FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Nový frontend synchronizačního middleware Timer2Ticket
Student:	Jakub Čermák
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je realizace uživatelsky přívětivého a dobře použitelného rozhraní (frontendu) již existující synchronizační služby (middlewaru) Timer2Ticket.

Postupujte v těchto krocích:
1. Řádně prostudujte existující synchronizační nástroj Timer2Ticket. Neopomeňte prostudovat i magisterskou práci Ing. Víta Štefana.
2. Na základě zjištěných poznatků z aplikace zajistěte návrh vhodného nového uživatelského rozhraní pro uživatele tohoto synchronizačního nástroje, s důrazem na možnost vhodné volby tarifů.
3. Dle návrhů implementujte minimálně prototyp daného řešení.
4. Prototyp či výslednou realizaci podrobte vhodným testům.
5. Pokuste se zajistit budoucí rozvoj vašeho řešení, případně minimálně navrhněte vhodná vylepšení či optimální směřování do budoucna.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 1. února 2022 v Praze.

Bakalářská práce

NOVÝ FRONTEND SYNCHRONIZAČNÍHO MIDDLEWARE TIMER2TICKET

Jakub Čermák

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Jiří Hunka
10. května 2022

Odkaz na tuto práci: Čermák Jakub. *Nový frontend synchronizačního middleware Timer2Ticket.* Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především svému vedoucímu Ing. Jiřímu Hunkovi za konzultace a cenné rady, které mi při tvorbě bakalářské práce poskytoval. Dále bych rád poděkoval Benedeku Molnárovi a Irině Kara-Sel, kteří mi pomohli zejména při sběru požadavků a následném návrhu. Děkuji také společnosti Jagu s. r. o., která mi pomohla při tvorbě loga a zajištění licence pro propagační web. Děkuji samozřejmě i všem testerům, rodině a každému, kdo mě při práci podporoval a motivoval.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 10. května 2022 .....

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací prototypu nového uživatelského rozhraní existující synchronizační služby Timer2Ticket. Věnuje se návrhu takového řešení, které umožní službu zpoplatnit a nabídnout uživatelům v rámci jednotlivých členství. Jejich návrh je také součástí této práce. Návrh se zakládá na provedené analýze pravděpodobného zákazníka, souvisejících nástrojů s touto službou a detailním sběru požadavků vycházejících ze strany zadavatele, provozujícího tuto službu, i současných uživatelů. Na vytvořený návrh navazuje popis technických specifik implementace prototypu. Na závěr je popsáno uživatelské testování připraveného prototypu a zpětná vazba od těchto uživatelů, na základě čehož jsou pak popsána možná vylepšení a budoucí rozvoj. Kromě nového uživatelského rozhraní se práce věnuje také návrhu souvisejícího propagačního webu této aplikace a vhodné volbě použitých technologií. Je také doplněna o doporučení k týmovému vývoji.

Klíčová slova webová aplikace, frontend, uživatelské rozhraní, Timer2Ticket, propagační web, synchronizace času, členství, platební brána, Jagu s. r. o., JavaScript, Vue.js, Vuetify, October CMS

Abstract

This bachelor thesis deals with the design and implementation of a prototype of a new user interface of the existing synchronization service Timer2Ticket. It is dedicated to the design of such a solution, which will enable the service to charge for the service and offer it to users within individual memberships. Their design is also part of this work. The design is based on an analysis of the probable customer, tools, which are related to this service, and a detailed collection of requirements arising from the contracting authority operating this service, as well as current users. The created design is followed by a description of the technical specifics of the prototype implementation. Finally, user testing of the prepared prototype and feedback from these users are described, based on which possible improvements and future development are then described. In addition to the new user interface, the work also deals with the design of the related promotional website of this application and the appropriate choice of technologies used. It is also supplemented by recommendations for team development.

Keywords web application, frontend, user interface, Timer2Ticket, promotional website, time synchronization, membership, payment gateway, Jagu s. r. o., JavaScript, Vue.js, Vuetify, October CMS

Seznam zkratek

FIT	Fakulta informačních technologií
ČVUT	České vysoké učení technické
SP2	Softwarový týmový projekt 2
SP1	Softwarový týmový projekt 1
API	Application Programming Interface
FAQ	Frequently Asked Questions
SI2	Softwarové inženýrství 2
SI1	Softwarové inženýrství 1
FPx	Funkční požadavek číslo X
NPx	Nefunkční požadavek číslo X
USA	United States of America
PS	Počítačový software
ITS	Informační technologie a služby
V	Vzdělávání
MR	Marketing a reklamy
NUR	Návrh uživatelského rozhraní
Lo-Fi	Low Fidelity
Hi-Fi	High Fidelity
BMPR	Balsamiq Mockups Projects
PDF	Portable Document Format
JPG	Joint Photographic Expert Group
PNG	Portable Graphics Format
SVG	Scalable Vector Graphics
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
SEO	Search Engine Optimization
JSX	JavaScript XML
XML	Extensible Markup Language
IDE	Integrated Development Environment
PJS.1	Programování v JavaScriptu
CMS	Content Management System
WYCIWYG	What You Code Is What You Get
ČR	Česká republika
EU	Evropská unie
IT	Information Technology
UML	Unified Modeling Language
UC	Use Case
DIČ	Daňové identifikační číslo
PSČ	Poštovní směrovací číšlo
DPH	Daň z přidané hodnoty
OSP	Open-source project
CLI	Command Line Interface
UI	User Interface
IP	Internet Protocol
MFA	Multi-Factor Authentication
URL	Uniform Resource Locator
PA1	Programování a algoritmizace 1
PA2	Programování a algoritmizace 2
AM	Ante meridiem
PM	Post meridiem

Úvod

V současné době má naprostá většina společností, zejména pak v IT, potřebu využívat ke správě svých projektů některý z projektových nástrojů. Tyto nástroje jsou primárně určeny k řízení projektů, které se obvykle skládají z dílčích úkolů, na kterých pracují konkrétní lidé. Pokud se pak jedná například o projekt, který je placený v závislosti na čase, který si pracovníci vykážou, plyne z toho nutně potřeba čas detailně evidovat.

To však souvisí s nutností přesně měřit strávený čas, nicméně projektové nástroje pro to typicky nejsou ideální. Řešení tohoto problému na druhé straně poskytují nástroje na měření času specializované. Uživatel je schopen si pomocí nich naprosto přesně změřit odpracovaný čas, i když často mění kontext své práce (tj. štěpí svoji pozornost mezi více projektů), a následně ho ručně přenést do svého projektového nástroje.

Nicméně ručním přenášením pak daná osoba ztrácí další drahocenný čas. Zároveň s touto činností roste možnost přenést čas chybně. Ať už jde tedy o pracovní projekt či touhu po detailní správě vlastního času, přirozeně se objevuje otázka, zda by tato rutinní práce nešla zefektivnit a eliminovat možnost chyb.

Odpovědí na tuto otázku pak může být právě nástroj Timer2Ticket, který umožní automaticky synchronizovat čas mezi výše zmíněnými projektovými nástroji a nástroji pro měření času. Mimo tento nástroj sice existují služby, které implementují řešení tohoto problému, nicméně se jedná o řešení pro konkrétní dvojice systémů jako jsou například Toggl Track a Jira [1], nikoliv o obecné řešení, jak sám uvádí Ing. Vít Štefan ve své diplomové práci [2], v rámci níž nástroj Timer2Ticket vznikl.

Výše zmíněný autor Ing. Vít Štefan úspěšně zprovoznil Timer2Ticket pro synchronizaci mezi projektovým nástrojem Redmine a nástrojem pro měření času Toggl Track, kterým se bude práce později věnovat v kapitole 2.2 o souvisejících nástrojích. Aplikace byla nasazena do provozu na jaře roku 2021, kdy jsem se s ní poprvé setkal na FIT ČVUT v rámci předmětu Softwarový týmový projekt 1, kde jsem se podílel na tvorbě portálu pro podporu výuky předmětu Databázové systémy, vyučovaného také na FIT ČVUT. Blíže jsem se s ním však seznámil až v průběhu léta a následném pokračujícím předmětu Softwarový týmový projekt 2, kdy jsem již nástroj plně využíval a spatřoval prostor pro rozvoj.

Systém byl Ing. Vítem Štefanem navržen co nejobecněji, aby byl v budoucnu schopen do synchronizace začlenit i další systémy kromě dvou výše zmíněných. Webový klient, který však uživateli slouží k ovládání synchronizace, není v současné době připraven na toto rozšíření. V této práci proto dojde k návrhu a realizaci prototypu nového uživatelského rozhraní výše zmíněného nástroje na základě kroků, které jsou detailněji popsány v kapitole 1. Uživatelské rozhraní bude navrženo s ohledem na možnost provádět synchronizaci mezi větším množstvím dvojic synchronizačních nástrojů a nebude se omezovat na dva výše zmíněné, nicméně samotné přidání dalších nástrojů již součástí této práce nebude. Nové řešení bude také klást důraz na možnost vhodné volby tarifů, tedy jednotlivých plánů členství, které si budou moci uživatelé této aplikace zvo-

lit, jak je tomu běžně u jiných placených nástrojů. V současné době je tento nástroj dostupný k použití plně zdarma.

Výsledkem této práce navazující na diplomovou práci [2] bude prototyp nového uživatelského rozhraní, který bude podroben uživatelskému testování pro zjištění možných dalších vylepšení a nedostatků. Na závěr práce rozeberu, jaké by bylo ideální budoucí směřování projektu.

Kapitola 1
Cíle

Cílem práce je návrh a prototypová realizace uživatelsky přívětivého a dobře použitelného rozhraní (frontendu) již existující synchronizační služby (middlewaru) Timer2Ticket, která zajišťuje synchronizaci mezi nástroji pro měření času a projektovými nástroji.

Dílčím cílem je analyzovat existující synchronizační nástroj Timer2Ticket, diplomovou práci Ing. Víta Štefana na FIT ČVUT a nástroje blízké uživateli nástroje Timer2Ticket. Dalším cílem je na základě zjištěných poznatků z analýzy a požadavků zákazníka navrhnout vhodné nové uživatelské rozhraní pro uživatele tohoto synchronizačního nástroje. Nové uživatelské rozhraní bude navrženo s důrazem na možnost vhodné volby tarifů. Tarify budou jednotlivé plány členství, které si bude moci uživatel této aplikace zvolit. Kromě toho bude při návrhu kladen důraz na umožnění synchronizace mezi více než dvěmi nástroji. Cílem této práce bude i volba vhodných technologií pro zjednodušení vývoje této aplikace.

Dalším dílčím cílem je implementovat prototyp. Provedu zde vysvětlení zajímavých částí, se kterými jsem se při implementaci setkal. Cílem je také podrobení prototypu testům, na jejichž základě budou identifikovány nedostatky vytvořeného řešení a možná vylepšení. Posledním cílem je stanovení možného rozvoje nového řešení a jeho optimální směřování do budoucna.

Kapitola 2

Analýza a sběr požadavků

V této kapitole nejprve zanalyzuji současný stav nástroje Timer2Ticket původně vytvořeného Ing. Vítem Štefanem v rámci jeho diplomové práce na FIT ČVUT a následně prozkoumám nástroje blízké uživatelům, na které má tento synchronizační nástroj cílit. Dále popíšu podobu spolupráce se softwarovým týmem z předmětu SP2, který se podílel na práci od sběru požadavků přes návrh až po naprostý začátek implementace. Na závěr na základě provedené analýzy určím základní požadavky na nový frontend a vytvořím podobu ideálního zákazníka.

2.1 Současný stav nástroje Timer2Ticket

V této podkapitole se zaměřím na detailní analýzu současného stavu webového klienta synchronizačního nástroje Timer2Ticket [3] za účelem zmapování vytvořených procesů, na základě kterých mimo jiné později navrhnu nové uživatelsky přívětivé řešení nedostatků.

2.1.1 Registrace a přihlašování uživatelů

Webový klient [3] je rozdělen na část pro přihlášené a nepřihlášené uživatele. Nepřihlášený uživatel se po příchodu do klienta objeví na přihlašovací stránce, na které je textovým popisem informován, že klient slouží právě k jednouchému nastavení synchronizace mezi projektovými aplikacemi a nástroji pro měření času.

Následně má možnost zadat své přihlašovací údaje a stisknutím tlačítka přejít do svého účtu. Přihlášení probíhá pomocí emailu a hesla, které má v současnosti nastavené dva povinné parametry – délku minimálně 8 znaků a přítomnost čísel. Zároveň má uživatel možnost, nechat si zapamatovat své údaje na později. Pokud uživatel zapomene heslo, má možnost si nechat zaslat nové na svůj email. Odtud bude odkazem validním po dobu šedesáti minut přesměrován do aplikace, kde dostane možnost změnit své heslo a následně se do aplikace opět přihlásit.

V případě, že uživatel ještě nedisponuje uživatelkým účtem, má možnost přejít na stránku s registrací. Ta probíhá ve dvou krocích. Nejprve uživatel zadá email, na který se mu pošle odkaz validní po dobu dvou dnů, přes který se opět dostane do aplikace, ve které dokončí registraci zvolením hesla. Jakmile uživatel tyto kroky dokončí, je přesměrován na přihlašovací stránku, odkud se může opět přihlásit a přejít do svého účtu.

V této části shledávám jako klíčové nedostatky zejména detailnější informovanost uživatele o účelu synchronizačního nástroje, čímž by se pravděpodobně dala získat řada nových uživatelů. Osobně bych zde také ocenil možnost přihlášení prostřednictvím některé ze sociálních aplikací jako je Google či Facebook.

2.1.2 Konfigurace nástrojů

Pokud je uživatel přihlášen poprvé či opakovaně, avšak nemá nakonfigurované nástroje, které mají být synchronizovány, aplikace ho vyzve ke konfiguraci těchto nástrojů. V případě nezájmu o konfiguraci je v tuto chvíli v rámci hlavního menu prostor pro odhlášení či změnu hesla, která probíhá obdobně jako při zapomenutí hesla. Aby uživatel mohl synchronizační nástroj použít musí proto projít konfigurací. V rámci ní se uživatel může kdykoliv vracet a je pravidelně notifikován prostřednictvím chybové lišty, pokud při vyplňování došlo k chybě.

V prvním kroku konfigurace dojde ke zvolení nástrojů, které se budou synchronizovat. V současné době jsou k dispozici právě pouze dříve zmíněné nástroje Redmine a Toggl Track. Samotný synchronizační nástroj byl však vytvářen s ohledem na budoucí přidání dalších nástrojů. [2]

V kroku druhém pak dojde ke konkrétní konfiguraci nástroje Redmine. Uživatel zde poskytne API klíč, který aplikaci umožní zastupovat ho v rámci jeho projektového systému (například za něj přidávat čas k úkolu). Dále udá přístupový bod API, který konkretizuje službu, se kterou má synchronizační nástroj komunikovat. Takový bod má každá organizace, která Redmine provozuje, specifický a jde vlastně o základ adresy. Po vyplnění těchto prvků uživatel nechá zkontrolovat zadané údaje, čímž synchronizační nástroj ověří, že se se zadanou službou lze korektně spojit. Kromě toho se také načtou možné výchozí aktivity pro časové záznamy. Zvolení aktivity zde má sloužit jako zjednodušení pro uživatele. Pokud v rámci svého měření času neuvede chtěnou aktivitu, přiřadí se mu zde zvolená výchozí. Po výběru aktivity může uživatel pokročit k dalšímu kroku.

Ve třetím kroku uživatel provádí konfiguraci nástroje Toggl Track. Ze stejného důvodu jako v přechozí konfiguraci uživatel zadá API klíč svého účtu v Toggl Tracku. Po úspěšném ověření korektnosti spojení se načtou dostupné pracovní prostory (tzv. *workspacy*), z nichž uživatel musí zvolit ten, pro který chce synchronizaci provádět. Jakmile je vše nastaveno, lze pokročit k předposlednímu kroku.

Ve čtvrtém kroku dochází k volbě načasování synchronizace. Uživatel zde má možnost rozlišit mezi synchronizací časových záznamů a samotných konfiguračních objektů jako jsou například projekty či úkoly. Aby se usnadnil celý proces, je přednastavena každodenní synchronizace jednou denně. Je však možnost zvolit pouze některé dny či naopak provádět synchronizaci až každou hodinu.

V kroku posledním pak uživatel pouze potvrdí, že chce zahájit zvolenou synchronizaci, čímž se přesune na hlavní stranu aplikace, kde bude informován o započetí probíhající první synchronizace konfiguračních objektů. [3]

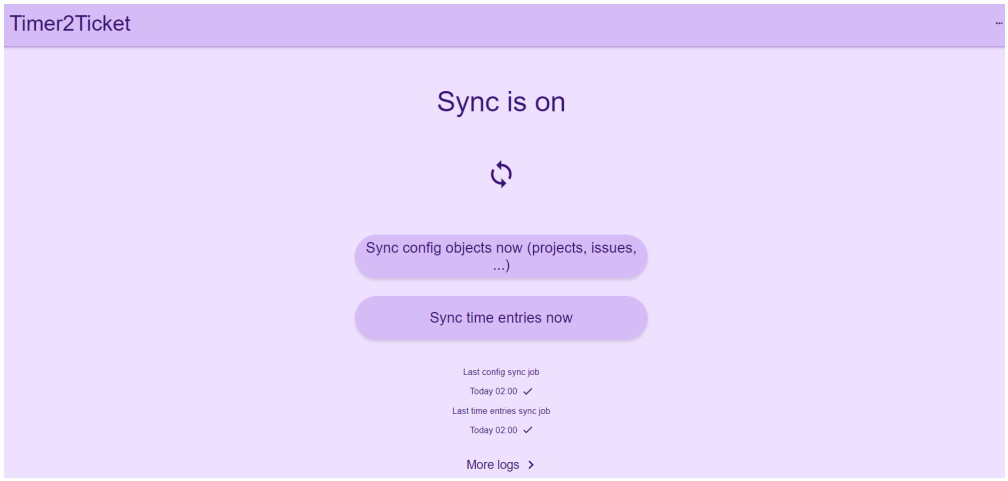
Při průchodu konfigurací jsem si uvědomil další dvě důležitá místa, ve kterých nacházím prostor pro zlepšení. Prvním z nich je využití prostoru. Současné řešení bylo vytvořeno minimalisticky s důrazem na vertikalitu aplikace, což mělo usnadnit zobrazování na menších zařízeních jako je například telefon. V tomto ohledu vertikalita problém vyřešila, nicméně zbytečně nutí uživatele například na počítači, aby se v rámci jednotlivých kroků posouval po stránce, i když by se vše vešlo na jednu obrazovku. Zároveň si myslím, že by bylo vhodné mít hlavní menu (s možností odhlášení) při vertikálním pohybu pevně ukotvené v horní části obrazovky.

Druhý nedostatek se týká samotné myšlenky přidání dalších synchronizačních nástrojů. Až v budoucnu přibude možnost provádět synchronizaci mezi jinými nástroji než těmi dostupnými nyní, nástroj je připraven pojmout volbu jiné synchronizační dvojice, nicméně současný klient dle mého názoru nemá připravené uživatelské rozhraní pro větší množství synchronizačních dvojic. Tomuto problému se budu později věnovat v samotném návrhu v kapitole 4.

2.1.3 Hlavní strana synchronizace

Pokud se do aplikace přihlásí uživatel, který již prošel konfigurací nástrojů, které budou synchronizovány, objeví se okamžitě na hlavní stránce, která pokrývá veškeré možnosti, které v rámci

Obrázek 2.1 Původní stav hlavní stránky webového klienta nástroje Timer2Ticket – horní část



svého účtu má. Popisovaná hlavní stránka je na obrázcích 2.1, 2.2 a 2.3, vytvořených ze současné aplikace [3] pro účely snazšího pochopení.

V horní oblasti získá uživatel informaci o tom, zda má synchronizaci v současné chvíli zapnutou. Naopak tlačítko s možností vypnout synchronizaci v případě, že je zapnutá, či zapnout synchronizaci v případě, že je vypnutá, se nachází až ve spodní části dlouhého verikálního prostoru. Neboť se jedná o podstatný krok, před jeho provedením ho uživatel musí v zobrazeném okně potvrdit. Zde si myslím, že by bylo pro uživatele příjemnější mít prvky se souvisejícím významem poblíž.

Pod informací o tom, zda synchronizace běží, jsou pro uživatele připraveny tlačítka, která mu umožní okamžitou synchronizaci. Opět se dělí na samotné časové záznamy a konfigurační objekty. Pod nimi pak uživatel nalezne oznámení o poslední proběhlé synchronizaci, opět rozděleno na rozdílné typy synchronizace. Součástí je jejich datum, čas a informace o úspěšnosti synchronizace či hlášení o neúspěchu v případě, že nastal nějaký problém.

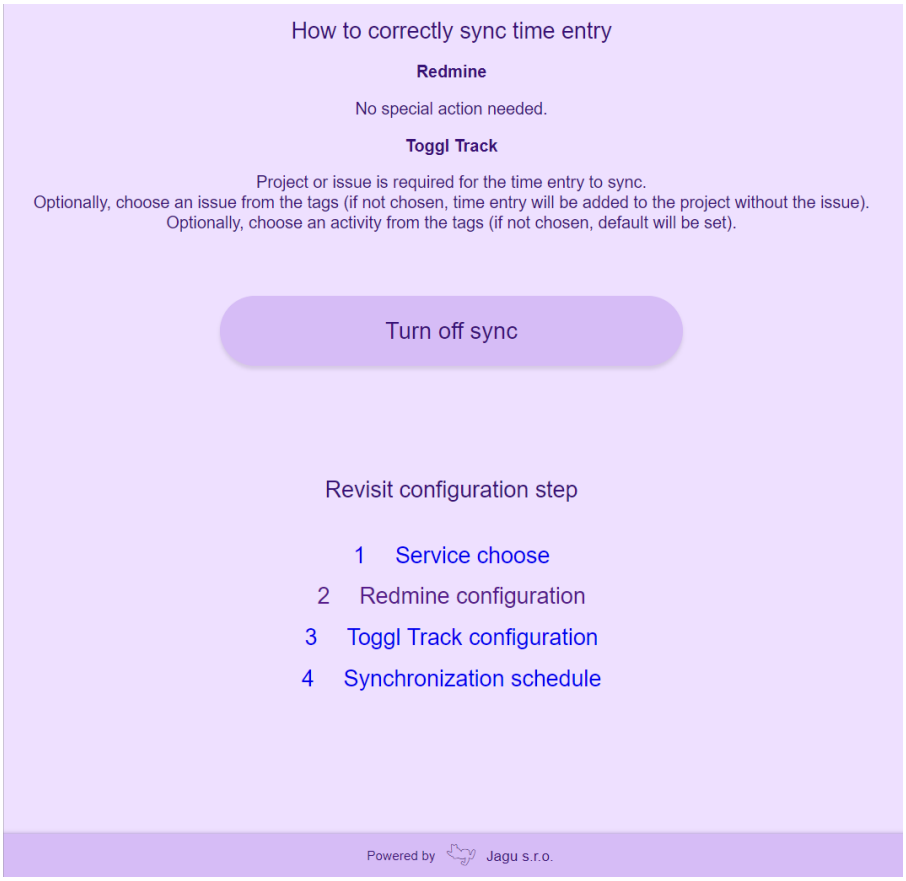
Následující prostor je věnován záložce, po jejímž rozkliknutí se uživateli zobrazí tabulka ve formě okna se záznamy (tzv. *logy*) o proběhlých synchronizacích. V současném řešení tabulka může obsahovat maximálně sto záznamů, které se musí vztahovat k synchronizacím za posledních devadesát dní. Po této době jsou totiž záznamy v současném řešení mazány s ohledem na jejich rychle rostoucí množství. Tabulka obsahuje detailnější informace. Konkrétně se jedná o datum a čas, na který byla synchronizace načasována, datum a čas, kdy byla dokončena, zda se jednalo o synchronizaci konfiguračních objektů či časových záznamů, zda původem byl přednastavený čas v samotném nástroji či akce uživatele, který prováděl okamžitou synchronizaci, a nakonec informace o úspěšnosti, stejně jako u oznámení o poslední proběhlé synchronizaci. [2, 3]

Další prostor je věnován informování uživatele o tom, jak má postupovat při používání jednotlivých nástrojů, které se rozhodl synchronizovat. Dle mého názoru je zde tato informace nevhodně umístěná, neboť se netýká samotné aplikace a bylo by vhodné ji vyčlenit stranou, například do nějakého instruktážního videa či manuálu.

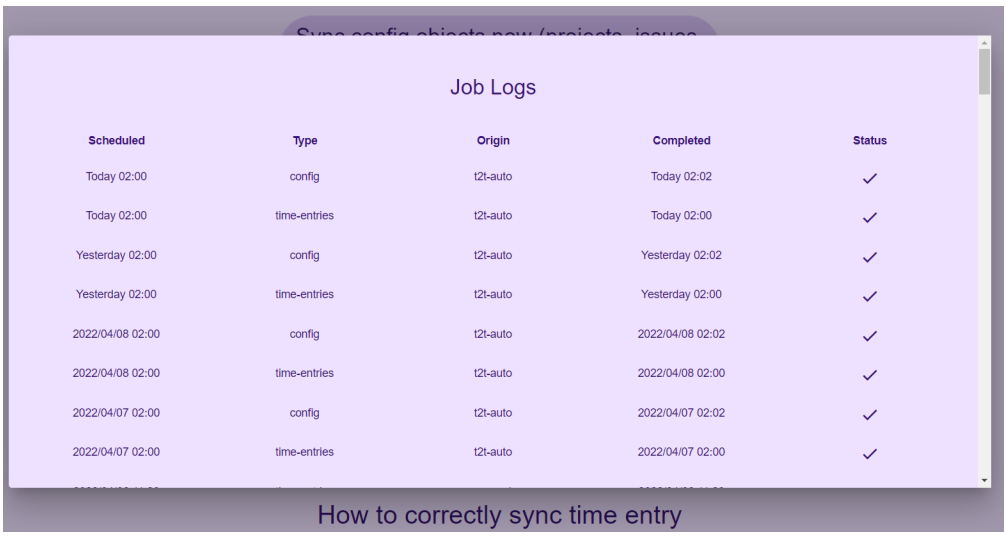
V poslední části hlavní stránky je k nalezení přehled konfiguračních kroků, které je odsud možné znovu navštívit a provést libovolné změny.

Jak jsem již zmínil v předchozí kapitole 2.1.2, stránka má opět vertikální uspořádání, které dle mého názoru není ideální z hlediska úspory prostoru. Zároveň jsem si zde uvědomil, jak podstatné bude uspořádat prvky tak, aby na uživatele působili dobře uspořádaným a nerušivým dojmem.

Obrázek 2.2 Původní stav hlavní stránky webového klienta nástroje Timer2Ticket – dolní část



Obrázek 2.3 Původní stav hlavní stránky webového klienta nástroje Timer2Ticket – rozbalené logy



2.2 Nástroje blízké uživateli Timer2Ticketu

Jelikož Timer2Ticket do jisté míry nahradí manuální práci, kterou jeho uživatel dříve musel vykonávat v rámci přenosu informací mezi nástroji pro měření času a projektovými nástroji, je žádoucí seznámit se také s nimi. Tato analýza pomůže poznat prostředí, na které je uživatel zvyklý. Zaměřím se zde zejména na ty, které byly zmíněny jako potenciální kandidáti do budoucího rozšíření této synchronizační služby v rámci diplomové práce Ing. Víta Štefana [2]. Je pravděpodobné, že uživatel bude mít s některým z nich zkušenost a bude proto výhodné, pokud uživatelské rozhraní nebude příliš vybočovat z řady.

2.2.1 Projektové nástroje

Projektový nástroj je systém, jehož základním úkolem je správa projektů. Řízení rozsáhlejších projektů není snadná záležitost, a proto i zkušený projektový manažer tyto nástroje uvítá. Jejich možností využívá řada firem, které díky nim mají dobrý přehled o stavu svých projektů. Nástroje zpravidla umožňují u jednotlivých projektů evidovat jednotlivé úkoly (požadaky), přiřazovat tyto úkoly konkrétním osobám a sledovat stav jejich vyhotovení. Osoby, které se na projektu podílejí, mají zpravidla možnost k němu i jeho jednotlivým úkolům vykazovat čas, který může firma využít jako metriku pro vyhodnocení finanční stránky projektu.

Mimo výše zmíněné možnosti tyto systémy typicky poskytují také prostor pro komunikaci v rámci vyhotovování úkolů. Kromě samotné správy projektů mohou sloužit také k reportování chyb či zaznamenávání nejrůznějších změnových řízení. Pro usnadnění práce pak poskytují i podpůrné prvky jako je Ganttův diagram, který uživateli pomůže grafickým způsobem pochopit stav daného projektu. [4]

Redmine

Redmine je svobodný open-source[1] software, který může kdokoliv (například jakákoliv firma) nainstalovat na svůj server a využívat ho dle potřeby. Není zde tedy potřeba řešit jakoukoliv formu placení za poskytnutou službu. Jedná se o systém, který je založený na rolích. Příkladem rolí může být vývojář, manažer či reportér. Na základě rolí může mít uživatel v projektu omezená práva.

Používá se jak pro řízení projektů, tak jakožto systém pro reportování chyb. Jeho uživatelé mohou využívat podpůrných prvků jako je výše zmíněný Ganttův diagram, kalendář či zasílání emailových oznámení (například, pokud je uživateli přiřazen úkol). Velmi užitečným prvkem je také wiki, na kterou lze umístit řadu podstatných informací k danému projektu. Uživatelům je k dispozici i speciální prostor pro dokumentaci či sdílení souborů. Samozřejmě poskytuje možnost sledovat stav projektu, úkolu i vykazování stráveného času na nich s možností popisu a specifikací aktivity (například „Administrativa“), ke které se čas vztahuje.

Jeho uživatelské rozhraní se vyznačuje svou jednoduchostí. Členění stránky se odvíjí od dvou horizontálních meníček, která uživatele provedou po jemu přiřazených úkolech a projektech, jejichž je součástí. Speciální prostor je věnován i uživatelskému účtu. [6, 2]

Jira

Jira [7] je na rozdíl od nástroje Redmine software, který je vyvíjen společností Atlassian a jedná se o komerční projekt. Stejně jako Redmine řeší základní problémy řízení projektů. Umožňuje práci na více projektech, přiřazování požadavků v rámci jednotlivých projektů uživatelům, přiřazování zodpovědných osob za dané úkoly, sledování stavu úkolů a projektů, nastavování priorit

[1] Open-source software [5] je takový software, jehož zdrojové kódy jsou plně dostupné, lze je měnit a vylepšovat. Vývoj typicky kvůli lepší koordinaci zastřešuje nějaká organizace, která software neprodává a peníze získává prostřednictvím nabízení služeb s daným softwarem souvisejících.

úkolům, evidování chyb, sdílení dokumentů a informací či nejrůznější statistiky. Důraz je kladen zejména na jednoduchost práce a komunikace mezi uživateli, což podpoří jejich agilní vývoj. Ten je podpořen i velmi moderním uživatelským rozhraním či podporou štítků, které si může uživatel vytvářet a označovat si jimi například své úkoly.

Cílem projektového nástroje Jira je zjednodušení práce jejím uživatelům, čehož dosahuje také tím, že sama provede lehkou customizaci na základě zjištěných dat o svém uživateli. Při nastavovnání uživatelského účtu je totiž uživatel dotázan například na přechozí znalost tohoto nástroje či zkušenosti s agilním vývojem. Zároveň je již při prvotním nastavování obeznámen s další předností, kterou je jednoduchost propojení s dalšími často používanými nástroji jako je komunikační aplikace Slack, nástroj k zaznamenávání chyb Sentry či řada dalších.

Z marketingového hlediska Jira velmi dobře využívá svou hlavní stránku, na níž v jednotlivých částech uvádí klíčové prvky svého softwaru. Vyzdvihuje zde své nástěnky (tzv. *boards*), které usnadní rozklad rozsáhlého projektu na menší části a které poskytují řadu pokročilých funkcí (jako například filtrace požadavků pro jednotlivé členy), či například *roadmapy*, které pomáhají udržet uživatelům v hlavě širší kontext o stavu projektu v grafické podobě. Mimo to stránka obsahuje i řadu konkrétních ukázek z aplikace, které dobře software přiblíží.

Jak jsem již zmínil, jedná se o komerční projekt. Jira nabízí však celkem čtyři plány, z nichž jeden je zcela zdarma. Ze zbylých tří může uživatel dva vyzkoušet zdarma po omezený čas. Ceny placených plánů se pak odvíjí od počtu možných uživatelů a konkrétní volby plánu. [8]

Nutcache

Nutcache [9] je dalším z řady projektových systémů. Kromě důrazu na řízení projektů za účelem zvýšení produktivity svých uživatelů však cílí také na hlídání finančního rozpočtu. Firma využívající tento systém tak může nastavovat upozornění na blížící se vyčerpání rozpočtu, což jí pomůže vyhnout se nebezpečnému přestřelení částky, kterou na realizaci projektu má.

Na rozdíl od většiny svých konkurentů disponuje možností přiřadit více osob na řešení jediného úkolu. Zda je tento prvek výhodný, záleží na jeho použití. Dle mého názoru to může značit nedostatečně rozdělenou práci, což může vyústit například ve stagnaci daného úkolu (za předpokladu, že řešitelé nebudou do práce motivováni). U úkolů lze stejně jako v předchozích nástrojích evidovat řadu prvků – jejich stav, řešitele, popis a další.

Dalším specifikem této aplikace je také logování času. Hlavním rozdílem oproti dříve zmíněným projektovým systémům je fakt, že neexistují samostatné časové záznamy pro jednotlivé měřené časy. Veškery změřený čas se eviduje přímo jako doba strávená při práci na daném úkolu.

Stejně jako Jira i Nutcache velmi dobře reprezentuje svůj projekt na své hlavní stránce. Potenciální uživatel je zde informován o hlavních přednostech systému, zejména pak na udržování projektového rozpočtu a přehlednou organizaci úkolů. Představuje se zde snadné propojení s komunikačními nástroji jako je Slack či Gmail a uživatel zde nalezne i FAQ[2] či hodnocení klientů. Za velmi dobrý marketingový krok považuji také videa s návody pro používání aplikace, o kterých systém informuje ihned po registraci.

Jedná se o komerční projekt stejnojmenné společnosti. V nabídce jsou celkem tři plány, z nichž jeden je zdarma. Plány mají pevně stanovené ceny a liší se v mnoha kritériích, z nichž mezi klíčové patří omezení počtu uživatelů, propracovanost možností měření času, velikost místa na dokumenty, možnost využívat Ganttův diagram či možnosti hlídání finančního plánu. Nad rámec těchto tří plánů je nabídnuta diskuze o speciálním plánu pro společnosti o více než 150 uživatelích, jak uvádí na své webové stránce [10].

ClickUp

Jako poslední projektový nástroj jsem se zde rozhodl úvest ClickUp [11]. Na rozdíl od ostatních je však o něco více komplexní a poskytuje možnost automatizovaných procesů, které si v něm

[2] Frequently asked questions – často kladené otázky.

může uživatel nastavit. Kromě samotného řízení projektů si klade za cíl nahradit funkcionalitu řady jiných systémů a centralizovat tak pracovní prostor na jedno místo – například i veškerou komunikaci, která by se mohla odehrávat v jiném specializovaném nástroji. Umožňuje proto i propojení se systémy jako je Jira a importování tamních dat. Zároveň umožňuje vytvoření customizovaných importů pro zatím nepodporované nástroje.

Jelikož se jedná o projektový nástroj, součástí je opět správa úkolů v rámci jednotlivých projektů. Umožňuje evidenci a odhady času, nastavování priorit, využívání štítků a řadu dalších funkcionalit. Specifikem je fakt, že je jedním ze systémů, kde úkol může být opět přiřazen více osobám. Jedná se také o jeden z nástrojů, pro který je vytvořena automatická synchronizace s nástrojem Toggl Track (pomocí API klíče a Chrome rozšíření). Řešení, které je zde však použito, není zamýšleno obecně pro synchronizaci mezi dalšími nástroji.

Hlavní propagační stránka aplikace je propracována velice kvalitně. Obsahuje řadu animací a ukázek přímo z aplikace. Představeno je zde nepřeberné množství nástrojů, jejichž činnost může právě ClickUp nahradit. Zároveň jsou zde popsány i možné integrace s řadou nástrojů (například s Google kalendářem).

Pro zájemce je připraveno celkem pět plánů. Jeden z nich je zdarma, avšak je doporučován pouze pro osobní využívání. Další plány mají dostupné funkcionality rozplánovány zejména s ohledem na velikost týmu. S ohledem na široký výběr jsou k dispozici i FAQ. [12]

2.2.2 Nástroje pro měření času

Nástroje pro měření času jsou aplikace, které slouží pro snadné zaznamenání a velmi přesné změření času spojeného s prací na určitém projektu či úkolu. Uživateli obvykle pouze stačí zvolit projekt či konkrétní úkol a zapne časomíru. Jakmile práci dokončí, časomíru opět vypne a získá časový záznam o vykonané práci [2]. Za obzvláště užitečné toto ovládání považují zejména uživatelé jako jsou projektoví manažeři, kteří často pracují souběžně na několika projektech a často střídají kontext své práce.

Právě k přesnému měření času je proto využívají řady firem, které se o vzniklé časové záznamy mohou opírat jako o metriku, která je základem pro ziskovost jejich projektů. Mohou je však využít i jednotlivci pro vlastní potřebu, protože časové záznamy mohou posloužit i jako metrika, která nám usnadní organizaci a plánování našeho času. Vzhledem k tomu, že se obvykle jedná o čas strávený na nějakém projektu, zpravidla bývají využívány právě ve spolupráci s projektovými nástroji, které pak poskytnou potřebné detailnější prostředky pro řízení projektů.

Toggl Track

Toggl Track je jednou z nejrozšířenějších aplikací pro měření času s více než pěti miliony uživately. Jak sami autoři uvádějí [13], aplikace byla vytvářena s důrazem na jednoduchost, která zajistí intuitivní používání bez nutnosti studování složitých manuálů. Spolu s elegantním designem pak Toggl Track pomáhá eliminovat stres z používání dodatečné aplikace, usnadňuje práci a zvyšuje produktivitu.

Pro měření času v této aplikaci lze přesně následovat obecný postup uvedený v kapitole 2.2.2. Zároveň je možné k časovému záznamu připojit popis a případné štítky, které jsou postačující podmínkou pro integraci do Timer2Ticketu, neboť na ně probíhá mapování z objektů v projektových systémech. [2]

Kromě samotného měření času, které je k dispozici na hlavní obrazovce, jsou uživateli k dispozici reporty o naměřeném času, které mu mohou být zasílány v pravidelných intervalech i na registrovaný email. V jejich rámci je možné provádět filtraci například skrze různé projekty či štítky. Zároveň jsou reporty zobrazovány respektujíce zvolený pracovní prostor (tzv. *workspace*), což je v Toggl Tracku základní prostředí, pod kterým se shromažďují jednotlivé projekty. Uživatel pak může mít více takových prostředí. [13]

Co se týče plánů, Toggl Track nabízí celkem čtyři, z nichž jeden je zdarma. U všech dává však příležitost, zkusit si je zdarma po dobu jednoho měsíce. Platby následně umožňuje měsíční či roční. Plány jsou primárně designované s ohledem na velikost týmu, který je bude používat, a v případě velkých společností nabízí možnost diskuze individuální ceny plánu. [14]

Clockify

Clockify [15] je další z řady aplikací, jež je používána miliony lidí pro meření času na jejich projektech. Svým uživatelským rozhraním se až nápadně podobá výše zmíněnému Toggl Tracku. Na rozdíl od něj ale pracuje více s horizontálním prostorem a kromě vertikálního meníčka využívá i horizontální, na které přesouvá infromace o uživatelském profilu, jež jsou v Toggl Tracku součástí toho vertikálního. V základu obsahuje například i funkcionality jako je filtrace časových záznamů či jejich extrahování. Nicméně i tyto prvky jsou součástí výše zmíněného Toggl Tracku.

Ač si klade za cíl zvýšit produktivitu svých uživatelů, v neplaceném režimu je řada funkcionalit nedostupná. Až teprve součástí placených plánů je umožněno například vytváření faktur, customizace exportů či možnost automatického zaokrouhlování času. Placené plány jsou celkem čtyři, cenově odstupňované dle přídatných prvků, které podporují možnosti týmové spolupráce v rámci jednotlivých *workspaců*. [15, 16]

Kromě zmíněných plánů a detailního rozboru funkcionalit jsou na propagačním webu k nahlédnutí řady ukázek aplikace. Cílí se zde na multiplatformnost, a proto jsou k dispozici náhledy jak z webových aplikací tak i z mobilních. Neboť má již aplikace široké množství zákazníků, využívá jejich recenzí jako součást propagace. [15]

Harvest

Poslední nástroj pro měření času, který zde uvedu je Harvest [17]. Umožňuje nejen měření času, ale zároveň přímé intuitivní vkládání času pro případy, že nebyl změřen tímto nástrojem. Čas je měřen po projektech a jednotlivých úkolech. V souvislosti s tím je možné vytvářet také faktury či sledovat pracovitost jednotlivých členů týmu. Lze například nastavovat časové kapacity, které uživatelé nesmějí jejich prací překročit s ohledem na rozpočet. Naplnění těchto limitů si lze i graficky zobrazit. Součástí sledování času však není práce se štítky jako v přechozích dvou nástrojích, což značně komplikuje potenciální možnost, jak by se dal Harvest v budoucnu integrovat do Timer2Ticketu, jak sám Vít Štefan uvedl ve své práci [2].

Oproti výše zmíněným nástrojům i plán zdarma obsahuje zdlouhavější prvotní nastavování, ve které si uživatel může nastavit pokročilé prvky jako jsou například upomínky, které mají zajistit, že uživatel si nezapomene čas měřit. Na jednu stranu to může být velice užitečná záležitost, ale pokud si chce uživatel nástroj jen vyzkoušet tak, jako jsem to učinil já, nastavování podrobnějších prvků ho může dle mého názoru odradit.

Harvest zaujal poměrně odlišnou strategii v rozdělení svých plánů. Nabízí pouze dva [18] v závislosti na počtu projektů, které může uživatel používat. V případě plánu zdarma se jedná o dva projekty, v placeném jich je neomezené množství. Z marketingového hlediska si myslím, že může být škoda, že zde není detailněji zacíleno na jednotlivé potenciální zákazníky.

