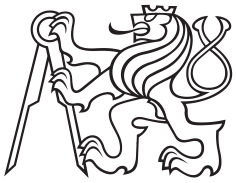
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Webová aplikace pro sběr, publikaci a analýzu informací

Nikolai Ogoltsov

Vedoucí: Ing. Pavel Náplava
Obor: Softwarové systémy
Studijní program: Otevřená informatika
Květen 2018

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Ogoltsov** Jméno: **Nikolai** Osobní číslo: **456907**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Softwarové systémy**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Webová aplikace pro sběr, publikaci a analýzu informací

Název bakalářské práce anglicky:

Web application for information collecting, publishing and analysis

Pokyny pro vypracování:

Vytvořte aplikaci, která umožní provádět základní sběr různých typů informací mezi různými uživateli. Základem aplikace musí být třívrstvá architektura, založená na kombinaci mikroslužeb, zajišťujících různé části fungování aplikace. Účelem aplikace je především sběr v lokálním, uzavřeném prostředí. Postupujte následovně:
1) seznamte se s problematikou sběru a hodnocení různého typu informací.
2) proveďte analýzu existujících aplikací a proveďte jejich porovnání s ohledem na požadavky, které upřesní vedoucí práce.
3) navrhněte novou aplikaci, která na principu kombinace různých mikroslužeb umožní sbírat vybrané typy informací a ty následně automaticky vyhodnocovat.
4) navrženou aplikaci realizujte minimálně v podobě funkčního prototypu.
5) společně s vedoucím práce vytvořte minimálně tři různé scénáře sběru informací, včetně konkrétních dat a jejich prostřednictvím vytvořený prototyp otestujte.

Seznam doporučené literatury:

- Richardson Chris, Microservices Patterns, With examples in Java, 2018, Manning Publications, ISBN 9781617294549
- Carnell John, Spring Microservices in Action, 2017, Manning Publications, ISBN 99781617293986

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Pavel Náplava, katedra ekonomiky, manažerství a humanitních věd FEL

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

Datum zadání bakalářské práce: **16.02.2018** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **30.09.2019**

Ing. Pavel Náplava podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. _____ Datum převzetí zadání

_____ Podpis studenta

Poděkování

Chtěl bych poděkovat Ing. Pavlu Náplavovi za vedení bakalářské práce, cenné rady, trpělivost a příjemnou spolupráci.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze, 25. května 2018

. . . . . .

Abstrakt

Online průzkumy jsou známé poměrně dlouhou dobu, i přesto ale nepřestávají být populární. Naopak, pokrok informačních technologií a metod sociální komunikace způsobuje jejich ještě větší efektivitu a jednoduchost realizace.

V současné době existuje velký počet služeb, které umožňují takové průzkumy provádět. Většina nabízí způsoby generace dynamických dotazníků, sběr odpovědí účastníků a následnou analýzu získaných dat, ale žádné z prozkoumaných řešení neumožňuje sběr dat za podmínek autonomní práce aplikací na mobilních zařízeních.

Uvědomění si existujícího omezení posloužilo jako inspirace pro analýzu, návrh a implementaci první fáze vývoje podobného softwaru. Důraz byl především kladen na vývoj dobře navrženého systému, připraveného na rozšíření o klientské aplikace dalších platforem a operačních systémů.

Ve výsledku práce byl realizován informační systém skládající se z několika propojených a snadně spustitelných komponent. Tvořící jej webové aplikace jsou kompatibilní s moderními prohlížeči jak stacionárních počítačů, tak i mobilních zařízení.

Klíčová slova: průzkum, webová aplikace, softwarová architektura, mikroslužba, autentizace, nasazení, agilní vývoj

Abstract

Online research methods have been known for a considerably long time, nevertheless they do not cease being popular. On the contrary – a progress of information technology and methods of social communication lead to their bigger efficiency and simplification of their implementation.

At present, there are numerous services that enable to implement such research methods. Most of them provide the ways of dynamic questionnairs generation, collection of respondents´ answers and resulting analysis of obtained data. However, none of the researched solutions enable the data collection under the condition of autonomous work of applications on mobile devices.

An identification of an existing constraint served as an inspiration for an analysis, design and implementation of the first development phase of a similar software for research purposes. The emphasis was placed especially on a development of a well-designed system prepared for a client application of additional platforms and operating systems extension.

The thesis resulted in a realization of an information system consisting of several connected and easily runnable components. Developed web applications are compatible with modern browsers of both the desktop computers and mobile devices.

Keywords: research, wep application, software architecture, microservice, authentication, deployment, agile development

Title translation: Web application for information collecting, publishing and analysis

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

The important thing is not to stop questioning.
Curiosity has its own reason for existence.

Albert Einstein

Jedním z nejvýraznějších rysů lidské sociální podstaty je schopnost vyjadřovat, vnímat a pobírat mínění ostatních. V dané kapitole a celé práci nás nebudou zajímat podmínky vzniku a projevování zmíněných jevů, ale budeme se věnovat tomu, jak se s nimi v současné době lidstvo naučilo zacházet, jaké jsou moderní technologie pro jejich vypracování a zkoumání.

Nepochybně si místo v řadě sociálně-humanitních sfér zasloužily oblasti agregace a analýzy informací. Podobné fundamentální vědy a jejich odvětví, zpočátku zkoumající anebo těsně spojité se zmíněnou problematikou, sdružují nespočetné množství lidí a institucí. Sice jejich motivace může být jak komerční a soukromá, tak i státní, ale ve větší míře se věnují podobným výzkumům za použití shodných metodik a nástrojů. Analýza sebraných dat poskytuje exaktní a včasnou informaci ohledně chování, potřeb, názorů a motivace některé cílové skupiny lidí. Například podniky s ní kalkulující jsou schopné vyvíjet své produkty a služby tak, že se budou cíleně přibližovat očekáváním klientů.

V současné době, kvůli úspěchům ve sférách komunikace a moderních informačních technologií, se významně změnila forma výzkumů a s nimi spojené realizované procesy. Méně než před dvaceti lety se začaly využívat tzv. **online průzkumy**, které vytěsnily tradiční „papírové“ analogy. Online průzkum lze definovat jako metodu průzkumu, která zapojuje sběr informací prostřednictvím moderních způsobů komunikace: webových a

mobilních aplikací, emailů, QR kódů apod. Velmi užitečnými vlastnostmi jsou dlouhodobé a zabezpečené ukládání senzitivních dat a nástroje pro okamžitou analýzu získaných dat. Podobné průzkumy jsou mnohem působivější než tradiční postupy, s ohledem na snadnost přístupu a finanční úsporu, které s sebou přinášejí.

Online průzkumy plně vyhovují většině požadavků uživatelů. Ale uveďme příklady, kdy to platit nebude:

- Pokud sběr odpovědí probíhá v podmínkách síťové nedostupnosti, podobné aplikace nejsou schopné vyměňovat informace s backendovými službami. Takovou situaci si lze jednoduše představit: pracovník některé firmy provádí průzkum ve vesnici, kde není dobré internetové pokrytí.
- Sběr odpovědí probíhá na mobilním zařízení a není nutné odesílat data pro ukládání v databázích třetích stran. Navíc někteří klienti vůbec nechtějí citlivá data někam odesílat z bezpečnostních důvodů.

Hlavní myšlenkou je tedy to, že podobné softwarové produkty by měly podporovat autonomní práci mobilních aplikací. Sběr odpovědí a jejich vyhodnocování může probíhat hned na klientských zařízeních buď úplně bez odesílání dat, anebo se zpožděným odesíláním.

Doba provedení práce je omezená, proto je cílem stanovená implementace prototypu informačního systému, který bude mít existujícím řešením podobnou funkcionalitu, ale navíc ještě bude nabízet jak návrhovou, tak i implementační bázi pro další vývoj mobilních klientských aplikací s podporou autonomní práce.

V následujících kapitolách popíšeme: výzkum existujících řešení, analýzu uživatelských požadavků a budoucího produktu, architektonický návrh, implementaci, otestování několika případů užití systému a organizaci práce na projektu.

Kapitola 2

Cíl práce

Cílem práce je návrh a implementace funkčního prototypu aplikace pro generaci forem sběru informací, provádění veřejných a privátních průzkumů, sběr výsledků a jejich vyhodnocení. Výsledný produkt je souhrnem několika propojených a snadně spustitelných komponent. Tvořící jej webové aplikace jsou kompatibilní s moderními prohlížeči jak stacionárních počítačů, tak i mobilních zařízení.

Analýze základních požadavků cílové skupiny uživatelů bude věnovaná významná část práce, ve které budou prozkoumána řešení existující na trhu. Kvalitní analýza získaných informací usnadní definici funkčních a nefunkčních požadavků na celý produkt a stane se klíčovou podmínkou pro návrh flexibilní, lehce rozšiřitelné a moderní architektury aplikace.

Předpokládá se, že bude implementován a otestován minimální počet základních požadavků a výsledné řešení bude nabízet dobře vypracované a zřejmé způsoby rozšíření existující funkcionality v budoucnu.

Kapitola 3

Analýza a specifikace požadavků

3.1 Základní informace o online průzkumech

Online průzkum jsme definovali jako metodu průzkumu, která zapojuje sběr informací prostřednictvím moderních způsobů komunikace: webových a mobilních aplikací, emailů, QR kódů apod. Velmi užitečnými vlastnostmi jsou dlouhodobé a zabezpečené ukládání senzitivních dat a nástroje pro okamžitou analýzu získaných dat. Podobné průzkumy jsou mnohem účinnější než tradiční postupy s ohledem na snadnost přístupu a finanční úsporu, které s sebou přinášejí.

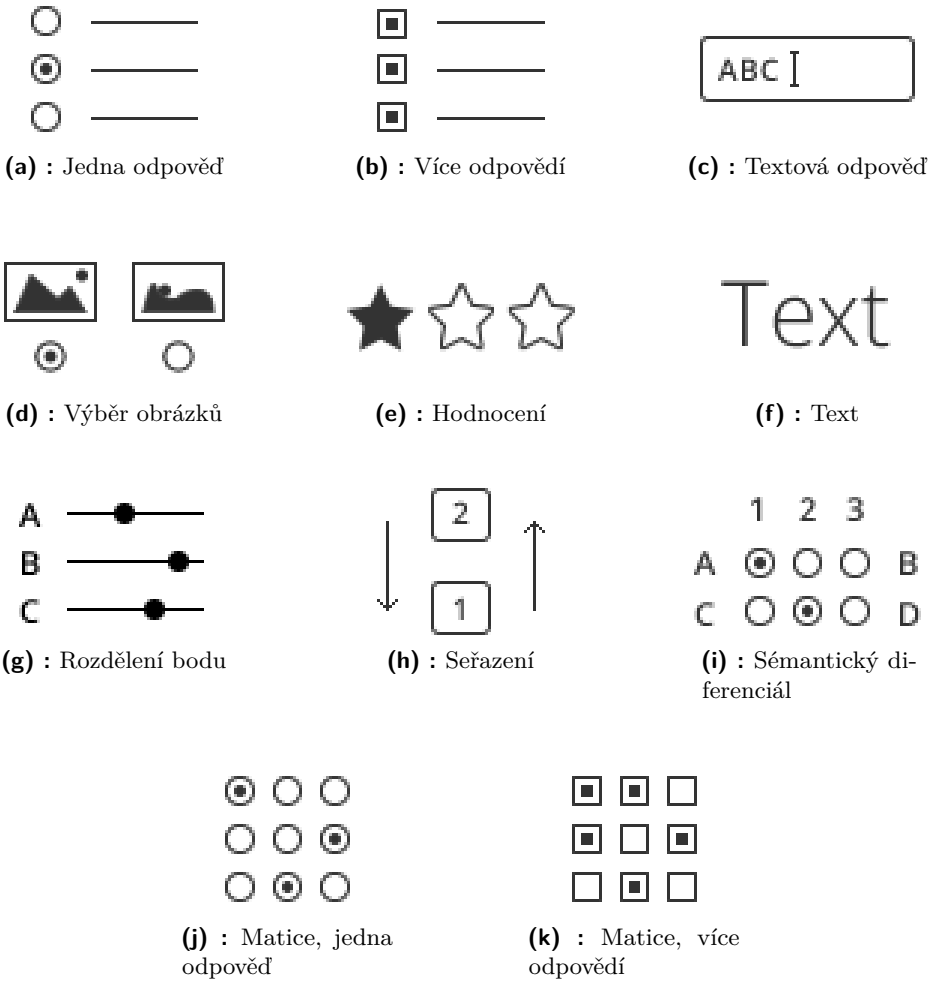
3.1.1 Klasifikace online průzkumů

Většina online průzkumů spadá do třech skupin, ze kterých každá plní odlišný a velmi důležitý účel:

- **Dotazník (survey):** dotazníky se skládají z několika otázek různých forem většinou cílených na formování přesného vyjádření účastníka o nějaké problematice.
- **Volba (poll):** jedná se o dotazník, ve kterém účastník musí odpovědět pouze na jednu položenou otázku s několika předdefinovanými odpověďmi.
- **Kvíz (quiz):** primární záměr kvízu je testování znalosti účastníka během studia, tréninku, přijímacího řízení apod. Na rozdíl od volby nebo dotazníku není cílem získat zpětnou vazbu a názor.

3.1.2 Klasifikace elementů pro sběr odpovědí

Online průzkum se skládá ze sady komponent, které určují jeho budoucí strukturu. K ní je vždycky aplikován nějaký vzhledový motiv s textovou náplní pro zobrazení účastníkům. Každý z elementů uvedených na obrázku 3.1 patří do skupiny základních komponent. Je jich sice více, ale další jsou většinou buď kombinací některých ze zmíněných anebo jejich blízkou modifikací.



(a) : Jedna odpověď

(b) : Více odpovědí

(c) : Textová odpověď

(d) : Výběr obrázků

(e) : Hodnocení

(f) : Text

(g) : Rozdělení bodu

(h) : Seřazení

(i) : Sémantický diferenciál

(j) : Matice, jedna odpověď

(k) : Matice, více odpovědí

Obrázek 3.1: Elementy tvořící online dotazníky

3.1.3 Klasifikace služeb pro provádění průzkumu

Forma poskytovaných produktů

Jednou z vlastností podobných softwarových produktů, podle které je lze rozlišovat, je způsob jejich nasazení a provozování. Zvolili jsme tři způsoby, pokrývající největší rozsah klientských požadavků:

- **Externí služba** běžící na serverech majitele. Je to klasická Client-Server webová aplikace, kterou klient dostává v podobě SaaS. Daný způsob je aktuálně nejvíce rozšířený a bude se hodit pro největší skupinu uživatelů.
- **Interní služba** běžící na serverech klienta. Je to klasická Client-Server webová aplikace, kterou klient dostává v podobě As-Is. Daný způsob se z pohledu uživatele systému od prvního liší pouze tím, že umožňuje lépe nakonfigurovat aplikaci podle svých potřeb a provozovat ji ve své infrastruktuře. Což může být lákavé jak z hlediska obchodních a marketingových předpokladů, tak i z bezpečnostních důvodů.
- **Aplikace pro koncové zařízení**. Jedná se o desktopové a mobilní aplikace, které budou schopné nabídnout v něčem omezenou funkcionalitu na klientských zařízeních, aniž by se museli připojovat k webovým službám. Daný způsob může být nejvíce vyžadován zákazníky, kteří potřebují úplnou bezpečnost sbíraných dat.

Základní dělení na podsystémy

Pokusili jsme se o definici několika základních subdomén typických pro každý softwarový produkt existující v zkoumané sféře. Jedná se o „high-level“ členění na logicky osamocené podsystémy, které činí funkci nezávislou na ostatních podsystémech. Ve skutečnosti dané rozdělení určuje role uživatelů, které pak budou využívat každou z přečíslených subdomén:

- Systém pro administraci firemních či uživatelských účtů, nastavení tarifních plánů a konfiguraci přístupových práv.
- Systém pro vytváření průzkumníkových formulářů, editaci jejich obsahu a vzhledových parametrů.
- Systém pro doručení dotazníků firemním klientům a sběr odpovědí cílové skupiny.
- Systém pro shromáždění uživatelských odpovědí, ochranu citlivých dat, mechanismy analýzy a dokumentace sebraných informací.

3.2 Rešerše

3.2.1 Motivace

Hlavní motivací provést dané rešerše pro nás byla možnost prozkoumat řešení existující v současné době, která umožnila určení nejlepších praktik obvyklých pro uživatele funkčností. Výsledky průzkumu nám pomohly provést konceptuální analýzu budoucího systému na mnohem vyšší úrovni: byly určeny základní požadavky uživatelů a jejich rozdělení na role, byly projeveny nefunkční požadavky, jako jsou například standardy zabezpečení uživatelských údajů a spektrum podporovaných platforem a zařízení.

3.2.2 Zdroje a kritéria průzkumu

Proto, abychom si mohli vybrat nástroje, které budeme porovnávat mezi sebou a používat v detailnějším průzkumu, potřebujeme je alespoň přibližně charakterizovat. Zvolené parametry umožní podrobit výběru nevhodné kandidáty a spolehlivěji definovat cílovou skupinu produktů, které nás zajímají.

- Produkt musí mít lokalizaci alespoň v českém (preferováno) nebo anglickém jazyce. Parametr se týká nejenom uživatelského rozhraní, ale i návodu k použití, textu smluv apod.
- Produkt musí mít smysluplnou dokumentaci. Měla by být popsána funkcionalita aplikace a veškeré další podmínky včetně ceny, omezení apod.
- Produkt musí být komerčně úspěšný. Daný parametr nemůžeme uvažovat za objektivní, protože není k dispozici žádná ověřená statistická hodnota. Nicméně jsme je posuzovali podle informací o významných klientech a jejich počtu, uvedených na webových stránkách produktu a souvisejících článcích.
- Produkt musí být buď úplně bezplatným anebo mít neplacenou verzi s omezenou, ale základní funkcionalitou.

Jako zdroje informací jsme se dohodli používat různé odborné články porovnávající produkty [7] [8] [9], odezvy jednotlivých klientů a domovské webové stránky poskytovatelů služeb.

3.2.3 Porovnání výsledků průzkumu

Proto, abychom dokázali vyvinout konkurenceschopný softwarový produkt, provedli jsme výzkum řešení existujících na trhu. Jelikož jsme byli omezení časově, bylo rozhodnuto pro výběr z pěti referenčních služeb, odpovídajících vybraným kritériím:

- Free Online Surveys [1]
- eSurvey Creator [2]
- Survey Monkey [3]
- QuestionPro [4]
- Survio [5]

Tabulky [C.1], [C.2] a [C.3] obsahují výslednou informaci o jejich funkcionalitě v části generace, sběru odpovědí a analýzy výsledků. Data uvedená v těchto tabulkách umožnila definovat většinu ze základních požadavků a případů užití vyvíjeného produktu.

Zkoumání daných produktů plně potvrzuje zmíněný dříve předpoklad. Žádný z nich neumožňuje offline provedení průzkumu. Tedy můžeme uvažovat o tom, že většina podobných služeb v době provedení práce autonomní sběr odpovědí nepodporuje, a že je kvůli tomu náš cíl ještě více opodstatněný.[6].

[1] Free Online Surveys: freeonlinesurveys.com
[2] eSurvey Creator: esurveycreator.com
[3] Survey Monkey: surveymonkey.com
[4] QuestionPro: questionpro.com
[5] Survio: survio.com
[6] Ke konci práce už takovou mořnost nabízí QuestionPro

3.3 Byznys požadavky

3.3.1 Funkční požadavky

Autentizace a autorizace

- FR_A1: Systém musí umožňovat registraci, přihlášení a odhlášení uživatele.
- FR_A2: Systém musí umožňovat registrací firemního účtu.
- FR_A3: Systém musí umožňovat udělení a odebírání přístupu uživatele k firemnímu účtu.
- FR_A4: Systém musí rozlišovat práva uživatelů pro operace v rámci jednoho účtu.

Generování prototypů a jejich publikace

- FR_G1: Systém musí umožňovat vytváření, odstraňování a editaci průzkumů.
- FR_G2: Systém musí umožňovat konfiguraci průzkumu ve formě poll.
- FR_G3: Systém musí umožňovat konfiguraci průzkumu ve formě survey.
- FR_G4: Systém musí umožňovat konfiguraci průzkumu ve formě quiz.
- FR_G5: Systém musí umožňovat stylizaci průzkumu alespoň třemi různými způsoby.
- FR_G6: Systém musí umožňovat přidání obrázků do průzkumu.
- FR_G7: Systém musí umožňovat přidání odkazů na externí zdroje do průzkumu.
- FR_G8: Systém musí umožňovat přidání základních druhů validace průzkumu.
- FR_G9: Systém musí umožňovat definici logických pravidel, podle kterých se dotazník bude chovat dynamicky (větvení).
- FR_G10: Systém musí umožňovat publikaci průzkumu do veřejné části, po které již editovatelný není.

Sběr odpovědí a jejich analýza

- FR_S1: Systém musí po publikací průzkumu zahájit sběr uživatelských odpovědí.

- `FR_S2`: Systém musí umožňovat zastavení / pokračování / ukončení sběru uživatelských odpovědí pro určitý dotazník.
- `FR_S3`: Systém musí umožňovat účast v průzkumu anonymního uživatele.
- `FR_S4`: Systém musí umožňovat účast v průzkumu přihlášeného uživatele.
- `FR_S5`: Systém musí umožňovat prohlížení seznamu odpovědí účastníků.
- `FR_S6`: Systém musí umožňovat prohlížení jednotlivých odpovědí účastníků.
- `FR_S7`: Systém musí umožňovat generaci reportu výsledků ukončeného dotazníku ve formě tabulek a grafů.
- `FR_S8`: Systém musí umožňovat export reportu ve formátech CSV a PDF.
- `FR_S9`: Systém musí umožňovat přístup k reportu pomocí vygenerovaného odkazu.

3.3.2 Kvalitativní požadavky a omezení

Webová aplikace pro mobilní zařízení

- `NFR_MWA1`: Funkcionalita musí mít správné chování popsané ve FR pro verze prohlížečů:
  - Google Chrome od verze 50.0 do aktuální verze ke konci vývoje aplikace (pro Android a IOS).
  - Safari od verze 9.1.3 do aktuální verze ke konci vývoje aplikace (pro IOS).
- `NFR_MWA2`: Aplikace musí být adaptována pro další rozměry obrazovky ve vertikální orientaci:
  - Od iPhone SE (320 x 568 points, 640 x 1136 pixels).
  - Do iPhone 8+ (414 x 736 points, 1242 x 2208 pixels).

Hlavní webová aplikace

- `NFR_WA1`: Funkcionalita musí mít správné chování popsané ve FR pro verze prohlížečů:
  - Google Chrome od verze 50.0 do aktuální verze ke konci vývoje aplikace.
  - Safari od verze 9.1.3 do aktuální verze ke konci vývoje aplikace.

