se rozhodli použít, přichází na řadu volba vývojového frameworku pro naši aplikaci. Zde máme na výběr volit z již umíněného Angularu, Reactu a Vue. Po poměrně krátkém stahování všech potřebných balíčků je aplikace připravena k prvotnímu spuštění. Druhý způsob vytvoření se liší pouze v použití příkazu *ionic start*, který nás provede stejným nastavením aplikace, ovšem z mého pohledu intuitivnějším.

Pro prvotní spuštění aplikace stačí pouze vstoupit do složky naší aplikace pomocí příkazu *cd myApp* a následné použití příkazu *ionic serve*. Po zadání druhého příkazu dojde ke spuštění naší aplikace ve webovém prohlížeči. Za zmínku stojí, že po každé změně v našem kódu dojde k automatickému obnovení a změny jsou okamžitě zobrazeny v prohlížeči.

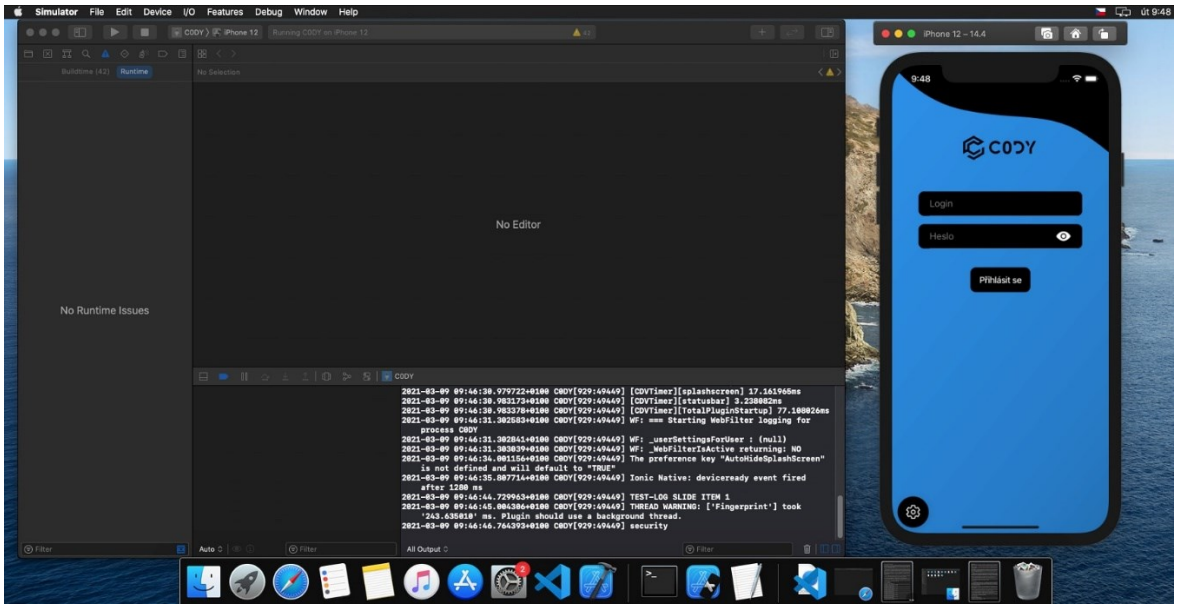
3.4 Vývoj

Výhodou je, že velká část vývoje může být prováděna právě ve webovém prohlížeči po zadání *ionic serve* do CLI. Díky tomu, že prohlížeče, jako jsou Google Chrome, Safari nebo Firefox, obsahují dev tools, lze námi napsaný kód rychle otestovat bez nutnosti kompilace a následného buildu v simulátoru nebo fyzickém zařízení [34].

Pro otestování na fyzickém zařízení nebo simulátoru zařízení je nutné prvně přidat zvolenou platformu. Pokud používáme Cordovu, jedná se o příkaz *ionic cordova platform add platformName*.

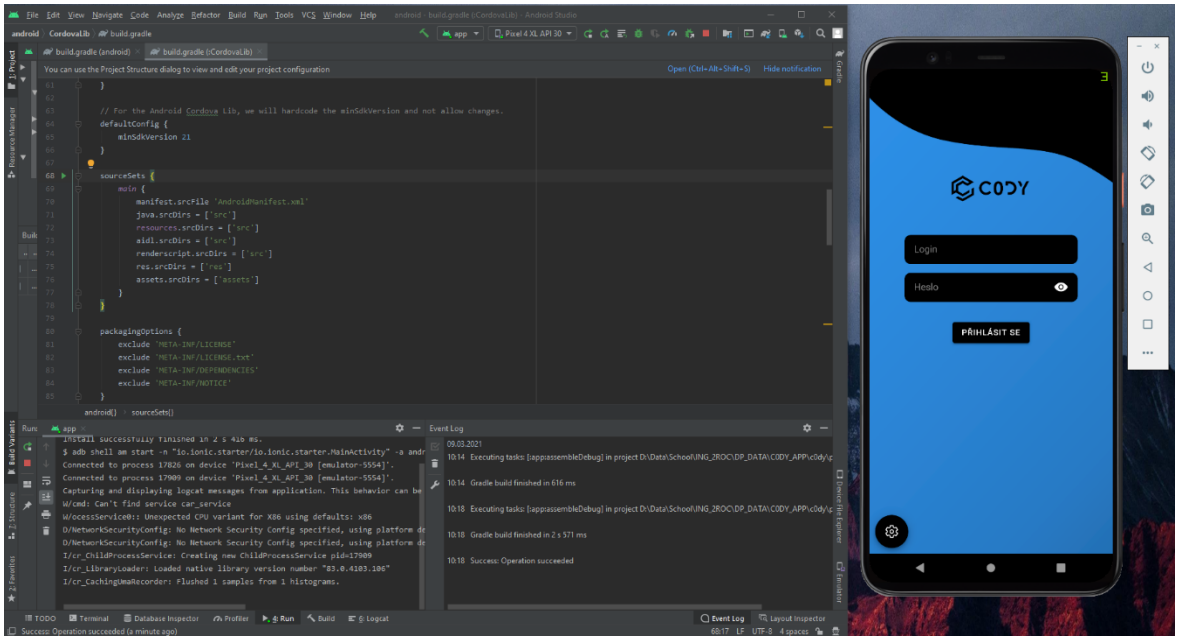
U operačního systému iOS se jedná konkrétně o příkaz *ionic cordova platform add ios.*U vývoje pro systém iOS je třeba zmínit, že je nutné mít zařízení s macOS společně s instalací *Integrated Development Environment* (IDE) Xcode. Následný build se provádí přes příkaz *ionic cordova build ios*. Po úspěšném sestavení lze aplikaci v Xcodu otevřít

a případně spustit v integrovaném simulátoru. U integrovaného simulátoru si lze zvolit konkrétní typ zařízení a poté provést build na toto simulované zařízení.



Obrázek 16: Simulátor zařízení iPhone 12 v IDE Xcode

U operačního systému Android je příkaz obdobný, a to *ionic cordova platform add android*. U Androidu není nutné řešit prostředí, ve kterém vývojář pracuje. Pro vývoj je používáno IDE Android Studio, které je možné používat jak z prostředí Windows, tak například macOS. I příkaz pro provedení buildu je shodný *ionic cordova build android.* Kromě klasického buildu na fyzické nebo simulované zařízení je možné vytvoření souboru *Android application package* (APK), pomocí kterého můžeme aplikaci nainstalovat na libovolné Android zařízení.



Obrázek 17: Simulátor zařízení Pixel 4 v IDE Android Studio

3.5 UI komponenty

Ionic aplikace používají balíček vlastních, velice kvalitních UI komponent, které značně zrychlují tvorbu uživatelského rozhraní aplikace [34]. Mezi tyto komponenty patří například modální okna, navigační lišty, seznamy a mnoho dalších. Velkou výhodou je změna stylování jednotlivých komponent podle operačního sytému zařízení. Komponenty tak skvěle zapadají do nativního prostředí iOS i Androidu [36].

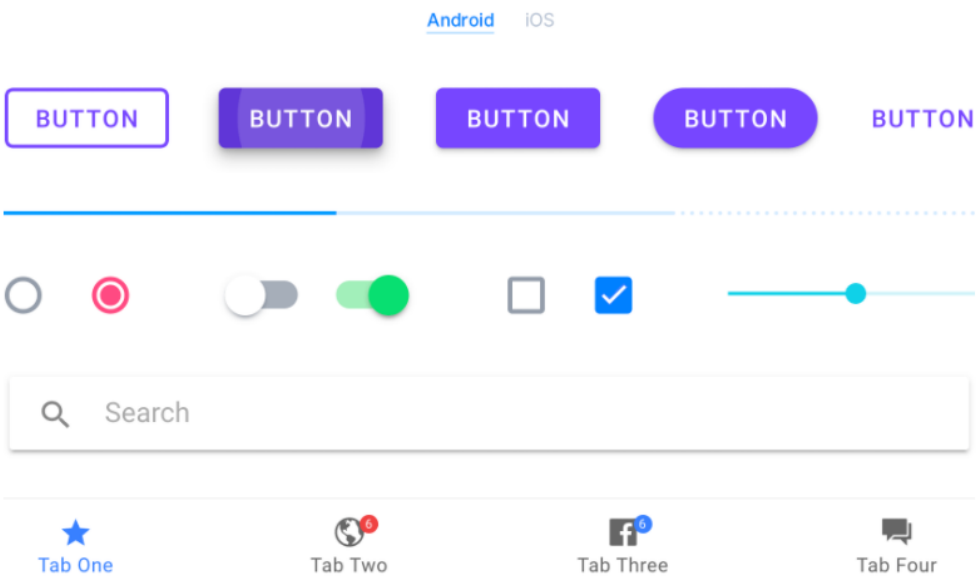
- Action Sheets
  - Action sheets zobrazují sadu možností se schopností akci potvrdit nebo zrušit.
- Alerts
  - Alerts představují skvělý způsob, jak uživateli nabídnout možnost vybrat si konkrétní akci nebo seznam akcí. Alerts lze přirovnat k potvrzovacím vyskakovacím oknům ze standardních webových aplikací.
- Badges
  - Badges jsou malou součástí zobrazení, která uživateli obvykle sděluje číselnou hodnotu.
- Buttons
  - Buttons umožňují uživatelům vykonávat akce. Jsou to základní způsoby interakce s aplikací a procházení aplikací.

- Cards
    - Cards jsou skvělým způsobem, jak zobrazit důležitý obsah. Mohou obsahovat obrázky, tlačítka, text a další prvky.
- Checkboxes
    - Checkboxes umožňují výběr více možností ze sady možností.
- Chips
    - Chips představují kompaktní způsob zobrazení dat nebo akcí.
- Content
    - Content je zásadním způsobem interakce s aplikací a procházení aplikací.
- Date & time picker
    - Date & time picker se používají k zobrazení rozhraní, které umožňují výběr data a času.
- Floating action buttons
    - Floating action buttons jsou kruhová tlačítka, která provádějí primární akci na obrazovce.
- Icons
    - Ikony navržené pro použití v aplikacích pro web, iOS, Android a desktop.
- Grid
    - Grid je výkonným systémem založeným na mobilních zařízeních pro vytváření vlastního rozvržení prvků.
- Infinite scroll
    - Infinite scroll umožňuje načítat nová data, když uživatel roluje aplikací.
- Inputs
    - Inputs umožňují uživatelům zadávat data do aplikace.
- Items
    - Items jsou univerzální kontejnery uživatelského rozhraní, který lze použít jako součást seznamu.
- Lists
    - Lists mohou zobrazovat řádky s informacemi, jako je seznam kontaktů, seznam skladeb nebo nabídku.
- Navigation

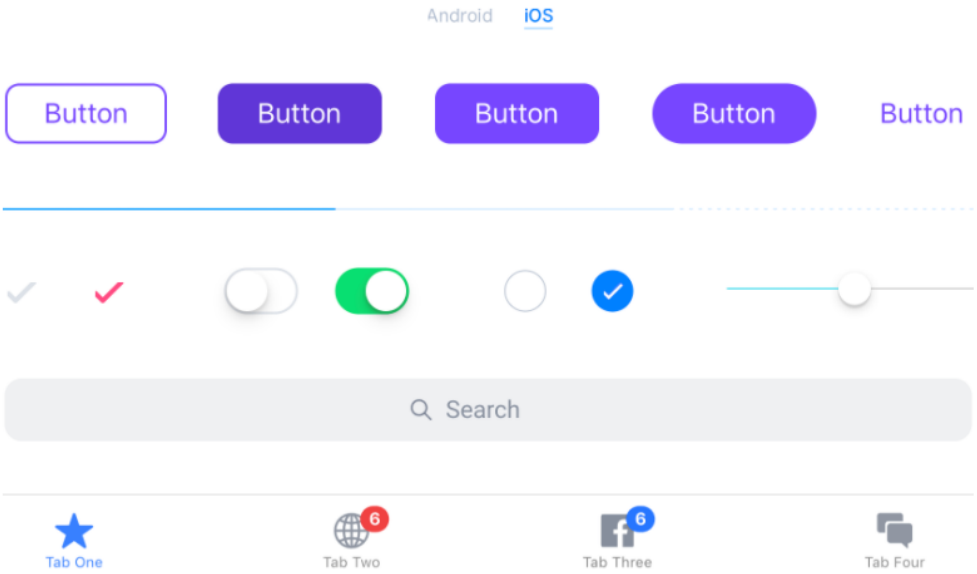
  - Navigation je způsob, jakým se uživatelé pohybují mezi různými stránkami v aplikaci.
- Menu
  - Menu je běžným navigačním prvkem. Může být trvale zobrazeno na obrazovce nebo v případě potřeby skryto.
- Modal
  - Modals se posouvají na obrazovce a mimo ni, aby zobrazily dočasné uživatelské rozhraní. Časté použití je u přihlašovací nebo registrační stránky.
- Popover
  - Popover poskytuje snadný způsob, jak prezentovat informace nebo možnosti beze změny kontextu. Lze ho přirovnat ke standardnímu HTML dropdown menu.
- Progress indicators
  - Progress indicators vizualizují postup operace nebo činnosti, například načítání.
- Radio
  - Radio inputs umožňují zobrazit seznam možností, ze kterých si uživatel zvolí jednu z nich.
- Refresher
  - Refresher poskytuje funkci pull-to-refresh komponenty obsahu.
- Reorder
  - Reorder umožňuje uživatelům přetahovat a měnit pořadí položek.
- Routing
  - Routing umožňuje navigaci na základě aktuální cesty.
- Searchbar
  - Search Bar se používá k vyhledávání nebo filtrování položek, obvykle z panelu nástrojů.
- Segments
  - Segments poskytují sadu exkluzivních tlačítek, která lze použít jako filtr nebo přepínač zobrazení.
- Select

- Select je podobný nativnímu HTML select, s několika vylepšeními pro řazení a výběr.

- Slides
  - Slides usnadňují vytváření komplexních uživatelských rozhraní, jako jsou galerie, výukové programy a rozložení na stránce.
- Tabs
  - Tabs umožňují navigaci pomocí karet. Jedná se o standardní navigační vzor v moderních aplikacích.
- Toast
  - Toast se používá k zobrazení oznámení přes obsah aplikace.
- Toggles
  - Toggles jsou vstupem pro možnosti true & false, často se používají pro možnosti a přepínače.
- Toolbars
  - Toolbars se používají k uchovávání informací a akcí týkajících se aplikace.



Obrázek 18: Příklad Ionic komponent pro Android [37]



Obrázek 19: Příklad Ionic komponent pro iOS [37]

II. PRAKTICKÁ ČÁST

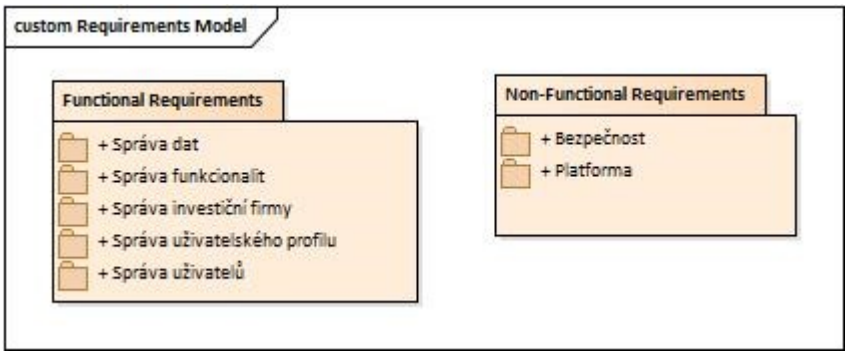
4 MOBILNÍ APLIKACE PRO PODPORU PRODEJE

Mobilní aplikace bude sloužit jako pomocný nástroj pro podporu prodeje komodit firem ve finančním sektoru. V případě firem použitých pro demonstraci v této práci se jedná o firmy, které zprostředkovávají investiční nabídky pro své klienty. Obecně má aplikace sloužit již registrovaným uživatelům, využívajícím webovou aplikaci této firmy, a tato mobilní aplikace jim zprostředkuje rychlý a přehledný přístup ke svým investicím, novým investičním nabídkám a dalším přehledům svých aktivit.

Úvod této části je věnován analýze funkčních a nefunkčních požadavků s jejich stručným popisem. Součástí je také Use Case Diagram s jednotlivými scénáři a návrh uživatelského rozhraní v programu Adobe XD. Závěr této části je věnován popisu použitých datových entit a samostatné kapitole o implementačně zajímavých částech při vývoji aplikace.