2.3 Týmový vývoj a teamleading

2.3.1 Zahájení spolupráce

Neboť bylo očividné, že nový návrh frontendu pro nástroj Timer2Ticket bude poměrně rozsáhlý, byl jsem rád, když mi ze strany vedoucího byla navržena možnost spolupráce se studenty z předmětu SP2, který je založen na týmové spolupráci. Rozhodl jsem se této možnosti využít a oslovit je. Z toho důvodu v této podkapitole vysvětlím zvolené postupy týmového vývoje.

Na začátku října jsem se poprvé setkal se třemi zájemci o účast na týmovém projektu. Představil jsem jim nástroj Timer2Ticket a základní představu toho, na jaké práci by se mohli podílet. Po krátké diskuzi se ukázalo, že dva z nich by měli zájem. S Irinou Kara-Sel a Benedekem Molnárem proto došlo k výměně kontaktů a požádal jsem je, aby si také prostudovali současný stav tohoto nástroje.

2.3.2 Forma komunikace

Jelikož nezbytnou součástí týmového vývoje je pravidelná komunikace, rozhodl jsem se pro dva základní komunikační kanály.

Prvním zvoleným prostředkem komunikace je Slack [19]. Jedná se komunikační platformu, která umožňuje vytvářet pro jednotlivé projekty tzv. *workspaces* neboli pracovní prostory, které se dále dělí na kanály, ve kterých může probíhat hromadná komunikace i sdílení souborů. Ta může však probíhat i přímo mezi uživateli. Tento kanál jsem zvolil zejména pro potřeby komunikace se zákazníkem, vedoucím své práce Ing. Jiřím Hunkou, který s ním má bohaté pracovní zkušenosti a preferuje ho. Zároveň byl do tohoto kanálu přizván i Ing. Vít Štefan pro případné konzultace s budoucími návrhy a řešeními.

Jako druhý komunikační kanál jsem zvolil Discord [20]. Jedná se o platformu, která umožňuje jak textovou komunikaci a sdílení souborů, tak i skupinové hovory s možností sdílení obrazu, a to až do počtu sta lidí. Zároveň je mezi spolužáky na Fakultě informačních technologií ČVUT hojně rozšířená, a proto jsem věděl, že bude i pro zbytek týmu komfortní pro používání. Tuto druhou plarformu jsem zvolil zejména pro účely interní komunikace s týmem. Jako dodatečný kanál, který měl sloužit zejména pro potřeby velmi urgentní komunikace, jsme se v rámci týmu rozhodli využít Facebook Messenger.

Kromě komunikačních kanálů bylo potřeba stanovit i pravidelné schůzky, protože pouhá textová komunikace je značně pomalá. Od začátku se proto stanovila pravidelná schůzka se zákazníkem, Ing. Jiřím Hunkou, která probíhala jednou týdně v úterý. Až na pár výjimek, které byly z důvodu koronavirových opatření nezbytné, probíhaly tyto schůzky kontaktním způsobem. Od zákazníka jsem zde získával přesnější požadavky, prezentoval odvedenou týmovou práci a konzultoval možnosti, které jsme jednotlivě v rámci týdenní iterace vytvořili. Po provedení analýzy současného stavu se jednalo zejména o sběr požadavků a následný návrh wireframů, jimž se budu věnovat později v kapitole 4.

S ohledem na nutnost řízení týmu jsem se zároveň rozhodl ještě pro interní týmové schůzky, které obvykle probíhaly den před schůzkou se zákazníkem, a na kterých mi zbytek týmu stručně představil pokrok, který v týdenní iteraci provedl. Zdiskutovali jsme zde vzniklé možnosti a já vždy připravil body k diskuzi se zákazníkem. Pro úsporu času jsme se v tomto případě rozhodli pro online schůzky, pro jejichž uskutečnění jsme využívali výše zmíněný Discord.

2.3.3 Model vývoje a forma spolupráce

Na počátku vývoje jsem zvažoval dvě možnosti, jak k vývoji přistoupit. První z nich bylo rozdělení projektu do menších celků, které by se vždy zanalyzovaly, navrhly a implementovaly. Druhým pak byla důkladná analýza systému, na základě které se zjistí požadavky a následně provede kompletní návrh nového frontendu, jehož prototyp v této práci budu implementovat a následně uživatelsky testovat. Ve výsledku jsem se rozhodl pro druhou možnost, protože jsem považoval za podstatné získat komplexní pohled na systém a ten by iterace po částech tak dobře neumožnila. Zvolený model životního cyklu by proto nejvíce odpovídal vodopádu s důkladnými konzultacemi, které měly za úkol zabránit právě hlavní nevýhodě tohoto modelu, kterou je selhání vývoje na základě špatné či nedostatečné analýzy a špatně určených požadavků, které by se následně při striktním dodržení tohoto modelu jen těžko daly měnit.

Spolupráci nám usnadnila i trojice nástrojů, Redmine, Toggl Track a samotný Timer2Ticket, které jsem již popsal dříve v kapitolách 2.2 a 2.1. V nástroji Redmine docházelo k pravidelnému

vytváření jednotlivých úkolů a obsahoval výkazy času podstatné pro mé spolupracovníky v rámci předmětu SP2. Zároveň posloužil jako centrální uložiště pro vytvářené materiály v rámci vývoje. Tím, že jsme všichni tyto nástroje používali, mohli jsme systém snáze analyzovat.

2.3.4 Zásady teamleadingu

Neboť souběžně s tím, co jsem začal pracovat na své bakalářské práci, jsem absolvoval předmět SI2[3] a zjistil jsem, že jeho součástí bude přednáška na teamleading, využil jsem toho a nastudoval si jeho základy. V této části proto zmapuji nejužitečnější zásady, které jsem z této přednášky načerpal a které by měl týmový vedoucí mít na paměti. Věřím, že mohou být užitečné i pro řadu dalších studentů, kteří budou vést týmové projekty či spolupracovat s jinými studenty v rámci své bakalářské či diplomové práce.

Jedná se zejména o následující čtyři body vytvořené dle [21]:

1. Je esenciální, aby týmový vedoucí zajistil, že jeho tým bude vždy vědět, co má dělat, měl jasné zadání, znal kontext své práce a věděl, proč ji dělá.
2. Nezbytné je, aby týmový vedoucí kontroloval odvedenou práci a její splnění dle zadání.
3. Potřebné je, aby členové týmu cítili zodpovědnost za svou práci.
4. Je vhodné znát své lidi, protože určení jejich slabých a silných stránek pomůže ve správném přidělování práce.

Na výše zmíněné body jsem se při řízení týmu snažil myslet a věřím, že to týmové práci pomohlo. Zpravidla se nám nestávalo, že by někdo odcházel ze schůzky s nejasným zadáním či nedostal zpětnou vazbu k odvedené práci. Díky uvědomění si čtvrtého bodu se mi také podařilo objevit zkušenosti Bena s marketingem a jeho smysl pro design. Jeho znalostí jsem v této oblasti později zužitkoval při tvorbě propagačního webu, kterému se více věnuji v kapitole 4.3.

2.4 Sběr uživatelských a zákaznických požadavků

V této kapitole provedu rozbor požadavků, které vznikly na základě analýzy původního systému a průběžné komunikace se zákazníkem. Samotnému detailnímu návrhu, který je postaven na těchto požadavcích bude věnována kapitola 4 o návrhu.

Kromě změn v rámci samotné aplikace Timer2Ticket jsem v rámci analýzy dospěl k nutnosti existence propagačního webu, který by dokázal upoutat pozornost potenciálního uživatele a informovat ho o základních možnostech a výhodách spojených se samotnou aplikací. Součástí této práce bude pouze samotný návrh propagačního webu, protože jeho řešení původně nemělo být součástí práce a rozsahem již na něj nezbyla kapacita. Samotné implementaci a současnému stavu propagačního webu se bude stručně věnovat až kapitola 7 vztahující se k budoucím rozvoji.

2.4.1 Požadavky na propagační web

Cílem propagačního webu bude zaujmout uživatele a sdělit mu základní informace. Při sběru požadavků jsem využil standardního rozdělení na požadavky funkční a nefunkční, jak je tomu vyučováno v předmětu SI1[4]. Funkční požadavky jsou takové, které se vztahují k funcionalitám a chování tvořeného systému. Nefunkční požadavky jsou takové požadavky na systém, které se netýkají jeho funkcionalit, nýbrž obecných vlastností. Dle toho je pak rozdělujeme do kategorií použitelnost, spolehlivost, výkon a podporovatelnost (respektive rozšiřitelnost). [22] Výsledkem sběru požadavků se ukazují být následující.

[3] Softwarové inženýrství 2 – povinný oborový předmět na FIT ČVUT v oboru Softwarové inženýrství.
[4] Softwarové inženýrství 1 – povinný oborový předmět na FIT ČVUT v oboru Softwarové inženýrství.

Poskytnutí informací o projektu - FP1[5]

Popis: Přinese zákazníkovi informace o tom, co projekt umí, jak mu uspoří čas a jaké mu přinese výhody. Vítaným prvkem bude přítomnost instruktážního videa a reálných ukázek z aplikace.

Typ: Funkční požadavek

Priorita: Vysoká

Náročnost: Nízká

Poskytnutí informací o jednotlivých tarifech - FP2

Popis: Nabídne možné varianty plánů[6], které bude zákazník moci využít. Plánem se zde myslí konkrétní forma členství – buď zdarma či placená.[7]

Typ: Funkční požadavek

Priorita: Vysoká

Náročnost: Nízká

Představí projekt jako open-source - FP3

Popis: Projekt představí jako open-source, jež si může kdokoliv sám zprovoznit i bezplatně bez jakýchkoliv tarifů.

Typ: Funkční požadavek

Priorita: Vysoká

Náročnost: Nízká

Spojení s tvůrci - FP4

Popis: Poskytne informace pro kontakt s tvůrci společnosti Jagu s. r. o. a umožní kontaktovat společnost Jagu s. r. o. provozující projekt ve variantě s jednotlivými plány.

Typ: Funkční požadavek

Priorita: Střední

Náročnost: Vysoká

Možnost donace - FP5

Popis: Poskytne možnost finančně podpořit společnost Jagu s. r. o., která projekt zaštiťuje.

Typ: Funkční požadavek

Priorita: Střední

Náročnost: Střední

Přizpůsobení jazyka - FP6

Popis: Bude využívat nastavení prohlížeče uživatele, aby se správně provedla volba jazyka webové stránky. Pokud stránka nebude daný jazyk podporovat, bude jako výchozí využita angličtina.

Typ: Funkční požadavek

Priorita: Střední

Náročnost: Střední

[5] FPx značí funkční požadavek číslo X.
[6] Ekvivalentním názvem také tarifů.
[7] Bude rozebráno v kapitole 4.1.

Přesměrování do samotné aplikace - FP7

Popis: Uživatel bude mít možnost přesunout se z propagačního webu do části přihlašování či registrace uživatele. V případě tohoto přechodu z propagačního webu bude pro uživatele případně přednastaven zvolený plán ke koupi.

Typ: Funkční požadavek

Priorita: Vysoká

Náročnost: Vysoká

Responzivní design - NP8[8]

Popis: Webová stránka bude uzpůsobena pro různé druhy zařízení (pro všechny velikosti obrazovek).

Typ: Nefunkční požadavek - podporovatelnost

Priorita: Vysoká

Náročnost: Střední

2.4.2 Požadavky na samotnou aplikaci

Základním cílem v rámci samotné aplikace je vytvoření pro uživatele co nejsnadněji použitelného uživatelského rozhraní, které tak umožní bez stresu zvýšit jejich produktivitu, snížit stres a uspořit čas. Sběr požadavků se musí zaměřit na budoucí rozdělení uživatelů dle jednotlivých členství a na fakt, že následný návrh musí počítat s přidáním dalších nástrojů a také možností mít více synchronizovaných dvojic nástrojů. Výsledkem sběru požadavků pro potřeby aplikace se ukazují být následující.

Správa uživatelů - FP1

Popis: Bude umožněna registrace nových uživatelů pomocí emailu a hesla. Registrace pomocí sociální služby[9] není nutností, avšak bude zákazníkem zaštitujícím projekt vítána. Registrovaní zákazníci (tj. uživatelé) se budou moci přihlásit a odhlásit. Uživatelé budou moci zažádat o resetování hesla (například v případě, že ho zapomenou) a heslo následně resetovat.

Typ: Funkční požadavek

Priorita: Vysoká

Náročnost: Vysoká

Správa nastavení profilu uživatele - FP2

Popis: Uživateli bude umožněno měnit své heslo a zobrazit emailovou adresu, pod kterou je registrován. Tu však nepůjde měnit bez obrácení se na provozovatele. Pro lepší informovanost si uživatel bude moci nastavit notifikace, které mu mají chodit. Nastavení notifikací půjde kdykoliv zobrazit a měnit. Uživatel si bude moci nastavit své časové pásmo.[10] Díky tomu bude uživatelova nastavená synchronizace probíhat ve správný čas. Tento prvek je důležitý zejména proto, že se nebudeme omezovat na český trh.[11] Zvolené časové pásmo půjde kdykoliv zobrazit a změnit. Uživatel bude moci aplikovat kupón a na základě toho obdržet

[8] NPx značí nefunkční požadavek číslo X.
[9] Sociální službou zde myslíme například Google či Facebook.
[10] S tímto požadavkem přišli moji spolupracovníci z týmu, za což bych jim chtěl poděkovat.
[11] Zdůvodnění tohoto faktu je detailněji rozebráno v kapitole 2.5.

připsané okamžité synchronizace.[12] Kupóny budou mechanismus, který bude sloužit k odměňování zákazníků, kteří na propagačním webu využili možnost donace. Za darovanou částku obdrží jako poděkování právě kupón na okamžité synchronizace.[13]

Typ: Funkční požadavek

Priorita: Střední

Náročnost: Střední

Správa členství uživatelů a placených služeb - FP3

Popis: Uživatel bude moci zobrazit informace o svém členství.[14] Bude moci zobrazit informace o možných změnách svého členství a změnit ho. Zároveň bude mít možnost zakoupit okamžité synchronizace. Zakoupené členství bude uživateli umožněno zrušit (tj. ukončit předplácení). Toto členství bude následně možné načíst jako předvolbu pro nákup nového členství.[15]

Typ: Funkční požadavek

Priorita: Vysoká

Náročnost: Vysoká

Správa platebních informací uživatele - FP4

Popis: Uživateli bude umožněno přednastavit své platební údaje. Tyto údaje bude uživatel moci kdykoliv zobrazit a měnit. Uživatel bude také moci zobrazit přehled zaplacených objednávek. K zaplaceným objednávkám bude dostupná faktura ke stažení.

Typ: Funkční požadavek

Priorita: Střední

Náročnost: Střední

Správa zaznamenaných logů - FP5

Popis: Záznamy o provedených synchronizacích bude zobrazovat tabulka, která bude umožňovat filtraci, řazení a stránkování záznamů. Bude dostupný prostor pro podrobnější informaci o statusu synchronizace, z něhož půjde informace vykopírovat.[16]

Typ: Funkční požadavek

Priorita: Střední

Náročnost: Střední

Správa spojení - FP6

Popis: Bude možnost, aby uživatel měl více aktivních spojení. Aktivní spojení pro účely této práce znamená konkrétní dvojici nástrojů, pro které je synchronizace nakonfigurována a probíhá mezi ní synchronizace.[17] Uživatel bude moci aktivní spojení deaktivovat a tato deaktivní opět aktivovat (pokud mu to bude jeho členství umožňovat). Návrh bude přizpůsoben faktu, že do aplikace budou časem přidány další nástroje, mezi kterými bude možné

[12] Stav, ve kterém se Timer2Ticket nacházel v době mé analýzy umožňoval kdykoliv a komukoliv provést okamžitou synchronizaci. V novém návrhu budou okamžité synchronizace prvkem, který nebude plně dostupný pro každého – bude jich pouze omezený počet. Více v kapitole 4.1.

[13] Jejich počet se bude odvíjet od darované částky.

[14] Jednotlivé členství neboli plány budou rozebrány v kapitole 4.1.

[15] Tento požadavek má za cíl zjednodušit práci uživatelům, kteří si zruší členství a následně se rozhodnou, že jim původní členství chybí.

[16] Vykopírování bylo jedním z požadavků ze strany zákazníka.

[17] Detailnější rozbor jednotlivých stavů spojení je k dispozici v kapitole 4.2.1.2.

synchronizovat. Uživatel bude moci nastavovat vlastní rozvrh synchronizace. Rozvrh synchronizace bude možné nastavit odlišně pro časové záznamy a konfigurační objekty. Tyto rozvrhy budou kdykoliv zobrazitelné a změnitelné. Uživatel bude mít k dispozici informaci o poslední provedené synchronizaci i o času provedení následující.

Typ: Funkční požadavek

Priorita: Vysoká

Náročnost: Střední

Notifikace uživatele - FP7

Popis: Za účelem dosažení komfortu uživatele bude maximálně notifikován při užívání aplikace o událostech, které se dějí.

Typ: Funkční požadavek

Priorita: Vysoká

Náročnost: Nízká

Oddělení placených a neplacených prvků - FP8

Popis: I přes to, že společnost Jagu s. r. o. bude aplikaci nabízet i v placené podobě, je potřeba při návrhu klást důraz na snadné oddělení částí aplikace, které se nepoužijí, pokud se někdo rozhodne aplikaci využívat neomezeně pro své vlastní potřeby bez dalšího nabízení za úplatu (nasadí si ji sám).

Typ: Funkční požadavek

Priorita: Vysoká

Náročnost: Střední

Open-source - NP9

Popis: Aplikace bude distribuována jako open-source software.

Typ: Nefunkční požadavek - použitelnost (konkrétněji přístupnost)

Priorita: Střední

Náročnost: Nízká

Responzivní design - NP10

Popis: Samotná aplikace bude uzpůsobena pro různé druhy zařízení (pro všechny velikosti obrazovek).

Typ: Nefunkční požadavek - podporovatelnost

Priorita: Střední

Náročnost: Střední

Webová aplikace - NP11

Popis: Bude se jednat o webovou aplikaci, jejíž primární podpora musí být zajištěna pro prohlížeč Google Chrome.

Typ: Nefunkční požadavek - podporovatelnost

Priorita: Vysoká

Náročnost: Nízká

Monitoring chyb - NP12

Popis: Aplikace musí používat nástroj pro monitoring chyb.

Typ: Nefunkční požadavek - podporovatelnost

Priorita: Střední

Náročnost: Nízká

2.5 Zaměření zákazníka

Kromě určení požadavků na samotnou aplikaci je také podstatné uvědomit si, kdo bude pravděpodobným uživatelem aplikace a tedy naším zákazníkem. Proto zde provedu analýzu uživatelů, kteří používají nástroje blízké Timer2Ticketu – konkrétně pro nástroje Redmine, Jira, ClickUp, Toggl a Harvest. Data pro analýzu v této kapitole mi poskytla platforma Enlyft [23], která obecně pomáhá společnostem identifikovat jejich pravděpodobné zákazníky.

2.5.1 Pravděpodobný původ zákazníka

První podstatnou informaci, kterou je potřeba si uvědomit, je, že pro úspěch aplikace bude opravdu nutné zacílit nejen na český, ale i na zahraniční trh. Tohoto závěru jsem docílil na základě tabulky 2.1 vytvořené na základě [24, 25, 26, 27, 28]. Ukazuje se, že drtivá většina potenciálních zákazníků je ze Spojených států amerických. Bude tedy opravdu vhodné využít jako výchozí jazyk pro tuto aplikaci angličtinu.

Nejvíce naměřených dat máme k dispozici u nástrojů Redmine a Jira, uvedu zde proto i následující příčky. V případě Redmine se s poměrně velkým odstupem na druhém místě umísťuje Francie s 8 % a na třetím Indie se 7 % následovaná Spojeným královstvím. Co se týče nástroje Jira, odstup zbylých zemí je také značný. Na druhém místě se umísťuje Spojené království s 8 % a na třetím místě Kanada s 5 % následovaná Indií. V případě ostatních nástrojů není datový základ tolik široký a tolik vypovídající, proto zde další příčky neuvádím.

Tabulka 2.1 Pravděpodobný původ zákazníka

	Redmine	Jira	ClickUp	Toggl	Harvest
Nejpravděpodobnější místo původu	USA	USA	USA	USA	USA
Procento zákazníků z tohoto místa	36 %	51 %	91 %	47 %	79 %
# společností pro data	6 956	89 847	17	13	47

2.5.2 Pravděpodobné pracovní odvětví zákazníka

Další podstatnou metrikou, kterou mi data poskytla, je informace o tom, odkud bude pravděpodobný zákazník pocházet, tedy na koho je potřeba cílit. Ukazuje se, že naprosto dominantní procentuální zastoupení potenciálních uživatelů bude pocházet z odvětví vývoje počítačového softwaru. Můžeme tedy počítat převážně s uživatelem technického charakteru.

Detailní rozbor cílových oborů je v tabulce 2.2 vytvořené na základě [24, 25, 26, 27, 28]. Uvádím zde první dvě příčky, neboť další již mají větší odstup.

Tabulka 2.2 Pravděpodobné pracovní odvětví zákazníka

	Redmine	Jira	ClickUp	Toggl	Harvest
1. místo	PS[18] (26 %)	PS (26 %)	PS (24 %)	PS (24 %)	PS (30 %)
2. místo	ITS[19] (18 %)	ITS (14 %)	V[20] (6 %)	ITS (16 %)	MR[21] (11 %)
# společností pro data	6 956	89 847	17	13	47

[18] Jedná se o zkratku pro odvětví vývoje počítačového softwaru.
[19] Jedná se o zkratku pro odvětví informačních technologií a služeb.
[20] Jedná se o zkratku pro odvětví vzdělávání.
[21] Jedná se o zkratku pro odvětví marketingu a reklamy.

2.5.3 Pravděpodobný rozměr společnosti jako zákazníka

Poslední významnou metrikou, kterou mi data poskytla, je rozměr společností, které by mohly mít teoreticky zájem o naši aplikaci. Z průzkumů vyplývá, že nástroje jsou nejpoužívanější mezi společnostmi, které mají mezi 10 až 50 zaměstnanci, tedy takového zákazníka bychom směrem od nich mohli očekávat.

S ohledem na výše zmíněný fakt si myslím, že bude podstatné pro takovéto či větší společnosti, aby kromě klasických tarifů, jejichž návrhu bude věnována kapitola 4.1, existovala také možnost kontaktovat provozovatele, kterým bude společnost Jagu s. r. o., a požádat ho o vytvoření speciální zlevněné hromadné nabídky. Tato možnost by měla být určitě jednou z informací uvedených na propagačním webu, jehož návrhu se bude podrobněji věnovat kapitola 4.3.

Detailní rozdělení rozměru společností, jakožto potenciálních zákazníků směřujících od současných uživatelů rozebíraných nástrojů, je k nalezení v tabulce 2.3 vytvořené na základě [24, 25, 26, 27, 28]. Opět uvádím pouze první dvě příčky, neboť počet společností na dalších významně klesá.

Tabulka 2.3 Pravděpodobný rozměr společnosti jako zákazníka

	Redmine	Jira	ClickUp	Toggl	Harvest
1. místo (na počet zaměstnanců)	10–50	10–50	10–50	50–200	10–50
2. místo (na počet zaměstnanců)	50–200	50–200	1–10, 50–200	10–50	50-200
# společností pro data	6 956	89 847	17	13	47

Kapitola 3
Volba technologií

Neodmyslitelnou částí vývoje software je také volba vhodných technologií, čemuž věnuji tuto kapitolu. Nejprve představím nástroje pro tvorbu wireframů a vysvětlím svoji volbu. V další části rozeberu svou volbu frontendového frameworku a programovacího jazyka. Následně pro účely propagačního webu zdiskutuji vhodné nástroje pro jeho tvorbu a vysvětlím svoji volbu. Na závěr provedu výběr vhodné platební brány pro samotnou aplikaci a opět zdůvodním svoji volbu.

3.1 Výběr nástroje pro tvorbu wireframů

Po získání základních požadavků na systém bylo nutné navrhnout, jakým způsobem budou tyto požadavky naplněny v rámci uživatelského rozhraní. Nejprve jsem začal sepisováním obyčejných textových postupů a scénářů. Ač je tato fáze důležitá pro stanovení základních pochodů, ukázalo se, že pro účely komunikace se zákazníkem i týmem není zrovna názorná. S ohledem na to jsem se rozhodl v další fázi použít právě nástroje pro tvorbu wireframů. Tento postup jsem stanovil na základě znalostí z přednášky [29] z předmětu o návrhu uživatelského rozhraní z magisterského studia na FIT ČVUT.

Wireframe je jakýsi „*zjednodušený pohled na umístění funkčních prvků ve finálním layotu*“ [30]. Tento zjednodušený model může mít různou úroveň detailu. Lze ho vytvářet pouze černobíle bez pokročilejších designových prvků, ale zároveň lze vytvořit vizuálně velmi estetické wireframy. Hlavní výhodou je ale především poměrně rychlá a snadná modifikace navržených stránek, což pomůže při častých změnách v době návrhu. V případě placeného projektu to pak může pomoci snížit i cenové náklady oproti stavu, kdy by se nástroj toho typu nevyužil. [29]

3.1.1 Balsamiq

Balsamiq je prvním z nástrojů na tvorbu wireframů, který jsem se pro účely této práce rozhodl zvážit, protože ho doporučuje samotný dříve zmíněný předmět NUR[1]. Jedná se o nástroj, který slouží zejména k vytváření tzv. Lo-Fi[2] prototypů. Lo-Fi prototyp je takový prototyp, který je oproti Hi-Fi prototypu[3] méně přesný a více abstraktní vzhledem k budoucí realitě, jak lze nahlédnout například na ukázce 4.12. Tato nízká úroveň přesnosti a využívání dummy obsahu[4]

[1] Návrh uživatelského rozhraní – předmět vyučovaný katedrou softwarového inženýrství v rámci magisterského studia na FIT ČVUT.

[2] Lo-Fi je zkratkou pro low fidelity, tedy nízkou úroveň přesnosti.

[3] Hi-Fi je zkratkou pro high fidelity, tedy vysokou úroveň přesnosti.

[4] Jedná se o obsah, který je pouze jakousi výplní. Například místo obrázku, který ještě není grafiky vytvořen se při prototypování wireframu využije obdélník, čímž se vymezí prostor pro budoucí obrázek.

umožňuje provádět rychlé změny. Wireframy lze vytvářet pro mobilní, webové i desktopové aplikace. [29, 31]

Mezi přednosti tohoto nástroje patří velmi snadné a intuitivní ovládání. Práce se provádí v prostoru (tzv. *space*), který může obsahovat více projektů. Balsamiq nabízí možnost vyzkoušet si tento nástroj zdarma po dobu třiceti dnů. V této zkušební verzi vše funguje plně v prohlížečové variantě – není omezen počet projektů ani spolupracujících uživatelů. Následně je pouze omezena možnost editovat své projekty, nicméně stále jsou dostupné k náhledu. Uživatel si může kdykoliv stáhnou svůj projekt ve formátu BMPR, který lze následně do Balsamiqu opět importovat. Teoreticky této možnosti může člověk zneužít a pokračovat na práci na svém projektu dále pod jiným účtem. Pokud však bude projektů více a bude na nich spolupracovat více lidí, rozhodně se to nevyplatí. Balsamiq přichází s možností používat nástroj v prohlížeči či jako desktopovou aplikaci a placené plány se pak liší dle zvolené varianty s ohledem na počet možných projektů a uživatelů v jednom prostoru. [31]

Kromě exportu ve formátu BMPR lze vytvořené wireframy získat také v PDF verzi. Balsamiq umožňuje i plnohodnotnou komunikaci mezi spolupracovníky na wireframech, vytváření komentářů k jednotlivým částem a zasílání notifikací na email, pod kterým se uživatel registruje, pokud někdo zareaguje na jeho komentář či ho někde zmíní. Mezi další přednosti může patřit například možnost integrace s Jirou či Google Drivem. [31]

3.1.2 Figma

Figma je druhým z nástrojů, který jsem zvažoval. V tomto případě se jedná o nástroj, který již umožňuje i vytváření prototypů typu Hi-Fi, tedy prototypu, který poskytuje poměrně přesný design, který bude finální aplikace či skutečná stránka mít, jak je demonstrováno například na ukázce 4.15 či 4.16. S ohledem na detail však trvá o něco déle wireframe vytvořit. Stejně jako Balsamiq, i Figma nabízí možnost tvořit wireframy pro všechny druhy aplikací. [29, 32]

I v případě Figmy je k dispozici třicetidenní zkušební verze zdarma. Tato verze však umožňuje v rámci daného týmu mít pouze jeden projekt a omezuje i počet souborů, ve kterých můžeme designovat. Vedle toho však nabízí i placené plány s neomezenými možnostmi. [32]

Kromě základního prototypování pro jednotlivce opět umožňuje i týmovou práci. Spolupracovníci si mohou práci prohlížet, a pokud je jim to povoleno, i editovat či komentovat. Figma [32] má také již v základu připraveno několik tutoriálů, které provedou nového uživatele možnostmi, které nástroj nabízí. Tento prvek mi přišel poměrně užitečný, protože oproti Balsamiqu mi uživatelské rozhraní přišlo méně intuitivní a tutoriály se mi při seznamování se s nástrojem hodily.

S ohledem na to, že Figma nabízí možnost vytvářet velice přesné wireframy, je umožněno také exportovat vzniklé HTML a CSS soubory, které lze využít v budoucí implementaci. Kromě toho lze exportovat wireframy pro další použití ve fromátu PDF, PNG, JPG či SVG. [32]

3.1.3 Volba nástroje pro tvorbu wireframů

Na základě vyzkoušení obou nástrojů jsem se pro účely vytvoření wireframů pro samotnou aplikaci rozhodl využít Balsamiq. Zvolil jsem tak zejména kvůli pro mě velice intuitivnímu ovládání, neboť jsem očekával, že wireframy budu během komunikace se zákazníkem několikrát měnit. Nevyplatilo by se zde proto vytvářet wireframe vyšší úrovně. Zároveň jsem usoudil, že časový limit zkušebních třiceti dnů je pro tento projekt dostatečný a v případě potřeby lze projekt snadno obnovit pomocí exportovaného BMPR.

Pro účely propagačního webu jsem se po diskuzích s Benem[5] rozhodl využít nástroj Figma. K tomuto závěru jsem došel, protože v případě propagačního webu jsem neočekával tolik změn

[5]Dříve zmíněný týmový spolupracovník z předmětu SP2, Benedek Molnár, student FIT ČVUT.

(není tak komplexní) a cítil jsem zde možnost využít právě výše zmíněné exportování generovaných HTML a CSS souborů. Názor Bena pro mě byl v tomto ohledu důležitý zejména proto, že měl od začátku zájem zapojit se později zejména při tvorbě propagačního webu.

3.2 Výběr frontendového frameworku

Vzhledem k tomu, že v této bakalářské práci budu realizovat prototypovou verzi nového uživatelského rozhraní nástroje Timer2Ticket, je pro mě klíčové, abych použil vhodný framework, který realizaci usnadní, neboť framework je jakási „*rozsáhlá sada vývojářských nástrojů, která pomáhá a určuje, jak má programátor aplikaci vyvíjet. Bez frameworku by bylo nutné vše programovat na zelené louce. S frameworkem programátor používá, upravuje a rozšiřuje určité připravené (zpravidla sofistikované a otestované) softwarové součásti, což vede k efektivní a kvalitně odvedené práci.*“ [33]

V současné době je frontendových frameworků celá řada. Z toho důvodu jsem se omezil na tři, které na základě průzkumu [34] z roku 2021 vzešly jakožto nejpoužívanější – React, Angular a Vue.js – a to přesně v tomto pořadí. Jejich využití je zároveň velice stabilní již po dobu pěti let, jak ukazuje na základě dat z tohoto průzkumu tabulka 3.1, což ve vývojáři může vyvolat určitý pocit jistoty, že nepoužívá naprosto novou a neozkoušenou technologii. Vzhledem k rozšířenosti jejich používání je také poměrně pochopitelné, že se jedná o frameworky, o kterých je za posledních pět let mezi frontendovými vývojáři největší povědomí. [34]

Tabulka 3.1 Použití frontendových frameworků React, Angular a Vue.js v letech 2017–2021

	React	Angular	Vue.js
Použití[6] v roce 2017	62 %	29 %	22 %
Použití v roce 2018	72 %	58 %	32 %
Použití v roce 2019	80 %	58 %	47 %
Použití v roce 2020	80 %	56 %	49 %
Použití v roce 2021	80 %	54 %	51 %

3.2.1 Angular

Angular je prvním ze tří frameworků, které zvážím. Jeho zakladatelem je společnost Google a poprvé byl vydán v září 2016, přičemž současný nejaktuálnější stabilní release se datuje do října roku 2020. Jedná se o open-source projekt založený na TypeScriptu. [35]

Jde o framework provozovaný pod MIT licencí [36], což uživatelům dává právo používat kód za libovolným účelem. Pod touto licencí může kdokoliv kód také měnit a modifikovat tak, aby vyhovoval jeho potřebám. Dle [37] má sice 80,3k hvězdiček na GitHubu, nicméně spokojenost s ním je v posledních letech výrazně nižší než z počátku, jak je detailně ukázáno v tabulce 3.2 založené na průzkumu [34].

Tabulka 3.2 Uživatelská spokojenost s Angularem v letech 2016–2021

	2016	2017	2018	2019	2020	2021
Spokojenost[7]	68 %	66 %	41 %	38 %	42 %	45 %

Angular může být na jednu stranu skvělou technologií, neboť se jedná o poměrně velký ekosystém, který umožňuje rychlé prototypování aplikací a nabízí i řadu komponent pro snadné vytvoření responzivních designových aplikací (tzv. *Angular Material*). Na druhou stranu se jedná

[6] Vyjádřeno jako procento z celkového počtu respondentů.
[7] Procento respondentů, kteří by znovu Angular použili, kdyby si mohli vybrat.

o framework, který není tak snadné se naučit, protože dokumentaci občas chybí některé detaily a její čtení může být matoucí. Další nevýhodou Angularu se také zdají být limitované možnosti SEO[8]. [35, 39]

3.2.2 React

Druhým z frameworků, který zde rozeberu je React. Jeho vývoj zaštituje další z velkých společností – tentokrát se jedná o Facebook. Poprvé byl React vydán v květnu 2013 a současný nejaktuálnější stabilní release se datuje do března roku 2021. Jedná se o open-source projekt založený na JSX[9]. [35]

Stejně jako Angular, i React je provozován pod MIT licencí [36]. Dle [41] má v současné době 185k hvězdiček na GitHubu a jeho úspěšnost může jeho uživateli potvrdit i fakt, že spokojenost s ním za posledních šest let ani jednou neklesla pod hranici osmdesáti procent, jak je detailně vidět z tabulky 3.3 založené na průzkumu [34].

Tabulka 3.3 Uživatelská spokojenost s Reactem v letech 2016–2021

	2016	2017	2018	2019	2020	2021
Spokojenost[10]	93 %	93 %	91 %	89 %	88 %	84 %

Za velkou výhodu tohoto frameworku můžeme považovat uživatelskou přívětivost pro nové vývojáře, neboť poskytuje řadu návodů, které jim pomáhají se vypořádat s prvotními komplikacemi při učení se tohoto frameworku. I přes to se však dle [39] doporučuje aspoň nějaká uživatelská znalost JavaScriptu, jinak bude učení se JSX náročnější. Mezi další výhody pak můžeme řadit velké množství znovupoužitelných komponent, které zjednodušují vývoj pestrého interaktivního uživatelského rozhraní a umožní následný lepší uživatelský prožitek. Na druhou stranu nevýhodou tohoto frameworku je fakt, že s ohledem na jeho rychlý vývoj jsou části dokumentace, zejména té novější, nekompletní či méně obsáhlé. [35, 39]

3.2.3 Vue.js

Posledním z frameworků, který jsem se rozhodl zvážit je Vue.js. Oproti Reactu a Angularu není Vue.js založeno a vyvíjeno nějakou konkrétní velkou společností. Jeho tvůrcem je Evan You a vývoj se opírá o aktivní komunitu, která má o rozvoj zájem. Poprvé bylo Vue.js vydáno v únoru 2014 a poslední stabilní release se datuje do dubna 2021. Jedná se o open-source projekt, který je založený na JavaScriptu. [35]

Stejně jako Angular a React, i Vue.js je provozováno pod MIT licencí [36]. Jeho uživatelé navíc nyní mají možnost použít buď Vue 2, což je starší verze, či Vue 3, která je nově od února roku 2022 defaultní [42]. Vue 2 již za svoji existenci dle informací z [43] nasbírala 194k hvězdiček. Její mladší sourozenec, Vue 3, která se poprvé objevila z kraje roku 2020 pak 28,5k hvězdiček dle [44]. Jeho úspěšnost může potvrdit obdobně jako v případě Reactu i průzkum [34], který ukazuje, že spokojenost se konstantně drží nad hranicí osmdesáti procent, jak je detailně znázorněno v tabulce 3.4 založené na průzkumu [34]. Ze všech tří zmíněných frameworků je v posledních čtyřech letech právě on tím, který se mají noví uživatelé stále největší chuť naučit i přes to, že je zde poměrně dlouho. Zdá se být tedy nejaktraktivnějším. Detailní srovnání jsem na základě [34] zpracoval do tabulky 3.5.

[8] SEO neboli search engine optimization znamená optimatilizaci pro vyhledávací nástroje. Brát ji v potaz je důležité, aby naše webová stránka či aplikace získala lepší pozici při vyhledávání. [38]

[9] JSX je zkratkou pro JavaScript XML, což je rozšíření JavaScript syntaxe. [40]

[10] Procento respondetů, kteří by znovu React použili, kdyby si mohli vybrat.

[11] Procento respondetů, kteří by znovu Vue.js použili, kdyby si mohli vybrat.

Tabulka 3.4 Uživatelská spokojenost s Vue.js v letech 2016–2021

	2016	2017	2018	2019	2020	2021
Spokojenost[11]	87 %	91 %	91 %	87 %	85 %	80 %

Tabulka 3.5 Srovnání zájmu uživatelů o Angular, React a Vue.js v letech 2016–2021

	2016	2017	2018	2019	2020	2021
Zájem o Angular[12]	50 %	36 %	25 %	23 %	21 %	16 %
Zájem o React[13]	75 %	71 %	68 %	61 %	58 %	48 %
Zájem o Vue.js[14]	51 %	70 %	70 %	64 %	63 %	50 %

Vue.js je označován za poměrně všestranný framework, který poskytne potřebné, ať už chce jeho uživatel vytvořit dynamickou webovou aplikaci či stabilní mobilní aplikaci. Kromě své všestrannosti je jeho velkou výhodou poměrně detailní a jasná dokumentace. Spolu s jednoduchou integrací se pak stává pro nováčky snadno naučitelným. A pokud znají JavaScript, HTML a CSS, budou mít pro jeho využití kvalitní základy.

Na druhou stranu nevýhodou může být o něco menší komunita vývojářů, než je tomu v případě velkých firem jako jsou Google či Facebook. To znamená také menší množství pluginů. Za nevýhodu lze považovat i flexibilitu, která občas vede k nesrovnalostem v kódu. [35, 39]

3.2.4 Volba frontendového frameworku

Na základě analýzy frontendových frameworků jsem se pro účely implementace prototypu rozhodl využít framework Vue.js. Zvolil jsem tak zejména kvůli tomu, že framework je v současné době velice atraktivní, nabízí kvalitní dokumentaci a umožňuje poměrně rychlé naučení se ho. Zároveň jsem se s ním osobně setkal v rámci předmětů SP1 a SP2 na FIT ČVUT, kde se mi s ním dobře pracovalo. Věřím také, že atraktivita tohoto frameworku může v budoucnu přispět i k budoucímu rozvoji tohoto projektu a případnému zapojení některého z dalších běhů předmětů SP1 či SP2, jejichž studenti jednoduchost a dobrou dokumentaci určitě ocení, neboť se v nich často setkávají se svým prvním frameworkem. Moji volbu podpořil také fakt, že na základě diskuze s několika pracovníky společnosti Jagu s. r. o., která projekt zaštiťuje, jsem zjistil, že je jim právě Vue.js v současné době blízké, což mě utvrdilo v tom, že budu mít případnou oporu v řešení problémů.

Na základě své zkušenosti z výše zmíněných předmětů SP1 a SP2 jsem se však rozhodl pro využití starší verze Vue 2. K tomu závěru jsem došel, protože v těchto předmětech jsme zvolili právě novější verzi Vue 3, pro kterou však ještě ani v současné době není plně připravena grafická knihovna Vuetify, která Vue.js poskytuje stovky modulárních responzivních komponent právě pro vytvoření kompletního uživatelské rozhraní.[15] Pro Vue 3 je pouze verze Vuetify 3 Beta, která je zatím určena pouze k testovacím účelům a není ještě plně funkční. Právě tento fakt mě motivoval použít Vue 2 s původní plně funkční verzí zmíněné grafické knihovny. [45, 46]

3.3 Výběr programovacího jazyka

Vzhledem k frameworku vybranému v kapitole 3.2.4 jsem se zde rozhodl zvážit, zda použít pro účely vytvoření prototypové implementace nového frontendu JavaScript či TypeScript, který použil v současném Timer2Ticketu Ing. Vít Štefan [2]. V této podkapitole proto stručně zanalyzuji podstatné rysy těchto jazyků pro účely této práce a zdůvodním svoji volbu zvoleného.

[12] Procento respondetů, kteří by se chtěli Angular naučit.
[13] Procento respondetů, kteří by se chtěli React naučit.
[14] Procento respondetů, kteří by se chtěli Vue.js naučit.
[15] Bližší využití této grafické knihovny popíšu v kapitole 5.

