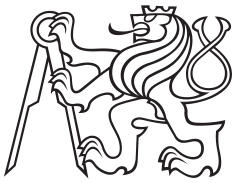
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Systém pro správu tanečního studia

Daniil Simon

Vedoucí: Ing. Pavel Náplava, Ph. D.
Obor: Softwarové inženýrství a technologie
Květen 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Simon** Jméno: **Daniil** Osobní číslo: **495640**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Systém pro správu tanečního studia

Název bakalářské práce anglicky:

A dance studio management system

Pokyny pro vypracování:

Navrhněte a realizujte systém pro podporu menšího tanečního studia. Postupujte následujícím způsobem:
1) Popište hlavní činnosti a procesy vybraného tanečního studia.
2) Analyzujte a popište možnosti zlepšení fungování pomocí podpůrného informačního systému.
3) Zmapujte existující řešení, která by bylo možné pro navržené zlepšení použít
4) Na základě existujících řešení a funkčních požadavků vytvořte pomocí nástrojů SW inženýrství zadání pro vytvoření systému nového, který bude obsahovat minimálně evidenci klientů, rezervační systém, integraci platební brány, „push“ notifikace atd.
5) Vyberte vhodnou platformu pro implementaci, která umožní používat systém jak pomocí webového prohlížeče nebo na mobilních zařízeních, systém implementujte.
6) Vytvořený systém ověřte pomocí uživatelských testů.

Seznam doporučené literatury:

Arlow, J., Neustat, I.: UML 2 a unifikovaný proces vývoje aplikací. Computer Press, ISBN: 978-80-251-1503-9, Praha 2007.
React Native, https://reactnative.dev/
Ionic framework, The mobile SDK for the Web, https://ionicframework.com/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Pavel Náplava, Ph.D. Centrum znalostního managementu FEL

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

Datum zadání bakalářské práce: **09.02.2023** Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **22.09.2024**

Ing. Pavel Náplava, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. | Datum převzetí zadání

Podpis studenta

Poděkování

Chtěl bych vyjádřit své upřímné poděkování svému vedoucímu, Ing. Pavlu Náplavovi, Ph.D., za jeho cenné rady, podporu a čas věnovaný během celého procesu psaní této bakalářské práce. Rovněž bych rád poděkoval všem klientům našeho studia a studentům, kteří se zúčastnili průzkumu a uživatelského testování.

Prohlášení

Tímto prohlašuji, že předložená bakalářská práce je mé vlastní dílo a že jsem uvedl veškeré zdroje informací v souladu s pokyny pro dodržování etických principů při zpracování akademické závěrečné práce.

Uznávám, že na mou práci se vztahují práva a povinnosti stanovené zákonem č. 121/2000 Sb., o právu autorském, ve znění pozdějších předpisů, zejména že České vysoké učení technické v Praze má právo uzavřít licenční smlouvu o užití této práce jako školního díla podle ustanovení § 60 odst. 1 zákona.

V Praze, 25 května , 2023

Abstrakt

Tato bakalářská práce se věnuje analýze, návrhu a vývoji technického řešení pro podporu existujícího tanečního studia. Práce se soustředí na problematiku a řešení problémů v této oblasti, s důrazem na možnosti řešení pomocí již existujících řešení. Na základě provedené analýzy bylo rozhodnuto o vývoji hybridní mobilní aplikace, která slouží jako podpůrný nástroj pro taneční studio.

V rámci analýzy byly zkoumány různé problémy a jejich řešení. Byly zvažovány možnosti využití již existujících řešení, jakož i vývoj nového řešení pro podporu tanečního studia. Byla provedena analýza různých frameworků, s cílem najít nejvhodnější řešení pro implementaci mobilní aplikace. Na základě této analýzy byl vybrán framework Ionic jako nejvhodnější pro implementaci hybridní mobilní aplikace.

Cílem této práce je vytvořit podpůrný nástroj pro existující taneční studio, který umožní klientům snadnou registraci na lekce, využívání intuitivního uživatelského rozhraní s integraci platební brány a poskytování "push" notifikací. Navrhované technické řešení bylo implementováno pomocí Ionic frameworku a bylo ověřeno pomocí uživatelských testů.

Klíčová slova: rezervační systém, taneční studio, Ionic, hybridní mobilní aplikace, iOS, Android, TypeScript, Stripe, platební brána, firebase

Vedoucí: Ing. Pavel Náplava, Ph. D.

Abstract

This bachelor's thesis focuses on analyzing, designing, and developing a technical solution to support an existing dance studio. The work addresses the issues and their solutions in this area, with emphasis on the possibilities of using existing solutions. Based on the analysis, the decision was made to develop a hybrid mobile application that would function as a support tool for the dance studio.

As part of the analysis, various problems and their solutions were explored. The possibilities of using existing solutions as well as developing new solutions to support the dance studio were considered. Different frameworks were analyzed to determine the most suitable solution for implementing the mobile application. Based on this analysis, the Ionic framework was selected as the most appropriate choice for developing a hybrid mobile application.

The primary goal of this work is to create a support tool for the existing dance studio that would enable clients to easily register for classes, use an intuitive user interface with integrate a payment gateway, and receive "push" notifications. The proposed technical solution was implemented using the Ionic framework and was validated through user testing.

Keywords: booking system, dance studio, Ionic, hybrid mobile app, iOS, Android, TypeScript, Stripe, payment gateway, firebase

Title translation: A dance studio management system

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

V této kapitole se snažím uvést čtenáře do problematiky, kterou se práce zabývá. Dále zde představuji motivaci a cíl této práce.

1.1 Předmluva

Tato práce se zaměřuje na vývoj systému pro správu a řízení tanečního studia, který bude poskytovat multiplatformní podporu (web/ios/android) pro klientské rozhraní. Cílem je umožnit klientům přehled o rozvrhu lekcí, umožnit jim registraci nebo zrušení lekcí, zobrazit stav platnosti permanentky a nabídnout možnost její nákupu přímo přes aplikaci.

1.2 Motivace a cíl

Od roku 2020 jsem majitelem tanečního studia. V počáteční fázi svého fungování bylo taneční studio malého rozsahu a zajišťovalo služby pro omezený počet klientů. Kapacita skupinových sezení byla maximálně 5 osob, s průměrem 3 účastníky na jednu lekci. Prostory, ve kterých se kurzy konaly, byly pronajímány na hodinové bázi. Tato situace umožnila efektivní obsluhu klientů ručním způsobem a nevyžadovala velkou časovou náročnost.

Avšak v průběhu času, díky rozšiřování povědomí o tanečním studiu, došlo k nárůstu počtu zájemců o služby, který vyústil v pronájem vlastních prostor a zvýšení kapacity skupinových sezení na 14 osob. Toto zvýšení počtu klientů způsobilo problémy v obchodních procesech, jako např. administrativní zátěž při zaznamenávání lekcí nebo komplikace s bankovními transakcemi u klientů z cizích zemí.

Pozorujeme, že máme značný potenciál pro rozvoj a rozšíření služeb, avšak naše současná technická infrastruktura nám brání v plném využití tohoto potenciálu. V současné době máme k dispozici pouze jednoho trenéra a plánujeme rozšířit tým. Pro udržení konkurenceschopnosti a efektivitu v obchodních procesech je nutné nalézt nebo vyvinout technické řešení, které by umožnilo rozšiřování počtu klientů a zlepšení kvality poskytovaných služeb.

Část I

Teoretická část

Kapitola 2

Současné problémy a navrhovaná řešení

V teto bakalářské práce se zaměřuji na řešení existujících technických problémů v tanečním studiu. Tyto problémy jsou nutné identifikovat a vyřešit s cílem zlepšit kvalitu poskytovaných služeb a podpořit rozvoj podnikání. Pro tento účel budu využívat znalosti získané během tříletého vysokoškolského studia, abych identifikoval současné problémy, určil existující řešení a pokud taková nejsou k dispozici, navrhnul a vyvinul vlastní řešení.

V této sekci se budeme věnovat analýze vnitřních procesů a činností v rámci navrhovaného podniku, identifikaci současných problémů a zhodnocení jejich dopadu na fungování podniku.

2.1 Hlavní problémy současného stavu (činnosti a procesy)

Identifikoval jsem následující procesy, které se v současné době provádějí ručně a vyžadují technické řešení. Tyto procesy jsem dále rozdělil do několika podskupin.

2.1.1 Řízení a moderování tanečních hodin

- registraci do tanečních hodin
- rušení tanečních hodin
- plánování rozvrhu tanečních hodin
- kontrola kapacity

V současné době jsou procesy spojené s řízením a moderováním tanečních hodin prováděny ručně, což znamená značnou náročnost časového rozsahu. Rozvrh hodin je sestavován na začátku každého týdne a klientům je zasílán prostřednictvím messengeru, aby si mohli vybrat vhodný čas a typ lekce. Každá registrace se ručně zaznamenává do sešitu a před každou hodinou, administrátor kontaktuje klienta, aby potvrdil svou účast. Tento způsob je nedostatečný, protože chybí jednotný systém, který by umožňoval klientům vidět aktuální rozvrh s přesným datem a časem, svou rezervaci a umožňoval

ji ručně upravit (zrušit nebo se zapsat), což má za následek, že klienti často zapomínají na hodiny nebo je ruší.

Aktuální model řízení a moderování tanečních hodin je velmi neefektivní, protože vyžaduje vysoké množství času na moderování ze strany správce a neposkytuje dostatečnou flexibilitu pro klienty. Navíc, klienti nejsou schopni vidět aktuální rozvrh hodin, na které se přihlásili, což znamená, že musí kontaktovat administrátora pro kontrolu nebo úpravu své registrace, což představuje zbytečné časové náklady pro klienty. Z důvodu ručního zadávání údajů pomocí notebooku, dále vznikají problémy s kontrolou obsazenosti, kdy administrátor může nevědomě přijmout více klientů, než může kapacita hodin přijmout.

2.1.2 Správa zakoupenych permanentek

- Evidence zakoupených permanentek
- Prodej permanentek
- Kontrola platnosti
- Prodloužení
- Zmrazeni (suspendování platnosti permanentky)

V současné době jsou údaje o permanentkách ukládány do tabulky v aplikaci Excel, což vyžaduje velké množství času na úpravu a aktualizaci údajů pro každého zákazníka. Tento postup je zdrojem častých chyb, které vyžadují další čas na jejich opravu. Navíc, pokud klient odjede například na dovolenou během platnosti permanentky, nebo pokud trenér onemocní, je třeba suspendovat nebo prodloužit platnost permanentky pro klienty, kteří se na hodinu nedostali. Tyto procesy by mohly být výrazně usnadněny jednotným systémem, který by umožnil snadnou moderaci permanentek prostřednictvím jednoduchého a intuitivního uživatelského rozhraní.

2.1.3 Online prodej

- Permanentek
- Inventáře

V současnosti jsou procesy prodeje permanentek a sportovního vybavení založeny na ručním způsobu platby přímo na hodině nebo prostřednictvím komunikace s manažerem. V blízké budoucnosti je plánováno rozšířit nabídku o prodej sportovního vybavení. Pro efektivní a pohodlný prodej permanentek a sportovního vybavení je nezbytné vytvořit jednotný systém pro správu a prodej těchto produktů.

2.1.4 Osobni oblast (personal account) zakaznika

- Historie docházky
- Věrnostní program
- Sběr dat o zakazniku
- Informace o aktuálním stavu předplatného

V současné době neexistuje pro klienty žádný centralizovaný systém, který by jim umožnil sledovat historii své účasti na lekcích, ani zbývající lekce v rámci své permanentky, je nezbytné vytvořit osobní zónu klienta, kde by byly shromážděny a zobrazeny tyto informace.

Sběr dat o zákaznících představuje nedílnou součást všech moderních podniků, které chtějí rozvíjet své aktivity na základě relevantních informací o svých zákaznících. Tyto údaje umožňují lepší pochopení potřeb zákazníků, cílení marketingových aktivit, zlepšení celkové zákaznické zkušenosti a optimalizaci obchodních procesů. Představují tak důležitý nástroj pro rozvoj podnikání.

V rámci cíle sběru dat o zákaznících se zamýšlíme získávat informace jako e-mailové adresy a telefonní čísla, abychom mohli lépe komunikovat s našimi zákazníky a personalizovat nabídky a marketingové kampaně. Tyto informace nám umožní lepší segmentaci zákazníků a cílení reklamy, což by v konečném důsledku mohlo vést ke zvýšení efektivity naší marketingové strategie a větší spokojenosti našich zákazníků [**8**].

2.1.5 Notifikace

- Upozornění o stavu předplatného
- Oznámení o rezervace lekce
- Oznámení o zrušení
- Oznámení o změnách

V současné době se komunikace s klienty provádí prostřednictvím messengeru, což může vést k nedostatečnému doručení oznámení a tím ke ztrátě informací. Pro optimalizaci této komunikace je potřeba využít e-mailových nebo SMS zpráv, push oznámení a dalších formátů, aby byla zajištěna spolehlivost doručení. Pravidelný kontakt s klienty je důležitý pro udržení jejich zájmu o naši společnost a služby [**1**], což přispívá k udržení a zvyšování prodejních čísel a získávání nových zákazníků. Textová upozornění jsou jednoduchým způsobem, jak aktualizovat klienty o významných změnách ve sledovaných tématech a tím podpořit podnikání.

Push notifikace jsou dobré, protože umožňují aplikacím odesílat upozornění na mobilní zařízení uživatele bez nutnosti otevřít aplikaci. Tato funkce umožňuje aplikacím posílat uživatelům důležité informace, jako jsou zprávy,

upozornění na události nebo nabídky, kdykoli se tak stane, aniž by museli uživatelé ručně otevírat aplikaci. Push notifikace také umožňují aplikacím zvyšovat svou uživatelskou aktivitu a zvyšovat pravděpodobnost, že uživatelé budou opakovaně používat aplikaci. Navíc, push notifikace mohou být personalizovány tak, aby se cílily na konkrétní skupiny uživatelů nebo konkrétní uživatele, což pomáhá zvyšovat efektivitu a relevanci upozornění. [**2**] [**3**]

Chceme implementovat systém automatických notifikací pro pravidelnou komunikaci se zákazníky, abychom mohli efektivně informovat o aktuálním stavu jejich permanentky a dalších důležitých změnách týkajících se našich služeb. Tyto notifikace budou posílány na mobilní telefony zákazníků a budou obsahovat jednoduché aktualizace o stavu jejich účtu a dalších relevantních informacích. Tato strategie bude poskytovat výhody, jako je zlepšení zákaznické zkušenosti a podpora rozvoje našeho podnikání.

2.1.6 Platba

- neexistence platebni brany
- mesicni reporty a analityka

Současný proces platby, který se provádí bankovním převodem nebo v hotovosti, je neefektivní a plýtvá časem. Proces sledování platby, ručního zasílání platebních údajů každému zákazníkovi a neustálé upomínání o platbu, je časově náročný. Navíc, realizace bankovního převodu a zadání údajů bez chyby, vyžaduje od zákazníka čas a může se stát, že platba se ztratí. Zavedení plateb pomocí QR kódů pomohlo klientům, nicméně tento proces pro nás stále není dostatečně pohodlný. Účetnictví, které je vedeno v tabulce Excelu, není dostatečně účinný. Chceme mít možnost generovat měsíční reporty, které by nám poskytly přehled o našem finančním stavu a umožnily by nám lépe rozpoznat trend v našich tržbách.

2.2 Způsoby řešení současných problémů

2.2.1 Průzkum mezi cílovou skupinou

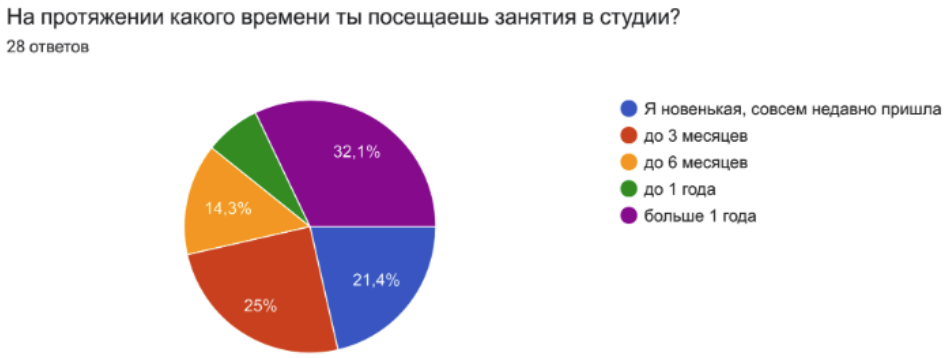
Byla provedena analýza preferencí klientů s cílem identifikovat potřeby a přání našich zákazníků v oblasti technologických řešení. Průzkum je založen na mé analýze problému tanečního studia a byl zaměřen na klienty našeho studia, kteří jsou většinou rusky mluvící, a měl za cíl získat informace o tom, jakým způsobem by mohly být současná řešení vylepšena a jaká řešení by klientům vyhovovala. Výsledky průzkumu byly popsány pod jednotlivými otázkami. Našeho průzkumu se zúčastnilo celkem **28** zákazníků

Dotaz:

- Jak dlouho se účastníte našich služeb?

Odpovědi:

- Nový zákazník
- 3 měsíce
- 6 měsíců
- Do 1 roku
- Více než 1 rok



Obrázek 2.1: Odpovědi na otázku č. 1

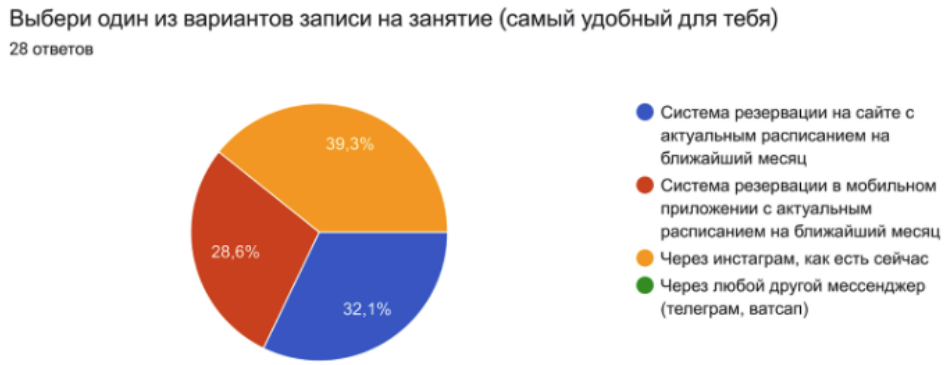
Pro uskutečnění objektivního průzkumu jsme zahrnuli do vzorku jak nové, tak i dlouhodobé zákazníky našeho studia. Tato strategie nám umožnila získat rozdílné perspektivy na základě délky zkušenosti zákazníků s našimi službami, což vede k vytvoření více komplexního pohledu na naše podnikání.

Dotaz:

- Výběr pohodlného způsobu pro rezervaci lekcí

Odpovědi:

- Rezervační systém na webových stránkách s aktuálním rozvrhem pro následující měsíc.
- Rezervační systém v mobilní aplikaci.
- Ponechání současného způsobu rezervace přes Instagram.
- Rezervace pomocí jiných messengerů.



Obrázek 2.2: Odpovědi na otázku č. 2

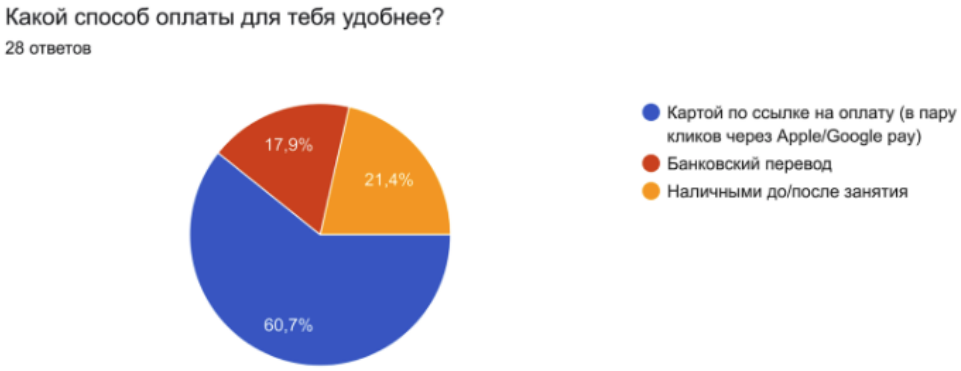
Výsledky průzkumu 2.2 ukazují, že 1/3 hlasujících se s aktuálním způsobem rezervace lekcí přes Instagram cítí pohodlně, zatímco zbytek dotázaných (2/3) preferuje rezervační systém na webových stránkách nebo v mobilní aplikaci s aktuálním rozvrhem na nadcházející měsíc. Tyto výsledky naznačují, že většina dotázaných není se současným způsobem rezervace lekcí spokojena, což by mohlo vést k odchodu klientů ke konkurenci se snazším způsobem rezervace.

Dotaz:

- Který způsob platby je výhodnější?

Odpovědi:

- Pomocí platební brány (Apple/Google Pay nebo kartou)
- Bankovní převod
- Hotovosti před/po lekce
- Rezervace pomocí jiných messengerů.



Obrázek 2.3: Odpovědi na otázku č. 3

Výsledky průzkumu 2.3 o způsobech platby za lekce ukázaly, že většina dotázaných považuje platbu přes platební bránu kartou nebo pomocí Apple/-Google pay za nejpohodlnější způsob platby. Přibližně polovina dotázaných

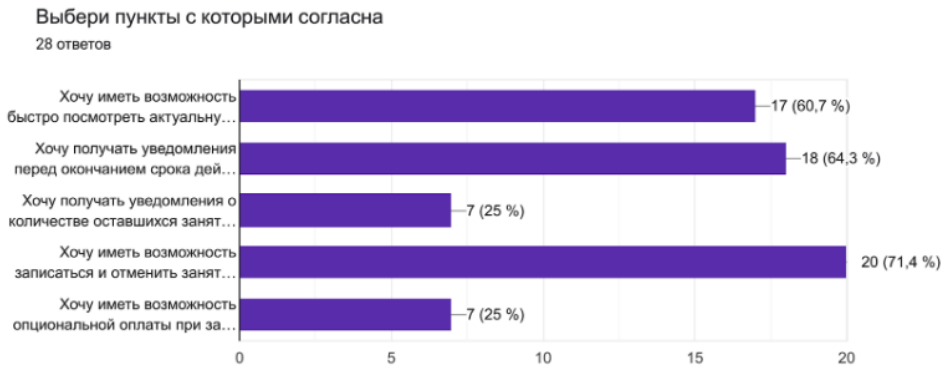
zvolila tuto možnost, zatímco pouze 21% dotázaných platí v hotovosti. Tyto výsledky naznačují, že naši klienti potřebují řešení pro platby kartou prostřednictvím připojení terminálu nebo služby, která umožňuje integraci platební brány.

Dotaz:

- Vyberte body, se kterými souhlasíte

Odpovědi:

- Chci mít možnost rychle zobrazit aktuální informace o předplatném.
- Chci být informován před vypršením předplatného
- Chci být informován/a o stavu predplatneho na konci každé lekce
- Chci mít možnost rezervovat a rušit lekce prostřednictvím webové stránky/aplikace.
- Chci mít možnost vybrat si různé typy platby (hotovost/karta/bankovni převod)



Obrázek 2.4: Odpovědi na otázku č. 4

Průzkum ukázal 2.4, že většina dotázaných (71,4%) preferuje možnost samostatně rezervovat a rušit lekce a rychle zobrazit aktuální informace o svém předplatném. Avšak, naše současná technická infrastruktura neumožňuje poskytnutí této možnosti. Navíc, většina dotázaných (mnoho lidí) také požaduje být upozorněni na vypršení jejich předplatného, avšak v důsledku nedostatku automatizovaného systému, musíme to dělat ručně.

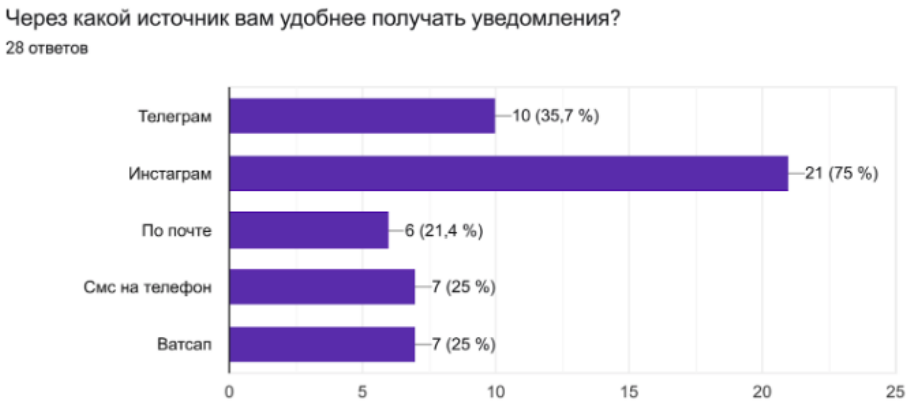
Dotaz:

- Z jakého zdroje vám vyhovuje přijímat oznámení?

