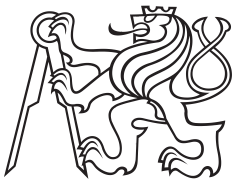
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Webová výuková aplikace Flashcards

Návrh a vývoj webové aplikace "flash cards"

Daniel Poustka

Vedoucí: Ing. Božena Mannová, Ph.D.
Obor: Softwarové inženýrství a technologie
Květen 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Poustka** Jméno: **Daniel** Osobní číslo: **491947**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Webová výuková aplikace flash cards

Název bakalářské práce anglicky:

Web learning application flash cards

Pokyny pro vypracování:

Úkolem bakalářské práce je vytvořit webovou výukovou aplikaci napodobující výuku pomocí papírových kartiček stylu otázka-odpověď, která by umožnila podporu samostudia a domácí přípravy na zkoušky..
1.Seznamte se s problematikou a proveďte rešerši dostupných existujících systémů, poskytující podobnou funkcionalitu.
2.Na základě vyhodnocení získaných poznatků, specifikujte požadavky na funkcionalitu navrhovaného systému.
3.Navrhněte architekturu systému
4.Seznamte se s technologiemi potřebnými pro vytvoření aplikace.
5.Zvolte nástroje pro implementaci a jejich volbu zdůvodněte.
6.Implementujte Frontend a Backend aplikace.
7.Otestujte aplikaci včetně uživatelských testů a výsledky vyhodnoťte.
Při zpracování využívejte prostředky softwarového inženýrství

Seznam doporučené literatury:

[1] Roger S. Pressmann Bruce Maxim: Software Engineering: A Practitioner's Approach , ISBN-10: 9780078022128
[2] https://www.ankiapp.com/
[3] https://collegeinfogeek.com/flashcard-apps

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Božena Mannová, Ph.D. kabinet výuky informatiky FEL

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

Datum zadání bakalářské práce: **11.02.2022** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **30.09.2023**

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

III. PŘEVZETÍ ZADÁNÍ

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

. | Datum převzetí zadání

Podpis studenta

Poděkování

Děkuji Ing. Boženě Mannové, Ph.D. za vedení mé bakalářské práce, zároveň tak za poskytnuté konzultace, profesionální přístup a užitečné rady, které celou práci obohatily. Též děkuji také rodině, přítelkyni, přátelům a svým studujícím kolegům za psychickou podporu.

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem k tomu pouze zdroje uvedené na konci práce, a to v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

V Praze, 20. května 2022

Abstrakt

Tato bakalářská práce je zaměřena na návrh a implementaci webové aplikace, která by umožnila podporu samostudia a domácí přípravy na zkoušky formou výukových karet - Flashcards.
Na základě provedené rešerše existujících řešení jsou definovány funkční i nefunkční požadavky, ze kterých se poté vychází při návrhu a implementaci vlastního řešení.
Výstupem implementační části je webová aplikace pro správu a práci s výukovými kartami, která byla otestována, nasazena a je plně funkční.

Klíčová slova: webová aplikace, aplikace, výuka, samostudium, flashcards, flashcard, kartičky, Java, SpringBoot, React, REST

Vedoucí: Ing. Božena Mannová, Ph.D. kabinet výuky informatiky FEL

Abstract

The Bachelor's thesis deals with design and implementation of the web application which would allow the support of self-study and home preparation for exams in the form of learing cards - Flashcards.
Based on research of existing solutions, functional and non-functional requirements are defined, which are then used to design and implement own solution.
The output of the implemetation part represents a web application for managing and working with learning cards which has been tested, deployed and fully functional.

Keywords: web application, application, education, self-education, flashcards, flashcard, cards, Java, SpringBoot, React, REST

Title translation: Web learning application Flashcards — Design and development of web application "flash cards"

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

1.1 Téma

Tématem bakalářské práce je webová výuková aplikace, napodobující výuku pomocí papírových kartiček stylu otázka-odpověď, která bude sloužit jako pomůcka pro přípravu ke zkouškám či výuku cizích slovíček, odborných výrazů a podobně strukturovaných otázek. Řešení bude využívat moderní webové technologie a nabízet přívětivé uživatelské rozhraní.

1.2 Cíl práce a cílová skupina

Cílem práce je seznámit se s výukovou metodou flashcards, provést rešerši aktuálních aplikací pro výuku touto formou, následně navrhnout a implementovat novou webovou aplikaci, která by sloužila jako učební pomůcka především pro studenty. Výstupem práce je spolu s detailní analýzou i fungující aplikace.

Hlavní funkce aplikace:

- Vytvářet kolekce karet
- Vytvářet karty v rámci kolekce
- Otáčet karty a odkrývat tím odpovědi k otázkám
- Sdílet karty jako veřejné kolekce karet

Více viz. podkapitola Funkční požadavky

Kapitola 2

Flashcards

Jednou z velmi populárních a oblíbených učebních metod současnosti jsou tak zvané flashcards, které nejčastěji nacházejí využití při výuce cizích jazyků, ale uplatnění mají daleko větší.