3.3.1 JavaScript

JavaScript je skriptovací programovací jazyk, který je podporován všemi hlavními internetovými prohlížeči. Jedná se o dynamicky[16] slabě[17] typovaný jazyk. Jeho komunita je oproti té, která je kolem TypeScriptu rozsáhlejší, což je možná způsobeno faktem, že naučit se JavaScript je jednodušší a umožňuje rychlý start projektu, zatímco naučit se TypeScript běžně trvá déle. Z hlediska naučení se je tedy JavaScript přirozená volba. Na druhou stranu vzhledem k slabému typování se může snadněji stát, že uživatel některé chyby objeví až v průběhu samotného běhu programu. [48, 49, 50]

3.3.2 TypeScript

TypeScript je na rozdíl od JavaScriptu objektově orientovaný programovací jazyk a přináší proto možnost využívat prvky jako je dědičnost či interfacy. Jedná se o silně[18] typovaný jazyk, který podporuje statické[19] i dynamické typování. TypeScript, stejně jako JavaScript, podporuje všechny javascriptové knihovny. Díky kontrole typů je v jeho případě snadnější detekce errorů (již při kompilaci) a IDE[20] má lepší podporu pro kontrolu syntaxe, což usnadní i debuggování.

TypeScript se považuje za vhodnou volbu pro větší projekty, protože pomáhá zlepšit udržitelnost kódu a čitelnost. Při jeho použití je však vhodné mít zkušený vývojový tým a předcházející znalost JavaScriptu vývojáři pomůže posunout se k TypeScriptu snadněji. [48, 49, 50]

3.3.3 Volba programovacího jazyka

Pro účely této práce jsem se rozhodl zvolit JavaScript. Učiním tak zejména s ohledem na to, že s TypeScriptem jsem se doposud nesetkal a s JavaScriptem mám pouze malou zkušenost z předmětů SP1 a SP2. Teprve nyní, v letním semestru 2022, absolvuji na FIT ČVUT předmět PJS.1 představující JavaScript podrobně. Myslím si proto, že při mých současných znalostech by volba TypeScriptu nebyla tak efektivní, ačkoliv oproti JavaScriptu nabízí řadu pokročilejších možností.

3.4 Výběr nástroje pro tvorbu propagačního webu

Protože v návaznosti na této práci se bude implementovat i samotný propagační web, na čemž již pod mým vedením začne Benedek Molnár, je žádoucí vybrat vhodnou technologii pro jeho implementaci. Pro tento účel jsem se rozhodl zvážit použití některého ze systémů pro správu obsahu (tzv. *CMS*), neboť právě obsah bude na propagačním webu nejdůležitější.

CMS je zkratkou pro Content Management System neboli „*systém pro jednoduchou a rychlou správu obsahu internetových stránek*“. Někdy je můžeme najít také pod názvem redakční nebo publikační systém. Výhodou těchto systémů je fakt, že správu internetových stránek umožňují typicky prostřednictvím webového rozhraní. Není potřeba nic instalovat, což zvyšuje flexibilitu při správě stránek. Zároveň zjednodušují proces tvorby webové stránky tak, že není potřeba mít hlubší znalost programování, a snižují náklady na provozování webových stránek. [51]

[16] Dynamicky typovaný jazyk je takový, kdy typ je nesen hodnotou, díky čemuž můžeme přiřazovat do jedné proměnné hodnoty různých typů. [47]

[17] Slabě typovaný jazyk je takový, ve kterém můžeme použít hodnotu určitého typu někde, kde se očekává jiný typ. Takovýto jazyk se totiž pokusí hodnotu převést na potřebný správný typ a operaci provést. V takovém jazyce tedy budeme moci například násobit číslo s řetězcem, který ve skutečnosti číslu odpovídá. [47]

[18] Silně typovaný jazyk je takový, ve kterém nemůžeme použít hodnotu určitého typu někde, kde se očekává jiný typ. V takovém jazyce tedy nebudeme moci například násobit číslo s řetězcem, který by ve skutečnosti číslu odpovídal. [47]

[19] Staticky typovaný jazyk je takový, kdy typ je nesen nejen hodnotou, ale i refetencí, což je o něco striktnější. Dané referenci pak nemůžeme přiřazovat hodnotu, která jejímu typu neodpovídá. [47]

[20] IDE neboli integrated development environment je zkratkou pro vývojové prostředí.

Pro účely propagačního webu zde rozeberu dva systémy. Prvním z nich bude pravděpodobně nejznámější redakční systém WordPress. Druhým bude o něco mladší systém October CMS, který však v současné době firma Jagu s. r. o., která Timer2Ticket zaštiťuje, používá na některých jejích webových stránkách a zdá se mi proto vhodnou alternativou, kterou je dobré zvážit. [52]

3.4.1 Wordpress

WordPress se ukazuje být nejpopulárnějším redakčním systémem, který podporuje kolem 43 % všech webových stránek na internetu. Jedná se o open-source systém postavený na PHP a MySQL, jehož začátek se datuje již do roku 2003. Byl založený pro zjednodušení publikování obsahu a jeho použití je plně zdarma. [53]

Mezi hlavní přednosti WordPressu můžeme dle [54, 55] řadit:

- Umožňuje bez hlubších technických znalostí vytvořit dobře vypadající webovou stránku. Lze použít předpřipravený motiv, který se následně customizuje.
- Jeho uživatel má právo prostřednictvím vytvořené stránky vydělávat.
- Obsahuje tisíce pluginů, které umožní stránkám přidat prvky jako fotogalerie či kontaktní formuláře. Některé jsou placené, mnoho jich je však zdarma.
- Je připraven pro SEO a nabízí i dodatečné SEO pluginy, které vyhledávání dále zjednoduší.
- Vzhledem k rozšířenosti existuje velká komunita, která může být nápomocná (například při řešení problémů).
- Veškerý použitý obsah si z WordPressu lze stáhnout ve formátu XML a následně použít v jiném redakčním systému v případě nutnosti změny.
- Umožňuje snadno dosáhnout responzivního designu a vysoké výkonnosti.

Díky výhodám zmíněným výše se uživateli značně zjednoduší vytváření webové stránky. Sám se však bude muset postarat o webhosting, což je jakýsi „*pronájem prostoru pro webové stránky*“. Pokud nemáme vlastní, typicky k tomu dochází na cizím serveru. [56] Zároveň se uživatel musí postarat o zajištění domény. Obojí je bohužel již narozdíl od samotného WordPressu placené.

3.4.2 October

October (někdy též October CMS) je také open-source systém. Stejně jako WordPress, i October je postaven na PHP a využívá Laravel framework. Jedná se však o výrazně mladší software, jehož první release se datuje do roku 2014. [57, 58]

Mezi hlavní přednosti Octoberu můžeme dle [58] řadit:

- Je vytvořen tak, aby zjednodušil vývoj kvalitních webových stránek vývojářům, ale umožnil ho i uživatelům bez technických zkušeností. Toho se snaží docílit intuitivním uživatelským rozhraním, díky čemuž je snadné se ho naučit používat poměrně rychle. Není potřebná ani znalost žádného nového frameworku.
- Negeneruje žádný kód a dodržuje WYCIWYG.[21]
- Poskytuje komponenty pro zrychlení a zjednodušení tvorby jednotlivých stránek, přičemž jednotlivé stránky eviduje v rozdílných souborech.
- Pro úpravy kódu umožňuje využít vestavěný editor.

[21] WYCIWYG je zkratkou pro What You Code Is What You Get neboli uživatel získá to, co napíše do kódu.

- Umožňuje používat pluginy a motivy z tzv. *Marketplace*. Ty tam může kdokoliv přidávat, díky čemuž jejich počet neustále roste. Díky tomu lze získat stejně jako v případě WordPressu rozšiřující prvky. Některé lze opět získat zdarma, za některé si uživatel musí zaplatit.
- Ve srovnání s ostatními CMS platformami nabízí nejsnadnější přidávání nových spolupracovníků do společné práce v Octoberu.
- Díky poměrně široké komunitě je dostupný v 36 jazycích.
- Zdarma nabízí plugin pro snadný překlad obsahu stránek do cizích jazyků. Dle [59] je překlad ve srovnání s WordPressem výrazně jednodušší.
- Pomáhá při zajišťování responzivního designu. Toho lze například v případě obrázků dosáhnout pomocí pluginu [60].
- Nezapomíná na SEO a poskytuje dodatečné pluginy pro snazší vyhledávání. [61]

Na základě výše zmíněných možností lze jednoznačně říci, že October nám pomůže s vývojem webové stránky. Na rozdíl od WordPressu však nejde o software zdarma. Nabízí celkem čtyři plány, přičemž nejlevnější umožňuje registraci zdarma a následné nakupování dodatečných licencí po devatenácti dolarech ročně. Nabízí ale také demo verzi, ve které si lze jeho možnosti vyzkoušet.

3.4.3 Volba nástroje pro tvorbu propagačního webu

Po srovnání obou nástrojů jsem se pro účely vytvoření propagačního webu rozhodl využít October. V základu mi oba nástroje nabízí podobné funkcionality, avšak zdá se mi velice zajímavá možnost překladů, kterou October nabízí, neboť pro propagační web se určitě bude hodit. Také předpokládám, že přijde vhod snadné přidávání nových spolupracovníků, pokud by se na projektu v budoucnu chtěl zapojit například nějaký další student.

Jistou nevýhodou bude sice nutnost zakoupení licence, nicméně tento fakt jsem prodiskutoval s vedoucím své práce a Ing. Oldřichem Malcem (oba zaštiťující společnost Jagu s. r. o.) a na volbě nástroje došlo ke shodě. Výhodu ve volbě tohoto nástroje spatřuji také v možnosti případné technické podpory, neboť Jagu s. r. o. již zkušenost s Octoberem má.

3.5 Výběr platební brány

Platební brána je „*řešení, které pomáhá k přenosu platby mezi e-shopem či prodejním webem a platícím zákazníkem. Jedná se o platformu, která umožní zákazníkovi hned několik platebních možností. (…) Platební brána v podstatě zprostředkuje propojení mezi zákazníkem, e-shopem a systémem banky. Zákazník může bez obav zapsat do formuláře/okna platební brány údaje své karty, aniž by došlo k jejímu zneužití. Většina platebních bran funguje na tzv. zabezpečení 3D Secure, které jejich data ochrání před zneužitím.*“ [62]

Jak již ukázala analýza zákazníka v kapitole 2.5.1, je nezbytné zacílit i na zahraniční trh. S ohledem na to provedu analýzu platebních brán, které by se daly využít v rámci tohoto synchronizačního nástroje. Následně na základě této analýzy zvolím nejvhodnější. Součástí této práce bude pouze výběr platební brány, nikoliv její následná integrace do nového frontendu s ohledem na rozsah práce (její přítomnost není podstatná pro otestování návrhu na novém prototypu). Na základě srovnání z [62] jsem se rozhodl zúžit svoji analýzu pro tento projekt na čtyři platební brány, které se zdají být nejvhodnější – ComGate, GoPay, PayU a Stripe. Zdroje [62, 63, 64, 65], z jejichž obsahu budu v této kapitole čerpat, není bez svolení jakkoliv povoleno používat. Proto abych je mohl použít, kontaktoval jsem emailem jejich marketingového manažera a majitele Ing. Jaroslava Janíčka a získal souhlas k jejich využití v rámci této bakalářské práce.

Na základě srovnání platebních bran z [66, 62] se brány, které se zde hodlám porovnávat zdají být rovnocenné v podstatných technických parametrech. Všechny využívají výše zmíněné

zabezpečení 3D Secure, poskytují možnost zapamatovat si platební kartu, přizpůsobit si vzhled, uskutečnit opakované platby a disponují veřejně dostupnou dokumentací API (mimo jiné pro účely integrace). Všechny zmíněné také umožňují platby v ČR i v zahraničí a poskytují desítky platebních metod, zahrnujíce platbu kartou i online bankovní platby. Všechny zmíněné až na Comgate také umožňují využití elektronických peněženek jako jsou Google Pay či Apple Pay.

3.5.1 Původ a zaměření

Jak jsem zjistil v kapitole 2.5.1, bude nutné zacílit nejen na českého, ale i na zahraničního zákazníka. Proto bude podstatné zvolit platební bránu, která bude známá nejen u nás, ale i ve světě a bude díky tomu pro uživatele co nejvěrohodnější.

ComGate je plně česká platební instituce, která je na trhu již od roku 2000 a využívá ji přes 10 000 firem. Na rozdíl od ní, GoPay je sice českého původu, avšak od letošního roku je jejím majoritním vlastníkem společnost Worldline, francouzská nadnárodní společnost poskytující platební a transakční služby již od 70. let minulého století. PayU je pak globální společnost, která sídlí v Polsku a v současnosti působí na 18 světových trzích. Tato platební brána se dle [62] zdá být ideální volbou pro obchodování na mezinárodní úrovni, protože například v rámci EU se s ní dá platit ve všech měnách. Zbývající Stripe je původem americká společnost, která pochází ze San Francisca a vznikla v roce 2011. U nás je však zprostředkována od roku 2020 britskou společností. Jedná se o „*populární a rychle rostoucí fintech*[22] *startup*“, který si rychle „*získává oblibu českých obchodníků, stejně jako těch světových*“. Umožňuje platby v téměř všech měnách, avšak je potřeba započítat náklady související s konverzí. [66, 67, 63, 65]

3.5.2 Typ brány

Dalším důležitým faktorem, který zde rozeberu a v rámci kterého můžeme pozorovat rozdílnost, je typ brány, který dané platební brány poskytují. Jedná se o tři typy bran – inline brána, redirect brána a mobile brána.

Inline brána (někdy také tzv. *iFrame*) je taková, kdy se „*platební okno zobrazuje přímo nad stránkami e-shopu*“. V tomto případě nedochází k žádnému přesměrování a vše se děje na pozadí naší aplikace, avšak zadání veškerých zákazníkových údajů je i tak zabezpečeno a prodávající k nim nemá přístup. [62]

Redirect brána je taková, kdy se „*platební okno neotevře na pozadí e-shopu, ale na samostatné stránce společnosti zajišťující platební bránu. Tato stránka je zajištěná speciálním certifikátem, takže je ve výsledku platba pro zákazníka stejně bezpečná, jako u inline brány*“. Její jistou nevýhodou se může zdát být fakt, že zákazník opustí naši aplikaci a dostane se do neznámého prostředí, což na nákup nemá pozitivní vliv, avšak stránku, na kterou je zákazník přesměrován lze upravovat a docílit tak pocitu, že naši aplikaci neopustil. [62]

Posledním možným typem brány je mobile brána. Jedná se o speciální případ brány, která „*nabízí webové prostřední optimalizované pro mobily tak, aby zákazník vše viděl v dostatečné velikosti a platba se mu pohodlně prováděla pomocí prstu na monitoru mobilního zařízení. (...) Mobilní brána se většinou automaticky přizpůsobí zařízení, ve kterém zákazník nakupuje. Pokud nakupuje na počítači, zobrazí se inline nebo redirect brána. Pokud ale nakupuje v mobilu, automaticky se otevře jako mobilní brána*“. Tento typ brány může být výhodou, protože pokud zákazník provádí nákup na svém mobilním zařízení a není to komfortní, může ho to odradit. [62]

Na základě [62] všechny čtyři zmíněné platební brány umožňují redirect bránu. Dle tohoto srovnání mají všechny ze zmíněných platebních bran kromě PayU, u které to nebylo zjištěno, také iFrame. Na základě podrobnějšího průzkumu [66] se však ukazuje, že i PayU tuto možnost nabízí.

[22] Fintech označuje pojem finančně-technologický.

Co se týče mobile brány, je poskytována pouze v případě GoPay a PayU. ComGate však důležitost tohoto prvku neopomíjí, pouze používá rozdílné řešení – obsah, který zobrazuje v případě iFrame či redirect brány, je mobilnímu prohlížeči uzpůsoben a je responzivní. Obdobný přístup k řešení tohoto problému pomocí responzivity zaujímá i Stripe. Responzivní design nabízí všechny ze zmíněných platebních bran. [62, 63, 65]

Závěrem lze tedy říci, že porovnávané platební brány jsou dobře použitelné ve všech třech podobách a toto důležité kritérium pro mě nebude rozhodujícím při výsledné volbě.

3.5.3 Cenové podmínky

Posledním důležitým kritériem, které zde zhodnotím je cenová dostupnost jednotlivých platebních bran. Detailní cenové podmínky vytvořené na základě [66, 62, 68] jsem pro přehlednost uvedl v tabulce 3.6.

Tabulka 3.6 Cenové podmínky platebních bran ComGate, GoPay, PayU a Stripe v roce 2022

	ComGate	GoPay	PayU	Stripe
Procentní poplatek z transakce	0,59–0,79 %	0,90–2,20 %	1,60 %	1,4–2,9 %
Fixní poplatek z transakce	1,80–2,40 Kč	3 Kč	1 Kč	6,50 Kč
Aktivační poplatek	0 Kč	0 Kč	999 Kč	0 Kč
Měsíční poplatek	0–149 Kč	0–190 Kč	0 Kč	0 Kč
Převod na bankovní účet	0 Kč	10 Kč	0 Kč	8Kč + 0,25 %

Brilantně ze srovnání z hlediska ceny vychází v základu platební brána ComGate, jejíž poplatek z transakce (fixní i procentuální) se odvíjí od toho, kolik peněz nám přes ni naši klienti zaplatí. Na základě toho, zda obrat přesáhne 50 000 Kč se také určuje, zda budeme platit měsíční poplatek 149 Kč za její využívání (v případě přesažení neplatíme). Aktivace i případné zrušení je bezplatné. [69]

GoPay má k stanovování poplatků podobný přístup jako ComGate. Jejich výše se také odvíjí od velikosti obratu, dle kterého vytváří tři kategorie. Pokud máme obrat do 15 000 Kč měsíčně, platíme nejvyšší procentuální poplatek za transakci, fixní poplatek, měsíční poplatek i bankovní převod. Pokud tuto hranici přesáhneme, zbavíme se měsíčních poplatků a procentuální poplatky se začnou odvíjet dle obratu jako u ComGate. Pokud přesáhneme i hranici 50 000 Kč, veškeré poplatky z transakcí i z bankovního převodu se stanoví na míru. Aktivace i případné zrušení je opět zdarma. [70]

V případě PayU je přístup odlišný. Poplatky zde nejsou děleny dle výše obratu, avšak v případě obratu vyššího než 2 miliony nabízí individuální řešení na míru. Pokud tuto hranici nepřesáhneme, platíme fixní i procentuální poplatek z každé transakce. Brána však nevyžaduje žádné měsíční poplatky ani poplatky za bankovní převod. Jedná se o jedinou z porovnávaných bran, v jejímž případě je aktivace placená, nicméně nabízí zkušební verzi, na základě které se můžeme rozhodnout, zda nám PayU vyhovuje. [64]

Stripe rozlišuje, zda dochází k platbě z karty evropské či jiné. Dle toho stanovuje evropským kartám nižší procentuální poplatek (1,4 %). Ostatní mají poplatek o něco vyšší (2,9 %, kromě Velké Británie, kde je poplatek speciální – 2,5 %). Součástí poplatků je vždy i fixní poplatek a poplatek za bankovní převod. V současné době nejsou účtovány žádné měsíční poplatky. Stripe se sice na jednu stranu zdá být poměrně striktní, co se poplatků týče, na druhou stranu však nabízí prostor k diskuzi. Umožňují kontakt s podporou a domluvu specifických podmínek, a to nejen v případě výššího obratu, ale například i u unikátních produktů. Aktivace platební brány je opět bezplatná. [62, 68]

3.5.4 Volba platební brány

Na základě provedené analýzy se mi ukazují dvě hlavní rozhodující kritéria pro volbu platební brány. Prvním je mezinárodní zaměření a druhým a je přístup k ceně a poplatkům.

Nejméně vhodná z hlediska zaměření se jeví ComGate, která je silně spojována hlavně s českým trhem. Ze zbývajících tří se zdá, že nejvhodnější z hlediska mezinárodního zaměření budou platební brány PayU a Stripe. Stripe pak obzvláště díky své rostoucí popularitě, kterou může podpořit i existující možnost využití zmíněných elektronických peněženek.

Na základě srovnání z hlediska ceny se mi pro účely tohoto projektu zdají být vhodné platební brány ComGate či Stripe. V případě ComGate je velice lákavý stabilně poměrně nízký poplatek, na druhou stranu v případě Stripe jsou sice poplatky vyšší, ale daly by se domluvit individuálně. GoPay sice perfektně rozlišuje poplatky dle výše obratu, nicméně v případě nižšího obratu jsou poplatky méně výhodné. PayU pak na druhou stranu neumožňuje žádné individuální výhody do specifické hranice 2 milionů – na rozdíl od Stripu, který je otevřený diskuzi.

Výše zmíněná dvě kritéria dle mé analýzy nejlépe splňuje platební brána Stripe. Abych si ověřil její vhodnost pro práci se zvolenými uživatelskými tarify, kterým se bude detailněji věnovat kapitola 4.1, emailem jsem kontaktoval její podporu, která mi potvrdila, že brána by měla být pro účely tohoto projektu dobře použitelná. Jejich odezva byla v řádu dnů, tedy poměrně rychlá, což je dalším pozitivem. Stripe je proto brána, kterou bych využil pro budoucí implementaci tohoto synchronizačního nástroje.

Kapitola 4
Návrh

V této kapitole nejprve vysvětlím návrh jednotlivých uživatelských tarifů, tedy možných členství. Následně provedu návrh samotné aplikace. Vysvětlím její rozdělení a po jednotlivých navržených stránkách rozeberu naplnění funkčních požadavků prostřednictvím případů užití. Na závěr rozeberu návrh propagačního webu a jeho navázání na samotnou aplikaci.

4.1 Volba uživatelských tarifů

Pro samotnou aplikaci i pro propagační web je důležité zvolit vhodné možnosti členství (ekvivalentně nazývané tarif či plán) pro uživatele. Vzhledem k tomu, že návrh samotné aplikace byl rozsáhlejší než návrh propagačního webu, s návrhem vhodných cenových rozmezí mi pomohl Ben, který na základě [71] zjistil, že v současné době je v České republice v IT odvětví asi 316 000 lidí a průměrný plat je zhruba 68 000 Kč. Dle [72] platí, že zhruba 3,5 % lidí, kteří v IT odvětví narazí na produkt, si ho zakoupí. Z toho vyplývá, že jen v rámci České republiky máme potenciálně až 11 000 uživatelů.

Na základě toho, že celý náš tým nástroj Timer2Ticket v současném stavu používal, jsme došly k závěru, že oproti manuálnímu zaznamenávání času a jeho následnému přenášení tímto nástrojem ušetříme v průměru asi 10 minut denně. Pokud budeme počítat s odhadem 20 pracovních dní (s osmihodinovou pracovní dobou) na měsíc, používáním současného Timer2Ticketu ušetříme 3,33 hodiny měsíčně, což odpovídá 1415 Kč (což je zhruba 63 $ při kurzu 22,46 korun za dolar).

Vzhledem k tomu, že návrh nového uživatelského rozhraní této aplikace musí počítat s tím, že budou do budoucna přidány další nástroje pro synchronizaci a bude umožněno provádět synchronizaci mezi více dvojicemi nástrojů současně, rozhodli jsme se, že jednotlivé tarify rozdělíme dle možného počtu souběžně synchronizovaných dvojic (tzv. aktivních spojení) a dle toho, v jak pravidelných intervalech bude umožněna synchronizace.

Přípravené tarify uvedené v tabulce 4.1 se mně i mému týmu zdají být pro potenciálního uživatele akceptovatelné a možná i atraktivní s ohledem na výše uvedený uspořený čas přepočítaný na peníze. Zároveň se ceny neliší výrazně od cen, se kterými se mohli setkat v jim blízkých aplikacích jako je například Jira [8], jejíž ceny se pohybují od 0 do 14,5 $. Při zvoleném rozdělení tarifů má zákazník možnost vyzkoušet aplikaci v omezené míře v rámci Hobby členství, které je zdarma. V tomto případě má jen jedno aktivní spojení a k pravidelné synchronizaci může docházet nejvýše jednou týdně. Junior členství, které stojí 5 $ měsíčně, umožní mít dvě aktivní spojení a pravidelné synchronizace provádět jednou denně. Toto členství je dobře použitelné, nejlepší formu však nabízí Senior členství, které umožní až 5 aktivních spojení a pravidelné synchronizace každou hodinu za cenu 8 $ měsíčně. Jeho cena má být motivační – uživatel získá za malé navýšení útraty výrazně více možností.

Tabulka 4.1 Volba uživatelských tarifů

	Hobby	Junior	Senior
Měsíční cena[1]	0 $	5 $	8 $
Počet aktivních spojení	1	2	5
Možnost pravidelné synchronizace	jednou týdně	jednou denně	jednou za hodinu

Aby členství nebylo tak striktně omezeno na tři skupiny, společně s týmem jsem došel k závěru, že bude vhodné umožnit ještě dokupovat samotná aktivní spojení. Uživatel, který se například rozhodne využít Junior členství, bude mu vyhovovat pravidelnost synchronizace, ale bude potřebovat tři aktivní spojení, má možnost si navíc zakoupit aktivní spojení za cenu 2 $ měsíčně (tím povýší základní možnosti tohoto členství). Tato cena je stejná pro dokupování v rámci všech členství.

Aby vytvořený model tarifů fungoval, není možné, aby bylo uživateli umožněno kdykoliv provádět okamžité synchronizace. Tento prvek bude proto vyčleněn. Uživatel si je bude moci zakoupit kdykoliv v balíčcích po 20 okamžitých synchronizací za cenu 4 $. Cena byla stanovena takto, aby byl uživatel motivován platit si členství typu Senior, kdy má umožněné pravidelné synchronizace každou hodinu. Balíček po 20 byl stanoven s ohledem na platební bránu a poplatky s ní spojené.

Na základě diskuzí se zákazníkem vyplynulo, že by rád nějakým způsobem odměnil každého, kdo v rámci propagačního webu využije možnost darovat projektu peníze. Pro účely toho jsem se rozhodl využít právě okamžitých synchronizací – dárce dostane za darování kupón, který pokud v rámci svého účtu v samotné aplikaci vloží, dostane okamžité synchronizace. Přesný počet není v rámci této práce stanoven.

4.2 Návrh samotné aplikace

Na základě sběru požadavků vyplynulo celkem šest oblastí (stránek), do kterých jsem tento návrh rozdělil. Ještě než se však dostanu k návrhu jednotlivých stránek, boční lišty a záhlaví, popíšu stavy uživatele, ve kterých se může nacházet, a stavy, kterých může nabývat konkrétní spojení. Následně popíšu návrh spojený se správou uživatelů. V další části se zaměřím na návrh stránky, která se bude zabývat jednotlivými spojeními. Dále popíšu návrh stránky s jednotlivými logy z aplikace. Na závěr pak rozeberu stránky uživatelského profilu. Pro snazší pochopení některých případů užití zde uvedu stránky vytvořené prostřednictvím zvoleného nástroje pro tvorbu wireframů.

Při návrhu jsem se snažil dbát na Nielsenovu heuristickou analýzu, která popisuje obecné principy pro návrh dobrého uživatelského rozhraní. Jakob Nielsen tyto principy rozdělil celkem do deseti skupin [73]. Pro účel mého návrhu mi přišlo nejpodstatnější následující:

- Uživatelé by měli být neustále informování o tom, co se v aplikaci děje, a to v jim blízké mluvě.
- Uživatelské rozhraní by mělo být konzistentní a držet krok s konvencemi, na něž jsou jeho uživatelé již zvyklí.
- Je podstatné myslet na prevenci chyb, které se mohou objevit.
- Uživatel se může snadno dostat do nechtěných situací a je potřeba mu vždy navrhnout nějaké snadné řešení.

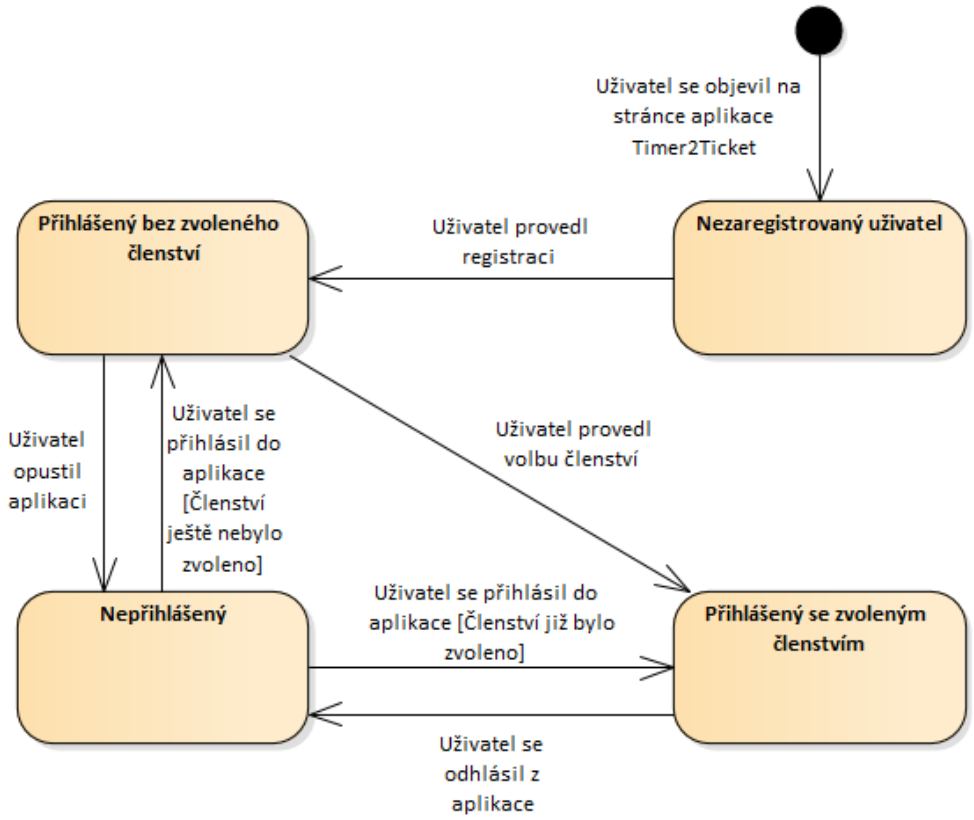
[1] Cena je stanovena v dolarech s ohledem na dříve provedené zaměření zákazníka.

- Uživatel by měl veškeré podstatné informace mít vždy na místě, kde je potřebuje. Není žádoucí, aby si je musel pamatovat z jiného místa. Zároveň by však neměl být zatěžován nepodstatnými informacemi.
- Je nejlepší, když uživatel ze systému sám pochopí, jak má čeho docílit a přirozeně mu rozumí. Pokud to není možné, je podstatné mu na vhodných místech poskytnout nápovědu.

4.2.1 Důležité stavy prvků aplikace

V této části vysvětlím, jakých stavů může nabývat entita uživatele a přechody mezi nimi. Obdobně pak popíšu entitu spojení. Pro usnadnění pochopení této problematiky zde využiji znalostí z předmětu SI1 a notace UML[2], a to konkrétně stavového diagramu, který patří do skupiny diagramů chování. K vytvoření potřebných diagramů využiji nástroje Enterprise Architect, s nímž jsem se seznámil také ve zmíněném předmětu. [74]

Obrázek 4.1 Stavy uživatele (vytvořeno v nástroji Enterprise Architect)



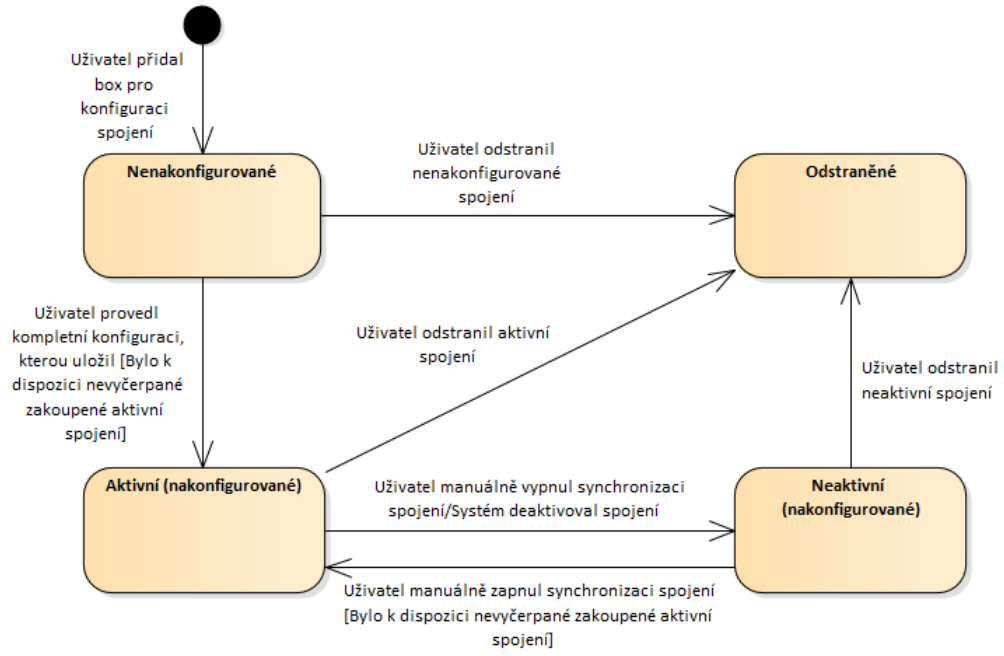
4.2.1.1 Stavy uživatele

Uživatel, který se dostane na stránky samotné aplikace má možnost se zaregistrovat či do aplikace přihlásit, pokud již má účet. Pokud uživatel provede registraci, stává se rovnou přihlášeným uživatelem, avšak nemá zvolené členství. V aplikaci může v tuto chvíli provádět volbu svého

[2]UML je zkratkou pro Unified Modeling Language, tedy jednotný modelovací jazyk využívaný v softwarovém inženýrství.

členství. Kromě toho může aplikaci opustit a stát se nepřihlášeným uživatelem. Pokud se později přihlásí, vrátí se do stavu přihlášeného uživatele bez zvoleného členství. Pokud v tomto stavu provede volbu členství, stane se uživatelem přihlášeným se zvoleným členstvím a do původního stavu už se nikdy nedostane. Takový uživatel může již aplikaci používat plně v rámci zvoleného členství. Odhlášením se stává nepřihlášeným. Následným přihlášením dojde k opětovnému přesunu do stavu přihlášeného uživatele se zvoleným členstvím.

Obrázek 4.2 Stavy spojení (vytvořeno v nástroji Enterprise Architect)



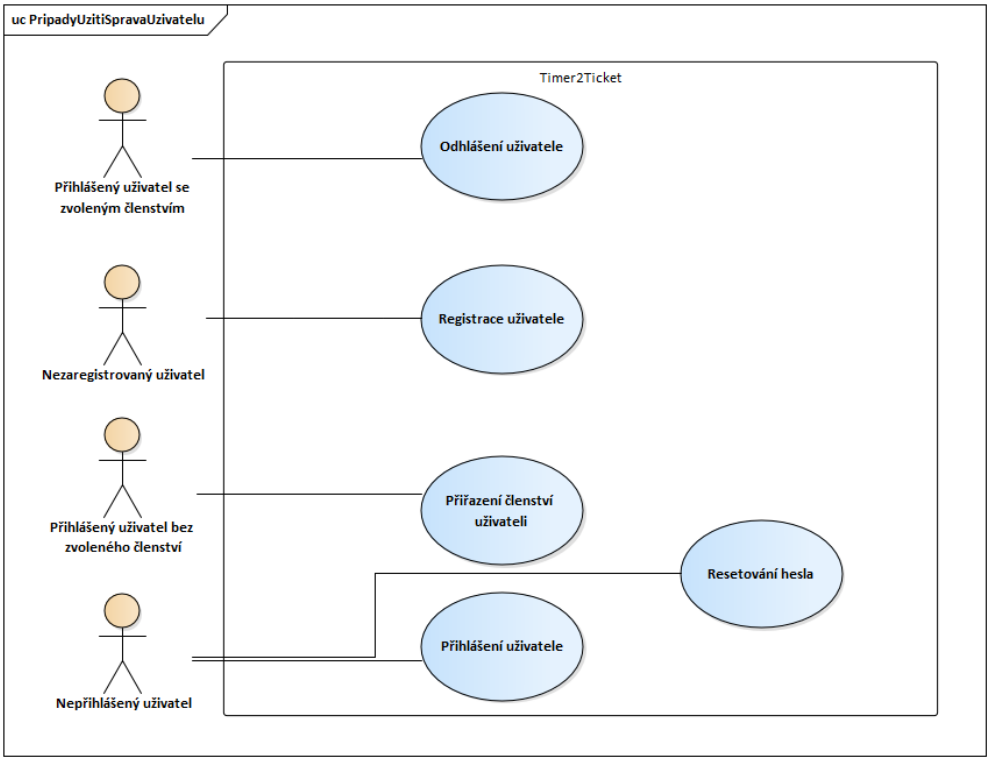
4.2.1.2 Stavy spojení

Spojení reprezentuje konkrétní dvojici nástrojů, které se dají nakonfigurovat pro synchronizaci (například jeden konkrétní Redmine a jeden konkrétní Toggl Track workspace). Uživatel bude spravovat jednotlivá spojení prostřednictvím zabalovatelných boxů. Pokud takový box na stránku přidá, získá nenakonfigurované spojení. Těchto spojení může mít neomezený počet.

Jakmile uživatel dokončí konfiguraci obou nástrojů určených k synchronizaci a stiskne tlačítko pro uložení, spojení se stává aktivním (nakonfigurovaným) a zahájí se první synchronizační proces. K přechodu do tohoto stavu je potřeba, aby uživatel měl k dispozici ještě nepoužité zakoupené aktivní spojení v rámci svého členství, které zde může vyčerpat. Je jich tedy omezený počet. V tomto stavu může uživatel synchronizaci kdykoliv manuálně vypnout a přesunout tak dané spojení do neaktivního stavu (avšak stále nakonfigurovaného). Do tohoto stavu se může aktivní spojení dostat i ve chvíli, kdy uživateli vyprší členství, sníží se mu počet aktivních spojení a systém spojení deaktivuje, aby dodržel počet aktivních spojení odpovídajících danému členství. Těchto spojení může mít uživatel opět nekonečně mnoho. Pokud má uživatel k dispozici alespoň jedno nevyčerpané zakoupené aktivní spojení, může ho použít a opět provést manuální aktivaci. V tomto případě nedojde k okamžitému synchronizačnímu procesu. Uživatel pouze obnoví aktivitu naplánového pravidelného synchronizačního cyklu.

Ze všech tří zmíněných stavu nakonec může uživatel kdykoliv spojení smazat, čímž se stane odstraněným a uživatel již neuvidí box tohoto spojení. Takové aktivní spojení samozřejmě přestane provádět naplánované synchronizace.

Obrázek 4.3 Diagram případů užití pro správu uživatelů vytvořený v nástroji Enterprise Architect



4.2.2 Návrh správy uživatelů

V této části popíšu návrh oblastí, do kterých se dostane uživatel, který ještě není registrovaný či přihlášený do aplikace. Zároveň popíšu prostředí uživatele, který ještě nemá zvolené členství a je přihlášený. Svůj návrh založím na případech užití realizujících jednotlivé funkční požadavky. Přehled jednotlivých případů užití souvisejících se správou uživatelů je na digramu 4.3. Na tvorbě wireframů v nástroji Balsamiq se v této části podílela Irina Kara-Sel.

UC1: Registrace uživatele

Tento případ užití neboli use case (tzv. UC) umožní uživateli, který ještě není registrovaný, aby se zaregistroval do aplikace prostřednictvím své emailové adresy a hesla. Návrh obrazovky pro registraci, který je k dispozici na přiloženém médiu, předpokládá, že zde bude také tlačítko, které umožní registraci prostřednictvím služby Google. Zároveň se v rámci formuláře počítá s odkazem, který umožní přesměrování na stránku s přihlášením. K dispozici zde budou také odkazy na smluvní podmínky.

Aktéři: Nezaregistrovaný uživatel

Počáteční podmínky: Nezaregistrovaný uživatel se nachází na stránce s registračním formulářem (buď se sem dostane přímo či je sem přesměrován z propagačního webu).

Základní scénář:

1. Uživatel vyplní emailovou adresu.
2. Systém ověří validitu formátu emailové adresy.

3. Systém informuje o úspěšném zadání emailové adresy.
4. Uživatel vyplní heslo.
5. Systém ověří požadavky na dostatečnou kvalitu hesla.
6. Systém informuje o úspěšném splnění požadavků na kvalitu hesla.
7. Uživatel vyplní potvrzení hesla.
8. Systém ověří shodu zadaných hesel.
9. Systém informuje o úspěšné shodě zadaných hesel.
10. Uživatel klikne na tlačítko „Registrovat“.
11. Systém zkontroluje, že ještě neexituje účet pod zadanou emailovou adresou, zaregistruje uživatele a přesune ho na stránku s volbou členství, kde ho informuje o úspěšné registraci.

Exception scénář: Tento scénář se spustí ve chvíli, kdy nastane problém s validitou formátu emailové adresy ve 2. kroku základního scénáře.

1. Systém uživatele informuje o problému s formátem zadané emailové adresy.
2. Uživatel pokračuje 1. krokem základního scénáře.

Exception scénář: Tento scénář se spustí ve chvíli, kdy nejsou splněny požadavky na kvalitu hesla v 5. kroku základního scénáře.

1. Systém uživatele informuje o nesplněných požadavcích na kvalitu hesla.
2. Uživatel pokračuje 4. krokem základního scénáře.

Exception scénář: Tento scénář se spustí ve chvíli, kdy je v 8. kroku základního scénáře zjištěno, že zadaná hesla nejsou stejná.

1. Systém uživatele informuje o rozdílnosti hesel.
2. Uživatel pokračuje 7. krokem základního scénáře.

Exception scénář: Tento scénář se spustí ve chvíli, kdy je v 11. kroku základního scénáře zjištěno, že již existuje účet zaregistrovaný pod zadanou emailovou adresou.

1. Systém uživatele informuje o nemožnosti vytvoření dalšího účtu pod zadanou emailovou adresou.
2. Uživatel zadá jinou emailovou adresu.
3. Systém ověří validitu formátu emailové adresy.[3]
4. Systém informuje o úspěšném zadání emailové adresy.
5. Uživatel pokračuje 10. krokem základního scénáře.