Odpovědi:

- E-mailem

- Push oznámením v mobilní aplikaci
- Jiný zdroj
- Přes sociální sítě
- SMS zprávou



Obrázek 2.5: Odpovědi na otázku č. 5

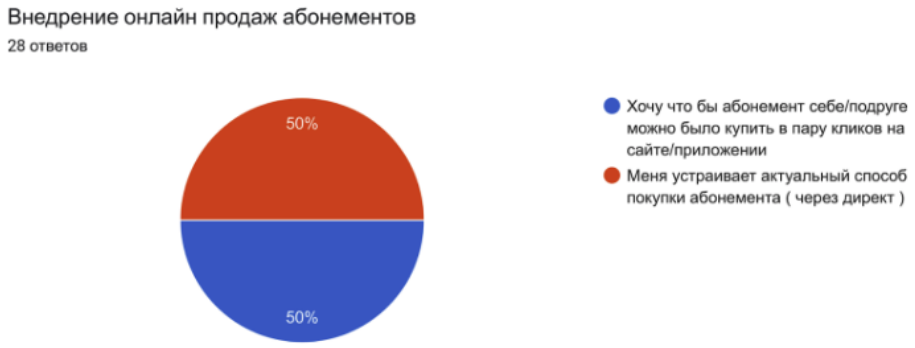
Náš průzkum zjistil 2.5, jaký zdroj oznámení preferují naši klienti pro přijímání informací o svém předplatném, rezervacích lekcí a dalších důležitých informacích týkajících se našeho studia. Výsledky nám umožní lépe přizpůsobit naše komunikační kanály potřebám našich klientů.

Dotaz:

- Jakým způsobem byste preferoval/a prodej permanentek?

Odpovědi:

- Online prostřednictvím webové stránky nebo mobilní aplikace
- Současný způsob prodeje



Obrázek 2.6: Odpovědi na otázku č. 6

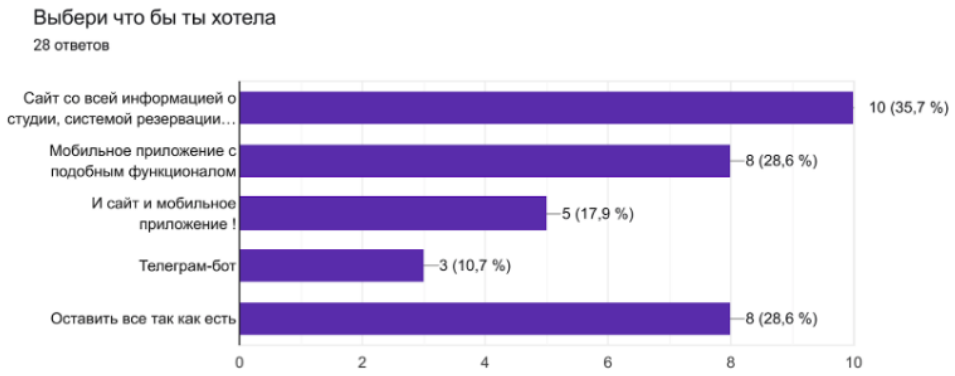
Výsledky průzkumu ukázaly 2.6, že polovina dotázaných by preferovala možnost nákupu permanentky prostřednictvím online kanálů, jako jsou webové stránky nebo mobilní aplikace. Z tohoto vyplývá, že tato skupina dotázaných hledá pohodlný a rychlý způsob nákupu. Zbývající polovina dotázaných se zdá být spokojena s současným způsobem nákupu, tedy osobním nákupem v hotovosti.

Dotaz:

- Zde jsou různá technická řešení, vyberte si ta, která se vám líbí a která byste rádi viděli v našem studiu.

Odpovědi:

- Zavedení webových stránek s informacemi o studiu, rezervačním systémem a osobním účtem
- Mobilní aplikace s podobnou funkčností
- Webové stránky spolu s mobilní aplikací
- Chatbot na sociálních sítích
- Nechci změnu, současný způsob mi vyhovuje.



Obrázek 2.7: Odpovědi na otázku č. 7

Pouze 29% dotázaných 2.7 chce ponechat věci tak, jak jsou, což opravdu není malé procento a znamená to, že odvádíme dobrou práci a snažíme se poskytovat dobré služby, přestože vše děláme ručně bez jakéhokoli technického řešení. Pro 71% dotázaných je důležité, aby byly k dispozici webové stránky se všemi informacemi o studiu, rezervačním systémem a osobním účtem, což naznačuje potřebu vytvořit technické řešení pro zlepšení efektivity a pohodlí pro klienty.

2.2.2 Určení strategie řešení

Výsledky tohoto průzkum potvrzují mé předpoklady a ukazují, že existuje potřeba implementovat technologická řešení pro zlepšení rezervačního systému a platebních metod. Z těchto výsledků vyplývá, že by bylo vhodné prozkoumat již existující technologická řešení, která by splňovala požadavky dotázaných a pokud by žádné vhodné řešení nebylo k dispozici, bylo by vhodné zvážit vytvoření vlastního.

Kapitola 3

Rešerše stávajících řešení

Na trhu je k dispozici mnoho online rezervačních softwarů, které nabízejí podobnou sadu nástrojů. Po dohodě s vedoucím v rámci mého výzkumu se zaměřím na analýzu tří nejpopulárnějších řešení dostupných na trhu. Tato řešení budou podrobena detailní analýze na základě stanovených kritérií a požadavků pro naše studio.

Při hledání existujících řešení jsem se spoléhal na dvě klíčové strategie. Za prvé, vycházel jsem z doporučení Google na základě dotazů jako je "**online booking and appointment scheduling**". Za druhé, bral jsem v úvahu recenze na webové stránce *https://www.getapp.com/*, která je jedním z předních online zdrojů informací o softwarových aplikacích pro podniky.

Během analýzy se budeme zaměřovat na klíčové funkce a vlastnosti těchto rezervačních softwarů, abychom zjistili, zda splňují stanovená kritéria a požadavky našeho studia. Prozkoumáme jejich rozhraní, možnosti správy rezervací a administrace, integraci s dalšími systémy, dostupnost mobilních a webových aplikací, podporu platebních systémů, schopnost ukládání údajů o zákaznících a možnost posílání notifikací.

V rámci analýzy budeme také zkoumat výhody a nevýhody jednotlivých softwarů, jejich cenovou politiku a dostupnou technickou podporu. Budeme se snažit získat informace o reference a recenze od uživatelů, kteří již tyto softwary používají, abychom získali ucelený obraz o jejich kvalitě a spolehlivosti.

Na základě této analýzy budeme schopni najit nejvhodnější rezervační software pro naše studio, který splňuje naše specifické požadavky a kritéria.

3.1 Kritéria výběru softwaru

Při definování požadavků na hledaný systém jsem stanovil několik kritérií, která musí být splněna.

Seznam požadavků zahrnuje následující body:

- Cena: Maximální měsíční náklady na systém nemusí přesáhnout částku 15 USD.

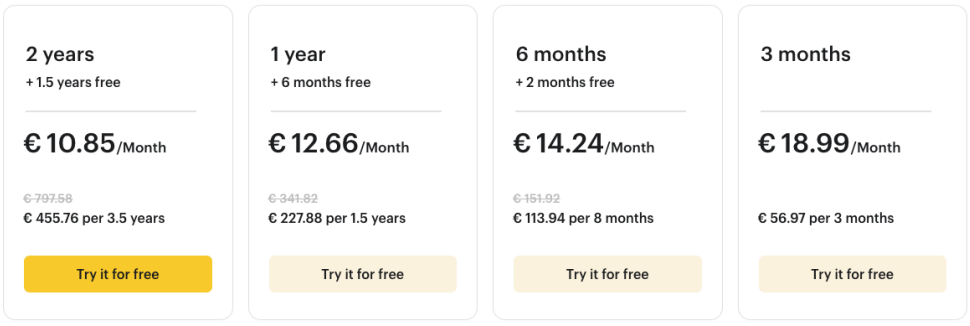
- Rezervační systém: Systém musí obsahovat funkci rezervace pro naše klienty.
- Administrátorský panel: Musí být k dispozici rozhraní pro správu podniku a administrátorské úkony.
- Otevřené API: Systém musí poskytovat otevřené a dobře dokumentované rozhraní API, které umožní integraci s dalšími systémy.
- Klientská aplikace: Systém musí poskytovat mobilní a webovou aplikaci pro naše klienty.
- Integrace platebního systému: Systém musí umožňovat připojení a integraci s platebním systémem pro pohodlnou a bezproblémovou platbu.
- Správa zákaznických dat: Systém musí umožňovat ukládání a správu údajů o našich zákaznících.
- Věrnostní systém: Systém musí poskytovat věrnostní program, alespoň ve formě možnosti nastavit předplatné pro naše zákazníky.
- Vícejazyčnost: Systém musí podporovat vícejazyčné rozhraní pro komunikaci se zákazníky v jejich preferovaném jazyce.

3.2 Altegio

Altegio je technologická platforma, která poskytuje automatizaci podnikových procesů pro společnosti poskytující služby. Jejím základem je technologická základna vytvořená vývojáři s desetiletými zkušenostmi v tvorbě vysoce kvalitních digitálních produktů.

1. Cena.

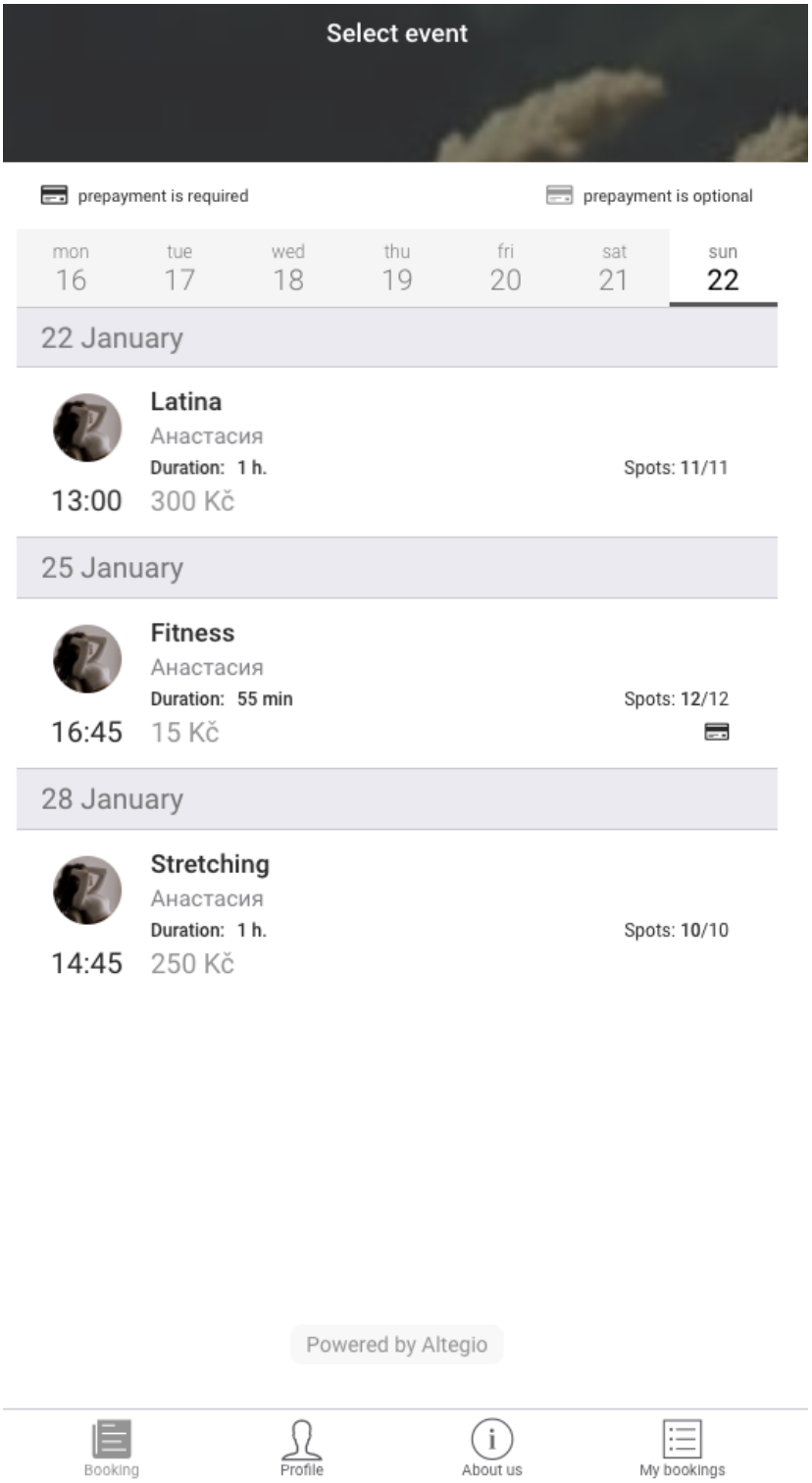
   Cena tohoto softwaru je rozumná a kolísá mezi 10 a 19 dolary za měsíc, v závislosti na délce smlouvy.



Obrázek 3.1: Ceny z oficiálních stránek [5]

2. Rezervační systém.

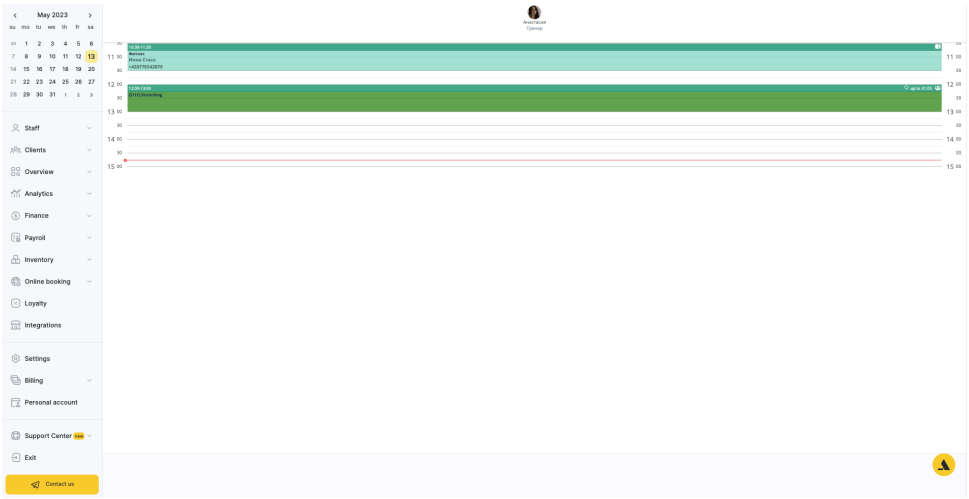
Altegio nabízí vlastní rezervační systém, kde si každý klient může prohlédnout rozvrh a spravovat své rezervace. Design kalendáře není nejmodernější 3.2, má trochu zastaralý vzhled. Navíc, kalendář je poskytovaný pod doménou a značkou Altegio, bez možnosti přidání vlastní domény a odstranění jejich značky.



Obrázek 3.2: Online widget rezervačního systemu Altegio

3. **Administrátorský panel.**
   Jedním z nejvýraznějších a klíčových předností firmy Altegio je její

administratorský panel 3.3, který nabízí širokou škálu funkcí pro řízení vašeho podnikání. Altegio disponuje vlastní databází znalostí, která vám poskytuje návod k optimálnímu a efektivnímu využití jejich systému, což vám umožňuje nejen spravovat všechny své obchodní operace, ale také sledovat jejich výkonnost.



Obrázek 3.3: Administrátorský panel systemu Altegio

4. **Otevřené API.**

   Jedním z klíčových přínosů systému Altegio je jeho otevřené API rozhraní s přehlednou dokumentací, které umožňuje výměnu dat s automatizačním a účetním softwarem, a také integraci online rezervací do webových stránek a mobilních aplikací třetích stran.

   API pro integraci s platformou Altegio se skládá ze dvou kategorií metod:

   Metody, které vyžadují autorizaci uživatele a partnera. Metody, které nepotřebují autorizaci uživatele, ale vyžadují autorizaci partnera. Pro přístup k oběma skupinám metod je nutná autorizace partnera, což zahrnuje předání jedinečného hash klíče partnerovi. Autentizace API požadavků je provedena v souladu s RFC 6749 "Resource Owner Password Credentials Grant".

   Veškerá komunikace protokolu je přenášena pomocí šifrování TLS (od verze 1.2). Existují limity, které činí 200 požadavků za minutu nebo 5 požadavků za sekundu na jednu IP adresu.

   Dokumentace je k dispozici na adrese : `https://developer.alteg.io/api#section/Altegio-API-Concepts`

5. **Klientská aplikace.**

   Systém Altegio ve své základní nabídce zahrnuje webový widget pro online rezervace zákazníků, který je možné integrace do existujících webových stránek bez dalších nákladů.

V kontextu možnosti vytvoření mobilní verze aplikace pro online rezervace bylo zjištěno, po konzultaci se zákaznickou podporou Altegia, že taková služba je dostupná, avšak za jednorázový poplatek ve výši 1000 EURO. Z tohoto důvodu lze konstatovat, že Altegio neobsahuje standardně možnost vytvoření mobilní aplikace pro online rezervace jako součást své základní nabídky, ale poskytuje pouze webový widget.

6. **Integrace platebního systému.**

   Podle informací z oficiální dokumentace , system Altegio ma realizovanou integrace se systémem Stripe. Nicméně, při testování bylo zjištěno, že tato integrace je stále ve vývojové fázi a neposkytuje aktivní funkcionalitu prostřednictvím online widgetu.

7. **Správa zákaznických dat.**

   Systém Altegio ukládat a spravovat databázi všech zákazníků, kteří provedli rezervaci.

   Díky tomu můžeme prostřednictvím notifikací komunikovat se svými stálými zákazníky a informovat je o různých aspektech našich služeb. Tyto notifikace jsou zasílány prostřednictvím e-mailu a zahrnují upozornění na nové služby a slevy, informace o rezervacích lekcí uskutečněných pomocí online rezervačního widgetu, oznámení o nadcházejících lekcích a možnost požadovat zpětnou vazbu po návštěvě. Systém rovněž umožňuje informování personálu o nových termínech a o změnách v naplánovaných termínech, například jejich odložení nebo zrušení.

   Altegio nabízí také možnost integrace s více než 25 různými SMS agregátory, což nám umožňuje zasílat zákazníkům oznámení prostřednictvím SMS zpráv nebo messengeru.

8. **Věrnostní systém.**

   Systém Altegio nabízí možnost vytváření předplatného pro naše klienty, umožňuje výběr služeb, které jsou jeho součástí, a automaticky je z účtu odebírá po poskytnutí vyučování. Ačkoli nám tento systém umožňuje konfigurovat předplatné individuálně pro každého klienta prostřednictvím administrátorského panelu, chybí možnost nákupu těchto předplatných přímo v systému. Tato omezení nás nutí vést ruční evidenci, i když bychom preferovali plnou integraci těchto funkcí do systému Altegio.

   Systém Altegio rovněž nabízí možnosti pro tvorbu různých propagačních akcí, slevových kupónů a dalších věrnostních aktivit. Tímto způsobem můžeme efektivněji oslovit naše zákazníky.

9. **Vícejazyčnost.**

   Aktuálně jse widget pro online rezervace na webových stránkách lokalizován do široké škály jazyků, včetně klíčových jazyků pro naše studio, jako je ruština, angličtina a čeština.

Altegio nabízí široké spektrum funkcí, včetně online widgetu **??** pro rezervace, mobilní aplikace, finančního a skladového účetnictví, podrobné analýzy, věrnostního programu, zasílání SMS a e-mailových zpráv zákazníkům a zaměstnancům, IP telefonie a více než 60 integrací s užitečnými podnikovými aplikacemi. Platforma také poskytuje otevřené rozhraní API s přehlednou dokumentací a rozsáhlou znalostní základnu pro snadnou integraci do podnikání.

Hlavní nevýhody:

V průběhu analýzy bylo zjištěno, že software Altegio má několik nevýhod. Uživatelé se stěžují na nespolehlivost software, když většina funkcí nefunguje správně. Dále uživatelé kritizovali omezenou funkčnost systému, například není možné nastavit volitelnou platbu při rezervaci. Tyto chyby se zdají být řešeny technickou podporou velmi dlouho. Uživatelé také kritizují, že uživatelské rozhraní není uživatelsky přívětivé a srozumitelné, což může ztěžovat používání software. Tyto nedostatky jsou potvrzeny i v mnoha špatných recenzích.

3.3 SimplyBook

SimplyBook je cloudový rezervační a plánovací systém, který umožňuje malým a středním podnikům automatizovat své procesy spojené s rezervacemi a plánováním schůzek. Systém je vhodný pro různé druhy služeb, jako jsou salony, lázně, fitness centra a další.

1. **Cena.**

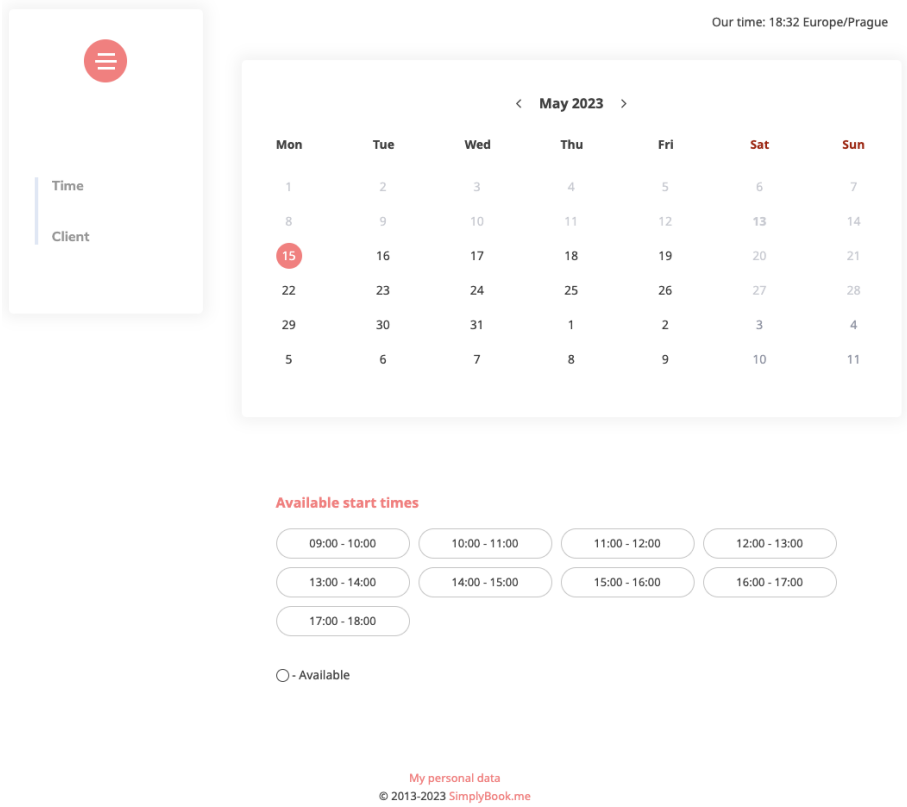
   Pro využití alespoň poloviny funkcí, které software SymplyBook nabízí, je nutné investovat 30 eur měsíčně nebo 300 eur ročně (což představuje 17% slevu při ročním předplatném). Tyto finanční nároky přesahují naše aktuálně dostupné rozpočtové možnosti.

Annually / Monthly You save 17%	Free €0 /month	Basic €9.9 /month Save 17% annually	Popular Standard €29.9 /month Save 17% annually	Premium €59.9 /month Save 17% annually	Premium plus €99.9 /month Save 17% annually
Included bookings	50	100	500	2000	5000
Custom Features	1	3	8	unlimited	unlimited
Users/Providers*	1	5	15	30	60
Admin App	✓	✓	✓	✓	✓
Client App	×	Branded	Branded	Branded	Branded
Booking Website	✓	✓	✓	✓	✓
Booking Widget	✓	✓	✓	✓	✓
Directory Listing	×	✓	✓	✓	✓
Coupons & Gift Cards	×	✓	✓	✓	✓
Sales (POS)	×	✓	✓	✓	✓
Link Removal	×	×	×	✓	✓
HIPAA	×	×	✓	✓	✓

Obrázek 3.4: Ceny z oficiálních stránek SimplyBook.me [**6**]

2. **Rezervační systém.**

V porovnání s platformou Altegio 3.2, SimplyBook nabízí širší spektrum přizpůsobení webových stránek určených pro rezervace. Administrátorské rozhraní poskytuje mnoho možností pro úpravy designu. Je k dispozici výběr z více než 15 různých šablon, možnost přizpůsobení barev tak, aby odpovídaly naší firemní identitě, a také prostor pro vložení loga naší společnosti. Kromě toho SimplyBook umožňuje změnu názvu domény na vlastní. V konečném důsledku, SimplyBook nabízí v oblasti přizpůsobení větší flexibilitu než Altegio, což je nesporně jeho výhoda.