Jedná se nejčastěji o papírové kartičky, kde na jedné straně se nachází pojem a na straně opačné definice, překlad či dodatečné informace. Proces je následující: student přečte pojem na jedné straně a snaží se ho definovat. Pokud neví či odpoví špatně, na druhé straně si přečte správnou odpověď a kartu odloží stranou k dalšímu procvičení [2][3].

Papírová forma karet má hned několik výhod. Na výrobu postačí obyčejný papír, tužka, propiska a případně pastelky a nůžky, tudíž výroba je jednoduchá, rychlá a především levná. Dále jsou velice univerzální. Nejčastěji se s nimi učí cizí slovíčka, ale uplatnění najdou snad ve všech odvětví. Od vysvětlování pojmů, letopočtů v dějepise přes periodickou tabulku prvků až třeba po opakování vzorců v matematice. Lze se s nimi zkrátka učit cokoliv a díky své velikosti a skladnosti i kdekoliv.

Normální čtení zápisků a poznámek není efektivní, protože jen čtete text a nemusíte si na nic vzpomínat. Flashcards je forma aktivního učení a nutí informace z paměti neustále vyvolávat, což vede k upevňování znalostí a skutečnému učení. Zároveň vás nezahlcují informacemi souvislého textu, ale informace rozkládají do karet s vysvětlením či nápovědou. Forma učení lze obměňovat. Můžete si je číst postupně podle vodopádové metody, jindy si číst nejprve odpovědi [4].

Nevýhodou papírových memorovacích karet je nutnost ruční výroby, složitá úprava již vytvořených karet, jejich poškození nebo nebezpečí ztráty. Obtížné je i doplnění dalších údajů či technická omezení jako nemožnost využití složitých obrázků, audia, videí či webových odkazů.

Aplikace pro správu a výuku pomocí flashcards mají stejné nebo podobné výhody jako jejich papíroví sourozenci, navíc ale přidávají nové možnosti využití:

- Tvorba statistik nad kartami
- Dostupnost napříč zařízeními a / nebo systémy
- Možnost sdílení karet s přáteli
- Snadné editace karet či rozšiřování celých sad
- Možnost výuky formou hry
- Snadnější plánování výuky
- Notifikace a připomenutí výuky
- Šetření životního prostředí díky bezpapírovému provedení

Kapitola 3

Analýza

Tato kapitola se věnuje rešerši již existujících aplikací podporujících výuku pomocí karet flashcards a následné definici funkčních i nefunkčních požadavků a případů užití. Na základě této rešerše jsou poté zvoleny technologie, které budou použity pro vývoj serverové a klientské strany aplikace.

3.1 Existující řešení

Na trhu se již nachází různorodá řešení s podporou různých platforem a zařízení. Nejčastěji se jedná o **desktop, webový prohlížeč, android, iOS**. Tato řešení obsahují různé funkcionality, případy užití a cenové relace.
Cílem této kapitoly je tak získat představu o aktuálně nejpopulárnějších řešení, popsat jejich výhody a nevýhody a shrnout důležité poznatky.

3.1.1 Quizlet

Quizlet je zřejmě nejpopulárnější platforma pro výuku pomocí simulace papírových kartiček a jednoduchých her. Nabízí přívětivé intuitivní uživatelské rozhraní, které bude vhodné pro všechny, co hledají jednoduché a elegantní využití.

Mezi zajímavé funkce Quizlet přináší i vkládání grafiky a vlastních audio nahrávek do karet. Je možné si kartičky vytvářet ručně či využít již předpřipravené, ale většinou zpoplatněné, sady pro vzdělávání jazyků a jiných odvětví. Právě k výuce jazyků je připojeno i strojové předčítání slov, které umožňuje zjistit správnou výslovnost. Quizlet také částečně toleruje překlepy v testech, kdy pokud se uživatel výrazně přiblížil správné odpovědi, ale omylem některé písmeno zaměnil, nabídne se mu možnost ruční opravy.

Většina výše zmíněných funkcí je zamčeno za předplatným. Po jeho zaplacení zaplacení se otevře aplikace bez reklam, s možností offline režimu, analytiky výkonu uživatele a jeho zlepšování nebo využití grafických prvků při tvorbě karet [5][6][3].

Výhody:

- Aplikace je jednoduchá na používání.
- Nabízí spoustu výukových módů
- Podporuje všechny nejpoužívanější platformy.
- Možnost výuky hrou.

Nevýhody:

- Mnoho funkcionalit skryto za placeným členstvím.