UC2: Přiřazení členství uživateli

Uživatel se po registraci dostane do aplikace, avšak ještě nemůže využívat jejích služeb, protože je bez členství. Tento případ užití uživateli umožní volbu členství, což mu umožní další používání aplikace. Návrh obrazovky pro volbu prvního členství, který je případně k dispozici na přiloženém médiu, počítá také se zobrazením hlášky o úspěšně dokončené registraci. Zároveň se do budoucna počítá s tím, že pokud uživatel přijde do aplikace z propagačního webu přes odkaz na konkrétní tarif, tento tarif mu zde bude přednastaven.

[3] Opět může proběhnout exception scénář reagující na nevalidní formát.

Aktéři: Přihlášený uživatel bez zvoleného členství

Počáteční podmínky: Nezaregistrovaný uživatel byl po registraci přesměrován do aplikace nebo se do aplikace opětovně přihlásil uživatel, který po registraci nezvolil žádné z nabízených členství a aplikaci opustil.

Základní scénář:

1. Uživatel z nabídky zvolí chtěné členství (Junior či Senior)[4].
2. Uživatel klikne na tlačítko „Pokračovat".
3. Systém zobrazí formulář s platebními údaji (DIČ, jméno a příjmení (jméno společnosti), ulice, PSČ, město, země).
4. Uživatel vyplní formulář.
5. Uživatel klikne na tlačítko „Pokračovat".
6. Systém zkontroluje, že byl formulář vyplněn korektně.
7. Systém přesměruje uživatele na platební bránu.
8. Uživatel prostřednictvím platební brány provede platbu členství.
9. Systém přesměruje uživatele zpět do aplikace na stránku „Spojení" a informuje ho o úspěšném nabytí členství.

Alternativní scénář: Uživatel v 1. kroku zvolí jako chtěné členství Hobby plán a následně klikne na tlačítko „Pokračovat". V takovém případě nedochází k žádné platbě a scénář pokračuje 9. krokem základního scénáře.

Alternativní scénář: Uživatel ve 2. kroku zvolí možnost „Přeskočit". Tento alternativní scénář je validní i pro volbu Hobby členství v prvním kroku. V takovém případě uživatel přeskakuje svoji volbu členství a je mu přiřazeno zdarma Hobby členství. Scénář pokračuje 9. krokem základního scénáře.

Alternativní scénář: Uživatel v 5. kroku zvolí možnost „Zpět". V takovém případě systém uživatele vrací k nabídce jednotlivých členství, kde pokračuje 1. či 2. krokem základního scénáře (na základě toho zda chce svoji původní volbu změnit).

Exception scénář: Tento scénář se spustí ve chvíli, kdy systém v 6. kroku základního scénáře zjistí, že formulář není vyplněn korektně, tedy chybí požadované povinné položky – jméno a příjmení (jméno společnosti).

1. Systém uživatele informuje o chybějících údajích.
2. Uživatel pokračuje 4. krokem základního scénáře.

UC3: Přihlášení uživatele

Tento případ užití umožní nepřihlášenému uživateli, aby se do aplikace přihlásil prostřednictvím své emailové adresy a hesla. Návrh obrazovky pro přihlášení, který je k dispozici na přiloženém médiu, předpokládá, že zde bude také tlačítko, které umožní přihlášení prostřednictvím služby Google. Zároveň se v rámci formuláře počítá s tlačítkem, které umožní přesměrování na stránku s registrací. K dispozici zde bude také odkaz na stránku pro resetování zapomenutého hesla.

Aktéři: Nepřihlášený uživatel

[4] Hobby členství je rozebráno zvláště v alternativním scénáři.

Počáteční podmínky: Nepřihlášený uživatel se nachází na stránce s přihlašovacím formulářem (buď se sem dostane přímo či je sem přesměrován z propagačního webu).

Základní scénář:

1. Uživatel vyplní emailovou adresu a heslo.
2. Uživatel klikne na tlačítko „Přihlásit“.
3. Systém ověří zda je možné se pomocí zadaných údajů přihlásit k uživatelově účtu.
4. Systém přesměruje uživatele do aplikace. Pokud se jedná o uživatele se zvoleným členstvím, dostane se na stránku „Spojení“. Pokud členství zvolené ještě nemá, dostane se do prostoru s volbou členství.

Exception scénář: Tento scénář se spustí ve chvíli, kdy systém ve 3. kroku základního scénáře zjistí, že uživatel se zadanou emailovou adresou a heslem neexituje.

1. Systém uživatele informuje o chybně zadaných přihlašovacích údajích.
2. Uživatel pokračuje 1. krokem základního scénáře.

UC4: Resetování hesla

Tento případ užití uživateli umožní provést resetování jeho hesla v případě, že ho zapomene.

Aktéři: Nepřihlášený uživatel

Počáteční podmínky: Nepřihlášený uživatel se nachází na stránce s přihlašovacím formulářem.

Základní scénář:

1. Uživatel klikne za odkaz pro resetování hesla.
2. Systém zobrazí formulář pro zadání emailové adresy, pro jejíž uživatelský účet má být heslo resetováno.
3. Uživatel vyplní emailovou adresu pod níž má zaregistrovaný svůj účet.
4. Uživatel klikne na tlačítko „Odeslat odkaz“.
5. Systém ověří, že pod zadanou emailovou adresou je nějaký účet zaregistrován.
6. Systém uživatele přesměruje na stránku s přihlášením, kde ho informuje, že na zadanou emailovou adresu byl zaslán email s odkazem pro resetování hesla.
7. Uživatel se prostřednictvím rozkliknutí zaslaného odkazu dostane na stránku se změnou hesla pro svůj účet.
8. Uživatel vyplní nové heslo.
9. Systém ověří, že zadané heslo splňuje požadované parametry na kvalitu.
10. Systém informuje uživatele o splnění požadavků na heslo.
11. Uživatel vyplní potvrzení hesla.
12. Systém ověří, že zadané potvrzení hesla je shodné se zadaným novým heslem.
13. Systém informuje uživatele o úspěšném ověření shody hesel.
14. Uživatel klikne na tlačítko „Resetovat heslo“.
15. Systém uživatele přesměruje na stránku s přihlášením, kde ho informuje, že heslo bylo úspěšně změněno.

Alternativní scénář: Uživatel ve 4. kroku zvolí možnost „Zrušit“. Takový krok značí, že uživatel chce ukončit resetování hesla. V takovém případě systém uživatele přesměruje na stránku přihlášení.

Exception scénář: Tento scénář se spustí ve chvíli, kdy je v 5. kroku základního scénáře zjištěno, že pod zadanou emailovou adresou ještě není registrován žádný účet.

1. Systém uživatele informuje o tom, že pod zadanou emailovou adresou ještě není zaregistrován žádný uživatelský účet.
2. Uživatel pokračuje 3. krokem základního scénáře.

Exception scénář: Tento scénář se spustí ve chvíli, kdy nejsou splněny požadavky na kvalitu hesla v 9. kroku základního scénáře.

1. Systém uživatele informuje o nesplněných požadavcích na kvalitu hesla.
2. Uživatel pokračuje 8. krokem základního scénáře.

Exception scénář: Tento scénář se spustí ve chvíli, kdy je ve 12. kroku základního scénáře zjištěno, že zadaná hesla nejsou stejná.

1. Systém uživatele informuje o rozdílnosti hesel.
2. Uživatel pokračuje 11. krokem základního scénáře.

4.2.3 Návrh boční lišty

Boční lišta bude součástí stránky „Spojení“, stránky „Logy“ i stránky „Profil“. Bude rozdělená na dvě části. V horní části bude logo aplikace Timer2Ticket, ve spodní části bude menu, které bude zajišťovat přesměrování mezi jednotlivými zmíněnými stránkami. Spodní část bude obsahovat také tlačítko „Upgrade“, které uživatele přemístí na stránku „Členství“, na které bude moci změnit, a tedy i provést upgrade, svého členství.

Neboť nejsem grafik, přišlo mi vhodné přenechat tvorbu loga někomu, kdo s tím má již zkušenost. Po dohodě se svým zadavatelem, Ing. Jiřím Hunkou, jsem se proto obrátil na společnost Jagu s. r. o., která má svého grafika Lukáše Olšu. Toho jsem za účelem vytvoření loga kontaktoval prostřednictvím systému Redmine, který společnost Jagu s. r. o. pro správu úkolů využívá. Jako výstup práce, kterou jsem mu zadal, vzniklo logo, které bude součástí stránek prototypové impementace.

4.2.4 Návrh záhlaví

Záhlaví bude stejně jako boční lišta součástí stránky „Spojení“, stránky „Logy“ i stránky „Profil“. Jeho levá část bude navazovat na logo a bude obsahovat informaci o stránce, na níž se uživatel nachází. V pravé části bude mít uživatel informaci o počtu dostupných okamžitých synchronizací a s tím spojené tlačítko „Koupit okamžité synchronizace“, které uživatele přemístí na stránku „Členství“, na které bude moci zakoupit další okamžité synchronizace. Vedle tohoto tlačítka bude nakonec také tlačítko, které uživatele odhlásí.

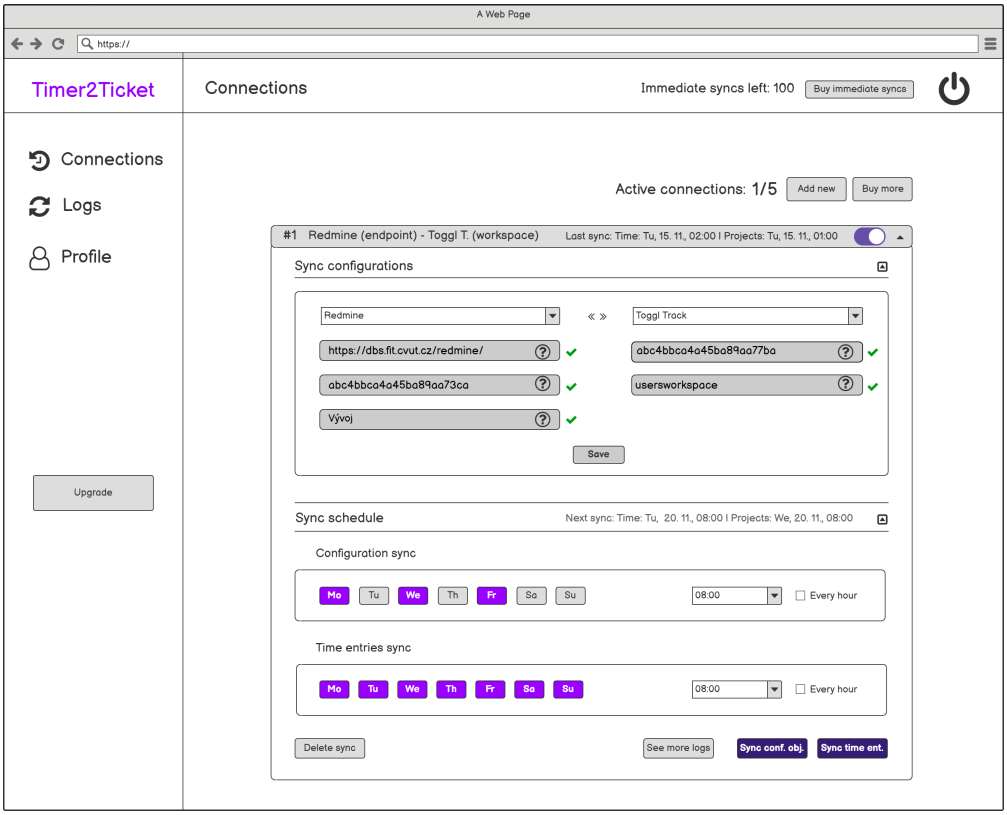
UC5: Odhlášení uživatele

Tento případ užití umožní uživateli, aby se z aplikace odhlásil. Toho docílí prostým klepnutím na tlačítko pro odhlášení v záhlaví stránek. Pro jednoduchost případu užití zde neuvádím scénář.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Uživatel již provedl volbu svého členství a nachází se na libovolné stránce ve svém uživatelském účtu.

Obrázek 4.4 Návrh stránky Spojení vytvořený v nástroji Balsamiq



4.2.5 Návrh stránky Spojení

V této části popíšu návrh stránky, která bude uživatelem pravděpodobně nejpoužívanější, protože na ní bude spravovat jednotlivá spojení a jejich synchronizaci. Návrh opět založím na případech užití realizujících jednotlivé funkční požadavky. Přehled jednotlivých případů užití souvidejících se správou spojení je na digramu 4.5. Wireframy v této části jsme vytvořili ve spolupráci s Irinou Kara-Sel. Finální návrh ukazující stránku s jedním aktivním spojením a všemi otevřenými boxy vytvořený v nástroji Balsamiq je pro snadnější pochopení na obrázku 4.4.

UC6: Přidání boxu spojení

Tento případ užití umožní uživateli přidat box, ve kterém dojde k nastavení synchronizace. Těchto boxů může být obecně více, než může být aktivních spojení, ve kterých se provádí synchronizace, a pokud není přítomen žádný, uživatel je informován, že nemá ještě žádné spojení.

Aktéři: Přihlášený uživatel se zvoleným členstvím

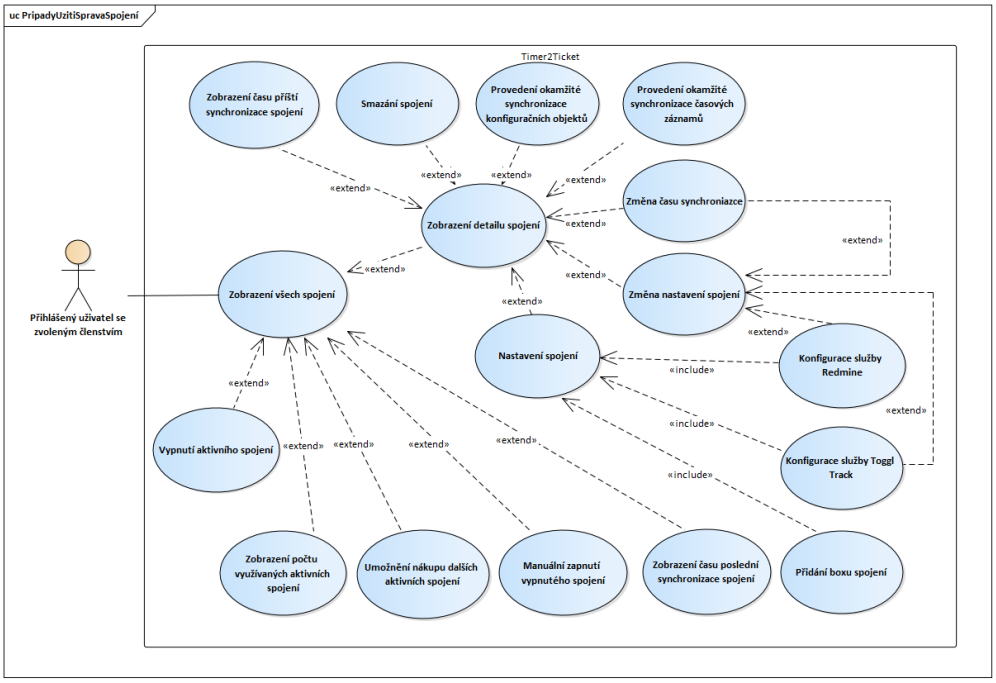
Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení“.

Základní scénář:

1. Uživatel klikne na tlačítko „Přidat nové“ v pravém rohu stránky.

2. Systém uživateli na tuto stránku přidá nový box, ve kterém může provést konfiguraci nové synchronizační dvojice. Box bude po přidání rozbalený. Bude mít rozbalenou i část na konfiguraci synchronizace. Box s konfigurací času synchronizace zůstane zabalený s obsahem defaultního nastavení.
3. Systém uživatele informuje o úspěšném přidání nového boxu.

Obrázek 4.5 Diagram případů užití pro správu spojení vytvořený v nástroji Enterprise Architect



UC7: Zobrazení všech spojení

Tento případ užití uživateli umožní zobrazit si všechna synchronizační spojení – spojení aktivní, mezi kterými probíhá synchronizace, spojení neaktivní, která ji mají dočasně pozastavenou, i spojení, která jsou nakonfigurována jen částečně. Systém uživateli zobrazí zavřené boxy s jednotlivými synchronizačními dvojicemi.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím má alespoň jedno aktivní, neaktivní či částečně nakonfigurované spojení a nachází se na stránce „Spojení“.

UC8: Umožnění nákupu dalších aktivních spojení

Tento případ užití uživateli umožní, aby se okamžitě dostal ze stránky se spojeními na stránku, ve které bude moci změnit svůj zakoupený počet aktivních spojení.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení“.

Základní scénář:

1. Uživatel klikne na tlačítko „Koupit více“ v pravém rohu stránky.
2. Systém uživatele přesměruje na stránku „Členství“.

UC9: Zobrazení počtu využívaných aktivních spojení

Tento případ užití uživateli umožní zjistit, kolik využívá aktivních spojení z celkového počtu zakoupených aktivních spojení. Tento údaj bude uživateli zobrazen na stránce „Spojení“ v pravé horní části stránky u tlačítek umožňujících nákup dodatečných aktivních spojení a přidání dalších boxů pro spojení.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení“.

UC10: Manuální zapnutí vypnutého spojení

Tento případ užití uživateli umožní, aby zapnul synchronizaci pro nakonfigurované vypnuté (tj. neaktivní) spojení.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení“ a spojení, které chce zapnout je manuálně vypnuté, ale nakonfigurované. Zároveň má uživatel zakoupené a ještě nepoužité alespoň jedno aktivní spojení, které na akci může použít.

Základní scénář:

1. Uživatel klikne na přepínač v záhlaví boxu daného nakonfigurovaného vypnutého spojení.
2. Systém zbarví přepínač.
3. Systém notifikuje uživatele o provedené akci zapnutí neaktivního spojení.

UC11: Vypnutí aktivního spojení

Tento případ užití uživateli umožní, aby vypnul synchronizaci pro aktivní spojení.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení“ a spojení, které chce vypnout je zapnuté (tj. aktivní).

Základní scénář:

1. Uživatel klikne na přepínač v záhlaví boxu daného aktivního spojení.
2. Systém odbarví přepínač na šedivou.
3. Systém notifikuje uživatele o provedené akci vypnutí aktivního spojení.

UC12: Smazání spojení

Tento případ užití uživateli umožní odstranit libovolné spojení (aktivní, neaktivní i ještě nenakonfigurované). Dojde k odstranění celého boxu (tj. i celé konfigurace). V případě aktivního spojení dojde samozřejmě také k jeho vypnutí.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení".

Základní scénář:

1. Uživatel klikne na záhlaví spojení, které chce odstranit.
2. Systém rozbalí box spojení.
3. Uživatel klikne na tlačítko „Odstranit spojení".
4. Systém box odstraní (a případná zapnutá synchronizace se ukončí).
5. Systém notifikuje uživatele o provedené akci odstranění spojení.

UC13: Provedení okamžité synchronizace časových záznamů

Tento případ užití uživateli umožní provést okamžitou synchronizaci časových záznamů pro dané nakonfigurované spojení.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení". Spojení, pro které chce provést okamžitou synchronizaci časových záznamů je nakonfigurované. Uživatel má na akci dostatek okamžitých synchronizací.

Základní scénář:

1. Uživatel klikne na záhlaví spojení, u kterého chce provést okamžitou synchronizaci časových záznamů.
2. Systém rozbalí box spojení.
3. Uživatel klikne na tlačítko „Synchronizovat časové záznamy".
4. Systém provede synchronizaci.
5. Systém notifikuje uživatele o provedené akci okamžité synchronizace.

UC14: Provedení okamžité synchronizace konfiguračních objektů

Tento případ užití uživateli umožní provést okamžitou synchronizaci konfiguračních objektů pro dané nakonfigurované spojení. Vzhledem k tomu, že scénář je analogický s tím v UC13, neuvádím ho zde. Uživatel zde kliká na tlačítko „Synchronizovat konfigurační objekty".

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení". Spojení, pro které chce provést okamžitou synchronizaci časových záznamů je nakonfigurované. Uživatel má na akci dostatek okamžitých synchronizací.

UC15: Nastavení spojení

Tento případ užití uživateli umožní nastavit spojení (tj. realizovat kompletní proces konfigurace) a učinit ho aktivním. Vzhledem ke komplexnosti konfigurace jsem tento případ užití dekomponoval a jeho součástí jsou i UC16 a UC17.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení“ a absolvoval UC6, ve kterém došlo k přidání boxu a rozbalení potřebných částí. Pro provedení tohoto případu užití musí mít ještě nepoužité zakoupené spojení, které se může stát aktivním.

Základní scénář:

1. Uživatel klikne na rozbalovací seznam v libovolném ze dvou sloupců.
2. Systém zobrazí jednotlivé služby, které jsou nabízené k synchronizaci.
3. Uživatel zvolí službu Redmine.[5]
4. Systém zobrazí vyžadovaná pole ke konfiguraci služby Redmine.
5. Uživatel provede konfiguraci služby Redmine dle UC16.
6. Uživatel klikne na rozbalovací seznam v druhém zbývajícím sloupci.
7. Systém zobrazí jednotlivé služby, které jsou nabízené k synchronizaci se službou zvolenou v prvním sloupci.
8. Uživatel zvolí službu Toggl Track.
9. Systém zobrazí vyžadovaná pole ke konfiguraci služby Toggl Track.
10. Uživatel provede konfiguraci služby Toggl Track dle UC17.
11. Uživatel klikne na tlačítko „Uložit“.
12. Systém provede první běh synchronizace, o čemž informuje uživatele, spojení zapne (stane se aktivním) a tlačítko „Uložit“ znefunkční.[6]

Alternativní scénář: Uživatel zvolí ve 3. kroku jako službu, kterou chce konfigurovat, službu Toggl Track a v 8. kroku službu Redmine. V takovém případě dojde ke změně pořadí kroků. Kroky 4 až 5 se provedou místo kroků 9 až 10 a naopak. Pořadí volby služeb je tedy naprosto irelevantní.

UC16: Konfigurace služby Toggl Track

Tento případ užití umožní uživateli provést konfiguraci služby Toggl Track, aby mohlo začít docházet k její synchronizaci.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení“ a absolvoval UC6, ve kterém došlo k přidání boxu a rozbalení potřebných částí. V boxu konfigurace byla vybrána služba Toggl Track. Tento případ užití je součástí UC15, avšak může být proveden i samostatně.

[5] Scénář, který je součástí nastavení spojení v této práci je pouze k Redmine a Toggl Tracku, protože pouze tyto dva systémy jsou v současné době aplikací podporovány. Přidání dalších není součástí této práce.

[6] Tlačítko zde má sloužit zejména pro přehlednost pro uživatele, neboť systém bude jednotlivé konfigurace postupně ukládat při vyplňování.

Základní scénář:

1. Uživatel vyplní API klíč svého účtu v Toggl Tracku.
2. Systém ověří korektnost spojení a načte si dostupné pracovní prostory.
3. Systém informuje uživatele o úspěchu a uloží si API klíč.
4. Uživatel klikne na rozbalovací seznam s dostupnými pracovními prostory.
5. Systém seznam zobrazí.
6. Uživatel zvolí některý z dostupných pracovních prostorů.
7. Systém uživatele informuje o úspěchu a uloží si zvolený prostor.

Exception scénář: Tento scénář se spustí ve chvíli, kdy nastane problém s ověřením spojení a načtením dostupných pracovních prostorů ve 2. kroku základního scénáře.

1. Systém uživatele informuje o problému se zadaným API klíčem.
2. Uživatel pokračuje 1. krokem základního scénáře.

UC17: Konfigurace služby Redmine

Tento případ užití umožní uživateli provést konfiguraci služby Redmine, aby mohlo začít docházet k její synchronizaci.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení“ a absolvoval UC6, ve kterém došlo k přidání boxu a rozbalení potřebných částí. V boxu konfigurace byla vybrána služba Redmine. Tento případ užití je součástí UC15, avšak může být proveden i samostatně.

Základní scénář:

1. Uživatel vyplní API klíč svého účtu v Redmine a bod API.
2. Systém ověří korektnost spojení a načte si dostupné výchozí aktivity pro časové záznamy.
3. Systém informuje uživatele o úspěchu a uloží si API klíč a bod API.
4. Uživatel klikne na rozbalovací seznam s dostupnými výchozími aktivitami.
5. Systém seznam zobrazí.
6. Uživatel zvolí některou z dostupných výchozích aktivit.
7. Systém uživatele informuje o úspěchu a uloží si zvolenou výchozí aktivitu.

Exception scénář: Tento scénář se spustí ve chvíli, kdy nastane problém s ověřením spojení a načtením dostupných výchozích aktivit ve 2. kroku základního scénáře.

1. Systém uživatele informuje o problému se zadanými údaji pro API klíč a bod API.
2. Uživatel pokračuje 1. krokem základního scénáře.

UC18: Zobrazení logů z boxu spojení

Tento případ užití uživateli umožní zjistit detailní informace o jednotlivých synchronizacích, které byly provedeny pro spojení v daném boxu. Jedná se o zobrazení synchronizací časových záznamů i konfiguračních objektů.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení“ v boxu spojení, jehož logy chce zobrazit.

Základní scénář:

1. Uživatel klikne na tlačítko „Prohlédnout si logy“.
2. Systém uživatele přesune na stránku „Logy“.
3. Systém uživateli vyfiltruje takové logy, které proběhly pro spojení v daném boxu.

UC19: Zobrazení detailu spojení

Tento případ užití uživateli umožní zobrazit si nastavení daného spojení. Uživatel toho dosáhne snadno rozkliknutím příslušného boxu spojení, který se dále skládá z boxu pro konfiguraci synchronizovaných služeb a boxu pro konfiguraci času synchronizace. Pro zjištění informací stačí stejně jako v případě hlavního boxu kliknout na záhlaví daných boxů, čímž se otevřou a zobrazí informace. Vnější box zároveň obsahuje veškerá tlačítka spojená s akcemi, které je se spojením možné provádět. Vzhledem k jednoduchosti zde neuvádím scénář (neměl by přidanou hodnotu).

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení“.

UC20: Změna času synchronizace

Tento případ užití uživateli umožní, aby změnil výchozí nastavení pravidelné synchronizace, které je nastavené od doby přidání boxu (za účelem zjednodušení procesu nastavení spojení). Toto nastavení bude uživateli poskytovat nejčastější možnou periodu synchronizace, kterou jeho členství umožňuje. Změněné nastavení půjde později stejným způsobem také měnit. Měnit půjde konfigurace času pro časové záznamy i konfigurační objekty. Pro obojí bude změna probíhat stejně, uvádím ji zde proto v jednom případu užití.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení“ a provedl otevření boxu spojení, jehož konfiguraci času pravidelné synchronizace chce změnit.

Základní scénář:

1. Uživatel klikne na záhlaví druhého boxu, který obsahuje nastavení času synchronizací.
2. Systém otevře box, který bude obsahovat původní nastavení času synchronizace.
3. Uživatel změní nastavení. Rozdílné nastavení lze provést pro synchronizaci časových záznamů a synchronizaci konfiguračních objektů. V případě Hobby členství zde může změnit volbu dne i denní hodiny. Pokud se jedná o Junior členství, může zvolit, které dny v týdnu má k synchronizaci docházet a opět denní hodinu. Pokud jde o Senior členství, může

uživatel navíc zakřížkovat, že má synchronizace ve zvolené dny probíhat každou hodinu periodicky vztaženo k nastavené denní hodině.[7]

4. Systém změněné nastavení času synchronizace uloží.

UC21: Změna nastavení spojení

Tento případ užití umožní uživateli změnit nastavení nakonfigurovaného spojení (tj. aktivního či neaktivního). Změnou je zde myšlena výměna kterékoliv ze dvou zvolených synchronizovaných služeb za jinou a její následná konfigurace i jakákoliv změna v rámci již nakonfigurovaných služeb zvolených v daném spojení (kromě změny výchozí aktivity pro časové záznamy v případě nástroje Redmine). Libovolnou změnu si lze vyložit jako odstranění původního spojení (tedy i jeho okamžitou deaktivaci) a rozpracování konfigurace nového spojení ve stavu nenakonfigurováno.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení". Box nakonfigurovaného spojení, u kterého chce provést změnu nastavení, otevřel přes jeho záhlaví. Otevřel i první vnořený box s konfigurací synchronizovaných služeb.

Základní scénář:

1. Uživatel provede změnu[8] v nastavení spojení. Změny provádí obdobných způsobem jako při samotném nastavování nového spojení.[9]
2. Systém při první změně původní spojení deaktivuje, nahradí ho novým nenakonfigurovaným a zfunkční tlačítko „Uložit".
3. Jakmile uživatel nechce provádět další změny, klikne na tlačítko „Uložit".
4. Systém provede první běh synchronizace, spojení zapne (stane se aktivním) a tlačítko „Uložit" znefunkční.

UC22: Zobrazení času poslední synchronizace spojení

Tento případ užití uživateli umožní zjistit, kdy naposledy došlo k synchronizaci časových záznamů a kdy došlo naposledy k synchronizaci konfiguračních objektů pro dané spojení. Textovou informaci, která obsahuje konkrétní datumy a časy, uživatel nalezne v záhlaví daného boxu spojení. Box není potřeba rozbalovat.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení". Spojení, u kterého chce zobrazit čas poslední synchronizace, je nakonfigurované a proběhl u něj již alespoň jeden běh synchronizace (tj. je aktivní či neaktivní).

UC23: Zobrazení času příští synchronizace spojení

Tento případ užití uživateli umožní zjistit, na kdy je naplánována příští synchronizace časových záznamů a na kdy je naplánována příští synchronizace konfiguračních objektů pro dané spojení.

Aktéři: Přihlášený uživatel se zvoleným členstvím

[7] V případě, že uživatel nemá členství na to, aby využil synchronizace každou hodinu či více dnů v týdnu, možnost je zešedlá.

[8] Co je myšleno změnou, je vydefinováno v popisu tohoto případu užití.

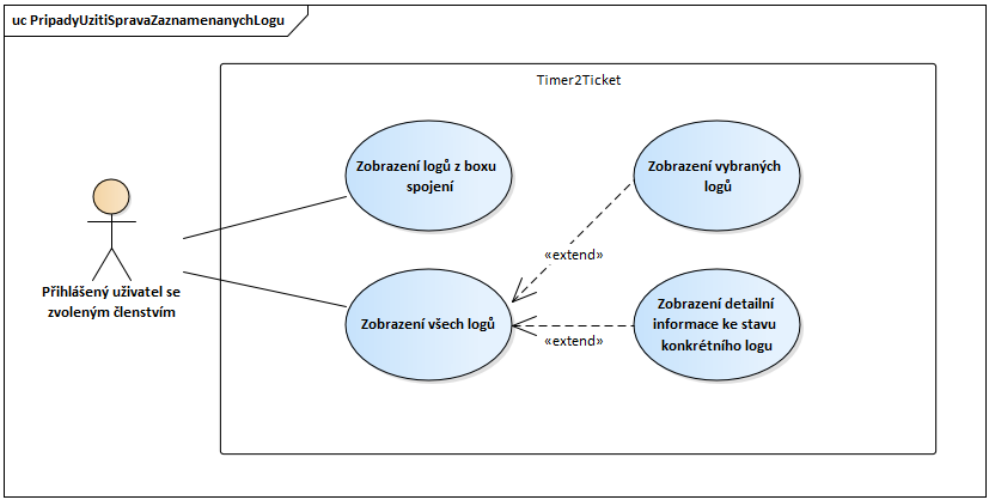
[9] Předpokládá se tedy, že zde mohou proběhnout i kompletní UC16, UC17 či UC20 i s jejich exception scénáři.

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Spojení“. Spojení, u kterého chce zobrazit čas příští synchronizace, je aktivní.

Základní scénář:

1. Uživatel klikne na záhlaví spojení, u kterého chce zjistit čas příští synchronizace.
2. Systém rozbalí box spojení a na druhém řádku, na kterém je k dispozici k otevření box s konfigurací rozvrhu synchronizace, je v jeho záhlaví textová informace, která obsahuje konkrétní datumy a časy, kdy k příští synchronizaci dojde. Box není potřeba otevírat.

Obrázek 4.6 Diagram případů užití pro správu logů vytvořený v nástroji Enterprise Architect



4.2.6 Návrh stránky Logy

V této části popíšu návrh stránky, jejímž hlavním úkolem bude správa logů, přičemž logem zde myslíme záznam o provedené synchronizaci časových záznamů či konfiguračních objektů. Návrh opět založím zejména na případech užití realizujících jednotlivé požadavky. Přehled jednotlivých případů užití souvisejících se správou logů je k dispozici na diagramu 4.6.

UC24: Zobrazení všech logů

Tento případ užití uživateli umožní zobrazit si všechny záznamy o provedených synchronizacích. Systém bude záznamy zobrazovat v tabulce na samostatné stránce „Logy“. Zvláštní stránka byla pro záznamy vyčleněna kvůli tomu, že pokud bude mít uživatel více spojení, může jejich počet rychle narůstat a oddělení od samostatných spojení se mi zdá být přehlednější. Pokud ještě nebudou existovat žádné logy, uživatel o tom bude textově informován. Tabulka bude stránkovaná a bude obsahovat celkem sedm sloupců. Záznamy bude možné seřadit dle libovolného sloupce. Těmi budou:

- **ID spojení** ID spojení bude identifikátor, například `#1`, který bude odpovídat boxu, ve kterém se nachází spojení, ke kterému záznam synchronizace patří. Předpokladám, že tento údaj usnadní uživateli vyhledávání daných spojení.
- **Spojení mezi** Údaj se bude v případě služby Redmine skládat z API pointu. V případě služby Toggl Track půjde o zvolený workspace. Údaje budou odděleny pomlčkou a budou

jednoznačně identifikovat záznamy mezi konkrétní dvojicí nástrojů na rozdíl od prvního sloupce, který odpovídá boxu. Pokud by tedy uživatel z nějakého důvodu přesouval jednu synchronizační dvojici z jednoho boxu do jiného, tímto sloupcem může konkrétní spojení vyfiltrovat.

- **Naplánováno na** Údaj vyjadřuje, na kdy byla naplánována synchronizace, které záznam odpovídá. Odpovídá údaji, který je zobrazován i v současném systému.
- **Typ** Údaj vyjadřuje, zda jde o záznam patřící k synchronizaci časových záznamů či konfiguračních objektů. Odpovídá údaji, který je zobrazován i v současném systému.
- **Původ** Údaj vyjadřuje, zda se jedná o záznam z pravidelné automatické synchronizace či synchronizace manuální, kterou uživatel vyvolal žádostí o okamžité provedení synchronizace. Odpovídá údaji, který je zobrazován i v současném systému.
- **Dokončeno** Údaj vyjadřuje, kdy byla dokončena synchronizace, které záznam odpovídá. Odpovídá údaji, který je zobrazován i v současném systému.
- **Status** Údaj vyjadřuje, zda došlo v rámci dané synchronizace k chybě. Odpovídá údaji, který je zobrazován i v současném systému.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Logy“.

UC25: Zobrazení vybraných logů

Tento případ užití umožní uživateli, aby si zobrazil pouze omezené množství logů, které ho zajímají. Funkcionalita může být důležitá zejména při větším počtu záznamů. Tabulka bude umožnovat provádět filtraci přes všechny sloupce. Tato filtrace půjde zrušit pro jednotlivé sloupce a v případě neexistence žádného výsledku nastavené filtrace, půjde zrušit všechny filtry jedním kliknutím.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Logy“.

UC26: Zobrazení detailní informace ke stavu konkrétního logu

Tento případ užití umožní uživateli, aby si zjistil podrobnější informaci o problémech, které nastaly během synchronizace. Pokud během synchronizace nastane problém, ve sloupci se statusem se objeví ikona s komentářem. Pokud uživatel na tuto ikonu najede myší, zobrazí se mu okno s komentářem k chybě, z něhož půjde text vykopírovat.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Logy“.

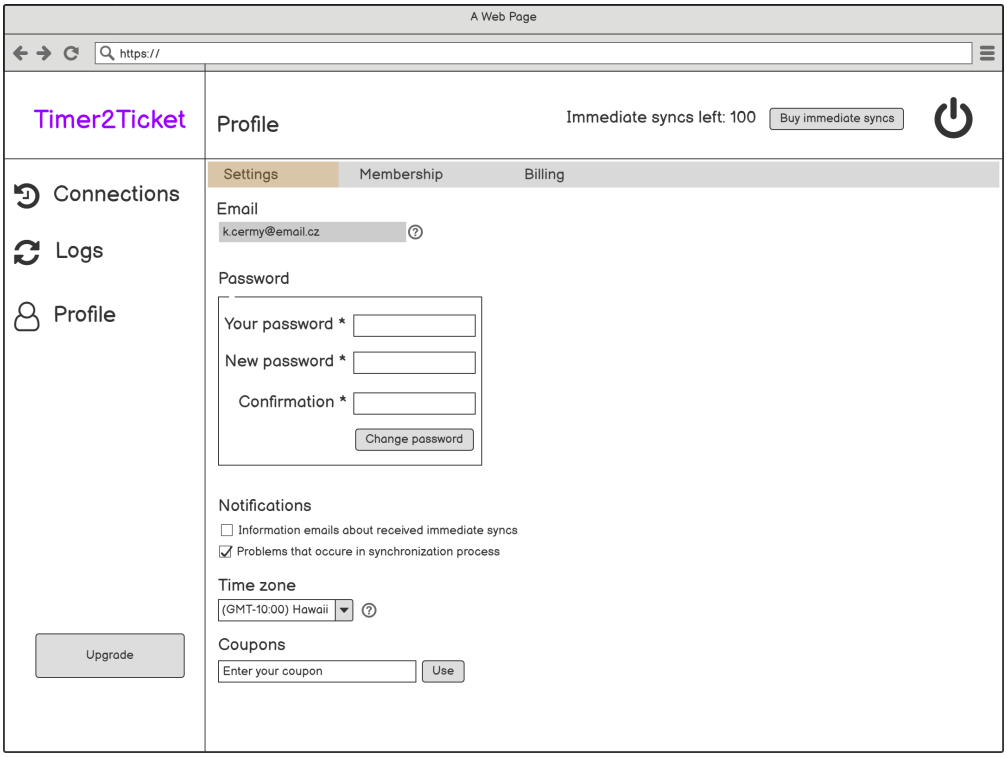
4.2.7 Návrh stránky Profil

V této části popíšu návrh tří stránek, které řadím pod uživatelský profil – „Nastavení“, „Členství“, „Platby“. Navigaci přes tyto tři stránky bude umožňovat horizontální menu, které se bude zobrazovat jako rozšíření záhlaví popsaného v části 4.2.4, pokud uživatel zvolí stránku „Profil“. Při jejím zvolení bude automatiky přemístěn na stránku obsahující nastavení jeho uživatelského profilu.

4.2.7.1 Návrh stránky Nastavení

V této části popíšu návrh stránky, jejímž učelem bude správa nastavení uživatelského účtu. Návrh této stránky, který jsem vytvořil v nástroji Balsamiq, je pro snadnější pochopení na obrázku 4.7. Přehled jednotlivých případů užití souvisejících se správou nastavení uživatelského profilu je k dispozici na diagramu 4.8.

Obrázek 4.7 Návrh stránky Nastavení vytvořený v nástroji Balsamiq



UC27: Změna hesla

Tento případ užití umožní uživateli, aby provedl změnu hesla svého uživatelského účtu prostřednictvím formuláře.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Nastavení“.

Základní scénář:

1. Uživatel vyplní své původní heslo.
2. Systém ověří, že zadané původní heslo odpovídá uživatelově heslu.
3. Systém informuje uživatele o správném zadání jeho hesla.
4. Uživatel vyplní nové heslo.
5. Systém ověří, že zadané nové heslo splňuje požadované parametry na kvalitu.
6. Systém informuje uživatele o splnění požadavků na heslo.

7. Uživatel vyplní potvrzení hesla.
8. Systém ověří, že zadané potvrzení hesla je shodné se zadaným novým heslem.
9. Systém informuje uživatele o úspěšném ověření shody hesel.
10. Uživatel klikne na tlačítko „Změnit heslo“.
11. Systém informuje uživatele o úspěšné změně hesla.

Exception scénář: Tento scénář se spustí ve chvíli, kdy systém ve 2. kroku základního scénáře zjistí, že uživatel zadal své původní heslo chybně.

1. Systém uživatele informuje o chybném zadání původního hesla.
2. Uživatel pokračuje 1. krokem základního scénáře.

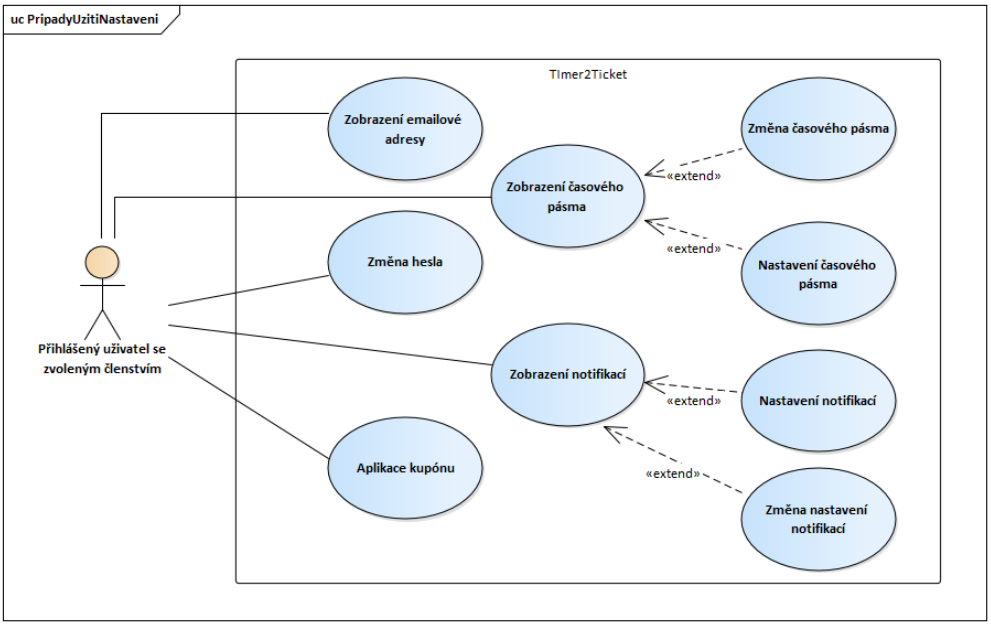
Exception scénář: Tento scénář se spustí ve chvíli, kdy nejsou splněny požadavky na kvalitu hesla v 5. kroku základního scénáře.