Obrázek 3.5: Rezervační webová stránka SimplyBook

3. **Administrátorský panel.**

Analogicky k Altegiu, i SimplyBook disponuje přehledným a komplexním administrátorským panelem, který nabízí možnost efektivního řízení všech obchodních procesů naší firmy.

4. **Otevřené API.**

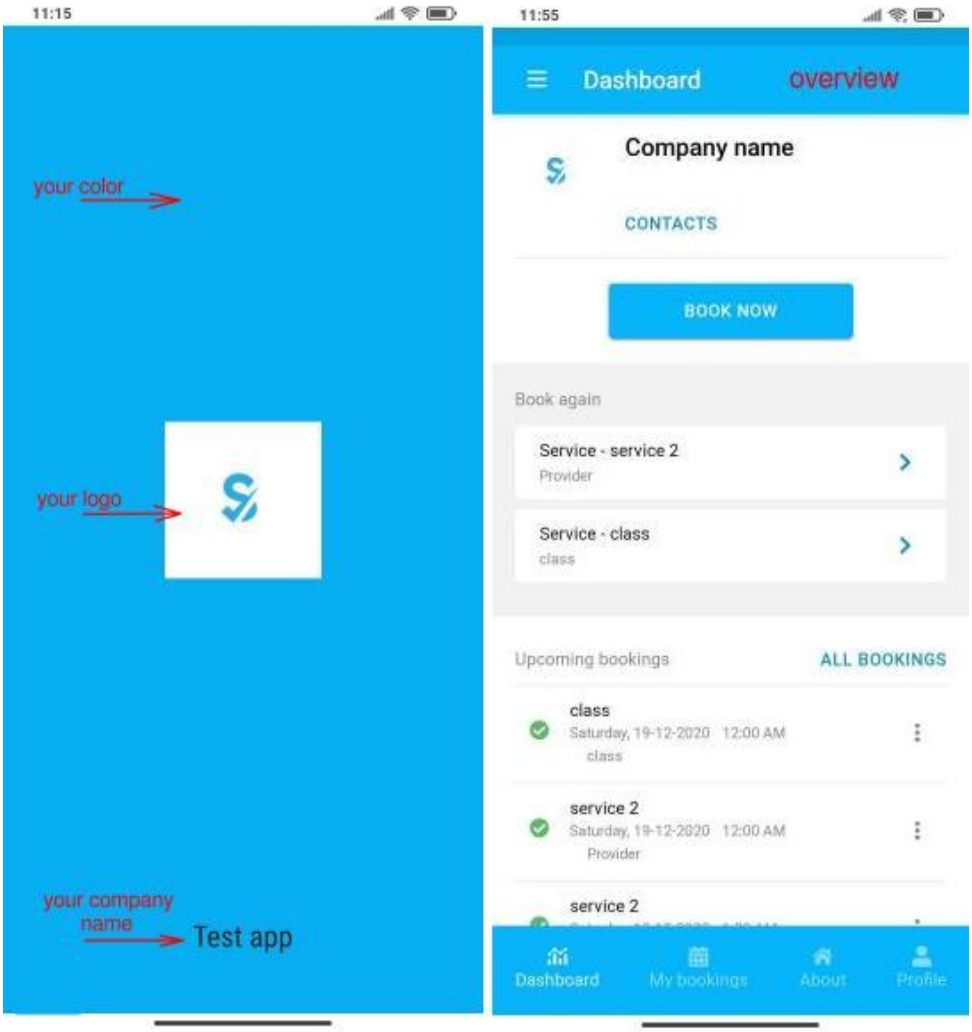
SimplyBook disponuje vlastním API, které poskytuje široké spektrum metod a funkcí. Tyto funkce umožňují spravovat rezervace, získávat informace o klientech, manipulovat s kalendářem a provádět mnoho dalších operací. To vše přispívá k možnosti vytvořit vlastní uživatelské

rozhraní, které je v souladu s potřebami a značkovou strategií konkrétního podniku.

API Simplybook používá protokol JSON-RPC 2.0.

5. **Klientská aplikace**

SimplyBook nabízí možnost použití mobilní aplikace pro klienty. Nicméně, k přizpůsobení této aplikace je nutné předplatit prémiový balíček, který stojí 50 USD. Je důležité poznamenat, že tato aplikace je v současné době v beta verzi a její funkce jsou značně omezené.Dále je nutné zmínit, že uživatelské rozhraní aplikace může působit neintuitivně a zastarale.



Obrázek 3.6: Mobilní klientská aplikace SimplyBook

6. **Integrace platebního systému.**

SimplyBook poskytuje možnost realizace plateb předem v momentě rezervace, což eliminuje nutnost řešení finančních transakcí po ukončení

služby. Tento systém nám nabízí integraci s více než třemi desítkami různých platebních systémů, což zvyšuje jeho flexibilitu.

Obrázek 3.7: Platební systémy, se kterými se SimplyBook může integrovat

7. **Správa zákaznických dat.**

   Zaznamenávání zákaznických dat je v rámci SimplyBook poskytováno bez jakýchkoli dodatečných nákladů. Systém také nabízí možnost importu stávajícího seznamu klientů (až 500 záznamů) ve formátu .csv, což představuje významné usnadnění procesu přenosu dat. SimplyBook tedy plně zvládá správu a evidenci zákaznických údajů, což jej činí užitečným nástrojem pro naše potřeby.

   E-mailová upozornění jsou standardní formou komunikace v rámci všech předplatných plánů SimplyBook.me. Pro rozšíření komunikace o SMS notifikace je nutné zaplatit dodatečný poplatek. Tyto SMS zprávy jsou účtovány individuálně, což je ekonomicky efektivní, neboť je účtován pouze skutečný počet odeslaných zpráv přes SMS bránu.

8. **Věrnostní systém.**

Tato uživatelská funkce umožňuje, aby určité služby byly k dispozici pouze pro členy s aktivním placeným členstvím. Tato funkce může být aplikována na vybrané služby nebo na všechny služby, pouze na lekce atd.

9. **Vícejazyčnost.**

   SimplyBook nabízí omezený výběr jazykových verzí, konkrétně pouze francouzskou a anglickou. Toto představuje významnou nevýhodu pro naše podnikání, jelikož naši hlavní zákazníci jsou rusky mluvící. Tento jazykový omezený výběr může potenciálně omezit naši schopnost efektivně komunikovat a poskytovat služby našim cílovým zákazníkům.

Hlavní nevýhody:

Systém SimplyBook.me byl hodnocen jako nedostatečně přehledný a matoucí pro ne-technické uživatele, což je způsobeno nedostatečně vyspělou architekturou a procesem. Navzdory tomu byla hodnocena technická podpora jako efektivní, s rychlým řešením problémů a snahou o rozvoj a doplňování nástrojů.

3.4 Acuity Scheduling

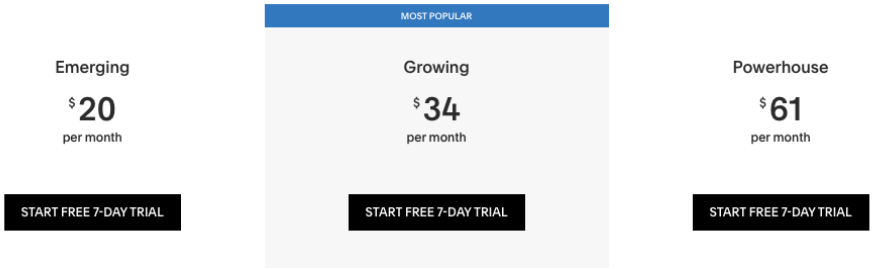
Hlavní funkce programu Acuity Scheduling jsou velmi podobné těm, které nabízí předchozí dvě řešení 3.2 3.3. Abysme však nepropadli opakování stejných informací, shrnu důvody, proč toto řešení nevyhovuje našim potřebám:

- Nejnižší cena za použití Acuity Scheduling je stanovena na 20 dolarů. Pokud bychom ale chtěli prodat předplatné, cena se zvyšuje na 34 dolarů. To je dvojnásobek ceny v porovnání s Altegio.

- Jazyková podpora je omezena pouze na angličtinu, což může představovat komunikační bariéru pro některé uživatele.

- Integrace s dalšími systémy je značně komplikovaná. To může představovat významnou překážku, pokud se budeme potřebovat spojit s jinými technologickými řešeními.

- Někteří uživatelé si stěžovali na neuspokojivé uživatelské rozhraní mobilní aplikace. Kromě toho jsou nejpokročilejší funkce dostupné pouze v rámci placeného tarifu.



Obrázek 3.8: Ceny z oficiálních stránek [7]

Ve srovnání s jinými řešeními je tak Acuity Scheduling ve své podstatě nedostatečný vzhledem k našim specifickým potřebám a očekáváním. Jeho omezené možnosti a vyšší náklady ho činí méně vhodným pro naše účely.

3.5 Závěr

Vytvořil jsem tabulku 3.1, ve které jsem každý kritérium 3.1 hodnotil na stupnici od 0 do 5, kde **0** znamená, že daný software plně odpovídá našim požadavkům, zatímco **5** znamená, že daný software nesplňuje požadované kritérium.

Tento systém hodnocení umožňuje kvantitativní srovnání různých softwarových řešení podle klíčových kritérií, což usnadňuje rozhodování o tom, který software je pro naše potřeby nejvhodnější. Celkově platí, že čím nižší je celkové skóre, tím lépe daný software vyhovuje našim požadavkům.

	Altegio	SimplyBook	AcuityScheduling
Cena	0	5	5
Rezervační systém	0	0	0
Administrátorský panel	0	1	1
Otevřené API	1	1	5
Klientská aplikace	5	3	3
Integrace platebního systému	3	0	0
Správa zákaznických dat	0	0	0
Věrnostní systém	0	2	5
Vícejazyčnost	0	4	5
Celkem	9	16	24

Tabulka 3.1: Srovnání řešení

Z tabulky je zřejmé, že z tří analyzovaných řešení je pro naše potřeby nejlepší volbou Altegio. Je cenově dostupné a nabízí téměř vše, co potřebujeme pro administraci a řízení naší taneční školy.

Hlavními problémy, s nimiž se setkáváme při výběru softwaru Altegio, jsou vysoké náklady na jejich klientskou mobilní aplikaci, která stojí 1000 eur, a také skutečnost, že není nastavena integrace s platebním systémem. Avšak vzhledem k tomu, že Altegio nám poskytuje přístup k jejich API,

bylo rozhodnuto v rámci této bakalářské práce vyvinout hybridní klientskou aplikaci, která bude fungovat na různých platformách, včetně iOS, Androidu a dokonce i webu. Altegio poskytuje otevřené API s dokumentací, což nám umožňuje provádět a nastavovat všechny administrativní procesy v administračním panelu Altegio a přenášet je do naší aplikace. Toto řešení je velmi výhodné, protože nás činí velmi flexibilními pro naše klienty a potenciálně nám umožňuje integrovat naši aplikaci s dalšími systémy, aby byla co nejvíce uživatelsky přívětivá.

Tímto se uzavírá teoretická část práce. V následující části se budu věnovat analýze a návrhu hybridní aplikace, která bude integrována se zvoleným systémem Altegio, jež představuje jádro našeho řešení.

Část II

Praktická část

Kapitola 4

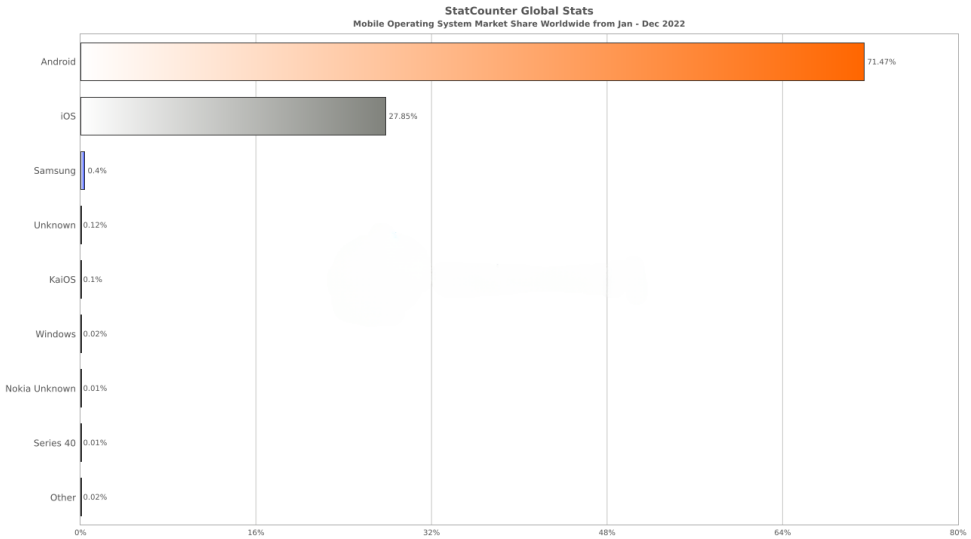
Analýza vlastního řešení

V následující kapitole se obecně zaměřím na aplikace, konkrétně na výhody vývoje hybridních aplikací. Dále provedeme analýzu různých frameworků, které jsou k dispozici pro vývoj hybridních aplikací, a následně vybereme ten, který je pro nás nejvhodnější.

4.1 Obecné informace k vývoji hybridních aplikací

4.1.1 Podíl na trhu mobilních operačních systémů

Celosvětový trh s mobilními operačními systémy je v současnosti ovládán dvěma dominantními subjekty, a to Googlem s jeho systémem Android a společností Apple s operačním systémem iOS. K roku 2021 Android ovládá přibližně 71,5% trhu, zatímco iOS drží podíl 27,8%. Ostatní mobilní operační systémy, včetně Windows Phone, Nokie a Samsungu, tvoří zbylých přibližně 1% trhu a jejich popularita je na poklesu.Tato analýza 4.1 naznačuje, že naše primární zaměření by mělo být na operační systémy iOS a Android.



Obrázek 4.1: Podíl mobilních operačních systémů na globalnim trhu za rok 2022 [4]

4.1.2 Typologie mobilních aplikací

Následující text vychází z [**10**]

1. **Nativní aplikace** jsou vytvářeny specificky pro určitý operační systém - Android nebo iOS - pomocí jazyků jako Java/Kotlin pro Android a Swift/Objective-C pro iOS. Nativní aplikace nabízejí vysokou úroveň výkonu a optimální využití zdrojů zařízení. Nicméně, vývoj a údržba těchto aplikací může být náročný a finančně nákladný, neboť každá platforma vyžaduje svůj specifický kód.

2. **Webové aplikace** jsou v zásadě webové stránky optimalizované pro mobilní zařízení. Jsou přístupné prostřednictvím webového prohlížeče a nevyžadují instalaci do zařízení. Tento typ aplikace je jednodušší na vývoj a údržbu, ale nemusí poskytnout tak plynulý uživatelský zážitek a bohaté funkce jako nativní aplikace. Webové aplikace jsou obvykle vyvíjeny pomocí technologií jako HTML, CSS a JavaScript a mohou být vytvářeny pomocí různých knihoven a frameworků, například React, Angular nebo Vue.js.

3. **Hybridní aplikace** kombinují prvky nativních a webových aplikací. Jsou vyvíjeny pomocí webových technologií a poté jsou zabalené do nativního kontejneru pro instalaci na zařízení. Hybridní aplikace mohou využívat nativní funkce zařízení a být distribuovány přes obchody s aplikacemi stejně jako nativní aplikace. Obvykle jsou vyvíjeny pomocí webových technologií a poté zabalené do nativního kontejneru pomocí nástrojů jako je Cordova nebo Ionic. Novější řešení, jako je React Native od Facebooku a Flutter od Googlu, umožňují vývojářům psát kód v jazyce JavaScript (React Native) nebo Dart (Flutter) a kompilovat jej do nativního kódu pro každou platformu, přičemž poskytují výkon a funkčnost srovnatelnou s nativními aplikacemi.

4.1.3 Přínosy a motivace pro vývoj hybridních aplikací

Vývoj hybridních aplikací přináší řadu výhod, které je činí atraktivní volbou pro mnoho organizací a vývojářů:

- Rychlost vývoje: Jelikož hybridní aplikace využívají stejný kód pro všechny platformy, jejich vývoj je obecně rychlejší než vývoj samostatných nativních aplikací pro každou platformu.

- Snížení nákladů: Vývoj a údržba jediné kódové základny, na rozdíl od několika samostatných, může výrazně snížit náklady na vývoj a údržbu aplikací.

- Zjednodušené aktualizace: Jelikož většina obsahu hybridní aplikace je webová, může být snadno a rychle aktualizována bez nutnosti uživatelů stahovat aktualizace z obchodu s aplikacemi.

- Přístup k nativním funkcím: Díky speciálním API rozhraním mohou hybridní aplikace využívat nativní funkce zařízení, jako je fotoaparát nebo akcelerometr, což bylo tradičně doménou nativních aplikací.

[10]

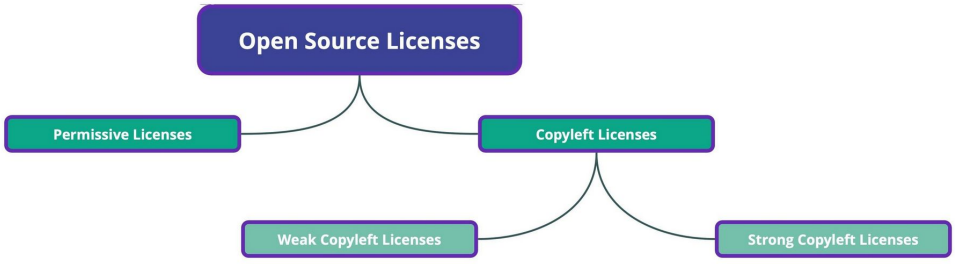
4.2 Výběr vhodného frameworku pro vývoj hybridní aplikace

V této kapitole provedu analýzu nejpopulárnějších frameworků pro vývoj hybridních aplikací, se zaměřením na jejich licenční podmínky, podporované platformy a hlavní výhody a nevýhody. Tato analýza nám pomůže lépe pochopit možnosti, které máme k dispozici, a informovaně se rozhodnout, který framework je nejvhodnější pro náš konkrétní projekt.

Při tvorbě následující analyzy bylo čerpáno z [14] [15] [16]

4.2.1 Rozdíly v licencování open-source software

V rámci open-source licencování se obecně rozlišují dvě základní kategorie: copyleftové a permisivní licence (viz. 4.2). Každá z nich představuje specifický přístup k řízení práv a povinností spojených s použitím, modifikací a distribucí softwaru.



Obrázek 4.2: Kategorie Open-Source licencí [11]

- **Copyleftové licence:** Tato kategorie licencí umožňuje bezplatné šíření a modifikaci původního díla, avšak s podmínkou, že všechny odvozené produkty musí být licencovány za stejných podmínek. Cílem copyleftové licence je zachovat svobodnou dostupnost daného díla pro budoucí využití, úpravy a distribuci. Mezi typické příklady copyleftových open-source licencí patří AGPL, GPL, LGPL, EPL a Mozilla, seřazené od nejvíce k nejméně restriktivním. [11] [12]
- **Permisivní licence:** Na rozdíl od copyleftových licencí, permisivní licence poskytují uživatelům výrazně větší flexibilitu v oblasti použití a distribuce softwaru. Permisivní licence, jako jsou například MIT nebo BSD, kladou na použití, modifikaci a distribuci softwaru minimální

omezení. U permisivních licencí je uživatelé mohou používat pro jakékoli účely, včetně komerčních, a nejsou povinni zveřejňovat zdrojový kód ani licencovat své modifikace za stejných podmínek. Mezi nejrozšířenější permisivní open-source licence patří Apache, MIT, BSD a Unlicense. [**11**] [**12**]

Open-Source Software License Types

Copyleft Licenses	Permissive Licenses
GNU General Public License (GPL)	Apache License
GNU LGPLv3.0 (Lesser General Public License)	BSD License
Mozilla Public License	MIT License
Eclipse Public License	Unlicense

Obrázek 4.3: Populární licence open-source software: Přehled nejčastěji používaných typů [**11**]

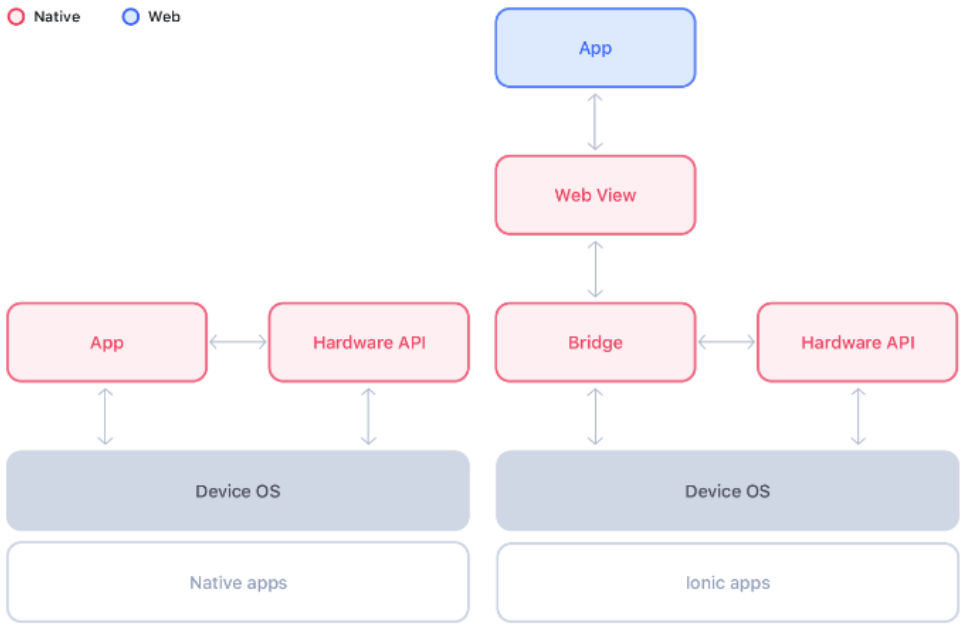
4.2.2 Ionic

Ionic je otevřený nástroj pro vývoj uživatelských rozhraní, který umožňuje vytvářet progresivní webove aplikace (PWA) a take výkonné a kvalitní mobilní aplikace s využitím webových technologií - HTML, CSS a JavaScript - s integracemi pro populární frameworky jako Angular, React a Vue.

Framework Ionic zahrnuje komponenty uživatelského rozhraní, knihovny pro vývoj front-endu a další designové prvky pro vytvoření mobilních aplikací pro Android a iOS.

Aplikace Ionic jsou vytvářené s využitím webových technologií a zobrazují se pomocí tzv. Web Views. 4.4

Moderní Web Views nabízejí mnoho vestavěných HTML5 API pro hardwarové funkce, jako jsou kamery, senzory, GPS, reproduktory a Bluetooth, ale někdy může být také nutné přistupovat k hardwarovým API specifickým pro danou platformu. V aplikacích Ionic lze k hardwarovým API přistupovat prostřednictvím bridge vrstvy, obvykle s využitím nativních pluginů, které poskytují JavaScriptová API. [**13**]



Obrázek 4.4: Architektura Web View [**13**]

Ionic CLI (Command Line Interface) je nástroj, který usnadňuje vývoj hybridních aplikací postavených na TypeScriptu a Node.js. Pomáhá při vytváření hybridních mobilních aplikací a zajišťuje pravidelné aktualizace. Ionic CLI nabízí vestavěný vývojový server, nástroje pro ladění a další vývojářské nástroje.

- **Hlavní výhoda:** Ionic využívá webové technologie (HTML, CSS, JavaScript), což usnadňuje vývoj se zkušenostmi s webovým vývojem. Díky tomu Ionic umožňuje rychle vytvářet prototypy a snadno udržovat aplikace.
- **Hlavní nevýhoda:** Vzhledem k tomu, že Ionic používá WebView pro zobrazování aplikací, výkon může být nižší než u nativních aplikací, zejména při složitých animacích a interakcích uživatelů.