3.1.2 Anki

Anki je bezplatnou aplikací pro tvorbu výukových kartiček pomocí desktopové (Windows i Linux), android i iOS aplikace. Na rozdíl od Quizlet se netěší lákavým designem ani gamifikovaným přístupem k učení. Prostředí není tak intuitivní a křivka učení práce s aplikací je strmější. Avšak po osvojení si základních operací a klávesových zkratek je tvorba karet prakticky okamžitá. Kartičky lze různě upravovat, formátovat, také lze přidat audio soubor či grafiku [7].

Také je možné vytvářet různé druhy karet. Kromě standardní i kartu s vynechaným textem k doplnění, kartu s prohozeným rubem a lícem a další. Tyto karty lze třídit a filtrovat podle různých proměnných jako téma, druh, štítek či kolekce.

Nabízí se zde i možnost sdílení kolekcí karet pomocí databáze, což šetří čas při vytváření vlastních balíčků. Karty a celé databáze lze také synchronizovat přes připojenou webovou službu.

Pro výuku je zde systém chytrého opakování, podle kterého losuje kartičky. Dokáže tak efektivně vštěpit informace minimálním počtem opakování. Systém lze i nastavit na požadovanou úroveň procentuální znalosti balíčku.

Aplikace lze také rozšířit o různé pluginy (rozšíření), díky kterým umožní výuku obrázků a jejich popisků, efektivnější tvorbu karet, lepší organizaci kartiček, poskytnutí statistik o opakování nebo motivaci k dalšímu vzdělávání [5][3][8].

Výhody:

- Aplikace je kompletně zdarma jako Open source[1] program.
- Nabízí širokou paletu úprav, funkcí a rozšíření.
- Jedná se o mocný nástroj pro tvorbu kartiček.

Nevýhody:

- Proti konkurenci zastaralé a neintuitivní uživatelské rozhraní.

[1] Počítačový software s veřejným zdrojovým kódem a možností podílení se na jeho tvorbě[9].

3.1.3 Brainscape

Brainscape se zdá na povrchu jako vcelku jednoduchá aplikace pro tvorbu a správu flashcard karet, ale nabízí i pokročilé funkce pro sledování pokroku a samotného zlepšování ve znalostech. Obecně se přístupem podobá aplikaci Anki, ovšem v hezčím a intuitivnějším kabátku [10].

Samotné vytváření karet a jejich sad je jednoduché a rychlé. Nabízí se i využití již předpřipravených sad či ty vytvořené sdílet s ostatními. K dispozici jsou i doplňující funkce jako pokročilý editor karet či možnost připojit grafiku či zvukové soubory, avšak je nutné si zakoupit předplatné.

K výuce nabízí i tzv. metodu "spaced repetition", která funguje na principu intervalových opakování za použití karet. S každou kartičkou jde provést i sebehodnocení na stupnici od jedné do pěti hvězdiček podle vlastního uvážení. Na základě tohoto hodnocení se následně definuje výše zmíněný časový interval potřebný k dalšímu procvičení výrazu.

Brainscape je dostupné na všech platformách formou webové aplikace, navíc i jako nativní aplikace pro mobilní zařízení. Díky tomu zajišťuje i synchronizaci mezi vícero zařízeními, které mohou být různého druhu [3][8].

Výhody:

- Možnost vlastního ohodnocení.
- Následný systém intervalů opakování.

Nevýhody:

- Verze zdarma je velmi ořezaná, nutnost zakoupit předplatné.

3.1.4 Cram

Cram je standardní flashcard aplikací, podporující různé způsoby učení formou kartiček včetně her [11].

Zajímavostí je možnost vytvořit ke každé kartičce nápovědu, která simuluje reálného zkoušejícího. Na základě odpovědi se uživatel ohodnotí, zda odpověděl dobře či špatně. Karty, které jsou zodpovězeny správně, se již dále v testu nezobrazují. Aplikace nabízí nejen výuku formou klasického otáčení karet, ale lze i automaticky vytvořit karty s možností vícero odpovědí či s doplněním chybějící fráze.

Aplikace Cram je dostupná na desktopu formou webové aplikace nebo nativní aplikace pro mobilní zařízení. Zajišťuje i synchronizaci uživatelské knihovny mezi vícero zařízeními [3][8].

Výhody:

- Možnost výuky hrou.
- Jednoduchost.

Nevýhody:

- Reklamy ve verzi zdarma.
- Málo pokročilých funkcí.

3.1.5 Shrnutí

V této kapitole byly shrnuty nejpopulárnější aplikace pro výuku formou karet flashcards. Souhrnný přehled je zobrazen v tabulce 3.1

Na základě tohoto průzkumu byly vyvozeny důsledky pro návrh vlastní webové aplikace.