- `NFR_WA2`: Bezpečnostní prvky budou implementované pouze na základní úrovni. Pro autentizaci uživatelů bude využit jeden ze známých Identity Providerů, například WSO2IS. Komunikace mezi klientskou a serverovou částí bude probíhat zabezpečeně, ale žádné sofistikované problémy bezpečnosti se řešit nebudou.

Backendové služby

- `NFR_BE1`: Výpadek jedné služby nemůže způsobovat vadné chování funkcionality, která se týká ostatních služeb.
- `NFR_BE2`: Služby musejí vystavovat RESTové rozhraní, které pro každou z nich musí být dobře popsané.

Nasazení

- `NFR_DEP1`: Výsledkem vyplnění práce bude aplikace skládající z několika komponent. Musí být navržen nástroj z běžné praxe, který by umožňoval jednoduchý a pro klienty pochopitelný způsob konfigurace a rozběhnutí jak celého systému, tak i každé jeho části zvlášť. Řešení musí být zadokumentováno a být podporováno v *nix operačních systémech a Windows.

3.4 Datové modely

Jelikož související problematika není triviální a je přímo spojená s mechanismy dynamického generování datových struktur, rozhodli jsme se uvést popis datového modelu aplikace na několika různých úrovních abstrakce. Motivovali jsme se nejenom potřebou zvýšení čitelnosti kapitoly, ale i nutností pochopit zásadní rysy celého systému a navrhnout správnou datovou kostru, od které by se pak větvila generická struktura bez rozporu s klíčovou myšlenkou.

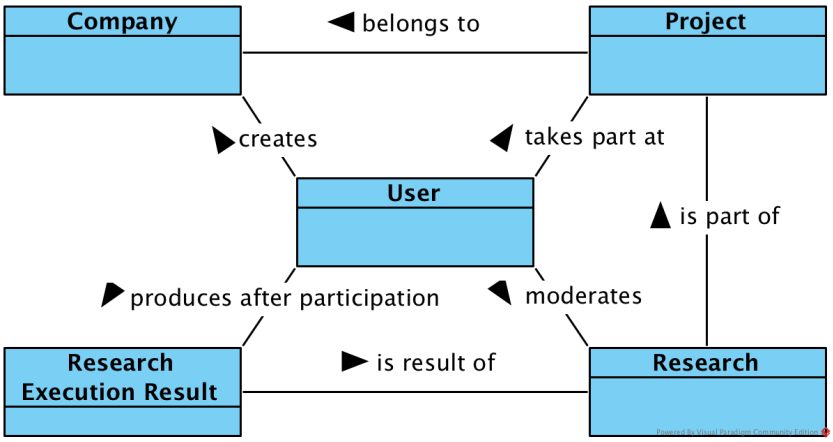
3.4.1 Konceptuální datový model

„Konceptuální datový model je datový model s největší úrovní abstrakce a shrnutí. Specifická informace pro určité platformy a jiné detaily implementace je z podobných modelů

eliminovaná. <...> Konceptuální datový model umožňuje identifikovat vysokoúrovňové klíčové entity byznysu a systému a stanoví relace mezi nimi. Kromě toho dané modely přispívají k definici klíčové podstaty problémů, které jsou se systémem spojeny.“ [10]

Na obrázku 3.2 je uvedeno pět základních entit, které jsme vyčlenili po definici byznys požadavků [3.3]. Podle našeho názoru všechny tvoří jádro oblasti, kterou jsme popsali dřív. Za jejich použitím lze plně popsat procesy probíhající v aplikaci:

- **Uživatel systému** (User) je reprezentací jakéhokoliv člověka, oprávněného provádět nad softwarovým produktem operace.
- **Firma** (Company) je reprezentací firemního účtu, který byl založen *uživatelem systému*. Je základní agregační a separační jednotkou, ke které pak budou aplikována pravidla vyúčtování, rozdělení rolí apod.
- **Projekt** (Project) je reprezentací projektu jedné z *firem*. S ohledem na to, že většinou ve firmě probíhá současná práce nad několika projekty, nad kterými můžou pracovat různé departementy, kolektivy a specialisté (*uživatele systému*), považovali jsme je za důležitou izolační jednotku.
- **Průzkum** (Research) je součástí *projektu* a je editován jeho participanty (*uživateli systému*). Po publikaci je průzkum k dispozici pro vyplnění účastníky (*uživateli systému*) do té doby, než bude zastaven.
- **Provedení průzkumu** (Research Execution) je reprezentací účasti v průzkumu. Shromažďované výsledky slouží ke kolekci odpovědí *uživatelů systému* a analýze jednotlivých odpovědí účastníků.



Obrázek 3.2: Konceptuální datový model

3.4.2 Logický datový model

Logický datový model je ve skutečnosti implementací a rozšířením konceptuálního datového modelu. Je to variace datového modelu, který částečně nebo vcelku reprezentuje byznys požadavky organizace a vyvíjí se před fyzickým modelem. Jedná se o detailnější popis datové struktury s entity ve formě objektu, tabulek, grafů apod. a jejich relací.

Hlavní model

Na obrázku D.1 je zobrazen logický datový model, rozšiřující konceptuální datový model, který byl popsán dříve. Na rozdíl od předchozího, daný model odpovídá nejenom entitám z reálného světa, ale i jejich „decomposed“ objektové struktuře. Tudíž ve výsledku přibývají nové atomárně rozdělené prvky, každý ze kterých má každý přesně definovaný implementační účel.

- **Účast v projektu** (Project Participation) popisuje uzlovou entitu, která spojuje *uživatele systému* a *projekt*. Entita byla oddělená ne kvůli potřebě převést many-to-many relaci na dvě relace one-to-many, ale pro účel uložení informace o účasti (v budoucích verzích aplikace): datové rozmezí účasti, přiřazené člověku rolí v projektu apod.
- **Prototyp průzkumu** (Project Prototype). Každý *průzkum* odpovídá některé pevné struktuře, která je určená implementací jeho prototypu. Prototyp slouží šablonou, do které *účastník projektu* ukládá data a přidává *vzhledovou konfiguraci*.
  - **Vzhledová konfigurace prototypu průzkumu** (Project Prototype Display Configuration) specifikuje vzhledové charakteristiky *prototypu průzkumu*.
  - **Téma prototypu průzkumu** (Project Prototype Theme) reprezentuje jedno vzhledové téma ze sady předurčené v systému. Každé z témat má stejnou strukturu a je aplikované stejným způsobem k *prvkům prototypu průzkumu*.
  - **Stránka prototypu průzkumu** (Project Prototype Page) Každá implementace *prototypu průzkumu* má pevně danou strukturu stránek, která se skládá z implementací daného rozhraní.
  - **Prvek prototypu průzkumu** (Project Prototype Element) Každá implementace *stránky prototypu průzkumu* má pevně danou strukturu vizuálních elementů (popsaných v kapitole [3.1.2]), která se skládá z implementací daného rozhraní.
- **Výsledek provedení průzkumu** (Research Execution Result). Každé *provedení průzkumu* má pevnou strukturu odpovědí účastníka, která je určená implementací výsledku. Implementace výsledku vždycky koresponduje s implementací *prototypu průzkumu*, protože struktura odpovědí odpovídá struktuře otázek.

- **Výsledek provedení stránky průzkumu** (Project Page Execution Result). Každá implementace výsledku provedení stránky odpovídá implementaci *stránky prototypu* a má stejnou související strukturu odpovědí.
- **Výsledek provedení prvku průzkumu** (Project Element Execution Result). Každá implementace výsledku provedení prvku odpovídá implementaci *prvku prototypu* a má stejnou související strukturu odpovědí.

Implementace rozhraní hlavního modelu

Na obrázku [D.2] je zobrazen logický datový model s popisem implementací rozhraní hlavního modelu [3.4.2].

- Prototyp průzkumu (Research Prototype)
  - **Poll Prototype**, **Quiz Prototype** a **Survey Prototype** jsou realizace prototypu pro každý druh průzkumů podle [3.1.1].
- Výsledek provedení průzkumu (Research Execution Result)
  - **Poll Execution Result**, **Quiz Execution Result** a **Survey Execution Result** jsou realizace výsledku provedení průzkumů pro každou implementaci prototypu.
- Prototyp stránky průzkumu (Research Prototype Page)
  - **Poll Prototype Page**, **Quiz Prototype Page** a **Survey Prototype Page** jsou realizace stránky prototypů pro každý druh průzkumů podle [3.1.1].
  - **Research Prototype Start Page**, **Research Prototype End Page** jsou realizace startové a poděkovací stránky, které jsou stejné pro všechny druhy průzkumů.
- Výsledek provedení stránky průzkumu (Research Page Execution Result)
  - **Poll Page Execution Result**, **Quiz Page Execution Result** a **Survey Page Execution Result** jsou realizace výsledku provedení stránky pro každou implementaci prototypu stránky.
- Prototyp prvku průzkumu (Research Prototype Element)
  - **Button Prototype Element** a **Text Prototype Element** jsou realizace neinteraktivních prvků prototypu, které umožňují změnit informaci v textových elementech rozhraní anebo tlačítkách.
  - **Text Answer Prototype Element**, **Single Choice Prototype Element** a **Multiple Choice Prototype Element** jsou realizace interaktivních prvků prototypu ze sady základních prvků popsaných v [3.1.2].
- Výsledek provedení prvku průzkumu (Research Element Execution Result)

- **Text Answer Execution Result**, **Multiple Choice Execution Result** a **Single Choice Execution Result** jsou realizace výsledku provedení prvku pro každou implementací interaktivních prvků.

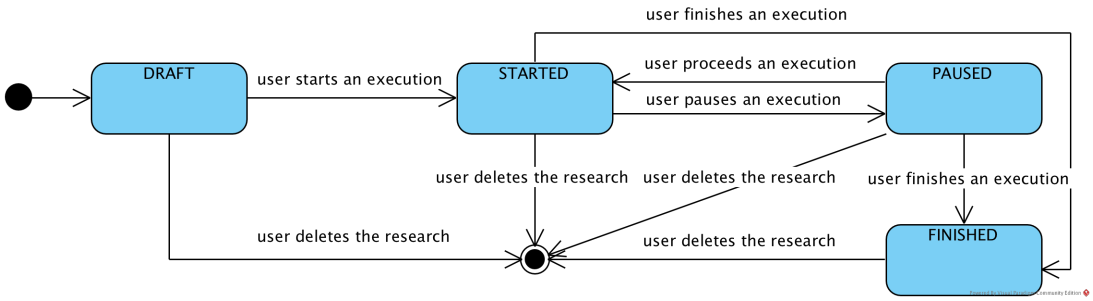
3.5 Stavové diagramy

„Stavový diagram zachycuje jednotlivé stavy objektu a přechody mezi nimi. Stavové diagramy se používají především pro popis chování určitého objektu napříč více případy užití a jejich vznik je spojen už s prvními objektově orientovanými technikami“. [11]

3.5.1 Životní cyklus průzkumu

Pod životním cyklem entity rozumíme souhrn stavů, omezení a přechodů v době mezi její instalací a zrušením. V případě průzkumu jsou dohromady čtyři. Schéma životního cyklu průzkumu je představeno na obrázku [3.3]:

- **Předběžný návrh** (DRAFT). V daném stavu průzkum může být editován moderátory. Po ukončení práce průzkumem jej moderátor zveřejňuje, tím ho převádí do nového stavu sběru odpovědí. Alternativně moderátor může smazat průzkum a entita ukončí svůj životní cyklus.
- **Sběr odpovědí** (STARTED). V daném stavu je možné se průzkumu zúčastnit. Ze stavu jsou možné přechody do stavu *pozastavení* a *ukončení* podle potřeb moderátorů průzkumu. Alternativně moderátor může smazat průzkum a entita ukončí svůj životní cyklus.
- **Pozastavený sběr odpovědí** (PAUSED). V daném stavu se průzkumu nikdo zúčastnit nemůže do té doby, až moderátor převede jej zpět do stavu *sběru odpovědí* anebo úplně *ukončí*. Alternativně moderátor může smazat průzkum a entita ukončí svůj životní cyklus.
- **Ukončený sběr odpovědí** (FINISHED). V daném stavu je možná jenom analýza sebraných dat. Obnovit sběr dat možné není, stejně jako i odpovědět: průzkum už není veřejně dostupný. Po ukončení sběru odpovědí jej moderátor může pouze smazat.



Obrázek 3.3: Stavový diagram životního cyklu průzkumu

Kapitola 4

Návrh

Výsledky konceptuální analýzy, provedené v předchozí kapitole č.3, budou podrobené vypracování a poslouží pro navazující technický popis složení aplikace. Cílem návrhu je smysluplný výběr principů vývoje systému: to, ze kterých komponent komunikujících mezi sebou se bude skládat, jakým návrhovým vzorům bude odpovídat a které běžné praktiky architektonického návrhu použije. Veškeré rozhodnutí se budou týkat jenom strukturálních a behaviorálních vlastností systému, žádné implementační detaily nebudou součástí této kapitoly.

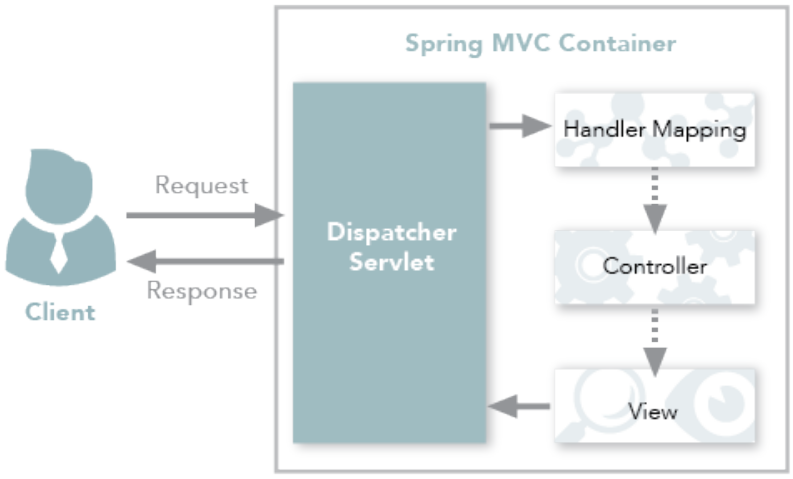
4.1 Client-server interakce

Konečným cílem práce je vývoj *webové* aplikace. Tudíž se jedná o klientskou aplikaci a webový server komunikující mezi sebou prostřednictvím HTTP protokolu. Hlavním cílem je definovat, v jaké míře budou zmíněné části systému navzájem spojené a kde se bude koncentrovat aplikační logika.

Webové služby můžeme rozdělit do dvou skupin podle toho, v jaké formě vystavují své rozhraní:

Vracejí formátované stránky

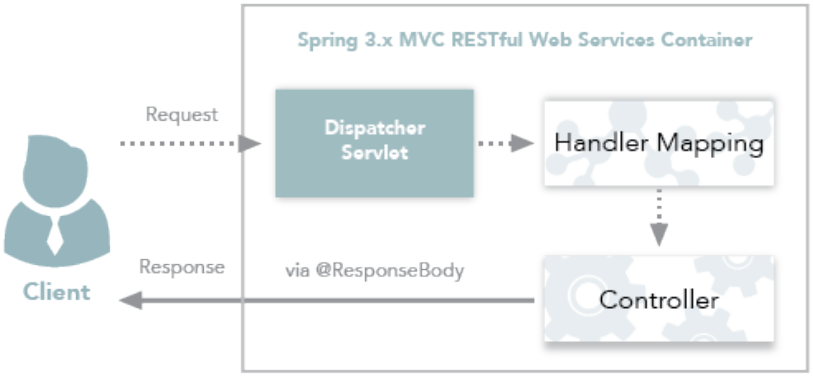
Na obrázku 4.1 je schematicky znázorněná práce aplikace, která využívá Spring MVC kontejner. Požadavek klientské aplikace je zpracováván podle návrhového vzoru Model-View-Controller. Controller na základě dotazu provádí potřebnou aplikační logiku. Výsledná data se ukládají do šablony View, a posílají se ve formě připraveného pro zobrazení dokumentu zpět v těle HTTP odpovědí.



Obrázek 4.1: Příklad MVC rozhraní v Spring frameworku [1]

Vracejí datové struktury

Na obrázku 4.2 je schematicky znázorněná práce aplikace, která vystavuje RESTové rozhraní. Na rozdíl od první varianty přispívají podobná řešení větší nezávislosti klientské části, tedy decouplingu celého systému. Dané API se lépe dokumentují, nabízejí větší prostor pro jejich návrh: od granularity poskytovaného rozhraní do možnosti výběru formátu komunikace (JSON, XML apod.). Pro určené požadavky je tato varianta nejvhodnější, protože kromě zmíněných výhod API může být používána i několika aplikacemi: hlavní a mobilní.



Obrázek 4.2: Příklad RESTového rozhraní v Spring frameworku [1]

4.2 Backend

4.2.1 Monolit nebo Mikroslužby

Backend monolitický nebo distribuovaný na několik mikroslužeb? Odpověď na otázku, který architektonický návrh ze dvou zmíněných má být použit, není zřejmá. Pro dosažení jednoznačného rozhodnutí v následujících sekcích uveďme definici každého přístupu spolu s argumentací pro a proti.

Monolit

Prakticky neexistuje jediná definice monolitické aplikace. Oficiálně je *monolit* architektonickém stylem anebo softwarovým návrhovým vzorem. Nicméně mimořádně výstižnou klasifikaci druhů monolitických aplikaci a jejich popis uvádí R.Annett ve svém článku [12] (ilustrační schémata autora: [D.4], [D.5], [D.6]). Podle něj existují tři variace monolitů:

- **Modulární monolit** se charakterizuje umístěním veškerých zdrojových kódů v jedné kódové bázi, která je kompilovaná najednou a produkuje jeden artefakt.
- **Alokační monolit**. Kód daného typu monolitu se nasazuje a doprovází nejednou. Sebraný jednou artefakt je přepravován do všech uzlů. Všechny komponenty mají ve stejný čas stejné verze bežícího softwaru.
- **Environmentální monolit**. Vždycky má jednu instanci aplikace anebo procesu, který vykonává práci celého systému.

Mikroslužby

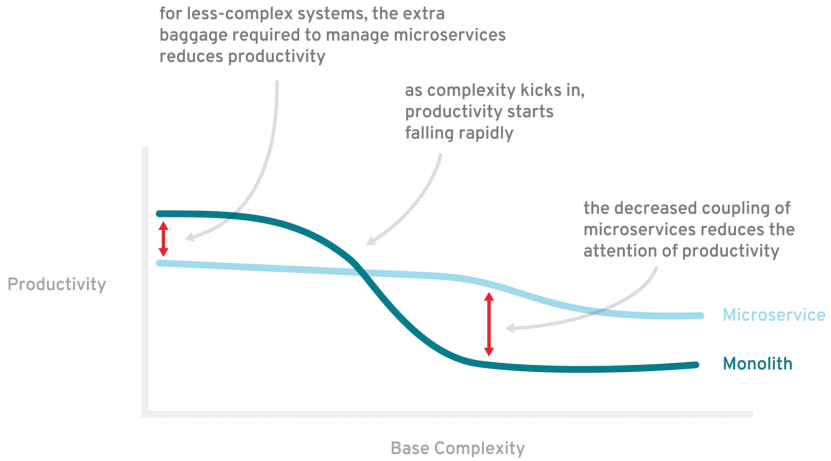
M.Fowler, velmi známý speaker a specialista v oblasti softwarového vývoje, definuje mikroslužby následovně: „Ve zkratce, mikroslužební architektonický styl je způsobem vývoje jedné aplikace ve formě sady menších služeb, každá z kterých běží ve svém procesu a komunikuje s ostatními prostřednictvím prostých mechanismů, často HTTP API. Dané služby se budují kolem své byznys způsobilosti a jsou nezávisle nasazené úplně automatizovanými nástroji. Základní minimum centralizovaného řízení takových služeb může být napsáno různými programovacími jazyky a používat různé technologie ukládání dat.“ [13]

Porovnání

Jak v knížce [14], tak i na svých webových stránkách [15] C.Richardson jmenuje výhody a nevýhody monolitických aplikací oproti mikroslužbám, které vývojáři potkávají v běžné praxi:

+ **Rychlost vývoje v raných fázích**. Na obrázku 4.3 je znázorněná závislost produktivity vývoje aplikace na její složitosti.
+ **Jednoduchost návrhu a technické realizace**. Správné rozdělení na mikroslužby není jednoduché a vyžaduje detailnější analýzu. Navíc, distribuované systémy jsou náročné i z implementačního hlediska: transakční operace, monitorování, implementace rozhraní a mnohé jiné specifické problémy mají pro ně komplikovanější řešení.
− **Ohromná komplexita zastrašuje vývojáře**. Hlavním problémem podobných aplikací je jejich složitost. Jsou příliš velké pro to, aby vývojář věděl, jak fungují všechny jejich části Nakonec, oprava chyb a implementace nové funkcionality začínají být náročné intelektuálně a časově.
− **Každodenní zpomalení práce**. Obrovské aplikace přetěžují a zpomalují vývojářská IDE. Buildování a rozběhnutí zabírá čím dále, tím více času. Ve výsledku cyklus editace-build-start-test prodlužuje a ovlivňuje produktivitu.
− **Bariéra pro agilní vývoj a nasazení**. Proces nasazení změn monolitické aplikace je dlouhý a bolestný. Většinou podobné akce probíhají jednou měsíčně v době nejmenší aktivity uživatelů systému. Mimo jiné při použití agilních metodik začínají být komplikovanými testování, stabilizace a práce s verzovacími nástroji.
− **Rozšiřitelnost systému je výzvou**. Různé aplikační moduly mají konfliktní požadavky, systém má většinou nestejnorodost.
− **Spolehlivost**. Protože všechny moduly běží ve stejné aplikaci nebo procesu, chyba jednoho modulu může způsobit výpadek celé aplikace. Nebo například únik paměti v relativně nedůležité části zničí ostatní.
− **Vyžaduje dlouhodobou věrnost technologickému stacku**. Architektura nutí vývojáře používat jeden technologický stack. Skoro není možná adaptace nových frameworků a jazyků. Pokus přepsat celou aplikaci na novou, a pravděpodobně lepší, technologii je extrémně drahý a riskantní.

Z uvedené argumentace lze usoudit, že pro stanovené účely je vhodnější mikroslužební architektura. Kromě toho, že bude plnit kvalitativní omezení `NFR_BE1`[3.3.2], navíc vyřeší sadu problémů, které budou vznikat při růstu komplexity celého systému, a bude souviset s vybranou metodou vývoje [6]. Rozdělení backendu aplikace na služby bylo provedeno v souladu s jejich byznys funkcemi: administrace průzkumů a jejich provedení.



Obrázek 4.3: Závislost produktivity vývoje aplikace na složitosti systému. [2].