1. Systém uživatele informuje o nesplněných požadavcích na kvalitu hesla.
2. Uživatel pokračuje 4. krokem základního scénáře.

Exception scénář: Tento scénář se spustí ve chvíli, kdy je v 8. kroku základního scénáře zjištěno, že zadaná hesla nejsou stejná.

1. Systém uživatele informuje o rozdílnosti hesel.
2. Uživatel pokračuje 7. krokem základního scénáře.

Obrázek 4.8 Diagram případů užití pro správu nastavení vytvořený v nástroji Enterprise Architect



UC28: Zobrazení emailové adresy

Tento případ užití bude realizován textovým polem s danou emailovou adresou. Nachází se v horní části obrazovky. Jeho součástí je také informační prvek, na který pokud uživatel najede myší, získá informaci o tom, že pro změnu emailové adresy může kontaktovat provozovatele.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Nastavení".

UC29: Nastavení notifikací

Tento případ užití umožní uživateli, aby si nastavil notifikace, které mu budou zasílány na emailovou adresu, pod níž má svůj účet zaregistrován. V současnou chvíli se počítá s notifikacemi informujícími o problémech, které se objevily v synchronizačním procesu, a notifikacích, které informují o obdržených okamžitých synchronizacích. Nastavení notifikací bude probíhat zaškrtnutím boxu na stránce „Nastavení". Ve výchozím nastavení budou oba typy notifikací vypnuté, uživatel si je může kdykoliv nastavit. Změna bude probíhat jednoduchým kliknutím na box příslušné notifikace. Případ užití pro změnu i zobrazení zde pro jejich trivialitu neuvádím.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Nastavení".

UC30: Nastavení časového pásma

Tento případ užití umožní uživateli, aby si nastavil jeho časové pásmo, díky čemuž bude mít uživatel zajištěno, že pro něj synchronizace probíhá ve zvolený čas. Dojde tak k zajištění časového posunu oproti provozovateli. Ve výchozím nastavení bez volby časového pásma k žádnému posunu docházet nebude. K nastavení bude docházet na stránce „Nastavení" prostřednictvím rozbalovacího seznamu, v němž si uživatel bude moci své pásmo vybrat. Vedle rozbalovacího seznamu se bude nacházet informační prvek, na který pokud uživatel najede myší, získá informaci o tom, proč je důležité volbu časového pásma provést. Změna bude probíhat obdobným způsobem jako prvotní volba časového pásma. Případ užití pro změnu i zobrazení zde pro jejich trivialitu neuvádím.

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Nastavení".

UC31: Aplikace kupónu

Tento případ užití umožní uživateli, aby využil svůj kupón na získání okamžitých synchronizací. Aplikace kupónu se bude provádět prostřednictvím textového pole na stránce „Nastavení".

Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím se nachází na stránce „Nastavení".

Základní scénář:

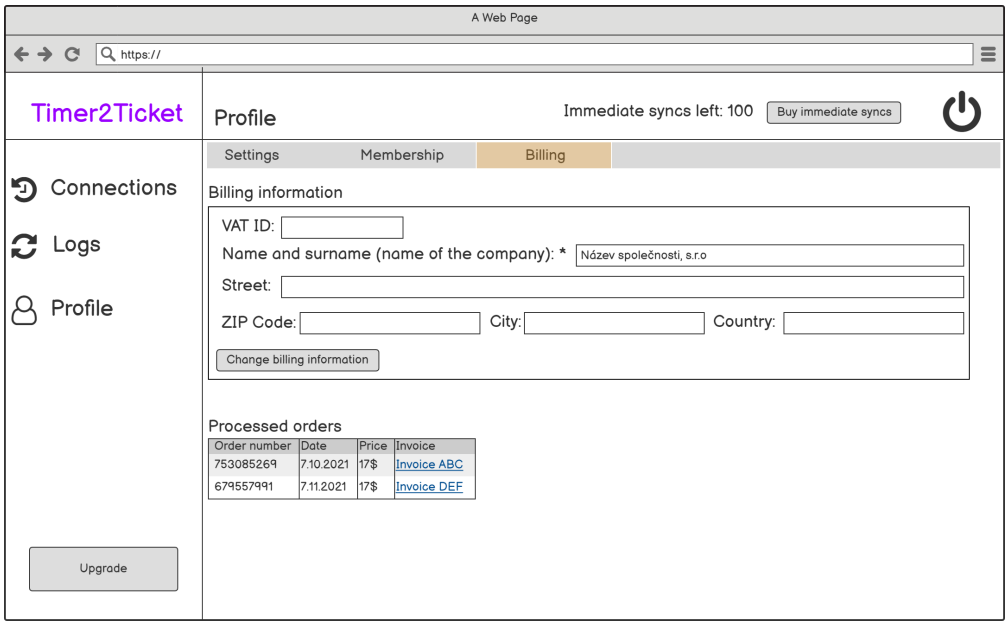
1. Uživatel vyplní kód svého kupónu.

2. Uživatel stiskne tlačítko „Použít“.
3. Systém ověří, že je kupón validní.
4. Systém informuje uživatele o úspěšném využití kupónu a připíše mu slíbený počet okamžitých synchronizací.

Exception scénář: Tento scénář se spustí ve chvíli, kdy systém ve 3. kroku základního scénáře zjistí, že je vložen nevalidní kód kupónu.

1. Systém uživatele informuje o tom, že kupón není validní (chybně zadán).
2. Uživatel pokračuje 1. krokem základního scénáře.

Obrázek 4.9 Návrh stránky Platby vytvořený v nástroji Balsamiq



4.2.7.2 Návrh stránky Platby

V této části popíšu návrh stránky, která se bude starat o správu platebních údajů a přístup uživatele k zaplaceným fakturám. Návrh této stránky, který jsem vytvořil v nástroji Balsamiq, je pro snadnější pochopení na obrázku 4.9. Přehled jednotlivých případů užití souvisejících se správou platebních informací je k dispozici na diagramu 4.10.

UC32: Nastavení platebních údajů

Tento případ užití umožní uživateli, aby v aplikaci uložil své platební údaje – DIČ, jméno a příjmení (jméno společnosti), ulici, PSČ, město a zemi. Díky přednastavení je nebude muset zadávat při každé jednotlivé platbě v aplikaci. K tomuto účelu bude sloužit formulář na stránce „Platby“. Současné nastavení platebních údajů si uživatel zobrazí ve stejném formuláři. Samostatný případ užití pro zobrazení a opakovanou změnu, která je obdobná nastavení, vzhledem k trivialitě neuvádím.

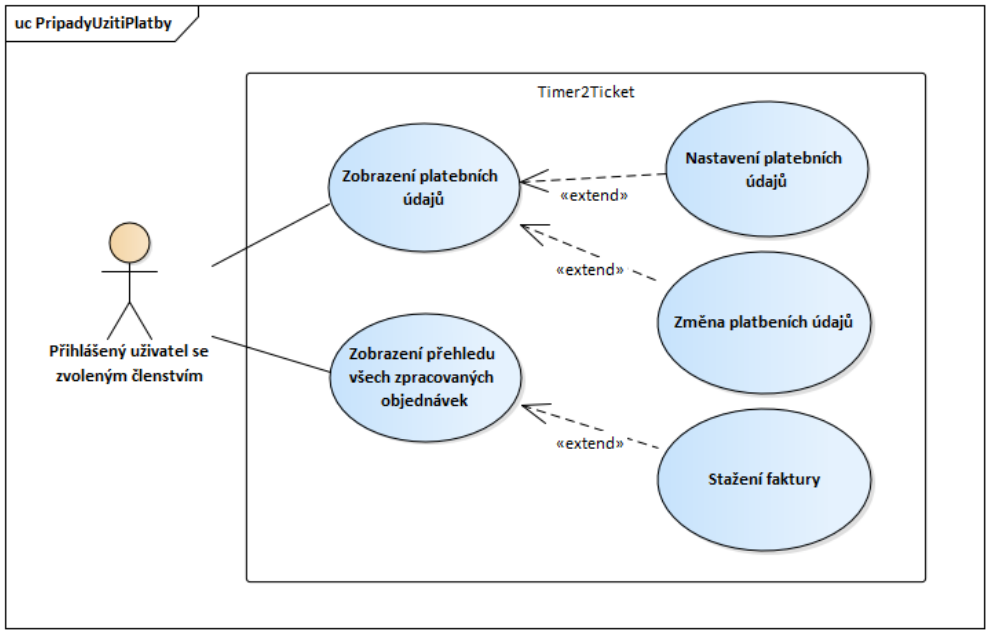
Aktéři: Přihlášený uživatel se zvoleným členstvím

Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím je na stránce „Platby“.

Základní scénář:

1. Uživatel vyplní prvky formuláře, které chce uložit.
2. Uživatel klikne na tlačítko „Změnit platební údaje“.
3. Systém uloží nastavené platební údaje a informuje uživatele o úspěšném uložení formuláře.

Obrázek 4.10 Diagram případů užití pro správu platebních informací uživatele vytvořený v nástroji Enterprise Architect



UC33: Zobrazení přehledu všech zpracovaných objednávek

Tento případ užití umožní uživateli, aby si zobrazil přehled zaplacených objednávek. V případě, že pro daného uživatele ještě neexistují žádné zaplacené objednávky, je o tom textově informován. Pokud již nějaké zaplacené objednávky existují, jsou uživateli zobrazeny v tabulce, která obsahuje čtyři sloupce. První obsahuje číslo objednávky. V druhém je k nalezení datum (zpracování objednávky) a ve třetím zaplacená cena. Tyto údaje jsou zde pro získání základního přehledu. Poslední sloupec pak obsahuje odkaz na samotnou fakturu, odkud jde faktura stáhnout, čímž uživatel získá detailní obsah dané faktury. Vzhledem k jednoduchosti případu užití stažení faktury ho uvádím pouze zde.

Aktéři: Přihlášený uživatel se zvoleným členstvím

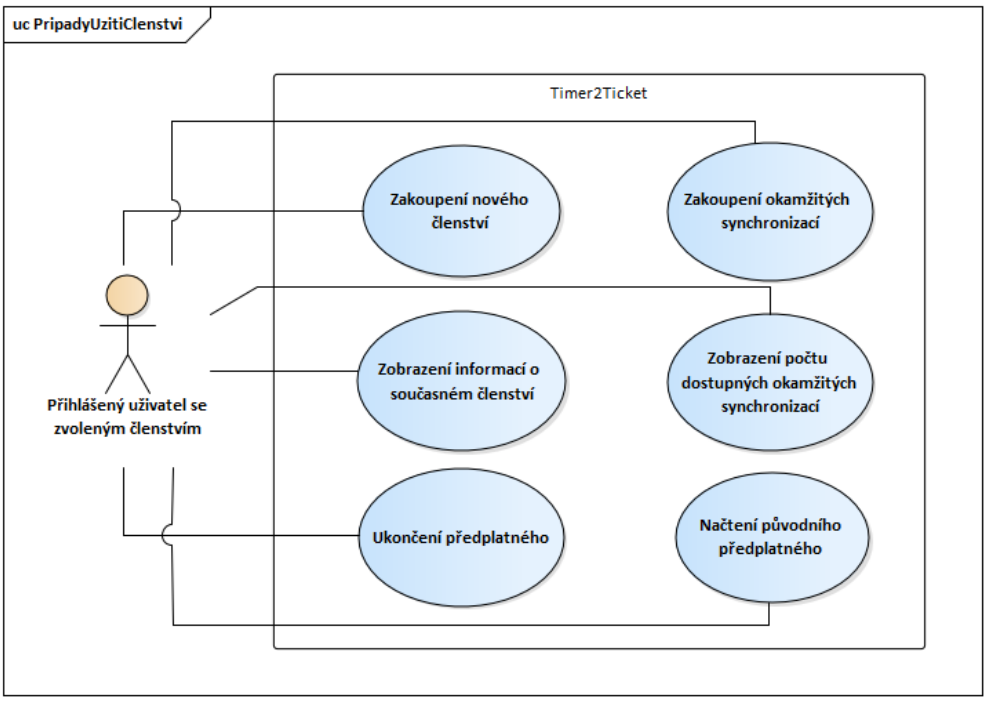
Počáteční podmínky: Přihlášený uživatel se zvoleným členstvím je na stránce „Platby“.

4.2.7.3 Návrh stránky Členství

V této části popíšu návrh stránky, která bude zobrazovat stav uživatelova současného členství, avšak bude mu také umožňovat toto členství změnit a nakoupit okamžité synchronizace. Myslím

si, že pro pochopení návrhu této stránky, bude názornější využít čistě textový popis bez dělení na konkrétní případy užití. Přehled jednotlivých případů užití související se správou členství uživatelů a placených služeb uvádím pro dokreslení souvislostí do diagramu 4.11. Aktérem všech znázorněných případů užití je přihlášený uživatel se zvoleným členstvím, který se nachází na stránce „Členství“. Finální návrh stránky vytvořený v nástroji Balsamiq je pro snadnější pochopení na obrázcích 4.12 a 4.13. Na obrázcích je naprostá většina zde popsaných prvků, nicméně abych pokryl veškeré případy a kombinace, jak mohou být prvky zobrazeny (zejména v části shrnutí údajů o platbě), musel bych zde přiložit opravdu vysoké množství obrázků. Detailní wireframy znázorňující veškeré situace jsou k nalezení na přiloženém médiu.

Obrázek 4.11 Diagram případů užití pro správu členství vytvořený v nástroji Enterprise Architect



Princip změny členství

Poměrně náročným prvek v rámci návrhu bylo vymyslet, jakým způsobem přistupovat ke změnám členství. Nakonec jsem dospěl k následujícímu řešení, které umožňuje v jakoukoliv chvíli uživateli provést upgrade i downgrade jeho členství na jiné. Princip je založen na tom, že pokud uživatel vlastní jakékoliv placené členství (tj. Junior členství, Senior členství či Hobby členství, které však není v základní podobě, ale navýšené o dodatečné aktivní spojení) a chce provést změnu svého členství na jiné placené členství, zaplatí si cenu nového členství na dobu jednoho měsíce, avšak aby mu nepropadl zbývající čas současného členství, bude mu první měsíc nového členství prodloužen. Doba, o kterou bude první měsíc prodloužen, bude stanovena tak, aby cena za ni byla odpovídající ceně, která by se zaplatila za zbývající počet dnů stávajícího členství. Po uplynutí měsíce a nadsazené doby bude uživatel pokračovat s pravidelným měsíčním předplatným. Tento model s kompenzací zbývajícího času jsem se rozhodl zvolit s ohledem na to, že možnost rozšířit základní členství dalšími aktivními spojeními je k dispozici až v rámci uživatelského profilu na stránce „Členství“. Tato možnost není součásti prvotní volby členství po registraci. Možnost rozšířit členství při volbě členství hned po registraci byla detailně rozebírána na schůzce s týmem i se zákazníkem. Hlavním důvodem, proč ji při prvotní volbě neuvést, bylo, aby první vstup do

aplikace byl pro uživatele co nejméně namáhavý a než se s aplikací seznámí, měl co nejméně nutného rozhodování.

Pokud uživatel bude chtít provést změnu libovolného placeného členství na základní Hobby členství (jediné zdarma), dostane na výběr ze dvou možností. První možností je, že ukončí automatické předplácení současného členství, avšak nechá ho doběhnout do data, do kterého je předplaceno. Pokud by uživatel tuto možnost zvolil a následně se v době, kdy dobíhá staré členství, rozhodl pro další změnu na některé placené členství, došlo by automaticky k prodloužení prvního měsíce, jak bylo před chvílí vysvětleno. Druhou možností je nabídka náhrady zbývajících dní stávajícího členství za jistý počet okamžitých synchronizací. Tento počet bude stanoven tak, aby cena za něj (pokud by jej uživatel kupoval samostatně), byla odpovídající ceně, která vyjde jako cena, která by se zaplatila za zbývající počet dnů jeho stávajícího členství. Pokud uživatel zvolí tuto možnost, dojde k ukončení předplatného, okamžité změně členství na základní Hobby plán a připsání slíbeného počtu okamžitých synchronizací.

Zbývající možností je, že uživatel bude vlastnit základní Hobby členství a nakupovat libovolné placené členství. V takovém případě dojde k obyčejnému nákupu bez jakékoliv změny doby prvního měsíce či přípisu okamžitých synchronizací. Plně zdarma je pouze Hobby členství s jedním aktivním spojením, změny mezi členstvími zdarma tedy nejsou možné a tento případ je vyřešen triviálně. Jak je z popisu již zřejmě patrné, návrh počítá s tím, že uživatel bude moci využívat opakovaných plateb, které platební brány nabízejí.

Rozložení stránky

Řádek nad jednotlivými boxy

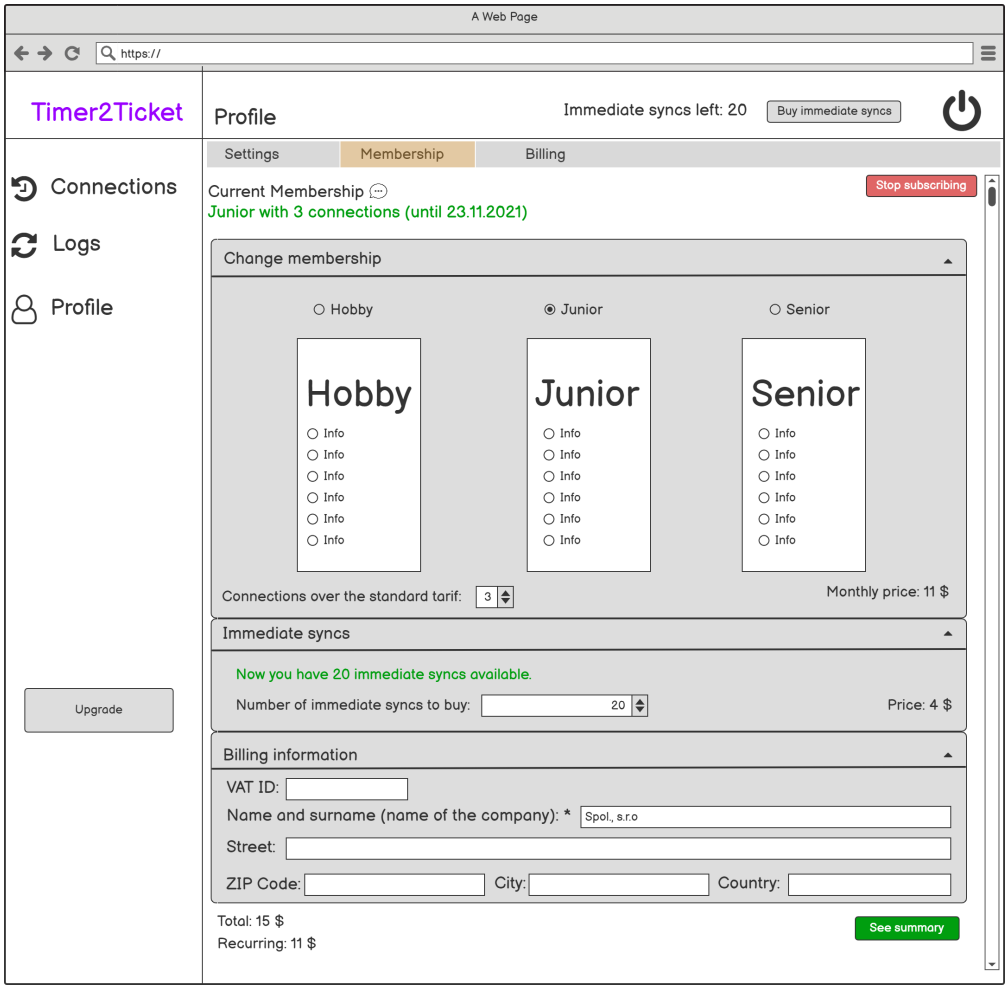
Stránka bude uživateli ve své levé horní části poskytovat informace o jeho současném členství. Jak bylo již zmíněno, uživatel má po registraci do aplikace možnost zvolit si jedno ze tří základních členství – Hobby, Junior či Senior. Tato členství mají předdefinovaný počet aktivních spojení, která v základu umožňují. Aktivní spojení si však lze dále dokoupit a základní členství tímto směrem rozšířit. Tato informace proto bude obsahovat název jednoho ze tří členství, což specifikuje umožněnou dobu pravidelné synchronizace, a informaci o počtu aktivních spojení, která má uživatel k dispozici, což se může lišit na základě toho, zda uživatel využil možnost nákupu aktivních spojení nad rámec základního plánu daného členství. Součástí bude také informace, do kdy má uživatel členství předplacené.

V pravé horní části bude k dispozici červené tlačítko „Zrušit předplácení“, které bude uživateli, který má jakékoliv placené členství, umožňovat ukončit automatické předplácení jeho členství. V takovém případě se uživateli zobrazí dialogové okno, ve kterém bude dotázán, zda chce opravdu členství ukončit a začít využívat základní Hobby členství. Současně s tím mu budou nabídnuty tři možnosti. První z nich je, že dialogové okno uzavře tlačítkem „Zrušit“ a od myšlenky zrušení předplácení upustí. Druhou z nich je, že ukončí automatické předplácení tlačítkem „Dokončit členství“, avšak nechá současné zakoupené členství doběhnout do data, do kterého je předplaceno. Teprve následně přejde do základního Hobby členství. Poslední možností je, že přijme nabízené okamžité synchronizace, jejichž počet bude v tomto dialogové okně uživateli sdělen. Tento počet bude stanoven tak, jak bylo vysvětleno v přechozí části popisující princip změny členství. Tuto možnost uživatel zvolí stiknutím tlačítka „Okamžité synchronizace“. V případě zrušení předplácení bude uživatel notifikován a tlačítko, na které původně klikal se změní na zelené tlačítko „Nahrát mé poslední předplatné“, které uživateli umožní načíst volbu posledního placeného členství (základního členství i aktivních spojení nad jeho rámec). Tato volba se načte do boxu níže na této stránce, který slouží pro změnu členství. Pokud uživatel ještě nikdy nedisponoval členstvím, za které by platil, žádné tlačítko v pravém horním rohu nebude.

Prostor jednotlivých boxů

Popsaný horní prostor této stránky je následován třemi rozbalitelnými boxy. Prvním bude box, který bude sloužit k nákupu (respektive změně) členství. Druhý box bude sloužit k nákupu

Obrázek 4.12 Návrh stránky Členství vytvořený v nástroji Balsamiq – otevřené boxy se zvolenou změnou členství, připraveným nákupem okamžitých synchronizací a vyplněnými povinnými údaji



okamžitých synchronizací. Třetí box bude sloužit k okamžité modifikaci platebních údajů pro platby, aniž by uživatel musel odcházet ze svého rozpracovaného nákupu. Tyto boxy budou při příchodu na stránku defaultně zabalené. Výjimkou budou případy, kdy uživatel využije tlačítka „Upgrade“ na boční liště, tlačítka „Koupit okamžité synchronizace“ na záhlaví či tlačítka „Koupit více“ na stránce „Spojení“. V takovém případě je totiž jasné, za jakým účelem uživatel na stránku přišel a lze mu tak usnadnit jejich rozbalením práci. V případě tlačítek „Upgrade“ a „Koupit více“ bude uživateli defaultně rozbalen první box se změnou členství. V případě tlačítka „Koupit okamžité synchronizace“ se automaticky rozbalí druhý box k tomu určený.

První box bude obsahovat informace o jednotlivých základních členstvích, která jsou uživateli nabízena. Bude si zde moci zvolit, které chce nově zakoupit. Zde již bude uživateli umožněno zakoupit i aktivní spojení nad rámec základního členství. Jakmile uživatel provede volbu základního členství či zvolí nákup aktivních spojení nad standardní rámec, bude mu v pravém dolním rohu tohoto boxu k dispozici informace o částce, kterou bude za zvolené členství (základní i rozšířené o dodatečné aktivní spojení) měsíčně platit.

Druhý box uživateli umožní nakoupit okamžité synchronizace. Uživatel zde bude informován o tom, jaký má v současné době dostupný počet okamžitých synchronizací. Jakmile zvolí nenulový počet okamžitých synchronizací, obdobně jako v případě prvního boxu, bude informován o ceně

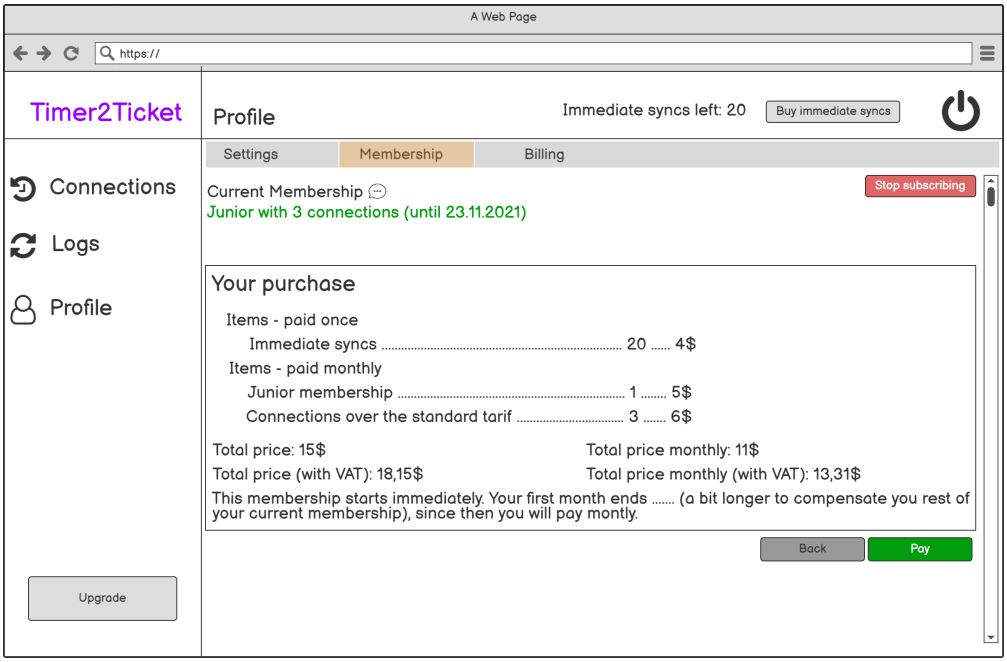
za zvolený počet okamžitých synchronizací. Cenu rozděluji takto po jednotlivých boxech, protože okamžité synchronizace jsou prvkem, který není součástí členství jako takového a nebude součástí opakovaných plateb.

Poslední box obsahuje platební údaje. Tento box obsahuje formulář, jehož pole jsou identická s těmi, které jsou součástí formuláře na stránce „Platby“. Údaje, které si uživatel předvyplnil na stránce „Platby“ se sem automaticky pro účely plateb předvyplní. Pokud však uživatel bude chtít využít jiné, může je zde při nákupu změnit a tím aktualizovat.

Řádek pod jednotlivými boxy

Pod posledním boxem bude v levé části stránky textová informace o celkové ceně za zvolené položky v prvních dvou boxech. Tato cena bude odpovídat té, kterou uživatel bude platit při první platbě a bude v ní tedy zahrnuta i cena jednorázové platby za okamžité synchronizace. Pod ní bude informace o ceně, která bude placena opakovaně za nově zvolené členství. Na pravé straně pod posledním boxem bude tlačítko „Prohlédnout si souhrn“, které uživateli zobrazí přehled o jeho nákupu před tím, než proběhne platba.

Obrázek 4.13 Návrh stránky Členství vytvořený v nástroji Balsamiq – shrnutí informací o nákupu pro zvolenou změnu členství a nákup okamžitých synchronizací z obrázku 4.12



Souhrn informací o nákupu

Zobrazený přehled informací o nákupu se bude lišit v závislosti na nákupu uživatele. V této části proto popíšu jednotlivé situace, které mohou nastat při kliknutí na tlačítko „Prohlédnout si souhrn“. V rámci návrhu jsem identifikoval následujících šest situací:

1. **Zvolení stavu, kdy uživatel nemůže být vpuštěn k přehledu** První možností je nastátí chybového stavu. K tomu dojde ve chvíli, kdy uživatel nevyplnil povinné informace z platebních údajů. Dojde k němu také ve chvíli, kdy uživatel zvolí počet aktivních spojení nad rámec základního členství, avšak nezvolí, které členství chce těmito aktivními spojeními rozšířit a využívat. Druhou možností je nastátí stavu, kdy uživatel neprovádí změny,

tedy nic nezvolil v žádném z prvních dvou boxů (ani změnu členství ani nákup okamžitých synchronizací) či provedl pouze změnu členství, která je však identická se členstvím, které v současné době má. Ve všech těchto případech uživatel bude notifikován o daném nedostatku a nebude puštěn do přehledu. Všechny další situace předpokládají, že tato situace nenastala.

2. **Zvolení pouze základního Hobby členství** Takový případ odpovídá situaci, kdy by uživatel chtěl zrušit předplácení tlačítkem z řádku nad jednotlivými boxy. Dojde proto k zobrazení stejného dialogového okna jako při kliknutí na toto tlačítko. Všechny další situace předpokládají, že tato nenastala.

3. **Zvolení základního Hobby členství a nákup okamžitých synchronizací** Zvolená situace předpokládá, že uživatel má v současné době jiné než základní Hobby členství (v opačném případě je tato situace degradována do té, které je popsána v následujícím bodě). V takovém případě uživateli tlačítko „Prohlédnout si souhrn“ zobrazí souhrn informací k nákupu. Ten bude obsahovat informaci o jednorázově placených položkách (tj. okamžitých synchronizacích) – jejich počet, cenu. Dále bude k dispozici celková cena bez DPH a celková cena s DPH. Vzhledem k tomu, že uživatel ve svém požadavku chtěl také provést změnu samotného členství, bude mu v rámci přehledu umožněno zvolit, zda chce své členství dokončit a pouze ukončit předplácení či chce získat uvedený počet okamžitých synchronizací jako kompenzaci – obdobně jako při zobrazení dialogového okna při samotném zrušení předplácení.

4. **Pouze nákup okamžitých synchronizací** Pokud se uživatel rozhodne, že nakoupí pouze okamžité synchronizace, zobrazí se mu přehled identický s tím z předchozího bodu. Jedinou rozdílností bude absence volby ukončení současného členství, k čemuž zde nedochází.

5. **Pouze změna členství** Tato situace nastává ve chvíli, kdy uživatel mění své členství na jiné. Předpokládá se, že nenastala žádná z výše popsaných situací. V této situaci bude uživateli zobrazen souhrn informací k nákupu. Ten bude obsahovat informaci o měsíčně placených položkách (tj. základním členství a aktivních spojeních rozšiřujících toto základní členství) – jejich počet a cenu. K dispozici bude celková měsíčně placená cena s i bez DPH. Uživatel bude také textově informován o tom, kdy končí jeho první placený měsíc, který může být o něco delší než kalendářní měsíc s ohledem na kompenzaci současného členství. Následující kalendářní měsíce budou placeny opakovanými platbami.

6. **Změna členství i nákup okamžitých synchronizací** Tato situace opět předpokládá, že nenastala žádná z přechozích. Uživatel je přesunut k souhrnu. Ten bude propojením předchozích dvou situací, kdy uživatel nakupoval zvlášť okamžité synchronizace a měnil členství. Přehled bude obsahovat informace o jednorázově placených položkých i položkách placených měsíčně. K dispozici bude také celková měsíčně placená cena (odpovídající ceně v následujících měsících) a celková cena placená při první platbě (odpovídající první platbě, kdy je cena navýšena o jednorázovou platbu okamžitých synchronizací). Nakonec bude uživatel opět informován o tom, kdy končí jeho první placený měsíc.

Pokud je uživatel puštěn k výše popsanému přehledu, zobrazí se tento přehled místo trojice boxů popsaných dříve v této kapitole. Zároveň s tím dojde k nahrazení dříve popsaného řádku pod boxy řádkem, který obsahuje na své pravé straně tlačítko „Zpět“ a „Zaplatit“. Prvním z nich se uživatel vrátí k boxům, ve kterých může změnit svůj nákup a platební údaje. Tyto boxy budou otevřené či zavřené tak, jak je nechal při přesunu ke shrnutí. Druhým dojde k přesměrování na platební bránu, která umožní provedení platby. V případě, že se ke shrnutí dostal uživatel v rámci situace tři (z výše popsaných), před přesunem do platební brány je ještě zkontrolováno, zda uživatel zvolil jednu z nabízených možností, jak dokončit proces ukončení členství. Pokud tak neučiní, po kliknutí na tlačítko „Zaplatit“ zůstane na shrnutí a je notifikován o chybějící volbě.

4.3 Návrh propagačního webu

Jak již při sběru požadavků vyplynulo, propagační web bude klíčovým prvkem pro upoutání pozornosti nových zákazníků. S ohledem na to je také potřeba ho navrhnout. Návrh v této části nebudu zakládat na případech užití realizujících jednotlivé funkční požadavky jako v dřívějších kapitolách, protože by došlo pouze ke zbytečné duplikaci s ohledem na zvolenou granularitu požadavků na propagační web. Vysvětlím zde rozdělení propagačního webu do jednotlivých sekcí dle wireframů, které na základě mnou zadaných úkolů v systému Redmine postupně vytvořil Benedek v nástroji Figma. Právě tuto část jsem Benedekovi přenechal zejména kvůli jeho smyslu pro design, který s upoutáním pozornosti zákazníka pomůže.

Obrázek 4.14 Propagační web – sekce upoutání pozornosti a funkcí aplikace (vytvořeno v nástroji Figma)

4.3.1 Rozdělení webu

Vzhledem k nepříliš vysokému množství informací, které bude nutné na propagační web umístit, jsem se rozhodl, že jednostránkový web bude dostatečný. V záhlaví proto bude fixní řádek, který bude v levé části obsahovat logo aplikace a v pravé části budou postupně odkazy do

jednotlivých sekcí, které uživatele budou pouze posouvat po dané stránce. Vedle nich budou tlačítka umožňující přihlášení a registraci do aplikace, obdobně jako například u [11], [15] či [17]. Tyto tlačítka budou zajišťovat přesměrování do samotné aplikace. V případě tlačítka pro přihlášení bude uživatel přesunut na odpovídající přihlašovací formulář. V případě registračního tlačítka bude přesunut na registrační formulář.

4.3.2 Části webu

Návrh je rozdělen do šesti sekcí, které jsou vizuálně i sémanticky odděleny. Jednotlivé části rozeberu v této podkapitole. První dvě sekce související s upoutáním pozornosti a nabízenými funkcionalitami jsou pro lepší pochopení na obrázku 4.14. Další dvě sekce související s možnými členstvími a informacemi o open-source jsou k dispozici na obrázku 4.15. Poslední dvě sekce věnující se detailnějším informacím o projektu a kontaktnímu formuláři jsou na obrázku 4.16.

Upoutání pozornosti

První sekce cílí na upoutání pozornosti uživatele, který se na stránku propagačního webu poprvé dostane. Bude informovat o hlavním přínosu aplikace – ušetří mu synchronizací čas. Kromě krátkého výstižného popisku zde bude prostor pro video, které shrne základní principy aplikace – co aplikace dělá – a vysvětlí potenciálnímu zákazníkovi, proč by měl Timer2Ticket používat. Součástí bude i tlačítko, které bude uživatele informovat o možnosti využití aplikace zdarma. Jeho klepnutím se uživatel dostane k registraci do aplikace.

Funkce aplikace

Druhá sekce bude obsahovat výčet klíčových prvků aplikace. V bodech poukáže na webovou aplikaci jakožto na open-source řešení synchronizace času a projektů mezi nejrůznějšími nástroji. Blíže přiblíží možné využití tohoto synchronizačního nástroje a představí prostředí aplikace pomocí reálné ukázky z ní, obdobně jako je tomu například u [11].

Členství

Další sekce se bude věnovat informacím o jednotlivých členstvích. Potenciálnímu zákazníkovi zde budou představeny všechny tři základní plány s informacemi o možných aktivních spojeních, době synchronizace a ceně. Ceny budou sloužit jako tlačítka, na která pokud uživatel klikne, bude přesunut do samotné aplikace na registrační formulář. Pokud následně provede registraci, bude mít při nákupu členství v aplikaci přednastaveno dané předvolené členství z propagačního webu. Jak již bylo zmíněno v návrhu samotné aplikace, při tomto prvním nákupu uživatel nebude mít možnost zakoupit aktivní spojení nad rámec základních členství, aby pro něj byl prvotní kontakt s aplikací jednodušší a neodradil ho.

Kromě jednotlivých členských plánů zde bude vyhrazen také prostor, ve kterém bude uživatel informován o možnosti kontaktovat provozovatele v případě zájmu o získání množstevní slevy v případě větších týmů a organizací, obdobně jako je tomu například u [10] nebo [12]. Tento fakt je podstatný zejména s ohledem na zjištění, že nástroje, z jejichž strany se očekává největší přísun zákazníků, jsou nejpoužívanější mezi společnostmi majícími 10–50 pracovníku (jak bylo detailněji rozebráno v kapitole 2.5.3).

Při diskuzích s týmem jsme dospěli k závěru, že by mohlo být dobrým marketingovým tahem nabídnout zde také cenové zvýhodnění pro studenty – například jim zpřístupnit Junior plán zdarma. Jednak si tím lze získat zákazníky do budoucna a zároveň se jedná o reklamu prakticky zadarmo.

Kromě zmíněných informací ke členstvím bude součástí této sekce i tlačítko, které bude umožňovat přesměrování k přihlášení do samotné aplikace. Nad jeho umístěním do této sekce panovaly

v týmu diskuze, nicméně na závěr jsme se ho sem rozhodli umístit. Jeho užití se předpokládá například ve chvíli, kdy někdo, kdo aplikaci již používá, ji bude představovat novému potenciálnímu uživateli, například svému známému. Shodli jsme se, že by mohl být poměrně reálný scénář, kdy takový uživatel svému známému představí možnosti členství, pod kterými může aplikaci využívat a následně mu bude chtít ukázat podrobněji fungovaní aplikace.

Obrázek 4.15 Propagační web – sekce členství a open-source (vytvořeno v nástroji Figma)

Open-source

Čtvrtá sekce bude uživatele informovat o možnosti využívat projekt jakožto open-source. Vysvětlí návštěvníkovi webu, že má možnost aplikaci dále vyvíjet a upravovat, pokud ji bude sdílet s ostatními. Zároveň si ji může bezplatně sám nasadit a v takové podobě následně využívat bez omezení na jednotlivé prvky v daných členstvích (tj. bez omezení na okamžité synchronizace, na počet aktivních spojení i na dobu synchronizace). Součástí této části bude také odkaz na projekt s jeho dokumentací.

Předpokládá se, že možnost nasadit si aplikaci samostatně by mohly chtít využít velké společnosti, čímž se vyhnou veškerým poplatkům. Tato možnost pro ně může být lákavá, na druhou stranu je pak může tížit pocit, že si tvůrci tohoto programu zaslouží odměnu za prospěšný projekt, proto součástí této sekce bude i tlačítko, které jim umožní společnosti Jagu s. r. o. přispět na podporu dalšího rozvoje projektu. Toto tlačítko uživateli umožní provést darování peněz, což může být učiněno například prostřednictvím PayPalu, tak jak je tomu u uživatelům blízkého

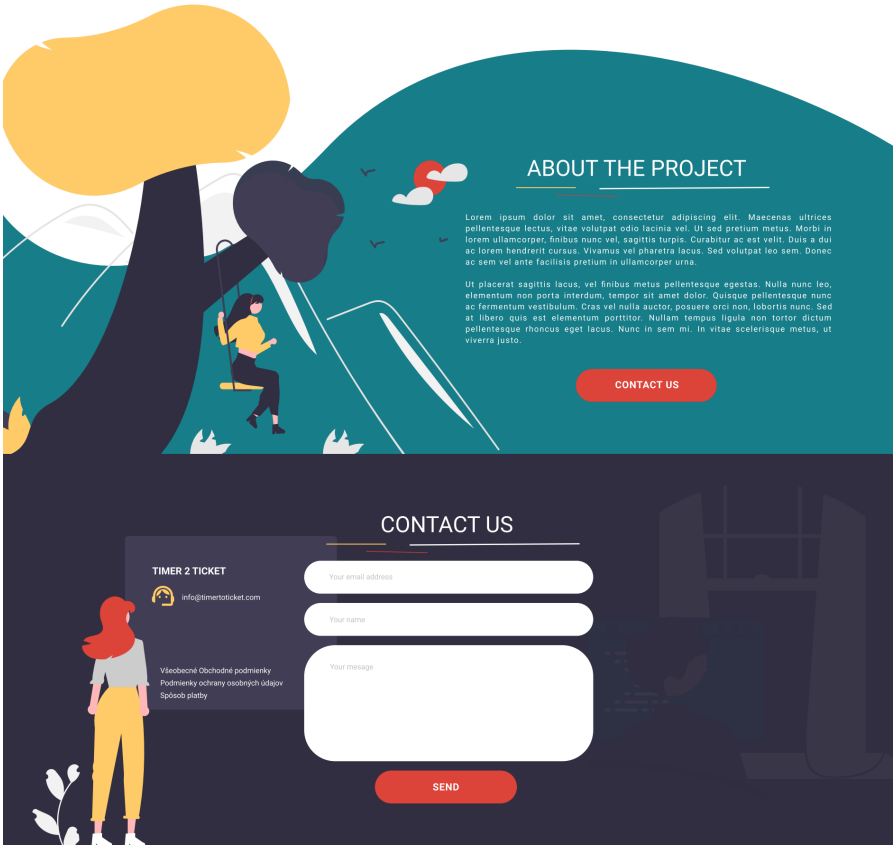
open-source projektu Redmine [6]. Za každou donaci bude osoba, která se rozhodne projekt podpořit, vyzvána k zadání emailu, na který jí bude v případě jeho vyplnění zaslán kupón s okamžitými synchronizacemi. Ten půjde použít následně v rámci samotné aplikace provozované společností Jagu s. r. o.