	Quizlet	Anki	Brainscape	Cram
Platforma	Web. aplikace + nativní mobilní aplikace	Windows, Linux + nativní mobilní aplikace	Web. aplikace + nativní mobilní aplikace	Web. aplikace + nativní mobilní aplikace
Finanční plán	Členství	Opensource	Předplatné	Předplatné, reklamy
Verze zdarma	Ano; velmi ořezaná	Ano	Ano; velmi ořezaná	Ano; některé funkce uzamčeny, reklamy
Reklamy v aplikaci	Ano (ve verzi zdarma)	Ne	Ne	Ano (ve verzi zdarma)
Design	Elegantní	Zastaralý, neintuitivní	Elegantní	Elegantní
Výukové režimy	Ano	Ano	Ano	Ano
Statistiky	Ano	Ano	Ano	Ano
Šablony karet	Ano	Ano	Ano	Ano
Multimédia v kartách	Ano; nutné členství	Ano	Ano; nutné předplatné	Ano
Pokročilé funkce	Ano; nutné předplatné	Ano; vyžaduje plugin	Ano; nutné předplatné	Málo
Předefinované sady	Ano; většinou zpoplatněné	Ne	Ano	Ano
Pluginy	Ne	Ano	Ne	Ne
Synchronizace napříč zařízeními	Automatická	Manuální	Automatická	Automatická
Sdílení kolekcí	Ano	Ne	Ano	Ano
Cena	686Kč/rok	Zdarma (iOS aplikace jednorázově $25)	$9.99/měsíc, doživotně $129.99	$5

Tabulka 3.1: Srovnání populárních Flashcard aplikací

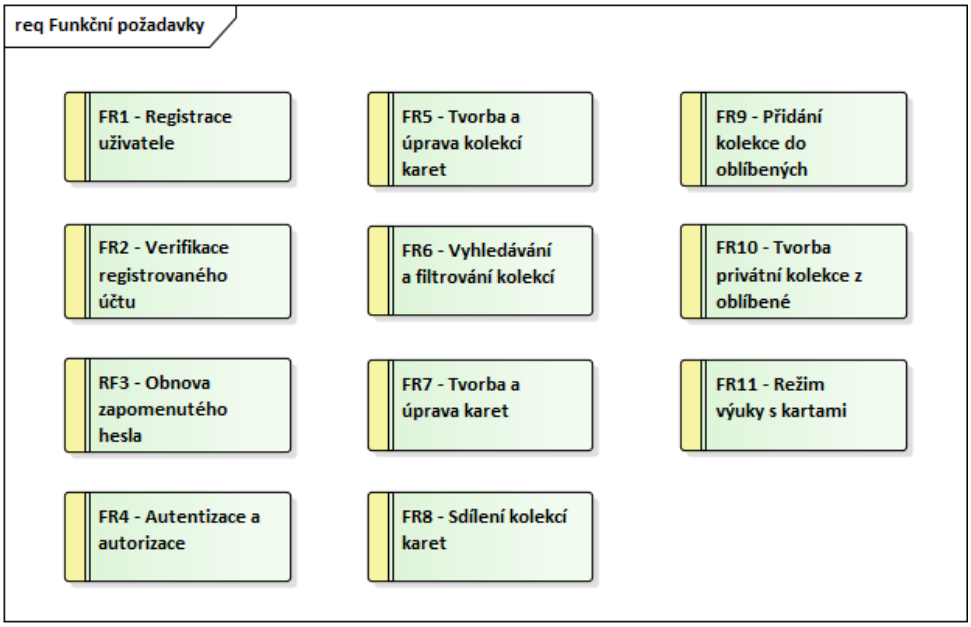
3.2 Analýza a sběr požadavků

Tato sekce je zaměřena na vytyčení hlavních požadavků pro návrh aplikace. Jejich definice je založená zadání práce, analýze existujících řešení, pravidelných konzultací s vedoucím práce, podstaty navrhované aplikace a vlastní iniciativě autora.
Posbírané požadavky byly následně rozděleny do 2 kategorií:

1. **Funkční požadavky** - definují chování, funkcionalitu, nezbytné úkoly či akce, které navrhovaná aplikace musí poskytovat nebo vykonávat. V následujícím textu jsou označené *FRn*, kde písmeno *n* označuje číslo funkčního požadavku.
2. **Nefunkční požadavky** - definují požadavky, které se přímo netýkají byznysové funkcionality, ale zaměřují se na architekturu aplikace, uživatelské rozhraní či prostředí pro běh aplikace [12]. V následujícím textu jsou označené *NFRn*, kde písmeno *n* označuje číslo nefunkčního požadavku.

3.2.1 Funkční požadavky

Pro aplikace bylo na základě analýzy vydefinováno celkem 11 hlavních funkčních požadavků. Jejich výčet je znázorněn na obrázku 3.1.
Podrobný popis funkčních požadavků je rozčleněn do podkapitol níže.