4.2.3 React Native

Při tvorbě následujícího textu bylo čerpáno z

React Native, původně vyvinutý jako platforma pro vývoj webových rozhraní v roce 2013, byl oficiálně uveden na trh pro vývoj hybridních mobilních aplikací v roce 2015 společností Facebook. Mnoho lidí srovnává React Native s Ionic, jelikož je považují za přímé alternativy.

React Native využívá JavaScript pro kompilaci uživatelského rozhraní aplikace, ale zobrazuje jej pomocí native-OS views. Například když v JSX používáme komponentu View, v našem JavaScriptovém kódu využíváme UIView z iOS. Podobně pro komponentu Image používáme komponentu UIImage z iOS, takže všechny komponenty, které používáme v React Native,

mají nativní deklaraci. Pro složitější funkce umožňuje implementaci kódu v jazycích nativních pro operační systém (Swift a Objective-C pro iOS, Java a Kotlin pro Android).

- **Hlavní výhoda:** React Native umožňuje psát kód v JavaScriptu a nabízí vysokou míru opakovaného využití kódu mezi platformami. To jej také činí atraktivním pro vývojáře, kteří jsou seznámeni s Reactem.
- **Hlavní nevýhoda:** Ačkoliv React Native nabízí možnost znovu použití kódu mezi platformami, v některých případech je stále potřeba napsat platformě specifický kód, což může zvýšit čas na vývoj. Kromě toho, React Native může narazit na problémy s výkonností při vytváření velmi komplexních uživatelských rozhraní a animací, neboť se spoléhá na JavaScript-native bridge pro komunikaci s nativními komponenty, což může způsobit zpoždění.

4.2.4 Flutter

V květnu 2017 společnost Google představila Flutter, open-source franework pro vývoj mobilních uživatelských rozhraní. Základním principem Flutteru je umožnit tvorbu nativních mobilních aplikací pro systémy iOS a Android z jediné kódové základny.

Flutter je tvořen dvěma klíčovými součástmi:

- **SDK (Software Development Kit):** SDK Flutteru nabízí nástroje, které vývojářům usnadňují tvorbu aplikací pro různé platformy, včetně systémů iOS, Android a Windows.
- **Uživatelské rozhraní:** Jde o sbírku opakovaně použitelných prvků uživatelského rozhraní, jako jsou tlačítka, textové políčka nebo posuvníky(sliders), které lze přizpůsobit konkrétním potřebám.

Pro vývoj s Flutterem se používá programovací jazyk Dart. Tento jazyk byl sice vytvořen společností Google již v říjnu 2011, avšak v posledních letech prošel významným vývojem.

- **Hlavní výhoda:** Flutter poskytuje vysoký výkon díky kompilaci do nativního kódu. To umožňuje vytvářet složité uživatelské rozhraní s plynulými animacemi bez ztráty výkonu.
- **Hlavní nevýhoda:** Flutter se využívá Dart, programovací jazyk, který není tak rozšířený jako JavaScript či C#. S tímto jazykem nemám dosud zkušenosti.

4.2.5 Xamarin

Tento open-source framework pro vývoj aplikací, který vytvořil Microsoft, se používá pro vývoj hybridnich aplikací pro iOS, Android a Windows.

Xamarin byl napsán pomocí jazyka C#, který je modernější a přináší vylepšení oproti jazykům Java a Objective-C. To, co činí Xamarin jedinečným, je schopnost přímo zahrnovat knihovny pro jazyky Java, C++ a Objective-C. Sbírka knihoven Xamarinu je obrovská. Tento rámec snižuje explicitní náklady a minimalizuje problémy s rozpočtem.

Xamarin je nyní přímo integrován do vývojářské platformy .NET s nástroji a knihovnami speciálně určenými pro vývoj aplikací pro Android, iOS, tvOS, macOS a Windows. .NET je platforma složená z nástrojů, programovacích jazyků a knihoven pro vytváření mnoha různých typů aplikací. Aplikace v .NET můžete psát v jazycích C#, F# nebo Visual Basic.

- **Hlavní výhoda:** Xamarin umožňuje využívat rozsáhlé knihovny .NET a psát kód v C#, což může být atraktivní pro ti, které jsou již seznámeny s ekosystémem .NET.

- **Hlavní nevýhoda:** Přestože Xamarin umožňuje psát kód v C# pro všechny platformy, některé aspekty aplikace mohou vyžadovat samostatnou implementaci pro každou platformu.

4.2.6 Konečný výběr

V důsledku podrobné analýzy a porovnání různých frameworků (viz. 4.5) pro vývoj hybridních aplikací jsem se rozhodl využít Ionic. Tento výběr je motivován několika faktory:

- Rychlost vývoje: Ionic využívá webové technologie, jako je HTML, CSS a JavaScript, což umožňuje rychlý vývoj aplikací. Jelikož je to technologie, se kterou jsem již dobře obeznámen, mohu efektivně využít své stávající dovednosti a znalosti pro rychlejší a hladší vývoj.

- Dokumentace: Ionic má velmi dobrou a podrobnou dokumentaci, která pokrývá širokou škálu témat a poskytuje jasné vysvětlení a příklady. To značně usnadňuje proces učení a řešení problémů během vývoje.

- Komunita: Ionic má silnou a aktivní komunitu vývojářů, která je zdrojem mnoha nástrojů, návodů a dalších zdrojů. To poskytuje velkou podporu při řešení problémů a zlepšování kvality kódu.

- Univerzálnost: Ionic podporuje vývoj aplikací pro různé platformy, včetně iOS, Android a web, což umožňuje jednoduché vytvoření a údržbu kódu napříč platformami.

Tyto faktory dohromady činí Ionic ideální volbou pro tento projekt, který si klade za cíl efektivní a rychlý vývoj kvalitní hybridní aplikace.

	Xamarin	Flutter	Ionic	React Native
Purpose	Cross-platform	Cross-platform	Hybrid	Cross-platform
Backed by	Microsoft	Google	Drifty	Facebook
Programming language(s)	C#	Dart	JS + HTML & CSS	JS, Java, C++, Objective-C, Python
Supported platforms	Android and iOS; Windows, macOS	Android and iOS	iOS and Android; Windows, BlackBerry	iOS and Android; Web
Code reusability	Up to 90%	Up to 70%	Up to 95%	Up to 85%
User interface	Native	Close to native	Not native	Native

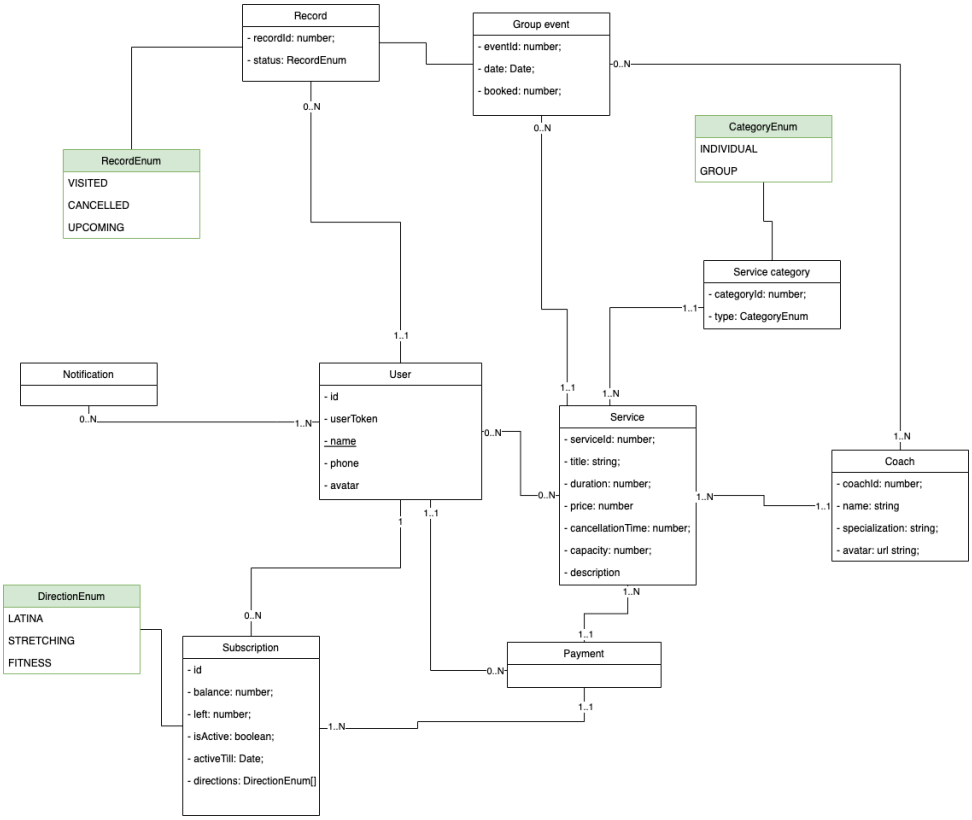
Obrázek 4.5: Porovnání frameworku

Kapitola 5
Návrh vlastního řešení

V následující kapitole se budeme zabývat návrhem hybridní mobilní aplikace pro rezervační systém pomocí softwarových nástrojů.

5.1 Diagram tříd

V rámci přípravy UML diagramu se budeme zaměřovat na identifikaci a modelování vztahů mezi jednotlivými entitami v rámci systému. Cílem je vytvořit kompletní a srozumitelný pohled na procesy a interakce, které se v rámci systému odehrávají. UML diagramy nám umožní vizuálně zobrazit a analyzovat vztahy mezi jednotlivými entity a tím nám pomohou vytvořit efektivní a funkční technické řešení.



Obrázek 5.1: Diagram tříd pro rezervacni systém

Class diagram (diagram tříd) je jedním z nejdůležitějších typů diagramů v metodologii Unified Modeling Language (UML), která se používá pro modelování a popis požadavků na software. Class diagram popisuje statickou strukturu systému, tj. třídy, jejich atributy a metody, vztahy mezi třídami a jejich omezení. Tyto informace se používají k vytvoření kódu aplikace. Class diagram je tedy významnou součástí projektování a analýzy systému, která pomáhá pochopit a specifikovat požadavky na systém a pomoci při komunikaci mezi vývojáři, analytiky a klienty. [9]

5.2 Případy použití

Usecase diagram, nebo také diagram případu použití, je typ diagramu, který slouží k popisu chování systému z pohledu uživatelů. Jsou používány k zobrazení interakce mezi uživateli a systémem. Usecase diagramy jsou užitečné pro pochopení požadavků na systém a pro návrh systému tak, aby splňoval tyto požadavky. [9]

- Uživatel - osoba, která využívá funkce systému
  1. Přihlášený uživatel - diagram[1] viz. 5.2

[1] Diagram případů užití byl vytvořen s určitou odchylkou od standardů, protože pokud

2. Nepřihlášený uživatel

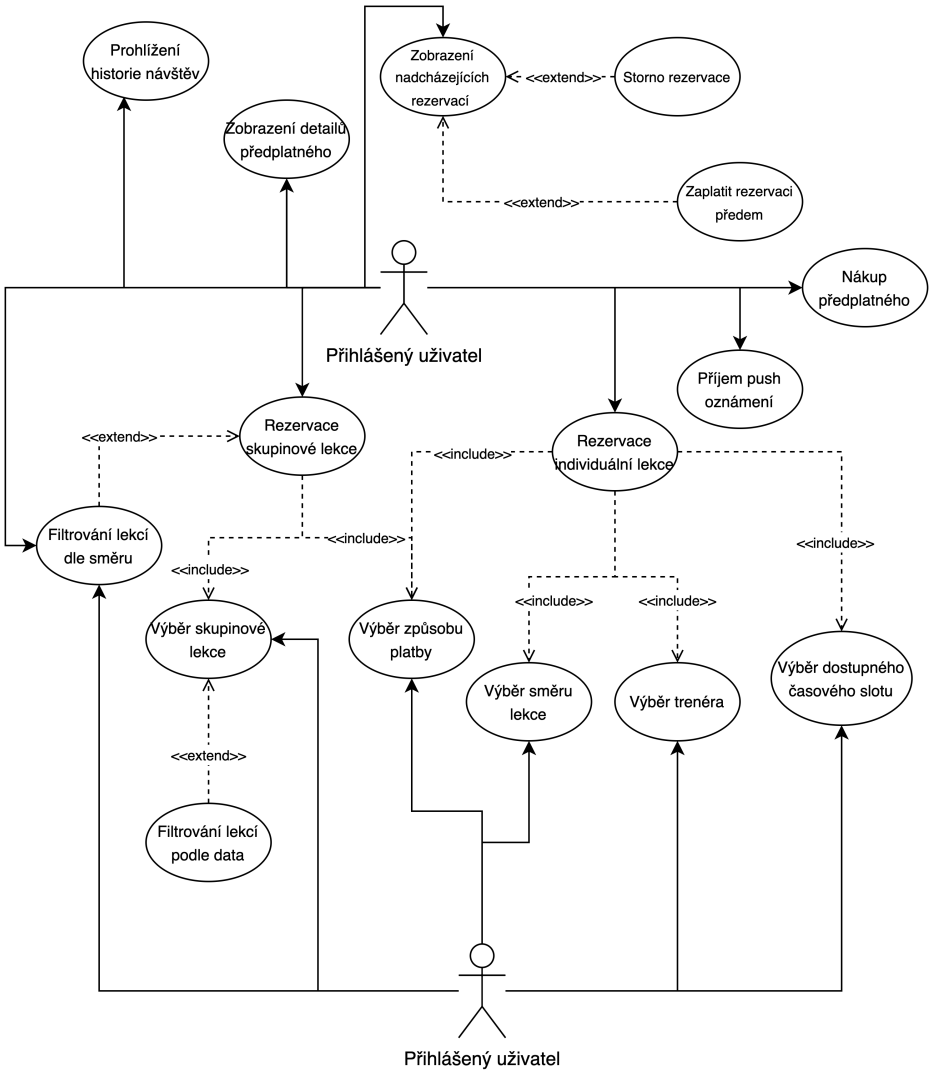
Pro daného aktéra jsem se rozhodl nevytvářet samostatnou tabulku, jelikož jeho interakce se systémem zahrnují pouze několik případů užití. Tyto případy užití mohou být stručně a jasně popsány v textové formě:

UC1: Autentizace v systému - Aktér se autentizuje v systému pomocí svých přihlašovacích údajů.

UC2: Ověření telefonního čísla - Aktér ověřuje své telefonní číslo pro zajištění další bezpečnosti účtu.

UC3: Registrace nového uživatele - V případě, že aktér ještě nemá v systému účet, provede registraci, která zahrnuje zadání potřebných informací a vytvoření přihlašovacích údajů.

bychom postupovali striktně podle nich, šipky od aktéra k případům užití by se křížily. Rozhodl jsem se proto pro tento formát, který je, jak se zdá, přehlednější a umožňuje lepší vizualizaci. Toto rozhodnutí bylo přijato v zájmu zlepšení srozumitelnosti a čitelnosti diagramu.



Obrázek 5.2: Přihlášený uživatel UC diagram.

5.3 Systemové požadavky

Systémové požadavky definujou funkce a schopnosti, které musí být implementovány v cílovém produktu. Rozlišujeme dva druhy těchto požadavků:

- **Funkční požadavky** jsou specifické vlastnosti nebo funkce produktu, které vývojáři musí implementovat, aby uživatelé mohli splnit své úkoly.
- **Nefunkční požadavky** nesouvisejí přímo s funkcionalitou systému, ale spíše definují, jak by systém měl fungovat.

Obě skupiny požadavků jsou důležité pro jasné pochopení toho, co systém musí a jak by měl fungovat.

V rámci stanovení systémových požadavků pro tento projekt jsem aplikoval metodu prioritizace známou jako MoSCoW. Metoda MoSCoW je nástroj, který umožňuje efektivně určovat důležitost jednotlivých požadavků. MoSCoW byl poprvé představen v metodice DSDM, kterou vyvinulo *Agile Business Consortium*. [**17**]

Tato metoda rozděluje požadavky do čtyř kategorií: Musí být (Must have), Mělo by být (Should have), Mohlo by být (Could have) a Nebude mít (Won't have). Tyto kategorie nám poskytují jasnou představu o kritičnosti jednotlivých požadavků pro úspěch projektu, a také umožňují efektivněji řídit zdroje v případě, že některé požadavky musí být odsunuty na pozdější fáze projektu.

- Musí být (**M**ust have): Požadavky v této kategorii jsou nezbytné pro úspěch projektu. Jsou to základní stavební kameny, bez kterých projekt nebude schopen splnit své cíle. Tyto požadavky musí být splněny za všech okolností.
- Mělo by být (**S**hould have): Požadavky v této kategorii jsou důležité, ale jejich nepřítomnost nesvede projekt k neúspěchu. Tyto požadavky přispívají k optimalizaci a efektivitě systému, ale mohou být kompromisovány, pokud je to nezbytné.
- Mohlo by být (**C**ould have): Tyto požadavky představují doplňkové funkce, které zlepšují uživatelskou zkušenost, ale nejsou pro fungování systému nezbytné. Tyto požadavky jsou obvykle odsunuty na pozdější fáze projektu v případě omezených zdrojů.
- Nebude mít (**W**on't have): Požadavky v této kategorii byly identifikovány, ale rozhodlo se, že nebudou v aktuální fázi projektu realizovány. Mohou být zahrnuty v plánu pro budoucí rozvoj systému.

Tato metodologie nám pomáhá lépe řídit a alokovat zdroje během vývoje systému a zajistit, že klíčové funkce budou implementovány jako první, zatímco méně důležité funkce mohou být odsunuty na pozdější fázi, pokud je to nutné. Kategorie požadavků je uvedena v závorce vedle názvu každého požadavku.

- **Funkční požadavky:**

  1. **FR1:** Autentizační modul (M)
     Systém musí poskytnout autentizační modul pro uživatele, aby se mohli přihlásit do svých účtů.
  2. **FR2:** Ověření telefonního čísla (M)
     Systém musí podporovat funkci ověření telefonního čísla, která umožní uživatelům ověřit své telefonní číslo pro zvýšení bezpečnosti účtu.
  3. **FR3:** Registrace uživatele (M)
     Systém musí umožnit novým uživatelům se zaregistrovat tím, že poskytnou potřebné informace a vytvoří přihlašovací údaje.

4. **FR4:** Správa rezervací (M)
   Systém musí umožňovat uživatelům prohlížet, vytvářet a rušit rezervace.
5. **FR5:** Správa předplatného (C)
   Systém musí poskytnout možnost prohlížení informací o předplatném a umožnit uživatelům zakoupit předplatné.
6. **FR6:** Zobrazení historie návštěv (S)
   Systém musí poskytnout uživatelům možnost prohlížet historii svých návštěv.
7. **FR7:** Filtrace lekcí (M)
   Systém musí umožnit uživatelům filtrovat lekce podle různých kritérií, jako je datum a směr.
8. **FR8:** Výběr trenéra (M)
   Systém musí umožnit uživatelům vybrat si trenéra pro individuální nebo skupinové lekce.
9. **FR9:** Výběr časového slotu (M)
   Systém musí umožnit uživatelům vybrat si dostupný časový slot pro jejich lekci.
10. **FR10.1:** Integrace platební brány pro přijímání plateb kartou (S)
    Systém musí být integrován s platební bránou, která umožňuje přijímání plateb kartou.
11. **FR10.2:** Podpora pro Apple Pay a Google Pay (C)
    Systém musí podporovat platební metody Apple Pay a Google Pay pro usnadnění transakcí pro uživatele těchto platform.
12. **FR11:** Integrace s externími systémy (M)
    Systém musí být integrován s API Altegio pro získání a zpracování dat.
13. **FR12.1:** Schopnost odesílat push notifikace (S)
    Systém musí být schopen odesílat push notifikace, aby mohl informovat uživatele o důležitých událostech a informacích.
14. **FR12.2:** Odesílání push notifikací v případě relevantních událostí nebo změn (C)
    Systém musí umožnit automatické odesílání push notifikací uživatelům v případě relevantních událostí nebo změn, které je ovlivňují.

- **Nefunkční požadavky:**

1. **NFR1:** Bezpečnost
   Aplikace musí zaručit bezpečný přístup k informacím pouze pro autorizované uživatele.
2. **NFR2:** Použitelnost
   Aplikace musí poskytnout intuitivní uživatelské rozhraní přístupné všem uživatelům bez ohledu na jejich věk a úroveň vzdělání.

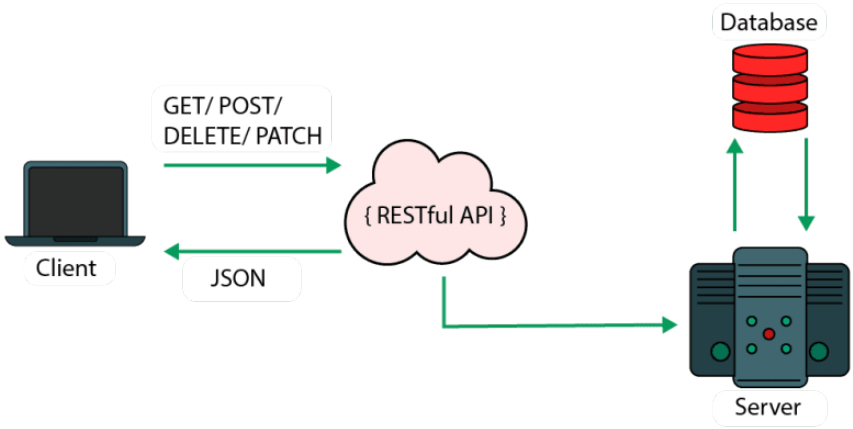
3. **NFR3:** Cross-platformnost
   Aplikace musí být kompatibilní a funkční na různých platformách - webové, mobilní pro iOS a Android.
4. **NFR4:** Rozšiřitelnost
   Systém by měl být navržen tak, aby umožňoval snadné začlenění nových funkcí, které nebyly předpokládány v době návrhu. To vyžaduje využití rozšiřitelných nástrojů a metod pro vývoj aplikace.
5. **NFR5:** Kompatibilita s prohlížeči
   Aplikace musí být kompatibilní se všemi hlavními webovými prohlížeči, včetně Chrome, Firefox, Opera a Safari.

5.4 Integrace s systemem Altegio

V mém projektu budu provádět integraci se systémem Altegio, který poskytuje své API s architekturou REST. To znamená, že budu využívat standardní HTTP metody pro komunikaci se systémem Altegio a získávání potřebných dat.

REST, neboli **RE**presentational **S**tate **T**ransfer, je architektonický styl pro distribuované hypermediální systémy.

Základní principy REST, které jsou popsány v knize "RESTful Web Services"od Leonarda Richardsona a Sama Rubyho [**18**], zahrnují:



Obrázek 5.3: Klient-serverová architektura REST API

- Klient-serverový architektonický styl 5.3: Systém je rozdělen na klienty, kteří iniciují požadavky, a servery, které zpracovávají požadavky a poskytují zdroje (například data nebo funkcionality).
- Bezestavovost: Každý klientův požadavek na server musí obsahovat veškeré informace potřebné k jeho zpracování, aniž by server ukládal stav mezi požadavky. Server by si neměl pamatovat předchozí požadavky od klienta.

- Kešování: Odpovědi serveru mohou být na klientovi kešovány, takže opakované požadavky na stejné zdroje mohou být zpracovány rychleji a snížit tak zátěž serveru.
- Jednotné rozhraní: Rozhraní mezi klientem a serverem musí být jednoduché, jednotné a omezené. Mělo by obsahovat minimální sadu operací (např. CRUD - vytvoření, čtení, aktualizace, odstranění) a využívat standardní HTTP metody (např. GET, POST, PUT, DELETE) pro přístup k zdrojům.
- Vrstvený systém: Architektura může být složena z několika vrstev, kde každá vrstva plní specifické funkce. Klienti nemusí být informováni o vnitřní struktuře systému a servery mohou být horizontálně škálovány přidáním dalších vrstev.

Tyto principy REST zajišťují jednoduchost, škálovatelnost, spolehlivost a výkon při vývoji webových služeb, což je činí snadno dostupnými s platformami.

5.5 Závěr kapitoly

V této kapitole byla provedena základní analýza budoucího software řešení. Byly vytvořeny základní kameny pomocí UML diagramů, na kterých lze stavět.

Kapitola 6

Implementace

Tato část práce je zaměřena na implementaci a popisuje využité nástroje a technologie, které byly nezbytné pro vývoj hybridní aplikace v rámci této bakalářské práce

6.1 Technologicky stack

V této podkapitole jsou popsány všechny technologie a programovací jazyky použité při vývoji hybridní aplikace.

Front-end

- Programovací jazyk: TypeScript.
- Framework: Ionic React.

Back-end

- Programovací jazyk: JavaScript.
- Framework: Express.js.
- Primární databáze: lokální úložiště na webu / userDefaults na iOS / sharedpreferences na Androidu.
- Dodatečné úložiště: Firebase Realtime database.