4.1 Analýza požadavků mobilní aplikace

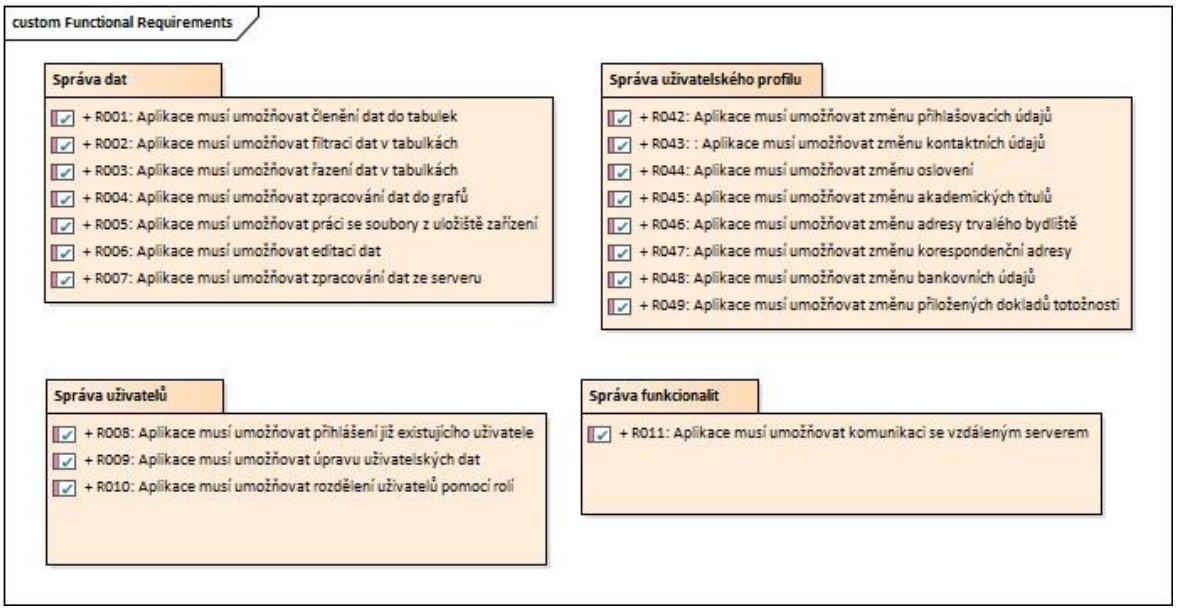
Tato kapitola obsahuje seznam funkčních a nefunkčních požadavků mobilní aplikace zpracovaných pomocí programu Enterprise Architects. Zpracování těchto požadavků stanovilo nezbytnosti, které musí výsledná aplikace obsahovat. Z výsledných požadavků bylo možné stanovit počet potřebných oken aplikace a také seznam funkcionalit, u kterých bylo prvně nutné zmapovat možná řešení a posléze z těchto možností vybrat řešení, které se skrz své vlastnosti, výhody a nevýhody nejvíc hodilo pro vyvíjenou aplikaci.



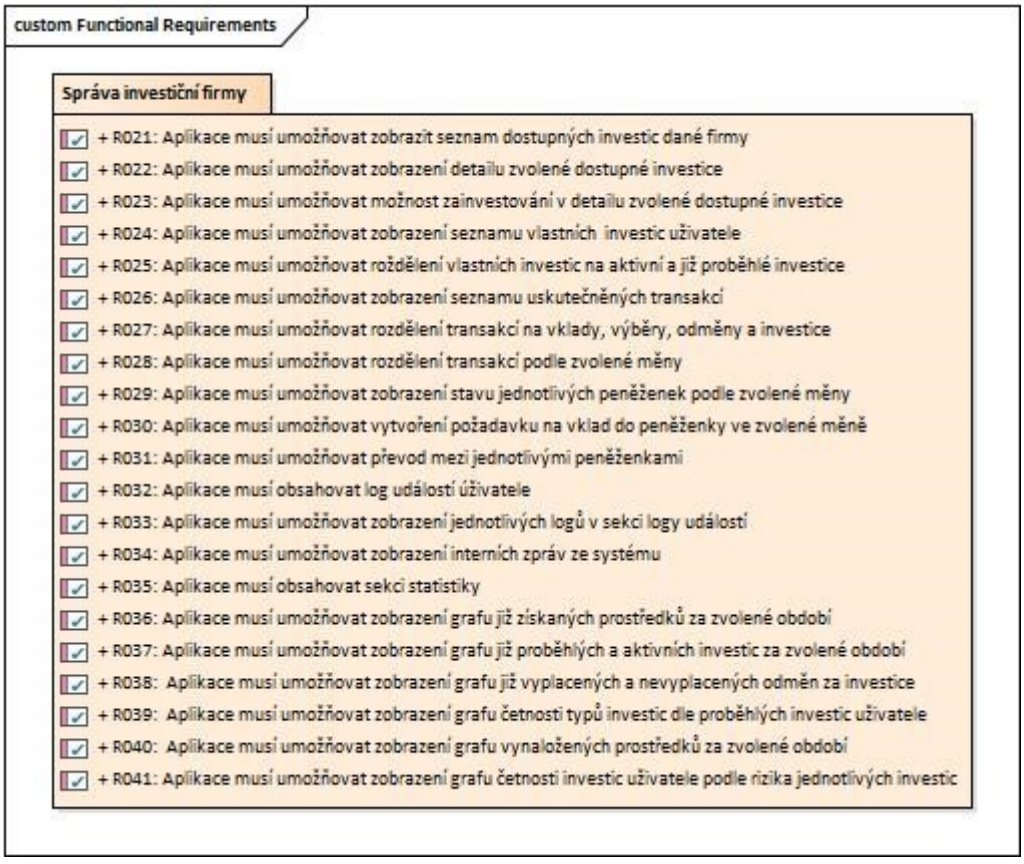
Obrázek 20: Analýza požadavků

4.1.1 Funkční požadavky

Funkční požadavky jsou rozděleny do několika kategorií podle typu požadovaných funkcionalit. Mezi tyto kategorie patří správa dat, správa uživatelů, správa funkcionalit, správa uživatelského profilu a správa investiční firmy.



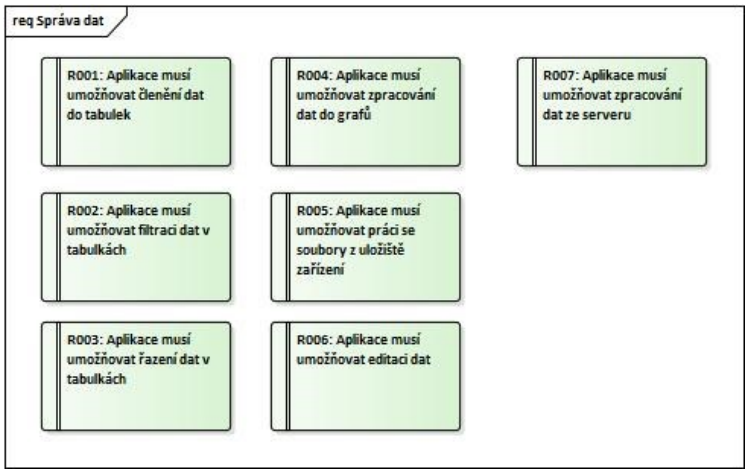
Obrázek 21: Funkční požadavky aplikace



Obrázek 22: Funkční požadavky aplikace

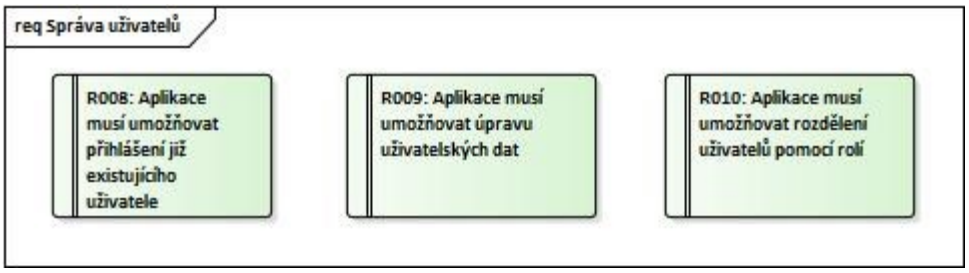
První kategorií funkčních požadavků je správa dat. Tato kategorie reprezentuje skupinu sedmi funkčních požadavků, které podmiňují potřebné funkcionality pro práci a úpravu dat v aplikaci. Z těchto požadavků vychází, že aplikace musí umět členit data do tabulek a v těchto tabulkách tyto data umět řadit a filtrovat. U některého typu dat musí aplikace umět

tyto data zobrazit v podobě grafů, ať už se jedná o grafy kruhové nebo spojnicové. Poslední dva požadavky podmiňují, že aplikace bude muset umět tato data získat a zpracovat ze serveru a rovněž musí poskytnout vhodnou funkcionalitu pro nahrání dat z uložiště zařízení.



Obrázek 23: Funkční požadavky – správa dat

Druhá kategorie se věnuje správě uživatelů. Požadavky stanovují, že do aplikace musí mít přístup již registrovaní uživatelé z webové aplikace. Těmto uživatelům musí být umožněna úprava jejich dat v profilu uživatele a jejich oprávnění bude na pozadí rozděleno dle odpovídajících rolí.

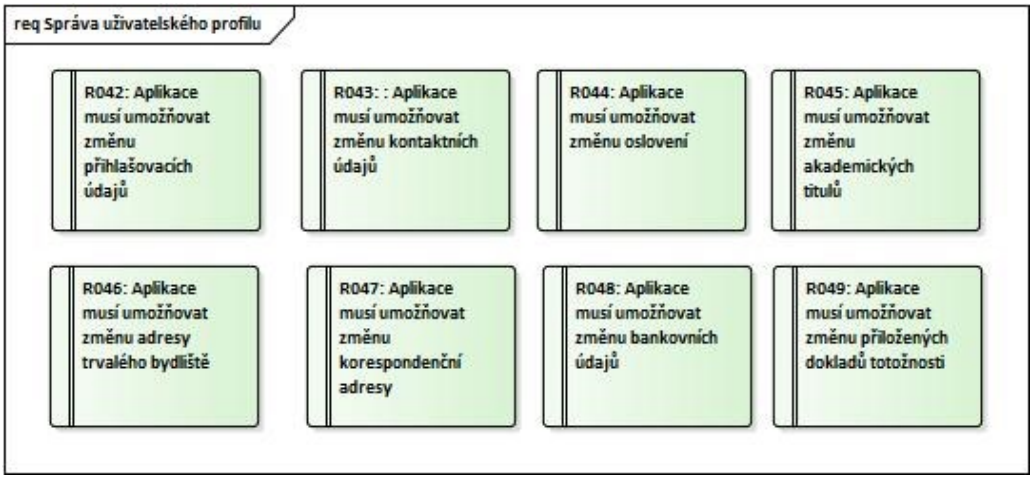


Obrázek 24: Funkční požadavky – správa uživatelů

Správa funkcionalit obsahuje obecné požadavky na aplikaci, pro které nebylo nutné vytvářet samostatnou kategorii. Mezi tyto požadavky se například řadí podmíněná komunikace se vzdáleným serverem, jelikož se jedná o API based aplikaci.

Obrázek 25: Funkční požadavky – správa funkcionalit

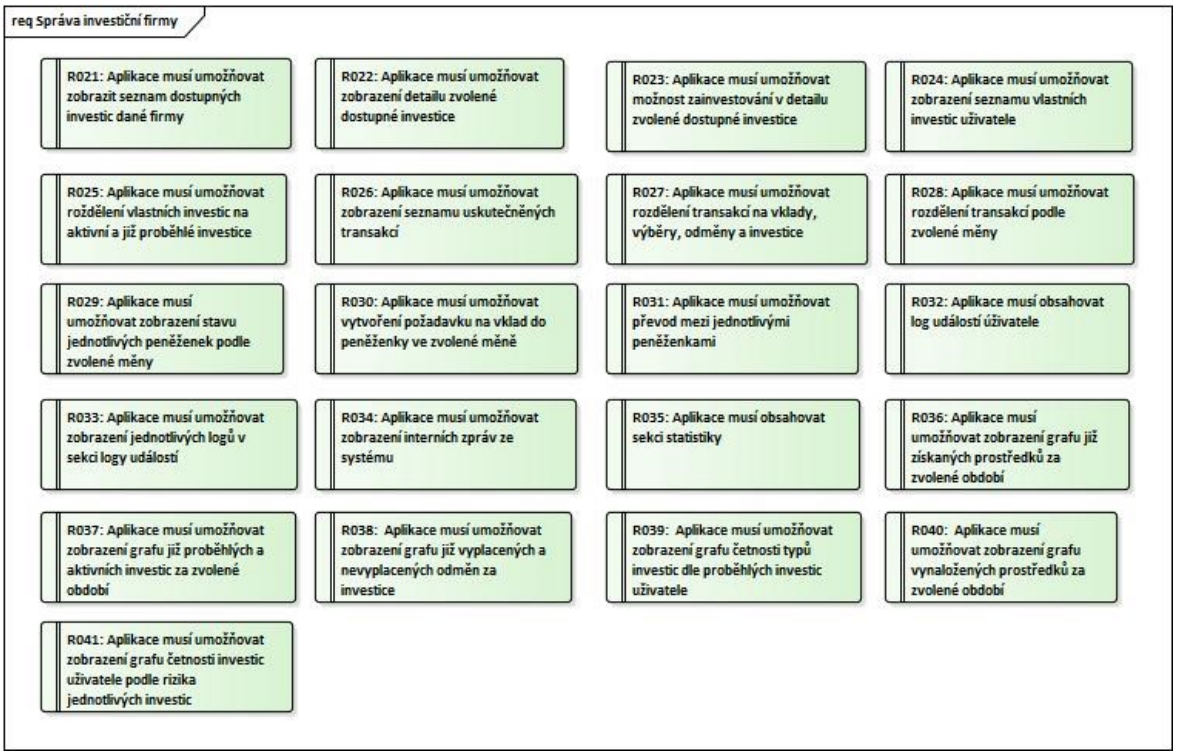
Další kategorií je správa uživatelského profilu, která z velké části obsahuje požadavky na změnu jednotlivých dat uživatele v uživatelském profilu nebo údaje potřebné pro přihlášení do aplikace. Jedním z požadavků je možnost změny přihlašovacích údajů, kde by mělo být uživateli umožněno změnit login a heslo do aplikace. Uživatel by měl být schopný změnit si své telefonní číslo, mail, oslovení nebo také akademické tituly. Rovněž se zde nachází požadavek na změnu adresy trvalého bydliště, popřípadě změna korespondenční adresy. U obou adres se jedná o změnu ulice, čísla popisného, města, poštovního směrovacího čísla a státu. Jako poslední je zde změna přiložených, osobních dokladů, která rovněž zahrnuje možnost nahrání nové fotky daného dokladu. Jednotlivé požadavky v této kategorii také reprezentují počet obrazovek potřebných pro sekci profilu uživatele.



Obrázek 26: Funkční požadavky – správa uživatelského profilu

Poslední kategorií funkčních požadavků je správa investiční firmy. Tato kategorie reprezentuje obsáhlou skupinu požadavků, které se zde vztahují k tomuto konkrétnímu typu firmy. Jedním z nejdůležitějších požadavků v této kategorii je umožnění zobrazení dostupných investic pro danou firmu, včetně zobrazení detailu zvolené investice, ve kterém může uživatel provést zainvestování. Po zainvestování je nutné uživateli přehledně zobrazit jeho

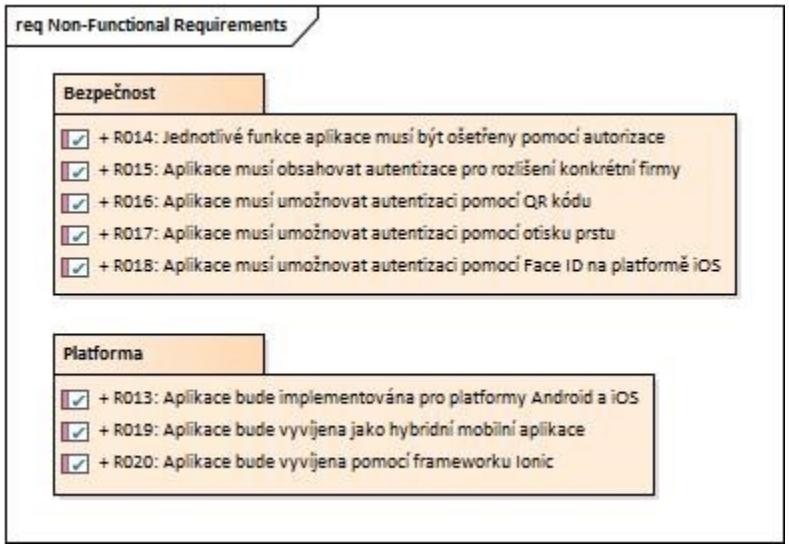
kompletní seznam investic, který bude možné filtrovat podle toho, zda se jedná o aktivní investici nebo již proběhlou. Aby bylo možné zainvestovat, uživatel musí mít potřebné finance v peněžence v rámci aplikace. Peněženky je možné mít ve dvou měnách, a to v eurech a českých korunách. Stav těchto peněženek musí být uživateli přehledně zobrazen a aplikace musí umět zobrazit informace potřebné k vytvoření požadavku na vklad do jedné z peněženek. Mezi těmito peněženkami bude možné finance převádět. Aplikace musí dle požadavků zahrnovat sekce obsahující informace o proběhlých transakcích. Do těchto informací patří například transakce při zainvestování, vyplacení odměn, vklad nebo výběr z peněženky nebo také již zmíněné transakce mezi peněženkami. Pro dlouhodobější sledování svého investičního účtu musí aplikace obsahovat sekci statistiky. V této sekci bude možné pomocí grafů sledovat získané a vynaložené prostředky za zvolené období, rozložení investičního portfolia, podíl nevyplacených a již vyplacených odměn, počet aktivních a proběhlých investic ve zvoleném období a graf četnosti investic podle míry rizika investice. Posledním požadavkem je sekce logů událostí uživatele, kde si může uživatel zobrazit logy o všech změnách, které nastaly v rámci jeho účtu v aplikaci.