Obrázek 3.1: Funkční požadavky

- **FR1 - Registrace uživatele**

  Aplikace bude umožňovat registraci nového účtu všem neautentizovaným uživatelům. Pro vytvoření nového účtu je vyžadována platná

emailová adresa, uživatelské jméno a alespoň osmimístné heslo s využitím nejméně jednoho velkého a malého písmena. Tímto způsobem je možné registrovat pouze účet běžného uživatele.

- **FR2 - Verifikace registrovaného účtu**

  Po úspěšné registraci bude vyžadováno ověření registrované emailové adresy zasláním verifikačním emailem se speciálním odkazem obsahující verifikační token. Po otevření odkazu s platným tokenem v prohlížeči bude účet aktivován. V opačném případě si půjde ověřovací email znovu vyžádat.

- **FR3 - Obnova zapomenutého hesla**

  Pokud uživatel své heslo zapomene, aplikace umožní heslo resetovat na základě zaslání zprávy na registrovaný email s odkazem obsahující unikátní token. Po otevření odkazu s platným tokenem bude možné nastavit heslo nové.

- **FR4 - Autentizace a autorizace**

  Po registraci a aktivování účtu je možné se přihlásit pomocí registrované emailové adresy a hesla. Pokud je přihlášený uživatel administrátor, získá i náležitá oprávnění pro správu aplikace.

- **FR5 - Tvorba a úprava kolekcí karet**

  Aplikace bude umožňovat vytvářet a spravovat osobní kolekce karet.

- **FR6 - Vyhledávání a filtrování v kolekcích**

  Jednotlivé kolekce půjde filtrovat na základě viditelnosti či kategorií. Aplikace umožní i vyhledávání podle názvu sady karet.

- **F76 - Tvorba a úprava karet**

  Aplikace bude umožňovat vytvářet a spravovat karty v rámci vytvořených kolekcí

- **FR8 - Sdílení kolekcí karet**

  Aplikace bude umožňovat veřejně sdílet vytvořené kolekce s ostatními uživateli. Výsadní práva úprav zůstávají autorovi.

- **FR9 - Přidání kolekce do oblíbených**

  Veřejné kolekce půjde označit jako oblíbené. Tyto kolekce se budou následně zobrazovat v seznamu oblíbených kolekcí. V tomto stavu je však nepůjde upravovat, protože výsadní práva úprav náleží autorovi kolekce.

- **FR10 - Tvorba privátní kolekce z oblíbené kolekce**

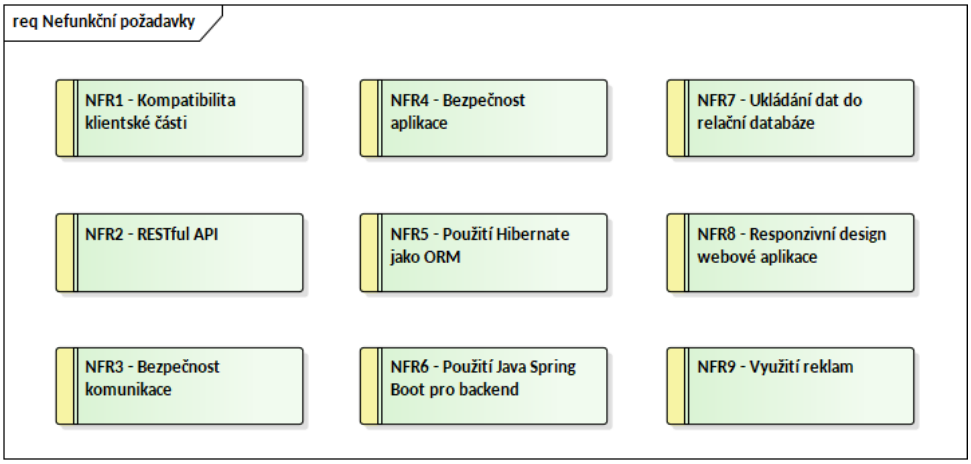
  Uložené oblíbené kolekce aplikace umožní zduplikovat jako privátní kolekci. Tato sada karet již nemá vazbu na originál a uživatel již má oprávnění kolekci měnit a přizpůsobovat.

- **FR11 - Režim výuky s kartami**

  Aplikace bude umožňovat spustit režim výuky nad vybranou kolekcí karet.

3.2.2 Nefunkční požadavky

Celkem bylo pro aplikaci na základě zadání a analýzy vybráno 9 hlavních nefunkčních požadavků. Jejich výčet je znázorněn na obrázku 3.2.
Podrobný popis nefunkčních požadavků je rozčleněn do podkapitol níže.



Obrázek 3.2: Nefunkční požadavky

- **NFR1 - Kompatibilita klientské části**

  Uživatelské rozhraní bude kompatibilní s aktuálními verzemi populárních webových prohlížečů, to znamená Google Chrome verze 96, Mozilla Firefox verze 94 a Safari 14 na iOS.