Komunikace mezi klientem a serverem

- REST API.

Tyto technologie byly vybrány na základě jejich výhod a jejich schopnosti by měli uspokojit požadavky projektu.

6.2 Architektura

V mé aplikaci, abych nespojoval logiku implementace a prezentace, jsem využil Redux a Redux Saga.

Při tvorbě následujícího textu bylo čerpáno z [**19**]

- **Redux** je nástroj pro správu stavu dat a uživatelského rozhraní v JavaScriptových aplikacích. Poskytuje centralizovanou správu stavu celé aplikace. Tím zajišťuje předvídatelnost stavu a umožňuje snadno sledovat změny stavu během provozu aplikace, což usnadňuje ladění a testování. Centrální koncepce Reduxu jsou **actions**, **reducers** a **store**.

  1. Reducer je čistá funkce, která přijímá state aplikace a action jako argumenty a vrací nový state.

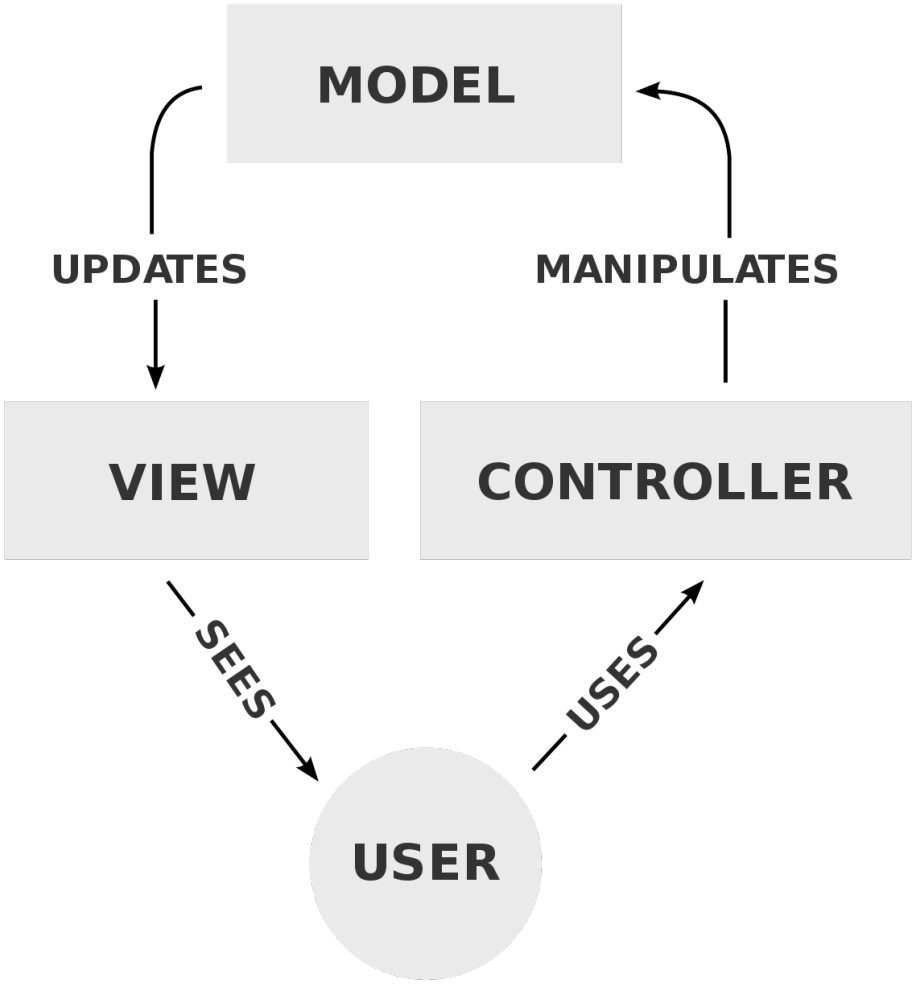
  2. State je data, se kterými komponenta (nebo komponenty) pracují - obsahuje data, která komponenta vyžaduje, a určuje, co komponenta renderuje.

  3. Action představují výhradní mechanismy, jež v aplikaci využívající Redux vyvolávají změny. Tyto akce nesou v sobě takzvaný payload, tedy data určená k modifikaci úložiště aplikace.

- **Redux Saga** je middleware, který přebírá řízení našich akcí, než dorazí přímo k reducerům.

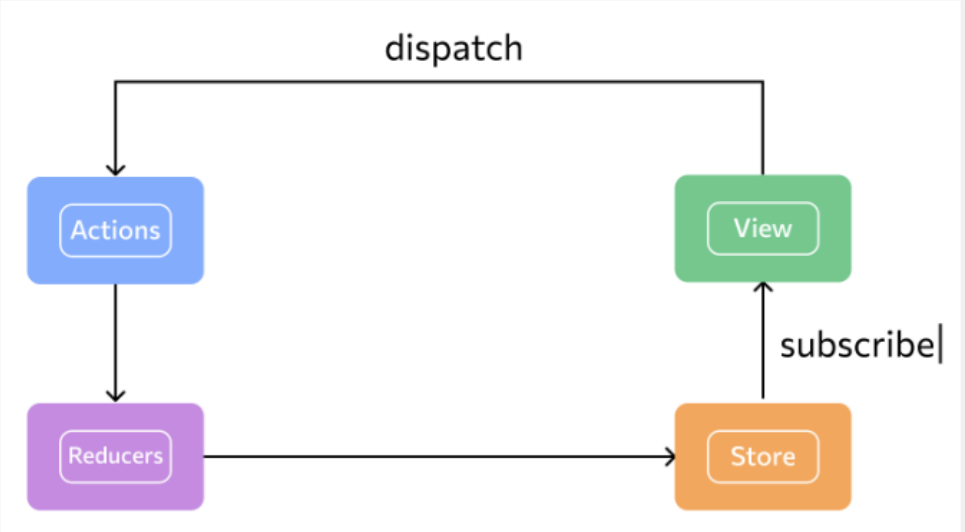
Model-View-Controller (MVC 6.1) je softwarový návrhový vzor, který se často používá pro vývoj uživatelských rozhraní, které dělí aplikaci na tři propojené části. Toto oddělení pomáhá řešit problémy spojené s udržitelností a škálovatelností projektu, a zároveň poskytuje možnost opětovného použití kódu a paralelní vývoj [**20**].

Model reprezentuje data a logiku aplikace. View zobrazuje data, tedy informuje uživatele o stavu modelu. Controller zpracovává vstupy uživatele a převádí je na příkazy pro model nebo pohled.



Obrázek 6.1: Architektura Model-View-Controller

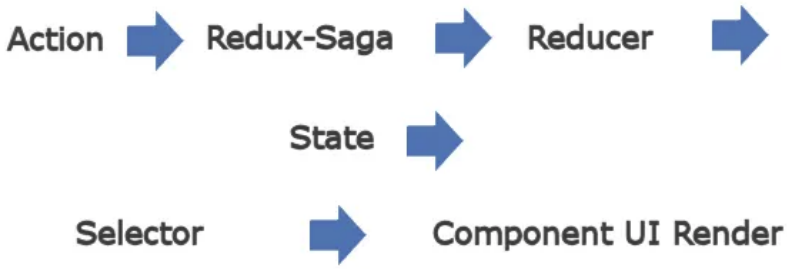
Redux připomíná MVC přístup 6.2. Komponenta funguje jako uživatelské rozhraní (View), action jako Controller, Reducer jako Model (udržuje aktuální stav) a Redux-Saga funguje jako pozadí služby. Store se podobá kontejneru pro stav, který sjednocuje všechny reducery.



Obrázek 6.2: Architektura Redux

Redux je jednoduše stavový kontejner, který podporuje asynchronní přístup. Když dojde k akci, objekt je odeslán do úložiště, poté je volán reducer a ten aktualizuje stav.

V případě asynchronního přístupu Redux používá middleware Redux-Saga (viz. 6.3). Ten je spuštěn po akci, ale před voláním reduceru.



Obrázek 6.3: Architektura Redux + Redux-Saga

6.3 Backend

Vzhledem k tomu, že můj projekt je založen na odesílání požadavků na API již existující služby, kde jsou uložena veškerá pro mne potřebná data, a následném zpracování a prezentaci těchto dat, nebylo třeba implementovat samostatnou backendovou aplikaci, která by mi poskytovala určité API. Zároveň jsem nepotřeboval ukládat velké množství dat.

Z backendu jsem potřeboval pouze několik endpointu pro integraci platebního systému a pro odesílání push notifikací, stejně jako malou databázi pro ukládání tokenů uživatelů, které jsou použity pro odesílání těchto push

notifikací. Proto bylo rozhodnuto využít službu Firebase, která nabízí služby typu Backend as a Service. Při tvorbě následujícího textu bylo čerpáno z [**21**]

6.3.1 Firebase

Firebase je poskytovatel cloudových služeb a zároveň backendové řešení, které umožňuje získávat strukturovaná data pro mobilní aplikace. Firebase je snadno použitelný a umožňuje rychlé čtení a zápis dat. Firebase lze využít pro vývoj aplikací pro iOS, Android, ale i webových aplikací s reálným časem dat a úložištěm a nabízí také řadu dalších produktů.

V rámci této práce využíváme následující funkce poskytované službou Firebase.

- **Real-time databaze**

  Firebase Realtime Database je databáze, která je hostována v cloudu. Výhodou tohoto řešení je, že data jsou ukládána jako JSON a synchronizována v reálném čase se všemi připojenými klienty. Tato funkcionalita je klíčová při vývoji multiplatformních aplikací, protože nezávisle na použité platformě (Apple, Android, JavaScript) všechny klienty sdílí jednu instanci Realtime Database a automaticky dostávají nejnovější data.

  Firebase Realtime Database nabízí několik klíčových schopností. První z nich je fungování v reálném čase. Na rozdíl od typických HTTP požadavků využívá databáze synchronizaci dat. To znamená, že jakmile dojde ke změně dat, všechna připojená zařízení dostávají tuto aktualizaci téměř okamžitě.

  Druhou významnou schopností je fungování v offline režimu. Aplikace zůstávají responzivní i bez internetového připojení, protože data jsou ukládána na disk. Jakmile se konektivita obnoví, klient synchronizuje veškeré změny se stavem serveru.

  Další klíčovou vlastností je přímý přístup k databázi z klientských zařízení, ať už se jedná o mobilní zařízení nebo webový prohlížeč. K zajištění bezpečnosti a validace dat slouží Firebase Realtime Database Security Rules.

- **Cloud functions**

  Firebase Cloud Functions nám umožňují automaticky spouštět backendový kód v reakci na události vyvolané funkcemi Firebase a HTTPS požadavky. Náš kód je uložen v cloudu Google a běží v řízeném prostředí, což eliminuje potřebu správy a škálování vlastních serverů.

  Jakmile jsme funkci napsali a nasadili, servery Google okamžitě přebírají řízení funkce. Funkci můžeme spustit přímo pomocí HTTP požadavku, prostřednictvím modulu Cloud Functions, nebo v případě tzv. "background functions"servery Google naslouchají událostem a spouštějí funkci, kdykoli je to vhodné.

- **FCM**

  Firebase Cloud Messaging (FCM), dříve známý jako Google Cloud Messaging (GCM), je bezplatná cloudová služba od Google. Tato služba umožňuje vývojářům aplikací posílat oznámení a zprávy uživatelům napříč různými platformami, včetně Android, iOS a webových aplikací.

  FCM poskytuje vývojářům softwaru schopnost odesílat push oznámení do jejich aplikací koncovým uživatelům prostřednictvím aplikačního programovacího rozhraní (API).

  Pro odesílání a příjem zpráv pomocí FCM jsou potřebné dva prvky: důvěryhodné prostředí nebo server, na kterém se vytvářejí, směrují a odesílají zprávy, a klientská aplikace pro Android, iOS nebo Web, která zprávy přijímá.

  Oznámení jsou zobrazována na uživatelově zařízení prostřednictvím FCM jménem aplikace. Zprávy s daty jsou přímo zpracovány aplikací, která je zodpovědná za doručení zprávy uživateli.

6.3.2 Autorizace

Jak jsem již předtím naznačil, potřebné data získáváme prostřednictvím Altegio REST API. Pro přístup k skupině metod ktere poskytuje Altegio je vyžadována autorizace partnera a uživatela. Jinými slovy, je nutné předat jedinečný hash **partner token**.

- Partner token

  Bearer autentizace (také nazývána autentizace pomocí tokenu) je schéma HTTP autentizace, které zahrnuje bezpečnostní tokeny nazývané Bearer tokeny. Název "Bearer autentizace"lze chápat jako "povolení přístupu držiteli tohoto tokenu". Bearer token je hash řetězec, obvykle generovaný serverem v reakci na požadavek na přihlášení. Při vytváření API požadavků musí HTTP autorizační hlavička obsahovat přístupový klíč ve formátu:

  Authorization: Bearer <partner token>

  Podobně jako základní autentizace by Bearer autentizace měla být používána pouze přes HTTPS (SSL).

- Uživatel

  Pro získání uživatelského API klíče se používá metoda auth.

Obrázek 6.4: endpoint pro autorizaci v systému Altegio

Samotný klíč musí být také odeslán v hlavičce požadavku (za partnerovým klíčem, oddělený čárkou):

Authorization: Bearer <partner token>, User <user token>

To, zda uživatel potřebuje autorizaci k práci s určitými entitami, by mělo být vidět v popisech formátu dat ktere lze najit v oficiální dokumentaci Altegio REST API `https://developer.alteg.io/api`.

6.4 Frontend

6.4.1 Struktura projektu

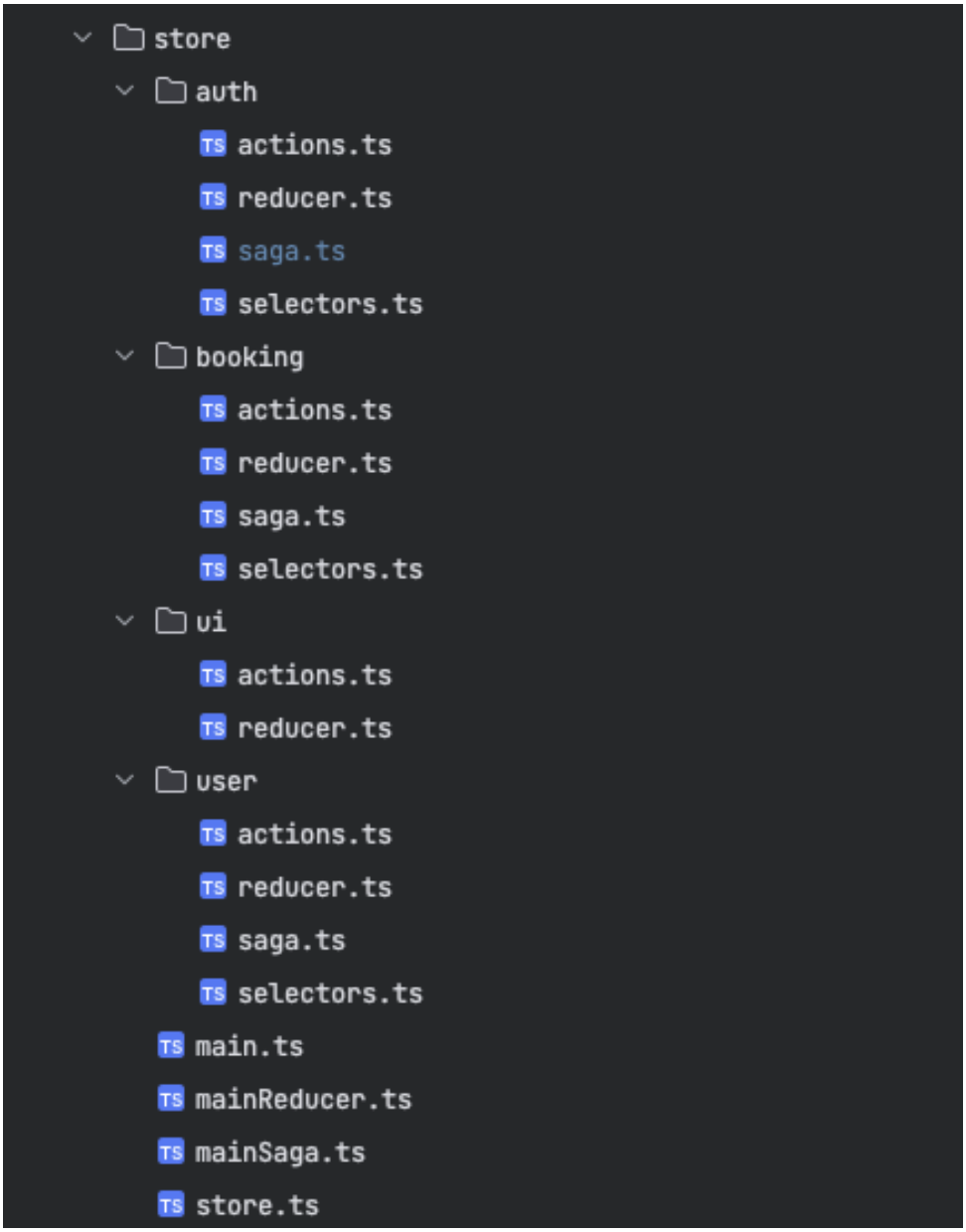
Rozhodl jsem se strukturu komponent organizovat do tří hlavních kategorií: *components*, *features* a *layout* . Tato organizace pomáhá udržet kód přehledný a usnadňuje navigaci v projektu.

- Složka components obsahuje obecné komponenty, které se používají na mnoha místech v aplikaci. Tyto komponenty jsou obecně nezávislé a mohou být použity ve více kontextech v rámci aplikace. Mohou to být tlačítka, formulářové prvky, karty a další podobné prvky.

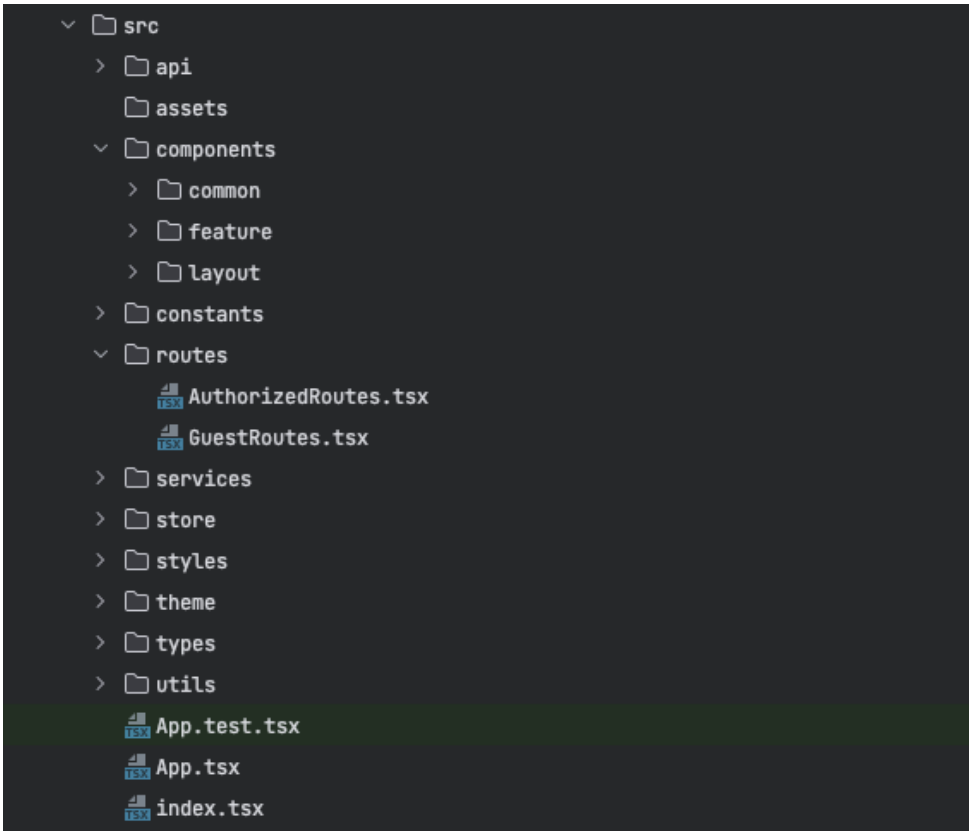
- Složka features obsahuje komponenty, které tvoří specifické stránky nebo funkce vaší aplikace. Tyto komponenty jsou více zaměřené na funkcionalitu a obvykle zahrnují určité logické operace. Mohou to být komponenty jako LoginPage, BookingConfirmationPage atd.

- V sekci layout se nachází komponenty, které definují obecný vzhled a strukturu aplikace. Zahrnují základní kostru, která je společná pro všechny stránky, například PageLayout nebo Header. Tyto komponenty definují, jak jsou jednotlivé části aplikace uspořádány a jak jsou prezentovány uživateli.

- Složka store - zde je koncentrována veškerá logika týkající se správy stavu pomocí Redux a Redux Saga.



Obrázek 6.5: Struktura složky **store** v projektu



Obrázek 6.6: Struktura projektu v prostředí Ionic

6.4.2 Ionic

Po důkladné analýze pro vývoj hybridní aplikace byl vybrán framework Ionic v kombinaci s Reactem. V této podkapitole se zaměřím na využití tohoto frameworku a jeho přednosti pro náš projekt.

- **Ionic UI** je součástí *Ionic Frameworku*, který poskytuje mnoho předdefinovaných uživatelských rozhraní (UI), které usnadňují a urychlují vývoj mobilních aplikací. Tyto komponenty UI jsou navrženy tak, aby byly přizpůsobitelné a konzistentní napříč různými platformami, jako jsou iOS, Android a web. Například komponenta *IonTab* vytváří navigaci na záložky v aplikaci. Toto je obvyklý styl navigace v mobilních aplikacích a Ionic jej usnadňuje implementací. Níže je uveden příklad použití *IonTab* v me aplikace.

```
<IonTabs>
  <IonRouterOutlet animated>
    <Switch>
      <Route exact={true} path="/" component={DashboardPage}></Route>
      <Route
        exact={true}
        path="/:tab(services)/coaches/:serviceId"
```

```
            component={CoachCatalogPage}
          ></Route>
          ...
          ...
          <Route
            exact={true}
            path="/bookingTime/:serviceId/:coachId"
            component={BookingTimePicker}
          ></Route>
          <Redirect to="/" />
        </Switch>
      </IonRouterOutlet>

      <IonTabBar>
        <IonTabButton tab="tab1" href="/">
          <IonIcon icon={homeOutline} />
        </IonTabButton>
        <IonTabButton tab="services" href="/services">
          <IonIcon/>
        </IonTabButton>
        <IonTabButton tab="tab3" href="/history">
          <IonIcon icon={cardOutline} />
        </IonTabButton>
      </IonTabBar>
    </IonTabs>
```

- **IonReactRouter** je komponenta z Ionic Frameworku, která využívá *react-router* pro správu routingu v React aplikacích..

  V me aplikaci IonReactRouter obaluje dva možné stavy aplikace, závislé na tom, zda je uživatel autorizován nebo ne.

  Pokud je uživatel autorizován, pak se renderují cesty pro autorizované uživatele pomocí komponenty AuthorizedRoutes. Pokud uživatel není autorizován, renderují se GuestRoutes, jako přihlášení a registrace.

  ```
  <IonReactRouter>
  {isAuthorized ? <AuthorizedRoutes /> : <GuestRoutes />}
  </IonReactRouter>
  ```

- **IonGrid** je komponenta poskytovaná Ionic Frameworkem, která umožňuje vytvářet responzivní layouty pomocí systému řádků a sloupců. *IonGrid* je inspirován populárním grid systémem Bootstrapu, ale je navržen tak, aby byl kompatibilní s filozofií designu Ionic. *IonGrid* se automaticky přizpůsobuje různým velikostem obrazovky a orientacím.

```
<PageLayout>
  <IonGrid>
    <IonRow>
      {availableSlotsArray?.map((slot, index) => (
        <IonCol size="4" key={index}>
          <TimeItem
            selectedDate={selectedDate}
            serviceId={match.params.serviceId}
            coachId={match.params.coachId}
            from={slot.from}
            to={slot.to}
          />
        </IonCol>
      ))}
    </IonRow>
  </IonGrid>
</PageLayout>
```

Důležitým atributem *IonCol* je **size**, který určuje, kolik prostoru v rámci řady daný sloupec zabírá. V tomto příkladě je nastaveno na "4", což znamená, že každý sloupec zabírá třetinu šířky mřížky (předpokládáme, že celkový počet sloupců v mřížce je 12, což je běžná praxe).