O projektu

Předposlední sekce bude věnována detailnějším informacím o projektu, jeho původu a týmu, který se zasadil o jeho vývoj. V této části bude také prostor pro případné plány do budoucna. Součástí bude i tlačítko, které uživatele posune ke kontaktnímu formuláři v další sekci.

Nevyhradil jsem na stránce speciální prostor pro FAQ, protože mě napadlo, že by mohlo být užitečné k často kladeným otázkám vytvořit krátké videotutoriály. Informace s odkazem na ně by mohla být součástí této sekce.

Obrázek 4.16 Propagační web – sekce o projektu a kontaktu (vytvořeno v nástroji Figma)



Kontakt

Poslední sekce bude věnována pro spojení se společností Jagu s. r. o., která vývoj projektu zašťituje. Pro účel kontaktu zde bude uveden email. Zároveň bude součástí také kontaktní formulář, do kterého budou návštevníci moci rovnou vyplnit svůj email, jméno a zprávu, se kterou chtějí kontaktovat společnost Jagu s. r. o. Součástí poslední části budou také odkazy na dokumenty související s obchodními podmínkami a podmínkami ochrany osobních údajů.

Kapitola 5
Implementace

V této kapitole nejprve popíšu, jak jsem využil Vue CLI pro vytvoření a správu projektu. Následně rozeberu jednotlivé oblasti implementace, s jejichž vývojem mi pomohly pluginy Vue I18n, Vue Router, Vuex a Vuetify. Poté vysvětlím svoji volbu nástroje pro monitoring chyb a jeho využití a následně rozeberu využití autentikační a autorizační služby Auth0. Součástí této kapitoly nebude popis samotné implementace, která vznikla jakožto realizace funkčních požadavků, protože by přesně odpovídal popsanému návrhu. Součástí nebude ani popis struktury implementovaného prototypu, protože ta je k dispozici v rámci popisu obsahu přiloženého média. Na závěr uvedu konkrétní využití session storage a event busu pro tento projekt.

5.1 Vue CLI

5.1.1 Vytvoření projektu

Pro vytvoření projektu jsem využil Vue CLI neboli Vue Command Line Interface, což je systém, který mi usnadnil veškerou počáteční konfiguraci projektu. Vue CLI mi pomohlo pro aplikaci nakonfigurovat tzv. Webpack, který pomáhá správně propojit veškeré závilosti a již během vývoje je optimalizuje, což zajistí rychlejší načítání [75]. Instalaci jsem provedl na základě návodu [76] pomocí npm, což je správce balíčků. Následně bylo potřeba inicializovat projekt.

Inicializaci lze provést dvěma způsoby. Buď lze zvolit základní variantu přes příkazovou řádku, jak již název tohoto systému napovídá, nebo lze použít Vue UI, tedy grafické uživatelské rozhraní. Vzhledem k jednoduchosti používání jsem zvolil možnost grafického rozhraní. Po zadání příkazu `vue ui` se spustí vlastní server na adrese `localhost:8000` a na něm grafické rozhraní pro správu projektů, které se pak uživateli automaticky otevře v prohlížeči.

Projekty je zde možné založit, ale také importovat, pokud již byly vytvořeny. V mém případě bylo potřeba vytvořit nový projekt, zvolil jsem proto tuto možnost. Dále je potřeba zvolit jméno projektu, zda chceme automaticky vytvořit Git repozitář a zda chceme využít yarn či npm správce balíčků. Já jsem již repozitář vytvořený na GitHubu měl, tuto možnost jsem proto nevyužil. Repozitář se nachází na adrese `https://github.com/Timer2Ticket/timer2ticket-client-vue` a ve větvi `BP_cermak` je i výsledný naimplementovaný prototyp. Jako správce balíčků jsem zvolil yarn, neboť jsem s ním měl již dobrou zkušenost z předmětů SP1 a SP2 vyučovaných na FIT ČVUT. Zároveň je zde uživateli umožněno zvolit pluginy, které chce použít. Má na výběr například z Vuex či Vue Routeru, které jsem oba využil a budu se jim více věnovat v sekcích 5.2.3 a 5.2.2. Pro přehlednost jsem zvolil možnost konfiguraci sjednotit do souboru `package.json`. Druhou možností je konfiguraci rozdělit do jednotlivých souborů. Na závěr jsem zde využil i možnosti nastavit pro projekt ESLint, který mi následně ve vývoji pomáhal dodržovat konzistentní styl psaní kódu, což kromě toho, že zlepšuje čitelnost, může i pomoci vyhnout se zbytečným chybám.

5.1.2 Správa projektu

Jakmile se v rámci Vue UI provede založení projektu, uživatel získá přístup do správy tohoto projektu v již zmíněné grafické podobě. Ta se skládá z několika záložek, jejichž počet se může měnit s ohledem na přidané pluginy.

První záložkou, která pro mě byla velice užitečná, je seznam nainstalovaných pluginů. Uživatel mezi nimi může provádět filtraci, zjistit nainstalovanou verzi daných pluginů a dokonce je zde informace o tom, zda existuje novější verze. Pokud ano, může zde do projektu stáhnout aktualizaci. Zároveň má dostupný odkaz na bližší informace k nainstalovanému pluginu. Součástí je i tlačítko umožňující okamžitě vyhledat nový plugin a nainstalovat ho do projektu.

Dále je k dispozici záložka s detailním přehledem projektových závislostí. Tato záložka poskytuje obdobné možnosti jako předchozí pro pluginy. Oproti přechozí stránce je však u každé závislosti k dispozici také přesná informace o tom, na jakou nejaktuálnější verzi je možné aktualizaci provést. Pokud aktualizace na novější možná není, není to zde ani umožněno. Závislosti zde lze i odstranit.

Následuje záložka umožňující detailnější konfiguraci projektu. Její podzáložky se opět odvíjí podle toho, jaké pluginy, které by se daly konfigurovat, jsou nainstalovány. Pro prototyp jsem zde žádné nastavení podrobněji neměnil.

Předposlední záložka pro mě byla pravděpodobně nejužitečnější, neboť umožňovala spouštění jednotlivých úloh nad kódem. Kromě spouštění úloh jako je kontrola chybějících klíčů pro překlady, jejichž množství se odvíjí od toho, které pluginy si uživatel přidá, je zde zejména možnost provádět build pro produkci a spouštět lokální vývojový server na adrese `localhost:8080`. Vývojový server pak disponuje tzv. Hot Module Replacement, což umožňuje, aby se veškeré změny, které v kódu provedeme, okamžitě objevily v aplikaci v prohlížeči [75]. V případě spuštění dané úlohy je uživatel informován o stavu provedené úlohy a případných chybách a varováních. Kromě toho získá i přehled velikostí jednotlivých závislostí, může snadno aplikaci otevřít či ukončit běh vývojového serveru. Všechny tyto informace nalezne na grafickém přehledu, ale v případě potřeby je mu k dispozici i výstup, který by dostal po proběhnutí úlohy při použití příkazové řádky.

Poslední stránkou, která je v mém případě přítomna, je stránka, která umožňuje vkládat a měnit překlady. Ta se zobrazila po přidání pluginu Vue I18n, kterému se budu později více věnovat v sekci 5.2.1. Osobně jsem ji nikdy neměl potřebu využít, protože jsem preferoval jejich přidávání přímo v kódu. Dokážu si však představit situaci, kdy by tato záložka byla velice užitečná. Věřím, že toto grafické rozhraní by byl schopen použít i člověk, který není programátor a dostal by za úkol přidat do aplikace potřebné překlady do nějakého dalšího cizího jazyka.

5.2 Využité pluginy

V této části nejprve detailněji popíšu, jak jsem provedl zajištění překladů pro aplikaci. Následně se budu věnovat zajištění směrování v aplikaci prostřednictvím Vue Routeru. Poté popíšu práci s daty pomocí Vuex a na závěr se budu věnovat využité grafické knihovně Vuetify, přičemž se zaměřím zejména na její pomoc při zajištění responzivního designu a notifikování uživatelů.

5.2.1 Překlady

Jak vyplynulo již při analýze zákazníka, nebude možné cílit pouze na český trh. Nutností pro tuto aplikaci proto bylo zajistit možnost přidat překlad do neomezeného počtu jazyků. K tomuto účelu jsem využil plugin Vue I18n, který mi tento fakt umožnil. Jeho přidání bylo docíleno prostřednictvím Vue CLI.

Pro použití pluginu se vytvoří složka `locales`, do které stačí pro každou jazykovou mutaci vytvořit jediný soubor (například `en.json` či `cs.json`), v rámci něhož lze překlady nadefinovat v JSON formátu, jak je znázorněno ve výpisu kódu 5.1. Právě do zmíněných souborů

```
{
    "paths": {
        "connections": "Connections",
        "logs": "Logs",
        "profile": "Profile",
        "settings": "Settings",
        "membership": "Membership",
        "billing": "Billing",
        "withoutMembership": "Without membership"
    },
    "logOut": "Log out",
    "...": "..."
}
```

Výpis kódu 5.1 Část připravených překladů v anglickém jazyce

```
export default new VueI18n({
    locale: navigator.language,
    fallbackLocale: process.env.VUE_APP_I18N_FALLBACK_LOCALE || "en",
    messages: loadLocaleMessages(),
});
```

Výpis kódu 5.2 Zajištění jazyku dle nastavení prohlížeče

jsem tedy při implementaci umístil překlady do českého a anglického jazyka. Pro přístup k nim pak stačí využít `$t("cestaKPrekladu")`, kde jakožto cesta k překladu může sloužit například `paths.connections`, který směřuje na text `Connections` z ukázky 5.1.

Pro pohodlí uživatele jsem se rozhodl použít volbu jazyka na základě nastavení prohlížeče uživatele. Toho jsem docílil speciálním nastavením volitelných parametrů do vytvářející se instance Vue I18n, jak je znázorněno na výpisu kódu 5.2. Na druhém řádku pomocí `navigator.language` zjistím jazyk, který má uživatel v současnosti nastavený v prohlížeči. Pokud Vue I18n nalezne překlady pro takový jazyk, uživateli se zobrazí stránka v jím nastaveném jazyce. V opačném případě se využije defaultní volba jazyka z třetího řádku. V takovém nastavení, které jsem zde učinil, překlad vždy skončí v anglickém jazyce, neboť `VUE_APP_I18N_FALLBACK_LOCALE` je defaultně také nastaveno na anglický jazyk [77]. Funkce `loadLocaleMessages()` nakonec jen překlady načítá.

5.2.2 Směrování

Jak bylo již z návrhu zřejmé, celá aplikace bude mít několik stránek, mezi kterými bude nutné uživatele přesměřovávat. Na vyřešení této problematiky jsem se rozhodl využít Vue Router.

Prvním nápadem na řešení by mohlo být využití klasického odkazu, například pomocí `<a>` tagu. Nicméně v takovém případě dochází k znovunačtení celé stránky, což není žádoucí a může to také značně prodloužit načítání. Tento problém je Vue Routerem řešen. Vue Router proto načítá pouze nový aktivní obsah stránky a nedochází k jejímu kompletnímu znovunačtení. Jak uvádí sami experti z Vue School, Vue Router (konkrétně `<router-link>` tag) je vhodné používat při směrování v rámci samotné aplikace, zatímco `<a>` tag je vhodné využívat pro externí linky mimo aplikaci. [78]

Abychom mohli pomocí `<router-link>` tagu směrovat, musíme mít směrování zadefinované,

tedy nastavené jednotlivé routy, jako například na ukázce 5.3. Každá z nich pak definuje komponentu, jejíž obsah se vyrenderuje uživateli, pokud využije danou `path`. Pokud tedy například bude chtít zobrazit `localhost:8080/profile`, vyrenderuje se mu obsah komponenty `Profile` ze složky `views`, která v mém případě obsahuje komponenty, které jsou vstupními body na jednotlivé stránky. Rozdělení na postupné načítání jednotlivých komponent zajišťuje tzv. lazy loading, který je umožněn díky použití arrow funkcí při importu komponent. Jednotlivě načítané části si lze pomocí komentáře v importech pojmenovat. Tyto komentáře, někdy nazývané jako magické komentáře, mají přesnou strukturu a změnit lze pouze název v uvozovkách. [78]

Užitečným prvkem, který jsem se zde také rozhodl využít, je pojmenování jednotlivých route. Pokud tedy někde v kódu chci zajistit přesměrování na některou ze zadefinovaných stránek, místo zapsání konkrétní `path` použiji její `name`. Pro načtení stránky nastavení profilu bych tedy dle ukázky 5.3 k `<router-link>` tagu zapsal `:to="{ name: 'Profile-Settings' }"`, kde `to` je tzv. prop, kterým kromě `<router-link>` tagu disponuje i řada komponent z Vuetify, což směrování výrazně usnadňuje. Výhodou tohoto způsobu je fakt, že případná změna `path` nijak neovlivní fungování aplikace. [78]

```
1       const routes = [
2           // ...
3           {
4               path: "/profile",
5               name: "Profile",
6               component: () =>
7                   import(/* webpackChunkName: "Profile" */
8                   "@/views/Profile"),
9               children: [
10                  {
11                      path: "/profile/settings",
12                      name: "Profile-Settings",
13                      component: () =>
14                          import(
15                          /* webpackChunkName: "ProfileSettings" */
16                          "@/views/ProfileSettings"),
17                  },
18                  // ...
19              ],
20          },
21          {
22              path: "*",
23              name: "Fallback",
24              component: () =>
25                  import(/* webpackChunkName: "Fallback" */
26                  "../views/Connections.vue"),
27          },
28      ]
```

Výpis kódu 5.3 Definování možných směrování mezi stránkami v aplikaci

Při definování jednotlivých route je podstatné dbát na pořadí, protože při směrování se komponenta, která se má vyrenderovat, vyhledává postupně dle první shody. Pokud bychom tedy `path` na řádku 22 ve výpisu kódu 5.3 uvedli jinde než na konci definic, došlo by s ní vždy ke shodě a routy zadefinované v pořadí pod ní by se nikdy uživateli nemohly načíst. Fallback je tedy nutné vždy uvádět poslední.

Vue Router poskytuje i řadu dalších poměrně pokročilých možností jako jsou například zanořené routy, které jsou vidět na ukázce 5.3 na řádku 9, či ochrana navigace, jak je demonstrováno na ukázce 5.4. Do routeru vytvořeného na první řádce je možné následně přidat navigátora (neboli strážce), který bude každý provoz mezi jednotlivými stránkami podrobněji usměrňovat, což se zajistí na řádce 18. Konkrétně v tomto případě vytvořený `navigationGuard` zjišťuje, zda uživatel již disponuje členstvím. Pokud ano a snaží se dostat do oblasti pro uživatele bez členství, je přesměrován na stránku se spojeními. Pokud naopak nemá zvoleno žádné členství a snaží se dostat do prostoru pro uživatele, kteří ho již mají zvolené, není mu to umožněno a je nucen zůstat ve svém omezeném prostoru pro uživatele bez členství. Pokud nenastane ani jedna z těchto situací, je uživatel strážcem vpuštěn na požadovanou stránku. Třetí řádek nemá s ochranou navigace nic společného. Pouze zajišťuje, že z adresy v prohlížeči, která se uživateli zobrazuje, zmizí `#`, který tam bez tohoto nastavení je. Kromě zmíněného pro mě byla užitečným i možnost posílání parametrů prostřednictvím Vue Routeru.

```
1       const router = new VueRouter({
2           routes,
3           mode: "history",
4       });
5
6       const navigationGuard = (to, from, next) => {
7           if (store.state.currentPass === "" &&
8               to.name !== "Without-Membership") {
9               return next({ name: "Without-Membership" });
10          }
11          if (store.state.currentPass !== "" &&
12              to.name === "Without-Membership") {
13              return next({ name: "Connections" });
14          }
15          return next();
16      };
17
18      router.beforeEach(navigationGuard);
```

Výpis kódu 5.4 Ochrana navigace a nastavení Vue Routeru

5.2.3 Práce s daty

Aplikace ve Vue.js zpravidla obsahuje velkou šíři komponent. Každá z nich má svá data, se kterými pracuje, mění je a případně zobrazuje. Občas je však potřeba, aby s některými daty pracovalo více komponent. Pokud tedy potřebujeme sdílet jistý stav mezi větším množstvím komponent, v případě menších projektů je možné využít Vue.js možností a přímo si data mezi sousedními (tj. nadřazenými a podřazenými) komponentami posílat. V případě rozsáhlejších projektů, za což tento považuji, je však vhodné zabývat se správou sdíleného stavu komponent zvlášť, už jen pro zachování jisté přehlednosti kódu. Řešení tohoto problému nabízí právě Vuex plugin. Jeho výhodou je také fakt, že práce s Vuex je poměrně snadno debuggovatelná, protože sdílený stav a jeho změny lze, stejně jako stav jednolivých komponent či směrování, sledovat přímo ve Vue Devtools, které se dají stáhnout jako rozšíření do prohlížeče. [79]

Poměrně jasnou potřebu takovéhoto sdíleného prostoru jsem zpozoroval již například při počáteční práci s jednotlivými členstvími, neboť stav členství, který měnily komponenty starající se o správu členství, byl následně potřebný v oblasti správy jednotlivých spojení.

```
1       export default new Vuex.Store({
2           state: {
3               // ...
4               currentPass: "",
5               currentConnectionsOver: 0,
6               currentMembershipFinishes: "2022-5-27",
7               // ...
8           },
9           getters: {
10              // ...
11              finishDate(state) {
12                  const changedVisualOfFinishDate =
13                      state.currentMembershipFinishes.split("-");
14                  const finish = new Date(
15                      Number(changedVisualOfFinishDate[0]),
16                      Number(changedVisualOfFinishDate[1]) - 1,
17                      Number(changedVisualOfFinishDate[2])
18                  );
19                  return (
20                      "" + finish.getDate() + ". " +
21                      (finish.getMonth() + 1) + ". " +
22                      finish.getFullYear() + ""
23                  );
24              },
25              // ...
26          },
27          mutations: {
28              // ...
29              changeCurrentPass(state, newPass) {
30                  state.currentPass = newPass;
31              },
32              // ...
33          },
34          actions: {
35              // ...
36              changeCurrentPass(context, newPass) {
37                  context.commit("changeCurrentPass", newPass);
38              },
39              // ...
40          },
41          modules: {},
42      });
```

Výpis kódu 5.5 Možnosti poskytnuté Vuex pluginem

Použití Vuexu [80] je vcelku přímočaré a zakládá se na čtyřech hlavních konceptech, tzv. `state`, `getters`, `mutations` a `actions`, které jsou na ukázce 5.5. Část `state` začínající na řádku 2 reprezentuje onen sdílený stav, ke kterému lze přistupovat z libovolné Vue.js komponenty a který odpovídá klasickým datům, která jsou v jednotlivých komponentách. Součástí ukázky je právě sdílený stav pro práci s uživatelovým členstvím.

Druhým konceptem jsou `getters`. Ty reprezentují sdílený stav, který vznikne odvozením ze

základního sdíleného stavu ve `state` části, případně i z jiného odvozeného sdíleného stavu. Konkrétním případem může být `finishDate` (začínající na řádku 11), který využívám v řadě komponent při zobrazování datumu konce členství a umožňuje mi jednotně změnit vizuální podobu koncového data. Tento koncept odpovídá tzv. `computed` vlastnostem, které jsou v jednotlivých Vue.js komponentách.

Třetím podstatným prvek jsou tzv. `mutations` neboli mutace, které slouží k provádění změn nad sdíleným stavem. Mutace přijímá vždy jako první argument `state` a nad ním následně provádí změnu. V případě ukázky dochází k přiřazení nového členství, na což mohou pak komponenty využívající tento sdílený stav rovnou zareagovat a využít ho. Právě při mutaci jsou o změně informovány dříve zmíněné Vue DevTools.

Na rozdíl od mutací, které slouží pouze pro přímou změnu sdíleného stavu ve `state` části a musí být synchronní, `actions` jsou tím, co se používá v jednotlivých komponentách pro zajištění změny stavu. Kromě provedení následných mutací, jak je ukázáno na řádku 37, jsou používány k provádění asynchronních operací jako je komunikace přes API s backendem. Vzhledem k tomu, že jsem prototyp na backend v rámci této práce nenapojoval, `action` i `mutation` z ukázky 5.5 vypadá velmi podobně, nicméně toto rozdělení bude užitečné právě při budoucím napojení na backend.

Poslední velice užitečnou možností, kterou Vuex vedle řady dalších pokročilých prvků nabízí, je rozdělení sdíleného stavu do více modulů, což pomůže v přehlednosti. Této možnosti jsem s ohledem na šíři sdíleného stavu při realizaci prototypu nevyužil. Dokážu si však představit její využití při rozšířeních v budoucnu. Použití jednotlivých částí Vuexu v konkrétních Vue.js komponentách zde neuvádím, protože je to k nalezení například v dokumentaci [80] a nemělo by to přidanou hodnotu (ve srovnání s popsanými hlavními koncepty na konkrétních případech).

5.2.4 Grafická knihovna

Jak již jsem zmínil v sekci 3.2.4 o volbě frontendového frameworku, rozhodl jsem se využít grafickou knihovnu Vuetify. Jedná se o knihovnu, která byla poprvé vydána v roce 2014 a je vyvíjená dle tzv. Material Designu, který se při tvorbě svých komponent inspiruje skutečným světem a snaží se tak dosáhnout co nejreálnějšího uživatelského prožitku. Material Design však kromě toho usiluje také o to, aby komponenty byly co nejvíce customizovatelné – barvy, typografické styly apod. Nespornou výhodou Vuetify je jistě také její aktivní komunita, zejména na Discordu, kde lze získat základní podporu při řešení problémů. [46, 81]

Knihovna Vuetify mi poskytla desítky opravdu designových komponent, které umožňují vytvořit uživatelské rozhraní, které podpoří uživatelký zážitek z aplikace. Kdybych měl detailně popsat využití veškerých jejích komponent, které jsem použil, pravděpodobně by to vydalo na desítky stránek a zároveň by to nemělo příliš přidanou hodnotu, protože dokumentace této knihovny je opravdu kvalitní a obsahuje řadu příkladů. Rozhodl jsem se zde tedy zaměřit zejména na možnosti, které mi poskytla při zajišťování responzivního designu a notifikování uživatele, což byly jedny z požadavků kladených na aplikaci. Kromě toho mi však zjednodušila i práci s formuláři, jejich vyhodnocování, tabulkami, jejich stránkováním, řazením a řadou dalších prvků.

5.2.4.1 Zajištění responzivity

Jak je uvedeno v samotné dokumentaci [46], Vuetify vytváří jednotlivé komponenty tak, aby bylo snadné je udělat responzivními. Se zajištěním responzivního designu mi výrazně pomohly zejména tzv. breakpointy pro jednotlivé obrazovky. Vuetify využívá Material Design breakpointů, které rozdělují obrazovky do celkem pěti skupin dle jejich velikosti. Při zobrazování jednotlivých komponent si lze následně zjišťovat, jaká je současná velikost obrazovky a dle toho ji uzpůsobit. Toho jsem využíval zejména pro změnu velikosti některých prvků, například tlačítek či textů, a pro změny v uspořádání stránky, tak aby rozložení lépe pasovalo na různou velikost obrazovek. Největší změny jsem pak samozřejmě musel provádět pro malé obrazovky, kdy bylo nutné obsah

organizovat převážně vertikálně. Kromě úprav samotné komponenty je možné ji pro dané velikosti obrazovek i plně odstranit.

Na výpisu kódu 5.6 demontruji zmíněné uzpůsobení komponenty na základě velikosti obrazovky dle jednotlivých skupin. Na druhém řádku dochází ke zjištění, zda se jedná o nejmenší ze skupin obrazovek, a pokud tomu tak není, nastavuji odlišné zobrazení – zde konkrétně negativní pravý margin, který se pro nejmenší typ obrazovky nehodil.

```
1       <v-list-item-content
2           :class="$vuetify.breakpoint.name === 'xs' ? '' : 'mr-n16'"
3       >
4           <!-- ... -->
5       </v-list-item-content>
```

Výpis kódu 5.6 Práce s velikostí obrazovky

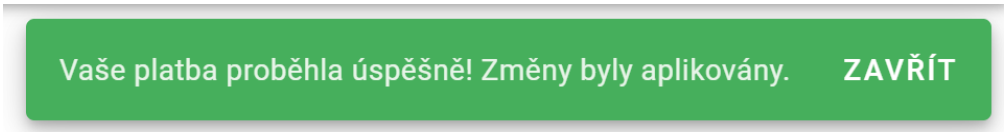
Kromě uzpůsobení na základě velikosti obrazovky dle jednotlivých skupin mi občas přišla také velice užitečná možnost zjistit konkrétní šířku obrazovky a na základě toho měnit vlastnosti komponent. Toho lze docílit snadno obdobným způsobem jako na ukázce 5.6, pouze místo `name` se zjistí `width`. Vzhledem k podobnosti zde konkrétní ukázku neuvádím.

Alespoň za stručnou zmínku stojí také tzv. mřížkový systém, známější možná pod svým anglickým názvem Grid system. Velice prospěšné mi zde byly zejména komponenty `v-row` a `v-col`, které, jak už jejich název napovídá, pomáhají jistým způsobem rozčlenit dostupný prostor do řádků a sloupců. Pro jednotlivé velikosti obrazovek lze každému sloupci `v-col` v řádku `v-row` přidělit množství bodů, které v něm z maximálního počtu dvanácti zaujímá. Pokud součet bodů u sloupců `v-col` v daném řádku `v-row` převyšuje toto číslo, je daný `v-col` přesunut do dalšího řádku (tentokrát nikoliv `v-row`, ale řádku viditelného z hlediska uživatelského).

5.2.4.2 Notifikace uživatelů

Pro notifikace uživatele jsem se rozhodl použít Vuetify komponentu `v-snackbar`, která mi umožnila přehledné zobrazování zpráv uživateli. Mohu u ní nastavit dobu zobrazení, pozici zobrazení, její barvy i zabudovat tlačítka, kterými může uživatel případně i reagovat na akci, o které je informován. Na obrázku 5.1 je ukázka notifikace, která se uživateli zobrazí například ve chvíli, kdy provede úspěšný nákup okamžitých synchronizací. Neboť se jedná o úspěch, zvolil jsem pro něj tradiční zelenou barvu. Pozice notifikací jsem zasadil do pravého horního rohu aplikace.

Obrázek 5.1 Ukázka notifikace o úspěšně provedené platbě a provedení změn



5.3 Sentry

V rámci analýzy současného stavu Timer2Ticketu jsem se dozvěděl, že pro monitoring chyb je v něm používán nástroj Sentry. Vzhledem k tomu, že jsem zjistil, že jeho užívání je ve firmě Jagu s. r. o., která tento projekt zaštituje, zvyklostí, rozhodl jsem se, že i nový frontend bude tento nástroj využívat.

Hlavní výhodou jeho používání je, že se stará o monitorování vzniklých chyb, které loguje. Pro jeho zprovoznění stačilo vytvořit nový projekt po boku projektu původního Timer2Ticketu

v tomto nástroji a Sentry ve vznikající Vue.js aplikaci inicializovat. Kromě reportovaní samotných chyb jsem byl díky němu schopen zjistit daleko více informací, které bych bez něj jen stěží dohledával – zejména v případě, že by chyba vznikla jinde, než u mě na stroji. Poskytuje širší kontext o chybě, která nastala. Pokud chyba vznikla opakovaně, je zde informace o tom, kdy nastala poprvé, kdy naposledy a u kolika uživatelů se objevila. Součástí je vždy i IP adresa, kterou byla chyba vyvolána, prohlížeč, v němž chyba vznikla, komponenta, v níž chyba začala, či například fáze životního cyklu, v níž k chybě došlo (v případě Vue.js aplikace). [82]

Kromě zalogování chyby nástroj umožňuje také další akce jako je přiřazení řešitele, označení chyby jako vyřešené, možnosti nastavit ignoraci chyby a řadu dalšího, bude tedy bezpochyby pro aplikaci užitečný. Za jeho výhodu lze považovat i možnost napojení na externí systémy jako je Slack, díky čemuž lze o chybách rychle notifikovat možné řešitele. [82, 83]

5.4 Auth0

Jak vyplynulo již při návrhu, pro přístup do aplikace musí proběhnout registrace a přihlášení uživatele. Proto jsem se rozhodl využít autentikační a autorizační[1] platformu Auth0 [84], která tyto požadavky plně řeší. Kromě těchto požadavků řeší samozřejmě i požadovanou možnost resetování hesla a korektní odhlašování uživatelů. Když si proto nyní zpětně vybavím detailní návrh wireframů prováděný pro tyto části, uvědomuji si, že oproti ostatním částem návrhu nebylo potřeba dělat ho tak detailní, protože řešení již existovalo (jen jsem to v té době ještě nevěděl).

Tato platforma je nabízena v několika členských plánech [85]. Pro účely vytvoření prototypu mi bohatě stačilo vytvořit Auth0 účet s plánem zdarma. V tom jsem pro svůj Timer2Ticket projekt musel vytvořit tzv. Auth0 Aplikaci, na kterou jsem svoji vytvářenou implementaci napojil, přičemž mi byl velice nápomocný jejich ukázkový návod [86], vytvořený přímo pro Vue.js aplikace, který mi poskytl potřebný Auth0 plugin. Pomohl mi také například s nastavení povolených URL, na které může být uživatel zpětně přesměrován po úspěšném přihlášení, či těch, na které může být přesměrován po odhlášení z aplikace. Zároveň mi poskytl potřebný kód pro navigátora `authenticationGuard`, který zajistí, že pokud uživatel není přihlášený, do aplikace se nedostane a je přesměrován na stránku s přihlášením, kde je nucen se přihlásit či zaregistrovat. Pokud přihlášený je, navigátor ho pustí na požadovou stránku aplikace. Tento navigátor funguje na obdobném principu jako `navigationGuard`, který jsem vytvořil sám a uvedl na ukázce 5.4.

Auth0 nabízí univerzální přihlašovací stránku [87], kterou přináší ve dvou designových variantách, jimi označovaných klasická a nová. Kromě designového hlediska mezi nimi není podstatný rozdíl. Z čistě osobního hlediska jsem proto zvolil klasickou variantu. Tato volba jde však velmi jednoduše kdykoliv změnit v nastavení Auth0. Užitečnou vlastností této univerzální přihlašovací stránky je také možnost její customizace. Lze zvolit barvy odpovídající vzhledu zbytku aplikace Timer2Ticket, je možné přidat i vlastní logo aplikace a řada dalšího. Možnost přidání vlastního loga jsem si vyzkoušel, avšak zatím ho tam nepřidával, neboť na mě na přihlašovací stránce působilo redundantním dojmem.

Už jen použitý základní plán zdarma umožní mít až 7 000 aktivních uživatelů a neomezený počet přihlášení. Zároveň poskytne možnost využít až dvě sociální spojení, čímž jsou označovány možnosti přihlášení přes služby jako je Google či Facebook, přičemž Google přihlašování jsem zprovoznil již v rámci prototypu. V rámci uživatelského účtu Auth0 je dostupná také záložka monitoring, která obsahuje řadu logů, například o přihlašování do aplikace a odhlašování z ní, což může být v budoucnu užitečné při řešení problémů a chyb. [86, 84]

Jedinou nevýhodu, kterou jsem v plánu zdarma zaznamenal, je, že doména pro přihlašování do aplikace se nedá změnit. Zůstává tam nyní proto řetězec vygenerovaný od Auth0. Avšak při využití jakéhokoliv z placených plánů již lze doménu customizovat. V případě úspěchu aplikace Timer2Ticket na trhu však předpokládám, že bude stejně nutné využít některý

[1] Autentizace se týká ověření pravosti identity, zatímco autorizace se týká udělení opravnění na určitou činnost a na autentizaci typicky navazuje.

z nich – kvůli umožnění většího množství aktivních uživatelů. Tento nedostatek plánu zdarma pro nás tedy zanikne. V placených plánech již také není limitován počet sociálních spojení či jsou poskytovány pokročilejší možnosti jako například vícefaktorové ověřování (tzv. MFA neboli Multi-Factor Authentication). Kromě připravených placených plánů, lišících se zejména cenou a možným počtem aktivních uživatelů, Auth0 nabízí také možnost kontaktovat je a získat řešení na míru. [85, 84]

5.5 Session storage

Jak bude později uvedeno v kapitole 6 o testování, při akceptačních testech se zadavatelem vyvstal dodatečně požadavek, aby v případě odstranění spojení bylo uživateli umožněno vzít akci zpět, pro případ, že by tak zásadní akci provedl omylem. S ohledem na to jsem k notifikaci, která informuje o odstranění boxu, přidal další tlačítko, které mu tuto činnost umožní.

Musel jsem však vyřešit, jakým způsobem zpracovat spojení, které má být odstraněno. Pro tento účel jsem se rozhodl využít session storage, což je lokální uložiště v prohlížeči. Pro přístup k němu lze použít JavaScript a data tam lze uložit do doby, než dojde k zavření prohlížeče. Následně jsou smazána. Déle je však udržovat nepotřebuji. Na rozdíl od toho velmi podobné local storage umožňuje uložení dat na neomezenou dobu. [88]

Když uživatel zadá požadavek na odstranění, uložím si do session storage příslušné spojení a jeho pozici. To učiním pomocí funkce `setItem`, kterou na `sessionStorage` zavolám. Jako první argument jí vždy poskytnu klíč, pod který chci ukládat, jako druhý argument pak uvedu ukládaný prvek. Pro ukládání však musím každý objekt převést na řetězec, protože jedině ty umí uložiště skladovat. Pokud následně uživatel klikne na tlačítko, které má zajistit vrácení spojení, získám ho i jeho pozici zpětně ze session storage pomocí funkce `getItem`, kterou na `sessionStorage` zavolám. Zde vždy stačí zadat pouze klíč, pod kterým je požadovaný řetězec uložen. Ten převedu zpět do podoby, ve které byl před převedením na řetězec, a spojení vložím na původní pozici.

5.6 Event bus

Při implementaci tlačítka „Upgrade“, které se nachází na boční liště aplikace, jsem se snažil docílit toho, aby při kliknutí na něj, vždy došlo k přesměrování uživatele na stránku „Členství“ (pokud tam ještě nebyl) a otevřít mu tam první box obsahující změnu členství. V případě, kdy byl uživatel na jiné stránce, využil jsem k tomu parametrů, které Vue Router umožňuje posílat, avšak v případě, kdy jsem se již na stránce nacházel, tato možnost použít nešla, protože nedocházelo ke směrování, a komponenty, mezi nimiž jsem potřeboval komunikovat, nebyly v rodičovském vztahu, což snadnou komunikaci mezi nimi také znemožňovalo.

Řešením tohoto problému se ukázal být tzv. event bus. Jedná se o separátní Vue.js instanci, která umožňuje emitovat (tj. vysílat) události i s daty v rámci jedné komponenty a následně jim naslouchat a reagovat na ně v jiné (bez jakéhokoliv mezikroku). Pro její využití mi stačilo ji pak importovat do obou komponent, do komponenty, která vysílala požadavek na otevření boxu, přidat do metody, která zpracovává kliknutí na tlačítko, druhý řádek z ukázky 5.7 a do komponenty, ve které jsem chtěl požadavek realizovat, možnost tohoto požadavku v `created` fázi komponenty zaregistrovat (aby byl očekáván v případě vyvolání), a to způsobem jakým je znázorněno na prvním řádku zmíněné ukázky. Obdobné řešení jsem pak s vytvořeným event busem následně použil i na několika dalších místech. [89, 90]

```
1  EventBus.$on("membership-change-open", () => { this.panel = [0]; });
2  EventBus.$emit("membership-change-open");
```

Výpis kódu 5.7 Použití event busu v komunikujících komponentách

Kapitola 6
Testování

V této kapitole nejprve vysvětlím, co to je a jaký význam pro mě mělo akceptační testování. Dále popíšu změny oproti původnímu návrhu, které byly na základě tohoto testování provedeny. Následně popíšu, co to jsou a jakým způsobem jsem využil testy použitelnosti. Uvedu také poznatky zjištěné na základě tohoto testování s jednotlivými uživateli. Na závěr pak stručně shrnu obecné závěry, které z testování s jednotlivci vyplynuly.

6.1 Akceptační testování

Akceptační testy jsou takové testy, které se typicky provádějí již na straně zákazníka dle předem připravených scénářů, a to za předpokladu, že veškeré předchozí testování se obešlo bez závažnějších problémů. Jejich cílem je ověřit, že mezi specifikací aplikace a samotnou aplikací nejsou žádné nesrovnalosti a vytvořený produkt má dohodnuté funkcionality a kvalitu. Pokud dojde ke zjištění jakýchkoliv nesrovnalostí, je potřeba je evidovat a co nejrychleji zajistit opravu, neboť se jedná o situaci, která může vést ke zpoždění dodávky softwaru i úspěchu projektu. [91, 92]

V rámci implementace jsem průběžně představoval hotové části jednotlivých stránek prototypu zákazníkovi, svému zadavateli, Ing. Jiřímu Hunkovi. Na základě těchto prezentací jsem dostával cennou zpětnou vazbu a podněty ke změnám. Nelze říci, že by se jednalo o akceptační testování v pravém slova smyslu, neboť nebylo prováděno podle předem připravených scénářů a kromě kontrol nesrovnalostí s původně vytvořeným návrhem se řešily i možné dodatečné funkcionality, nicméně cílem těchto představení bylo zajištění spokojenosti zákazníka s finálním prototypem. Troufám si proto tvrdit, že o jistou formu akceptačního testování šlo.

6.1.1 Změny oproti původnímu návrhu

V této sekci rozeberu podstatné změny, které byly v implementaci prototypu provedeny rozdílně vzhledem k popsanému návrhu na základě diskuzí s Ing. Jiřím Hunkou. Zároveň zde uvedu prvky, které byly nově přidány – kromě řady tooltipů, které jsem přidal, a jejichž výčet by neměl přidanou hodnotu.

Přidání Undo tlačítka

První z funkcionalit, která byla v původním návrhu opomenuta, avšak po diskuzích později byla přidána, je tlačítko „Undo“ v rámci notifikace, která informuje o odstranění boxu se spojením. Hlavní motivací k jejímu přidání byl fakt, že uživatel může na tlačítko odstranění kliknout omylem a bez této funkcionality by musel celou konfiguraci dané synchronizační dvojice provádět znovu.

Kromě této varianty byla diskutována i možnost využití dialogového okna, které by uživatele nutilo krok odstranění potvrdit. Nakonec jsem se však rozhodl pro tuto variantu a zahrnul odstraňování do následných testů použitelnosti, abych zjistil pohled dalších uživatelů na tuto funkcionalitu a případný zájem o dialogové okno.

Přidání rozbalitelného uživatelského profilu

Další z provedených změn se týká informací roztříštěných po boční liště a záhlaví – konkrétně tlačítek „Upgrade“ a „Koupit okamžité synchronizace“ a informace o počtu dostupných okamžitých synchronizací. Řada aplikací má ve svém pravém horním rohu dostupný jakýsi stručný rozbalitelný uživatelský profil, jehož součástí je i možnost odhlášení. Jmenovitě se může jednat například o Google účet v prohlížeči. Při diskuzích jsem byl na tuto skutečnost upozorněn a rozhodl se roztříštěné prvky centralizovat do takovéhoto profilu. Kromě počtu dostupných okamžitých synchronizací jsem přidal ještě informace o současném členství. Přidal jsem zde také email, pod nímž je uživatel registrován, odkaz na stránku s nastavením profilu a zmíněné tlačítko pro odhlášení. Neboť se jedná o naprosto nový prvek oproti návrhu, je k nahlédnutí v příloze A na obrázku A.1.

Začlenění uživatelů bez členství do prostoru aplikace

Třetí změna, kterou jsem oproti návrhu provedl, se týká přihlášených uživatelů bez zvoleného členství. Původní návrh vyčleňoval tyto uživatele z jinak tradičního prostředí aplikace tím, že jim nezobrazoval ani záhlaví ani boční lištu. Jak si možná již pečlivý čtenář všiml, tento fakt uživateli v původním návrhu neumožňoval přistoupit k tlačítku odhlášení a byl proto v tomto ohledu jistě chybný. Už jen z toho důvodu jsme se při diskuzích rychle shodli na této změně a boční lištu a záhlaví jsem přidal i pro uživatele bez zvoleného členství. Funkcionality, které jsou na nich jinak funkční pro uživatele se zvoleným členstvím, jsem pro uživatele bez zvoleného členství samozřejmě znefunkčnil.

Změna značení u ceny okamžitých synchronizací

Následující implementačně drobná avšak pro uživatele možná důležitá změna se týká nákupu okamžitých synchronizací. Původní box, v němž došlo k této změně, je na obrázku 4.12. K ceně, která se nachazí v pravé části boxu jejich nákupu, bylo přidáno znaménko plus. Při diskuzích jsme totiž došli k závěru, že v kombinaci s měsíční cenou placenou za členství by jinak další informace o ceně mohla působit matoucím dojmem.

Změna podoby možných členství

Původní myšlenkou návrhu bylo, aby vzhled jednotlivých členství odpovídal tomu, který bude na propagačním webu, a to zejména proto, aby byla zachována konzistence mezi propagačním webem a samotnou aplikací. Tento jejich původní design je k nalezení na návrhu 4.15 a zůstal zachován na propagačním webu, kde je více prostoru. Když jsem však chtěl naprosto identický design použít v aplikaci, při rozbalení boxu na stránce „Členství“ jsem se potýkal s problémem. Tento design neumožňoval jednoduše zajistit, že se celé vejdou na jednu obrazovku (s ohledem na záhlaví aplikace a stránky „Členství“). Po diskuzi se zadavatelem jsme se proto rozhodli zachovat pouze barevný ráz jednotlivých členství a klíčové informace uspořádat do jednotlivých řádků, na což je možné nahlédnout v příloze A na obrázku A.2. Kromě této designové změny jsem zde také přidal barevné zvýraznění zvolené změny členství. Tento pozměněný design jednotlivých členství jsem pak použil také při volbě prvního členství.