- **NFR2 - RESTful API**

  Serverová část aplikace bude vystavovat RESTful API umožňující veškeré potřebné CRUD operace, včetně vyhledávání či filtrování. Využití REST rozhraní nabízí možné budoucí rozšíření nativními aplikacemi.

- **NFR3 - Bezpečnost komunikace**

  Zabezpečení veškeré komunikace klientské části s REST API bude realizováno HTTPS protokolem. Potřebný certifikát SSL bude zajištěn provozovatelem po úspěšné implementaci a následném nasazení do produkčního prostředí.

- **NFR4 - Bezpečnost aplikace**

  Aplikace bude splňovat realizaci přihlašování pomocí standardu OAuth2. Komunikace přihlašování bude zabezpečena protokolem HTTPS viz. N2.1. Po přihlášení získá uživatel identifikační token omezený na

dané sezení (session). Po vypršení autorizačního tokenu bude nutné znovu zadat přihlašovací údaje.

- **NFR5 - Použití Hibernate jako Object-Relational Mapping (ORM)**

  Jako objektově relační mapování bude použita knihovna Hibernate. Tento požadavek vyplývá ze zadání práce.

- **NFR6 - Použití Java Spring Boot pro serverovou stranu**

  Pro serverovou stranu bude použit framework Java Spring Boot. Tento požadavek vyplývá ze zadání práce.

- **NFR7 - Ukládání dat do SQL databáze**

  Aplikace bude splňovat perzistentní ukládání dat využitím SQL relační databáze. Tento požadavek vyplývá z druhu ukládaných dat a zadání práce.

- **NFR8 - Responzivní design webové aplikace**

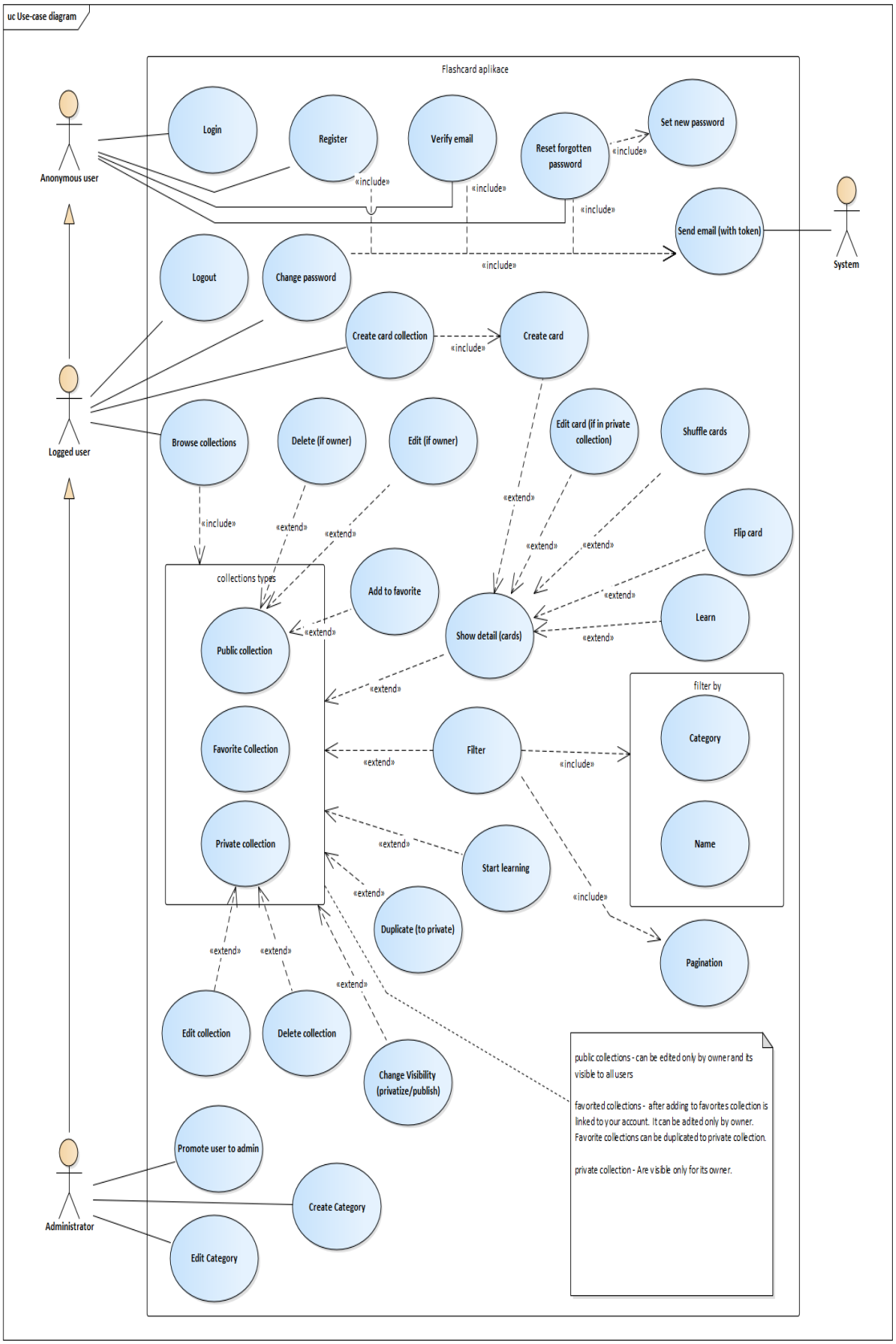
  Webové rozhraní bude realizováno jako Single-Page-Application. Zobrazení stránky musí vyhovovat desktopovým PC i zařízením typu mobilní telefon s menší zobrazovací plochou.