6.4.3 Redux

V rámci své aplikace využívám technologii Redux pro správu stavů a Redux Saga jako middleware pro řízení asynchronních požadavků na API Altegio. Úložiště Redux aplikuje design pattern Observer, kde jednotlivé komponenty rozhraní sledují globální stav aplikace a podle něj dynamicky upravují svůj obsah a chování.

Pro správu stavů jsou ve stromu aplikace definovány čtyři hlavní uzly: **ui**, **auth**, **booking** a **user** 6.5. Každý z nich reprezentuje určitou oblast aplikace a ukládá relevantní data.

- Modul user zastřešuje všechny údaje přihlášeného uživatele, které jsou poté zobrazovány na jednotlivých stránkách aplikace. Asynchronní požadavky na API Altegio pro získání aktuálních dat jsou implementovány pomocí Axios funkcí v souboru user/auth.js.

- Modul auth spravuje informace týkající se přihlášení uživatele a klíčové uživatelské údaje, jako je telefonní číslo a uživatelský token nezbytný pro komunikaci s API Altegio.

- Modul booking zahrnuje veškeré údaje potřebné pro rezervaci skupinových nebo individuálních lekcí, jako jsou existující směry a trenéři přiřazení k těmto směrům.

- Modul ui zajišťuje interakci s uživatelem prostřednictvím zobrazování informačních a chybových hlášení, takzvaných toastů. Ty se uživateli zobrazují v případě, že požadavek na server vrátí chybová hlášení.

Pro každý ze stavů je v rámci architektury aplikace definován také soubor selektorů, které slouží pro dotazování se na aktuální údaje uložené v těchto stavech na různých místech aplikace.

6.4.4 Používané knihovny a API

V průběhu implementace frontendové části aplikace jsem využil několik závislostí poskytovaných prostřednictvím správce balíčků npm. Tyto knihovny přispěly k obohacení funkčnosti aplikace, zlepšení uživatelského rozhraní a efektivizaci procesu vývoje.

1. **Axios**: Tato knihovna poskytuje HTTP klienta, který umožňuje komunikovat s externími API.
2. **SwiperJS**: Moderní JavaScriptova knihovna specializovana na vytváření dynamických karuselů, posuvníků a přechodových efektů.
3. **Formik**: Knihovna pro React, která výrazně zjednodušuje proces tvorby a validace formulářů.
4. **Ion Icons**: Toto je open-source knihovna, poskytující širokou škálu kvalitních ikon pro webové aplikace.
5. **Date-fns**: Tato knihovna slouži pro manipulaci s daty v JavaScriptu.
6. **Styled Components**: Knihovna pro stylizaci komponent v Reactu.
7. **React OTP Input**: Knihovna umožňuje snadné vytvoření pole pro zadání jednorázových kódů (OTP - One Time Password) v Reactu.
8. **Yup**: Knihovna pro validaci dat v JavaScriptu.
9. **History**: Knihovna poskytuje rozhraní pro manipulaci s historií prohlížeče.

6.5 Uživatelské rozhraní a struktura aplikace

Při návrhu uživatelského rozhraní pro naši aplikaci jsem se zaměřil na vytvoření intuitivního a snadno použitelného designu. K tomu jsem využil nástroj Figma, což je online služba určená pro vývoj uživatelských rozhraní a prototypování.

Tato podkapitole podrobně popisuju jednotlivé prvky našeho uživatelského rozhraní. Každý element je doprovázen jeho vizuální reprezentací, kterou lze najit v Příloze A, a popisem jeho funkčnosti. To nám umožní lépe pochopit, jak jednotlivé komponenty přispívají k celkové uživatelské zkušenosti.

Na obrázku A.1 je zobrazena základní přihlašovací stránka. Bez přihlášení do účtu uživatel nezíská přístup k datům. První stránka obsahuje pole pro zadání telefonního čísla. Jakmile je telefonní číslo zadáno, zobrazí se tlačítko pro přihlášení. Po jeho stisknutí je uživatel přesměrován na stránku A.2 pro ověření telefonního čísla. V tomto okamžiku jsou údaje o telefonním čísle odeslány do systému Altegio, který následně zasílá ověřovací kód přes messenger (nejprve je pokus o zaslání přes Telegram, v případě neúspěchu pak přes WhatsApp).

Po zadání ověřovacího kódu je nový uživatel přesměrován na registrační stránku A.5, kde jsou po něm vyžadovány nezbytné údaje - konkrétně jeho e-mail a celé jméno. Pokud je uživatel již registrován, je přesměrován na hlavní stránku A.3

Aplikace má celkem tři záložky. První záložka A.3 představuje hlavní stránku, druhá záložka A.6 je určena pro rezervace a třetí A.11 představuje historii návštěv uživatele. Tyto záložky jsou realizovány prostřednictvím komponenty Ion Tabs.

Na hlavní stránce uživatel vidí veškeré potřebné informace. Pod hlavičkou se nacházejí všechny aktivní nebo ještě neaktivované permanentky, které si uživatel zakoupil. Každá permanentka obsahuje informace o názvu, zbývajícím počtu lekcí, celkovém počtu lekcí, seznamu směrů, na které se permanentka vztahuje, a pokud je permanentka již aktivována, tak i datum, do kdy je platná. Pokud má uživatel více permanentek, pak se seznam permanentek zobrazí ve slideru, kde může uživatel pomocí horizontálního scrollování prstem listovat mezi permanentkami. Pod permanentkami je informace o nadcházejících trénincích seřazených podle data. Tyto prvky obsahují všechny potřebné údaje o tréninku - směr, cenu, trenéra, čas. V pravém horním rohu kontejneru je šipka směřující dolů, což je Ionic prvek ion-action-sheet, dialogové okno, které zobrazuje sadu parametrů. Toto okno se zobrazuje nad obsahem aplikace, jak je uvedeno na obrázku A.4.

Druhá záložka, stránka pro rezervaci, zobrazuje seznam služeb nabízených naší taneční studií. Tyto služby jsou rozděleny do dvou kategorií - individuální a skupinové. Tady jsou uvedeny důležité informace, jako je cena a délka trvání jednotlivých služeb..

Proces rezervace individuálních lekcí se liší od procesu rezervace skupinových lekcí. Nejprve se podíváme na rezervaci individuálních lekcí.

Po výběru jakékoliv individuální služby je uživatel přesměrován na stránku výběru trenéra A.7. Jakmile si uživatel vybere trenéra, dostane se na stránku pro výběr času A.8. Na této stránce je potřeba, aby klient nejprve zvolil preferované datum a poté časový rozsah lekce (který je závislý na pracovním rozvrhu trenéra a délce vybrané služby). Po úspěšném výběru je uživatel přesměrován na stránku potvrzení rezervace A.9, kde je zobrazeny všechny informace o lekci, na kterou se chce přihlásit, pro kontrolu. Na stránce potvrzení je také přepínač, který umožňuje klientovi nám poskytnout informaci, zda chce lekci zaplatit online.

Pokud jde o skupinové lekce, postup je jednodušší. Když uživatel vybere skupinovou službu, je okamžitě přesměrován na stránku pro výběr data a

času A.10, kde jsou zobrazeny všechny možné lekce s různými trenéry podle směrů, které si uživatel vybral na stránce výběru služeb (může také zobrazit rozvrh všech služeb kliknutím na tlačítko "zobrazit vše").

Třetí záložka, stránka historie návštěv, poskytuje klientovi celkové informace o návštěvách naší taneční studie. Tato stránka je rozdělena do dvou sekci. První obsahuje nadcházející lekce, zatímco druhá obsahuje informace o navštívených nebo zrušených lekcích. Stav zápisu je označen buď zelenou barvou, pokud se jedná o navštívenou lekci, nebo červenou, pokud byla lekce zrušena.

6.6 Integrace platební brány

Při tvorbě následujícího textu bylo čerpáno z [**22**] [**23**] [**24**] [**25**]

V ramci navrhu naše aplikace jsem dospěl k závěru, že potřebujeme implementovat platební bránu pro zpracování transakcí. Po analýze různých možností jsme se rozhodli pro službu Stripe.

Stripe se prezentuje jako poskytovatel platebních služeb, který umožňuje bezproblémové zpracování online plateb. Jde o platební bránu, která automaticky zpracovává transakce s kartami a poskytuje ochranu proti podvodům jak pro společnost, tak pro zákazníka. Za tyto služby si Stripe účtuje malý poplatek z každé transakce.

Podle dat společnosti BuiltWith je Stripe využíván na více než 3 miliony aktivních webových stránek pro přijímání plateb. Data společnosti Datanyze zase ukazují, že Stripe drží 15,49% podíl na trhu platebních služeb, pokud hodnotíme podle počtu webových stránek využívajících tento software.

Co se týče platebních metod, Stripe podporuje celkem 32 z nich.

Výběr Stripe jako naší platební brány byl motivován řadou výhod:

- Stripe přijímá většinu platebních metod včetně kreditních a debetních karet (Visa, Mastercard), Apple Pay, Android Pay, Bitcoin a dalších kryptoměn. Nabízí také možnost přijímat bankovní převody.
- Stripe Payments je kompatibilní s počítači, systémy iOS, Androidem a v podstatě jakýmkoli jiným operačním systémem s přístupem k internetu.
- Kvalitní dokumentaci, kterou Stripe poskytuje. Na webových stránkách Stripe [**25**] najdeme řadu užitečných příruček pro obchodníky a vývojáře . Dokumentace Stripe API je důkladná, srozumitelná a velmi přehledná. Toto výrazně usnadňuje práci při integraci Stripe do našeho projektu.

Jak jsem v naší aplikaci implementoval platební bránu Stripe.

Díky kódu na straně klienta a na straně serveru jsme byli schopni vytvořit formulář pro objednávky s UI prvky, které byly vyvinuty samotným Stripe, umožňujícími různé způsoby platby.

Pro implementaci uživatelské platební brány bylo nutné vytvořit tzv. *endpoint*, na který se odkazuji z front-endu, když klient chce zaplatit rezervaci.

Pro vytvoření tohoto *endpointu* jsem použil framework *Express.js*, který mi umožňuje pomocí JavaScriptu napsat příslušnou funkci, kterou jsem následně nasadil na *Firebase Cloud Functions*.

Server:
Prvním krokem bylo instalování balíčku Stripe a jeho import do našeho projektu pomocí API klíče.

```
npm install --save stripe
```

Následně jsem vytvořil *endpoint*, který vytváří objekt *PaymentIntent*. *PaymentIntent* sleduje životní cyklus platby klienta, uchovává informace o všech neúspěšných pokusech o platbu a zajišťuje, že z klientovy karty bude stržena částka pouze jednou. *Endpoint* vrací klientův tajný klíč od *PaymentIntentu* v odpovědi, aby bylo možné dokončit platbu na straně klienta.

```
const functions = require("firebase-functions");
const express = require("express");
const stripe = require("stripe")("${process.env.REACT_APP_STRIPE_API_KEY}");

const endpointSecret = "${process.env.REACT_APP_WEBHOOK_ENDPOINT_SECRET}";

app.post("/create-payment-intent", async (req, res) => {
  const { amount } = req.body;
  const paymentIntent = await createPaymentIntent(req.body);
  res.send({
    clientSecret: paymentIntent.client_secret,
  });
});
```

Při vytváření objektu *PaymentIntent* můžeme povolit a zakázat různé platební metody. Před tím, než zobrazíme platební formulář, Stripe vyhodnotí měnu, omezení platebních metod a další parametry, aby určil seznam podporovaných platebních metod.

Frontend:
Na straně klienta jsem vytvořil samostatný React komponent *PaymentForm*, do kterého jsem integroval platební formulář Stripe. Musel jsem použít *Stripe.js* a UI knihovnu *Stripe Elements*, aby byly splněny požadavky PCI a aby bylo zajištěno, že platební údaje klienta přímo přecházejí do Stripe a nikdy nejsou zasílány na náš server.

```
npm install --save @stripe/react-stripe-js @stripe/stripe-js
```

Při volání funkce *loadStripe()* s naším veřejným API klíčem Stripe nastavujeme knihovnu Stripe. Je důležité volat *loadStripe()* mimo render komponenty, aby se předešlo opětovnému vytváření objektu Stripe při každém renderu.

Hodnotu, kterou jsme získali od *loadStripe()*, je třeba předat poskytovateli *Elements*. To umožňuje dceřiným komponentům získat přístup k Stripe službě prostřednictvím spotřebitele *Elements*. Dále předáváme tajný klíč klienta jako možnost poskytovateli *Elements*.

```
const stripePromise = loadStripe("${process.env.REACT_APP_STRIPE_PUBLIC_KEY}");

const PaymentForm = ({ amount, activity_id }: IProps) => {
  const [clientSecret, setClientSecret] = useState<string>();

  const clientId = useSelector(getClientId);
  const name = useSelector(getUserName);
  const phone = useSelector(getUserPhone);
  const email = useSelector(getUserEmail);

  useEffect(() => {
    getClientSecret(amount, clientId, activity_id, name, phone, email).then(
      setClientSecret
    );
  }, [amount]);

  return (
    <>
      <IonHeader>
        <IonToolbar>
          <IonTitle>Pay {amount}</IonTitle>
        </IonToolbar>
      </IonHeader>
      <IonContent>
        <Elements
          options={{ clientSecret, appearance: { theme: "stripe" } }}
          stripe={stripePromise}
        >
          <CheckoutForm />
        </Elements>
      </IonContent>
    </>
  );
};
```

Komponenta *CheckoutForm* obsahuje v sobě Stripe komponentu *PaymentElement*. Ta vkládá do stránky dynamický formulář ve formě iframe, který sbírá platební údaje pro různé metody platby.

Pokud klient klikne na tlačítko pro platbu, zavolám funkci *confirmPayment()*, kterou poskytuje *PaymentElement*. Té předám parametr *return-url*, který definuje, kam by měl Stripe přesměrovat uživatele po dokončení platby. U plateb, které vyžadují autentizaci, Stripe zobrazí modální okno pro au-

tentizaci 3D Secure nebo přesměruje klienta na stránku pro autentizaci, v závislosti na použité metodě platby. Po úspěšném dokončení autentizace je klient přesměrován na URL adresu specifikovanou v *return-url*.

```
const CheckoutForm = () => {
  const stripe = useStripe();
  const elements = useElements();

  const [readyToPay, setReadyToPay] = useState<boolean>(false);
  const canPay = elements && stripe && readyToPay;
  const handlePay = async () => {
    if (!canPay) return;
    const { error } = await stripe.confirmPayment({
      elements,
      confirmParams: {
        return_url: "http://localhost:8100/thank-you-page/",
      },
    });

    console.log({ error });
  };

  return (
    <>
      <IonCard>
        <IonCardContent>
          <PaymentElement
            id="payment-element"
            options={{ layout: "tabs" }}
            onChange={({ complete }) => setReadyToPay(complete)}
          />
        </IonCardContent>
      </IonCard>

      <StyledPrimaryButton expand="full" onClick={handlePay} disabled={!canPay}>
        Pay
      </StyledPrimaryButton>
    </>
  );
};
```

Stripe odesílá mnoho událostí během procesu platby a po dokončení platby. Využil jsem webhooky ve Stripe Dashboardu, abych vytvořil příjmač událostí po události payment-intent.succeeded, který odesílá požadavek na náš endpoint /stripe-webhook. Tento endpoint vytváří a platí rezervace v systému Altegio.

```
app.post("/stripe-webhook", async (request, response) => {
```

```
  const sig = request.headers["stripe-signature"];

  let event;

  try {
    event = stripe.webhooks.constructEvent(
      request.rawBody,
      sig,
      endpointSecret
    );
  } catch (err) {
    console.log({ err });
    response.status(400).send(`Webhook Error: ${err.message}`);
    return;
  }

  const res = await handlePaymentStripe(event);

  console.log(res);
  response.send(res);
});

const processPayment = async (event) => {
  const { metadata, amount } = event;
  if (!metadata?.client_id) return { success: false, error: "No client id" };
  if (!metadata?.activity_id)
    return { success: false, error: "No activity id" };
  if (!metadata?.email) return { success: false, error: "No email" };
  if (!metadata?.phone) return { success: false, error: "No phone" };
  if (!metadata?.name) return { success: false, error: "No name" };

  const { activity_id, client_id, email, phone, name } = metadata;

  const record_id = await altegioRequest.createBooking(
    name,
    phone,
    email,
    activity_id
  );

  const document_id = await altegioRequest.getDocumentId(
    activity_id,
    client_id
  );
  await altegioRequest.setAttendance(record_id);
  const success = altegioRequest.createPayment(document_id, amount / 100);
```

```
  return { success };
};

handlePaymentStripe = (event) => {
  switch (event.type) {
    case "payment_intent.succeeded":
      return processPayment(event.data.object);
    default:
      return { success: false, message: `Unhandled event type ${event.type}` };
  }
};

module.exports = handlePaymentStripe;
```

Kapitola 7
Testování

Následující kapitola se zaměřuje na dvě klíčové fáze vývoje naše aplikací, konkrétně na testování API a uživatelské testování. Tyto dvě složky hrají zásadní roli v zajištění technické správnosti a uživatelské přívětivosti našeho systému.

7.1 Testování API

V průběhu mého výzkumu a vývoje aplikace bylo nezbytné ověřit a otestovat API Altegio. Jedním z nejúčinnějších způsobů testování API je použití nástroje zvaného Postman.

Postman je populární platforma, která umožňuje vývojářům testovat požadavky API před jejich implementací do kódu. Může generovat a odesílat požadavky na koncové body API, analyzovat odpovědi a pomoci vytvářet automatizované testy pro zajištění konzistence a kvality API.

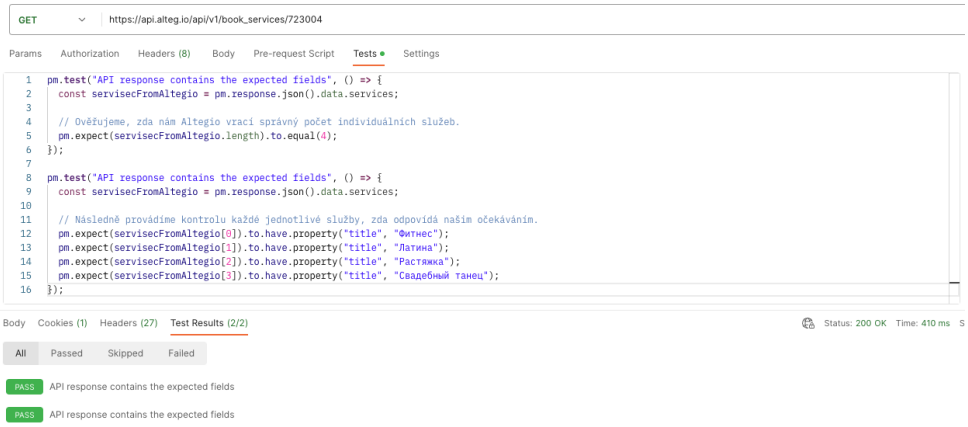
Nejprve jsem musel nastavit Postman pro práci s API Altegio. To zahrnovalo nastavení autentizace a přístupových tokenů, které byly potřebné pro interakci s API.

Poté, co jsem měl Postman nastaven, jsem začal testovat různé koncové body API. Pro každou funkci nebo službu, kterou API nabízelo, jsem vytvořil odpovídající požadavek v Postmanu. Například, pokud API Altegio nabízelo funkci pro získání informací o taneční lekci, vytvořil jsem v Postmanu požadavek GET na odpovídající koncový bod.

Při odesílání požadavků na API jsem pečlivě sledoval a analyzoval odpovědi. Důležitou částí tohoto procesu bylo ověření, že API správně reagovalo na každý požadavek - že vracelo správné kódy stavu, data a hlavičky.

V případě, že odpověď API nebyla v souladu s očekáváním, jsem analyzoval chybové zprávy nebo data a zkoumal, jaké změny nebo opravy by mohly být potřebné.

Přestože byl tento proces časově náročný, byl nezbytný pro zajištění kvality a funkčnosti aplikace. Díky testování API v Postmanu jsem mohl zajistit, že aplikace správně komunikuje s API Altegio a že všechny funkce a služby fungují správně.



Obrázek 7.1: Příklad testu v Postman.

7.2 Testování UI

V rámci testování uživatelského rozhraní jsem vytvořil speciální dotazník, který absolvovalo 8 účastníků. Dotazník obsahoval 6 následujících otázek:

1. Jak byste ohodnotili vzhled a design uživatelského rozhraní naší mobilní webové aplikace?
   - Velmi se mi líbí
   - Líbí se mi
   - Neutrální názor
   - Nelíbí se mi
   - Vůbec se mi nelíbí
2. Bylo pro vás srozumitelné, jak navigovat v aplikaci a rezervovat lekce?
   - Ano, vše bylo naprosto srozumitelné
   - Většinou ano
   - Neutrální názor
   - Byly některé nejasné momenty
   - Ne, bylo těžké pochopit
3. Pokud jste měli problémy s pochopením fungování aplikace, můžete popsat, se kterými konkrétními prvky nebo funkcemi jste se setkali?
4. Co byste chtěli přidat nebo změnit v uživatelském rozhraní naší aplikace, aby se zlepšila jeho použitelnost a efektivita?
5. Jak byste hodnotili celkový uživatelský zážitek při použití naší aplikace?
   - Velmi uspokojivý
   - Uspokojivý

- Neutrální
- Neuspokojivý
- Velmi neuspokojivý

6. Máte nějaké další připomínky nebo návrhy na zlepšení uživatelského rozhraní naší aplikace?

Na základě výsledků testování, které lze najít v Příloze B, lze konstatovat, že celkový dojem z uživatelského rozhraní je pozitivní. Jedinné věci, které respondenti chtěli změnit, jsou velikost a barva textu. Některým se zdál text příliš malý a nevýrazný.

7.3 Testovací scénáře

V rámci testování uživatelského rozhraní byly také provedeny testy samotné aplikace podle následujících testovacích scénářů:

Registrace nového uživatele.

1. Zadejte telefonní číslo ve správném formátu (telefonní číslo musí být ve formátu +420).
2. Po vyplnění telefonního čísla se zobrazí tlačítko "Přihlásit se"- klikněte na něj.
3. Zadejte kód, který vám přišel na vaše telefonní číslo.
4. Zadejte své údaje - jméno a e-mail.
5. Klikněte na tlačítko "Dokončit registraci".

Očekávaný výsledek: uživatel je zaregistrován a přesměrován na hlavní stránku.

Rezervace a platba za skupinovou lekci

Vstupní podmínky: Uživatel je přihlašen.

Testovací údaje pro kreditní kartu

- **Číslo karty:** 4242 4242 4242 4242
- **Datum expirace:** libovolné budoucí datum
- **CVV kód:** libovolné tři číslice

1. Z hlavní stránky přejděte na stránku rezervace (2. záložka).
2. Vyberte skupinovou službu.

3. Vyberte datum konání skupinové lekce.
4. Vyberte dostupný čas konání skupinové lekce.
5. Na stránce s potvrzením lekce zkontrolujte všechny údaje o skupinové lekci.
6. Klikněte na posuvnik "Zaplatit online".
7. Klikněte na tlačítko "Zarezervovat".
8. V modálním okně zadejte testovací údaje kreditní karty.
9. Klikněte na tlačítko "Zaplatit".
10. Vraťte se na hlavní stránku.

Očekávaný výsledek: Uživatel si rezervoval místo na skupinové lekci a na hlavní stránce se objeví záznam o nadcházející lekci.

Rezervace individuální lekce

Vstupní podmínky: Uživatel je přihlašen.

1. Z hlavní stránky přejděte na stránku rezervace (2. záložka).
2. Vyberte požadovanou individuální službu.
3. Vyberte trenéra, se kterým chcete absolvovat lekci.
4. Zvolte datum, kdy se bude konat individuální lekce.
5. Vyberte dostupný čas, ve kterém se bude lekce konat.
6. Na stránce s potvrzením lekce zkontrolujte všechny údaje o individuální lekci.
7. Klikněte na tlačítko "Zarezervovat".
8. Vraťte se na hlavní stránku.

Očekávaný výsledek: Uživatel si úspěšně zarezervoval individuální lekci a na hlavní stránce se zobrazí informace o nadcházející lekci.

Smazání rezervace lekce

Vstupní podmínky: Uživatel již má rezervaci na nadcházející lekci.

1. Přihlaste se do uživatelského účtu.
2. Z hlavní stránky přejděte na stránku historie návštěv, která se nachází na třetí záložce.

3. Klikněte na nadcházející lekci, kterou chcete zrušit.
4. V modálním okně, které se objeví, stiskněte tlačítko "Zrušit".

Očekávaný výsledek: Uživatel úspěšně zrušil rezervaci lekce. Zrušená lekce se přidá do seznamu historie návštěv s označením "Zrušeno".