Změna podoby chybových hlášek při provádění nákupu nového členství

Na přiloženém médiu jsou k dispozici detailní wireframy celé aplikace, jejichž množství je tak rozsáhlé, že jsem ho v kompletní podobě neumisťoval ani do přílohy. Jejich součástí je i podoba chybových hlášek, které návrh předpokládal a mohou nastat při nákupu nového členství. Původně se počítalo s červenou textovou informací pod tlačítkem „Prohlédnout si souhrn“ na stránce „Členství“. Když jsem však prezentoval naimlementované řešení zadavateli, ukázalo se, že informace je obtížně dohledatelná a uživatel tak neví, co se stalo. Neboť se však nejedná o klasické vyplňování formuláře, který by se jednoduše zvalidoval a obarvil, textovou informaci jsem nahradil dialogovým oknem informujícím o nastátých problémech. Abych byl konzistentní, použil jsem stejné řešení i v případě první volby členství. Ukázka vytvořené chybové hlášky formou dialogového okna je k nalezení v příloze A na obrázku A.6.

Přidání platebních údajů do souhrnu informací o nákupu

Další ze změn se týkala souhrnu informací o nákupu, jehož původní navržená podoba je součástí obrázku 4.13. Vzhledem k tomu, že cílem souhrnu je informování uživatele o kompletní podobě nákupu, který navolil v předchozím kroku před stiskem tlačítka „Prohlédnout si souhrn“ a chystá se ho realizovat, bylo na místě přidat do souhrnu také informace o zadaných platebních údajích, protože pokud je má zadané z dřívějška a box s nimi nerozbalil, mohl je zapomenout změnit, i když by to jinak chtěl učinit. Byly proto přidány veškeré možné platební údaje a u neuvedených byla pro přehlednost vyznačena jejich absence. Na upravený souhrn lze nahlédnout v příloze A na obrázku A.3.

Přidání editovacích tlačítek do souhrnu informací o nákupu

I tato změna se týkala souhrnu informací o nákupu, jehož původní navržená podoba je součástí obrázku 4.13. Zde se jedná o naprosto novou funkcionalitu, na kterou vyvstal požadavek při prezentaci implementovaného řešení. Její funkčnost měla zjednodušit pohyb mezi souhrnem a jednotlivými boxy, v nichž probíhá tvorba nákupu. Došlo proto k přidání tlačítek „Upravit“ k jednotlivým částem souhrnu nákupu. Ta zajistí přechod k jednotlivým boxům dostupným před zmáčknutím tlačítka „Prohlédnout si souhrn“ a rozbalí pouze ten box, v němž chce uživatel provést úpravu. Na upravený souhrn lze nahlédnout v příloze A na obrázku A.3.

Změna polohy přepínače aktivity spojení

Původní návrh stránky „Spojení“ počítal s tím, že jednotlivé boxy spojení budou zabírat celou šíři dostupného prostoru. Při vytváření prototypu se však projevil problém s tím, že se nedařilo jednoduše oddělit fungování přepínače aktivity a rozbalování boxu. Pokud by tedy tlačítko umožňující aktivaci a případnou deaktivaci spojení bylo součástí záhlaví boxů, mohlo by se stávat, že při změně aktivity spojení se bude uživateli také otevírat či zavírat box, což by pro něj mohlo být matoucí. Ukázal jsem tento fakt zadavateli a dohodli jsme se na alternativním řešení, ve kterém se přepínač přesunul vedle daného boxu, na což lze nahlédnout v příloze A na obrázku A.4. Tato změna také zjednodušila řešení responzivity a zejména problém s přehledným uspořádáním údajů v záhlaví hlavního boxu na menších obrazovkách.

Změna detailních informací u záznamů o provedené synchronizaci

Poslední změna, kterou zde uvedu se týká stránky „Logy“. Neboť grafická knihovna Vuetify poskytuje předpřipravené tabulky, zvolil jsem připravený základ a ten si následně musel doupravit dle potřeb. Například jsem musel separátně přidat filtrační řádek. Na druhou stranu mi však komponenta poskytla možnost, jak snadno zobrazit detailní informaci o daném záznamu – opět jistou formou rozevíratelných boxů. Využil jsem proto této připravené možnosti, která splňuje

požadavek na možnost vykopírování informace, a použil ji místo původně zamýšleného okna. Zároveň jsem prozatím nechal možnost detailní informace i pro záznamy, u nichž problém nenastal. Nabízí se díky tomu prostor pro lepší informovanost uživatele. Změnu jsem následně představil na diskuzích a byla akceptována. Ukázka detailní informace o záznamu ze synchronizace je k nahlédnutí v příloze A na obrázku A.5.

6.2 Testování použitelnosti

„*Uživatelské testování, nebo více výstižně řečeno testování použitelnosti, je druh experimentu, kdy se snažíme projít s uživatelem či zákazníkem web, e-shop či aplikaci a hledáme problémy, které mohou snížit či úplně znemožnit jejich použití. Výhodou testu použitelnosti je, že nám odhalí možná úskalí, která jsme při návrhu produktu či služby nedomysleli.*“ [93]

Díky provedení uživatelského testování jsme proto schopni zjistit, zda se uživatelům s aplikací dobře pracuje, a snížit tak riziko, že by mohlo dojít k jejímu následnému neúspěchu. Kromě typického testování na vytvořeném prototypu ho však lze použít i na hotové aplikaci, u které chceme zjistit nedostatky. Může jich být celá řada, typicky se však jedná o záležitosti jako je špatné pojmenování, kvůli čemuž uživatel neví, co pojem znamená, nevhodné umístění informací či nestandardní chování prvků. [93]

Klíčovým prvkem při testech použitelnosti je vhodná volba a počet testerů. Dle [94] se ukázalo, že aby se podařilo odhalit veškeré problémy použitelnosti, je potřeba testovat s nejméně patnácti uživateli. Není však vhodné provádět testování postupně s jednotlivci a po každém testování zapracovat změny, protože bychom měli příliš malý vzorek pro získání objektivních dat k posouzení výsledku testování. Zároveň také není vhodné provést pouze jednu iteraci se všemi patnácti testery, protože u pozdějších testerů bychom nezískávali prakticky žádné nové poznatky. Dle [94] je ideální variantou provést testování iterativně a po každé iteraci na základě zjištěných poznatků zapracovat změny. Je však potřeba nezapomínat na to, že i v zapracovaných změnách se mohou opět objevit potíže. Nejlepší je zvolit tři fáze a pro každou tři až pět testerů, přičemž dolní hranice je určena kvůli potřebné diverzitě v uživatelském chování a horní je stanovena dle zjištění, která [94] provedli. Testovat v dané iteraci s více než pěti uživateli je mrhání času. Zároveň platí, že největší množství nových informací typicky získáme od prvního testera. Množství nových informací dodávaných dalšími stále více klesá. Množství testerů pro dané iterace je odlišné v případě, kdy bychom měli více cílových skupin – to však není můj případ. [94]

Na základě výše zjištěných poznatků jsem se rozhodl využít celkem čtyřmi testery, tedy střed z doporučeného intervalu, a jak se později při testování ukázalo, nově objevené nedostatky s přibývajícími testery opravdu ubývaly a již objevené se stále častěji opakovaly. Co se týče volby testerů, abych získal relevantní data, volil jsem z cílové skupiny, tedy z IT odvětví. Pro lepší diverzitu dat jsem dva z testerů zvolil ze současných uživatelů nástroje Timer2Ticket a dva jakožto osoby nepoužívající tento software (avšak opět z IT odvětví).

6.2.1 Obecnosti proběhlého testování

Testování jsem provedl 27. 5. 2022 na FIT ČVUT na základě testovacího scénáře z přílohy B a dotazníku z přílohy C. Testování jsem prováděl kontaktně, vždy na klidném místě, kde jsme nebyli rušeni. Před začátkem testování jsem se každého z testerů zeptal, zda si mohu nahrát zvuk a jejich práci na obrazovce. Všichni s tím souhlasili a umožnili mi tyto nahrávky použít pro zpracování výstupu z testování do této práce. Všechny jsou proto také případně k dispozici na přiloženém médiu. Zároveň jsem je před začátkem testování poprosil, aby se snažili komentovat, co při testování dělají, abych byl lépe schopen zaznamenat jejich vnímání aplikace. Při průchodu testovacím scénářem jsem jim pak vždy po dílčích částech dával prostor pro dodatečné vyjádření k proběhlé části. Za dílčí úspěch testování považuji fakt, že se všemi jsme se vešli do časového

limitu 45 až 60 minut na test, který by neměl být překročen, jinak dochází ke ztrátě pozornosti. Tyto rady a řadu dalších pro zlepšení kvality testování jsem získal z [95, 96].

6.2.2 Poznatky z testování s jednotlivými uživateli

V této sekci postupně detailně rozeberu provedené testování s jednotlivými testery. Na úvod vždy stručně uvedu, v jakém kontextu se daná osoba testování použitelnosti účastnila. Následně shrnu pozitivní závěry, které z testovní s ní vyplynuly, a nakonec nedostatky, které se při daném testování objevily. Součástí budou i návrhy na změny a vylepšení, které mi v rámci testování sdělili.

6.2.2.1 Ing. Oldřich Malec

Kontext: Ing. Malec je vyučujícím na FIT ČVUT a zaměstnanec společnosti Jagu s. r. o. znalý aplikace Timer2Ticket, neboť ji aktivně používá při své práci, kde se mimo jiné zaměřuje právě na frontend aplikací.

Pozitivní závěry: Ing. Malec se v aplikaci, zejména pak v částech, které byly obsaženy i v původním Timer2Ticketu, pohyboval velmi svižně a dopředu prozkoumával i části, které ještě nebyly na řadě ve scénáři, což pravděpodobně ovlivnila právě i jeho předchozí zkušenost s touto aplikací. Zároveň svůj pohyb po aplikace opravdu hodně komentoval, sděloval své myšlenky a nápady na zlepšení, díky čemuž jsem získal opravdu bohatou zpětnou vazbu. Ocenil, že:

- Aplikace disponuje velkou šíří notifikací uživatele a nápovědných tooltipů.
- Při provedení volby jednoho nástroje k synchronizaci dojde k úpravě seznamu zbylých nástrojů, které lze synchronizovat, dle toho, které jsou připraveny na synchronizaci s ním.
- Změna rozvrhu synchronizace je přehledná.
- Má možnost využívat stručný uživatelský profil v pravém horním rohu. Často ho využíval při pohybu po aplikaci.
- Má možnost rozšiřovat základní členství dalšími aktivními spojeními nad rámec základního tarifu.
- Při žádosti o ukončení daného členství může získat kompenzaci ve formě okamžitých synchronizací.
- V případě časové zóny, která obsahuje velké množství časových pásem, je možné vyhledávat.
- Aplikace má líbivý design.

Mezi další pozitiva vyplynulá z tohoto testování lze řadit:

- Správně identifikoval povinná pole při vyplňování formulářů.
- Tabulku se zpracovanými objednávkami označil za užitečnou.
- Z vytvořených členství si byl schopen vybrat vyhovující. Sdělil mi, že naprosto ideální by pro něj bylo provádět synchronizaci časových záznamů jednou denně a konfiguračních objektů dvakrát denně. Zároveň se však nechce zbytečně zatěžovat obsluhou aplikace a používáním okamžitých synchronizací. Dokázal by si představit využívání aplikace se členstvím Junior, který mu přesně vyhovuje možným počtem aktivních spojení – druhé by si nyní rád do současné aplikace přidal. Synchronizace každou hodinu by pro něj byla pohodlná, pět spojení by však nevyužil. Cena za Senior členství se mu zdála příznivá, tedy jeho použití nevyloučil.

- Případnou přítomnost možnosti měnit pořadí boxů neoznačil za pro něj jakkoliv podstatnou funkcionalitu (spíše jako hezký doplněk).

Objevené nedostatky a návrhy na změny:

1. Při první volbě či změně členství by se mu líbilo, kdyby mohl klikat i na čtverce s jednotlivými členstvími. Klidně by v takovém případě odstranil radio buttons.
2. Při změně konkrétního času synchronizace by jakožto Evropan ocenil, kdyby komponenta, v níž výběr provádí, umožňovala 24hodinový formát času místo 12hodinového s volbou AM/PM. Doporučil mi, že rozhodnutí o tom, kterou variantu z časových formátů zvolit, by se mělo dát pomocí nastavení prohlížeče.
3. Při změně rozvrhu synchronizace chtěl provést uložení, avšak nevěděl kde. Neočekával, že se změna uloží sama. V této části je buď potřeba přidat notifikaci o provedené změně či tlačítko na uložení. Stejný problém odhalil při změně nastavení notifikací a časového pásma v nastavení profilu.
4. V případě čtveřice dlouhých tlačítek v boxu spojení by ocenil přítomnost ikon, které by mu zjednodušili zapamatování významu tlačítek. Obdobně jako jsou ikony na boční liště (v hlavním menu). Přišel také s návrhem na úpravu tooltipu pro tlačítka na okamžité synchronizace. Ocenil by zde nápovědu, kde je může zakoupit ve chvíli, kdy mu došly. Zároveň by se mu líbilo, kdyby měl informaci o dostupných okamžitých synchronizacích přímo u tlačítek, které je umožňují.
5. Zarazilo ho chování komponenty starající se o nákup okamžitých synchronizací, která umožňovala nákup pouze po dvaceti a byla připravena na tuto inkrementaci pomocí zobrazených šipek. Nejprve očekával, že bude moci číslovku i psát, což mu komponenta nedovolila a jeho změny nulovala. Teprve následně z nutnosti situace využil šipek. Navrhl zde, že by se mu zdálo přirozenější, mít informaci o tom, kolik stojí jeden balíček, a volit pouze počet balíčků, díky čemuž by ani nebylo nutné validovat požadovaný násobek dvaceti.
6. Očekával, že na stránce se členstvími bude mít předvoleno členství, které právě má. Vzhledem k tomu, že stránka umožňuje nákup okamžitých synchronizací i změnu členství, nebyl si jistý, co má udělat, když zvolil nějaké členství a nemohl zrušit jeho výběr (pouze změnit). Přiznal, že intuitivnější by pro něj bylo mít oddělené okamžité synchronizace od změny členství, když jsou placeny jednorázově. Když však zjistil, že je používám jako kompenzaci při rušení členství, současný návrh mu začal dávat smysl. Má však zkušenost s platební bránou Stripe a upozornil mě, že si není jistý, jak pokročilé možnosti bude nabízet pro účely mého návrhu – zejména náhrady zbytku členství za okamžité synchronizace a propojení jednorázové a měsíční platby.
7. Nebyl mu jasný význam tlačítka pro zobrazení souhrnu objednávky, které má v angličtině název „See summary“. Nejprve uvedl, že by zde očekával spíše „Proceed to payment“, avšak když si všiml, že pokud zvolí pouze Hobby členství s jedním aktivním spojením, nemusí dojít k platbě, ale pouze ukončení předplatného, zavrhl i tuto variantu. Uvedl, že nejpřehlednější by pro něj bylo, kdyby se text tlačítka dynamicky měnil mezi „Proceed to payment“ a „Cancel subscription“.
8. Na přehledu vytvořené objednávky by se mu líbilo, kdyby mohl pomocí tlačítka pro úpravu provést úpravu přímo ve vytvořeném souhrnu.
9. V rámci seznamu logů by mu více vyhovovalo, kdyby mohl sloupec s identifikátorem boxu spojení filtrovat na základě výběru z možností, které by se mu automaticky vyplnily na základě existujících boxů. Zároveň by se mu líbilo, kdyby mohl detail o konkrétním záznamu synchronizace otevřít kliknutím kamkoliv na řádek – ne pouze na šipku na pravé straně záznamu. Také mi doporučil využití odlišné ikony pro zobrazení statusu záznamu. Vhodnější by se mu zdálo zvolit ikonu bez rámu, aby to v uživateli neevokovalo možnost kliknutí na ni.

10. V zobrazení na mobilu mu přišlo, že tabulka logů obsahuje redundantní horizontální pruhy (okraje boxů). Bez nich by to pro něj bylo přehlednější. Na mobilu by také preferoval nějak akcentovat názvy sloupců, které informují o možnostech jednotlivých členství, protože mu přišly málo výrazné. V zobrazení na mobilu by také preferoval uspořádání textových prvků v boxu rozvrhu spojení po řádcích.
11. Očekával by v angličtině odlišný název tlačítka pro načtení posledního použitého plánu (po zrušení předplatného). Místo „Load my last subscription plan“ by použil „Renew last subscription plan“.
12. Při zajišťování posunu času z důvodu přestěhování na Americkou Samou se toho nejprve snažil dosáhnout pomocí změny času v rozvrhu. Až následně, když mu to přišlo těžkopádné, se podíval do nastavení profilu, kde nalezl volbu časové zóny.
13. Tabulka se zpracovanými objednávkami by mu přišla užitečnější, kdyby umožňovala filtraci.
14. Doporučil mi zachovat konzistenci v barvách tlačítek. U tlačítek se změnou hesla, použitím kupónu a změnou platebních údajů bylo zapomenuto šedé zbarvení, i pokud jsou aktivní.
15. Líbilo by se mu, kdyby měl možnost zvolit vlastní pojmenování pro daný box spojení.
16. Na menší obrazovce, například pro mobil, mu chybělo tlačítko pro otevření boční lišty (hlavního menu).
17. Preferoval by, kdyby volba času synchronizace daného spojení neobsahovala nastavení hodiny v případě zaškrtnutí synchronizací každou hodinu. Nedává mu to smysl.
18. Uvedl, že notifikaci o obdržených okamžitých synchronizacích by nevyužíval. Naopak by se mu hodilo, kdyby mohl být notifikován o tom, že k synchronizaci došlo.

6.2.2.2 Ing. Filip Glazar

Kontext: Ing. Glazar je vyučujícím na FIT ČVUT a zaměstnanec společnosti Jagu s. r. o. znalý aplikace Timer2Ticket, neboť ji aktivně používá při své práci.

Pozitivní závěry: Ing. Glazar se pohyboval po aplikaci velice plynule, a to i mezi jednotlivými stránkami. Stejně jako v případě Ing. Malce se, až na výjimečné situace související s nedostatky návrhu uvedenými níže v přehledu těchto nedostatků, zastavoval pouze ve chvílích, kdy mi chtěl sdělit jeho postřehy a návrhy na změny. Díky tomu jsem byl i v jeho případě schopen získat mnoho podnětů na možné změny. Ocenil, že:

- Aplikace obsahuje velké množství tooltipů, zejména v oblasti konfigurace nástrojů.
- U jednotlivých boxů spojení má tlačítko, které ho přesune k odpovídajícím záznamům o synchronizaci vyfiltrovaným pro daný box.

Mezi další pozitiva vyplynulá z tohoto testování lze řadit:

- Chválil design a celkově prostředí aplikace.
- Uvedl, že pro jeho používání aplikace mu žádná klíčová funkcionalita neschází.
- Tabulku se zpracovanými objednávkami označil za užitečnou.
- Ačkoliv žádné členství přesně nepasuje k jeho potřebám, z vytvořených členství si byl schopen vybrat vyhovující. Vybral by si Senior členství, protože je zvyklý provádět synchronizace každou hodinu. Nevyužil by však jeho možnost pěti aktivních spojení. Nyní by mu stačilo jedno. Cena je sice nejvyšší ze všech základních plánů, ale byla by pro něj akceptovatelná. Přiznal, že zde může být trochu ovlivněn tím, že již zná výhody používání této aplikace, ale i tak se mu cena zdála pro programátora velice příznivá.

- To, že při volbě času synchronizace daného spojení lze nastavit hodinu i přes to, že je nastaveno pravidelné synchronizování každou hodinu, ho nijak nerušilo. Nepůsobilo by na něj nikterak matoucím dojmem, pokud by se však v takovém případě změnila volba jen na minuty. Bylo by to pro něj také v pořádku.
- Nepřítomnost možnosti měnit pořadí boxů mu nevadila. Využití by nalézal pouze v případě, že by se například tlačítka pro okamžitou synchronizaci přesunula vně boxu.

Objevené nedostatky a návrhy na změny:

1. Při vyplňování formuláře s platebními informacemi se objevily potíže s málo výraznou informací o povinných polích, které je nutné vyplnit. Automaticky vyplnil všechny, a když jsem se ho následně dotázal, zda si všiml, co bylo povinné a co volitelné, nevěděl. Také nevěděl, kam zadat číslo popisné.
2. Při přidání boxu spojení se projevil nevhodně nastavený časovač, který umožňuje automatické rozbalování konfigurace nástrojů po přidání boxu. Box se tedy nerozbalil, a jak jsem již očekával, Ing. Glazar okamžitě vyslovil žádost na tuto funkcionalitu.
3. Byl by radši, kdyby jednotlivé boxy na stránce „Členství“ byly výrazněji odlišeny. Jak jsou blízko u sebe, zdálo se mu, že splývají. Líbilo by se mu, kdyby buď byly dále od sebe či se jejich barva změnila z bílé na nějakou jemně pastelovou. Podobný problém popsal i na stránce „Spojení“, kde mu splýval box s konfigurací spojení a nastavením rozvrhu synchronizace. Po uložení konfigurace box s rozvrhem přehlédl a chvíli ho musel hledat.
4. Doporučil mi, abych u přepínače aktivity spojení signalizoval rozdílnost stavu aktivní a deaktivováno přímo textovým popiskem nad daným přepínačem.
5. Formulář se změnou hesla očekával na speciální stránce.
6. Při čtení notifikací občas nestíhal notifikaci dočíst a zažádal o prodloužení času jejich zobrazení.
7. Líbilo by se mu, kdyby při odstranění boxu spojení měl dialogové okno i tlačítko pro vrácení akce v notifikaci, když už musel dohledávat všechny informace ke konfiguraci daného spojení. Přiznal však, že o sobě ví, že v tomto případě je jeho požadavek hodně specifický.
8. V souhrnu objednávky by se mu líbilo, kdyby u jednotlivých placených položek měl informaci o tom, zda jsou placené jednorázově či měsíčně, přímo vedle ceny, nejen v levém informačním sloupci.
9. Líbilo by se mu, kdyby boční lišta (hlavní menu) mohla být zmenšitelná i na větších obrazovkách. Chování přirovnal k tomu, které je na GitLabu.
10. Množství textových informací v boxu spojení o příští a poslední synchronizaci na něj bylo příliš velké a nepřehledné. V případě poslední synchronizace by více ocenil jen ikonu informující o úspěchu či neúspěchu (barevný křížek či fajfku). Kvůli notifikaci o zahájení první synchronizace, která zmizela příliš pozdě, si nevšiml přidané informace o poslední provedené synchronizaci, díky čemuž mělo být jasné, že tato synchronizace již doběhla. Uvedl, že v případě barevné značky by se orientoval lépe.
11. Znaménko plus u nákupu okamžitých synchronizací, které bylo přidáno v akceptačních testech se zadavatelem, na něj působilo matoucím dojmem, když měl rozbalený pouze tento box (s nákupem okamžitých synchronizací). Nechápal k čemu se přičítá. Vše se mu ujasnilo až na souhrnu, kde již byla informace bez znaménka plus.
12. Při pohybu po aplikaci příliš nevyužíval stručný uživatelský profil v pravém horním rohu. Přiznal, že pro něj nemá velký význam. Přišel se zajímavou myšlenkou, která by ho pro něj udělala atraktivnějším, avšak podmíněnou pojmenováním boxu spojení. Líbilo by se mu, kdyby v tomto stručném profilu bylo nějaké rozbalovací menu, v němž by šlo dle pojmenování vybrat daný box spojení a přímo odsud pro něj provést okamžité synchronizace. Pokud by tato funkcionalita nebyla, ocenil by informaci o počtu okamžitých synchronizací přímo někde na stránce „Spojení“.

13. Uvítal by možnost notifikování na email o tom, že došlo k okamžitým synchronizacím. Informace o provedení pravidelných synchronizací by pravděpodobně s ohledem na zvolenou hodinovou periodu synchronizace nevyužíval. Možnost být notifikován o obdržených okamžitých synchronizacích by spíše nevyužil. Navíc anglický textový popis pro něj nebyl dostatečně přesný – nebyl si jistý, zda jde o informaci, že je obdržel či se mu provedly.
14. Stejně jako Ing. Malec by ocenil možnost mít vlastní pojmenování u daného boxu spojení. Také mu chyběla notifikace či tlačítko na uložení při změně časové zóny, notifikací a rozvrhu synchronizace spojení. Stejně tak označil jako nevhodný i název tlačítka pro přechod k přehledu objednávky, ačkoliv po něm intuitivně sáhl při postupu v objednávce. Na menší obrazovce také očekával tlačítko pro otevření boční lišty.

6.2.2.3 Ondřej Wrzecionko

Kontext: Ondřej je student a zároveň vyučující (cvičící) v předmětech PA1 a PA2 na FIT ČVUT. V současné době je v oboru zaměstnaný na poloviční úvazek. Do doby testování byl neznalý aplikace Timer2Ticket.

Pozitivní závěry: Na vzdory mému očekávání se Ondřej v aplikaci zorientoval velmi rychle i přes to, že se s ní setkal naprosto poprvé. Intuitivně se pohyboval procesem konfigurace nástrojů, jejich změnou i procesem nákupu na stránce „Členství“, kde měl dokonce tendenci předbíhat testovací scénář. Stejně jako předchozí dva testeři, i on pečlivě komentoval svůj průchod aplikací, díky čemuž mi usnadnil dokumentaci testování. Ocenil, že:

- Má možnost využít rychlý přesun po aplikaci prostřednictvím stručného uživatelského profilu v pravém horním rohu. Využíval ho ze všech testerů nejčastěji.
- Dochází ke zčervenání informací o vybraném členství při volbě prvotního členství a změnách členství.
- Má možnost pohybovat se po konfiguraci nástrojů pomocí klávesových zkratek, které se chovaly dle jeho očekávání.

Mezi další pozitiva vyplynulá z tohoto testování lze řadit:

- Ve formuláři s platebními informacemi si všiml hvězdičky a správně určil, které údaje jsou povinné.
- Chválil design uživatelského rozhraní a přehledný proces konfigurace nástrojů i volby času pravidelné synchronizace.
- Již s třetím testerem se ukázal potenciál všech tří členství i okamžitých synchronizací. Přiznal, že z nabízených členství by mu pravděpodobně stačilo základní Hobby členství s jedním spojením. Synchronizace jednou týdně by pro něj sice nebyla občas dostatečná, ale s dokoupením okamžitých synchronizací by mu to již stačilo. Pokud ne, s ohledem na cenu by byl ochotný investovat i do Junior členství. Možností Senior členství by nevyužil.
- Nepřítomnost možnosti měnit pořadí boxů mu nevadila. Funcionalitu by s ohledem na jeho použití aplikace nevyužil.
- Posílání informací o zpracovaných objednávkách na email by mu stačilo. Tabulku se zpracovanými objednávkami by nevyužíval, ale nijak ho nerušila.

Objevené nedostatky a návrhy na změny:

1. Při deaktivaci spojení narazil na potíž s pozicí přepínače. Nejprve ho napadlo odstranit všechny dny z rozvrhu synchronizace. Když však zjistil, že to není možné, hledal dále, a to ve výsledku i na jiných stránkách než je stránka „Spojení“. Přepínače pro deaktivaci si všiml až při opětovném příchodu na tuto stránku, když byly spojení zabalené. Na

závěr se zamyslel, co mohlo způsobit takový problém s přepínačem, a přišel s návrhem zafixovat ho, aby při rozbalení boxů nemizel z obrazovky. Usoudil, že problém u něj nastal pravděpodobně právě zmizením přepínače z dohledu při rozbalení boxů.

2. Líbilo by se mu, kdyby celé řádky v tabulce logů byly klikatelné a dalo se pomocí nich filtrovat box spojení, který patří k řádku, na který uživatel klikl.
3. Potřeboval by delší čas na notifikace, zejména pak při odstraňování, kde může provést vrácení akce. Při prodloužení času notifikace by nevyžadoval potvrzení formou dialogového okna, ale jeho přítomnost by mu také nevadila. Jako případný nadstandardní prvek navrhnul existenci koše, do kterého by se po jistou dobu odstraněné boxy ukládaly.
4. Objevil problém při zadávání kupónu. Chtěl místo kliknutí na tlačítko provést potvrzení kupónu tlačítkem „Enter“ a došlo k opětovnému načtení stránky. Tento problém bude potřeba opravit, protože jeho postup je určitě očekávané chování uživatele.
5. Za užitečné by považoval notifikace o provedených synchronizacích. Nedokáže si však představit, že by mu chodily na email každou hodinu. Osobně by preferoval takovéto notifikace dostávat jednou denně někdy ke konci dne.
6. Preferoval by, kdyby se hodina u volby času v rozvrhu synchronizace znefunkčnila, pokud uživatel zvolí synchronizace každou hodinu.
7. Působilo na něj matoucím dojmem, že tlačítko „Upgrade“ v stručném uživatelském profilu naviguje na změnu členství, ne pouze na navýšení. Preferoval by zde odlišný název.
8. Informaci o počtu dostupných okamžitých synchronizací na stránce „Spojení“ by uvítal.
9. Tlačítko pro ukončení předplatného pro něj bylo redundantní a dávalo by mu větší smysl v případě, že by bylo možné pracovat s aplikací i bez členství. Přemýšlel zda na ukončení předplatného použít toto tlačítko či zvolit změnu členství na Hobby s jedním aktivním spojením, obdobně jako Ing. Malec. Bez nápověd a problémů však rychle zvolil jednu z možností, z nichž obě vedou k cíli, a členství ukončil. Nepovažuji tedy tuto redundanci za podstatný problém.
10. Stejně jako na Ing. Malce na něj nepůsobil dobře 12hodinový formát času s volbou AM/PM při změně rozvrhu synchronizace. Líbilo by se mu, kdyby mohl čas zadávat přes klávesnici ručně. Měl také stejný problém s fungováním komponenty pro nákup okamžitých synchronizací a také mu chyběla notifikace či tlačítko na uložení při změně časové zóny, notifikací a rozvrhu synchronizace spojení. Pokoušel se také provádět volbu nového členství kliknutím na obrázek (přislušný čtverec) a projevil zájem o tuto funcionalitu.
11. Stejně jako u Ing. Glazara se projevil problém s nevhodně nastaveným časovačem, který umožňuje automatické rozbalování konfigurace nástrojů po přidání boxu spojení.

6.2.2.4 Martin Dvořák

Kontext: Martin je student na FIT ČVUT. Aplikaci před testováním nikdy nepoužíval, ale měl o ní povědomí od kolegů ze společnosti Jagu s. r. o.

Pozitivní závěry: Rozhodně pozitivní informací je fakt, že i u druhého uživatele, který neměl bližší znalost s aplikací Timer2Ticket, se neprojevily žádné zásadní potíže při orientaci v celé aplikaci. Pohyboval se plynule a sám od sebe popisoval prvky, se kterými se setkával, takže bylo zřejmé, že drtivou většinu chápe naprosto správně. Ocenil, že:

- Může mít nakonfigurováno více spojení a střídat ty, které jsou aktivní. Našel dokonce konkrétní případ užití této situace – pokud by měl například někdo dvě zaměstnání a první polovinu týdne pracoval v jednom zaměstnání a druhou půlku týdne v druhém.
- Má možnost provést vrácení odstraněného boxu spojení. Přiznal, že když je k dispozici vracení, nemá pro něj význam mít potvrzovací dialog.

- Zejména při konfiguraci nástrojů má řadu tooltipů pomáhajících v nalezení potřebných údajů.
- Má pro něj nejpodstatnější informace o daných spojeních dostupné i bez rozbalení boxů.
- Je dostupná možnost registrace a přihlášení pomocí služby Google, kterou často používá.

Mezi další pozitiva vyplynulá z tohoto testování lze řadit:

- Chválil grafiku vytvořeného uživatelského rozhraní a příjemné nastavování rozvrhu synchronizace spojení.
- Žádné dodatečné notifikace nad rámec uvedených by si nastavovat nepotřeboval. Využil by pravděpodobně jen notifikace o chybách, protože už tak mu chodí mnoho emailů.
- Stručný uživatelský profil obsahuje informace, které by tam neočekával, ale byly pro něj užitečné.
- Tabulku se zpracovanými objednávkami označil za užitečnou.
- Stejně jako předchozí si byl z vytvořených členství schopen vybrat takové, které by vyhovovalo jeho potřebám. Uvedl, že by pravděpodobně začal na vyzkoušení s Hobby členstvím. Jeho potřebám by však později nejvíce vyhovovalo Junior členství, protože by chtěl synchronizace alespoň jednou denně. Senior členství by pro něj bylo zbytečné, protože pro jeho potřeby poskytuje až příliš velké množství aktivních spojení.

Objevené nedostatky a návrhy na změny:

1. Líbilo by se mu, kdyby měl při prvním příchodu do aplikace předvoleno nějaké členství. Očekával by Hobby členství. Zároveň očekával, že tlačítka, která jsou v této fázi nepoužitelná, nejen že nebudou funkční, ale ani na ně nepůjde kliknout.
2. Přišel s návrhem prvku, který sám označil za nadstandardní. Uživateli by bylo hezké při prvním vstupu do aplikace udělat jakýsi prvotní průvod aplikací.
3. Líbilo by se mu kdyby se země do formuláře o platebních informacích mohla vyplňovat výběrem z rozbalovatelného seznamu. Upozornil mě také na nutnost validovat formát DIČ.
4. Bylo by pro něj jasnější, kdyby se hodina u volby času v rozvrhu synchronizace znefunkčnila, pokud uživatel zvolí synchronizace každou hodinu. Zároveň přišel s návrhem, že by součástí tooltipu u synchronizací každou hodinu mohl být odkaz k nákupu Senior členství.
5. U přepínače aktivity spojení by ocenil notifikaci po provedení změny. Zároveň by mu vyhovovalo, kdyby přepínač byl i součástí rozbaleného boxu, aby k němu neztratil přístup po rozbalení boxu a posunu obrazovky. Případně by se mu líbilo, kdyby se dalo deaktivovat spojení zrušením všech dnů v týdnu.
6. Nevyužíval by často změny pořadí boxů, ale líbilo by se mu mít jako první taková spojení, která jsou aktivní.
7. Nebyl si jistý tím, co znamenají konfigurační objekty. Ocenil by tam vysvětlivku, že jde například o projekty a úkoly.
8. Když prováděl okamžitou synchronizaci, očekával, že bude nějak informován, že synchronizace doběhla. Nevšiml si, že po jejím doběhnutí se provedla změna času v informaci o poslední provedené synchronizaci, a zvažoval, zda vůbec k synchronizaci došlo. Po provedení této akce jsem ho informoval o skutečnosti změny této informace a požádal ho, aby se zamyslel nad tím, co by mu pomohlo, protože synchronizace může trvat déle a notifikace by mohla přijít později, kdy by už kontext jeho práce mohl být úplně jiný. Přišel zde s návrhem změnit informaci o čase poslední sychronizace v záhlaví boxu na načítací kolečko, a to do doby než se dokončí prováděná synchronizace. Po dokončení se teprve změní na text. Stejné načítací kolečko by ocenil i při první synchronizaci prováděné automaticky po uložení spojení.

9. Jako první z testerů si všiml, že pokud změní nástroj, který má být synchronizován, a změnu uloží, provede se okamžitě první synchronizace, což označil jako možnost, jak kdykoliv provést okamžitou synchronizaci. Tato možnost je samozřejmě validní, ale vzhledem k tomu, že aplikace má zjednodušit práci, tento proces by byl pro uživatele kontraproduktivní, tedy se nepředpokládá. Dalo by se mu však zabránit tím, že by se pamatovaly použité dvojice spojení i po jejich odstranění a následně se kontrolovalo, zda uživatel vytváří novou dvojici či již dříve používanou a na základě toho se rozhodovat, zda spustit okamžitou první synchronizaci či nikoliv. Nabízí se zde ještě situace, kdy by uživatel příliš často měnil rozvrh synchronizace, aby docílil častějších sychronizací, než v základu jeho členství nabízí. Opět by to bylo proti kompletní myšlence této aplikace, avšak v případě nutnosti by se tomu pravděpodobně dalo zabránit například omezeným počtem změn rozvrhu synchronizace daného spojení za týden.
10. Tlačítko umožňující načtení posledního použitého plánu, který měl před přestupem na základní Hobby plán, nepovažoval za užitečné s ohledem na to, že výběr plánů není tak široký.
11. Líbilo by se mu, kdyby byl někde na stránce „Členství“ informován o tom, že současné členství již jen dobíhá. Nevšiml si malého tooltipu a doporučil mi, abych text s informací o dobíhajícím členství barevně odlišil od původního, kdy členství nedobíhá.
12. Přišel s návrhem, aby členství, u kterého bylo ukončeno předplácení a dobíhá, bylo umožněno kdykoliv provést znovu volbu změny a získat okamžitě Hobby členství s kompenzací v podobě okamžitých synchronizací.
13. Jak jsem již očekával, objevily se u něj i mnohé nedostatky, které vyplynuly již od předchozích testerů. Měl potíž s určením povinných platebních informací při nákupu prvního členství. Očekával, že při změně nastavení notifikací, časového pásma a rozvrhu synchronizace bude přímo notifikován o provedené změně či bude mít dostupné tlačítko k uložení změn. Očekával také, že bude mít na stránce „Členství“ předvolené současné členství a změnu bude moci provádět kliknutím na obrázek. Také projevil zájem o informaci o počtu dostupných okamžitých synchronizací na stránce „Spojení“ i možnost editovat souhrn provedené objednávky pomocí upravovacích tlačítek přímo bez nutnosti vracet se k jednotlivým boxům na stránce „Členství“. Setkal se také s problémem při potvrzení kupónu tlačítkem „Enter“ či problémem násobku dvaceti v případě komponenty nákupu okamžitých synchronizací.

6.2.3 Shrnující poznatky z proběhlého testování

Testování použitelnosti se ukázalo být v současném stavu velice prospěšné. Ujistilo mě, že vytvořený návrh a dle něj implementovaný prototyp neobsahují žádné zásadní nedostatky ani mezery, které by se těžko opravovaly, a je pro uživatele dobře rozčleněný. Za velké pozitivum považuji, že uživatelé se v aplikaci poměrně rychle zorientovali a bez větších obtíží pohybovali. Kladně hodnotím i fakt, že každý si byl schopen zvolit členství, které by odpovídalo jeho potřebám.

Zároveň jsem získal řadu podnětů ke změnám, které by mě bez tohoto testování těžko napadly a přitom mohou výrazně pomoci uživatelům v používání aplikace. Veškeré tyto změny, které by bylo vhodné provést a byly popsány u jednotlivých testerů v sekci 6.2.2, jsem evidoval jako úkoly v projektovém systému Redmine, který je využíván v rámci softwarových týmových projektů na FIT ČVUT a je dostupný na adrese `https://dbs.fit.cvut.cz/redmine/projects/timer2ticket`.

Úkoly reagující na vzniklé podněty jsem nevytvořil pouze ke třem z nich. Zatím jsem se rozhodl ponechat odstraňování boxů bez dialogového okna. Souhlasím však s nutností prodloužení časového limitu pro možnost vrácení provedené akce. Zároveň jsem se prozatím rozhodl ponechat změnu hesla na stránce „Nastavení“ a rozhodl jsem se nerealizovat návrh na existenci koše ze třetího bodu možných změn od Ondřeje. Do kontextu této aplikace mi tento návrh příliš nesedí.

Kapitola 7
Budoucí rozvoj

V této kapitole popíšu optimální směřování projektu v budoucnu a detailněji vysvětlím zejména první kroky, které bude vhodné provést v návaznosti na tuto bakalářkou práci. Stručně však nastíním i další oblasti rozvoje této aplikace a na závěr popíšu optimální navázání na práce započaté na propagačním webu.

7.1 První kroky dalšího rozvoje aplikace

Abych zajistil hladký budoucí rozvoj, jak jsem již zmínil v rámci shrnutí poznatků z testování v sekci 6.2.3, v reakci na objevené nedostatky a návrhy na vylepšení jsem k nim vytvořil úkoly v projektovém systému Redmine, který jsem po celou dobu vývoje nového frontendu využíval a je dostupný na adrese `https://dbs.fit.cvut.cz/redmine/projects/timer2ticket`. Další rozvoj vzniklého prototypu by se měl určitě nejprve ubírat tímto směrem.

Jedná se v součtu o 48 úkolů, z nichž ukázka je k nalezení v příloze D na obrázcích D.1 a D.2. Nemá cenu, abych zde jednotlivé úkoly postupně rozebíral, neboť všechny obsahují detailní popis objevených nedostatků a případných návrhů na změny. V některých případech, kde se objevilo více možných řešení, jsem tam popsal všechny navržené řešení a k tomu uvedl svůj názor, ke kterému bych se přiklonil. Kromě detailního popisu nalezených problémů a návrhů na změny, kterým jsem úkoly opatřil, jsem je také rozdělil dle priority. Prioritu jsem nastavoval zejména s ohledem na četnost daného nedostatku v provedeném testování či dle toho, jak velká poptávka po daných změnách byla. Dle priority je tedy potřebné se jim začít věnovat.

Počítám s tím, že v budoucnu budu sám na projektu dále participovat. Pro rychlejší posun však věřím, že by bylo vhodné využít například některý z týmů v předmětech SP1 či SP2, kterým jsem ochotný být oporou v dalším rozvoji projektu. Neboť se v těchto předmětech studenti často seznamují se svým prvním frameworkem, předpokládám, že právě volba v současné době atraktivního frontendového frameworku Vue.js a grafické knihovny Vuetify mohou studenty zaujmout a podpořit tak další rozvoj. Abych také další práci zjednodušil, kromě nastavení priority jsem k jednotlivým úkolům přidal svůj odhad náročnosti. Původně užívaný Redmine pole pro odhad náročnosti u úkolů neobsahoval, musel jsem ho v nastavení přidat. Díky nastavení tohoto odhadu se budou úkoly případně snadněji rozdělovat mezi studenty, kteří mají již s frameworkem nějakou zkušenost a chtějí si zkusit něco pokročilejšího, a studenty, kteří jsou méně zkušení. Některé z úkolů jsou podobného rázu či se týkají blízkých prvků, vytvořil jsem proto mezi těmito úkoly tzv. souvislosti, díky čemuž jsou úkoly propojeny a pohyb mezi nimi je snadnější.

Jak jsem již popsal v sekci 6.2 o testování použitelnosti, tento typ testování je vhodné provádět nejlépe ve třech iteracích. Po zapracování nedostatků a změn z evidovaných úkolů proto bude vhodné pokračovat provedením další iterace testování použitelnosti, zjistit posun a další případné nedostatky či návrhy na změny k zapracování.