Obrázek 27: Funkční požadavky – správa investiční firmy

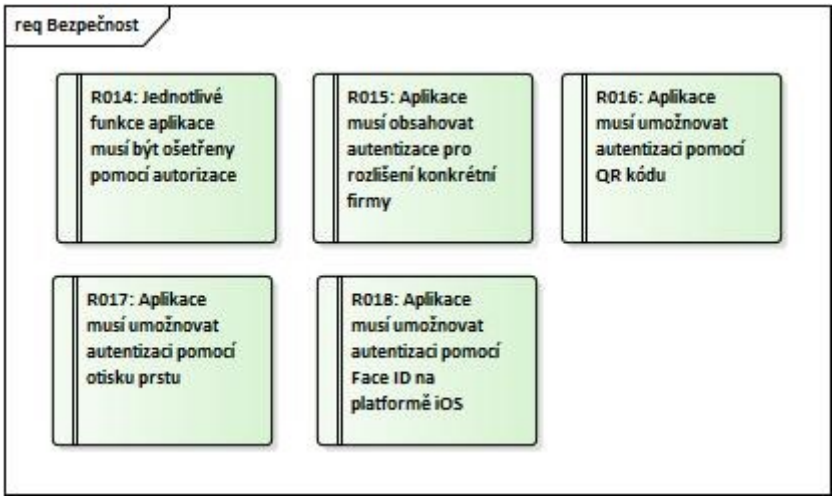
4.1.2 Nefunkční požadavky

Nefunkční požadavky jsou členěny do dvou kategorií. První z kategorií je bezpečnost, druhou pak požadavky na platformu.



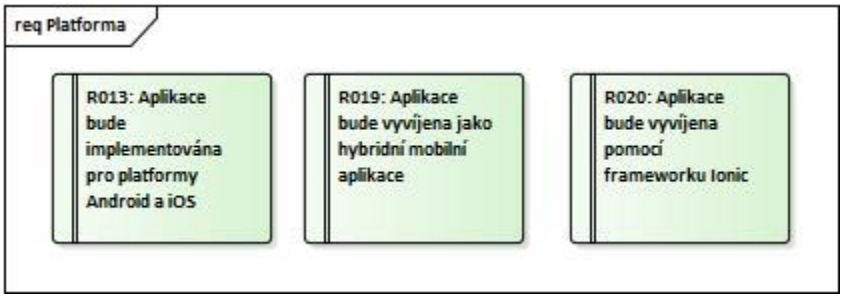
Obrázek 28: Nefunkční požadavky aplikace

Bezpečnost se primárně zaměřuje na autorizaci a autentizaci. Jednotlivé funkce a přístup k obrazovkám aplikace musí být ošetřen pomocí autorizace. Prvotní přístup do aplikace musí být chráněn prostřednictvím autentizace. V prvním kroku bude nutné přidání konkrétní firmy, kdy uživatel z webové aplikace naskenuje QR kód nebo zadá vygenerovaný kód a následně bude daná firma přidána do jeho seznamu firem. Ke způsobu přihlášení k této firmě se vztahuje zbytek požadavků v této kategorii. Aplikace musí umožňovat přihlášení pomocí otisku prstu, pokud to zařízení uživatele umožňuje. U platformy iOS musí být aplikace schopná provést autentizaci pomocí Touch ID nebo Face ID.



Obrázek 29: Nefunkční požadavky – bezpečnost

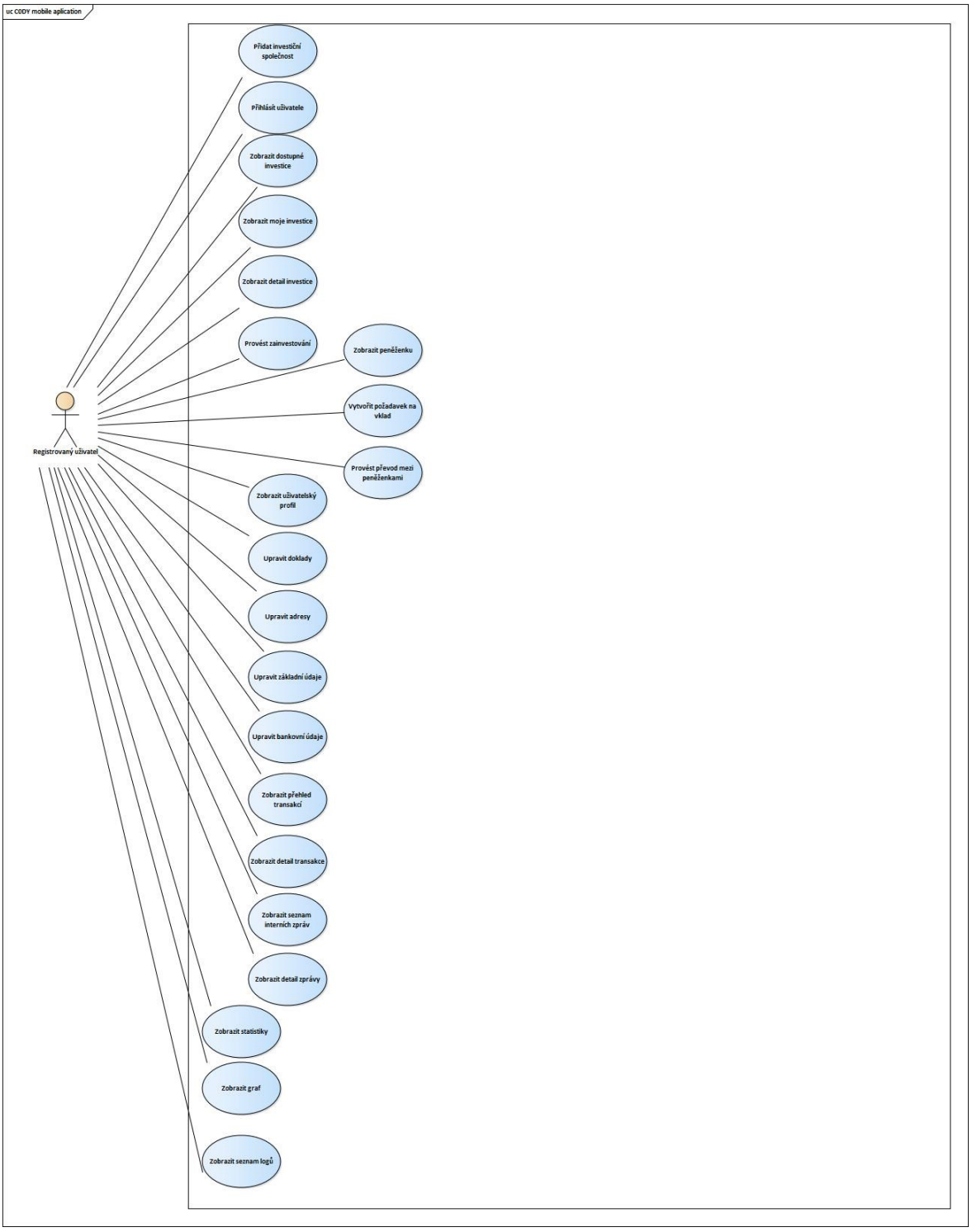
Požadavky v kategorii platforma definují cílené platformy, pro které musí být výsledná aplikace implementována. V případě této aplikace se jedná o platformu Android a iOS. Z dalších požadavků vyplývá zvolený přístup vývoje mobilní aplikace společně se zvoleným frameworkem. Pro vývoj byl zvolen přístup hybridní mobilní aplikace za použití vývojového frameworku Ionic.



Obrázek 30: Nefunkční požadavky – platforma

4.1.3 Use Case Model

Use case model popisuje základní funkcionalitu aplikace a operace, které jsou pro uživatele dostupné. V celé aplikaci figuruje jeden aktér, a to registrovaný uživatel. Druhá polovina této podkapitoly obsahuje scénáře k jednotlivým Use Casům, v některých případech včetně alternativních scénářů.



Obrázek 31: Use Case Model aplikace

..

Tabulka 1: Scénář – Přidat investiční společnost

Jméno scénáře	Přidat investiční společnost
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když chce registrovaný uživatel přidat investiční společnost do aplikace. 2. Uživatel klikne na tlačítko "+". 3. Uživatel zvolí preferovanou metodu přidání. 4. Systém zobrazí preferovanou metodu přidání. 5. Systém zkontroluje správnost získaných údajů. 6. Systém uloží společnost do uložiště zařízení. 7. Systém zobrazí společnost v listu společností
Vedlejší scénáře	4a. Uživatel zvolí přidání pomocí číselného kódu. 4b. Uživatel zvolí přidání pomocí QR kódu. 5a. Zadané údaje nesouhlasí s žádnou existující společností. 5b. Zadané údaje se shodují s již přidanou firmou.

Tabulka 2: Alternativní scénář – Uživatel zvolí přidání pomocí číselného kódu

Jméno scénáře	Přidat investiční společnost – Uživatel zvolí přidání pomocí číselného kódu
Alternativní scénář	1. Systém zobrazí vyskakovací okno se vstupem pro kód. 2. Uživatel zadá kód a stiskne tlačítko „Potvrdit“.

Tabulka 3: Alternativní scénář – Uživatel zvolí přidání pomocí QR kódu

Jméno scénáře	Přidat investiční společnost – Uživatel zvolí přidání pomocí QR kódu
Alternativní scénář	1. Systém spustí čtečku QR kódu. 2. Uživatel oskenuje QR kód.

Tabulka 4: Alternativní scénář – Zadané údaje nesouhlasí

Jméno scénáře	Přidat investiční společnost – Zadané údaje nesouhlasí s žádnou existující společností
Alternativní scénář	1. Systém vyhodí chybovou hlášku a informuje uživatele o chybném zadání údajů.

Tabulka 5: Alternativní scénář – Zadané údaje se shodují s již přidanou firmou

Jméno scénáře	Přidat investiční společnost – Zadané údaje nesouhlasí s žádnou existující společností.
Alternativní scénář	1. Systém vyhodí chybovou hlášku a informuje uživatele o pokusu duplicitního přidání.

Tabulka 6: Scénář – Přihlásit uživatele

Jméno scénáře	Přihlásit uživatele
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když se chce registrovaný uživatel přihlásit do aplikace. 2. Uživatel zvolí společnost pro přihlášení. 3. Systém zobrazí přihlašovací formulář. 4. Uživatel zadá přihlašovací údaje a stiskne tlačítko „Přihlásit se“. 5. Systém zkontroluje zadané údaje. 6. Uživatel je přihlášen.
Vedlejší scénáře	5a. Zadané údaje nejsou validní.

Tabulka 7: Alternativní scénář – Zadané údaje nejsou validní

Jméno scénáře	Přihlásit uživatele – Zadané údaje nejsou validní
Alternativní scénář	1. Systém vyhodí chybovou hlášku a informuje uživatele o chybném zadání přihlašovacích údajů.

Tabulka 8: Scénář – Zobrazit dostupné investice

Jméno scénáře	Zobrazit dostupné investice
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když chce uživatel zobrazit seznam dostupných investic. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko „Tržiště“. 4. Systém zobrazí seznam dostupných investic.
Vedlejší scénáře	4a. Není dostupná žádná investice.

Tabulka 9: Alternativní scénář – Není dostupná žádná investice

Jméno scénáře	Zobrazit dostupné investice – Není dostupná žádná investice
Alternativní scénář	1. Systém vyhodí varovnou hlášku a informuje uživatele, že momentálně nejsou dostupné žádné investiční nabídky.

Tabulka 10: Scénář – Zobrazit moje investice

Jméno scénáře	Zobrazit moje investice
Zúčastnění aktéři	Registrovaný uživatel

Hlavní scénář	1. Use Case začíná, když chce uživatel zobrazit seznam svých investic. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko „Investice“. 4. Systém zobrazí seznam investic uživatele.
Vedlejší scénáře	4a. Není dostupná žádná investice uživatele

Tabulka 11: Alternativní scénář – Není dostupná žádná investice uživatele

Jméno scénáře	Zobrazit moje investice – Není dostupná žádná investice uživatele
Alternativní scénář	1. Systém vyhodí varovnou hlášku a informuje uživatele, že doposud neprovedl žádnou investici.

Tabulka 12: Scénář – Zobrazit detail investice

Jméno scénáře	Zobrazit detail investice
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když chce uživatel zobrazit detail zvolené investice. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko „Investice“. 4. Systém zobrazí seznam investic. 5. Uživatel klikne na zvolenou investici. 6. Systém zobrazí detail zvolené investice.
Vedlejší scénáře	4a. Není dostupná žádná investice uživatele

Tabulka 13: Alternativní scénář – Není dostupná žádná investice uživatele

Jméno scénáře	Zobrazit detail investice – Není dostupná žádná investice uživatele
Alternativní scénář	1. Systém vyhodí varovnou hlášku a informuje uživatele, že doposud neprovedl žádnou investici.

Tabulka 14: Scénář – Provést zainvestování

Jméno scénáře	Provést zainvestování
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když chce uživatel uskutečnit investici. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko „Tržiště“. 4. Systém zobrazí seznam dostupných investic. 5. Uživatel klikne na zvolenou investici. 6. Systém zobrazí detail zvolené investice. 7. Uživatel zvolí částku a klikne na tlačítko „Potvrdit investici“. 8. Systém zkontroluje dostupnost zainvestování.

	9. Systém potvrdí úspěšné zainvestování.
Vedlejší scénáře	4a. Není dostupná žádná investice. 8a. Investice už není dostupná.

Tabulka 15: Alternativní scénář – Není dostupná žádná investice

Jméno scénáře	Provést zainvestování – Není dostupná žádná investice
Alternativní scénář	1. Systém vyhodí varovnou hlášku a informuje uživatele, že momentálně nejsou dostupné žádné investiční nabídky.

Tabulka 16: Alternativní scénář – Investice už není dostupná

Jméno scénáře	Provést zainvestování – Investice už není dostupná
Alternativní scénář	1. Systém vyhodí varovnou hlášku a informuje uživatele, že daná investice už není dále dostupná.

Tabulka 17: Scénář – Zobrazit peněženku

Jméno scénáře	Zobrazit peněženku
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když chce uživatel zobrazit peněženku v aplikaci. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko „Peněženka“. 4. Systém zobrazí seznam peněženek uživatele. 5. Uživatel klikne na zvolenou peněženku. 6. Systém zobrazí detail peněženky.
Vedlejší scénáře	-

Tabulka 18: Scénář – Vytvořit požadavek na vklad

Jméno scénáře	Vytvořit požadavek na vklad
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když chce uživatel provést vklad do peněženky. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko „Peněženka“. 4. Systém zobrazí seznam peněženek uživatele. 5. Uživatel klikne na zvolenou peněženku. 6. Systém zobrazí detail peněženky. 7. Uživatel klikne na tlačítko „Dobít peněženku“. 8. Systém zobrazí formulář pro vytvoření vkladu do peněženky. 9. Uživatel vyplní částku a stiskne tlačítko „Vygenerovat QR kód“. 10. Systém zobrazí QR kód pro platbu.

Vedlejší scénáře	-

Tabulka 19: Scénář – Provést převod mezi peněženkami

Jméno scénáře	Provést převod mezi peněženkami
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když chce uživatel převést prostředky mezi peněženkami. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko „Peněženka“. 4. Systém zobrazí seznam peněženek uživatele. 5. Uživatel klikne na zvolenou peněženku. 6. Systém zobrazí detail peněženky. 7. Uživatel klikne na tlačítko „Převod“. 8. Systém zobrazí formulář pro převod mezi peněženkami. 9. Uživatel vyplní částku a stiskne tlačítko „Potvrdit“. 10. Systém ověří požadavek na převod. 11. Systém potvrdí převod mezi peněženkami.
Vedlejší scénáře	10a. Nedostatek prostředků ve zvolené peněžence.

Tabulka 20: Alternativní scénář – Nedostatek prostředků ve zvolené peněžence

Jméno scénáře	Provést převod mezi peněženkami – Nedostatek prostředků ve zvolené peněžence
Alternativní scénář	1. Systém vyhodí chybovou hlášku a informuje uživatele o nedostatku prostředků ve zvolené peněžence.

Tabulka 21: Scénář – Zobrazit uživatelský profil

Jméno scénáře	Zobrazit uživatelský profil
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když chce uživatel zobrazit svůj profil v aplikaci. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko s ikonou osoby. 4. Systém zobrazí profil uživatele.
Vedlejší scénáře	-

Tabulka 22: Scénář – Upravit doklady

Jméno scénáře	Upravit doklady
Zúčastnění aktéři	Registrovaný uživatel

Hlavní scénář	1. Use Case začíná, když chce uživatel upravit své doklady v aplikaci. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko s ikonou osoby. 4. Systém zobrazí profil uživatele. 5. Uživatel klikne na tlačítko „Doklady totožnosti“. 6. Systém zobrazí seznam dostupných dokladů. 7. Uživatel klikne na zvolený doklad. 8. Systém zobrazí detail dokladu. 9. Uživatel provede změny a klikne na tlačítko „Uložit změny“. 10. Systém zkontroluje provedené změny. 11. Systém uloží provedené změny dokladu a informuje uživatele o úspěšném uložení.
Vedlejší scénáře	10a. Validace provedených změn selhala.

Tabulka 23: Alternativní scénář – Validace provedených změn selhala

Jméno scénáře	Upravit doklady – Validace provedených změn selhala
Alternativní scénář	1. Systém vyhodí chybovou hlášku a informuje uživatele o špatně provedených změnách, které nesplňují potřebná kritéria.

Tabulka 24: Scénář – Upravit adresy

Jméno scénáře	Upravit adresy
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když chce uživatel upravit své adresy v aplikaci. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko s ikonou osoby. 4. Systém zobrazí profil uživatele. 5. Uživatel klikne na tlačítko „Moje adresy“. 6. Systém zobrazí adresy uživatele. 7. Uživatel provede změny a klikne na tlačítko „Uložit změny“. 8. Systém uloží provedené změny adres a informuje uživatele o úspěšném uložení.
Vedlejší scénáře	-

Tabulka 25: Upravit základní údaje

Jméno scénáře	Upravit základní údaje
Zúčastnění aktéři	Registrovaný uživatel

Hlavní scénář	1. Use Case začíná, když chce uživatel upravit své základní údaje v aplikaci. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko s ikonou osoby. 4. Systém zobrazí profil uživatele. 5. Uživatel provede změny a klikne na tlačítko „Uložit změny“. 6. Systém uloží provedené změny a informuje uživatele o úspěšném uložení.
Vedlejší scénáře	-

Tabulka 26: Upravit bankovní údaje

Jméno scénáře	Upravit bankovní údaje
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když chce uživatel upravit své bankovní údaje v aplikaci. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko s ikonou osoby. 4. Systém zobrazí profil uživatele. 5. Uživatel klikne na tlačítko „Bankovní údaje“. 6. Systém zobrazí detaily bankovních účtů. 7. Uživatel provede změny a klikne na tlačítko „Uložit změny“. 8. Systém zkontroluje provedené změny. 9. Systém uloží provedené změny bankovních údajů a informuje uživatele o úspěšném uložení.
Vedlejší scénáře	9a. Validace provedených změn selhala.