7.4 Výsledky testů a plán dalšího rozvoje

Během uživatelského testování jsme identifikovali následující problémy:

- Při pokusu o odstranění rezervace individuální lekce vrátí systém Altegio chybu, i když byl požadavek správně sestaven. Byla zaslána žádost o podporu.
- Na platformě Android push notifikace nevyžadují povolení pro odesílání upozornění - automaticky je označují jako povolené.
- Přesměrování po úspěšné platbě v mobilních aplikacích nefunguje správně a místo toho přesměrovává do prohlížeče - je třeba nastavit přesměrování zpět do aplikace.
- Na platformě Android je problematické ovládání posuvníku předplatného.
- V některých oblastech chybí zástupné symboly (placeholders).

Během práce na bakalářské práci jsem také nestihl implementovat některé z nezbytných požadavků:

- Apple Pay/Google Pay nejsou připojeny - platbu lze provést pouze pomocí karty.
- Není implementována nákup předplatného.
- Push notifikace jsou připojeny pouze pro platformu Android. Pro systém iOS je potřeba vývojářský účet, který stojí 100 dolarů ročně, a také nebyla nastavena systém odesílání push notifikací, které lze prozatím odesílat pouze přímo přes konzoli Firebase.

Výše uvedené body budou opraveny a implementovány ve druhé verzi aplikace.

Kapitola 8

Závěr

Původním cílem této bakalářské práce bylo předložit technické řešení pro řízení tanečního studia a v rámci analýzy byla také identifikována potřeba vyvíjet hybridní klientskou aplikaci.

V rámci této bakalářské práce byly provedeny následující úkoly:

1. Byla provedena analýza současného stavu tanečního studia, identifikovány jeho hlavní problémy a definována strategie pro jejich řešení.
2. Byla provedena analýza již existujících řešení na trhu, která by mohla vyhovovat našim potřebám na základě předem definovaných kritérií.
3. Bylo nalezeno technologické řešení (Altegio), které pokrývalo část řízení tanečního studia.
4. Byla identifikována potřeba vytvořit hybridní klientskou aplikaci prostřednictvím integrace se systémem Altegio.
5. Byla provedena analýza aplikace a navrh pro její vývoj.
6. Byla vyvinuta a otestována první verze aplikace.
7. Byl stanoven plán pro budoucí verze aplikace.

Cíl práce byl částečně splněn. Bylo nalezeno softwarové řešení - Altegio - pro správu tanečního studia a byla implementována první verze hybridní klientské aplikace. Nicméně v aplikaci ještě nejsou implementovány některé funkce, jako je nákup permanentek, a push notifikace nejsou zapojeny pro iOS verzi. Podrobněji o těchto aspektech diskutuji v závěru testování (viz 7.4).

Dalšími kroky v rámci této práce budou dopracování funkčnosti aplikace, oprava drobných chyb a její nasazení a vydání na App Store a Google Play Store.

Tato bakalářská práce měla mnoho přínosů, nejen z hlediska výsledků projektu, ale také z hlediska mého osobního a profesního rozvoje.

- Rozvoj dovedností: Tato práce mi umožnila se naučit, jak vyvíjet hybridní aplikace. Pochopil jsem základní principy a technologie hybridního vývoje, což mi otevírá nové příležitosti v oblasti mobilního vývoje.

- Rozšíření znalostí: Seznámil jsem se s řadou nových nástrojů a technologií, které se používají v hybridním vývoji. To mi poskytlo širokou škálu nástrojů, které mohu použít v budoucích projektech.
- Praktické použití Redux a Redux Saga: Poprvé jsem použil Redux a Redux Saga, což mi poskytlo cennou praxi a hlubší pochopení těchto nástrojů pro správu stavu aplikace.
- Seznámení s Firebase: Naučil jsem se pracovat s Firebase, což je významná dovednost, protože Firebase je široce používaný nástroj pro vývoj mobilních a webových aplikací.
- Vylepšení analytických schopností: Tento projekt mě také nutil k rozvoji a uplatnění mých analytických schopností.
- Efektivita provozu tanečního studia: Kromě předchozích bodů, tato bakalářská práce přinesla také hmatatelný přínos pro efektivní provoz mého tanečního studia. Díky řešení Altegio a vytvoření vlastní hybridní aplikace jsme byli schopni optimalizovat některé klíčové aspekty provozu, jako je řízení rezervací a komunikace s klienty. Toto nejen zlepšilo celkovou efektivitu provozu, ale také zlepšilo zážitek našich klientů z našich služeb.

Celkově řečeno, tato bakalářská práce mi poskytla cenné zkušenosti a dovednosti, které budu moci využít v budoucích projektech a profesním rozvoji.

Bibliografie

[1] CSM. *Why It's Important to Send Notifications to Your Customers*. Online. Dostupné z: https://www.customerservicemanager.com/why-its-important-to-send-notifications-to-your-customers/. [cit. 2023-01-15].

[2] LINKEDIN. *5 Reasons You Should Use Push Notifications for Your Business*. Online. Dostupné z: https://www.linkedin.com/pulse/5-reasons-you-should-use-push-notifications-your-business-buesnel/. [cit. 2023-01-15].

[3] ENGAGE.SO BLOG. *How Can Push Notification Marketing Improve Customer Engagement?*. Online. Dostupné z: https://engage.so/blog/how-can-push-notification-marketing-improve-customer-engagement/. [cit. 2023-01-20].

[4] STATCOUNTER, Global Stats. *Mobile OS Market Share Worldwide (Monthly) - Jan 2022 to Dec 2022*. Online. Dostupné z: https://gs.statcounter.com/os-market-share/mobile/worldwide/monthly/202201-202212-bar. [cit. 2023-03-14].

[5] ALTEGIO. *Pricing*. Online. Dostupné z: https://alteg.io/en/info/pricing. [cit. 2023-03-25].

[6] SIMPLYBOOK. *Pricing*. Online. Dostupné z: https://simplybook.me/en/pricing. [cit. 2023-03-25].

[7] ACUITYSCHEDULING. *Pricing*. Online. Dostupné z: https://acuityscheduling.com/signup.php. [cit. 2023-03-25].

[8] FORBES TECH COUNCIL. *15 Things Consumer Data Can Do for a Company*. Online. Dostupné z: https://www.forbes.com/sites/forbestechcouncil/2019/05/20/15-things-consumer-data-can-do-for-a-company/?sh=595a44874943. [cit. 2023-05-02].

[9] Software Testing Help. *UML Diagram Tutorial: Learn Types of UML Diagrams with Examples*. Online. Dostupné z: https://

www.softwaretestinghelp.com/uml-diagram-tutorial/. [cit. 2023-02-12].

[10] AMAZON WEB SERVICES. *The Difference Between Web Apps, Native Apps, and Hybrid Apps*. Online. Dostupné z: https://aws.amazon.com/compare/the-difference-between-web-apps-native-apps-and-hybrid-apps/. [cit. 2023-05-10].

[11] EPAM Solutions Hub. *Open Source Licenses: Definition, Types, and Comparison*. Online. Dostupné z: https://solutionshub.epam.com/blog/post/open-source-licenses-definition-types-and-comparison. [cit. 2023-05-12].

[12] SNYK. *Open Source Licenses*. Online. Dostupné z: https://snyk.io/learn/open-source-licenses/. [cit. 2023-05-12].

[13] IONICFRAMEWORK. *Web View*. Online. Dostupné z: https://ionicframework.com/docs/core-concepts/webview [cit. 2023-04-24].

[14] MOBILEAPPDAILY.COM. *Top 10 Best Hybrid App Frameworks for Mobile App Development*. Online. Dostupné z: https://www.mobileappdaily.com/best-hybrid-app-frameworks [cit. 2023-04-25].

[15] EL PASSION. *What is React Native and When to Use It?*. Online. Dostupné z: https://www.elpassion.com/blog/what-is-react-native-and-when-to-use-it [cit. 2023-04-29].

[16] HACKERNOON.COM. *Top 6 Hybrid Mobile App Development Frameworks for Both Android and iOS*. Online. Dostupné z: https://hackernoon.com/for-both-android-and-ios-top-6-hybrid-mobile-app-development-frameworks [cit. 2023-05-07].

[17] AGILE BUSINESS CONSORTIUM. *MoSCoW Prioritisation*. Online. Dostupné z: https://www.agilebusiness.org/dsdm-project-framework/moscow-prioririsation.html. [cit. 2023-05-18].

[18] RICHARDSON, Leonard; RUBY, Sam. *RESTful Web Services*. O'Reilly Media, 2008. ISBN: 9780596529260, ISBN: 0596529260 Dostupné z: https://books.google.cz/books?id=RQVu5YN59loC. [cit. 2023-05-12]

[19] REDUX. *Redux Fundamentals Tutorial: Part 3 - State, Actions, and Reducers*. Online. Dostupné z: https://redux.js.org/tutorials/fundamentals/part-3-state-actions-reducers. [cit. 2023-05-18]

[20] MOZILLA. *MVC (Model-View-Controller)*. Online. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/MVC. [cit. 2023-05-07]

[21] STONEHEM, Bill . *Google Android Firebase: Learning the Basics*. First Rank Publishing, Jun 29, 2016. ISBN: 9781365223075, ISBN: 9781535004466 Dostupné také z: https://books.google.cz/books/about/Google_Android_Firebase_Learning_the_Bas.html?id=JeeODAAAQBAJ&redir_esc=y [cit. 2023-05-19].

[22] BACKLINKO. *Stripe Users: How Popular Websites Are Using Stripe*. Online. Dostupné z: https://backlinko.com/stripe-users [cit. 2023-05-21].

[23] MERCHANT MAVERICK. *How Does Stripe Work?*. Online. Dostupné z: https://www.merchantmaverick.com/how-does-stripe-work/ [cit. 2023-05-17].

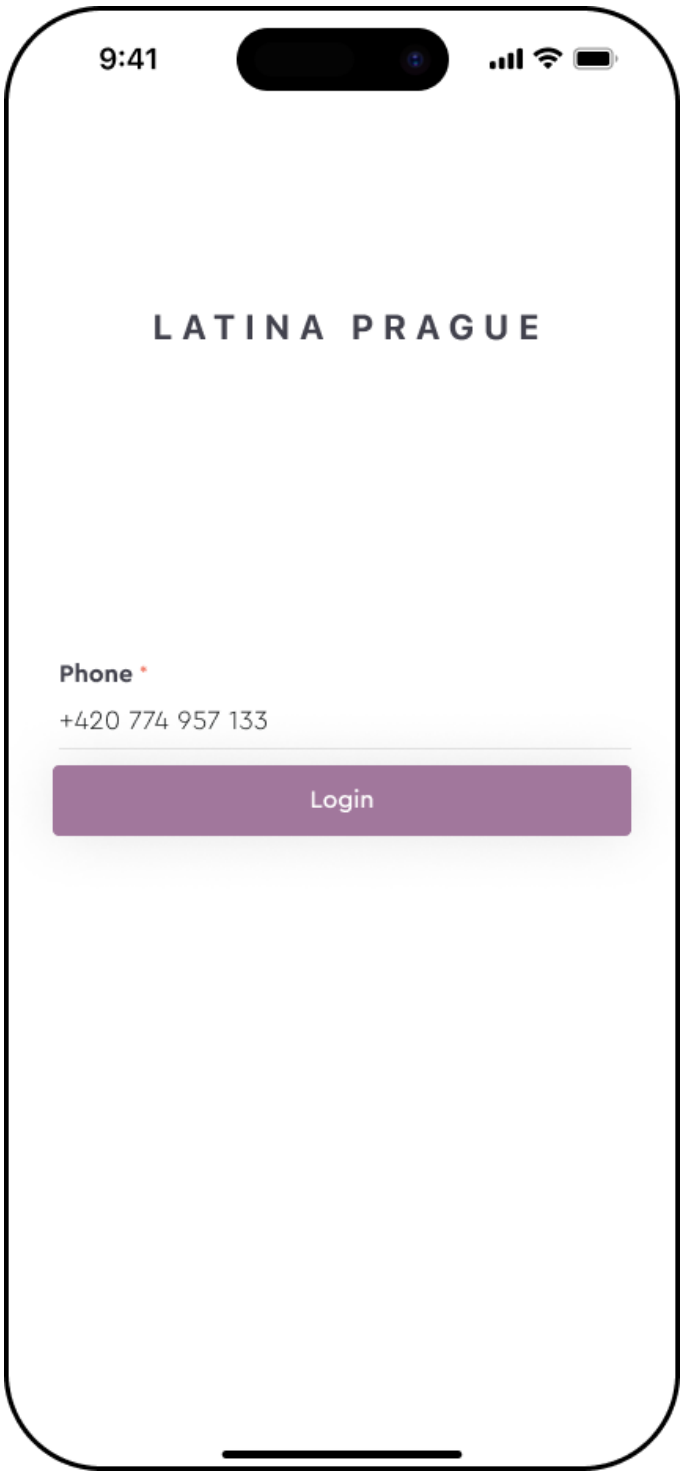
[24] DIRECTPAYNET. *Pros and Cons of Using Stripe: Is It Worth It?*. Online. Dostupné z: https://directpaynet.com/pros-and-cons-of-using-stripe-is-it-worth-it/ [cit. 2023-05-17].

[25] STRIPE. *Stripe API Reference*. Online. Dostupné z: https://stripe.com/docs/api [cit. 2023-05-17].