- **NFR9 - Využití reklam**

  Do frontendu bude začleněna reklama pro pokrytí výdajů za provoz, případně zisk. Poskytovatelem reklamy bude společnost Google přes službu GPT (Google Publisher Tags).

3.3 Případy užití

V této kapitole si představíme definované aktéry, kteří budou s navrhovanou aplikací interagovat, a jednotlivé případy užití (use-cases), identifikované v předchozí kapitole section 3.2. Jejich přehled a znázornění vztahů je na obrázku 3.3



Obrázek 3.3: Diagram případů užití

3.3.1 Aktéři

V rámci sběru a analyzování požadavků byli vydefinování tři různí aktéři pro případy užití, kteří budou s navrhovaným softwarem interagovat. Tito aktéři jsou následující:

1. **Anonymní uživatel** - Aktér představující anonymního návštěvníka webové aplikace.
2. **Přihlášený uživatel** - Aktér představující již přihlášeného běžného uživatele webové aplikace.
3. **Administrátor** - Aktér představující již přihlášeného uživatele s administrátorskými právy v rámci navrhované aplikace.

3.4 Technologie

Použité technologie částečně vycházejí z analýzy požadavků. I tak je ale potřeba se před samotným návrhem obeznámit s populárními řešeními a technologiemi, jejich silnými a slabšími stránkami.
Při výběru se braly v potaz požadavky na systém, cíl práce, jednoduchost řešení, kvalita dokumentace technologie i osobní zkušenost.

3.4.1 Databáze

Databáze je organizovaná kolekce strukturovaných informací, dat či údajů uložených typicky v počítačovém systému [13]. Přinášení efektivní persistenci dat pro různé druhy aplikací. Umožňují rychlé a bezpečné čtení, vytváření, úpravu či mazání velkého množství dat pro více uživatelů zároveň.

Existují různé druhy databázových systémů lišící se implementací dané firmy či určením. Každý z těchto systémů má však své pozitivní i negativní stránky.

SQL vs. NoSQL

Základní rozdělení databází je na SQL a NoSQL. Zkratka SQL znamená *structured query language* neboli strukturovaný dotazovací jazyk. Jak tedy zkratka napovídá, SQL databáze mají předem pevně danou strukturu nebo schéma pro definování a manipulaci s daty. Pozdější změna schématu může být náročná a složitá při neporušení existujících dat.

Naopak NoSQL databáze mají schéma dynamické a data mohou být bez struktury či členění. Tyto data se mohou ukládat v následujících formách: sloupcově orientovaná, dokumentově orientovaná, grafová nebo formou klíč-hodnota [14].
Nevýhodou ovšem je nemožnost dosáhnout plnohodnotné podpory transakč-

ního ACID[2] modelu [15].

Nyní se podíváme na dostupné nejpopulárnější technologie databázových systémů.

Pokud bychom zvažovali NoSQL databázi, v současné době jsou populární následující NoSQL databázové systémy: MongoDB, Amazon DynamoDB, CouchBase, Neo4j a další.

U navrhované aplikace budeme předem znám danou podobu dat a ACID vlastnosti jsou pro takovou enterprise aplikaci důležité, proto se dále zaměříme pouze na SQL databáze.

Oracle

Firma Oracle nabízí multiplatformní řešení databázového systému s pokročilými možnostmi zpracování dat, jednoduchou škálovatelností a vysokým výkonem oproti konkurenci.
K dispozici je řešení v cloudu i na vlastním serveru skrze licence.
Oracle databáze je vhodná především pro rozsáhlé aplikace velkých firem.

Klady: [16]

- Podpora SQL i NoSQL
- Podpora PL/SQL příkazů
- Platformová nezávislost
- Vysoký výkon databáze
- Podpora vícero modelů databáze
- Bezpečnost

Zápory:

- Cena - standardní cloudové edice databáze na virtuálním stroji vychází v přepočtu na 4,5Kč / hodinu s Oracle CPU [17].
- Složitější instalace

MySQL

MySQL je relační databázový systém, který v roce 2010 koupil a nyní zastřešuje Oracle jako Open source[3] software [18].
Databáze podporuje i tzv. pohledy, triggery, uložené procedury, několik druhů úložišť a komunikace probíhá pomocí SQL dotazů.
Od počátků byla optimalizována především pro rychlost, to si vyžádalo cenu některých zjednodušení ci bezpečnostních rizik [19].