Tabulka 27: Alternativní scénář – Validace provedených změn selhala

Jméno scénáře	Upravit bankovní údaje – Validace provedených změn selhala
Alternativní scénář	1. Systém vyhodí chybovou hlášku a informuje uživatele o špatně provedených změnách, které nesplňují potřebná kritéria.

Tabulka 28: Scénář – Zobrazit přehled transakcí

Jméno scénáře	Zobrazit přehled transakcí
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když chce uživatel zobrazit přehled svých transakcí v aplikaci. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko „Více“. 4. Systém zobrazí navigační menu. 5. Uživatel zvolí položku „Transakce“. 6. Systém zobrazí tabulku s transakcemi uživatele.

Vedlejší scénáře	-

Tabulka 29: Scénář – Zobrazit detail transakce

Jméno scénáře	Zobrazit detail transakce
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když chce uživatel zobrazit detail zvolené transakce. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko „Více“. 4. Systém zobrazí navigační menu. 5. Uživatel zvolí položku „Transakce“. 6. Systém zobrazí tabulku s transakcemi uživatele. 7. Uživatel klikne na zvolenou transakci. 8. Systém zobrazí detail zvolené transakce.
Vedlejší scénáře	5a. Není dostupná žádná transakce uživatele.

Tabulka 30: Scénář – Zobrazit seznam interních zpráv

Jméno scénáře	Zobrazit seznam interních zpráv
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když chce uživatel zobrazit seznam interních zpráv. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko s ikonou obálky. 4. Systém zobrazí seznam interních zpráv uživatele.
Vedlejší scénáře	-

Tabulka 31: Scénář – Zobrazit detail interní zprávy

Jméno scénáře	Zobrazit detail interní zprávy
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když chce uživatel zobrazit detail interní zprávy. 2. Uživatel klikne na tlačítko s ikonou obálky. 3. Systém zobrazí seznam interních zpráv uživatele. 4. Uživatel klikne na zvolenou interní zprávu. 5. Systém zobrazí detail interní zprávy.
Vedlejší scénáře	4a. Není dostupná žádná interní zpráva.

Tabulka 32: Scénář – Zobrazit statistiky

Jméno scénáře	Zobrazit statistiky
Zúčastnění aktéři	Registrovaný uživatel

Hlavní scénář	1. Use Case začíná, když chce uživatel zobrazit sekci statistiky. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko „Více“. 4. Systém zobrazí navigační menu. 5. Uživatel zvolí položku „Statistiky“. 6. Systém zobrazí sekci statistiky.
Vedlejší scénáře	-

Tabulka 33: Scénář – Zobrazit graf

Jméno scénáře	Zobrazit graf
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když chce uživatel zobrazit graf v sekci statistiky. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko „Více“. 4. Systém zobrazí navigační menu. 5. Uživatel zvolí položku „Statistiky“. 6. Systém zobrazí sekci statistiky. 7. Uživatel klikne na dropdown "Typ grafu". 8. Systém zobrazí možné typy grafů. 9. Uživatel vybere typ grafu. 10. Systém zobrazí požadovaný graf.
Vedlejší scénáře	7.a Uživatel zvolí graf získaných prostředků za zvolené období. 7.b Uživatel zvolí graf proběhlých a aktivních investic za zvolené období. 7.c Uživatel zvolí graf vyplacených a nevyplacených odměn za investice. 7.d Uživatel zvolí graf četnosti typů investic dle proběhlých investic uživatele. 7.e Uživatel zvolí graf vynaložených prostředků za zvolené období. 7.f Uživatel zvolí graf četnosti investic uživatele podle rizika jednotlivých investic.

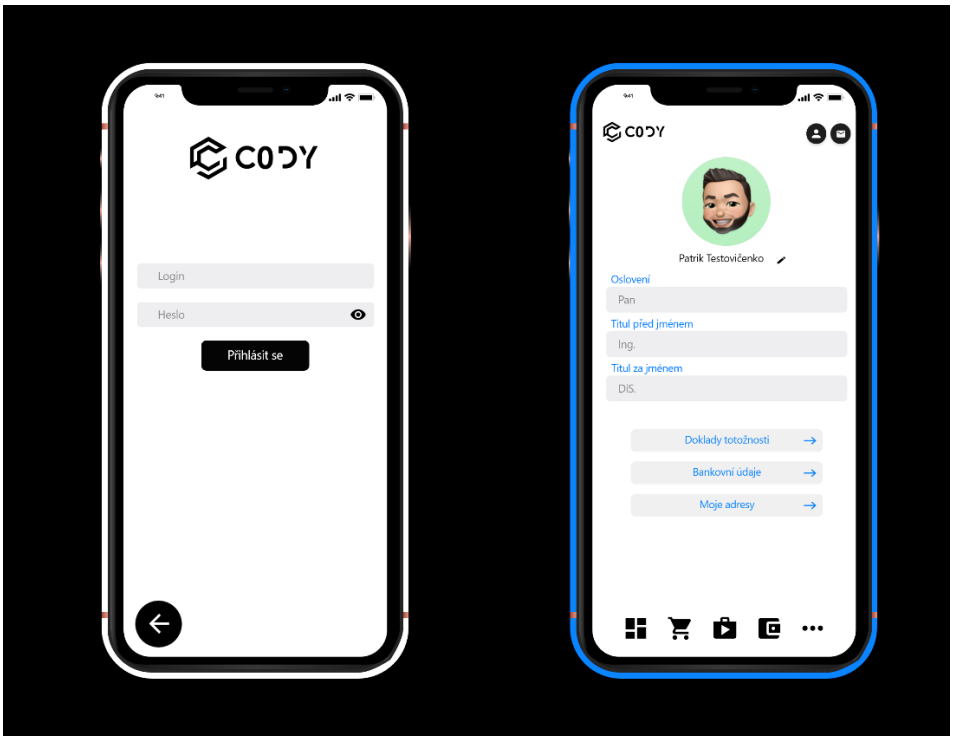
Tabulka 34: Scénář – Zobrazit seznam logů

Jméno scénáře	Zobrazit seznam logů
Zúčastnění aktéři	Registrovaný uživatel
Hlavní scénář	1. Use Case začíná, když chce uživatel zobrazit seznam systémových logů v aplikaci. 2. Uživatel se přihlásí do aplikace. 3. Uživatel klikne na tlačítko „Více“. 4. Systém zobrazí navigační menu. 5. Uživatel zvolí položku „Log událostí“. 6. Systém zobrazí seznam systémových logů uživatele.

Vedlejší scénáře	-

4.2 Návrh uživatelského rozhraní

Pro návrh uživatelského rozhraní byl použit program Adobe XD. Tento program byl zvolen z důvodu obsažení již předdefinovaných šablon pro konkrétní zařízení, ať už se jedná o celou řadu mobilních zařízení, tak web. Další výhodou je také hezké grafické zpracování návrhu, které na rozdíl od běžně používaných wireframe návrhu působí uživatelsky více přívětivě. Při návrhu bylo využito oficiálního Apple Design Resources balíčku pro Adobe XD, který obsahuje všechny prvky uživatelského rozhraní pro aktuální verzi systému iOS 14. Tento balíček značně zrychlil proces samotného návrhu a také poskytnul lepší a reálnější představu o možné podobě aplikace. Funkce Adobe XD kromě návrhu obrazovek umožnily dané obrazovky a jejich elementy propojit, díky čemuž z návrhu vzniklo funkční demo samotné aplikace, které kromě podoby obrazovek poskytlo představu o toku mezi jednotlivými obrazovkami a celkovou funkcionalitou aplikace. Při návrhu bylo celkově vytvořeno 45 obrazovek včetně úvodní obrazovky a obrazovky reprezentující ikonu plochy.



Obrázek 32: Ukázka návrhu z prostředí Adobe XD

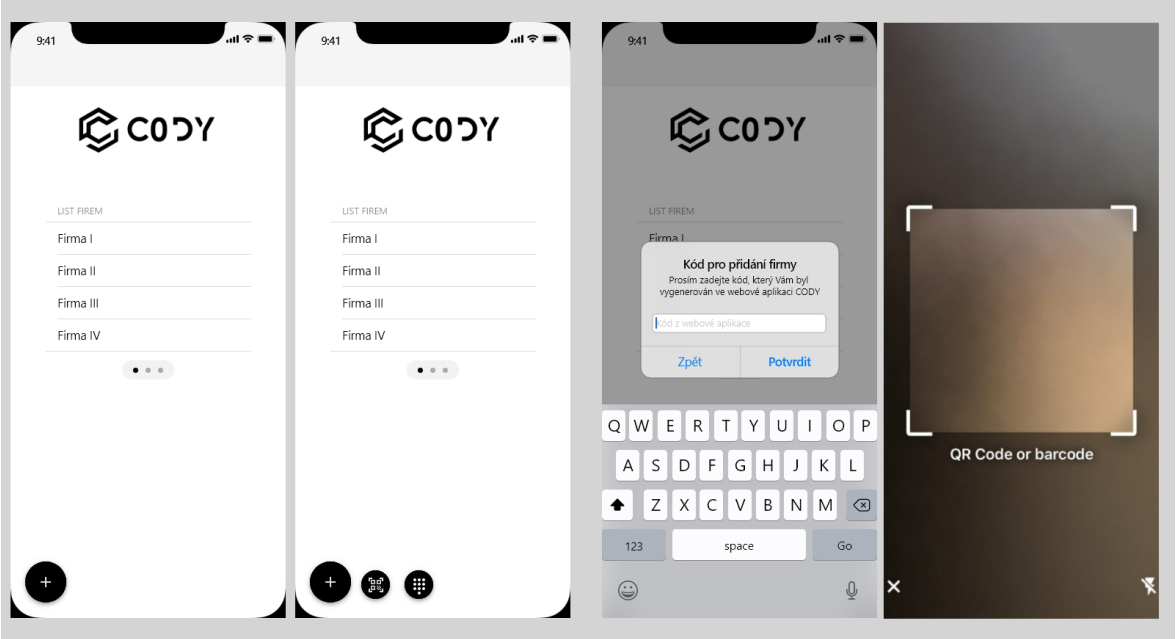
4.2.1 Úvodní obrazovka a ikona aplikace

V první fázi návrhu uživatelského rozhraní byla navržena ikona a úvodní obrazovka aplikace. Vzhled obou těchto prvků je založen na barvách používaných ve webové aplikaci této společnosti, díky čemuž může uživateli navodit pocit známého prostředí.

Obrázek 33: Návrh ikony a úvodní obrazovky aplikace

4.2.2 Seznam společností a přidání nové společnosti

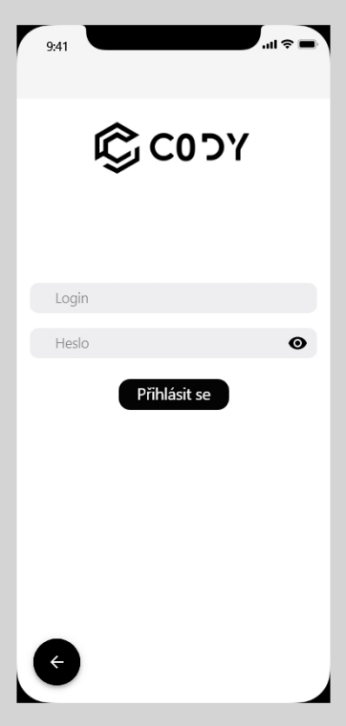
V dalším kroku proběhl návrh obrazovek, které slouží k zobrazení seznamu společností a přidání nové společnosti do tohoto seznamu. Při návrhu byla zohledněna i situace, kdy uživatel přidá větší množství společností do aplikace. Pro tyto účely bylo do návrhu začleněno stránkování, které bude ve finální aplikaci zobrazovat čtyři až pět společností na stránku. Pro přidání společnosti slouží tlačítko v levém spodním rohu. Po jeho stisku se zobrazí nabídka akcí s jednotlivými možnostmi přidání, ze kterých si uživatel zvolí tu, kterou preferuje. Jedná se o přidání pomocí oskenování QR kódu z webové aplikace nebo zadáním vygenerovaného kódu do zobrazeného okna. Oba tyto způsoby lze vidět na navržených obrazovkách na obrázku č. 33.



Obrázek 34: Návrh obrazovky – seznam společností

4.2.3 Přihlašovací obrazovka

Po zvolení společnosti dochází k přesměrování na přihlašovací obrazovku (obrázek č. 34). Jedná se o klasickou přihlašovací obrazovku, obsahující dva vstupy pro login a heslo společně s tlačítkem pro potvrzení zadaných údajů. V pravém rohu vstupu pro heslo se nachází ikona oka, která slouží k zobrazení nebo zpětnému skrytí zadávaného hesla. Pro návrat na obrazovku s přehledem společností slouží tlačítko ve spodním levém rohu.

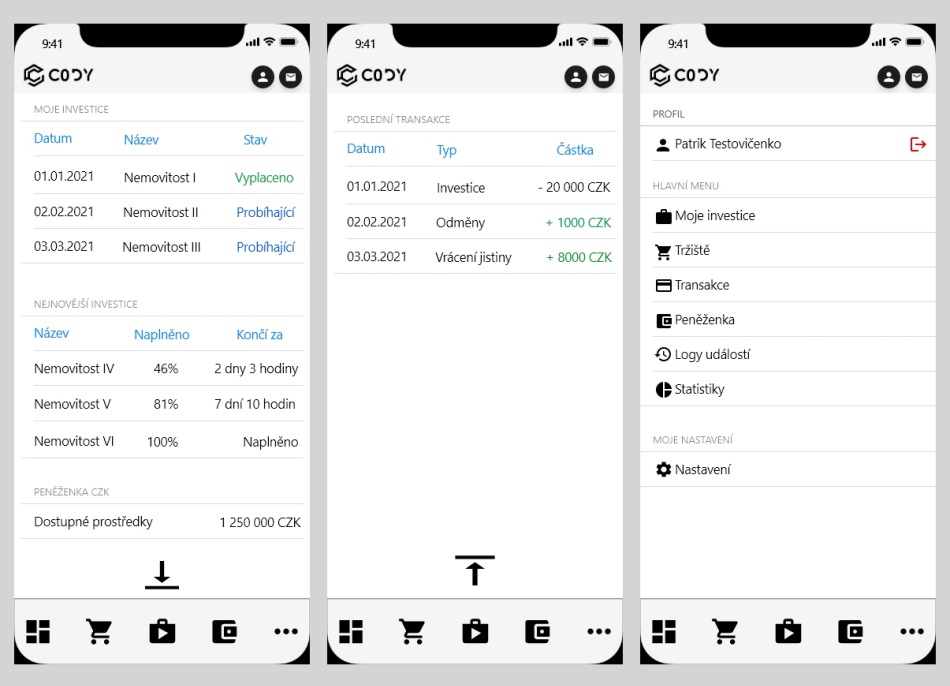


Obrázek 35: Návrh přihlašovací obrazovky

4.2.4 Dashboard a navigační menu

Úspěšné přihlášení umožní následný přístup na obrazovku se stručnou sumarizací důležitých údajů uživatele. V první části se nachází list posledních investic uživatele, ve kterém jsou obsaženy informace, kdy bylo zainvestování provedeno a zda je daná investice vyplacena nebo stále probíhá. Důležitou částí je sekce se seznamem nejnovějších investic, ve které uživatel vidí ty nejaktuálnější investiční nabídky, s mírou jejich naplnění a časem uzavření. Na této obrazovce jsou dále dostupné informace o stavu prostředků v jednotlivých peněženkách a také seznam posledních provedených transakcí.

Pro přechod mezi jednotlivými obrazovkami slouží menu ve spodní části obrazovky, které obsahuje odkazy na čtyři nejdůležitější části aplikace. Jedná se o úvodní dashboard, tržiště s investicemi, sekci „Moje investice“ a přehled peněženek. Pátý prvek v tomto menu slouží k zobrazení obrazovky s navigačním menu pro celou aplikaci. Tato obrazovka rovněž slouží pro odhlášení uživatele z aplikace pomocí červené ikony vedle jména uživatele.