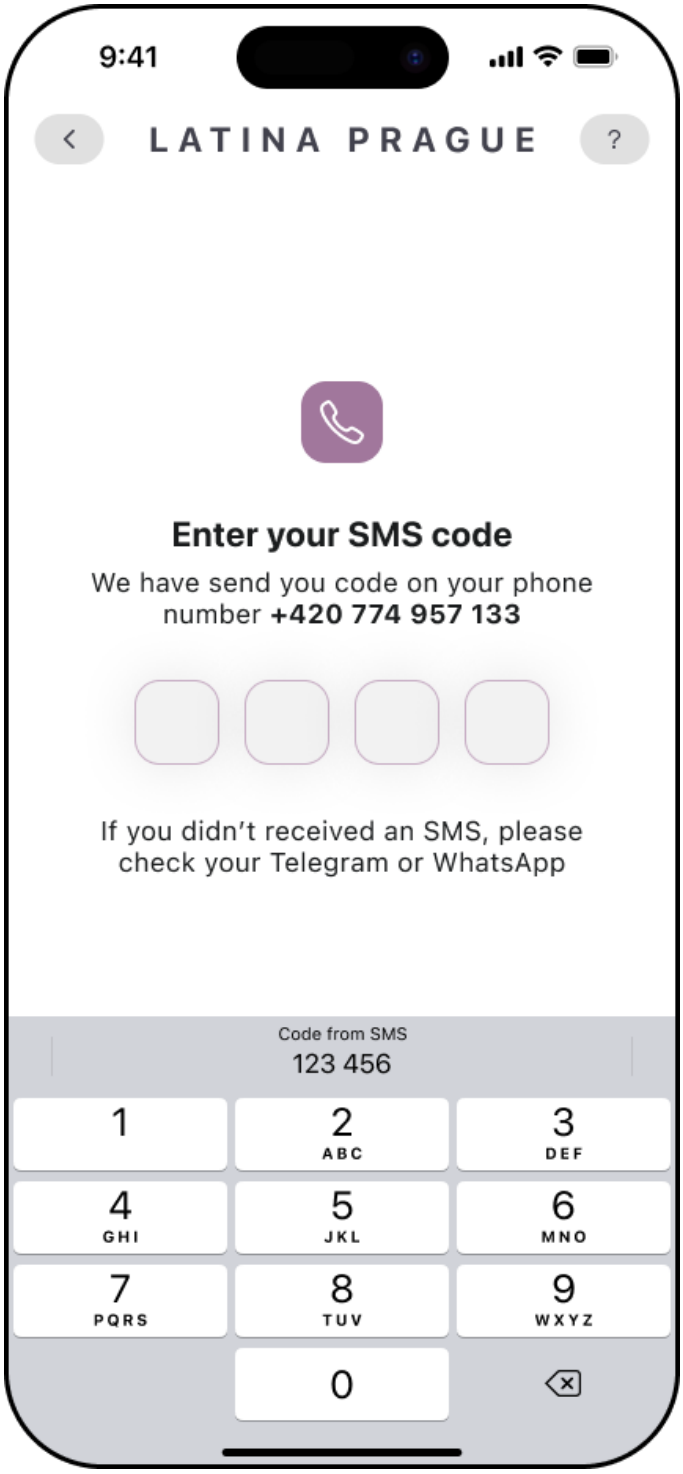
Přílohy

Příloha A

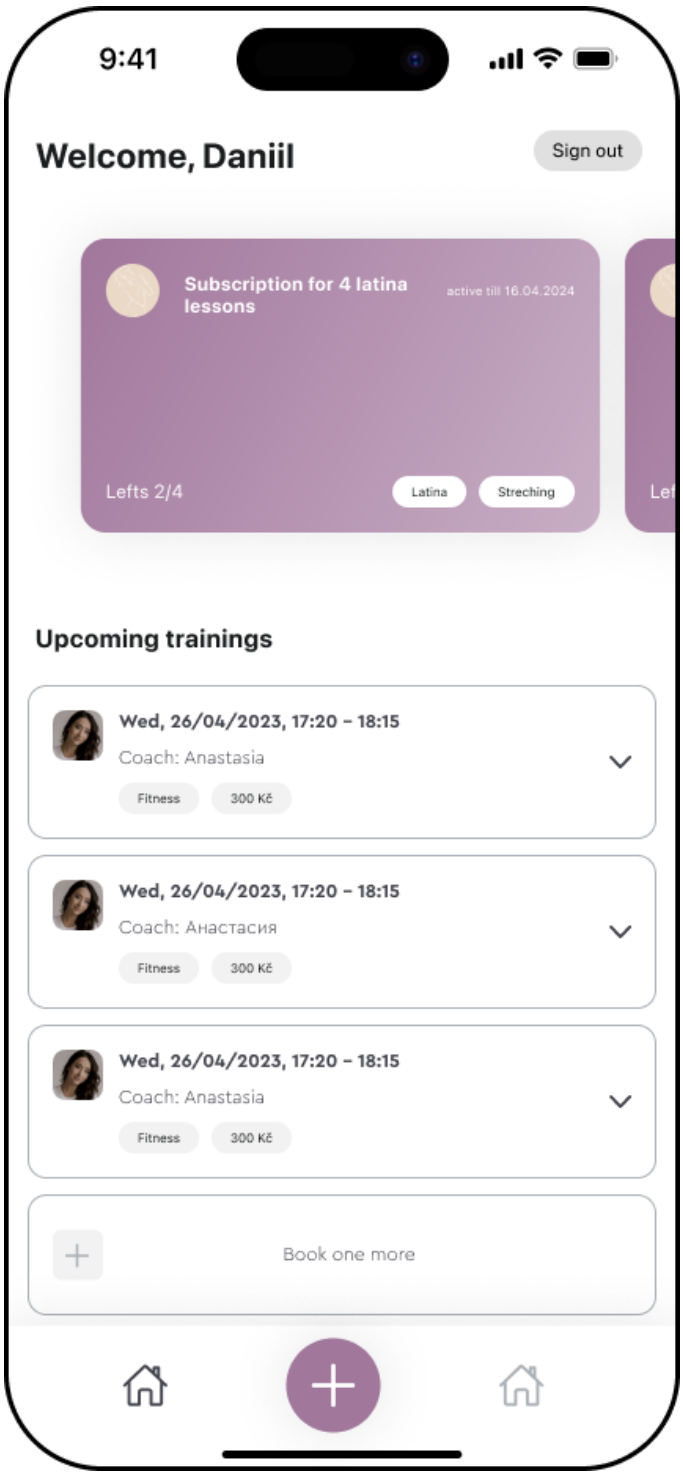
Ukázky uživatelského rozhraní



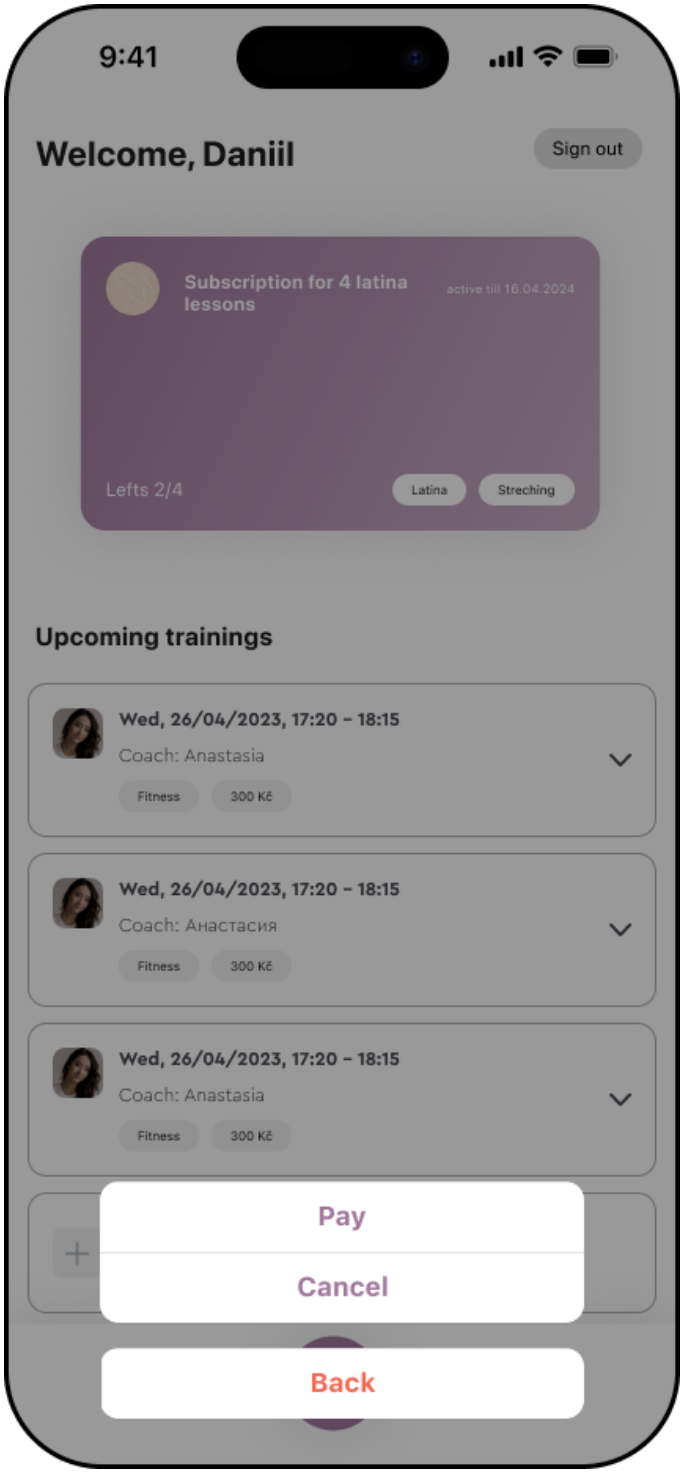
Obrázek A.1: Přihlašovací stránka



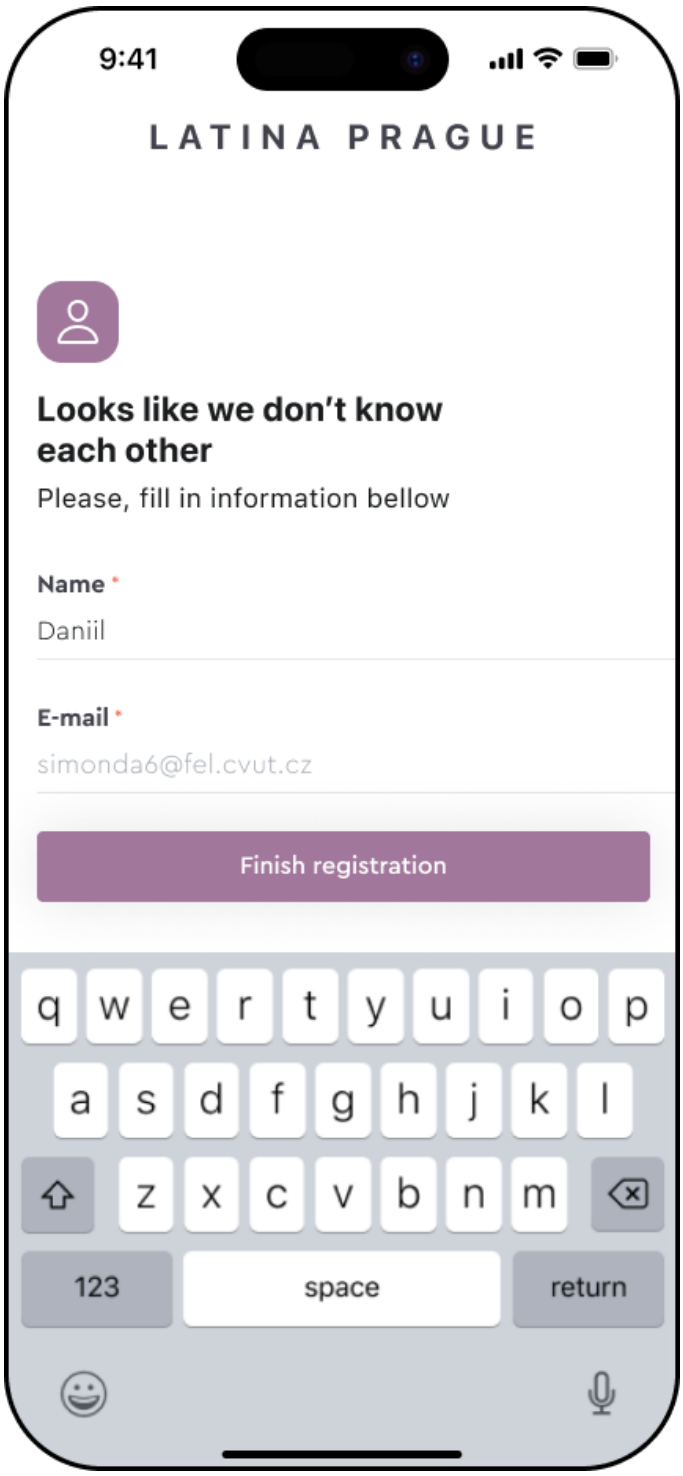
Obrázek A.2: Stránka ověřování telefonu



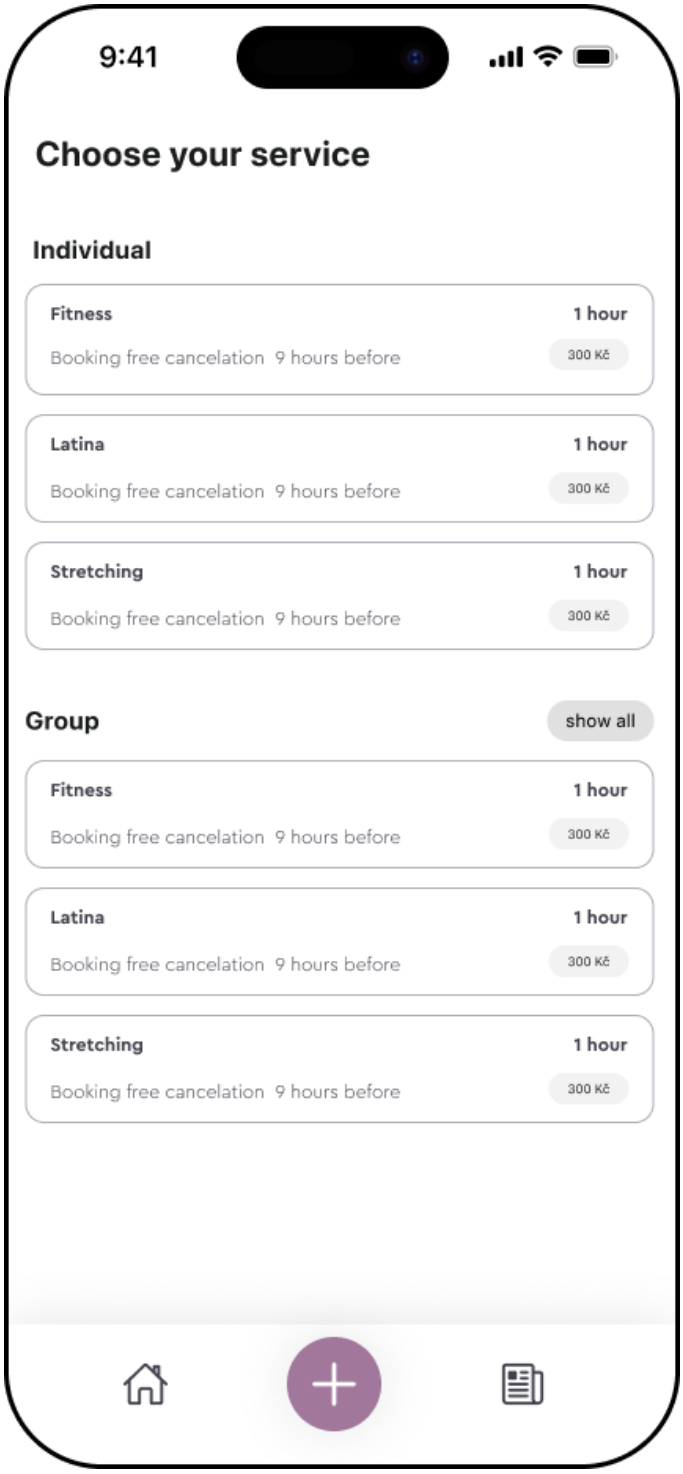
Obrázek A.3: Hlavní stránka



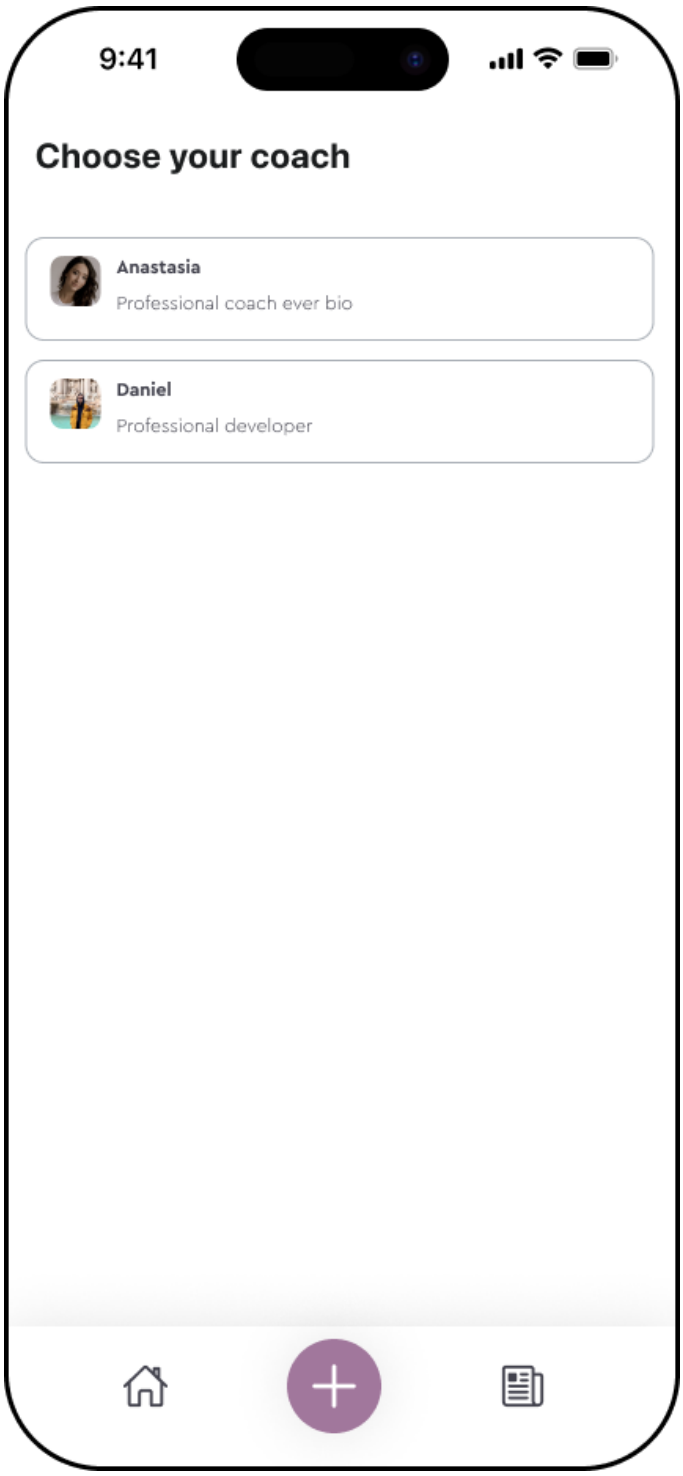
Obrázek A.4: Akce po kliknutí na lekci



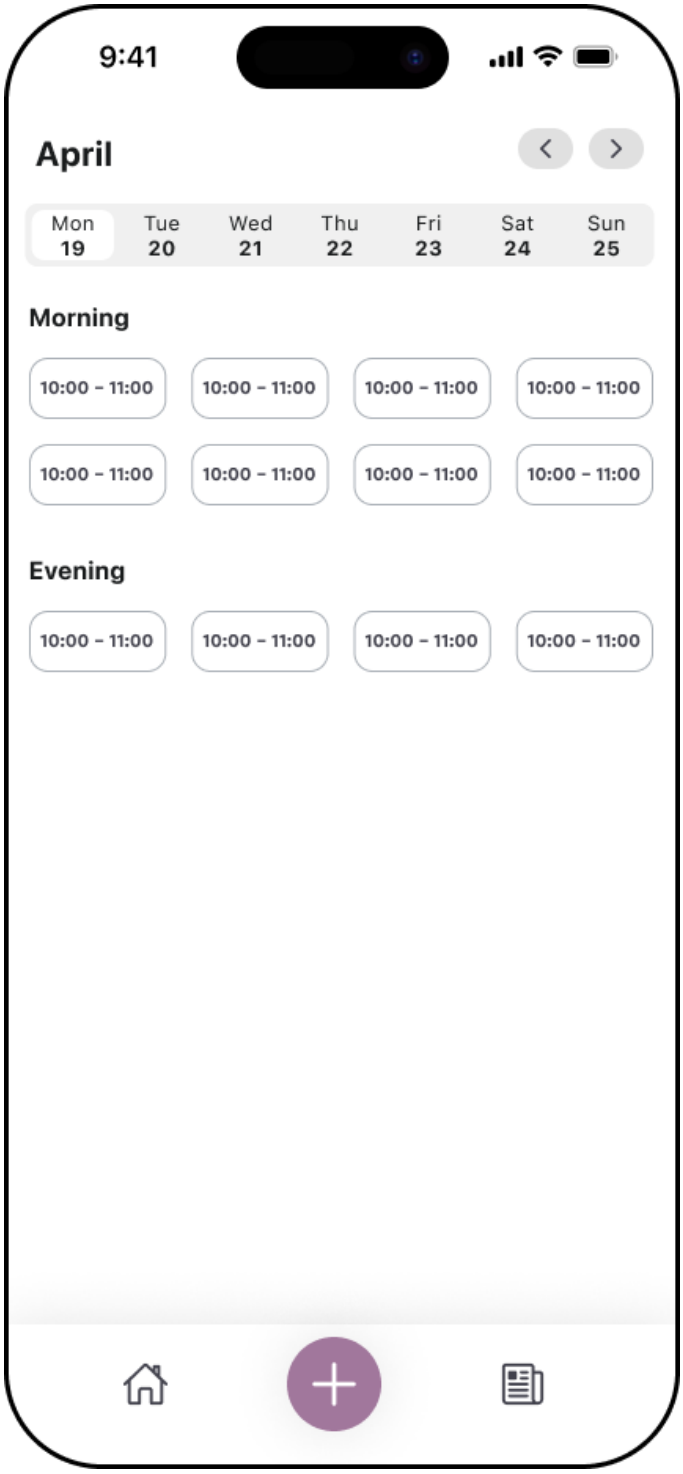
Obrázek A.5: Registrační stránka



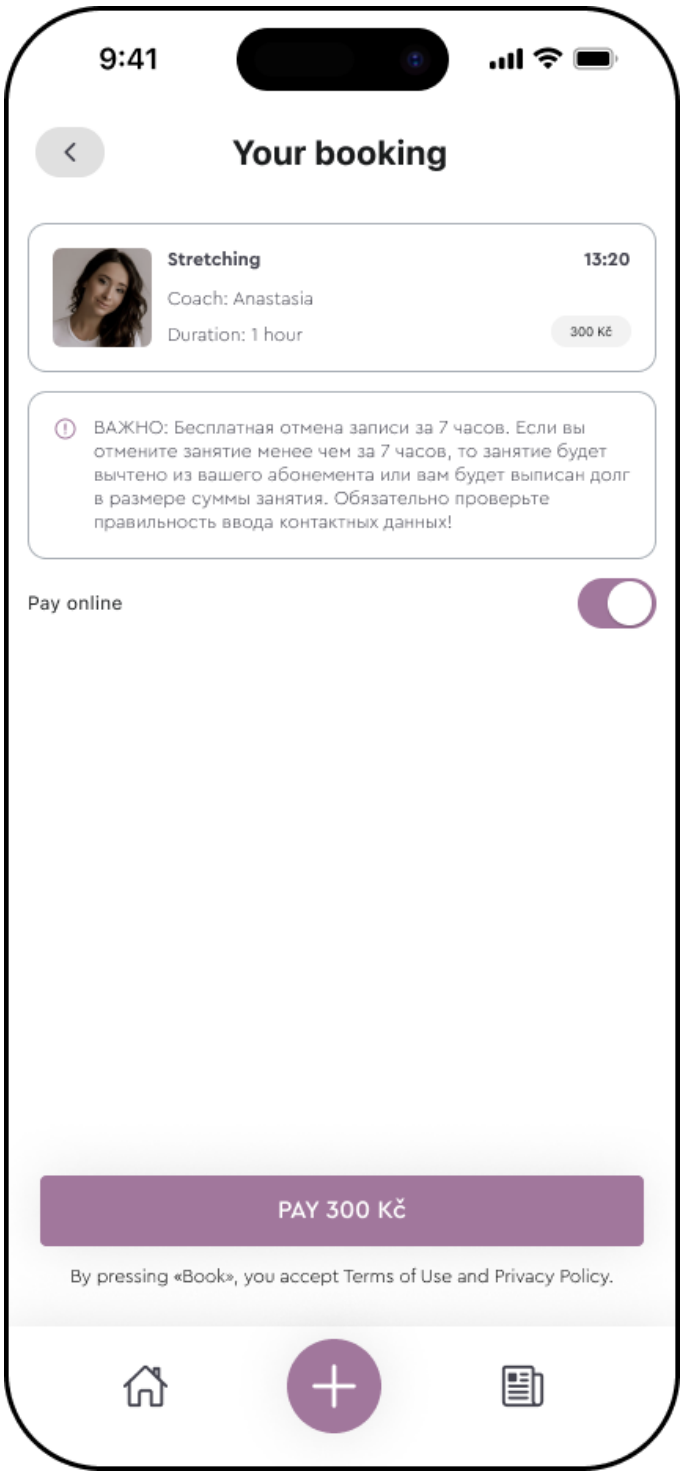
Obrázek A.6: Stránka s výběrem služby



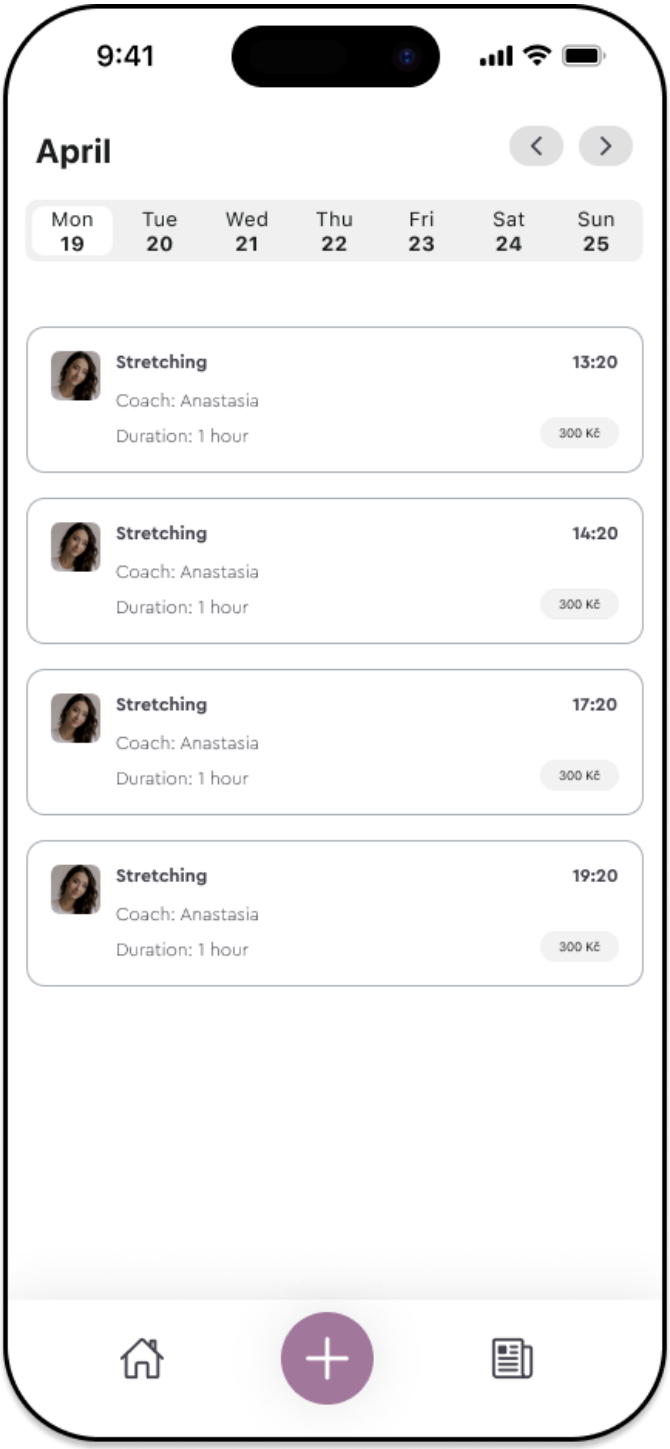
Obrázek A.7: Stránka s výběrem trenéra



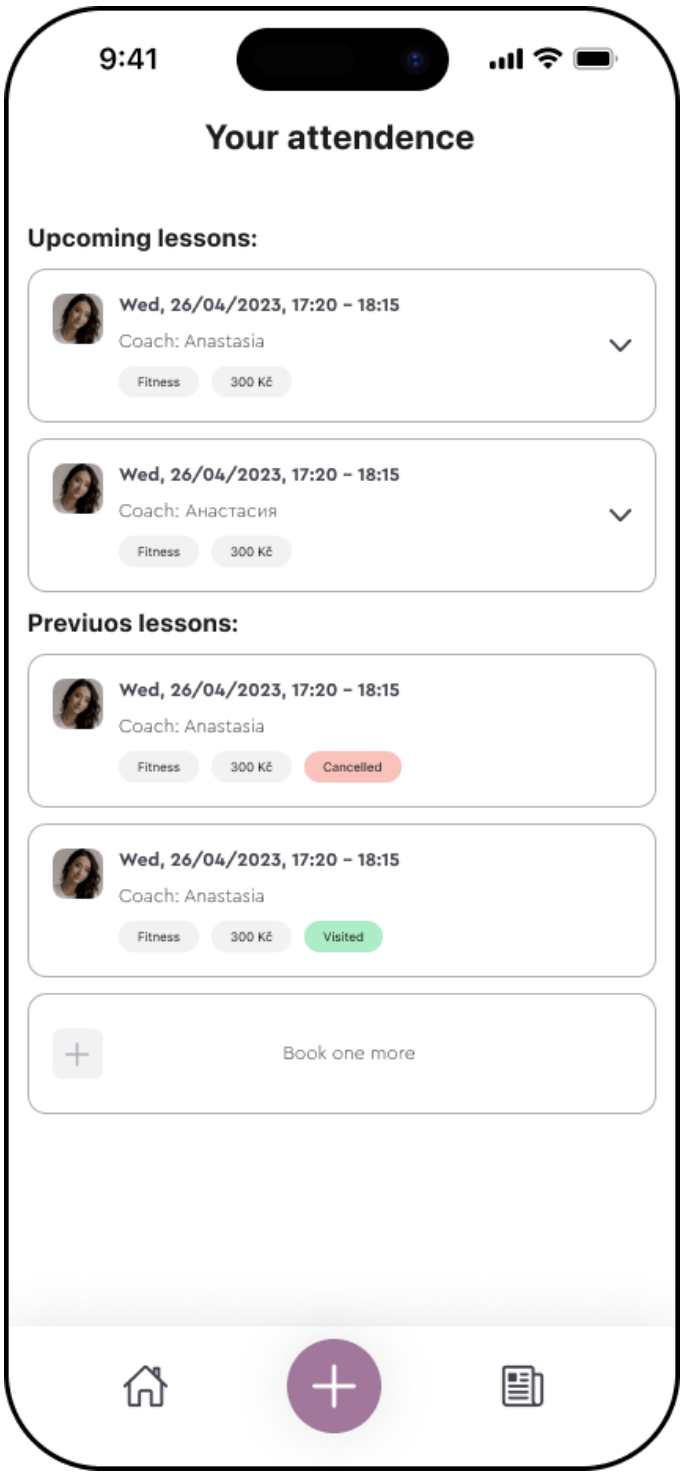
Obrázek A.8: Stránka s výběrem časového slotu



Obrázek A.9: Stránka pro potvrzení rezervace



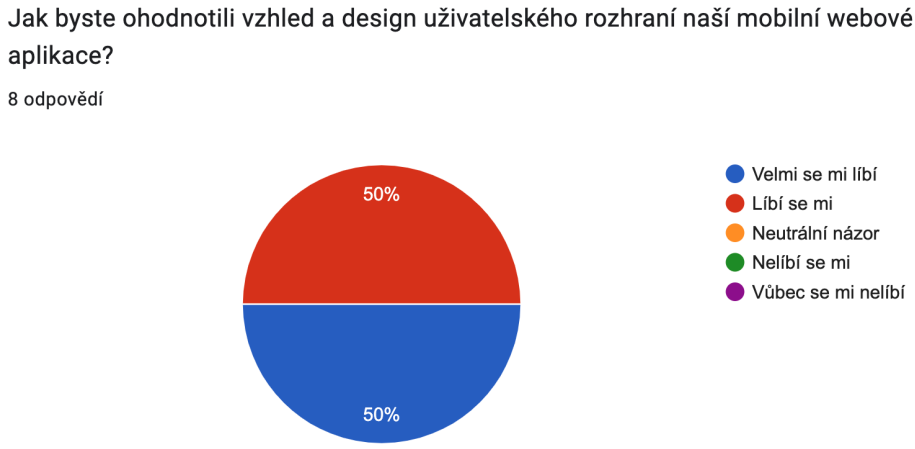
Obrázek A.10: Stránka pro výběr skupinové aktivity



Obrázek A.11: Stránka s historií návštěv

Příloha B

Výsledky průzkumu testování UI



Obrázek B.1: Odpovědi na otázku č. 1 dotazníku

Obrázek B.2: Odpovědi na otázku č. 2 dotazníku

Pokud jste měli problémy s pochopením fungování aplikace, můžete popsat, se kterými konkrétními prvky nebo funkcemi jste se setkali?

1 odpověď

Jen co by mělo znamenat ten dolní plus, jinak nic

Obrázek B.3: Odpovědi na otázku č. 3 dotazníku

Co byste chtěli přidat nebo změnit v uživatelském rozhraní naší aplikace, aby se zlepšila jeho použitelnost a efektivita?

2 odpovědi

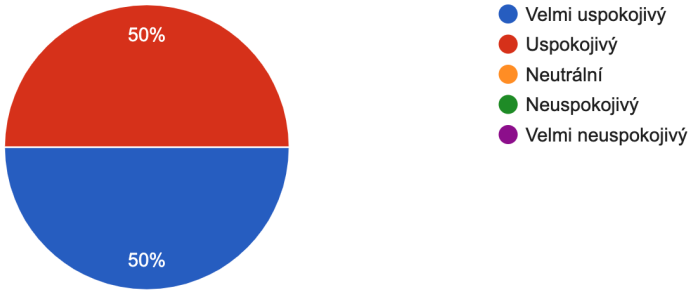
Zvětšit font nebo přidat možnost ho měnit

Aby byl hlavní text trošku větší, tmavší a čitelnější

Obrázek B.4: Odpovědi na otázku č. 4 dotazníku

Jak byste hodnotili celkový uživatelský zážitek při použití naší aplikace?

8 odpovědí



Obrázek B.5: Odpovědi na otázku č. 5 dotazníku

Máte nějaké další připomínky nebo návrhy na zlepšení uživatelského rozhraní naší aplikace?

2 odpovědi

Ne, jinak je tohle GUI docela moderní a minimalistické, což se mi líbí

Míchání zarovnání prvků uživatelského rozhraní je běžný problém a není cizím uživatelským rozhraním Apple, což je průšvih. Je lepší, když všechny prvky sledují stejné zarovnání. Ať už na levé, střední nebo pravé straně.

Obrázek B.6: Odpovědi na otázku č. 6 dotazníku