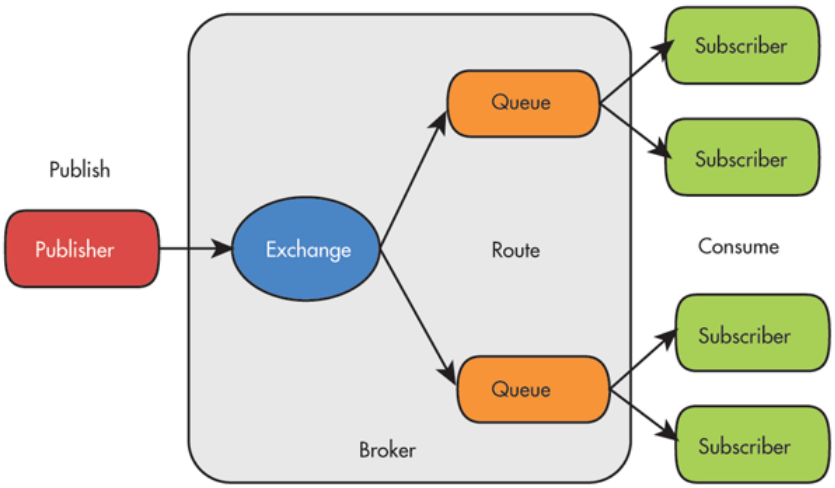
4.2.2 Komunikace mezi mikroslužby

Jelikož jsme vyčlenili dvě mikroslužby a víme, že jejich počet může růst během vývoje a přidání dalších požadavků, je velmi důležité navrhnout korektní způsob komunikace mezi nimi.

Víme, že nedostupnost jedné části systému nesmí způsobit vadné chování ostatních. Proto je zřejmé, že interakce musí probíhat asynchronně. Velmi známým přístupem k propojení oddělených částí distribuovaných systémů je organizace softwarové infrastruktury, známá jako „*Message-oriented middleware*“. Našim cílům přispěje „**Message broker**“, architektonický vzor pro validaci, transformaci a přesměrování zpráv. V aplikaci použijeme jeden z „*message brokerů*“, implementující AMQP. Obrázek [4.4] demonstruje typický úkol podobných implementací.

Vybrané řešení má následující výhody:

- Oddělení publisherů zpráv a jejich consumerů
- Možnost ukládání zpráv
- Navigace zpráv, je řízená deklarativně
- Centralizované monitorování a management všech zpráv



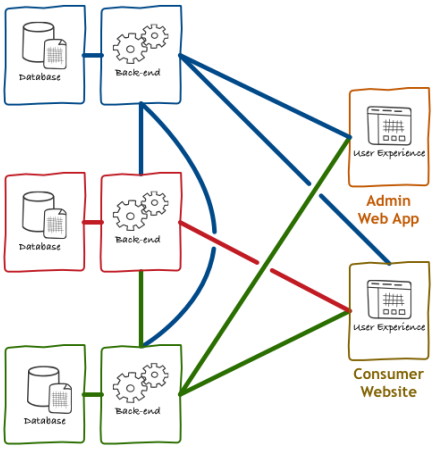
Obrázek 4.4: Znázornění práce message brokeru. [3]

4.3 Gateway

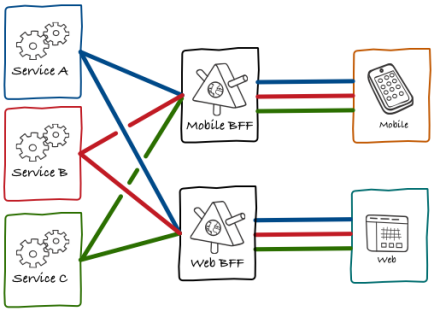
Gateway neboli *Backend for Frontend* (BFE) – tak se jmenuje architektonický návrhový vzor, který umožňuje se zbavit situace zobrazené na schématu [4.5a].
Případ webové aplikace s mnohými backendovými služby a několika klientskými aplikacemi, které se na ně obracejí, není unikátní. Přečíslíme nejzásadnější z problémů, které jsou spojené s podobnou organizací backendového API:

- Různé klientské aplikace, buď webové anebo mobilní, vyžadují různá data v různých formách. Každá z aplikací má svojí specifickou strukturu stránek, mezistránkovou navigaci, zobrazovaná data a funkce uživatelského rozhraní.
- Změna API jedné backendové služby je nerozdělitelně spojená se změnami v logice a struktuře dat všech klientů.
- Komplikovaná logika HTTP interakce a sběru dat klientské aplikace při nutnosti komunikace s několika zdroji.
- Rychlost síťové komunikace je dramaticky pomalejší, než komunikace mezi mikroslužbami běžícími v jednom prostředí. Podle statistiky, jeden dotaz do gatewaji, která agreguje potřebná data, je logaritmicky rychlejší, než několik menších dotazů do stejných mikroslužeb bez ní.

Gatawaje řeší všechny zmíněné problémy a v současné době jsou dost často používané. Příklad jejich aplikace je zobrazen na schématu [4.5b]



(a) : Rozhraní backendu bez aplikace návrhového vzoru BFE.



(b) : Rozhraní backendu s aplikaceí návrhového vzoru BFE.

Obrázek 4.5: Demonstrace návrhového vzoru BFE. [4]

4.4 Frontend

Výsledný systém bude obsahovat dvě klientské aplikace: webovou aplikaci pro generaci průzkumů, analýzu sebraných dat a řízení projektů a webovou mobilní aplikaci pro sběr uživatelských odpovědí.

Cíl práce stanoví nutnost návrhu systému s ohledem na jeho následující rozšíření o nativní mobilní aplikace. V souladu s daným požadavkem bylo rozhodnuto o použití multiplatformního frameworku, který umožňuje vývoj jedné aplikace kompilované jak pod operační systémy Android a IOS, tak i pro rozběhnutí ve formě webové aplikace.

Výhodou vybraného řešení je jeho implementační jednoduchost: aplikaci pro tři platformy je možné napsat za použití stejných s hlavním frontendem technologií, bez ohledu na implementační detaily, které jsou unikátní pro každou platformu.

Nevýhodou je především výkonnost podobných aplikací, která nás zajímat nebude, protože žádné grafické a jiné náročné operace provádět nepotřebujeme. Mezi jiné je známá nedokonalost integrace aplikací s hardware prvky mobilních zařízení, jako jsou accelerometer, GPS a další. Tento problém není aktuální taky.

4.5 Autorizace a autentizace

4.5.1 Vstupní podmínky

Návrh systému autorizace a autentizace je ve velké části ovlivněn architektonickým návrhem celé aplikace, funkcionálními a nefunkcionálními požadavky.

- **Bezpečnost**. Řešení musí být důvěryhodné a obsahovat téměř základní bezpečnostní prvky, mít rozumné vlastnosti ochrany proti zneužití systému a sledovat obecně přijaté praktiky zabezpečení síťové komunikace.
- **Osvědčený postup**. Řešení musí být výstižným a jednoduchým implementačně pro každou backendovou mikroslužbu a frontendové aplikace pro různé platformy. Za nejlepší způsob aplikace považujeme konfiguraci existujících, dobře dokumentovaných postupů.
- **Administrace uživatelských prav**. Řešení musí podporovat flexibilní nastavení a odebírání uživatelských rolí. Za výhodu považujeme přítomnost uživatelského rozhraní pro technické a administrační účely.
- **Podporovatelnost a rozšiřitelnost**. Řešení musí být navrženo tak, aby se dalo jednoduše provozovat a udržovat, případně vyvíjet pro další byznys cíle.
- **Uživatelská přívětivost**. Výhodou bude podpora moderních způsobů přihlášení, na které jsou cílové uživatele zvyklí: přihlášení pomocí účtů od sociálních sítí, SSO autentizace apod.

4.5.2 JWT token

JWT je otevřeným veřejným standardem, který definuje kompaktní a soběstačný způsob bezpečného předávání informace mezi komunikujícími stranami ve formě JSON objektů.

- Kompaktní: Díky svému malému rozměru můžou být JWT přeposílané jako parametr URL, parametr tělesa POST dotazu anebo v HTTP hlavičce.
- Soběstačný: hlavní část tokenu obsahuje uživatelská data a redukuje počet potřebných dotazů k databázi do jednoho.

Předávaná informace může být ověřená a mít důvěru, protože je digitálně podepsaná. Podepsané tokeny slouží k verifikaci integrity tzv. prohlášení, které token obsahuje. V

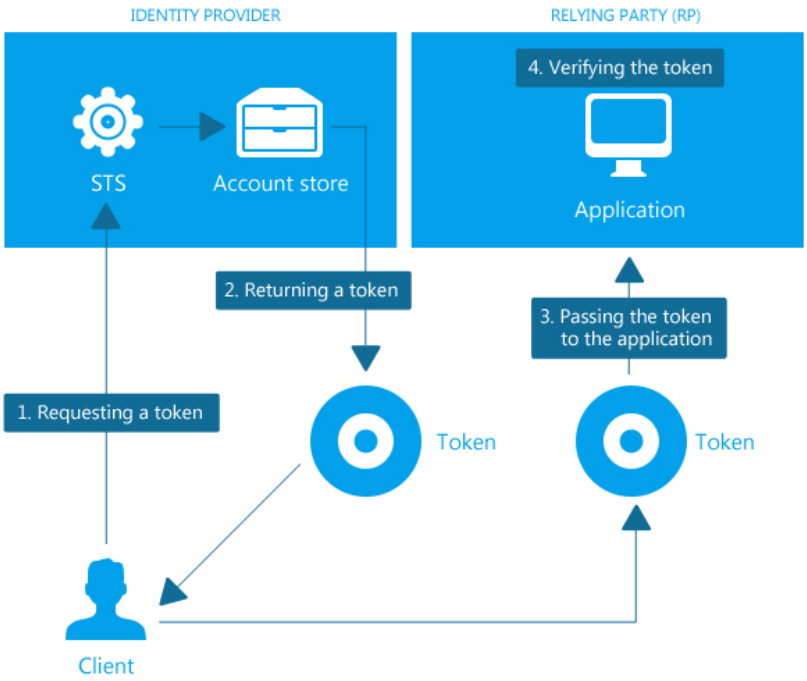
případě podepsání tokenů dvojicí privátních a veřejných klíčů, jejich signatura taky svědčí o tom, že pouze strana udržující privátní klíč mohla podepsat token.

JWT může obsahovat nejrůznější atributy vztahující se k uživatelským datům, jako jsou: jméno uživatele, emailová adresa, avatar apod. Dané atributy jsou zmiňované jako „prohlášení“ (en: „Claim“) a můžou být získané z tokenů aplikacemi.

4.5.3 Token-based autentizace

Definice seznamu požadavků k autentizačnímu systému nám umožnila dojít k závěru, že nejvíce vyhovujícím je postavení procesu autentizace a autorizace na předávání tokenů. Základem podobných řešení je doprovázení podepsanými tokeny veškerých síťových dotazů od klientských aplikací do API backendových služeb pro účely verifikace uživatelských práv na provádění té či jiné operace.

Schéma [4.6] demonstruje způsob autorizace dotazů backendového API pomocí přeposílaného tokenu, který klientská aplikace vyměňuje za heslo s poskytovatelem autentizace.



Obrázek 4.6: Zabezpečení komunikace mezi klientskou aplikací a backendem za pomoci JWT tokenů. [5]

4.6 Výsledný architektonický návrh

Na závěr kapitoly uveďme diagram agregující veškeré architektonické návrhové řešení a připomínky. [D.3] - výsledný architektonický návrh celého systému.

Kapitola 5

Implementace

5.1 Backend

Jazykem, na kterém byl napsán backend, je Java verze 8. Výběr je očividný: jiné takové technologické platformy, která by měla podobnou rozsáhlost v softwarovém segmentu podnikových aplikací, zatím není.

5.1.1 Spring Boot

Spring je v současné době nejznámějším Java frameworkem, který před několika lety začal vytlačovat ostatní známé poskytovatele Java EE specifikace především díky své jednoduchosti oproti konkurentům. Od té doby se dramaticky zvětšil počet projektů, oborů a technologií, které daný open-source projekt zahrnuje. Je o něm napsáno velké množství knih, popisujících jak jeho problémy, tak i nepochybné přínosy, proto v práci uvedeme pouze definici frameworku:

> „Framework Spring poskytuje vyčerpávající programovací a konfigurační model pro moderní Java-based podnikové aplikace s možností nasazení v libovolném prostředí. Klíčovou vlastností Spring je infrastrukturní podpora na aplikační úrovní: Spring se zaměřuje na „instalatérství“ podnikových aplikací tak, aby se týmy mohly soustředit na byznys logiku aplikace, bez zbytečného svázání specifickými prostředími.“ [16]

Backendové služby budeme stavit na „lehčí“ pro vývoj verzí Spring – Spring Boot, který pro ně poslouží jako kostra. Takové aplikace jsou připravené k práci víceméně hned po inicializaci projektové struktury. Ostatní pod-frameforky Spring, které jsou v aplikaci použité, popíšeme v souvisejících sekcích dané kapitoly.

5.1.2 Autorizace

Všechny služby mají zabezpečené veřejné API. Identifikace uživatelů a autorizace jejich práv z předávaných v HTTP hlavičce JWT tokenů (více v [4.5.3]) je zajišťovaná frameworkem Spring Security.

Kromě toho, že řeší mnohé bezpečnostní problémy, je velmi snadně konfigurovatelný. Celé nastavení bezpečnostního systému je deklarativní a umisťuje se na méně než čtyřiceti řádcích kódu: [1]. Ověření přístupu uživatele v té či jiné roli, nebo kontrola stavu jeho připojení jsou možné dvěma způsoby:

- Určení specifických adres a povolených operací v konfiguračním adaptéru Spring Security [1].
- Zadání podmínek pro exekuci bloků kódu za pomoci anotací *@Preauthorize*, *@Postauthorize* a jazyku SpEL. Například: *@PreAuthorize(„isAuthenticated()“)*

5.1.3 Rozhraní

Reprezentační vrstva se skládá z *kontrolerů*, *vstupního modelu*, *výstupního modelu* ve formě Resource a *assemblerů*, které mají za úkol transformaci domain modelu na výstupní model.

Všechny kontrolery dohromady tvoří REST API webové služby, kde každý z nich je implementací Spring REST kontroleru a skládá se z mapování HTTP požadavků na aplikační funkce. Příklad zaktualizování průzkumu je znázorněn na obrázku [2]. Požadavek je identifikován podle relativní cesty a HTTP metody. V aplikaci pro provedení CRUD a příkazových operací jsou využívané metody GET, POST, PATCH a DELETE.

Dokumentace

Pro splnění nefunkčního požadavku `NFR_BE2`[3.3.2] je nastavená dokumentace celého API všech služeb. Jednoduchým a dobře konfigurovatelným řešením, které generuje celou dokumentaci a nabízí interaktivní webové rozhraní pro dotazování služeb, je Swagger.

Swagger se dobře integruje do Spring aplikace a pro definici základní dokumentace vyžaduje pouze přidání metadat ke všem kotrolerům, jejich metodám a modelům. Taková informace je předávaná prostřednictvím anotace, je dobře čitelná a neznečišťuje kód (naopak začíná být srozumitelnější).

Po rozběhnutí služby Swagger je dostupný na adrese *api/swagger-ui.html*. Snímky dokumentace jsou k dispozici v příloze: [F.1] a [F.2].

5.1.4 Byznys logika

Byznys logika se soustředí ve služební vrstvě a domainových entitách. V podstatě se jedná o přechodovou variantu mezi klasickou třívrstovou aplikací a aplikací psanou v DDD. Aplikace nebyla vyvinuta v klasickém DDD, protože na etapě implementace prototypu by vyžadované DDD vzory (CQS, CQRS, Event Sourcing a další) mohly zbytečně zkomplikovat celý systém.

Služební vrstva je zastupovaná servisními komponenty, které transakčně zpracovávají nedoménové operace, jako je například vytváření entit, validace práv apod. Doménová logika naopak kontroluje konzistenční stav entit a jejich životní cyklus. Příklad zaktualizování průzkumu je znázorněn na obrázcích [3] a [4].

Posílání zprav

V návrhu [4.2.2] bylo vybráno posílání zpráv, jako způsob asynchronní komunikace mezi mikroslužbami. Pro její implementaci jsme využili integrační servis ActiveMQ, který plně podporuje specifikaci Java Messaging Services. Příklad posílání zpráv mezi službami je znázorněn na obrázku [5].

5.1.5 Repozitáře, domain model a databáze

V dané práci je používán velmi mocný open-source systém řízení databází DBMS Postgres. Postgres se vztahuje k object-relational systémům, ale to, který relační databázový systém je využíván, na backendovou implementaci žádný vliv nemá. Pro to, abychom mohli zapouzdřit závislou na platformě interakci, jsme využili ORM framework Spring Data, který zároveň poskytuje implementaci specifikaci Java Persistance Api.

Spring Data mimo jiné umožňuje mapování struktury doménových objektů na relační strukturu databáze a sestavování chytrých dotazů pomoci JPQL. Výhodou Spring Data je snadná integrovatelnost do Spring projektu a redukce psaného kódu. Příklad toho, jak kompaktní je doménová entita průzkumu a její repozitář, je znázorněn na obrázku [6].

5.1.6 Pomocné knihovny

Lombok

Pro zbavení se boilerplate kódu se velmi dobře hodí projekt Lombok. Výsledně byly použité následující anotace:

- `@Getter` `@Setter` pro generaci getterů a setterů POJO objektů.
- `@AllArgsConstructor` `@NoArgsConstructor` pro generaci standardních konstruktorů entit.
- `@Builder` generuje implementaci návrhového vzoru Builder pro danou entitu.
- `@EqualsAndHashCode` pro generaci bázových metod *equals* a *hashCode* za použitím polí entity.
- `@ToString` pro generaci bázové metody *toString* za použitím polí entity.
- `@Slf4j` pro generaci statické a konstantní instance loggeru Slf4j.

5.2 Gateway

V rámci vývoje prototypu gatawaje implementované nebyly. Hlavní příčinou tomu bylo časové omezení a poměrně nízká priorita architektonického řešení [4.3] oproti většině byznys požadavků.

Pro současnou implementaci nepřítomnost gatewaji způsobuje snížení rychlosti interakce mezi klientskou aplikací a backendem: agregace výsledků síťových dotazů je zatím realizovaná v logice frontendů, každý z kterých volá jednotlivé mikroslužby backendu.

Nicméně chceme zmínit, že v případě budoucí implementace se předpokládá, že gatewaje budou plnit funkce *jističe* (Circuit Breaker) spolu s funkcí monitorování komunikace s backendy. Podrobněji si o podobných praktikách lze přečíst ve článcích: [17] [18].

5.3 Frontend

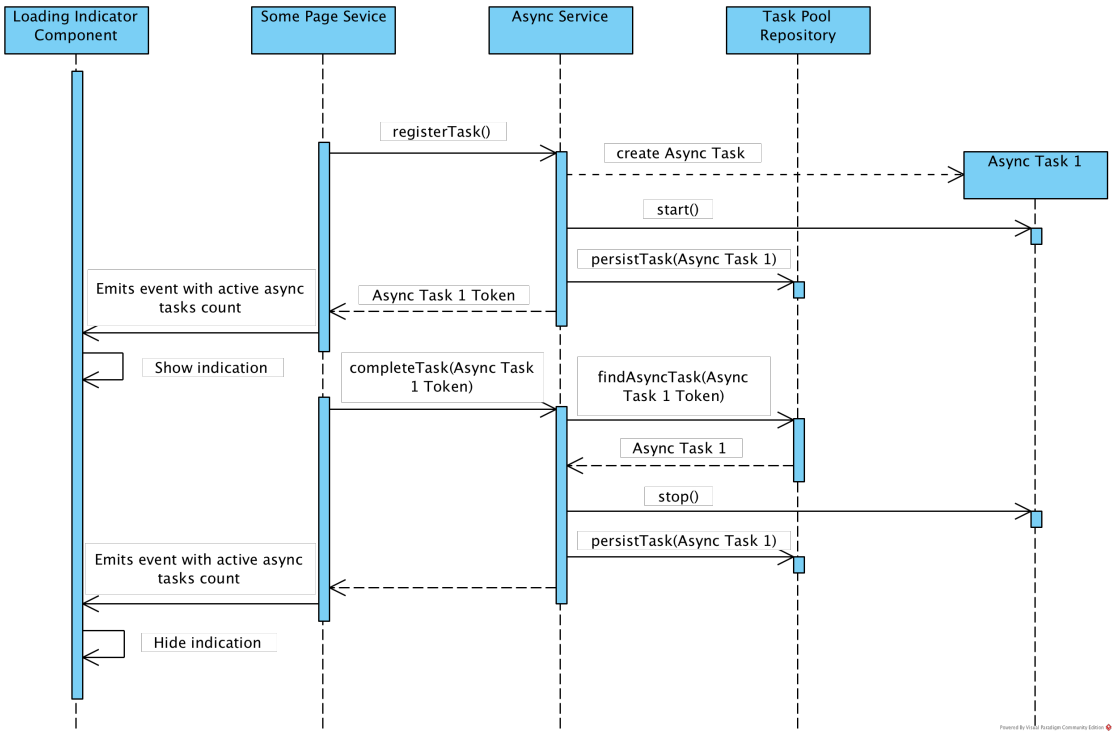
Pro napsání soběstačné jednostránkové webové aplikace (SPA) byla využita moderní open-source platforma – framework Angular 2+ postavená na Typescript. Velmi flexibilní architektura vyvíjených aplikací spolu s velkou sadou základních nástrojů a typizovaným jazykem Typescript umožňuje stavět složitější, dobře zabezpečené a lépe podporované systémy.

5.3.1 Autentikace a authorizace

Autentizační logika je soustředěná v AuthService, který využívá standardní knihovnu Auth0 - auth0-js. Veškerou interakci s API Auth0 zapouzdří tato knihovna podle algoritmu, popsáného v kapitole [5.5].

Po získání uživatelského profilu a tokenu autentizace se data zapisují do Local Storage prohlížeče tak, aby se neztrácela po obnovení stránky nebo uzavření aplikace. Chránění informací přihlášeného uživatele je důležité pro provedení operací:

- **Posílání autorizační informace spolu s HTTP požadavky**. HTTP Interceptor zachycuje všechny dotazy před jejich odesláním na backend, přidává nutnou informaci a přeposílá dále. Demonstrace kódu je k dispozici v příloze [7].



Obrázek 5.1: Sekvenční diagram zpracovávání asynchronních operací.

- **Kontrola povolení přístupu k určitým stránkám**. V aplikaci jsou definované tak zvané routingy – přiřazení stránkovým komponentům aplikace adresy, na které se budou zobrazovat. Pro kontrolu, zda má uživatel povolený přístup k routingu, je používán AuthGuard. Demonstrace kódu je k dispozici v příloze [8].

5.3.2 Asynchronní operace

Základem informativního a přívětivého uživatelského rozhraní je indikace vypracování asynchronních operací: síťových dotazů, náročných výpočtů a pomalých read-write procesů.