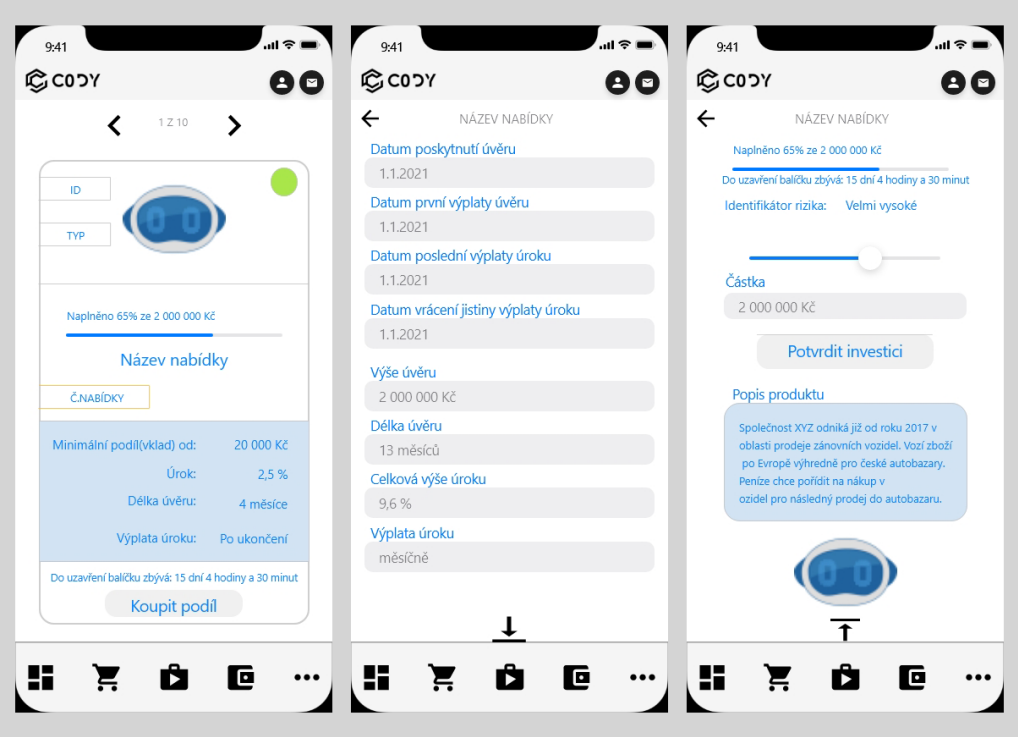
Obrázek 36: Návrh úvodní obrazovky a obrazovky s navigačním menu

4.2.5 Tržiště a detail investice

Jednou z hlavních částí aplikace je tržiště s dostupnými investičními nabídkami. Každá investice je reprezentována kartou neboli dlaždicí, která je umístěná po celé obrazovce (viz obrázek č. 37). Nad dlaždicí jsou umístěny tlačítka pro přepínání mezi všemi dostupnými investicemi, kdy jejich celkový počet je zobrazen mezi těmito tlačítky. V dlaždici jsou obsaženy ty nejdůležitější informace, které může uživatel potřebovat pro prvotní seznámení s investiční nabídkou. V horní části dlaždice se nachází fotografie, id, typ nabídky a barevné kolečko umístěné v pravé části reprezentuje riziko dané investice. Naplněnost investičního balíčku je zobrazena ukazatelem průběhu umístěným ve středu dlaždice společně s číslem nabídky a názvem investice. Údaje v modrém poli obsahují ty nejdůležitější informace, které mohou uživatele zajímat a z tohoto důvodu jsou oproti ostatním informacím podbarveny tak, aby upoutaly pozornost a uživatel se na ně zaměřil. Nachází se zde informace o minimálním vkladu, úroku, délce úvěru a způsobu výplaty úroku. Ve spodní části dlaždice se nachází časový indikátor uzavření dané investiční nabídky a tlačítko, které v případě zájmu uživatele přesměruje na detail investice.

V detailu investice se nachází výčet podrobnějších informací o uživatelem zvolené investici (viz obrázek č. 37). Nejdůležitější je zde část pro potvrzení případného zainvestování, kdy při zadávání částky může uživatel využít posuvníku nebo klasického vepsání

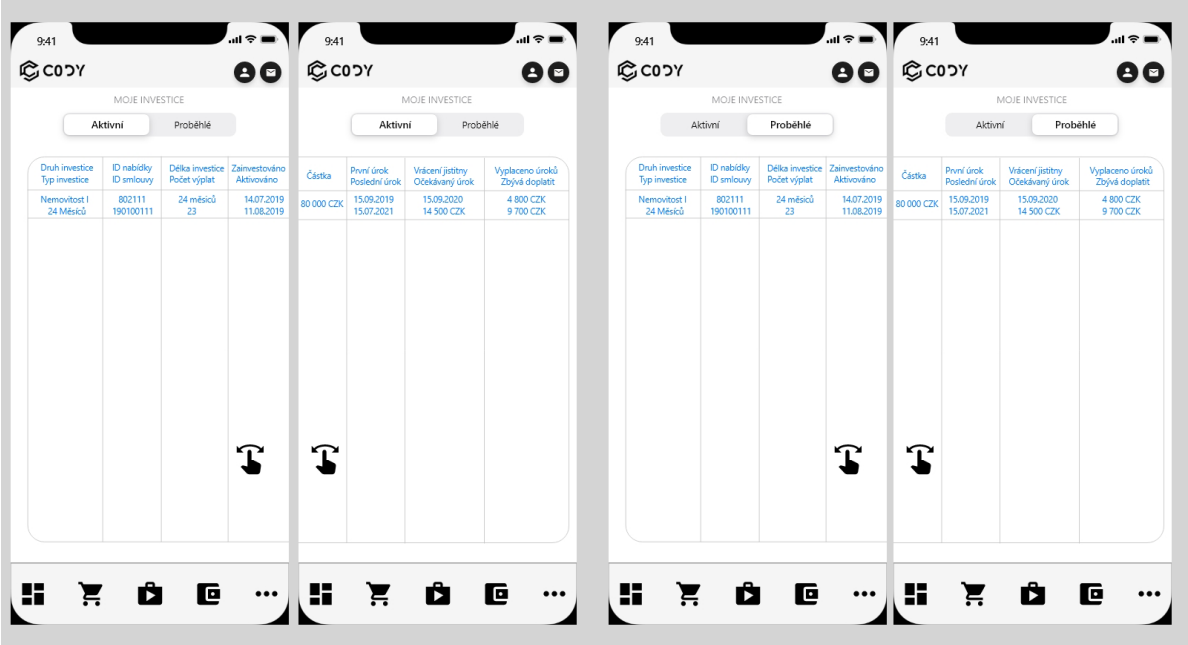
do zobrazeného vstupu. Zadávaná částka musí samozřejmě splňovat požadavek na minimální vklad a hodnota maximální se odvíjí od naplněnosti dané investice. Pod částí pro zainvestování se nachází delší textový popis a případné fotografie k investici.



Obrázek 37: Návrh obrazovky – tržiště a detail investice

4.2.6 Sekce „Moje investice“

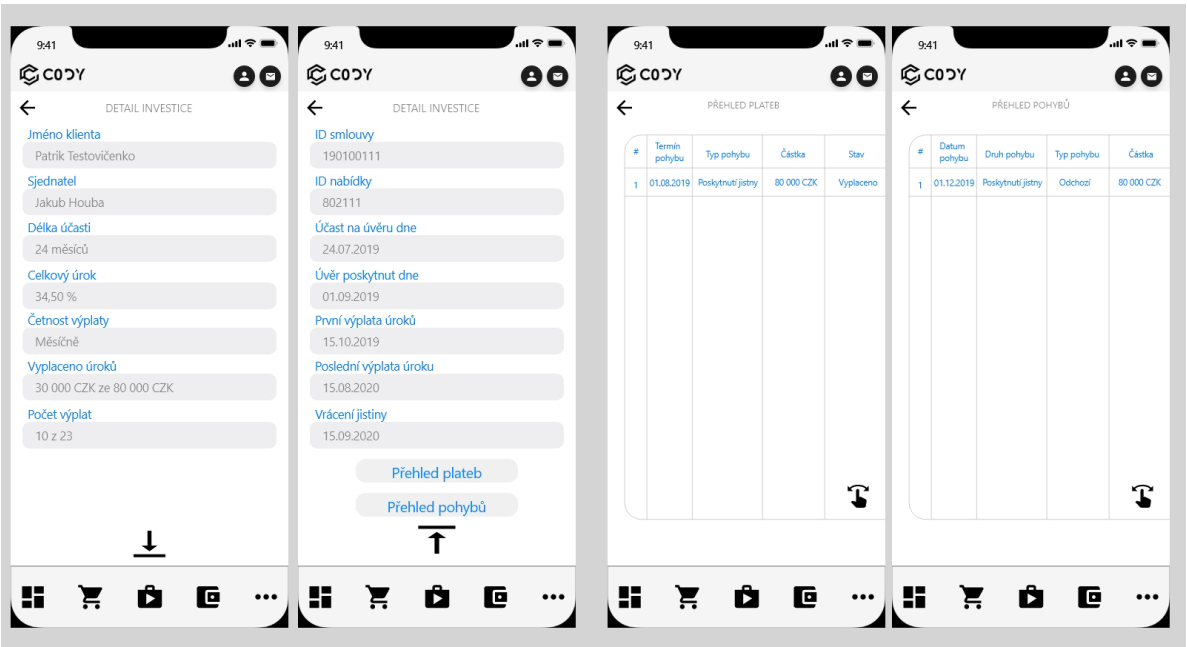
Další důležitou částí je sekce „Moje investice“. Tato sekce slouží k zobrazení investic, do kterých uživatel zainvestoval. Tyto investice se dále dělí na aktivní a proběhlé a jsou zobrazeny v tabulkách v jednotlivých sekcích (viz obrázek č. 38). Pro změnu typu investice slouží přepínací tlačítko umístěné v horní části obrazovky. Zobrazené tabulky jsou prostorově velice objemné a jednotlivé záznamy obsahují velké množství informací. Obecně lze konstatovat, že se jedná o nejobtížnější prvek v aplikaci z důvodu poměrně malé velikosti zobrazovacího prostoru s ohledem na množství informací, které je v něm nutné zobrazit. Po kliknutí na řádek tabulky dojde k přesměrování na detail vybrané investice.



Obrázek 38: Návrh obrazovky – „Moje investice“

4.2.7 Detail investice a přehled pohybů a plateb

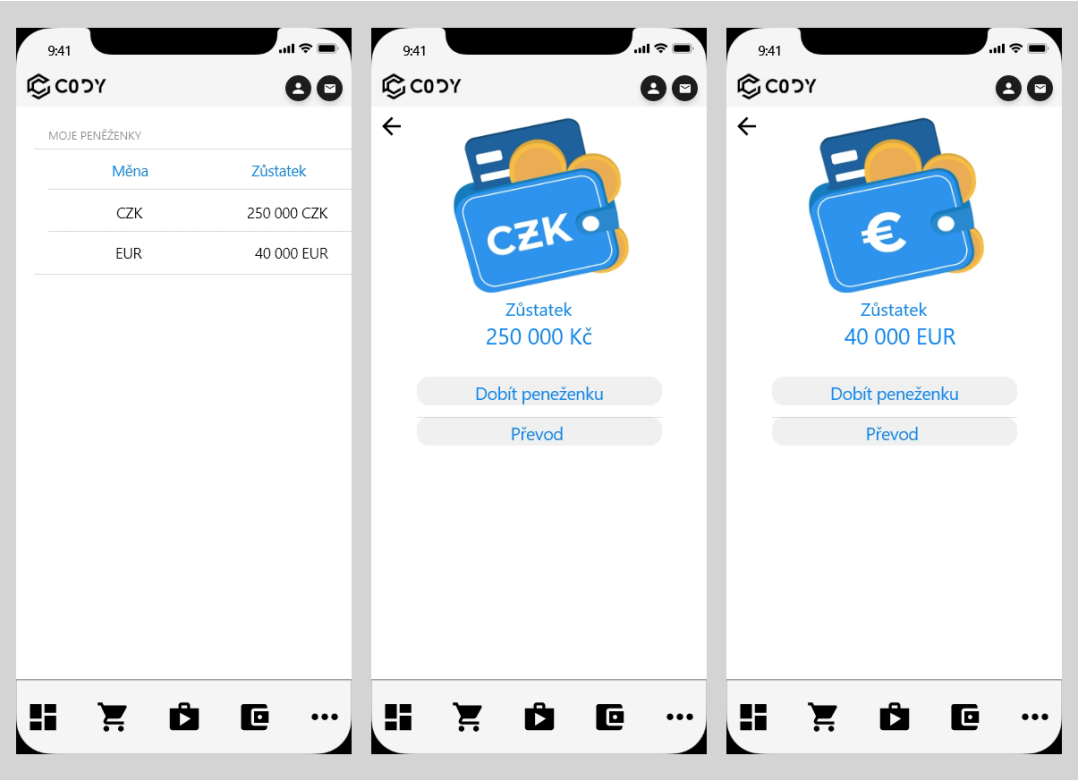
V detailu jsou přehledněji znázorněny informace o zvolené investici a oproti tabulce v sekci „Moje investice“ jsou zde doplňující údaje o sjednateli investice a počtu vyplacených výplat. Ve spodní části se nachází tlačítka, pomocí kterých dojde k přesměrování na obrazovky obsahující tabulky s přehledem plateb a pohybů (viz obrázek č. 39).



Obrázek 39: Návrh obrazovky – detail investice a přehled plateb a pohybů

4.2.8 Sekce „Moje peněženky“ a detail peněženky

Pro možnosti zainvestování je nutné mít příslušné prostředky v interní peněžence aplikace. Pro přehled nad financemi v aplikaci slouží sekce „Peněženky“. V přehledu peněženek je zobrazena měna dané peněženky s dostupným zůstatkem. Po kliknutí na jednu ze zobrazených peněženek dojde k přesměrování na její detail, kde je zobrazen její stav společně s tlačítky, pomocí kterých lze uskutečnit převod do druhé peněženky nebo zobrazit údaje pro její nabití.

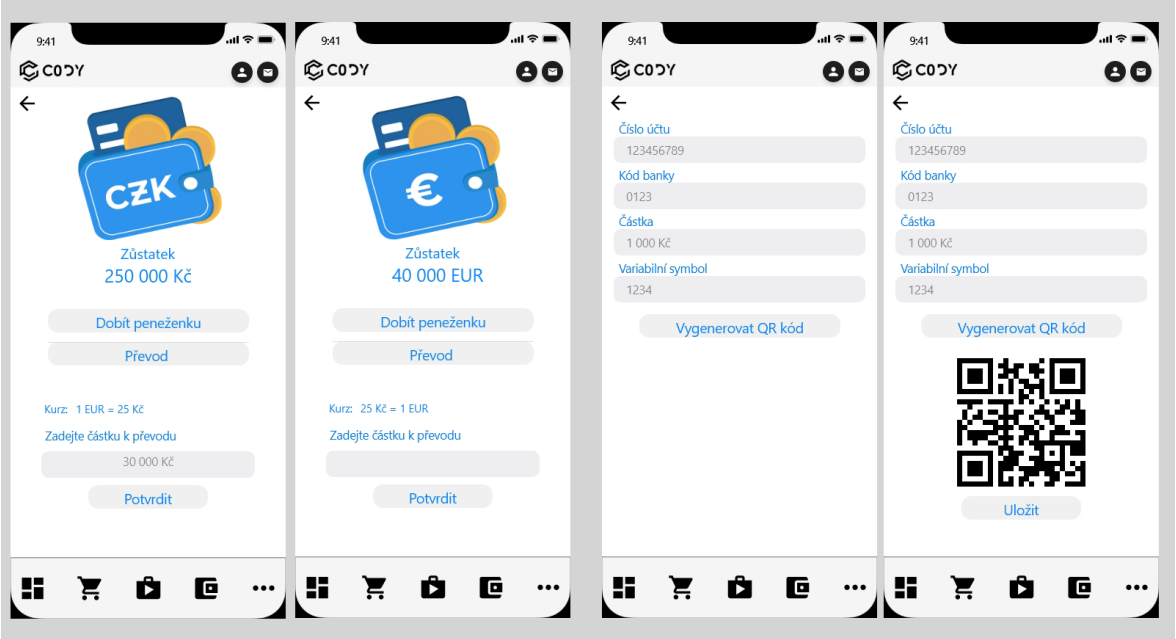


Obrázek 40: Návrh obrazovky – „Moje peněženky“ a detail peněženky

4.2.9 Nabití peněženky a převod mezi peněženkami

U dobití peněženky je naplánována implementace funkcionality pro vygenerování QR kódu, který lze uložit do galerie a použít v internetovém bankovnictví. V této obrazovce uživatel doplňuje pouze částku, kterou chce převést do peněženky. Hodnoty jako je číslo účtu, kód banky a variabilní symbol jsou statické a uživatel je nemůže měnit.

Po stisknutí tlačítka pro převod je zobrazen příslušný vstup pro zadání množství prostředků k převodu a také aktuální kurz, který bude po potvrzení převodu použit.

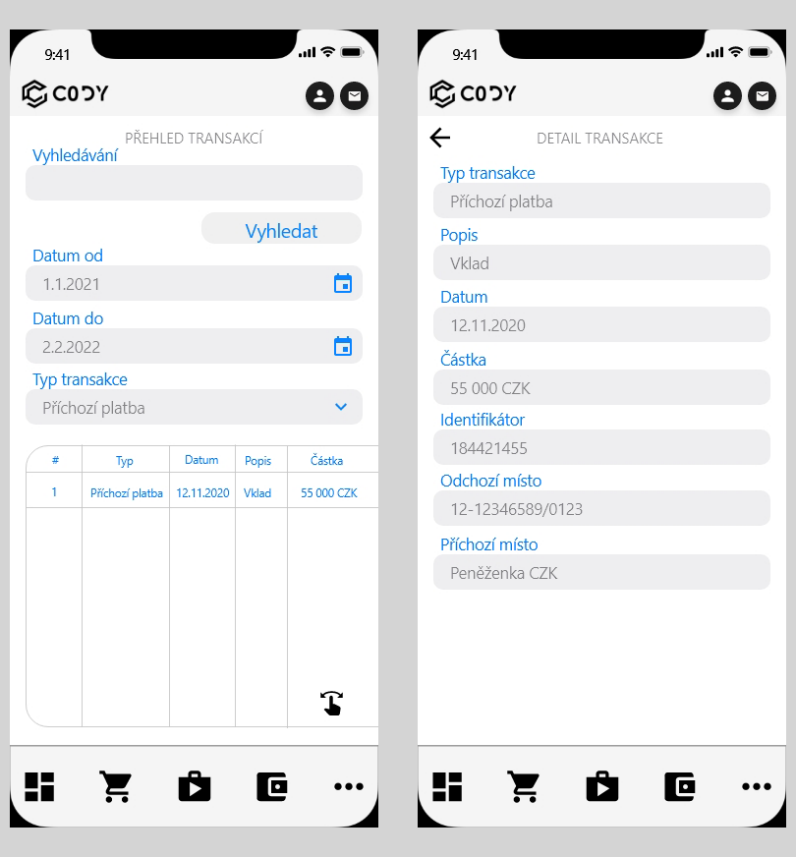


Obrázek 41: Návrh obrazovek pro nabití peněženky a převod mezi peněženkami

4.2.10 Přehled transakcí a detail transakce

Pro sledování toku finančních prostředků v aplikaci slouží sekce transakce. Hlavním prvkem této sekce je tabulka obsahující všechny typy transakcí, jako jsou vklady, výběry, výplaty odměn nebo vrácení jistin. U každé transakce je v tabulce záznam o typu transakce, datu provedení transakce, doplňujícím popisu a hlavně částce. Návrh počítá s možností filtrování této tabulky pomocí vstupu pro vyhledávání, zadáním časových rozsahů nebo specifikováním typu transakce (viz obrázek č. 42).