7.2 Další oblasti rozvoje

V této části stručně nastíním další z možných oblastí rozvoje tohoto projektu. Nejprve zmíním možný rozvoj pro nekomerční použití, následně stručně uvedu oblast napojení na platební bránu a na backend. Nakonec pak pro úplnost zmíním i možnost přidání dalších nástrojů k synchronizaci. I k těmto jednotlivým oblastem jsem v rámci zmíněného projektového nástroje Redmine vytvořil úkoly, již však ne tak detailní.

7.2.1 Rozvoj řešení pro nekomerční použití

Prototyp implementovaný na základě vytvořeného návrhu obsahuje připravená jednotlivá členství tak, jak budou moci fungovat v komerční variantě provozované společností Jagu s. r. o. Pro účely nekomerčního používání a nasazení jinými subjekty je však potřeba odstínit placenou vrstvu, tedy vše zprovoznit bez omezení zdarma.

V zásadě by se mělo jednat pouze o znefunkčnění některých prvků, protože v nekomerčním stavu by měli mít uživatelé možnost mít neomezený počet aktivních spojení, provádět okamžité synchronizace kdykoliv a nastavovat rozvrh synchronizace vždy bez omezení, na rozdíl od stavu komerčního se členstvími. V nekomerčním režimu proto nebude mít význam stránka „Členství“ ani stránka „Platby“. S ohledem na to bude v takovou chvíli potřeba na všech zbývajících stránkách jednoduše znefunkčnit a zneviditelnit veškeré odkazy vedoucí na tyto stránky. Znefunkčnit a zneviditelnit bude potřeba také veškeré informace související se členstvím uživatele, omezeným počtem okamžitých synchronizací, kupóny a pochopitelně i stránku s volbou prvotního členství. Jak jsem již zmínil výše v úvodu k sekci 7.2, i k této oblasti rozvoje jsem vytvořil úkol, kam jsem nutné změny zaznamenal.

7.2.2 Napojení platební brány

V rámci této práce jsem provedl mimo jiné i volbu vhodné platební brány pro tuto aplikaci, kterou jsem nakonec zvolil zejména s ohledem na cenu, poplatky a potřebu mezinárodního zaměření. Do vytvořeného prototypu jsem ji však zatím neintegroval, ve finální realizaci však bude integraci platební brány potřeba samozřejmě provést.

Na základě uvedených kritérií vzešla z mé analýzy nejlépe platební brána Stripe. Při dokončování této práce jsem byl však v rámci testování jedním z testerů, který tuto platební bránu používá, upozorněn, že před napojením ještě bude potřeba prověřit, zda bude poskytovat vhodnou podporu pro náhradu zbytku členství za okamžité synchronizace a propojení jednorázové a měsíční platby, protože s takovým použitím se u ní zatím nesetkal. V krajním případě se může stát, že bude potřeba implementovat vlastní řešení této problematiky.

7.2.3 Napojení na backend

Další z oblastí budoucího rozvoje se týká napojení aplikace na backend. Současný prototyp nebyl napojen na backend, neboť se pro implementaci nového (respektive modifikaci starého) zatím nikdo nenašel a nešlo využít původní řešení, protože v této práci vytvořené frontendové rozšíření, které má v budoucnu nahradit původního webového klienta vytvořeného v rámci diplomové práce Ing. Víta Štefana, není pouze refaktoringem původního, avšak přináší opravdu velké množství nových funkcionalit a zejména práci se členstvími, se kterými původní implementace samozřejmě nepočítala.

Pro přechod od prototypu k výsledné realizaci proto bude dále také nutné vydefinovat potřebné endpointy, zajistit potřebný backend a následně na něj frontendové řešení vytvořené v této práci napojit.

7.2.4 Přidání dalších nástrojů k synchronizaci

Neboť tato práce je k začátku května 2022 nejaktuálnější prací týkající se nástroje Timer2Ticket, pro úplnost zmíním i možnost přidání dalších nástrojů k synchronizaci. Řešení synchronizace vytvořené Ing. Vítem Štefanem v rámci jeho diplomové práce cílilo zejména na obecnost, tedy aby se snadno daly přidat další projektové nástroje a nástroje pro měření času, které budou moci být synchronizovány [2]. S ohledem na prototyp nového uživatelského rozhraní vytvořený v této práci, který již byl vytvářen tak, aby mohla probíhat synchronizace i mezi větším množstvím dvojic nástrojů, dává ještě větší smysl přidat další nástroje po bok těch již zprovozněných, tedy nástrojů Redmine a Toggl Track.

7.3 Dokončení implementace propagačního webu

V této práci jsem ve spolupráci s týmem z předmětu SP2 provedl sběr požadavků na propagační web aplikace Timer2Ticket a jeho návrh. Provedl jsem také analýzu vhodné technologie na tvorbu propagačního webu, z níž vzešel nejlépe October CMS, jak byl již popsáno v sekci 3.4.3. Následně mi společnost Jagu s. r. o. pomohla se zajištěním potřebné licence a nasazením Octoberu na adresu `https://www.timer2ticket.com`, na kterou dochází také k přesměrování z adresy `https://www.timer2ticket.cz`.

V závěru předmětu SP2 ještě softwarový tým, konkrétně Benedek Molnár, začal pracovat na implementaci ve zvoleném nástroji. Naimplementované části mi postupně ukazoval a já validoval shodu s vytvořeným návrhem, nicméně došlo k vytvoření pouze opravdového základu, protože času již moc nezbývalo. Z částí webu popsaných v kapitole 4.3.2 došlo k vytvoření základu prvních tří – Upoutání pozornosti, Funkce aplikace a Členství. Video uvedené v části sloužící k upoutání pozornosti bylo zatím vloženo pouze ukázkové a do části s funkcemi aplikace zatím nebyl vložen obrázek pro živou ukázku z hotové aplikace. Myslím si, že by mohlo být užitečné po dokončení samotné aplikace vytvořit i sadu videí, která budou sloužit jako tutoriál pro snadnější seznámení se s touto aplikací (kromě videa, které bylo zmíněno již v návrhu a bude sloužit ke shrnutí základních principů aplikace a upoutání pozornosti).

Zmíněné části je potřeba v budoucím rozvoji doplnit. Zároveň je potřeba vytvořit i zbylé tři sekce, které jsem v návrhu popsal, a doplnit do záhlaví webu vytvořené logo aplikace. Nesmí se zapomenout ani na přidání překladů, zajištění responzivity celého propagačního webu a jeho napojení na samotnou aplikaci, do níž bude probíhat přesměrování, jak bylo také v návrhu popsáno. Aby bylo snadné navázat i v této oblasti, vytvořil jsem ve zmíněném systému Redmine opět úkoly, které pomohou s orientací v pracích, které jsem zde uvedl a kterými je potřeba navázat.

Kapitola 8
Závěr

Cílem této práce bylo navrhnout nové uživatelsky přivětivé rozhraní již existující synchronizační služby Timer2Ticket a následně implementovat prototyp navrženého řešení. Při návrhu bylo třeba dbát na vhodnou volbu členských plánů, díky nimž půjde službu monetizovat, a zároveň na fakt, aby bylo nové uživatelské rozhraní připraveno na přidání dalších nástrojů k synchronizaci, a ta mohla probíhat mezi více než jednou dvojicí.

Abych byl schopen provést kvalitní návrh, provedl jsem analýzu řešení vytvořeného Ing. Vítem Štefanem, při čemž jsem se zaměřil zejména na webového klienta, kterého má nový frontend v budoucnu nahradit. Na základě této analýzy jsem identifikoval jeho nedostatky. S pomocí studentů předmětu SP2 a díky řadě diskuzí se svým zadavatelem, který je zároveň provozovatelem současného Timer2Ticketu, jsem stanovil funkční i nefunkční požadavky na nový systém.

Provedl jsem také důkladnou analýzu nástrojů, které jsou uživatelům služby Timer2Ticket blízké a uvědomil si, že pokud má být aplikace úspěšná a výdělečná, kromě nového frontendu bude pro ni potřeba vytvořit i propagační web, který ji zpopularizuje. Zejména v části propagačního webu jsem získal podporu od softwarového týmu z předmětu SP2, který mi pomohl s návrhem a začátkem jeho implementace v nástroji October CMS. Díky tomu jsem získal také cennou zkušenost s teamleadingem.

Výsledky mé analýzy potencionálního zákazníka této aplikace poukázaly na to, že největší podíl uživatelů nástrojů, které jsou Timer2Ticketu blízké, pochází z odvětví vývoje počítačového softwaru a zejména z USA. Toto zjištění mě ovlivnilo zejména při výběru vhodné platební brány, kdy její světová rozšířenost pro mě byla jedním z hlavních faktorů při rozhodování.

Na základě sesbíraných požadavků jsem zpracoval návrh nového uživatelského rozhraní, který jsem rozdělil do tří základních logických celků (*Spojení*, *Logy* a *Profil*). Při návrhu celku *Spojení* jsem kladl důraz na možnost synchronizovat současně více dvojic různých nástrojů, o což jsem opřel návrh jednotlivých členských plánů. Vzniklé tři plány jsem rozdělil na základě počtu umožněných aktivních spojení, možné doby synchronizace a ceny, kterou jsem určil zejména na základě přepočtů z ušetřeného času. Pro zjednodušení návrhu jsem využil nástrojů Balsamiq a Figma, určených pro tvorbu wireframů.

Při realizaci prototypu jsem se držel obdobného rozdělení jako při návrhu. Prototyp jsem vytvořil s pomocí frontendového frameworku Vue.js, který jsem si vybral na základě důkladné analýzy frontendových frameworků, a grafické knihovny Vuetify. Tu jsem na tomto projektu využíval poprvé a byla mi velice přínosná zejména širokým množstvím komponent, které pro tvorbu uživatelského rozhraní poskytuje.

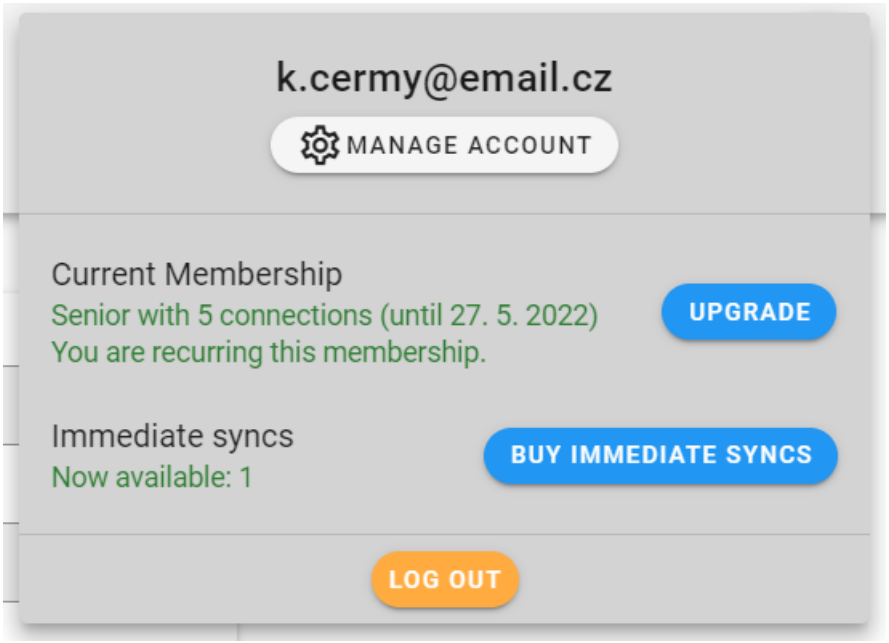
Vytvořený prototyp se všemi požadovanými funkcionalitami jsem podrobil uživatelskému testování. To mi pomohli uskutečnit uživatelé současné služby Timer2Ticket z firmy Jagu s. r. o. a studenti FIT ČVUT, kteří s nástrojem neměli žádnou zkušenost. Na základě tohoto testování jsem získal cennou zpětnou vazbu na vytvořený prototyp a nápady na možné změny do budoucna.

Na závěr jsem sepsal kroky, které by bylo vhodné provést při dalším rozvoji tohoto prototypu. Kromě napojení na backend bude potřeba provést také integraci platební brány, která byla v této práci zvolena. Vzhledem k tomu, že bych se rád na budoucím rozvoji tohoto projektu podílel a věřím, že o něj může být zájem i mezi jinými studenty, v rámci systému Redmine, který jsem používal, jsem pro účely dalšího rozvoje vytvořil konkrétní úkoly, kterými lze dále pokračovat, a to v rámci prací na propagačním webu i samotné aplikaci.

Příloha A

Ukázky změn provedených po testování

Obrázek A.1 Stručný rozbalovatelný uživatelský profil v pravém horním rohu aplikace



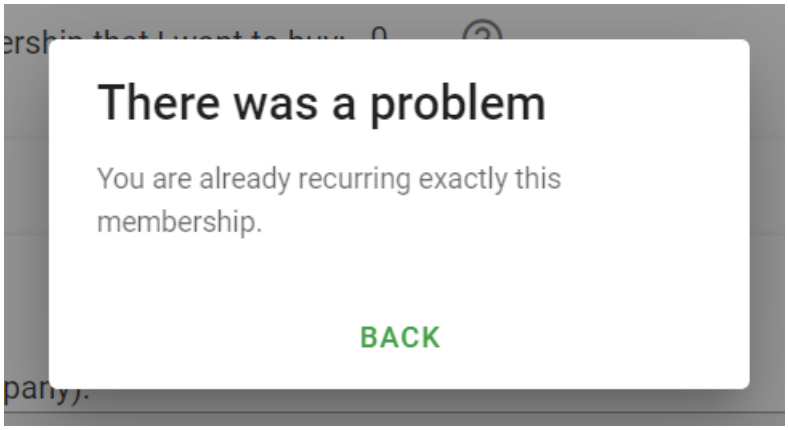
Obrázek A.2 Upravený design jednotlivých členství v rámci boxu se změnou členství a vybraným Senior členstvím

Obrázek A.3 Souhrn informací o nákupu s přidanými platebními údaji a tlačítky pro snadnou úpravu – se zvolenou změnou členství a nákupem okamžitých synchronizaci z obrázku 4.12

Obrázek A.4 Aktivní spojení s přepínačem aktivity přesunutým mimo záhlaví hlavního boxu

Obrázek A.5 Tabulka s logy vyfiltrovaná pro třetí spojení s rozbaleným ukázkovým detailem prvního záznamu

Obrázek A.6 Upravená chybová hláška informující uživatele, že nelze znovu provést nákup členství, které již má zakoupeno



Příloha B

Testovací scénář

Představte si, že z propagačního webu, který Vás informoval o možnostech Timer2Ticketu jste se dostali do aplikace, ve které jste provedli registraci (informování o základních prvcích provedu vzhledem k rozpracovanosti propagačního webu slovně s obrazovým doplněním informací jednotlivých členství přímo z propagačního webu). Nyní se nacházíte v aplikaci po prvním přihlášení.

B.1 Výběr prvního členství

1. Rozhodli jste se využívat Junior členství, proveďte jeho výběr a pokračujte.
2. Vyplňte všechny potřebné údaje pro provedení platby a pokračujte.

Po provedení úkonů uživatele budu informovat o mezikroku s platební bránou, který bude ve finální realizaci, ale není součástí prototypu.

B.2 Zprovoznění synchronizace pro jednu dvojici nástrojů

1. Přidejte spojení, abyste mohli provést konfiguraci nástrojů k synchronizaci.
2. Proveďte nastavení obou nástrojů, které chcete vzájemně synchronizovat. Pro účely testování zvolte jejich nastavení libovolně (nezáleží na vyplněném obsahu polí, musí však být vyplněna).
3. Jakmile jste s konfigurací nástrojů spokojeni a je kompletní, zprovozněte mezi nimi synchronizaci.
4. Uvědomili jste si, že jste v Toggl Tracku zvolili špatný workspace. Proveďte jeho změnu na libovolný jiný z nabídky a zajistěte opětovnou aktivitu nové synchronizační dvojice.
5. Nalezněte informaci o čase příští synchronizace.
6. Nové úkoly v práci obvykle obdržíte kolem osmé hodiny ráno. Změnte proto dobu synchronizace pro konfigurační objekty na tento čas.
7. Rozhodli jste se také, že nechcete provádět žádné synchronizace o víkendu. Upravte vhodně nastavení doby.

Po provedení uživatele upozorním na fakt, že pro další se to dělá obdobně. S ohledem na další části scénáře konfiguraci dalších spojení provádět nebudeme či ji na žádost testera rychle připravím sám (pro lepší názornost). Případně na to nechám uživateli prostor po dokončení scénářů či v rámci dotazníku po testování – aby se s aplikací mohl seznámit lépe.

B.3 Upgrade členství

1. Právě jste okolnostmi v práci donuceni provést okamžitou synchronizaci konfiguračních objektů a časových záznamů. Pokuste se o to.
2. Zjistili jste, že na druhou akci nemáte dostatek okamžitých synchronizací. Nalezněte prostor, kde se dají nakoupit.
3. Zvolte nákup libovolného počtu okamžitých synchronizací.
4. Všimli jste si možnosti změny členství a napadlo Vás, že by mohlo být užitečné využít Senior plán díky tomu, že nabízí synchronizace každou hodinu. Proveďtě jeho volbu a až s ní budete spokojeni, pokračujte.
5. Prohlédněte si Vaši objednávku.
6. Cena se Vám zdá příznivá, rozhodli jste se proto, že nakoupíte větší množství okamžitých synchronizací rovnou. Proveďte tuto změnu a následně se vraťte k sumarizaci Vaší objednávky.
7. Proveďte zaplacení.
8. Nalezněte informaci o současném členství a počtu dostupných okamžitých synchronizací.

Po provedení úkonů uživatele budu informovat o mezikroku s platební bránou, který bude ve finální realizaci, ale není součástí prototypu.

B.4 Akce spojené se spojením

1. Vraťte se ke spojení, u kterého jste chtěli původně provádět okamžitou synchronizaci a uskutečněte ji.
2. Spojení deaktivujte (již nechcete provádět pravidelné synchronizace).
3. Představte si, že uběhl nějaký čas a rozhodli jste se, že chcete získat logy o tom, jak probíhaly vaše synchronizace pro spojení nakonfigurované v prvním boxu. Nalezněte odpovídající záznamy v tabulce.
4. Zajímají Vás především záznamy o synchronizacích času. Nalezněte pouze ty a seřaďte si je dle doby dokončení vzestupně.
5. O prvním nalezeném záznamu zjistěte detailní popis informující o provedené synchronizaci.
6. Vraťte se k nakonfigurovanému spojení a odstraňte ho, již ho nechcete používat. Představte si, že jste odstranili špatné spojení a akci vraťte.
7. U spojení, které jste chtěli odstranit změňte dobu synchronizace konfiguračních objektů tak, aby probíhala každou hodinu.
8. Spojení opět aktivujte a nalezněte čas příští synchronizace konfiguračních objektů.

B.5 Ukončení předplatného a návrat k původnímu

1. Přehodnotili jste situaci a rozhodli se, že Vám nakonec bude stačit základní Hobby členství s jedním spojením. Ukončete předplatné svého současného členství.
2. Ukončete prostřednictvím okamžitých synchronizací.
3. Uběhl nějaký čas a změna se Vám nezamlouvá. Zakupte Váš původní plán a získejte členství, které jste měli naposledy.

B.6 Změna nastavení

1. Přestěhovali jste se na Americkou Samou a nastavená synchronizace se Vám začala provádět v jiný čas, než jste chtěli. Nastavte pro svůj účet časový posun.
2. Vzhledem k tomu, že se Vám začaly množit problémy s probíhajícími synchronizacemi, chcete o nich být informováni, nastavte si příslušné notifikace.
3. Aplikace se Vám i přes to líbí. Za donaci provedenou na propagačním webu jste získali kupón IMMEDIATE-SYNCS-FREE a rozhodli jste se je získat. Použijte ho, abyste je dostali.

B.7 Opuštění aplikace

1. Najděte možnost odhlásit se.
2. Odhlašte se.

Příloha C

Dotazník po testování

1. Co Vám na novém Timer2Ticketu nevyhovuje či úplně chybí?
2. Co se Vám na něm líbí?
3. Jak se Vám líbí rozdělení jednotlivých členských plánů? Dokázali byste si z nich vybrat nějaký, který by Vám vyhovoval? Případně jaký?
4. Zdá se Vám užitečný stručný uživatelský profil v pravém horním rohu?
5. Uvítali byste přítomnost informace o okamžitých synchronizacích přímo na stránce Spojení?
6. Dává Vám smysl nastavení času synchronizace v případě synchronizování každou hodinu? Pokud ne, pomáhá v tom tooltip u nastavení každohodinové synchronizace?
7. Využili byste možnost měnit pořadí jednotlivých boxů spojení?
8. Uvítali byste ještě nějaké typy notifikací, které by se daly nastavit?
9. Přijde Vám užitečná tabulka zpracovaných objednávek nebo Vám stačí zasílání těchto informací na email?

Příloha D

Ukázka úkolů vzniklých pro budoucí rozvoj

Obrázek D.1 Ukázka části seznamu úkolů, které byly vytvořeny pro budoucí rozvoj aplikace Timer2Ticket

Požadavek #6060: Varianta formátu času dle locale prohlížeče
Požadavek #6061: Přidání notifikace o uložení změny rozvrhu synchronizace
Požadavek #6062: Úprava tlačítek z boxu spojení
Požadavek #6063: Změna tooltipu pro tlačítka okamžité synchronizace
Požadavek #6064: Změna nákupu okamžitých synchronizací
Požadavek #6065: Přidání notifikace o uložení změn (notifikace, časové pásmo) v nastavení profilu
Požadavek #6066: Dynamické změny labelu u tlačítka See summary
Požadavek #6067: Úprava objednávky v přehledu její sumarizace
Požadavek #6068: Automatické vyplnění existujících boxů spojení
Požadavek #6069: Rozbalení detailu po kliknutí kamkoliv na řádek
Požadavek #6070: Změna ikony pro zobrazení statusu záznamu synchronizace
Požadavek #6071: Odstranění okrajů filtračních boxů v tabulce logů (na mobilu)
Požadavek #6072: Změna labelu tlačítka pro načtení posledního členského plánu
Požadavek #6073: Filtrační řádek do tabulky zpracovaných objednávek
Požadavek #6074: Změna barvy tlačítek, když jsou aktivní (stránka nastavení a platební informace)
Požadavek #6075: Zvýraznění názvů sloupců u možností členství (na mobilu)
Požadavek #6076: Změnit rozložení textových informací v boxu rozvrhu času spojení (na mobilu)
Požadavek #6077: Pojmenování boxu spojení

Obrázek D.2 Ukázka konkrétního vytvořeného úkolu s nastavenou prioritou, odhadem náročnosti, popisem i provázaností na ostatní úkoly

Bibliografie

1. TOGGL. *Toggl Track: Jira Time Tracking* [online]. 2022 [cit. 2022-03-11]. Dostupné z: https://toggl.com/track/jira-time-tracking/.
2. ŠTEFAN, Vít. *Synchronizační middleware mezi projektovým systémem a aplikací na sledování času stráveného na projektech* [online]. Praha, 2021 [cit. 2022-03-15]. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/94628/F8-DP-2021-Stefan-Vit-dp_stefan_vit_2021.pdf. Diplomová práce. České vysoké učení technické v Praze.
3. JAGU, s. r. o. *Timer2Ticket* [soft.]. 2021 [cit. 2022-03-12]. Dostupné z: https://app.timer2ticket.com/.
4. FLOWII, s. r. o. *Flowii: Nejlepší systémy na projektové řízení* [online]. 2020-10-27 [cit. 2022-03-15]. Dostupné z: https://www.flowii.com/cz/blog/top-21-nejlepsi-systemy-na-projektove-rizeni.
5. SHOPTET, a. s. *Shoptet: Open Source* [online]. 2022 [cit. 2022-03-15]. Dostupné z: https://www.shoptet.cz/slovnik-pojmu/open-source/.
6. LANG, Jean-Philippe et. al. *Redmine* [online]. 2014 [cit. 2022-03-15]. Dostupné z: https://redmine.org/.
7. ATLASSIAN. *Jira Software* [online]. 2022 [cit. 2022-03-15]. Dostupné z: https://www.atlassian.com/software/jira.
8. ATLASSIAN. *Jira Software: Plans and pricing* [online]. 2022 [cit. 2022-04-01]. Dostupné z: https://www.atlassian.com/software/jira/pricing.
9. NUTCACHE. *Nutcache* [online]. 2022 [cit. 2022-03-15]. Dostupné z: https://www.nutcache.com/.
10. NUTCACHE. *Nutcache: Compare plans* [online]. 2022 [cit. 2022-03-15]. Dostupné z: https://www.nutcache.com/pricing/.
11. CLICKUP. *ClickUp: One app to replace them all* [online]. 2022 [cit. 2022-03-15]. Dostupné z: https://clickup.com/.
12. CLICKUP. *ClickUp: Pricing* [online]. 2022 [cit. 2022-03-15]. Dostupné z: https://clickup.com/pricing.
13. TOGGL. *Toggl Track* [online]. 2022 [cit. 2022-03-13]. Dostupné z: https://toggl.com/track/.
14. TOGGL. *Toggl Track: Pricing Plans* [online]. 2022 [cit. 2022-03-13]. Dostupné z: https://toggl.com/track/pricing/.
15. COING, Inc. *Clockify* [online]. 2022 [cit. 2022-03-15]. Dostupné z: https://clockify.me/.

16. COING, Inc. *Clockify: Extra features* [online]. 2022 [cit. 2022-03-15]. Dostupné z: https://clockify.me/extra-features.
17. HARVEST. *Harvest* [online]. 2022 [cit. 2022-03-15]. Dostupné z: https://www.getharvest.com/.
18. HARVEST. *Harvest: Simple, flexible pricing* [online]. 2022 [cit. 2022-03-15]. Dostupné z: https://www.getharvest.com/pricing.
19. SLACK TECHNOLOGIES, LLC. *Slack* [online]. 2022 [cit. 2022-03-17]. Dostupné z: https://slack.com/.
20. DISCORD. *Discord* [online]. 2022 [cit. 2022-03-17]. Dostupné z: https://discord.com/.
21. HLAVATÝ, Martin; PETŘÍK, Michal. *Projektové řízení a teamleading* [online]. 2021 [cit. 2022-03-17]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/429308/course/section/70554/2021_2022/11_PMTL.pdf.
22. MLEJNEK, Jiří. *Analýza a sběr požadavků* [online]. 2022 [cit. 2022-04-02]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/506241/mod_resource/content/7/03.prednaska.pdf.
23. ENLYFT. *Enlyft: Accelerate your customer acquisition* [online]. 2022 [cit. 2022-03-19]. Dostupné z: https://enlyft.com/.
24. ENLYFT. *Enlyft: Companies using ClickUp* [online]. 2021 [cit. 2022-03-19]. Dostupné z: https://enlyft.com/tech/products/clickup.
25. ENLYFT. *Enlyft: Companies using Harvest* [online]. 2021 [cit. 2022-03-19]. Dostupné z: https://enlyft.com/tech/products/harvest.
26. ENLYFT. *Enlyft: Companies using Atlassian JIRA* [online]. 2021 [cit. 2022-03-19]. Dostupné z: https://enlyft.com/tech/products/atlassian-jira.
27. ENLYFT. *Enlyft: Companies using Redmine* [online]. 2021 [cit. 2022-03-19]. Dostupné z: https://enlyft.com/tech/products/redmine.
28. ENLYFT. *Enlyft: Companies using Toggl* [online]. 2021 [cit. 2022-03-19]. Dostupné z: https://enlyft.com/tech/products/toggl.
29. PAVLÍČEK, Josef. *UI design steps* [online]. 2020 [cit. 2022-03-20]. Dostupné z: https://docs.google.com/presentation/d/1CldSHyC8D8cN2LGrqUVJ2U5eEKn5z9MpkFPmzwZX4Zc/edit#slide=id.p.
30. KUBÍK, Milan. *Webnia: Co je to wireframe a proč je důležitý pro vývoj webu* [online]. 2018-05-09 [cit. 2022-03-20]. Dostupné z: https://webnia.cz/deje-se/co-je-to-wireframe-a-proc-je-dulezity-pro-vyvoj-webu.
31. BALSAMIQ STUDIOS, LLC. *Balsamiq* [online]. 2022 [cit. 2022-03-20]. Dostupné z: https://balsamiq.com/.
32. FIGMA, Inc. *Figma* [online]. 2022 [cit. 2022-03-20]. Dostupné z: https://www.figma.com/.
33. DAMI DEVELOPMENT, s. r. o. *DAMI: Framework - Co to znamená* [online]. 2022 [cit. 2022-03-21]. Dostupné z: https://www.damidev.com/slovnik/framework.
34. GREIF, Sacha. *Frontend frameworks and libraries* [online]. 2021 [cit. 2022-03-21]. Dostupné z: https://2021.stateofjs.com/en-US/libraries/front-end-frameworks.
35. SPACE-O TECHNOLOGIES. *Space-O Technologies: Comparing the Best Front End Frameworks in 2022* [online]. 2022-03-01 [cit. 2022-03-23]. Dostupné z: https://www.spaceotechnologies.com/front-end-frameworks/.
36. SNYK. *What is MIT license* [online]. 2022 [cit. 2022-03-23]. Dostupné z: https://snyk.io/learn/what-is-mit-license/.

37. KALPAKAS, George et. al. *GitHub: Angular* [online]. 2022 [cit. 2022-03-23]. Dostupné z: https://github.com/angular/angular.

38. MOZ, Inc. *What is SEO* [online]. 2022 [cit. 2022-03-23]. Dostupné z: https://moz.com/learn/seo/what-is-seo.

39. MONOCUBED. *Monocubed: List of 10 Best Front end Frameworks to Use For Web Development* [online]. 2022-03-08 [cit. 2022-03-23]. Dostupné z: https://www.monocubed.com/blog/best-front-end-frameworks/.

40. META PLATFORMS, Inc. *React: Introducing JSX* [online]. 2022 [cit. 2022-03-23]. Dostupné z: https://reactjs.org/docs/introducing-jsx.html.

41. O'SHANNESSY, Paul et. al. *GitHub: React* [online]. 2022 [cit. 2022-03-23]. Dostupné z: https://github.com/facebook/react.

42. YOU, Evan. *Vue 3 as the New Default* [online]. 2022-01-10 [cit. 2022-03-23]. Dostupné z: https://blog.vuejs.org/posts/vue-3-as-the-new-default.html.

43. YOU, Evan et. al. *GitHub: Vue 2* [online]. 2022 [cit. 2022-03-23]. Dostupné z: https://github.com/vuejs/vue.

44. YOU, Evan et. al. *GitHub: Vue 3* [online]. 2022 [cit. 2022-03-23]. Dostupné z: https://github.com/vuejs/core.

45. VUETIFY. *Vuetify: Vuetify 3 Beta* [online]. 2022 [cit. 2022-03-24]. Dostupné z: https://next.vuetifyjs.com/en/getting-started/installation/.

46. VUETIFY. *Vuetify: Why Vuetify* [online]. 2022 [cit. 2022-03-24]. Dostupné z: https://next.vuetifyjs.com/en/introduction/why-vuetify/.

47. LEMPERA, Milan. *Programovací jazyky a typy* [online]. 2017-10-16 [cit. 2022-03-26]. Dostupné z: https://blog.lempera.cz/2017/10/programovaci-jazyky-typy.html.

48. MONOCUBED. *TypeScript vs JavaScript: Why Choose TypeScript Over JavaScript* [online]. 2022-03-22 [cit. 2022-03-24]. Dostupné z: https://www.monocubed.com/blog/typescript-vs-javascript/.

49. HARTMAN, James. *TypeScript vs JavaScript: What is the Difference* [online]. 2022-03-12 [cit. 2022-03-24]. Dostupné z: https://www.guru99.com/typescript-vs-javascript.html#9.

50. GEEKFLARE. *Typescript vs Javascript: Understanding the Difference* [online]. 2022-02-21 [cit. 2022-03-24]. Dostupné z: https://geekflare.com/typescript-vs-javascript/.

51. SHOPTET, a. s. *Shoptet: Co je CMS* [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://www.shoptet.cz/slovnik-pojmu/cms/.

52. CHANNEL CROSSINGS, s. r. o. *Channel Crossings* [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://www.chc.cz/.

53. FOUNDATION, WordPress. *WordPress: Democratize Publishing* [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://wordpress.org/about/.

54. WPBEGINNER, LLC. *WPBeginner: 15 Best and Most Popular CMS Platforms in 2022 (Compared)* [online]. 2022-01-01 [cit. 2022-03-26]. Dostupné z: https://www.wpbeginner.com/showcase/best-cms-platforms-compared/.

55. FOUNDATION, WordPress. *WordPress* [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://wordpress.org/.

56. ACTIVE 24, s. r. o. *Co je webhosting a jak vybrat ten nejlepší* [online]. 2020 [cit. 2022-03-26]. Dostupné z: https://www.active24.cz/jak-na-tvorbu-webu/tvorba-stranek-pokrocila/co-je-web-hosting-a-jak-vybrat-ten-nejlepsi.

57. GEORGES, Samuel et. al. *GitHub: October CMS* [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://github.com/octobercms/october.
58. BOBKOV, Alexey; GEORGES, Samuel. *October CMS: Features* [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://octobercms.com/features.
59. LORENC, Pavel. *Výhody OctoberCMS před WordPressem a Jommlou* [online]. 2022-12-12 [cit. 2022-03-26]. Dostupné z: https://pavellorenc.cz/vyhody-octobercms-pred-wordpressem-joomlou.
60. OFFLINE. *October CMS: ResponsiveImages plugin* [online]. 2015-12-21 [cit. 2022-03-26]. Dostupné z: https://octobercms.com/plugin/offline-responsiveimages.
61. PATEL, Anand. *October CMS: SEO Extension* [online]. 2014-10-14 [cit. 2022-03-26]. Dostupné z: https://octobercms.com/plugin/anandpatel-seoextension.
62. PROANIMAL, s. r. o. *Nejlepší platební brána 2022* [online]. 2022 [cit. 2022-03-27]. Dostupné z: https://www.5nej.cz/srovnani-platebnich-bran/.
63. PROANIMAL, s. r. o. *Platební brány: ComGate recenze* [online]. 2021-08-30 [cit. 2022-03-27]. Dostupné z: https://www.5nej.cz/comgate-recenze/.
64. PROANIMAL, s. r. o. *Platební brány: PayU recenze* [online]. 2021-03-09 [cit. 2022-03-27]. Dostupné z: https://www.5nej.cz/payu-recenze.
65. PROANIMAL, s. r. o. *Platební brány: Stripe recenze* [online]. 2021-03-09 [cit. 2022-03-27]. Dostupné z: https://www.5nej.cz/stripe-recenze/.
66. COMGATE PAYMENTS, a. s. *ComGate: Srovnání platebních bran 2022* [online]. 2022 [cit. 2022-03-27]. Dostupné z: https://www.comgate.cz/blog/srovnani-platebnich-bran.
67. WORLDLINE. *Worldline* [online]. 2022 [cit. 2022-03-27]. Dostupné z: https://worldline.com/.
68. STRIPE, Inc. *Stripe: Pricing details* [online]. 2022 [cit. 2022-03-27]. Dostupné z: https://stripe.com/en-cz/pricing#pricing-details.
69. COMGATE PAYMENTS, a. s. *ComGate: Sazebník platební brány* [online]. 2021-03-01 [cit. 2022-03-27]. Dostupné z: https://www.comgate.cz/sazebnik-platebni-brany.
70. GOPAY, s. r. o. *GoPay: Cena* [online]. 2022 [cit. 2022-03-27]. Dostupné z: https://www.gopay.com/cs/cena/.
71. NATTRASS, William. *High demand for employees in Czech IT sector leads to impressive wage growth* [online]. 2021-09-30 [cit. 2022-04-01]. Dostupné z: https://www.expats.cz/czech-news/article/companies-in-the-czech-it-sector-are-struggling-to-find-employees-pushing-up-wages.
72. BAILYN, Evan. *B2B Conversion Rates By Industry* [online]. 2020-09-16 [cit. 2022-04-01]. Dostupné z: https://firstpagesage.com/seo-blog/b2b-conversion-rates-by-industry-fc/.
73. NIELSEN, Jakob. *10 Usability Heuristics for User Interface Design* [online]. 2020-11-15 [cit. 2022-04-23]. Dostupné z: https://www.nngroup.com/articles/ten-usability-heuristics/.
74. MLEJNEK, Jiří. *Analýza problémové domény* [online]. 2022 [cit. 2022-04-01]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/506247/mod_resource/content/5/04.prednaska.pdf.
75. JAHR, Adam. *Vue CLI 3: Creating our Project* [online]. 2022 [cit. 2022-04-27]. Dostupné z: https://www.vuemastery.com/courses/real-world-vue-js/vue-cli.
76. JIANG, Haoqun et. al. *GitHub: Vue CLI* [online]. 2018 [cit. 2022-04-27]. Dostupné z: https://github.com/vuejs/vue-cli/tree/v2.

77. KAWAGUCHI, Kazuya. *Vue CLI: Plugin I18n* [online]. 2022 [cit. 2022-04-27]. Dostupné z: https://www.npmjs.com/package/vue-cli-plugin-i18n.

78. O'BRIEN, Debbie; KYRIAKIDIS, Alex; HAUG, Rolf. *Vue Router for Everyone* [online]. 2022 [cit. 2022-04-28]. Dostupné z: https://vueschool.io/courses/vue-router-for-everyone.

79. KYRIAKIDIS, Alex; HAUG, Rolf. *Vuex for Everyone* [online]. 2022 [cit. 2022-04-28]. Dostupné z: https://vueschool.io/lessons/meet-vuex.

80. YOU, Evan et. al. *Vuex: Getting Started* [online]. 2022-04-07 [cit. 2022-04-28]. Dostupné z: https://vuex.vuejs.org/guide/.

81. GOOGLE. *Material Design: Introduction* [online]. 2022 [cit. 2022-04-29]. Dostupné z: https://material.io/design/introduction.

82. FUNCTIONAL SOFTWARE, Inc. *Sentry* [online]. 2022 [cit. 2022-04-29]. Dostupné z: https://sentry.io/welcome/.

83. KUTÁČ, Pavel. *Sentry: Jak a proč logovat chyby* [online]. 2020-04-21 [cit. 2022-04-30]. Dostupné z: https://www.kutac.cz/pocitace-a-internety/sentry-jak-a-proc-logovat-chyby.

84. AUTH0, Inc. *Auth0: Secure access for everyone, but not just anyone* [online]. 2022 [cit. 2022-04-30]. Dostupné z: https://auth0.com/.

85. AUTH0, Inc. *Auth0: Pricing* [online]. 2022 [cit. 2022-04-30]. Dostupné z: https://auth0.com/pricing.

86. ARIAS, Dan. *Auth0: The Complete Guide to Vue.js User Authentication with Auth0* [online]. 2021-05-27 [cit. 2022-04-30]. Dostupné z: https://auth0.com/blog/complete-guide-to-vue-user-authentication/.

87. AUTH0, Inc. *Auth0: Universal Login* [online]. 2022 [cit. 2022-04-30]. Dostupné z: https://auth0.com/universal-login.

88. JAHODA, Bohumil. *Uložiště localStorage* [online]. 2015-09-17 [cit. 2022-05-01]. Dostupné z: https://jecas.cz/localstorage.

89. LOTANNA, Nwose. *Using event bus in Vue.js to pass data between components* [online]. 2019-09-18 [cit. 2022-05-01]. Dostupné z: https://blog.logrocket.com/using-event-bus-in-vue-js-to-pass-data-between-components/.

90. BEMENDERFER, Joshua. *How To Create a Global Event Bus in Vue 2* [online]. 2021-03-24 [cit. 2022-05-01]. Dostupné z: https://www.digitalocean.com/community/tutorials/vuejs-global-event-bus.

91. KITNER, Radek. *Typy testování software* [online]. 2021 [cit. 2022-05-03]. Dostupné z: https://kitner.cz/testovani_softwaru/typy-testovani-software-trideni-testu/.

92. HLAVA, Tomáš. *Testování software: Fáze a úrovně provádění testů* [online]. 2011-08-21 [cit. 2022-05-03]. Dostupné z: http://testovanisoftwaru.cz/tag/akceptacni-testovani/#acceptance.

93. VOJÁK, Michal. *Jak dělat uživatelské testování: Úvod do uživatelského testování* [online]. 2020-09-15 [cit. 2022-05-03]. Dostupné z: https://designdev.cz/jak-delat-uzivatelske-testovani.

94. NIELSEN, Jakob. *Why You Only Need to Test with 5 Users* [online]. 2000-03-18 [cit. 2022-05-03]. Dostupné z: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.

95. VOJÁK, Michal. *Příprava testu použitelnosti* [online]. 2020-09-29 [cit. 2022-05-03]. Dostupné z: https://designdev.cz/priprava-testu-pouzitelnosti.

96. VOJÁK, Michal. *Průběh testu použitelnosti* [online]. 2020-10-19 [cit. 2022-05-03]. Dostupné z: `https://designdev.cz/prubeh-testu-pouzitelnosti`.

Obsah přiloženého média

- `README.txt` ..... stručný popis obsahu média
- `Implementace` ..... implementace samotné aplikace
  - `t2t-app.zip` ..... zdrojové kódy implementace samotné aplikace ve formátu ZIP
- `Text` ..... text práce
  - `BP - Timer2Ticket.pdf` ..... text práce ve formátu PDF
  - `Zdroj.zip` ..... zdrojová forma textu práce ve formátu ZIP
- `Wireframy` ..... wireframy k návrhu
  - `Propagační web.pdf` ..... wireframy propagačního webu
  - `Samotná aplikace.pdf` ..... wireframy samotného nového frontendu aplikace
- `Záznamy z testování` ..... záznamy z testování
  - `Testovaní - Filip Glazar.mkv` ..... záznam z testování s Ing. Filipem Glazarem
  - `Testovani - Martin Dvořák.mkv` ..... záznam z testování s Martinem Dvořákem
  - `Testovaní - Oldřich Malec.mkv` ..... záznam z testování s Ing. Oldřichem Malcem
  - `Testovaní - Ondřej Wrzecionko.mkv` ..... záznam z testování s Ondřejem Wrzecionko