Proto, aby daná logika byla implementovaná v jednom místě, jsme realizovali asynchronní službu AsyncService, která je schopná registrovat operace a uzavírat je. AsyncService používá TaskPoolRepository pro ukládání entit typu AsyncTask a reaktivně poskytuje informaci o počtu jejich aktivních instancí. Tato informace je důležitá pro vizuální komponentu indikace asynchronních operací: při počtu větším než nula přechází do aktivního stavu. Popsaný algoritmus je znázorněn na obrázku [8].

5.3.3 Pomocné knihovny

RxJS

RxJS je knihovna pro reaktivní programování, které používá Observable pro zjednodušení kompozice asynchronního a callback-based kódu. Detailnější popis projektu RxJS je dostupný na webových stránkách projektu [1].

Bootstrap

Ve větší míře je projekt stylizován pomoci knihovny Bootstap a její odvětví pro Angular 2+ – NgBootsrap. Obě dvě nabízí vizuální komponenty připravené pro použití, adaptivní souřadnicový systém a další nástroje stylizace HTML stránek. Detailnější popis použitých projektů je dostupný na webových stránkách projektů [2] [3].

Markdown

Formátovaný text stránek průzkumů je uložený a přeposílá se v textové podobě. Proto, abychom nevytvářeli novou syntaxi formátovaného textu, specifickou pro danou aplikaci, a nevytvářeli svůj formátovací nástroj, bylo rozhodnuto využít Markdown.

Editační dialog pro Markdown je zkonfigurovaným komponentem knihovny covalent/text-editor [4]. Pro zobrazení formátované verze textu v průzkumech se používá další knihovna, angular-bootstrap-md [5].

[1] RxJS: reactivex.io/rxjs/
[2] Bootstrap: getbootstrap.com
[3] NgBootsrap: ng-bootstrap.github.io
[4] Covalent MD editor: npmjs.com/package/@covalent/text-editor
[5] Angular Bootstrap MD: npmjs.com/package/angular-bootstrap-md

5.4 Mobilní frontend

V návrhu [4.4] bylo zdůvodněno použití multiplaforemního frameworku pro implementaci mobilní aplikace. V této kapitole vybereme, který framework nejlépe vyhovuje určeným potřebám, a uvedeme implementační detaily jeho použití.

5.4.1 Ionic vs. React Native

Svatá válka „Java nebo C++“ má stylově podobné moderní pokračování: aktuálně na webu existuje velké množství názorů, zda jsou multiplatforemní aplikace pokrokem nebo neopodstatněným zjednodušením mobilního vývoje, které způsobuje více problémů, než jich řeší. Ale všichni se schodují na tom, že opravdu konkurují Ionic a React Native.

Žádný z nich není lepší než ostatní. Každý framework má své silné stránky a je přizpůsoben pro řešení specifických problémů. Sice React Native ukazuje vyšší efektivitu, protože kompiluje aplikaci do nativního kódu a používá nativní komponenty místo frameworku Cordova, ale cílům dané práce a technologickému stacku Ionic vyhovuje více. Ionic podporuje vývoj Angular 2+ aplikací a rychlý testovací cyklus (nevyžaduje emulator nebo fyzické zařízení).

5.4.2 Odlišení od hlavního frontendu

Od hlavní webové aplikace se mobilní webová aplikace skoro neliší implementačně. Obě dvě používají framework Angular 2+ a jsou psané v jazyce Typescript. Většina služeb má úplně stejnou implementaci, rozdělení do vrstev se nemění, datový model a struktura komponent se doslova shodují.

Existují ale i drobná odlišení. Například, Ionic projekty mají jinou souborovou strukturu a nepodporují Angular navigaci: Ionic používá zásobník a umožňuje URL odkazování určitých stránek pouze pomoci mechanizmu deeplinking.

5.5 Poskytovatel autentizace

Hlavní kritéria pro výběr určité služby, poskytující autentizační mechanismus, byla definována v návrhu [4.5]. Po vyhledávání jak hotových produktu (SaaS), tak i konfigurovatelných systémů, byl vybrán Auth0 [6].

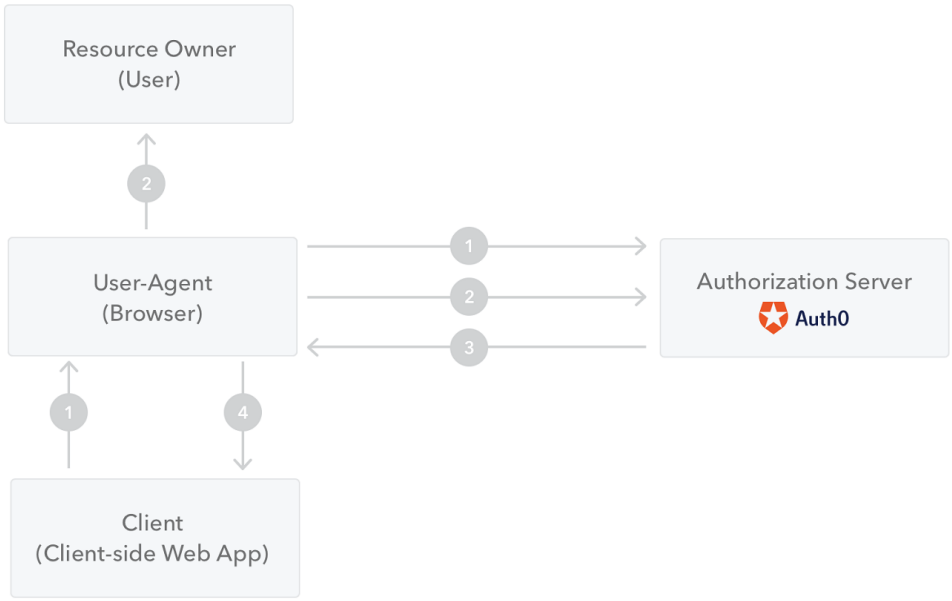
Služby Auth0 nejsou bezplatné, když se jedná a poměrně velké systémy, ale existují podmínky bezplatného využití, které pro potřeby jak vývoje, tak i menší webové aplikace, budou nadmíru stačit.

Auth0 nabízí velký počet administračních a monitorovacích nástrojů, hodících se naším cílům a překrývajících všechny možné budoucí požadavky. Kromě detailní dokumentace svých produktů jsou k dispozici i nejrůznější učební materiály o bezpečnosti webových aplikací.

Po shrnutí dokumentace schéma standardního využití služeb Auth0 pro získání autentizačního tokenu [5.2] můžeme popsat následovně [6]:

1. „Authorization Code flow“ je spuštěn přesměrováním uživatele prohlížečem na autorizační stránku Auth0.
2. Následujícím krokem Auth0 zobrazí zámkový dialog, který umožňuje uživateli vyplnit přihlašovací údaje nebo alternativně se přihlásit pomocí jednoho z povolených Identity Provideru (Google, sociální sítě a další).
3. Po autentizaci je uživatel přesměrován zpět na aplikační stránku, známou jako „Redirect URI“ anebo „Callback URL“. Pomocí přidání parametru do přesměrovací adresy je kód předáván do aplikace.
4. Klientská aplikace vyměňuje získaný kód za podepsaný token síťovým dotazem služby Auth0.

[6] Auth0: https://auth0.com



Obrázek 5.2: Auth0. Authoriztion Code Flow: výměna tokenů pro autorizaci klient-server dotazů. [6]

Kapitola 6

Organizace práce

6.1 Motivace

Před začátkem práce jsme se zamysleli nad organizací práce na projektu. Mezi ostatní vstupní podmínky patřila jedna, která nás nejvíce vedla k výběru vhodné metodologie organizac – omezený čas. Proto jsme před tím, než jsme začali vývoj, jsme vzali v úvahu několik objektivních faktorů:

- Vývoj softwaru je částí průzkumu: cíle se můžou měnit na základě mezivýsledku bez zásadních úbytků.
- Analytickému vypracování projektu bylo věnováno významně málo času v porovnání s architektonickou složitostí návrhu.

Výsledně bylo rozhodnuto pro organizaci využít agilní metodiky vývoje, které umožňují rychlý vývoj software a přispívají k reakci na operativní změny požadavků v průběhu vývojového cyklu. Po průzkumu a analýze různých agilních metodologií jsme se rozhodli počítat iterativní a inkrementální framework Scrum jako nejvhodnější variantu.

> **Agilní metodiky** jsou skupiny metod původně určené pro vyvíjení softwaru a založené na iterativním a inkrementálním vývoji. Umožňují rychlý vývoj softwaru a zároveň dokáží reagovat na změnu požadavků v průběhu vývojového cyklu. Podle těchto metodik se správnost systému ověří jedině pomocí rychlého vývoje, předložení zákazníkovi a následných úprav dle zpětné vazby. [19]

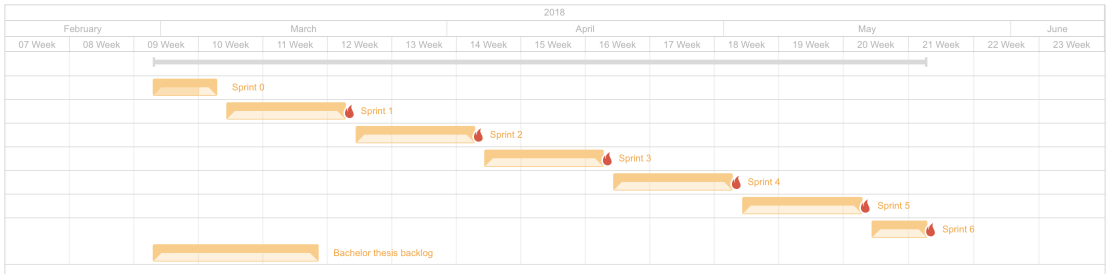
Scrum je procesním frameworkem, používaným pro řízení projektového vývoje a jiné intelektuální práce. Scrum je empirický, protože pomáhá týmům stanovit hypotézu, jak něco funguje podle jejich názoru, ji ověřit, zamýšlet se nad výsledkem a provést vhodné úpravy. Tak se správně využívá framework. Scrum je strukturován tak, že umožňuje týmům do něj zařazovat praktiky jiných frameworků, které v jejich kontextu dávají smysl. [20]

6.2 Popis procesu vývoje

Prostudovali jsme různé praktiky a způsoby uplatnění Scrum a dohodli se, jak jej budeme využívat během vývoje, které procesy se nám budou hodit.

6.2.1 Iterace

Časový úsek mezi začátkem práce a jejím ukončením jsme rozdělili na iterace se stejnou dobou trvání (13 dnů) – sprinty:



Obrázek 6.1: Organizace práce. Rozdělení na sprinty.

- Sprint 0 (28.02.2018 – 06.03.2018): Technický dluh
- Sprint 1 (08.03.2018 – 20.03.2018): Správa uživatelských údajů a účtů
- Sprint 2 (22.03.2018 – 03.04.2018): Generace a editace dotazníků
- Sprint 3 (05.04.2018 – 17.04.2018): Mobilní aplikace
- Sprint 4 (19.04.2018 – 01.05.2018): Mobilní aplikace
- Sprint 5 (03.05.2018 – 15.05.2018): Testování aplikace, oprava chyb
- Sprint 6 (17.05.2018 – 22.05.2018): Konečné opravy dokumentace

Sice každý ze zaplánovaných sprintů měl předurčené téma, ale využití Scrum nám umožnilo to, že náplň každého z nich se může změnit podle aktuálních potřeb a prostředků.

6.2.2 Projektový backlog

> **Projektový Backlog** je seřazený seznam uživatelských potřeb a zadání, které jsou v produktu známy. Jedná se o jediný zdroj požadavků na jakékoli změny, které je třeba provést v produktu. [19]

Před začátkem projektu jsme naplnili projektový backlog souhrnem User story, které v našem případě odpovídaly seznamu požadavků z analytické části.

Bachelor thesis backlog	0%			
FR_A1 Systém musí umožňovat registraci uživatele	0%	Open	High	
FR_A2: Systém musí umožňovat registrací klientského účtu	0%	Open	High	
FR_A3: Systém musí umožňovat udělení a odebírání přístupu uživatele ke klientskému účtu	0%	Open	Medium	
FR_A4: Systém musí rozlišovat práva uživatelů pro operace v rámci jednoho účtu	0%	Open	Low	
FR_G1: Systém musí umožňovat vytvářet, odstraňovat a editovat prototypy dotazníků	0%	Open	High	
FR_G2: Systém musí ukládat každou operaci, kterou uživatel provede s prototypem	0%	Open	Low	
FR_G3: Systém musí umožňovat publikaci prototypu do veřejné části, po které již editovatelný není	0%	Open	High	
FR_G4: Systém musí umožňovat konfiguraci dotazníků ve formách poll a survey	0%	Open	Medium	
FR_G5: Systém musí umožňovat stylizaci dotazníků alespoň třemi různými způsoby	0%	Open	High	
FR_G6: Systém musí umožňovat přidání obrázků a odkazů na externí zdroje do dotazníků	0%	Open	Medium	
FR_G7: Systém musí umožňovat přidání základních druhů validace dotazníků	0%	Open	Medium	
FR_G8: Systém musí umožňovat definici logických pravidel, podle kterých se dotazník bude chovat dynamicky (větvení)	0%	Open	High	
FR_S1: Systém musí pro opublikovaný prototyp vytvořit odpovídající dotazník	0%	Open	Medium	
FR_S2: Systém musí umožňovat zahájit / zastavit / pokračovat / ukončit sběr uživatelských odpovědí pro určitý dotazník	0%	Open	Medium	
FR_S3: Systém musí umožňovat odpovědět na dotazník jak anonymnímu uživateli, tak i uživateli, který zatím účet nemá	0%	Open	Medium	
FR_S4: Systém musí umožňovat prohlížet seznam výsledků a jednotlivé odpovědí uživatelů	0%	Open	High	
FR_S5: Systém musí umožňovat generaci reportu výsledků ukončeného dotazníku ve formě tabulí a grafů	0%	Open	Medium	
FR_S6: Systém musí umožňovat export reportů ve formátech CSV a PDF	0%	Open	Low	
FR_S7: Systém musí umožňovat přístup k reportů pomoci vygenerovaného odkazu	0%	Open	Lowest	
FR_MOB1: Systém musí umožňovat export dotazníků do souboru, který je možné nahrát do mobilní aplikaci	0%	Open	Highest	
FR_MOB2: Systém musí podporovat reprezentaci dotazníků na mobilních zařízeních	0%	Open	Highest	
FR_MOB3: Systém musí podporovat zběr odpovědí na mobilních zařízeních	0%	Open	Medium	
NFR_1 Systém musí být snadně spistitelný (containerizace)	0%	Open	High	

Obrázek 6.2: Organizace práce. Projektový backlog.

6.2.3 Fáze

Před začátkem každého sprintu jsme se setkali v rolích vlastníka produktu a týmu vývojářů a provedli následující události:

- **Sprint review**. Tým demonstroval práci splněnou během sprintu, provedl stručnou analýzu dosažených výsledků s vizualizací: burndown chart, cumulative chart.

  > Sprint review se koná na konci sprintu, aby se zkontroloval přírůstek a v případě potřeby přizpůsobil projektový backlog. Během sprint review tým se stakeholdery probírá, co bylo během sprintu splněno. Na základě tohoto a jakýchkoli změn v produktovém backlogu účastníci formují backlog následujícího sprintu. [20]

- **Backlog grooming**. Vlastník produktu obnovil prioritu každé user story v projektovém backlogu.

  > Backlog grooming je fáze, během které vlastník produktu a někteří nebo všichni členové týmu obnovují backlog proto, aby si byli jisti, že backlog obsahuje náležité zadání, že jsou upřednostňované a že položky v horní části backlogu jsou připraveny k dalšímu vypracování. [19]

- **Sprint planning**. Tým provedl dekompozici user story a zahrnul zadání do backlogu následujícího sprintu v souladu s jeho kapacitou.

  > Náplň práce, která má být provedená během sprintu, je definovaná na plánování sprintu. Tento plán vytváří společně celý Scrum tým. [20]

Kapitola 7

Návod k rozběhnutí aplikace

7.1 Zdrojové kódy

Pro zachování historie vývoje a verzování zdrojových kódů jsme použili distribuovaný systém pro zprávu verzí Git. Využívali jsme Git v souladu s Gitlab Flow - souhrnem doporučení a obecné přijatých praktik vedení Git repozitářů.

Hlavním repozitářem Git dané práce je ThesisGeneralModule, který má následující strukturu:

1. **ResearchCompositionService**. Git submodul, který je odkazem na repozitář backendové služby ResearchCompositionService.
2. **ResearchExecutionService**. Git submodul, který je odkazem na repozitář backendové služby ResearchExecutionService.
3. **ResearchCreatorDesktopFrontend**. Git submodul, který je odkazem na repozitář frontendové aplikace ResearchCreatorDesktopFrontend.
4. **ResearchCreatorMobileFrontend**. Git submodul, který je odkazem na repozitář frontendové aplikace ResearchCreatorMobileFrontend.
5. **ThesisText**. Git submodul, který je odkazem na repozitář s textem bakalářské práce.
6. *docker-compose.yml*. Deklarativní konfigurační soubor pro rozběhnutí všech komponent za pomocí Dockeru.
7. *.gitmodules*. Git soubor pro definici nastavení submodulů.

Pro stáhnutí celé kódové báze použijte příkaz:

```
git clone --recursive <odkaz na ThesisGeneralModule>
```
[1]

7.2 Poskytovatel autentizace

Poskytovatel autentizace Auth0 už byl popsán v kapitole [5.5], ve které jsme zdůvodnili, proč je vhodným řešením a jaké výhody má. Teď postupně popíšeme, jak nastavit jeho služby pro začátek použití:

1. Založit si uživatelský účet na stránce auth0.com/signup
2. Zaregistrovat webovou aplikaci. Detailní návod je k dispozici v článku auth0.com/docs/quickstart/spa/angular2.
3. Do konfiguračního souboru src/environments/environment.ts webové aplikace uvést paramentry zadané při registraci: clientID, domain a callbackURL.
4. Zaregistrovat backendové API. Detailní návod je k dispozici ve článku auth0.com/docs/quickstart/backend/java-spring-security.
5. Do konfiguračního souboru application.properties obou mikroslužeb uvést paramentry zadané při registraci: auth0.issuer a auth0.apiAudience.

7.3 Kontejnerizace

Pro nasazení komponent do jakéhokoliv prostředí jsme propracovali jeden společný konfigurační soubor a několik pomocných pro náročnější komponenty, které vyžadují složitější nastavení. Všechno – tedy dvě databáze, dvě backendové mikroslužby, jeden Message Broker, webová aplikace a mobilní webová aplikace – je spouštěno jedním příkazem.

To je možné teprve krátkou dobu, od roku 2013, díky Dockeru. Docker se od té doby šíří a stává se stále populárnější platformou mezi specialisty sféry softwarového vývoje. Na jeho stránkách je uvedená definice kontejnerových systémů:

[1] Odkaz na hlavní repozitář: https://gitlab.com/ogoltnik_bachelor/ThesisGeneralModule (je nutné mít přístup)

> **Platforma kontejnerů** je kompletní řešení, které organizacím umožňuje vyřešit více problémů v různých požadavcích. Jedná se spíše o technologii a orchestraci – poskytuje trvale udržitelný přínos v celé organizaci tím, že poskytuje všechny součásti, které podnik vyžaduje, včetně zabezpečení, správy, automatizace, podpory a certifikace v průběhu celého životního cyklu aplikace. [21]

Kromě všech jeho výhod, je také velmi mocným nástrojem pro organizaci mikroslužebních aplikací: Docker kontejnery jsou lehké a ideální pro vývoj systémů distribuovaných na menší části. Zrychlují vývoj, nasazení a zvrácení desítek a někdy i stovek kontejnerů skládajících jednu aplikaci. Snadno použitelné nástroje usnadňují vytváření, nasazení a údržbu složitých Docker-systémů.

V příloze je uveden kód souboru *docker-compose*, který popisuje všechny komponenty a jejich komunikaci [9] [10]. Aby byl jedním příkazem rozběhnut celý systém, je potřeba si nainstalovat Docker a Docker-Compose podle návodu z domovských stránek produktu.

Po stáhnutí zdrojových kódů je možné rozběhnout aplikaci jedním příkazem:

```
docker-compose up
```

Adresa hlavní webové aplikace po spuštění:
`localhost:4200`

Adresa mobilní webové aplikace po spuštění:
`localhost:8100`

Adresa dokumentace API služby generace průzkumů po spuštění:
`localhost:8080/api/swagger-ui.html`

Adresa dokumentace API služby sběru odpovědí po spuštění:
`localhost:8090/api/swagger-ui.html`

Kapitola 8

Testování

V následující kapitole bude popsané testování některých nejdůležitějších případů užití. Testování jsme provedli pro kontrolu uživatelské přivětivosti aplikace a souladu implementace s očekáváními uživatelů. Pro ověření výsledků implementace byli vybraní účastnicí testování. Každý participant měl k dispozicí detailní popis všech případů užití a zkusil využít aplikaci v souladu s danou instrukcí. Na závěr testování jsme zaregistrovali reakci všech účastníků a podrobili ji analýze.

8.1 Výběr účastníků

Před začátkem testování jsme definovali vlastnosti participantů a jejich rozdělení na skupiny. Přesné určení vyhledávacích podmínek nám umožnilo najít lidi, kteří reprezentují širší spektrum uživatelů spadajících do cílové skupiny.

8.1.1 Screener

Účastníky jsme rozdělili do dvou skupin podle základních rolí systému: administrátoři účtů a průzkumů a účastnící průzkumů. Pro lepší představu o schopnostech a očekáváních participantů, jsme také chtěli vědět o zaměření jejich práce nebo vzdělávání: technické, humanitní apod. Za důležitou podmínku jsme také uvažovali minulou zkušenost člověka s účastí v průzkumech a na jejich vytváření.

Výsledný screener pro odběr účastníků měl následující formu dotazníku:

1. Je Vaše každodenní činnost spíše spojena s humanitními vědami, nebo technickými? (Humanitní / Technické)
2. Zúčastnil jste se někdy online průzkumu? (Ano / Ne)
3. Vytvářel jste někdy online průzkumy a prováděl je? (Ano / Ne)
4. Jste starý(á): (méně než 18 let, 19-30 let, 30-40 let, 41-50 let, více než 50 let)

8.1.2 Účastníci

Na základě výsledků screeneru jsme vybrali osm lidí. Do skupiny účastníků připadli čtyři – kartézský součin 1) a 2). Do skupiny tvůrců taky patřili čtyři lidé - kartézský součin 1) a 3). Čtvrtá otázka posloužila k filtraci účastníků, kteří jsou mladší než 18 let nebo starší než 50 let.

8.2 Scénáře testování

8.2.1 TC1: Autentizace

Daný scénář testuje případy užití, spojené s autentizačními procesy registrace, přihlášení, odhlášení a změny hesla v systému vytváření průzkumů:

1. Přejděte na domovskou stránku webové aplikace.
2. Vytvořte v systému uživatelský profil.
3. Po přihlášení do systému se odhlaste.
4. Přihlaste se do systému pomocí aktuálních přihlašovacích údajů.
5. Změňte aktuální přihlašovací údaje.