Detail transakce je možné otevřít z tabulky po kliknutí na vybranou transakci. Kromě údajů obsažených v tabulce jsou zde informace o příchozím a odchozím místu a identifikátoru transakce.

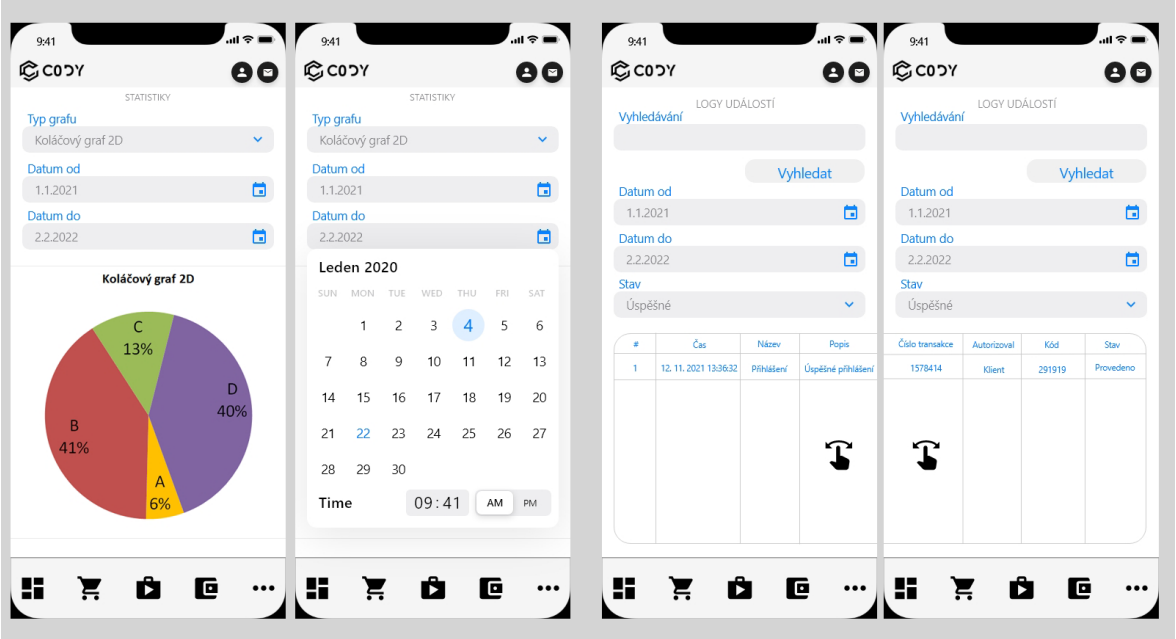


Obrázek 42: Návrh obrazovky – transakce a detail transakce

4.2.11 Statistiky a logy událostí

V případě častého investování je žádoucí přehledné zpracování dlouhodobých statistik v rámci účtu uživatele. Pro tyto účely slouží sekce statistiky, kde jsou v rozbalovacím seznamu zobrazeny všechny dostupné typy grafů, vstupy pro volbu časového rozsahu a finální graf umístěný pod těmito prvky. Může se jednat o graf vynaložených prostředků nebo graf četnosti typů investic v portfoliu uživatele.

Sekce „Logy událostí“ slouží k monitorování veškeré aktivity v rámci uživatelského účtu. Tím jsou myšleny akce uskutečněné uživatelem, jako je například přihlášení do systému, změna osobních údajů nebo převod mezi peněženkami a také akce provedené systémem související s tímto účtem. Každý z logů je definován názvem, podrobnějším popisem, časem události, stavem a označením, kdo danou událost autorizoval. Některé z logů obsahují záznam o přiděleném číslu transakce nebo přiděleném kódu. Z důvodu podrobného sledování aktivity bylo nutné do návrhu zahrnout filtrování tabulky pomocí vstupu pro vyhledávání, zvolení časového rozsahu a specifikování stavu události.

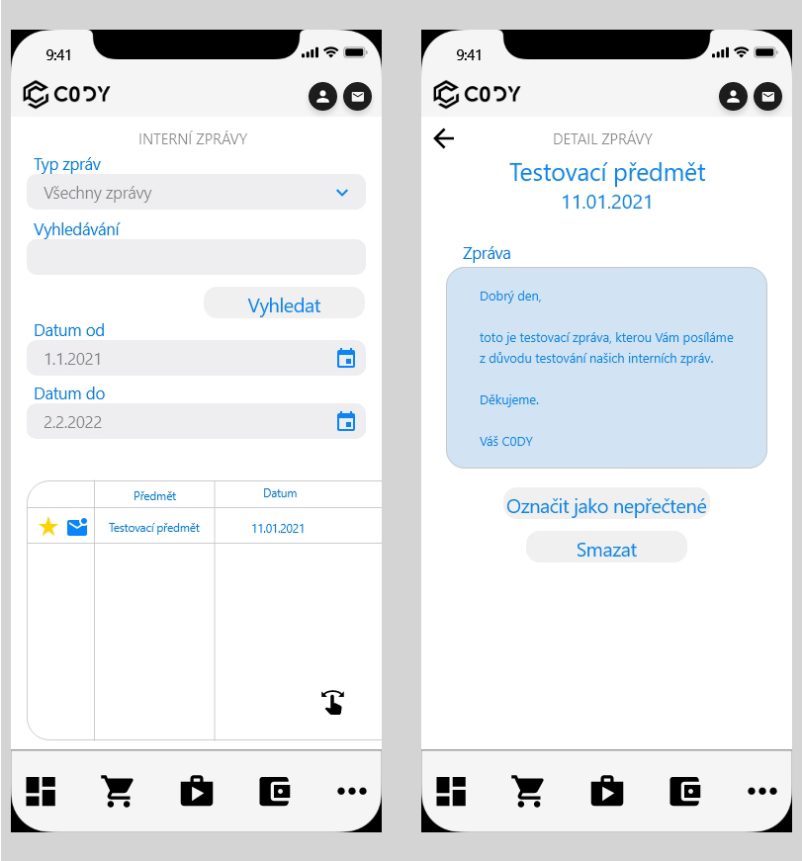


Obrázek 43: Návrh obrazovek – statistiky a logy událostí

4.2.12 Interní zprávy a detail zprávy

Pro informování uživatelů o důležitých událostech nebo změnách v systému slouží interní zprávy. Tyto zprávy jsou dostupné v tabulce ve stejnojmenné sekci, kde je u každé zprávy uveden její předmět a datum, kdy tato zpráva dorazila. To, zda byla daná zpráva otevřena či nikoliv, indikuje druhá ikona v prvním sloupci. První sloupec tabulky rovněž obsahuje ikonu hvězdy, která slouží k označení vybrané zprávy jako důležitou a takto označené zprávy lze zobrazit pomocí zvolení příslušného typu zprávy v rozbalovacím menu v úvodu obrazovky. Kromě specifikování typu zprávy lze nastavit časový rozsah doručení zpráv nebo vyhledávat konkrétní předmět zprávy.

Detail zprávy obsahuje samotný text zprávy společně s dvěma tlačítky sloužícími ke smazání a označení zprávy za nepřečtenou.



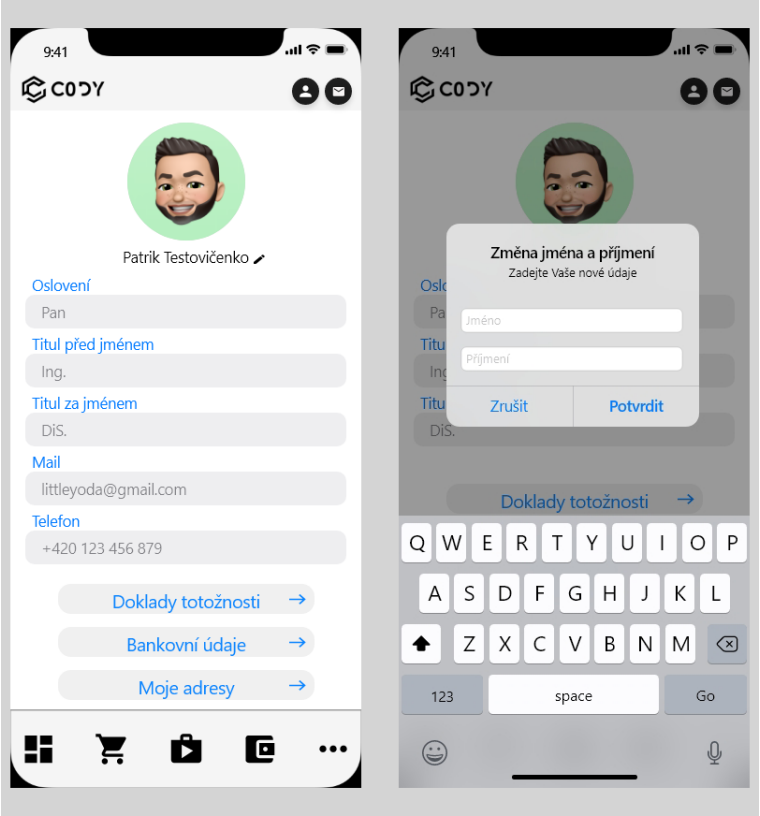
Obrázek 44: Návrh obrazovky – interní zprávy a detail zprávy

4.2.13 Uživatelský profil

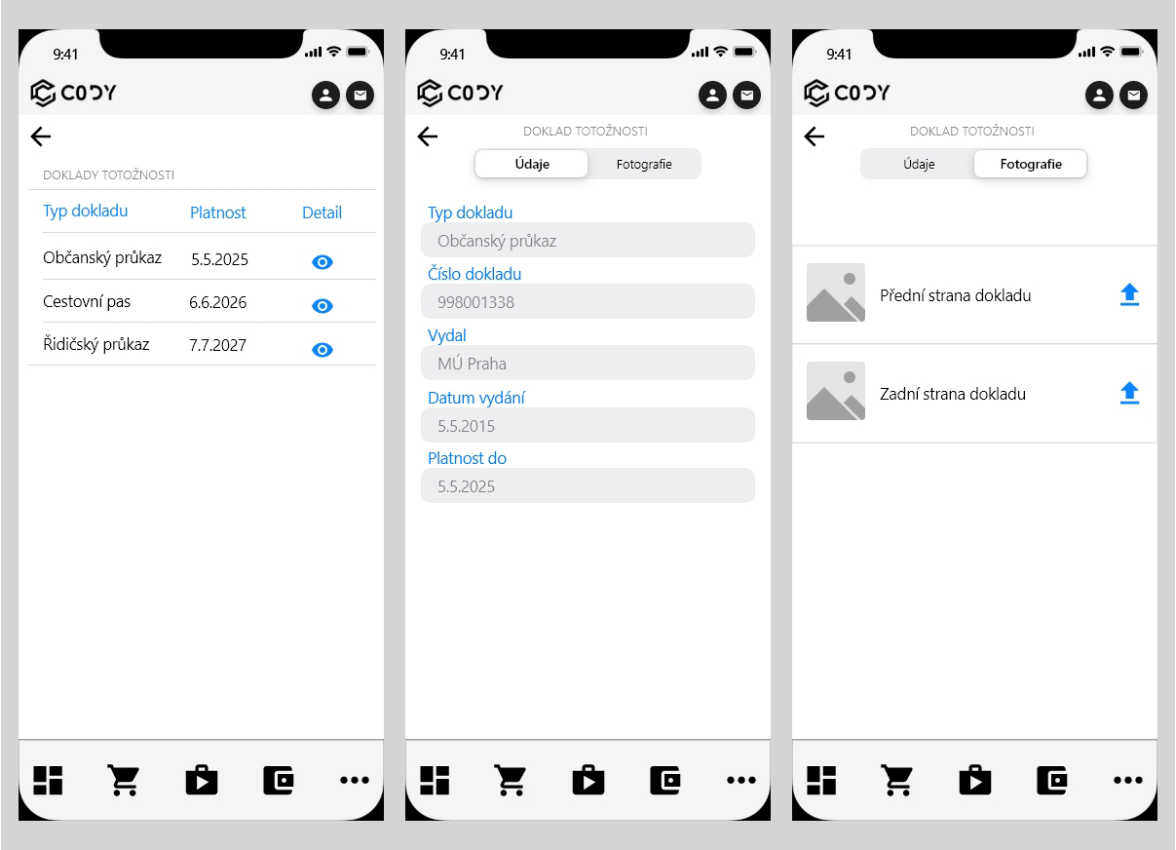
Uživatelský profil je prostor, který obsahuje veškeré osobní údaje uživatele v aplikaci. Kromě jejich zobrazení slouží tato sekce také k jejich úpravě. Při přístupu na uživatelský profil je nejprve zobrazena obrazovka se základními údaji jako je jméno, příjmení, oslovení, akademické tituly, email a telefonní číslo. Pro jejich změnu stačí najít příslušný vstup a údaj změnit. U jména a příjmení je změna provedena pomocí okna, které se zobrazí po stisku editační ikony za jménem uživatele. Ve spodní části této obrazovky se nachází tlačítka, které odkazují na obrazovky s doklady totožnosti, bankovními údaji a adresami uživatele.

Doklady totožnosti jsou stěžejní informace uživatelského účtu z důvodu důkladného ověření totožnosti uživatele. U každého uživatele jsou vyžadovány dva doklady totožnosti, z nichž jeden musí být občanský průkaz. Změna u těchto dokladů povede k pozastavení funkčnosti účtu, dokud nedojde k manuální kontrole změn. Po přístupu do této sekce je nejprve zobrazena tabulka se seznamem všech zadaných dokladů s jejich platností a po kliknutí

na vybraný doklad dojde k přesměrování na jeho detail. V detailu je dále uvedeno číslo daného dokladu, datum vydaní, kdo doklad vydal a fotografie obou stran. V návrhu je počítáno s možností tyto fotografie uložit do galerie, popřípadě nahrát nové.



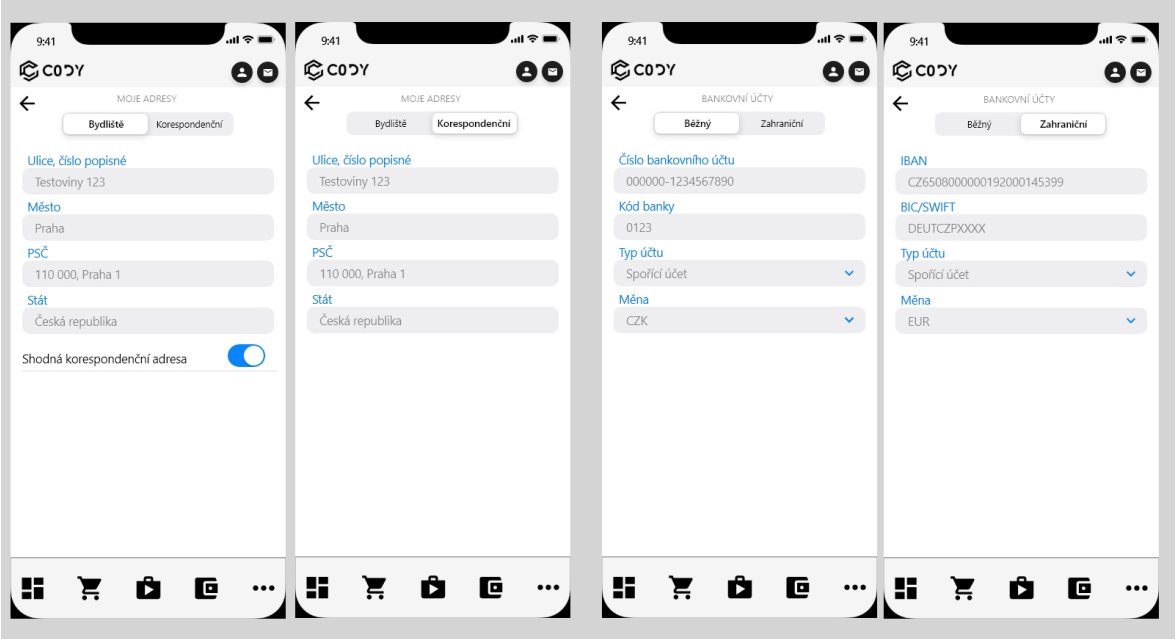
Obrázek 45: Návrh obrazovky – profil uživatele



Obrázek 46: Návrh obrazovky – doklady totožnosti a detail dokladu

Adresy uživatele jsou dostupné v samostatné sekci „Moje adresy“. V této sekci je uvedena adresa trvalého bydliště uživatele a v případě potřeby je zde možné zadat i adresu korespondenční.

Pro vyplacení odměn z investování nebo přijetí požadavku na nabití interní peněženky musí uživatel zadat údaje o svém primárním bankovním účtu. Tyto údaje se následně nachází v části bankovní účty, kde je uvedeno číslo tohoto účtu, kód banky, typ účtu a používaná měna. V návrhu bylo nutné počítat i s variantou přidání druhého zahraničního účtu, kdy jsou jednotlivé účty od sebe odděleny pomocí přepínacího tlačítka v úvodu obrazovky.



Obrázek 47: Návrh obrazovky – „Moje adresy“ a bankovní účty

4.3 Datové entity

V aplikaci je pracováno s několika různými datovými entitami, které jsou získány pomocí GET a POST dotazů ze serveru. Tyto entity jsou po získání zpracovány v příslušných kontrolérech a zobrazeny na obrazovkách aplikace. Tato sekce se věnuje jejich stručnému popisu a výčtu důležitých proměnných, které obsahují.

4.3.1 Uživatel

Entita User obsahuje veškerá potřebná data o přihlášeném uživateli v aplikaci. Jedná se o běžné uživatelské údaje reprezentující identitu daného uživatele společně s jeho kontaktními údaji nebo akademickými tituly (viz obrázek č. 48). Pro potřeby aplikace jsou součástí této entity podrobná data o adresách, bankovních účtech a dokladech totožnosti uživatele (viz obrázek č. 49). Všechna tato data jsou získána při prvotní registraci uživatele a v rámci této aplikace je lze nalézt v uživatelském profilu.

Obrázek 48: Entita Uživatel

Obrázek 49: Entita Uživatel – adresy, bankovní účty, doklady totožnosti

4.3.2 Společnost

Každá společnost v aplikaci je reprezentována touto entitou. Objekt této entity je obdržen po úspěšném přidání společnosti pomocí vygenerovaného kódu nebo oskenování QR kódu z webové aplikace. Získaný objekt je následně uložen do lokálního uložiště, kde se nachází pole se všemi již přidanými společnostmi.

Jednou z proměnných objektů je *uid*, které slouží k identifikaci společnosti v zabezpečeném uložišti aplikace, pro zisk refresh api tokenu potřebného pro přihlášení pomocí

faceID, touchID a otisku prstu. Proměnná *api* slouží k nastavení správné api adresy v server servisu pro volaní GET a POST dotazů. Pro zobrazení obrázku v přehledu společností a zobrazení loga v hlavičce aplikace slouží proměnné *imageUid* a *logoUid*. Tyto proměnné jsou poslány pomocí dotazu GET na server, který následně vrací odkaz na samotný obrázek. Poslední proměnnou je objekt *colors*, který obsahuje proměnné definující celkové barevné rozložení aplikace pro každou společnost.

```
company: {
  uid: "uid_server_1",
  name: "C0DY",
  description: "CODY Company"
  api: "c0dy"
  imageUid: "uid_image_1",
  logoUid: "uid_logo_1",
  colors: {
    mainColor: "#428cef",
    background: "#e0e0e0"
  }
}
```

Obrázek 50: Entita Společnost

4.3.3 Investice

Investice je v aplikaci reprezentována pomocí dvou různých entit (viz obrázek č. 51). První entita popisuje investici z pohledu uživatele, kdy se jedná o investici, do které již bylo zainvestováno. Druhá entita definuje parametry nabízené investice v tržišti aplikace.

U každé aktivní nebo proběhlé investice je definován typ investice společně s její délkou v proměnných *offerName* a *name.* Pro uživatele jsou důležité informace popisující stav a celkový průběh vyplácení odměn z investice. Tyto informace obsahují proměnné *paymentsAmountSum*, *paymentsAmountPaid* a *paymentsAmountLeft*. Pro přehlednější sledování plateb a pohybů obsahuje každá investice pole s těmito údaji, které jsou následně zobrazeny v příslušných sekcí detailu investice. Datum první a poslední výplaty společně s datem vrácení jistiny uvádí proměnné *firstInterestDate*, *lastInterestDate* a *principalReturnDate*. Pro snadnější dohledání má každá investice vlastní *identifier, offerIdentifier* a záznam, kdo danou investici sjednal.

Entita investice v tržišti obsahuje několik shodných proměnných, které byly popisovány v předchozím odstavci. Kromě nich jsou v této entitě obsaženy hlavně informace sloužící k celkovému popisu a definici parametrů pro každou nabízenou investici v aplikaci. Jedná se například o minimální vklad a aktuální naplněnost v proměnných *minBondAmount*

a *filling* nebo riziko investice a způsob výplaty úroku v proměnných *risk* a *paymentFrequency*.

Obrázek 51: Entity Investice

4.3.4 Peněženky

Jedná se o nejjednodušší entitu v aplikaci. Získané pole obsahuje všechny dostupné peněženky v aplikaci. V současné verzi se jedná o peněženku s českými korunami a eury. V budoucím rozšíření je dále plánováno přidání peněženky operující s tokeny.

Každá z peněženek osahuje *id* pro identifikaci, používanou měnu a zůstatek přihlášeného uživatele.

Obrázek 52: Entita Peněženky

4.3.5 Transakce

Transakce se vyskytují napříč celou aplikací, ať se jedná o provedení investice, převod mezi peněženkami nebo již vyplacené odměny. Z tohoto důvodu je nutné umět tyto transakce třídit, a to je možné pomocí proměnné *type*. Díky této a proměnné *description* je možné určit, zda se jedná o vklad, výběr, vyplacení odměny nebo provedení investice. Jelikož se v aplikaci vyskytují dvě peněženky s různými měnami, je nutné uchovávat kromě částky v proměnné *amount* také informaci o použité měně. Datum provedení transakce reprezentuje položka *dateOfTransaction*.

```
transaction: {
  uid: "uid_tran_1",
  type: "DEPOSIT_BANK",
  dateOfTransaction: "2021-04-16",
  description: "Vrácení jistiny",
  amount: 100000.0,
  currency: "CZK"
}
```

Obrázek 53: Entita Transakce

4.3.6 Interní zpráva

Interní zprávy slouží pro informování uživatele o důležitých událostech a novinkách v aplikaci. Každá interní zpráva je reprezentací této entity a je definována typem pro rozlišení zprávy, předmětem, datem doručení a samotným textem zprávy. Proměnné *opened*, *deleted* a *important* označují, zda byla daná zpráva otevřena, smazána nebo označena za důležitou.

```
message: {
  uid: 1,
  category: "NEWS",
  subject: "Nová investice",
  date: "1.1.2021",
  text: "Novou investicí tento měsíc ..",
  important: false,
  opened: false,
  deleted: false,
}
```

Obrázek 54: Entita Interní zpráva

4.3.7 Log události

Tato entita definuje strukturu logu události, který je vytvořen po provedení sledovaných činností v aplikaci. Položka *createdAt* reprezentuje datum a čas, kdy byla tato činnost provedena. Logy událostí jsou rozděleny do několika kategorií, a tato informace je obsažená v proměnné *type*. Pro zobrazení typu logu v prostředí aplikace slouží proměnná *category*, která obsahuje hlavní kategorii příslušného a společně s proměnnou *subCategory* a *text* pomáhá podrobněji popsat zachycenou činnost. Samozřejmě je nutné identifikovat, kdo danou činnost vyvolal nebo autorizoval. Tuto informaci obsahuje proměnná *authorizedBy*, kdy se může jednat o uživatele nebo systém. Položka *status* pomáhá rozeznat konečný stav provedeného úkonu. Pomocí této proměnné lze například určit, zda se jedná o záznam úspěšného nebo neúspěšného přihlášení. U určitých typů logů je uveden identifikační kód v proměnné *logCode*.

```
log: {
  createdAt: "01.01.2021 13:20",
  category: "AUTHENTICATION",
  subCategory: "LOGIN",
  text: "User logged in",
  authorizedBy: "CLIENT",
  logCode: 1,
}
```

Obrázek 55: Entita Log události

4.4 Vývoj aplikace

Tato sekce popisuje nejdůležitější a nejzajímavější části implementace prototypu mobilní aplikace. Aplikace byla vyvíjena jako hybridní s využitím javascriptového frameworku Ionic. V rámci tohoto frameworku byl zvolen Apache Cordova a programovací jazyk Angular.

Primární vývoj a testování probíhalo pro platformu Android. Kvůli využití nativní HTTP knihovny nebylo možné aplikaci testovat pomocí prohlížeče s využitím příkazu *ionic serve*. Vývoj tedy probíhal pomocí častého sestavování s následným testováním pomocí virtuálního zařízení v programu Android Studio. V pokročilé fázi vývoje testování probíhalo na dvou fyzických zařízeních. Jednalo se o mobilní telefony Xiaomi Redmi 9C a Samsung Galaxy A3. Rozdíly mezi těmito modely byly jak v jejich stáří, velikosti displeje, dostupných funkcionalitách, tak ve verzi Androidu. Tyto různé parametry byly vyhovující pro rozsáhlejší

testování a představu o fungování aplikace na různých zařízeních. Pro operační systém iOS byly prováděny periodické testy po aplikování většího množství funkcionalit za účelem ověření funkčnosti aplikace na tomto systému. K tomuto testování byly využity virtuální zařízení v aplikaci Xcode.

Celá aplikace je koncipována jako API based. Veškeré získávání dat a jejich úprava probíhá pomocí HTTP dotazů na server. U většiny obrazovek dochází po přístupu k volání metody GET pro zisk potřebných dat a k jejich následnému zobrazení. Celý vývoj byl směřován tak, aby aplikace/klientská část sloužila jako zobrazovací jednotka a veškeré operace s daty zpracovával server. Mezi tyto operace lze zařadit filtrování, stránkování, kategorizaci nebo složitější matematické výpočty. Pro dodatečnou funkcionalitu bylo využito několika dostupných pluginů, které jsou obsáhleji popsány v samostatné kapitole.

4.4.1 Autentizace a zabezpečení aplikace

Aplikace pracuje s velkým množstvím citlivých a osobních údajů uživatelů. Z tohoto důvodu byl kladen velký důraz na použití vhodných metod zabezpečení. Tato kapitola popisuje zvolené metody zabezpečení u jednotlivých způsobů autentizace a přenosu dat.

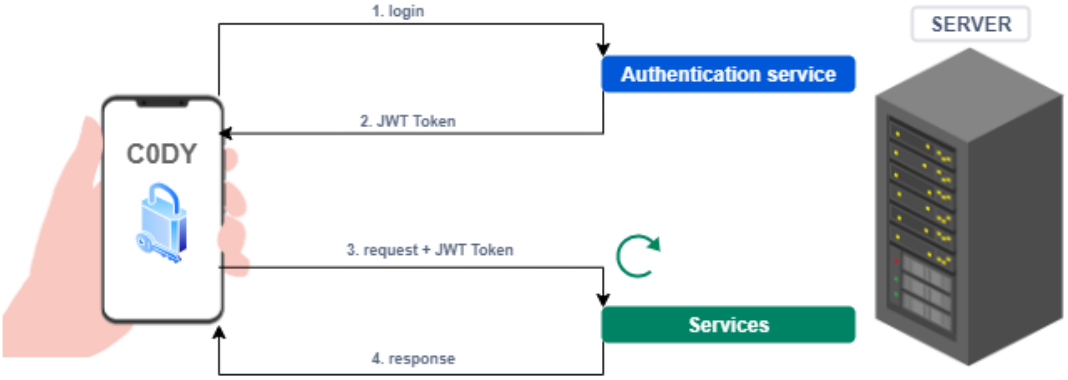
Standardní přihlášení

Základním kamenem zabezpečení je upravený *JSON Web Token* (JWT) token. Tento token je od serveru získán po úspěšném přihlášení pomocí loginu a hesla (viz obrázek č. 56). Samotný přenos tokenu je zabezpečen *Hypertext Transfer Protocol Secure* (HTTPS) protokolem a uložen pomocí *secured cookies*. Díky nastavení *httpOnly* nelze k těmto cookies přistupovat skrze Java-Script nebo prohlížeč.

S každým následným dotazem je tento token odeslán a ze strany serveru dochází k ověření, zda se jedná o jím vytvořený token a zda je stále platný. Při přechodech mezi obrazovkami rovněž dochází k vyvolání tak zvaného *healthChecku*. Při vyvolání této akce je na server poslán dotaz pro kontrolu platnosti JWT tokenu, kdy v případě zjištění, že tento token není platný nebo došlo k jeho časové expiraci, dochází k odhlášení uživatele z aplikace.

Samotný JWT token obsahuje data sloužící k autentizaci, jako jsou role nebo přístupová práva. Ve většině případů se jedná o informace, které není nutní opakovaně načítat

z databáze z důvodu časové náročnosti. Tato data jsou hashována a tento hash následně zašifrován, kdy šifrovací klíč je znám pouze serveru. U tohoto řešení je používán upravený JWT token, kdy dochází k dalšímu šifrování těchto dat s cílem znemožnit jejich čtení.



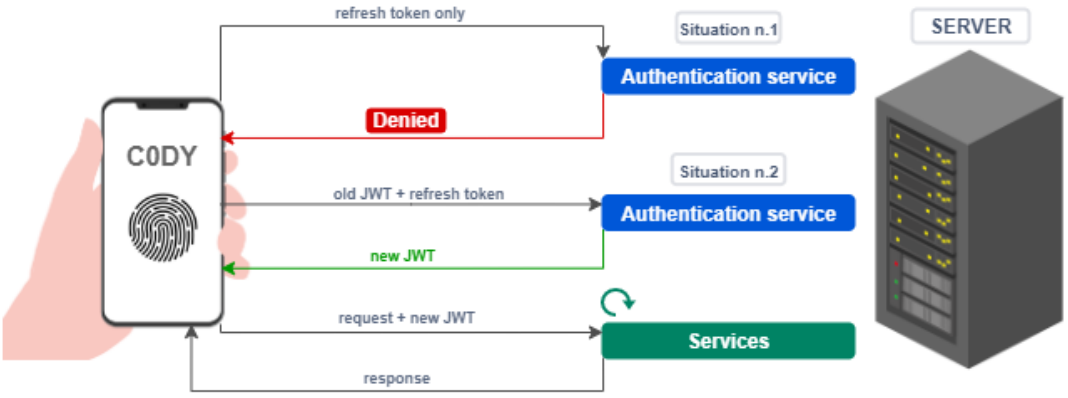
Obrázek 56: Schéma standardního přihlášení

Rozšířené přihlášení

Rozšířené přihlášení je přihlášení s využitím otisku prstu, Touch ID nebo Face ID. Při použití tohoto způsobu přihlášení nejsou zasílány přihlašovací údaje ve formě loginu a hesla, a proto není možné provést standardní autentizaci a zaslat ověřovací JWT token. Pro tyto účely byl implementován takzvaný refresh token.

U prvotního přihlášení je nutné, aby se uživatel přihlásil skrze standardní údaje a v sekci *Nastavení* aktivoval možnost rozšířeného přihlášení, které jeho zařízení nabízí. Při samotné aktivaci dochází k odeslání *Universally unique identifier* (UUID) zařízení na server, kde je tento identifikátor spojen s daným účtem a dále používán jako refresh token. Tento token je společně s expirovaným JWT tokenem posílán při rozšířeném přihlášení jako další identifikátor zabezpečení. Po úspěšném přihlášení obdrží zařízení nový JWT token, který je následně posílán s každým dalším dotazem na server (viz obrázek č. 57).

V budoucím rozšíření bude UUID nahrazeno refresh tokenem vygenerovaným na straně serveru a uloženým pomocí Ionic Secure Storage v zařízení uživatele.



Obrázek 57: Schéma rozšířeného přihlášení

4.4.2 Použité pluginy a knihovny

V průběhu vývoje aplikace bylo využito několika rozšiřujících pluginů, které umožňují využít pokročilých funkcí daného zařízení. Nadcházející podkapitoly obsahují výčet použitých pluginů společně s popisem jejich konkrétního použití ve vyvinuté aplikaci.

HTTP

Veškeré získávání dat a filtrování v tabulkách probíhá pomocí dotazů GET a POST na server dané společnosti. Pro tyto účely byla implementována nativní knihovna HTTP a vytvořena servisní třída v aplikaci pro zpracování jednotlivých dotazů. Tato servisní třída obsahuje univerzální funkce *get* a *post*, do kterých je při jejich volání vložen typ požadavku s případnými daty, pokud jsou u požadavku vyžadovány. U dotazu POST bylo nutné provést validní nastavení hlaviček zasílaných s tímto dotazem tak, aby byly tyto dotazy ze strany serveru úspěšně zpracovány. Pro úspěšné zpracování je nutné, aby hlavičky těchto dotazů obsahovaly X-XSRF token, který je obdržen po úspěšném přihlášení. V aplikaci se celkově vyskytuje okolo 50 různých dotazů na server.

```
get(request, data){

  let url = this.apiPre + this.companyApi + this.apiPost + request;
  let params = data;
  let self = this;

  var promise = new Promise(function(resolve, reject) {

    self.http.get(url, params, "")
    .then((response:HTTPResponse) => {

      resolve(response);
    })
    .catch((error:any) => {

      reject(error);
    });

  });

  return promise;
}
```

Obrázek 58: Univerzální funkce *get*

```
post(request, data){

  let url = this.apiPre + this.companyApi + this.apiPost + request;
  let params = data;
  let self = this;

  let headers =
  {
    "Access-Control-Allow-Credentials": "true",
    "Accept": "application/json",
    "Content-Type": "application/json",
    "Access-Control-Expose-Headers": "X-Auth-Token",
    "Access-Control-Allow-Origin": "true",
    "origin-device" : "MOBILE",
    "X-XSRF-TOKEN":  this.token.token,
  };

  var promise = new Promise(function(resolve, reject) {

    self.http.post(url, params, headers)
    .then((response:HTTPResponse) => {

      resolve(response);
    })
    .catch((error:any) => {

      reject(error);
    });
  });

  return promise;
}
```

Obrázek 59: Univerzální funkce *post*

Barcode Scanner

Plugin Barcode Scanner umožňuje čtení a následné zpracovaní QR kódů pomocí fotoaparátu telefonu. Této funkcionality bylo využito pro přidávání společnosti do aplikace pomocí oskenování QR kódu z webové aplikace.