8.2.2 TC2: Administrace účtu

Daný scénář testuje případy užití, spojené s administračními procesy vytváření firemních účtů, souvisejících projektů a průzkumů:

1. Jste přihlášený(á) a je otevřená domovská stránka.
2. Vytvořte v systému firemní účet [parametry] a přejděte na jeho stránku.
3. Vytvořte v systému projekt [parametry] a přejděte na jeho stránku.
4. Vytvořte v systému průzkum [parametry] a přejděte na jeho stránku.

8.2.3 TC3: Editace prototypu průzkumu

Daný scénář testuje případy užití, spojené s procesy editace prototypů průzkumů a jejich stylizací.

1. Jste přihlášený(á) a je otevřená stránka prototypu průzkumu.
2. Vyberte jedno vzhledové téma průzkumu a uložte změny.
3. Nastavte průhlednost pozadí průzkumu a uložte změny.
4. Přidejte obsah první stránky průzkumu, aby výsledek byl podobný obrázku [obrázek s výsledným vzhledem] a uložte změny.
5. Editujte obsah stránky s dotazem tak, aby výsledek byl podobný obrázku [obrázek s výsledným vzhledem] a uložte změny.
6. Přidejte obsah poslední stránky průzkumu, aby výsledek byl podobný obrázku [obrázek s výsledným vzhledem] a uložte změny.

8.2.4 TC4: Publikace průzkumu

Daný scénář testuje případy užití, spojené s publikačními procesy startu, zastavení, pokračování a ukončení provedení průzkumu.

1. Jste přihlášený(á) a je otevřená stránka prototypu průzkumu.
2. Nastartujte provedení průzkumu. Ověřte, že průzkum je dostupný na vygenerovaném URL a není editovatelný.

3. Zastavte provedení průzkumu. Ověřte, že průzkum není dostupný na vygenerovaném URL a není editovatelný.
4. Pokračujte provedení průzkumu. Ověřte, že průzkum je dostupný na vygenerovaném URL a není editovatelný.
5. Ukončete provedení průzkumu. Ověřte, že průzkum je dostupný na vygenerovaném URL a není editovatelný.

8.2.5 TC5: Účastí v průzkumu

Daný scénář testuje případy užití, spojené s účastí v průzkum

1. Dostal(a) jste odkaz na dotazník [odkaz]. Vyplňte jej. (varianta 1)
2. Dostal(a) jste odkaz na dotazník [odkaz]. Vyplňte jej. (varianta 2)
3. Dostal(a) jste odkaz na dotazník [odkaz]. Vyplňte jej. (varianta 3)

8.3 Shrnutí nejčastějších problémů

Chyby použitelnosti byly rozdělený do tří skupin pro další klasifikaci a upřednostnění chyb.

8.3.1 Kritické chyby použitelnosti

Kritické chyby použitelnosti působí na uživatele nejnegativněji ze všech. Takového problému je nutné se zbavit co nejdříve.

- TC1: Dva ze čtyř účastníků byli zmatení nepřítomností stránky uživatelského profilu, která zatím implementovaná nebyla.
- TC3: Všichni účastníci měli větší komplikace s MarkDown notací. Ukázalo se, že není jednoduchá pro pochopení a extrémně zpomaluje práci nad textovými formami.

8.3.2 Chyby použitelnosti

Chyby použitelnosti nepřekážejí činnosti uživatele, ale přinášejí zbytečnou komplexitu systému a nechávají pocit nedopracovaného produktu.

- TC1: Tři ze čtyř účastníků byli zmateni tím, že pro změnu hesla je nutné se odhlásit. Dialogové okno změny hesla je dostupné pouze při přihlášení a registraci.
- TC5: Aplikace nepodporuje dialogová okna chyby aplikace a chyby síťového přípojení. Implementace takových komponent je závazná.

8.3.3 Kosmetické chyby

Kosmetickýmch chyb se nebudeme zbavovat v etapě vývoje prototypu cíleně. Zaznamenáme je ale pro budoucí kontrolu, kdy se bude stabilizovat celý systém a zahrnuta funkcionalita.

- TC2: Jeden účastník měl připomínku, že není zřejmý přechod na stránku firmy (projektu, průzkumu) pomocí kliknutí na kartičce firmy (projektu, průzkumu).
- TC5: Mobilní aplikace nemá implementovanou domovskou stránku, proto aplikace zatím nepodporuje žádné odkazy, kromě odkazů na průzkumy.

Kapitola 9

Vyhodnocení výsledků práce

Následující sekce se budou věnovat výsledkům provedené práce a možným zlepšením systému. Veškeré snímky uživatelského rozhraní frontendových aplikací jsou k dispozici v příloze [F].

9.1 Vyplněné požadavky

- ✔ **FR_A1:** Systém umožňuje registraci [F.3a], přihlášení [F.3b] a odhlášení uživatele [F.3c].
- ✔ **FR_A2:** Systém umožňuje registraci firemního účtu [F.6] [F.7].
- ✔ **FR_G1:** Systém umožňuje vytváření [F.11], odstraňovaní [F.10] a editaci průzkumů [F.12].
- ✔ **FR_G2:** Systém umožňuje konfiguraci průzkumu ve formě poll [F.15] [F.16].
- ✔ **FR_G5:** Systém umožňuje stylizaci průzkumu alespoň třemi různými způsoby [F.14].
- ✔ **FR_G6:** Systém umožňuje přidání odkazů na externí zdroje do průzkumu [F.13].
- ✔ **FR_G10:** Systém umožňuje publikaci průzkumu do veřejné části, po které již editovatelný není [F.17].
- ✔ **FR_S1:** Systém po publikaci průzkumu zahájí sběr uživatelských odpovědí.
- ✔ **FR_S2:** Systém umožňuje zastavení / pokračování / ukončení sběru uživatelských odpovědí pro určitý dotazník [F.17].
- ✔ **FR_S3:** Systém umožňuje účast anonymního uživatele v průzkumu [F.5a] [F.5b].

- ✔ **FR_S5:** Systém umožňuje prohlížení seznamu odpovědí účastníků [F.17].
- ✔ **NFR_MWA1:** Funkcionalita mobilní webové aplikace má správné chování popsané ve FR pro verze prohlížečů:
  - Google Chrome od verze 50.0 do aktuální verze ke konci vývoje aplikace (pro Android a IOS).
  - Safari od verze 9.1.3 do aktuální verze ke konci vývoje aplikace (pro IOS).
- ✔ **NFR_MWA2:** Aplikace je adaptována pro další rozměry obrazovky ve vertikální orientaci:
  - Od iPhone SE (320 x 568 points, 640 x 1136 pixels).
  - Do iPhone 8+ (414 x 736 points, 1242 x 2208 pixels).
- ✔ **NFR_WA1:** Funkcionalita webové aplikace má správné chování popsané ve FR pro verze prohlížečů:
  - Google Chrome od verze 50.0 do aktuální verze ke konci vývoje aplikace.
  - Safari od verze 9.1.3 do aktuální verze ke konci vývoje aplikace.
- ✔ **NFR_WA2:** Bezpečnostní prvky jsou implementované na dobré úrovni. Pro autentizaci uživatelů je využit jeden ze známých Identity Providerů, Auth0. Komunikace mezi klientskou a serverovou částí probíhá zabezpečeně [5.5].
- ✔ **NFR_BE1:** Výpadek jedné služby nebude způsobovat vadné chování funkcionality, která se týká ostatních služeb [4.2].
- ✔ **NFR_BE2:** Služby vystavují RESTové rozhraní [5.1.3], které je pro každou z nich dobře popsané [5.1.3].
- ✔ **NFR_DEP1:** Výsledkem vyplnění práce je aplikace, skládající se z několika komponent. Byl navržen nástroj z běžné praxe, který umožňuje jednoduchý a pro klienty pochopitelný způsob konfigurace a rozběhnutí jak celého systému, tak i každé jeho části zvlášť. Řešení je zadokumentováno a je podporované v *nix operačních systémech a Windows [7].

9.2 Částečně vyplněné požadavky

- › **FR_G3:** Systém musí umožňovat konfiguraci průzkumu ve formě survey. Pozn.: Datový model a logika provedení všech operací včetně publikace podporují typ průzkumu Survey. Není implementované uživatelské rozhraní pro několikaprvkové a několikastránkové průzkumy, což může být uděláno poměrně jednoduše a vyžaduje „mechanické“ naprogramování.

- **FR_G4:** Systém musí umožňovat konfiguraci průzkumu ve formě quiz. Pozn.: Datový model a logika provedení všech operací včetně publikace podporují typ průzkumu Quiz. Není implementované uživatelské rozhraní pro několikaprvkové a několikastránkové průzkumy, což může být uděláno poměrně jednoduše a vyžaduje „mechanické“ naprogramování.
- **FR_S6:** Systém musí umožňovat účast v průzkumu přihlášeného uživatele. Pozn.: mobilní aplikace nemá integrací s Auth0, protože zatím nenabízí funkcionalitu, která musí mít autorizační logiku. To jinak může být vyřešeno přidáním 30-40 řádků kódu.

9.3 Nevyplněné požadavky

- **FR_A3:** Systém musí umožňovat udělení a odebírání přístupu uživatele k firemnímu účtu.
- **FR_A4:** Systém musí rozlišovat práva uživatelů pro operace v rámci jednoho účtu.
- **FR_G7:** Systém musí umožňovat přidání obrázků do průzkumu.
- **FR_G8:** Systém musí umožňovat přidání základních druhů validace průzkumu.
- **FR_G9:** Systém musí umožňovat definici logických pravidel, podle kterých se dotazník bude chovat dynamicky (větvení).
- **FR_S7:** Systém musí umožňovat generaci reportu výsledků ukončeného dotazníku ve formě tabulí a grafů.
- **FR_S8:** Systém musí umožňovat export reportu ve formátech CSV a PDF.
- **FR_S9:** Systém musí umožňovat přístup k reportu pomocí vygenerovaného odkazu.

Kapitola 10

Závěr

Cílem práce byl definován návrh a implementace funkčního prototypu aplikace, která by umožňovala generaci forem sběru informací, provádění veřejných a privátních průzkumů a sběr výsledků vyhodnocení. Inspirací tohoto tématu byla představa, že žádná známá řešení v dané sféře neumožňují autonomní sběr odpovědí na mobilních zařízeních. Proto bylo rozhodnuto v první fázi daného projektu implementovat informační systém, který by posloužil dobře navrženým fundamentem dalšího vývoje, rozšiřujícího standardní funkcionalitu.

Nejdříve jsme prozkoumali řešení [3.2] existující na trhu a definovali základní požadavky cílové skupiny uživatelů [3.3]. Výsledky průzkumu jsme podrobili detailní analýze [3], která velice napomohla ke korektnímu návrhu flexibilní, lehce rozšířitelné a moderní architektury aplikace [4]. Před začátkem implementace [5] systému jsme definovali používané principy organizace práce a naplánovali její přibližný průběh [6]. Tři z mnoha případů užití aplikace jsme otestovali na cílové skupině uživatelů [8]. Během práce jsme taky použili nejvíce vhodný způsob jednoduchého rozběhnutí celého systému a popsali jej v návodu pro rozběhnutí aplikace [7].

Výsledný produkt je souhrnem několika propojených a snadně spustitelných komponent. Tvořící jej webové aplikace jsou kompatibilní s moderními prohlížeči jak stacionárních počítačů, tak i mobilních zařízení. Podařilo se nám vyplnit [9] nejenom všechny body zadání, ale i dvacet jeden funkční a nefunkční požadavek.

Měl jsem radost danou práci vyplnit. Umožnila mi vyzkoušet nashromážděné během studia znalosti a aplikovat je na téma, které mě v tuto dobu zajímalo - návrh architektury distribuovaných systémů. Doufám, že také poslouží dobrou bází pro další rozpracování, vždyť zaměření rozvoje je dost: nativní klientské aplikace. gatewaje, doplnění funkcionality

hlavní webové aplikace a další.

Příloha A

Seznam použitých zkratek

JSON	JavaScript Object Notation
JWT	JSON Web Token
QR code	Quick Response Code
FR	Functional Requirement
FR_A	FR Autentizace a autorizace
FR_G	FR Generování prototypů a jejich publikace
FR_S	FR Sběr odpovědí a jejich analýza
CSV	Comma-Separated Values
PDF	Portable Document Format
NFR	Non-functional Requirement
NFR_MWA	NFR Webová aplikace pro mobilní zařízení
NFR_WA	NFR Webová aplikace
NFR_BE	NFR Backendové služby
NFR_DEP	NFR Nasazení
OS	Operating System
IOS	iPhone OS
SE	Special Edition
WSO2	Web Service Oxygen
WSO2IS	WSO2 Identity Server
REST	Representational State Transfer
HTTP	HyperText Transfer Protocol

MVC	Model-View-Controller
API	Application Programming Interface
XML	eXtensible Markup Language
AMQP	Advanced Message Queuing Protocol
BFE	Backend for Frotend
GPS	Global Positioning System
URL	Uniform Resource Locator
Java EE	Java Enterprise Edition
SpEL	Spring Expression Language
CRUD	Create, Read, Update, Delete
DDD	Domain Driven Design
CQS	Command-query separation
CQRS	Command-query responsibility segregation
ActiveMQ	Active Message Queue
DBMS	Database Management System
ORM	Object-Relational Mapping
JPQL	Java Persistence Query Language
POJO	Plain Old Java Object
Slf4j	Simple Logging Facade for Java
SPA	Single Page Application
ReactiveX	Reactive Extensions
RxJS	ReactiveX library for JavaScript
HTML	HyperText Markup Language
SaaS	Software as a Service
ID	IDentification
TC	Test Case
CSS	Cascading Style Sheets
XLS	Microsoft Excel spreadsheet
DOCX	Microsoft Word 2007/2010/2013 document
UUID	Universally Unique Identifier

Příloha B

Literatura

[1] Srivatsan Sundararajan. Spring framework: @restcontroller vs @controller. [online], Září 2015. [cit. 07-04-2018]. Dostupné z: https://www.genuitec.com/spring-frameworkrestcontroller-vs-controller/.

[2] Markus Eisele. Modern java ee design patterns. [online], Leden 2016. [cit. 07-04-2018]. Dostupné z: https://www.oreilly.com/ideas/modern-java-ee-design-patterns.

[3] Stan Schneider. What's the difference between dds and amqp? [online], Duben 2013. [cit. 07-04-2018]. Dostupné z: http://www.electronicdesign.com/embedded/what-s-difference-between-dds-and-amqp.

[4] Phil Calçado. The back-end for front-end pattern (bff). [online], Září 2015. [cit. 07-04-2018]. Dostupné z: http://philcalcado.com/2015/09/18/the_back_end_for_front_end_pattern_bff.html.

[5] Sitefinity. Backward compatibility. [online]. [cit. 07-04-2018]. Dostupné z: https://docs.sitefinity.com/overview-authentication-models.

[6] Auth0. Authentication for client-side web apps. [online]. [cit. 07-04-2018]. Dostupné z: https://auth0.com/docs/application-auth/current/client-side-web.

[7] Tirena Dingeldein. The 12 best free and open source survey tools to power your research. [online], Duben 2018. [cit. 07-04-2018]. Dostupné z: https://blog.capterra.com/best-free-survey-tools-power-your-research/.

[8] Scott Nesbitt. 4 open source alternatives to surveymonkey. [online], Únor 2017. [cit. 07-04-2018]. Dostupné z: https://opensource.com/article/17/2/tools-online-surveys-polls.

[9] Megan Marrs. 7 best survey tools: Create awesome surveys for free! [online], Prosinec 2017. [cit. 07-04-2018]. Dostupné z: https://www.wordstream.com/blog/ws/2014/11/10/best-online-survey-tools.

[10] Technopedia. Conceptual data model. [online]. [cit. 07-04-2018]. Dostupné z: https://www.techopedia.com/definition/28026/conceptual-data-model.

[11] Petra Rejnková. Stavový diagram. [online], 2009. [cit. 07-04-2018]. Dostupné z: http://uml.czweb.org/stavovy_diagram.htm.

[12] Robert Annett. What is a monolith? [online], Listopad 2014. [cit. 07-04-2018]. Dostupné z: http://www.codingthearchitecture.com/2014/11/19/what_is_a_monolith.html.

[13] Martin Fowler. Microservices. [online], Březen 2014. [cit. 07-04-2018]. Dostupné z: https://martinfowler.com/articles/microservices.html.

[14] Chris Richardson. *Microservice Patterns*. 2017. [cit. 07-04-2018]. Dostupné z: MEAP Edition Manning Early Access Program Microservice Patterns Version 6.

[15] Chris Richardson. Pattern: Monolithic architecture. [online]. [cit. 07-04-2018]. Dostupné z: http://microservices.io/patterns/monolithic.html.

[16] Spring framework. [online]. [cit. 07-04-2018]. Dostupné z: https://projects.spring.io/spring-framework/.

[17] Rafael Salerno. How netflix's circuit breaker works. [online], Září 2016. [cit. 07-04-2018]. Dostupné z: https://dzone.com/articles/spring-cloud-netflix-how-works-circuit-breaker-eur.

[18] Martin Fowler. Circuitbreaker. [online], Březen 2014. [cit. 07-04-2018]. Dostupné z: https://martinfowler.com/bliki/CircuitBreaker.html.

[19] Agile glossary. [online]. [cit. 07-04-2018]. Dostupné z: https://www.agilealliance.org/agile101/agile-glossary/.

[20] Sprint planning. [online]. [cit. 07-04-2018]. Dostupné z: http://www.scrumguides.org/scrum-guide.htmlevents-planning.

[21] Docker. [online]. [cit. 07-04-2018]. Dostupné z: https://www.docker.com/what-docker.

Příloha C

Seznam statistických tabulek

		Free Online Surveys	eSurvey Creator	Survey Monkey	Question Pro	Survio
Druh průzkumu	Poll	✔	✔	✔	✔	✔
	Quiz	✔		✔		
	Survey	✔	✔	✔	✔	✔
Autorizace účastníků	Podepisovatelný dotazník	✔				
	Anonymní dotazník	✔	✔	✔	✔	✔
Stylizace	Pomocí CSS		✔			
	Výběr z několika předurčených témat	✔	✔	✔	✔	
Možnost přidání media	Foto	✔	✔	✔	✔	✔
	Video	✔	✔	✔	✔	✔
	Soubor		✔			
	Firemní logo	✔	✔			
Aktivita při ukončení dotazníku	Redirect na externí stránku	✔		✔		
	Statická poděkovací stránka	✔	✔	✔	✔	✔
Randomizace pořadí otázek				✔		✔
Dynamické dotazníky chovající se podle logických pravidel		✔	✔	✔	✔	✔
Automatická internacionalizace celého dotazníku			✔	✔	✔	
Možnost přidání linků na externí zdroje		✔	✔			
Validace uživatelských odpovědí				✔		
Možnost editovat existující dotazník						✔
Omezený čas pro vyplnění dotazníku			✔			

Tabulka C.1: Porovnání existujících řešení. Tvorba, návrh a vzhled.

	Free Online Surveys	eSurvey Creator	Survey Monkey	Question Pro	Survio
Podpora UI pro mobilní nástroje		✔		✔	✔
Protekce přístupu k průzkumu heslem		✔		✔	✔
Prevence opakované účasti		✔			
Možnost vlastnit svoji (pod)doménu	✔	✔		✔	
Posílání emailu s notifikací o vyplnění	✔	✔	✔	✔	✔
Integrace do sociálních sítí	✔		✔	✔	✔
Možnost pozvání přes email	✔		✔	✔	✔
Možnost použiti dotazníku na svých webových stránkách (iframe)	✔		✔	✔	✔
Možnost nastavit datum vypršení dotazníku			✔		
Nastavení maximálního počtu odpovědí			✔		
Možnost kontroly ip-adresy klientu a její blokací			✔		✔

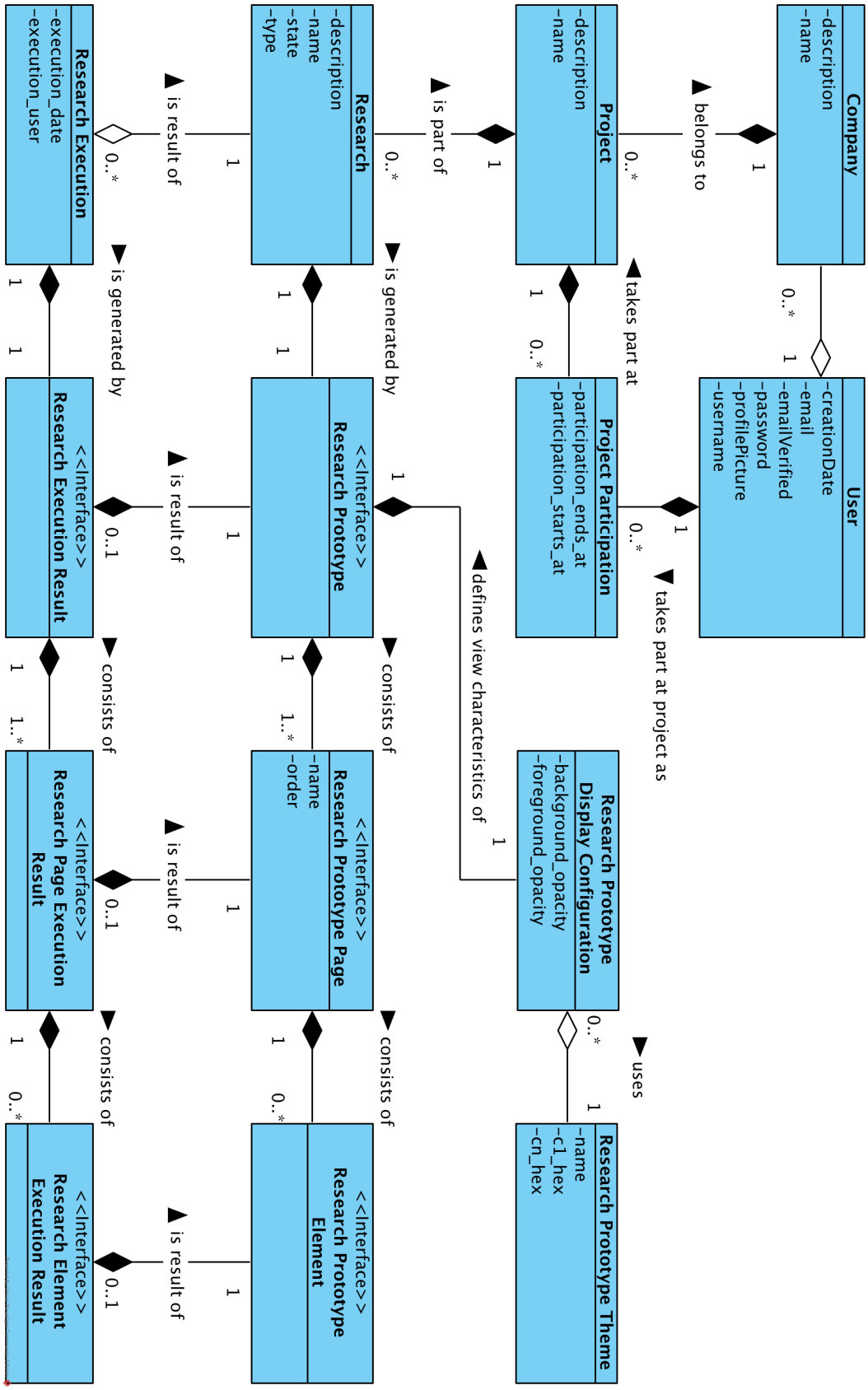
Tabulka C.2: Porovnání existujících řešení. Publikace průzkumů a sběr odpovědí.