Base64 To Gallery

Plugin Base64 To Gallery slouží k uložení obrázku ve formátu base64 do galerie telefonu. Tento plugin je v aplikaci využíván k uložení vygenerovaného QR kódu pro internetové bankovnictví v sekci *Dobít peněženku* a k uložení obrázků dokladů totožnosti v profilu uživatele. Před pokusem o uložení do galerie telefonu je nutné zkontrolovat zapisovací práva.

```
downloadQR(){

  if (!this.hasWriteAccess) {
    this.checkPermissions();
  }

  const canvas = document.querySelector('canvas') as HTMLCanvasElement;
  const imageData = canvas.toDataURL('image/jpeg').toString();
  var data = imageData.split(',')[1];

  this.base64ToGallery
    .base64ToGallery(data, { prefix: '_img', mediaScanner: true })
    .then(
      async res => {

        var notDada = {text: "QR kód uložen do galerie.", color: "success", icon: "qr-code"}
        this.notService.showNotification(notDada);
      },
      async err => {

        var notDada = {text: "Uložení selhalo.", color: "danger", icon: "bug"}
        this.notService.showNotification(notDada);
      }
    );
}
```

Obrázek 60: Funkce pro uložení QR kódu do galerie telefonu

Fingerprint AIO

Tento plugin umožňuje přístup k rozšiřujícím metodám autentizace, jako je otisk prstu, Touch ID nebo Face ID. Tato funkcionalita je využita jako možnost přihlášení do aplikace. Prvotní přihlášení je vždy standardní a tato možnost přihlášení musí být aktivována v sekci nastavení.

Při přístupu na obrazovku s přihlášením si aplikace ověří, zda je rozšiřující možnost přihlášení dostupná a zda je v rámci aplikace aktivována. V případě úspěšného ověření je zobrazeno příslušné okno a uživatel je vyzván k autentizaci pomocí nalezené metody.

Camera

Plugin Camera umožňuje přístup a následné nahrání obrázku z galerie telefonu do aplikace. Tento plugin je využíván v profilu uživatele pro změnu profilové fotky a fotek dokladů totožnosti.

```
loagFromGallery(){

  this.camera.getPicture({

    sourceType: this.camera.PictureSourceType.PHOTOLIBRARY,
    destinationType: this.camera.DestinationType.DATA_URL

  }).then((res) => {

    this.imgUrl = ''+ res;

    var notDada = {text: "Úspěšné nahrání obrázku.", color: "success", icon: "image"}
    this.notService.showNotification(notDada);

  }).catch(error => {

    var notDada = {text: "Nahrání obrázku selhalo.", color: "danger", icon: "bug"}
    this.notService.showNotification(notDada);
  })
}
```

Obrázek 61: Funkce pro nahrání obrázku z galerie telefonu

Ngx-QRCode

Ngx-QRCode je Angular knihovna umožňující generování QR kódů v aplikaci. Pomocí této knihovny je generován QR kód pro internetové bankovnictví v sekci *Dobít peněženku*.

U vytváření QR kódu pro internetové bankovnictví je nutné dodržet požadovanou strukturu dat, aby byl tento kód přijat (viz obrázek č. 62). Pro vytvoření vzorového bankovního účtu bylo využito random *International Bank Account Number* (IBAN) generátoru z důvodu, že i u toho čísla jsou jistá pravidla na jeho tvar a nelze si toto číslo vymyslet.

```
SPD*1.0*ACC:CZ6508000000192000145399*AM:200*CC:CZK*X-VS:090421*
```

Obrázek 62: Struktura dat QR kódu pro internetové bankovnictví

Obrázek 63: Zobrazení QR kódu v aplikaci

Android Permissions

Tento plugin umožňuje zjistit stav oprávnění aplikace u zařízení Android. Plugin byl použit z důvodu objevujícího se problému s oprávněním při ukládání obrázku do galerie zařízení. Vytvořená funkce ověří dostupnost všech nutných oprávnění a v případě úspěšného ověření je zahájeno ukládání. V situaci, kdy některé oprávnění chybí, je uživatel vyzván k jeho udělení.

Device

Plugin Device dokáže poskytnout mnoho důležitých informací o zařízení, na kterém daná aplikace běží. V aplikaci je tento plugin využit pro zisk *uuid* zařízení, které se následně používá jako refresh token při autentizaci pomocí otisku prstu, Face ID nebo Touch ID.

Chart.js

Chart.js je bezplatná javascriptová knihovna pro vizualizaci dat ve formě grafů. Knihovna byla použita k vytvoření grafů reprezentující dlouhodobé statistiky o investování uživatele v sekci *Statistiky*. Vyvinutá verze aplikace obsahuje 6 typů grafů.

4.5 Budoucí rozšíření aplikace

Požadavky na aplikaci stanovené v rámci této diplomové práce byly splněny. Kromě prototypu vznikla i řada potencionálních vylepšení a rozšíření, které by mohly být integrovány při budoucím vývoji aplikace.

Prvním vylepšením je implementace systému push notifikací, který by uživatele informoval o nových investicích a důležitých událostech v aplikaci. Výhodou tohoto řešení je možnost informovat uživatele i v situaci, kdy aktivně nepoužívá aplikaci. Z prvotních návrhů možných řešení je momentálně nejpravděpodobnější řešení pomocí Google Firebase.

Aplikace obsahuje funkcionalitu pro nahrávání obrázku z galerie za účelem změny profilového obrázku a fotek dokladů totožnosti. Především u dokladů totožnosti je nutné, aby nahrané fotky byly čitelné a měly doporučený formát. I když většina telefonů obsahuje jednoduchý editor, ve kterém lze fotky upravovat, ne u všech zařízení je tento editor přehledně řešen nebo o nich všichni uživatelé nevědí. Kvůli těmto důvodům je v budoucím vývoji plánována integrace ořezávání obrázku přímo v aplikaci při jeho nahrávání. Cílem této implementace je snížení počtu špatných fotek dokladů objevených při jejich manuální kontrole.

V tomto typu aplikací je s uživatelem a jeho účtem spojeno mnoho dokumentů, jako jsou například smlouvy, obchodní podmínky nebo interní předpisy. V budoucím rozšíření se počítá s možností tyto dokumenty otevřít a případně uložit do uložiště zařízení.

Jako poslední zmíněným rozšířením je integrování Ionic Secure Storage pro uložení refresh tokenu potřebného pro přihlášení pomocí otisku prstu, Face ID a Touch ID Ionic Secure Storage poskytuje komfortní a zabezpečené uložení data na daném zařízení s jednoduchým přístupem. Rozšíření bude aplikováno v pozdější fázi vývoje z časových důvodů a také nutnosti placeného vývojářského účtu Ionic Enterprise.

ZÁVĚR

V rámci této diplomové práce se podařilo vytvořit multiplatformní mobilní aplikaci pro podporu prodeje s funkčním sestavením pro Android i iOS. Primární pozornost byla věnována vývoji pro systém Android. Před samotným vývojem byl rozsáhle zpracován celkový návrh aplikace v podobě funkčních a nefunkčních požadavků, případů užití a kompletního návrhu uživatelského rozhraním včetně funkčního prototypu v prostředí Adobe XD. Tento návrh velice usnadnil získání celkové představy o velikosti aplikace, množství práce a pomohl tuto práci rozdělit na jednotlivé vývojové milníky. Velice kladně byl ze strany společnosti přijat vytvořený prototyp, jelikož v rané fázi vývoje poskytl vizuální představu o podobě aplikace.

Vzhledem k požadavkům na multiplatformost a technologiím využívaným v rámci firmy byl pro vývoj zvolen framework Ionic. Ionic má velice kvalitní a podrobnou dokumentaci, takže prvotní nastavení projektu a celkové pochopení fungování frameworku je velice snadné. Pozitivně hodnotím balíček UI komponent, které jsou responzivní a fungují dobře na obou platformách. Menší problém nastává s pokusem o větší úpravu těchto komponent mimo jejich parametry, kdy může dojít ke ztrátě její funkčnosti. Počet nativních pluginů je rovněž velký, což umožňuje použití tohoto frameworku u různých typů aplikací. Nutné je však zmínit, že u těchto pluginů není nijak garantována jejich podpora a jejich vývoj může být kdykoliv ukončen.

V teoretické části práce byly popsány možnosti vývoje mobilních aplikací společně s výčtem nástrojů, které lze k vývoji použít. Zpracování těchto částí pomohlo k celkovému pochopení jednotlivých přístupů s jejich výhodami a nevýhodami a také k získaní přehledu o aktuálních trendech a novinkách při vývoji mobilních aplikací.

Hlavním výstupem této práce je prototyp aplikace pro podporu prodeje, která bude po svém dokončení sloužit jako doplňující rozšíření stávající webové aplikace společnosti Czech Premium Product s.r.o. . Část zpracovaná v rámci této práce reprezentuje úvodní část vývoje, po které bude následovat důkladné testování pro obě cílené platformy a aplikování případných rozšíření, které byly popsány v předchozí kapitole.

SEZNAM POUŽITÉ LITERATURY

[1] Native, Hybrid, or Web Apps: What Is the Best Approach for Lasting Success? [online]. [cit. 2020-23-12]. Dostupné z: https://medium.com/better-programming/native-hybrid-or-web-apps-what-is-the-best-approach-for-lasting-success-91afbb872d89

[2] Native Apps, Web Apps or Hybrid Apps? What's the Difference? [online]. [cit. 2020-23-12]. Dostupné z: https://www.mobiloud.com/blog/native-web-or-hybrid-apps

[3] Mobile Operating System Market Share Worldwide [online]. [cit. 2020-23-12]. Dostupné z: https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-202001-202012-bar

[4] Tablet Operating System Market Share Worldwide [online]. [cit. 2020-23-12]. Dostupné z: https://gs.statcounter.com/os-market-share/tablet/worldwide/#monthly-202001-202012-bar

[5] Vývoj hybridní aplikace pro podnikání, srovnáváme pro a proti [online]. [cit. 2020-25-12]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-hybridni-aplikace

[6] Ionic React vs React Native [online]. [cit. 2020-29-12]. Dostupné z: https://ionicframework.com/resources/articles/ionic-react-vs-react-native

[7] Apache Cordova Architecture [online]. [cit. 2020-29-12]. Dostupné z: https://ionicframework.com/resources/articles/ionic-react-vs-react-native

[8] Hybridní, nativní nebo webové aplikace? [online]. [cit. 2021-02-01]. Dostupné z: http://www.hybridniaplikace.cz/srovnani.html

[9] Advantages and disadvantages of native mobile application development [online]. [cit. 2021-02-01]. Dostupné z: https://www.writersevoke.com/advantages-and-disadvantages-of-native-mobile-application-development/

[10] What's the Difference between Native vs. Web vs. Hybrid Apps? [online]. [cit. 2021-02-01]. Dostupné z: https://getgist.com/difference-between-native-vs-web-vs-hybrid-apps/

[11] How To Choose Between Native, Hybrid or Web App For Your Business [online]. [cit. 2021-02-01]. Dostupné z: https://buildfire.com/choose-native-hybrid-web-mobile-app/

[12] GRIFFITH, Chris W. Mobile app development with Ionic: cross-platform apps with Ionic, Angular, and Cordova. Revised edition. Sebastopol, CA: O'Reilly Media, [2017]. ISBN 9781491998120

[13] SINGH, Indermohan. Ionic Cookbook: Recipes to create cutting-edge, real-time hybrid mobile apps with Ionic. 3. Birmingham: Packt Publishing, [2018]. ISBN 978-1788623230

[14] Mobile: Native Apps, Web Apps, and Hybrid Apps [online]. [cit. 2021-04-01]. Dostupné z: https://www.nngroup.com/articles/mobile-native-apps/

[15] SAINI, Gaurav. Hybrid Mobile Development with Ionic. Birmingham: Packt Publishing, [2017]. ISBN 978-1785286056

[16] A Guide to Mobile App Development: Web vs. Native vs. Hybrid [online]. [cit. 2021-04-01]. Dostupné z: https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/

[17] The state of PWAs in 2020 per OS [online]. [cit. 2021-04-01]. Dostupné z: https://simplabs.com/blog/2020/06/10/the-state-of-pwa-support-on-mobile-and-desktop-in-2020/

[18] React Native vs. Xamarin vs. Ionic vs. Flutter: Which is best for Cross-Platform Mobile App Development? [online]. [cit. 2021-15-01]. Dostupné z: https://www.apptunix.com/blog/frameworks-cross-platform-mobile-app-development/

[19] Flutter vs. React Native vs. Xamarin [online]. [cit. 2021-29-01]. Dostupné z: https://blog.logrocket.com/flutter-vs-react-native-vs-xamarin/

[20] React Native vs Flutter vs Xamarin: A Comparative Guide [2020] [online]. [cit. 2021-29-01]. Dostupné z: https://www.codementor.io/@mahil/react-native-vs-flutter-vs-xamarin-a-comparative-guide-2020-11evx1f7q4

[21] Xamarin vs React Native vs Ionic vs NativeScript: Cross-platform Mobile Frameworks Comparison [online]. [cit. 2021-29-01]. Dostupné z: https://www.altexsoft.com/blog/engineering/xamarin-vs-react-native-vs-ionic-vs-nativescript-cross-platform-mobile-frameworks-comparison/

[22] React Native: What it is and how it works [online]. [cit. 2021-29-01]. Dostupné z: https://medium.com/we-talk-it/react-native-what-it-is-and-how-it-works-e2182d008f5e

[23] Advantages and Disadvantages of Using React Native [online]. [cit. 2021-29-01]. Dostupné z: https://techexactly.com/blogs/advantages-and-disadvantages-of-using-react-native

[24] Pros and Cons — Xamarin vs React Native vs Ionic vs Flutter vs PhoneGap vs Appcelerator Titanium — Which One is Right for You? [online]. [cit. 2021-29-01]. Dostupné z: https://tempesthouse.medium.com/pros-and-cons-xamarin-vs-react-native-vs-ionic-vs-flutter-vs-phonegap-vs-appcelerator-titanium-73e155b260f0

[25] Flutter vs React Native vs Xamarin for Cross Platform Development [online]. [cit. 2021-30-01]. Dostupné z: https://hackernoon.com/flutter-vs-react-native-vs-xamarin-for-cross-platform-development-5f92cfb178ff

[26] What is Xamarin? [online]. [cit. 2021-30-01]. Dostupné z: https://docs.microsoft.com/cs-cz/xamarin/get-started/what-is-xamarin

[27] Flutter architectural overview [online]. [cit. 2021-01-02]. Dostupné z: https://flutter.dev/docs/resources/architectural-overview

[28] How NativeScript works? [online]. [cit. 2021-12-02]. Dostupné z: https://www.kodehero.in/how-nativescript-works/

[29] An Introduction to NativeScript [online]. [cit. 2021-12-02]. Dostupné z: https://code.tutsplus.com/articles/an-introduction-to-nativescript--cms-26771

[30] How NativeScript Works [online]. [cit. 2021-13-02]. Dostupné z: https://docs.nativescript.org/core-concepts/technical-overview#how-nativescript-works

[31] Flutter vs NativeScript: Everything You Need to Know [online]. [cit. 2021-15-02]. Dostupné z: https://www.simform.com/flutter-vs-nativescript/

[32] What is Ionic Framework? and Why Use it over Other Frameworks? [online]. [cit. 2021-20-02]. Dostupné z: https://hackr.io/blog/ionic-framework

[33] The Good and the Bad of Ionic Mobile Development [online]. [cit. 2021-20-02]. Dostupné z: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-ionic-mobile-development/

[34] Ionic Framework [online]. [cit. 2021-20-02]. Dostupné z: https://ionicframework.com/docs

[35] Capacitor: Cross-platform Native Runtime for Web Apps [online]. [cit. 2021-20-02]. https://capacitorjs.com/docs

[36] Vytvořte si appku v Angularu, aneb výhody frameworku Ionic [online]. [cit. 2021-20-02]. https://fragaria.cz/blog/2017/03/31/vytvorte-si-appku-v-angularu/

[37] Introducing Ionic 4: Ionic for Everyone [online]. [cit. 2021-21-02]. https://ionicframework.com/blog/introducing-ionic-4-ionic-for-everyone/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Application Programming Interface
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
UX	User Experience
OS	Operating system
SDK	Software Development Kit
GPS	Global Positioning System
CPU	Central Processing Unit
GPU	Graphics Processing Unit
URL	Uniform Resource Locator
PWA	Progressive Web Apps
IL	Intermediate language
ART	Android runtime
MCW	Managed Callable Wrappers
AOT	Ahead-of-Time
MVC	Model-view-controller
MVVM	Model–view–viewmodel
CLI	Command Line Interface
BSD	Berkeley Software Distribution
HTTP	Hypertext Transfer Protocol
XML	Extensible Markup Language
DOM	Document Object Model
UI	User Interface

IDE	Integrated Development Environment
APK	Android application package
IBAN	International Bank Account Number
JWT	JSON Web Token
HTTPS	Hypertext Transfer Protocol Secure
UUID	Universally unique identifier

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: OBSAH CD

Přiložené CD obsahuje:

- Text
    - DP_Valuch-Patrik.pdf – diplomová práce ve formátu pdf
- Zdrojové kódy a návrh v aplikaci Adobe XD
    - DP_praktickaCast_Valuch-Patrik.zip – všechny zdrojové kódy a návrh aplikace

PŘÍLOHA P II: VYJÁDŘENÍ SPOLEČNOSTI CZECH PREMIUM PRODUCT S.R.O.

VYJÁDŘENÍ SPOLEČNOSTI CZECH PREMIUM PRODUCT S.R.O

S panem Patrikem Valuchem dlouhodobě spolupracujeme na různých projektech v oblasti vývoje webových aplikací a proto jsme souhlasili s jeho nabídkou zpracování mobilní aplikace C0DY v rámci jeho diplomové práce. Pan Valuch s námi průběžně konzultoval výstupy jeho práce a reagoval na naše případné požadavky na změny. Velice kladně hodnotíme finální podobu aplikace, která vizuálně působí velmi hezky a splňuje požadavky na stanovenou funkcionalitu. Těšíme se na pokračování intenzivní práci vedoucí k finálnímu dokončení této aplikace a k jejímu nabídnutí našim zákazníkům. V závěru bychom chtěli vyzdvihnout vytvořený prototyp v programu Adobe XD, který vznikl v rámci této práce. Díky jeho kvalitnímu zpracování ho lze z naší strany využit k prezentačním účelům, dokud nebude tato aplikace dokončena.

26.4.2021 Ing. Pavel Nedvídek, v.r.