		Free Online Surveys	eSurvey Creator	Survey Monkey	Question Pro	Survio
Kreslení grafů		✔	✔	✔	✔	✔
Generace tabulek se statistikou		✔	✔	✔	✔	✔
Prohlížení individuálních odpovědí		✔		✔	✔	
Možnost porovnání několika datových setů		✔		✔		
Generace reportů	PDF	✔	✔	✔	✔	✔
	XLS	✔	✔	✔	✔	✔
	CSV	✔	✔	✔	✔	✔
	DOCX				✔	✔
	HTML				✔	✔
Zveřejnění výsledků	Odkaz		✔	✔	✔	✔
	Google Docs				✔	
	Dropbox				✔	

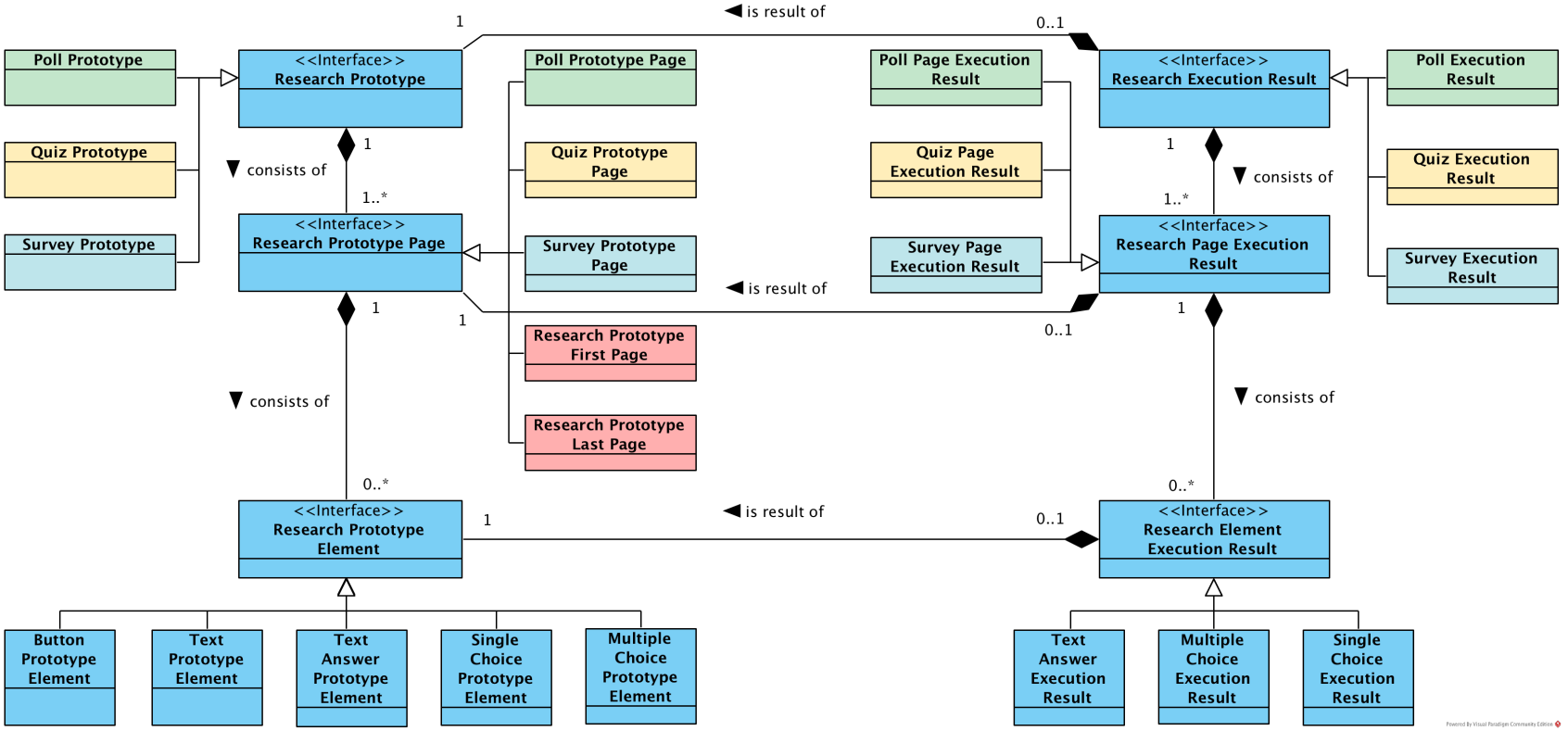
Tabulka C.3: Porovnání existujících řešení. Analýza výsledků a reporty.

Příloha D

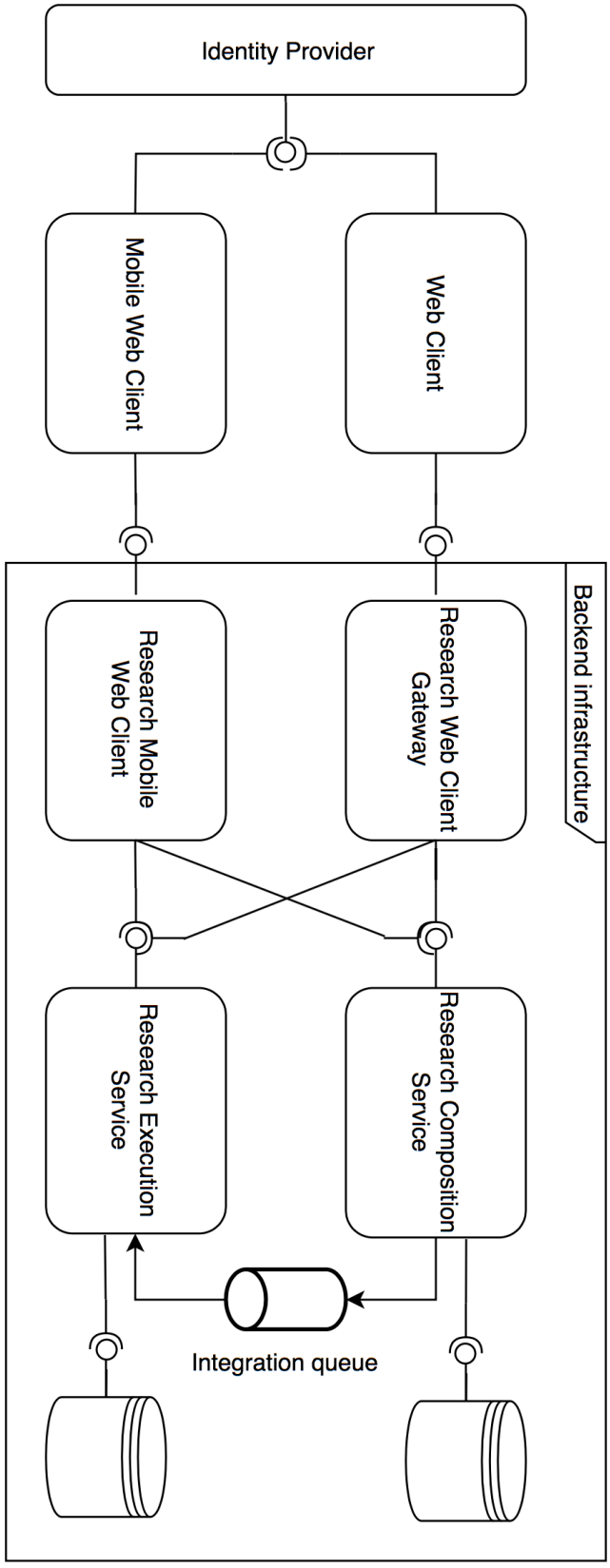
Seznam diagramů



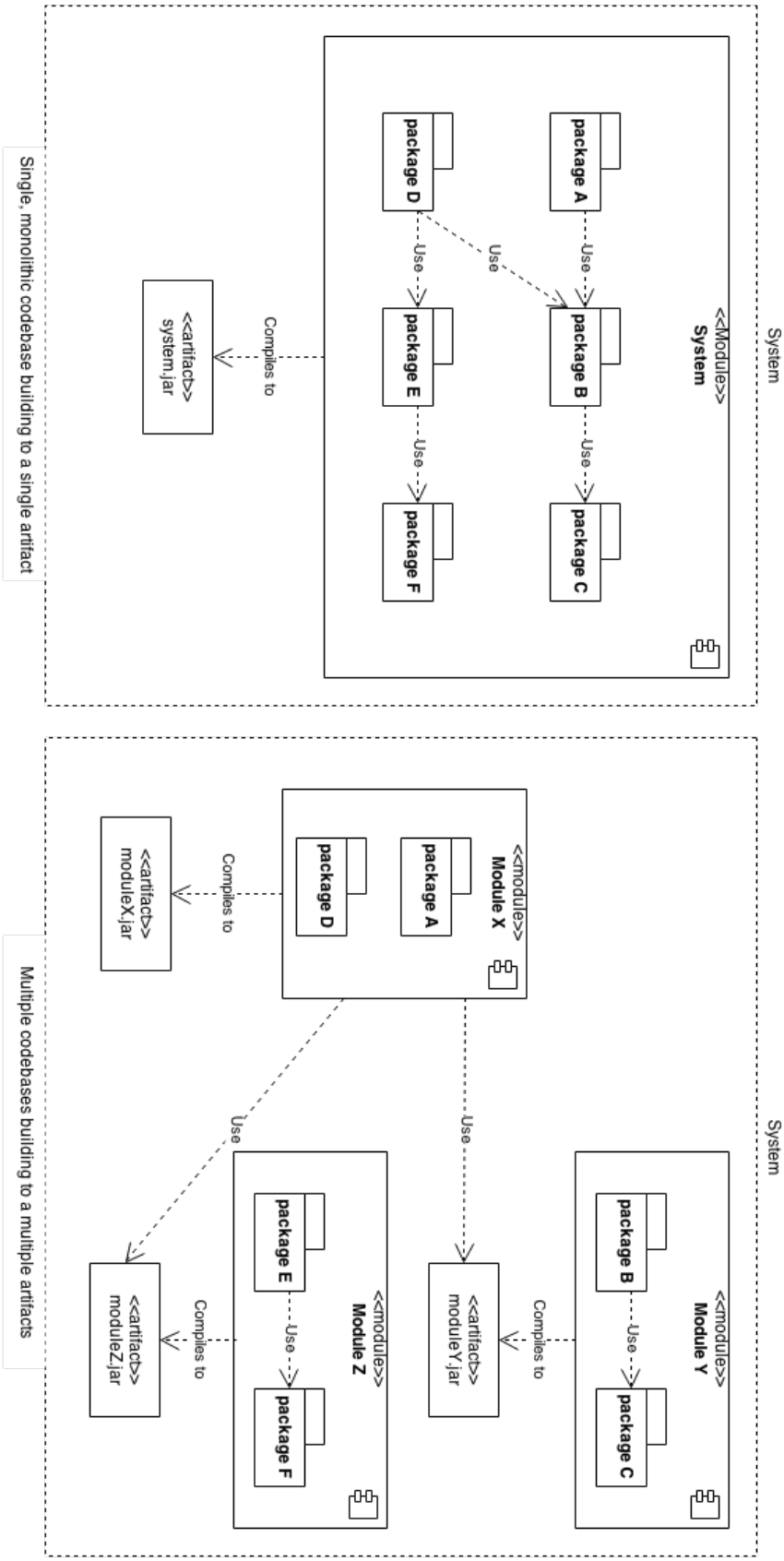
Obrázek D.1: Logický datový model. Hlavní schéma.



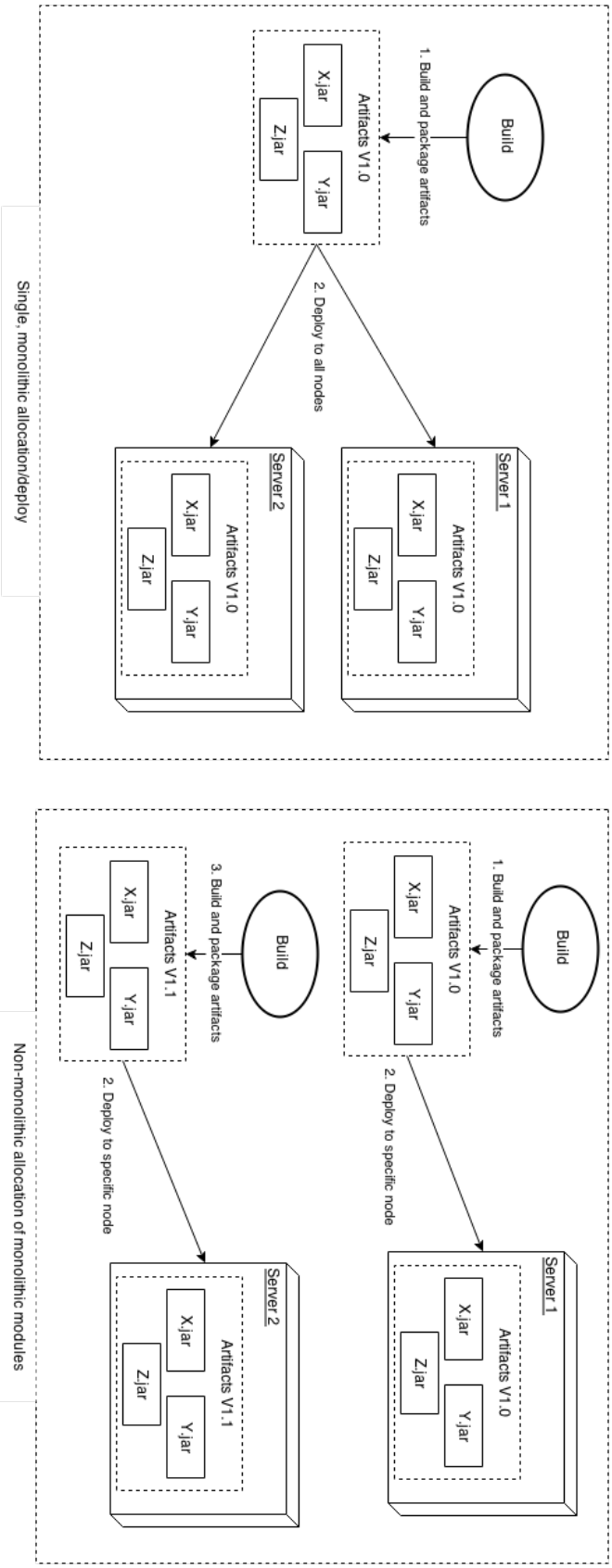
Obrázek D.2: Logický datový model. Generalizace prototypů a jejich potomků.



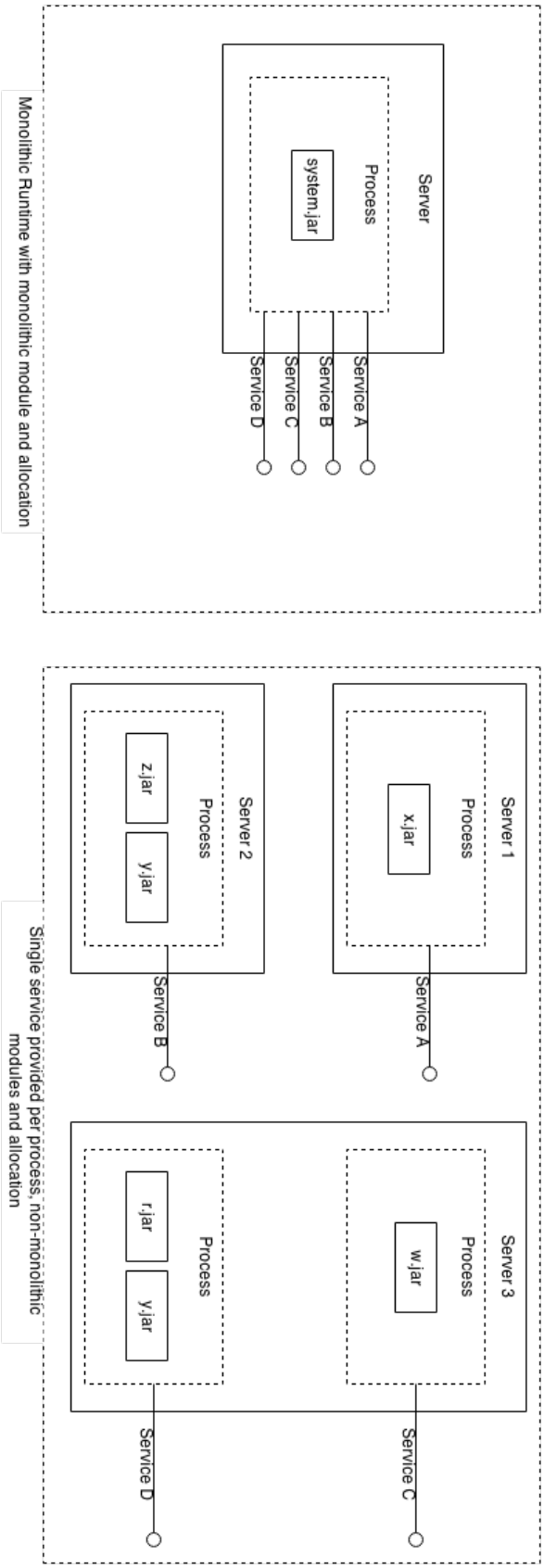
Obrázek D.3: Výsledný architektonický návrh.



Obrázek D.4: Schéma modulárního monolitu podle [1].



Obrázek D.5: Schéma alokačního monolitu podle [1].



Obrázek D.6: Schéma environmentálního monolitu podle [1].

Příloha E

Demonstrace kódu

```java
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    @Value(value = "${auth0.apiAudience}")
    private String apiAudience;
    @Value(value = "${auth0.issuer}")
    private String issuer;

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        JwtWebSecurityConfigurer
                .forRS256(apiAudience, issuer)
                .configure(http)
                .cors().and()
                .authorizeRequests()
                .antMatchers("/api/**").authenticated()
                .anyRequest().permitAll()
                .and()
                .csrf().disable();
    }

    @Bean
    public CorsConfigurationSource corsConfigurationSource() {
        CorsConfiguration configuration = new CorsConfiguration();
        configuration.setAllowedOrigins(ImmutableList.of("*"));
          configuration.setAllowedMethods(ImmutableList.of("HEAD", "GET", "POST", "PUT",
→                                   "DELETE", "PATCH"));
        configuration.setAllowCredentials(true);
→               configuration.setAllowedHeaders(ImmutableList.of("Authorization",
→                           "Cache-Control", "Content-Type"));
        UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration("/**", configuration);
        return source;
    }
}
```

Listing 1: Backend. Konfigurace Spring Security jedné z mikroslužeb.

```java
// cvut.fel.bachelor.research.composition.api.rest.ResearchController:
@PatchMapping("/{researchId}")
@ApiOperation("Update research authenticated user participates or owns")
public ResearchResource update(@PathVariable("companyId") UUID companyId,
                               @PathVariable("projectId") UUID projectId,
                               @PathVariable("researchId") UUID researchId,
→                                  @Valid @NotNull @RequestBody ResearchUpdateRequest
→                                       updateRequest
) {
    // return a service execution result
    return service.update(companyId, projectId, researchId, updateRequest, assembler);
}
```

Listing 2: Backend. Příklad mapování HTTP operace. Zaktualizování informaci průzkumu.

```
// cvut.fel.bachelor.research.composition.service.ResearchService:
→         public <R> R update(UUID companyId, UUID projectId, UUID researchId,
→     ResearchUpdateRequest updateRequest, ResourceAssembler<Research, R> assembler) {
    // check if research exists
    requireResearch(companyId, projectId, researchId);
    // retrieve research
    Research research = researchRepository.get(researchId);
    // check user permissions
    research.getProject().requireToBeOwnerOrParticipant(getAuthenticatedUserName());
    // update entity
    research.update(updateRequest);
    // persist updated data
    research = researchRepository.save(research);
    // assemble resource and return
    return assembler.assemble(research);
}
```

Listing 3: Backend. Příklad servisní logiky. Zaktualizování informaci průzkumu.

```
// cvut.fel.bachelor.research.composition.domain.Research:
public void update(ResearchUpdateRequest updateRequest) {
    // can not be updated if has been already started
    if (state != ResearchState.DRAFT) {
        throw new ResearchIllegalOperationException(id, state, "update");
    }

    // check if name was updated
    if (!Strings.isNullOrEmpty(updateRequest.getName())) {
        name = updateRequest.getName();
    }
    // check if description was updated
    if (!Strings.isNullOrEmpty(updateRequest.getDescription())) {
        description = updateRequest.getDescription();
    }
    // check if prototype was updated
    if (!Strings.isNullOrEmpty(updateRequest.getSerializedPrototype())) {
        serializedPrototype = updateRequest.getSerializedPrototype();
    }
}
```

Listing 4: Backend. Příklad doménové logiky. Zaktualizování informaci průzkumu.

```
// Odesílací strana
// cvut.fel.bachelor.research.composition.service.ResearchDomainEventListener
@EventListener
→       public void handleResearchStarted(ResearchStartedDomainEvent event,
→       @Value("${research.publishing.queue.name}") String publishingQueueName) {
    // retrieve research entity
    Research research = researchRepository.get(event.getResearchId());
    // build JMS message body
    ResearchStartMessage researchStartMessage = ResearchStartMessage.builder()
            .id(research.getId())
            .companyId(research.getProject().getCompany().getId())
            .projectId(research.getProject().getId())
            .state(research.getState())
            .type(research.getType())
            .serializedPrototype(research.getSerializedPrototype())
            .build();
    // send message
→           jmsTemplate.convertAndSend(publishingQueueName, researchStartMessage,
→                                      auditMessagePostProcessor);
}

// Přijímací strana
// cvut.fel.bachelor.research.execution.service.MQListener
@JmsListener(destination = "${research.publishing.queue.name}")
private void receiveStartMessage(@Payload ResearchStartMessage researchStartMessage) {
    Objects.requireNonNull(researchStartMessage);
    // create domain object instance
    Research research = Research.builder()
            .executionApiAccessId(UUID.randomUUID())
            .id(researchStartMessage.getId())
            .companyId(researchStartMessage.getCompanyId())
            .projectId(researchStartMessage.getProjectId())
            .type(researchStartMessage.getType())
            .state(researchStartMessage.getState())
            .serializedPrototype(researchStartMessage.getSerializedPrototype())
            .build();
    // persist object
    researchRepository.save(research);
}
```

Listing 5: Backend. Příklad posílaní zprav mezi služby. Publikace průzkumu.

```java
@Getter
@Setter
@NoArgsConstructor
@EqualsAndHashCode(of = "id", callSuper = true)
@Entity
public class Research extends BaseEntity {

    @Id
    @GeneratedValue
    private UUID id;

    @ManyToOne
    →           @JoinColumn(name = "project_id", foreignKey = @ForeignKey(name =
    →                                "fk_research_project_id"))
    private Project project;

    @Column(name = "name")
    private String name;

    @Column(name = "description")
    private String description;

    @Builder.Default
    @Enumerated(EnumType.STRING)
    private ResearchType type = ResearchType.POLL;

    @Builder.Default
    @Enumerated(EnumType.STRING)
    private ResearchState state = ResearchState.DRAFT;

    @Column(name = "serialized_prototype", columnDefinition = "TEXT")
    private String serializedPrototype;

    ......
}

@Repository
public interface ResearchRepository extends CrudRepository<Research, UUID> {

    default Research get(UUID id) {
        Optional<Research> entity = findById(id);
        if (!entity.isPresent()) {
            throw new ResearchNotFoundException(id);
        }
        return entity.get();
    }

}
```

Listing 6: Backend. Příklad mapování doménové entity průzkumu a jejího repozitáře.

```
@Injectable()
export class TokenInterceptor implements HttpInterceptor {

  constructor(public auth: AuthService) {
  }

  intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    let url: string = request.url;
    let headers: HttpHeaders = request.headers;

    if (this.auth.isAuthenticated()) {
      // add an authorization header if user is authenticated
      headers = headers.set('Authorization', 'Bearer ' + this.auth.accessToken);
    }
    // add service baseUrl
    url = url.replace('composition-be/', environment.http.service.compositionBaseUrl);
    url = url.replace('execution-be/', environment.http.service.executionBaseUrl);
    // return updated request
    return next.handle(request.clone({url: url, headers: headers}));
  }
}
```

Listing 7: Frontend. HTTP intercepor pro přidání autorizační informaci.

```
// definice routingů aplikace
export const ROUTES: Routes = [
  {path: '', component: WelcomeComponent},
  {path: 'companies', component: CompaniesComponent, canActivate: [AuthGuard]},
  {path: 'companies/:id', component: CompanyComponent, canActivate: [AuthGuard]},
  {path: 'companies/:companyId/projects/:projectId', component: ProjectComponent,
    canActivate: [AuthGuard]},
  {path: 'companies/:companyId/projects/:projectId/researches/:researchId', component:
    ResearchComponent, canActivate: [AuthGuard]},
  {path: 'callback', component: CallbackComponent},
  {path: '**', redirectTo: ''}
];

// Implementace služby pro kontrolu přístupu
@Injectable()
export class AuthGuard implements CanActivate {

  constructor(private authService: AuthService, private router: Router) {
  }

  canActivate(): boolean {
    // compute decision
    const canActivate = !!this.authService && this.authService.isAuthenticated();
    // redirect to the start page if not authorized
    if (!canActivate) {
      this.router.navigate(['/']);
    }
    // return value
    return canActivate;
  }
}
```

Listing 8: Frontend. Příklad zabezpečení přístupu k stránkám aplikace.

```
version: '2'

services:
  mq:
    image: webcenter/activemq:latest
    container_name: thesis_mq
    ports:
      - 8161:8161
      - 61616:61616
      - 61613:61613
    environment:
      ACTIVEMQ_NAME: amq
      ACTIVEMQ_REMOVE_DEFAULT_ACCOUNT: 'True'
      ACTIVEMQ_ADMIN_LOGIN: admin
      ACTIVEMQ_ADMIN_PASSWORD: admin
      ACTIVEMQ_CONFIG_QUEUES_ResearchPublishingQueue: 'research-publishing-queue'

  desktop_frontend:
    build: ./ResearchCreatorDesktopFrontend
    image: desktop_frontend
    container_name: thesis_desktop_fe
    ports:
      - 4200:80

  mobile_frontend:
      build: ./ResearchCreatorMobileFrontend
      image: mobile_frontend
      container_name: thesis_mobile_fe
      ports:
        - 8100:8100

  composition_service:
    build: ./ResearchCompositionService
    image: composition_service
    container_name: thesis_composition_be
    ports:
      - 8080:8080
    environment:
          - spring.datasource.url=jdbc:postgresql://composition_db:5432/composition
          - spring.datasource.username=composition
          - spring.datasource.password=composition
          - research.publishing.queue.name=research-publishing-queue
          - spring.activemq.broker-url=tcp://mq:61616
    links:
      - composition_db
      - mq
    depends_on:
      - composition_db

# viz. pokračování listingu dále
```

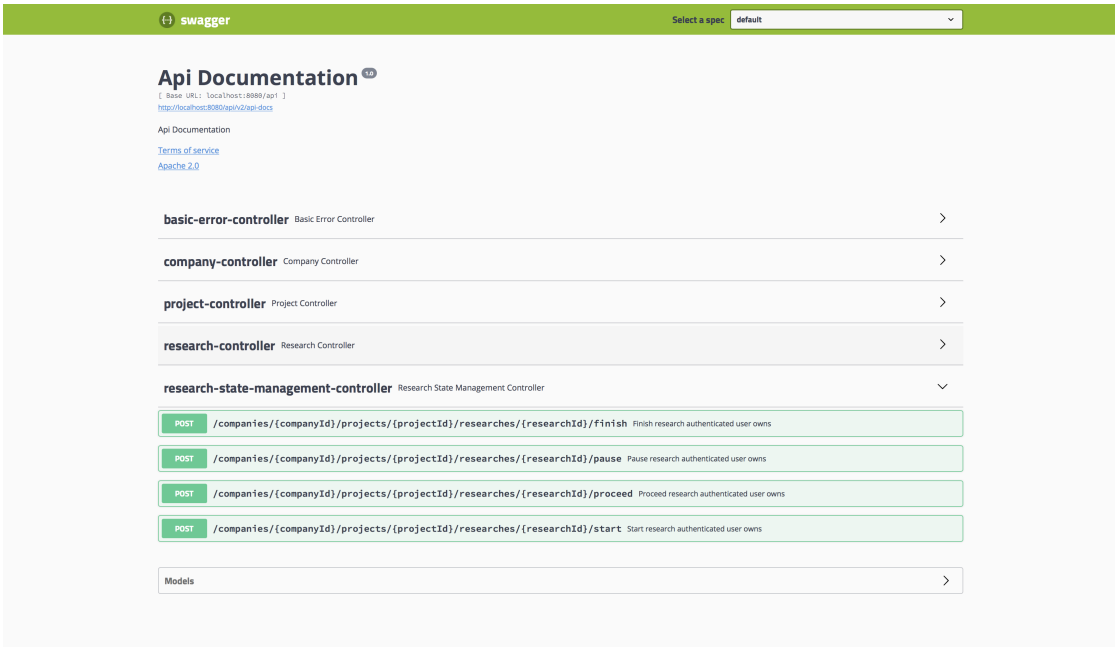
Listing 9: Docker. Ukázka konfiguračního souboru pro rozběhnutí celého systému. 1

```
# pokračování předchozího listingu

  execution_service:
    build: ./ResearchExecutionService
    image: execution_service
    container_name: thesis_execution_be
    ports:
      - 8090:8080
    environment:
          - spring.datasource.url=jdbc:postgresql://execution_db:5432/execution
          - spring.datasource.username=execution
          - spring.datasource.password=execution
          - spring.activemq.broker-url=tcp://mq:61616
    links:
      - execution_db
      - mq
    depends_on:
      - execution_db

  # FIXME: should be implemented as two databases under the same DBMS instance. Was
                            designed as simple as possible :)
  composition_db:
    image: postgres
    container_name: composition_db
    ports:
    - 5434:5432
    environment:
      POSTGRES_USER: "composition"
      POSTGRES_PASSWORD: "composition"
      POSTGRES_DB: "composition"

  execution_db:
    image: postgres
    container_name: execution_db
    ports:
    - 5433:5432
    environment:
      POSTGRES_USER: "execution"
      POSTGRES_PASSWORD: "execution"
      POSTGRES_DB: "execution"
```

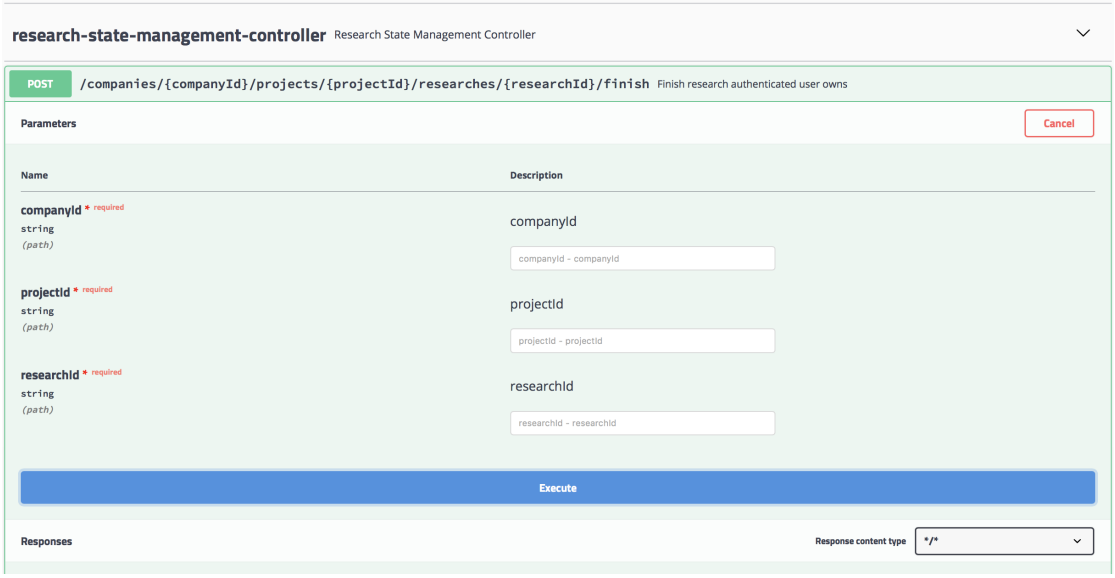
Listing 10: Docker. Ukázka konfiguračního souboru pro rozběhnutí celého systému. 2

Příloha F

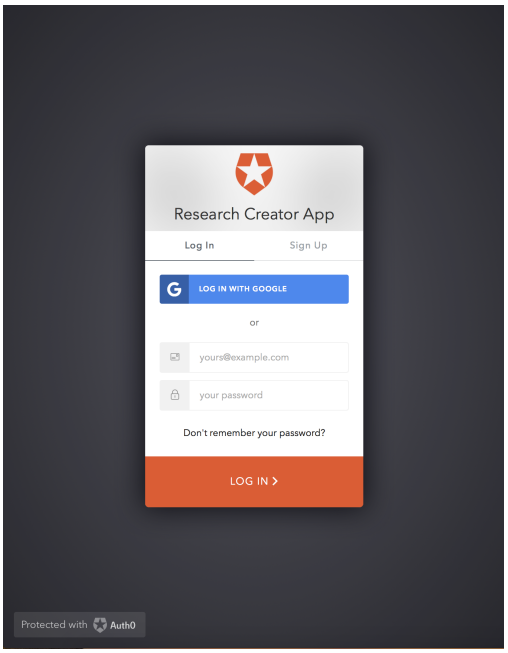
Seznam snímků uživatelského rozhraní



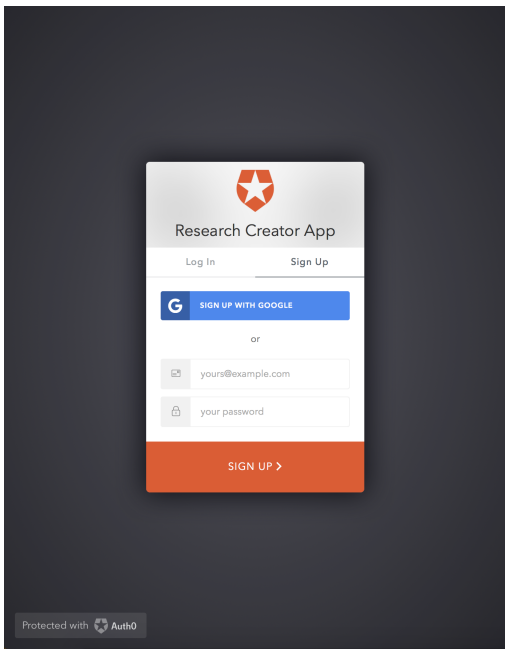
Obrázek F.1: Rozhraní Swagger dokumentace pro službu sestavování průzkumů.



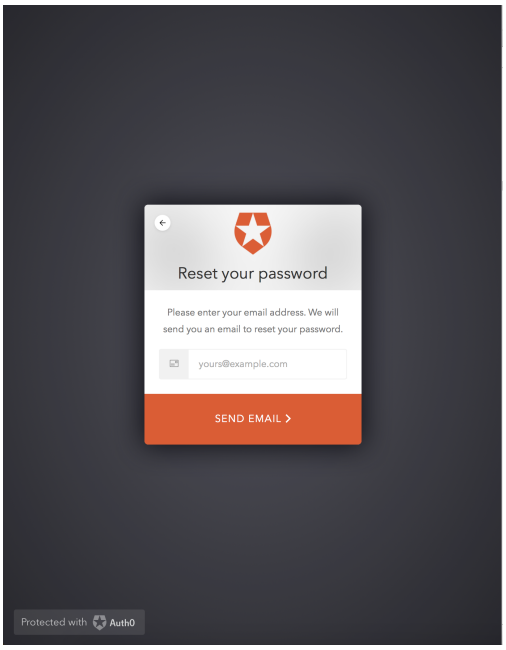
Obrázek F.2: Interaktivní rozhraní Swagger pro ukončení průzkumu.



(a) : Dialog přihlášení

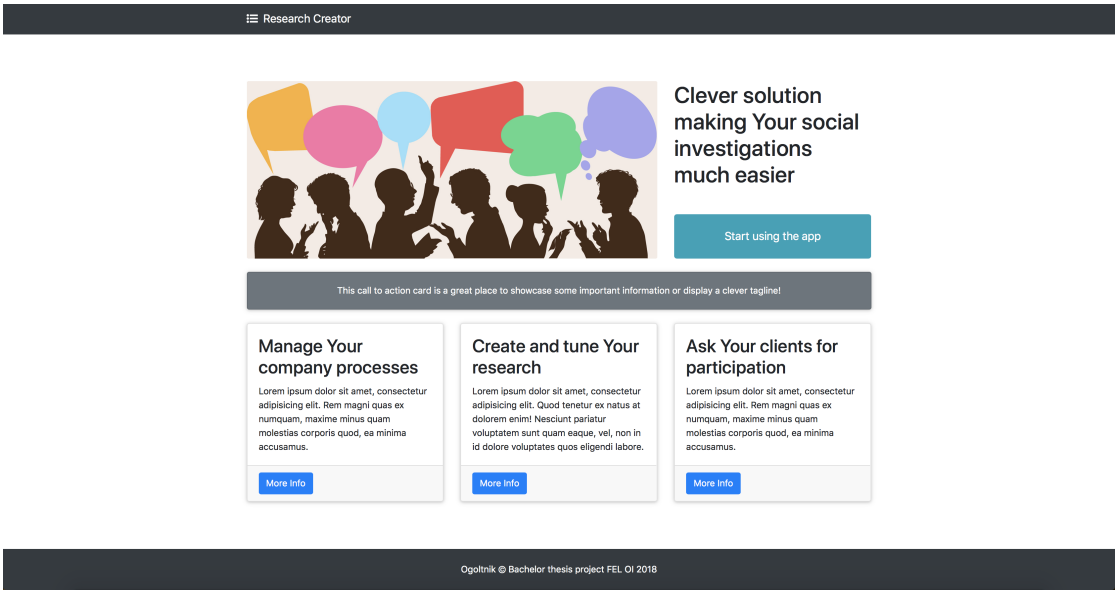


(b) : Dialog registrace

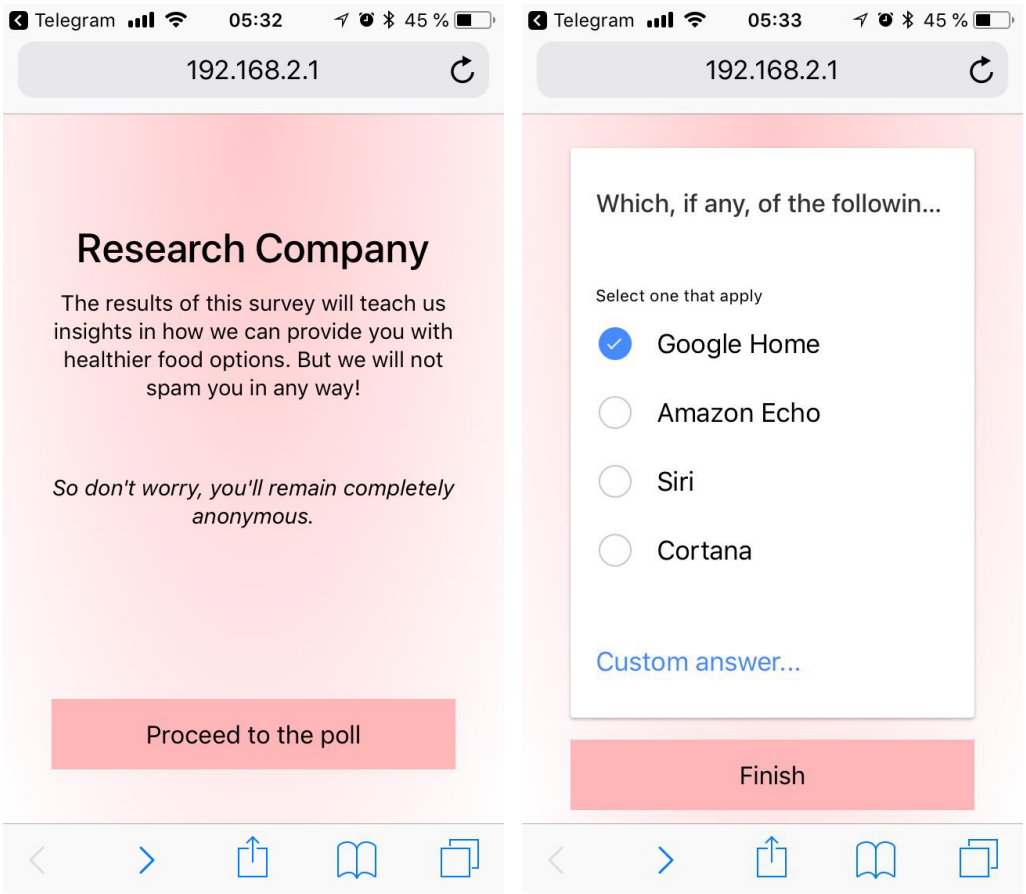


(c) : Dialog změny hesla

Obrázek F.3: Webová aplikace. Autentizace

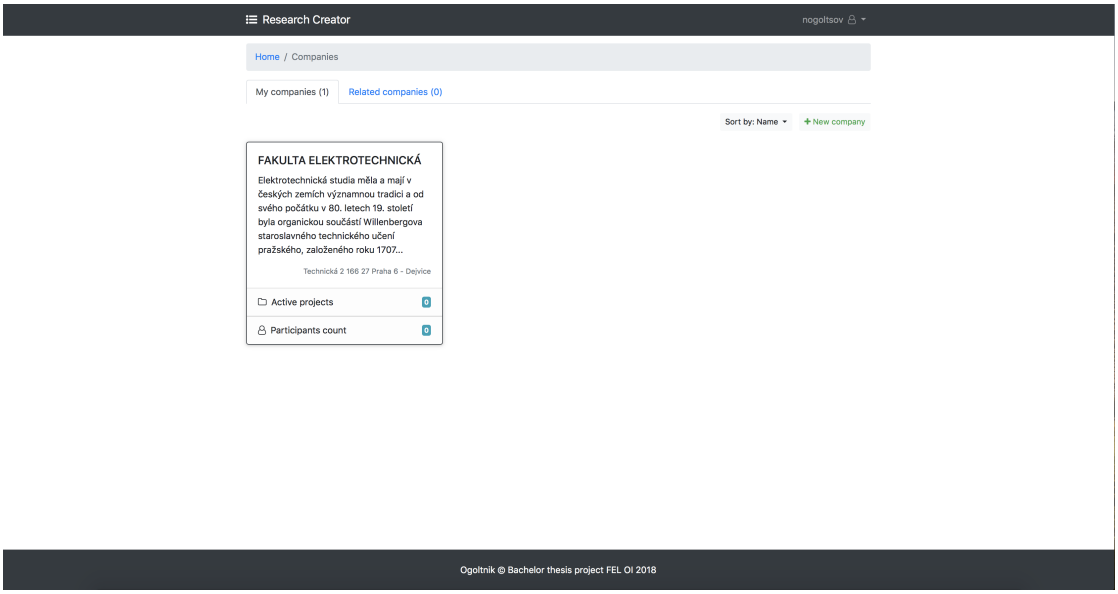


Obrázek F.4: Webová aplikace. Domovská stránka

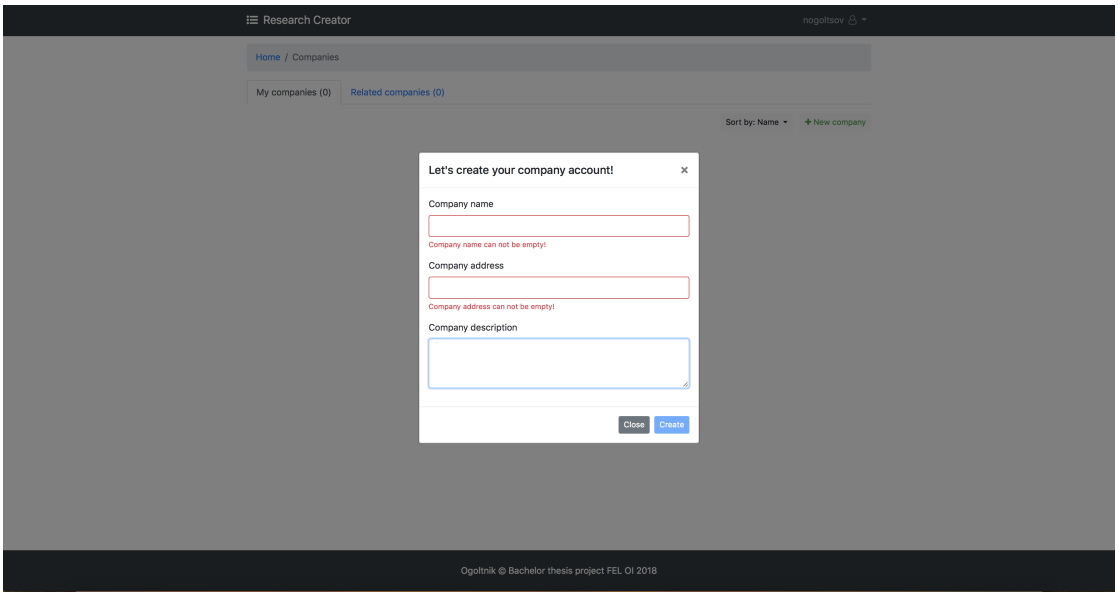


(a) : Úvodní stránka **(b) :** Single-choice otázka

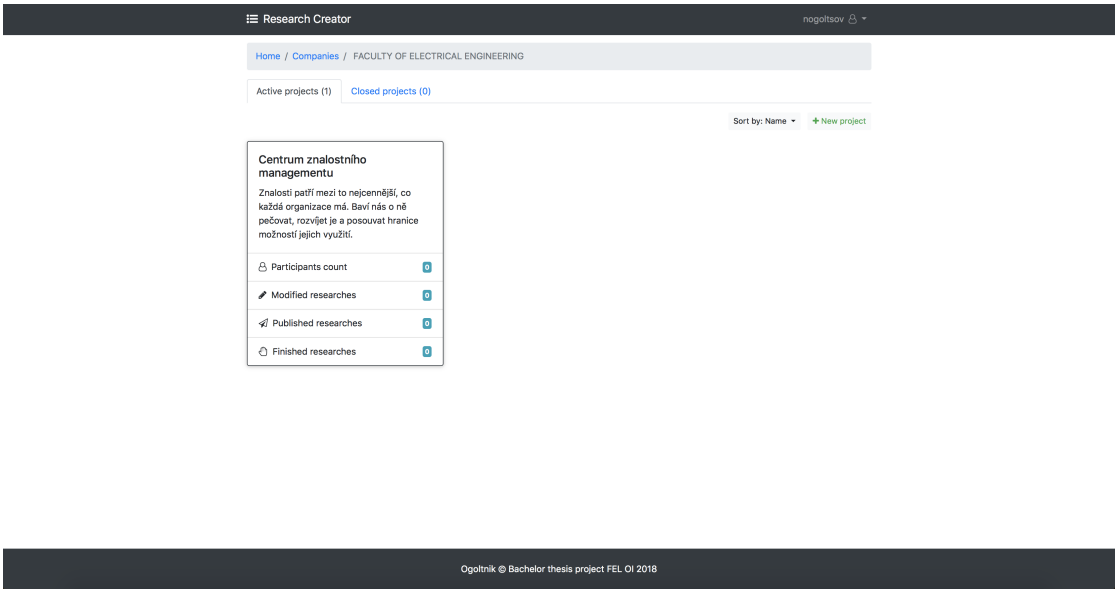
Obrázek F.5: Mobilní webová aplikace. Jednoduchý dotazník



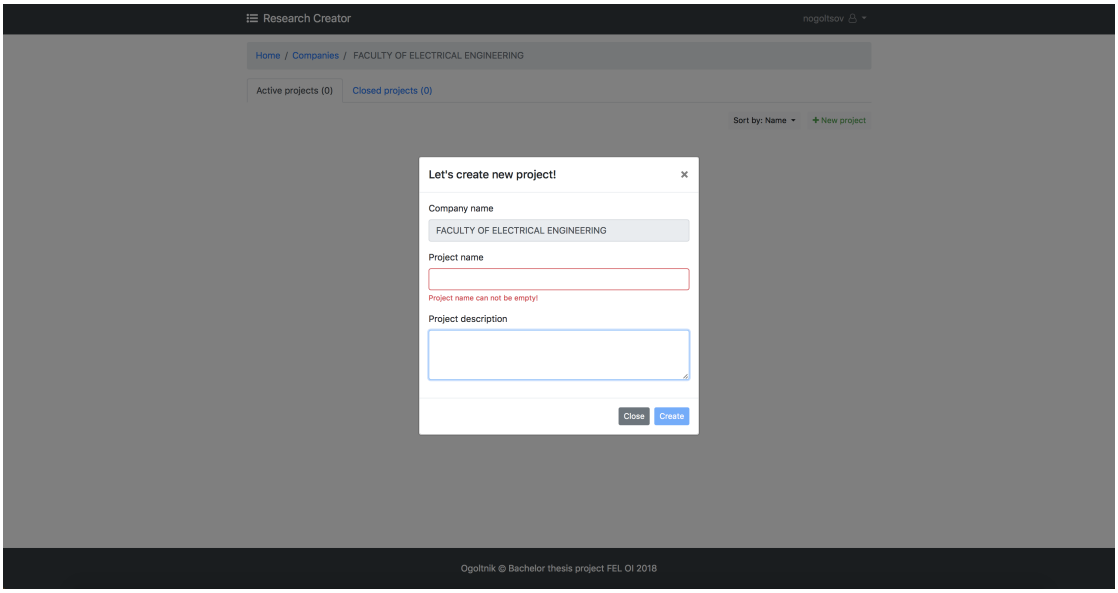
Obrázek F.6: Webová aplikace. Seznam firemních účtů



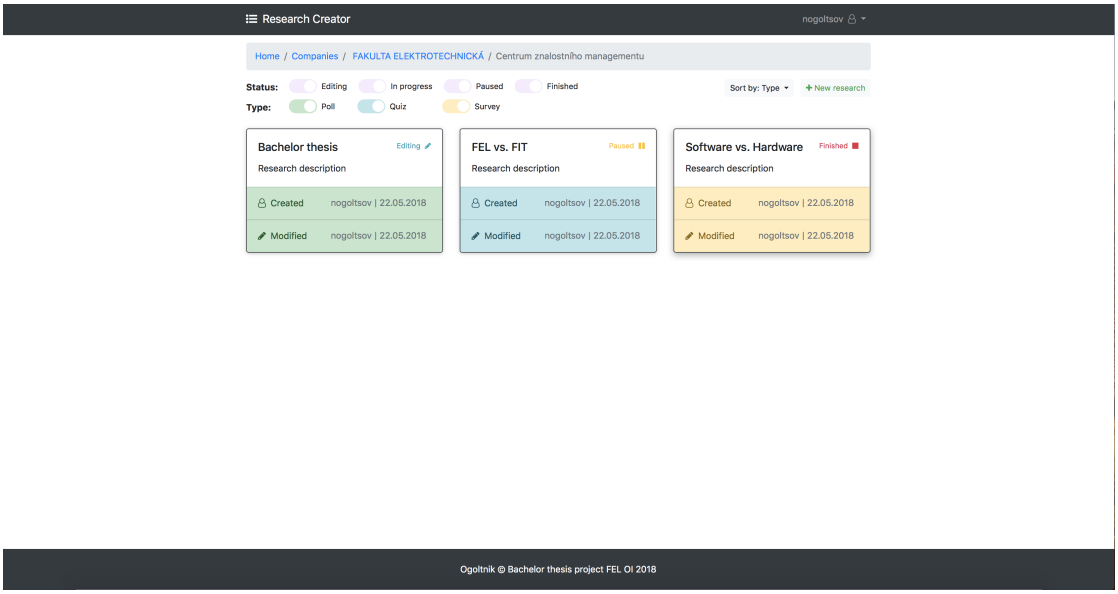
Obrázek F.7: Webová aplikace. Dialog vytváření firemního účtu



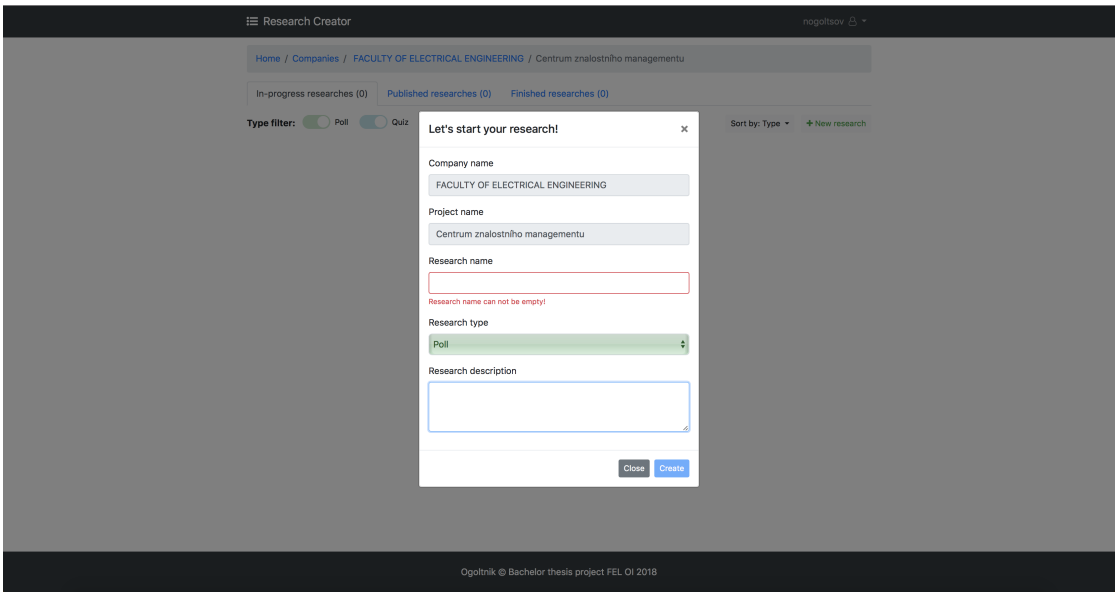
Obrázek F.8: Webová aplikace. Seznam projektů



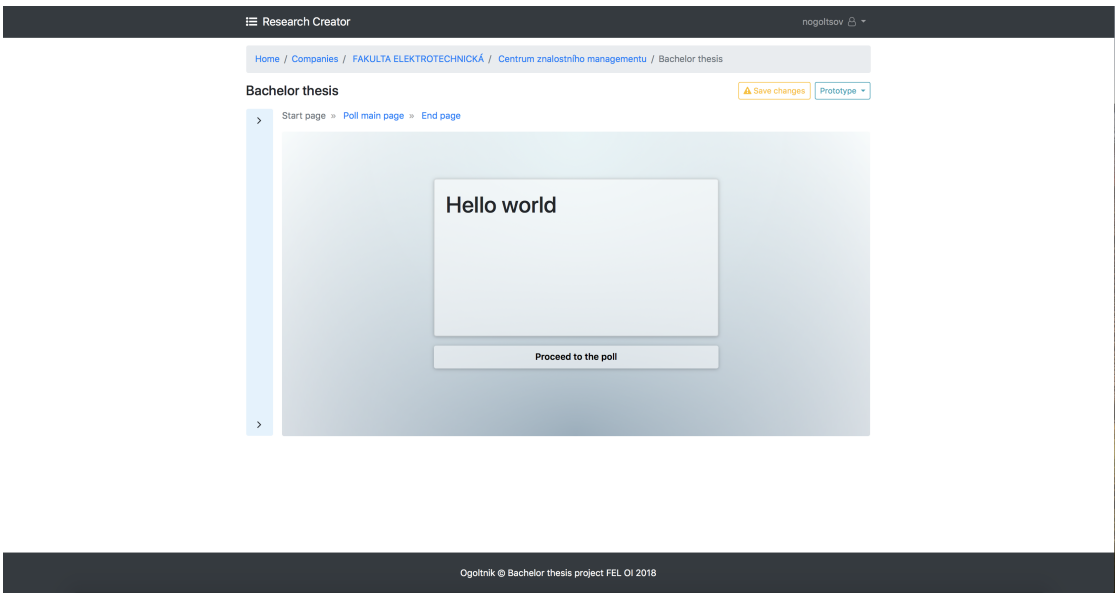
Obrázek F.9: Webová aplikace. Dialog vytváření projektu



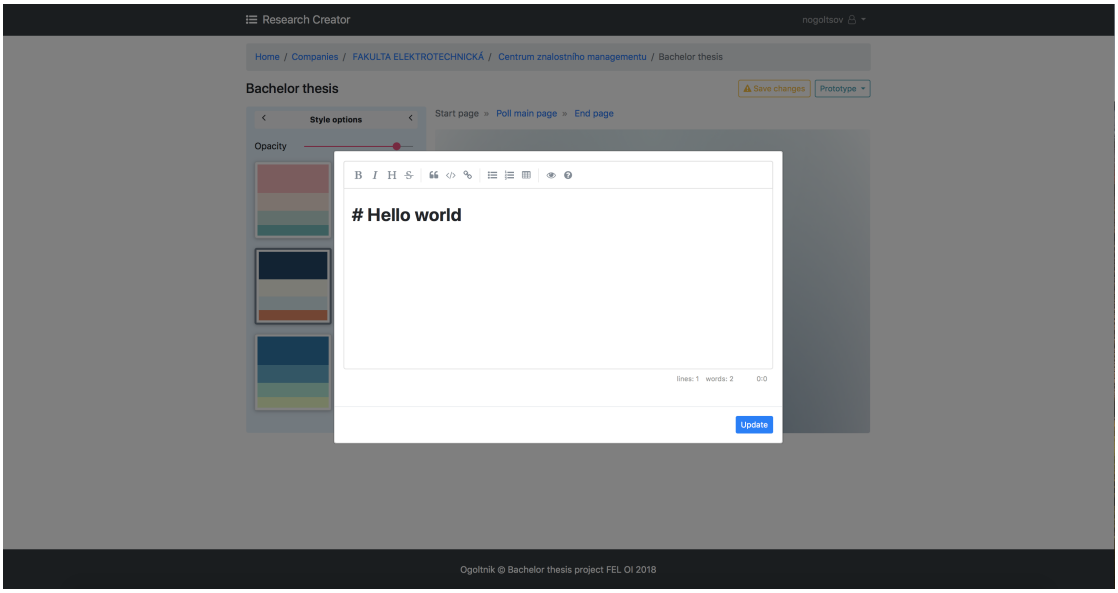
Obrázek F.10: Webová aplikace. Seznam průzkumů



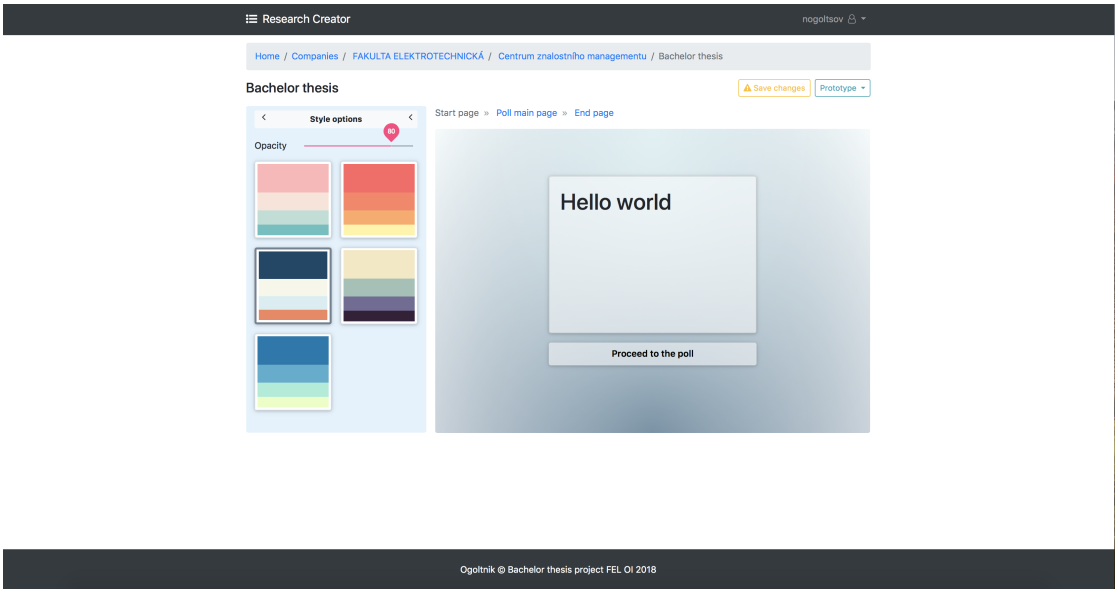
Obrázek F.11: Webová aplikace. Dialog vytváření průzkumu



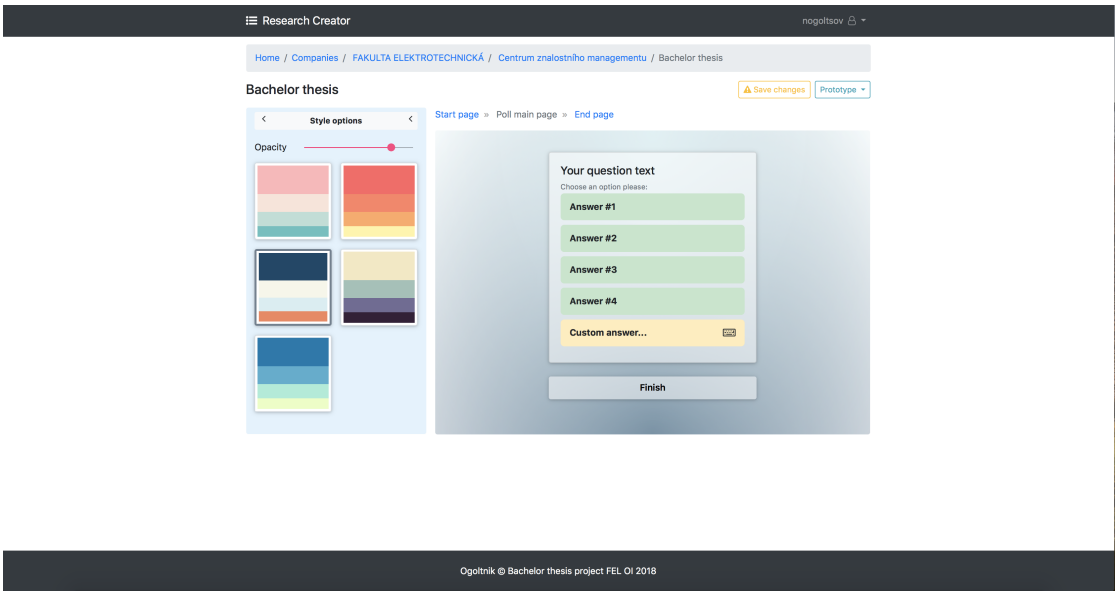
Obrázek F.12: Webová aplikace. Stránka editace prototypu průzkumu



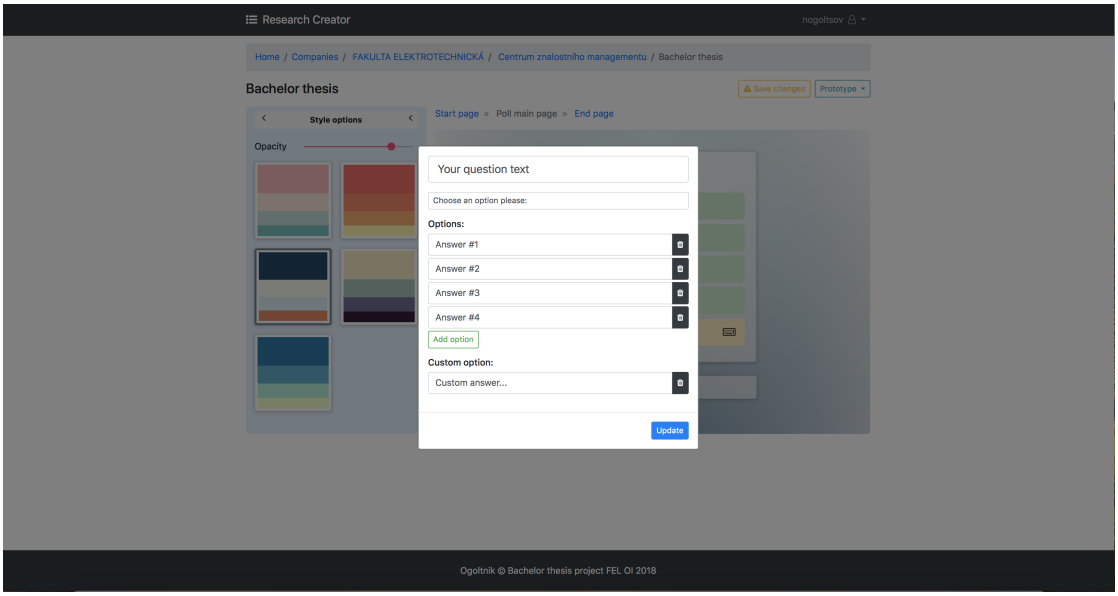
Obrázek F.13: Webová aplikace. Markdown dialog pro editaci textu



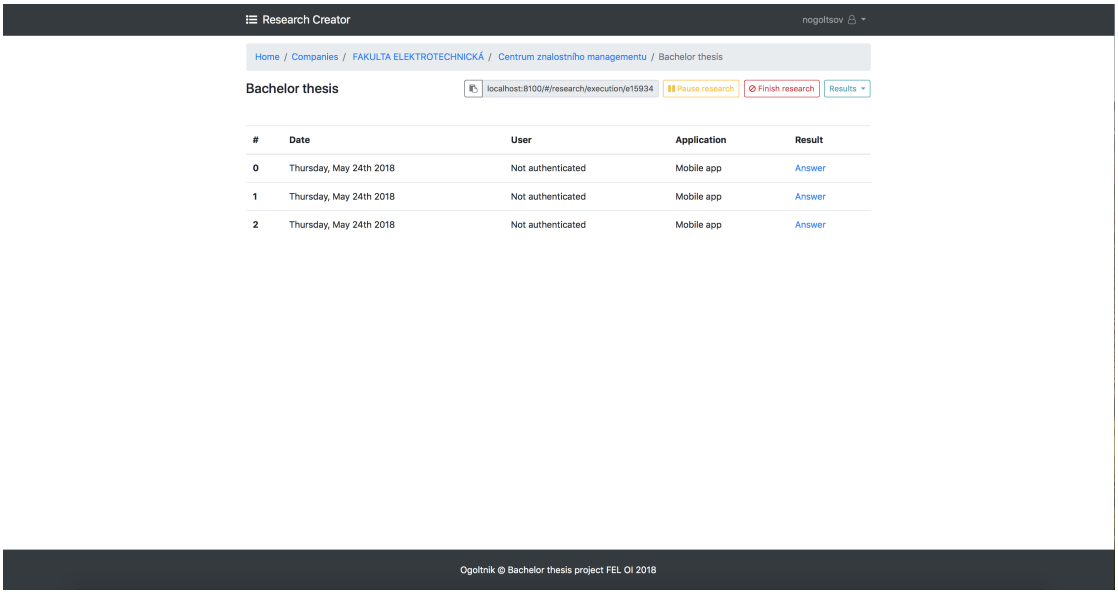
Obrázek F.14: Webová aplikace. Stylizace prototypu průzkumu



Obrázek F.15: Webová aplikace. Náhled elementu single-choice



Obrázek F.16: Webová aplikace. Editace elementu single-choice



Obrázek F.17: Webová aplikace. Seznam výsledků průzkumu