[2] Čtyři databázové vlastnosti: atomicita (**A**tomicity), konzistence (**C**onsitency), izolovanost (**I**solation), trvalost (**D**urability)

[3] Počítačový software s veřejným zdrojovým kódem a možností podílení se na jeho tvorbě[9].

Využití nachází především v konfiguracích pro implementaci webových stránek v souboru technologií LAMP[4].

Klady:

- Zdarma (open source)
- Podpora ACID (při použití InnoDB společně s NDB Cluster Storage Engine)
- Podpora některých pokročilých funkcí (pohledy, uložené procedury, triggery)
- Platformová nezávislost
- Rychlost
- Možnost výběru z několika druhů úložišť

Zápory:

- Horší bezpečnost
- Méně propracované zálohování
- Nepodporuje plný SQL standard

MSSQL

Microsoft SQL Server je relační databázový a analytický systém, vyvíjený firmou Microsoft. Využití nachází jak u podnikových systémů, webových aplikací, tak i u datových skladů. Je zřejmě největším konkurentem pro databázové systémy společnosti Oracle.

Klady: [16]

- Multiplatformní
- Vysoký výkon databáze
- Podpora vícero modelů databáze
- Bezpečnost

Zápory:

- Relativně vysoká šance na chybu či poškození dat
- Složitější instalace
- Nepodporuje PL/SQL příkazů

[4]Kombinace open source technologií pro implementaci webových stránek, nejčastěji: systém GNU/**L**inux, web. server **A**pache, databáze **M**ySQL, jazyk **P**HP[20].

PostgreSQL

PostgreSQL, zjednodušeně Postgres, je jednou z nejpopulárnějších Open source objektově-relačních databázových systémů [21].
Databáze splňuje jak kompletní ACID přístup, tak i ANSI SQL standard. Také nabízí jak funkce jako pohledy, triggery, uložené procedury a cizí klíče, tak i pokročilé funkce: například uživatelem definovaných datových typů, formátu JSON, full-textové vyhledávání, ukládání obrázků, videa, audia a další [22].
Využití nachází u podnikových aplikací, datových skladů nebo obdobně jako soubor nástrojů LAMP u dynamických webových stránek různých velikostí [23].

Klady:

- Zdarma (open source)
- Podpora PL/pgSQL (obdoba PL/SQL od firmy Oracle) a scriptovacích jazyků
- Podpora různorodých doplňků
- Odolnost vůči chybám a nekonzistencím
- Bezpečnost
- Multiplatformní
- Podpora pokročilých funkcí

Zápory:

- Nepodporuje NoSQL
- V porovnání s databázemi od Oracle či Microsoft zaostává ve výkonu
- Rozšiřitelnost řešení
- Kvůli otevřenosti kódu nastávají problémy s implementací funkcí oproti uzavřeným řešením

Zvolené řešení

Dle předchozí analýzy bude pro ukládání dat použit databázový systém typu SQL. Z vybraných databázových implementací dopadlo řešení PostgreSQL pro účel navrhované aplikace nejlépe [24]. Svoji konkurenci předčil nejen cenou (zdarma) ale i výkonem a pokročilými funkcemi.
Pro další srovnání následuje tabulka 3.2 vybraných systémů:[1]

	SQL Server	Oracle	MySQL	PostgreSQL
Nezávislost na platformě	Ne	Ano	Ano	Ano
Vhodné řešení pro webový Frontend	ASP.NET	Java	PHP	PHP
Cena	$24,999 (Enterprise) $5,999(Standard) $49(Developer) zdarma(Express)	Enterprise: $40,000 / CPU $800 / uživatel RAC: $20,000 / CPU $400 / uživatel Security Pack : $10,000 / CPU	Zdarma €479 za podporu	Zdarma
Business Intelligence	Ano	Ano	Ne	Ne
Bezpečnost	Vynikající	Dobrá (pomalé opravy a více průniků)	Špatná (více mezer a pomalé opravy)	Vynikající
Podpora XML	Uložení, indexace, validace, kontrola obsahu	Uložení	Žádná	Uložení, indexace, validace
Administrace	Snadná, přehledná	Nutno hlouběji znát problematiku databází	Nástroje třetí strany pro správu přes PHP	Nástroje pro správu přes PHP nebo klientskou aplikací
Vhodná velikost projektu (informativní)	Střední a velké projekty pro Enterprise edici, malé a střední pro Standard a Express	Velké projekty	Malé a střední	Střední a velké

Tabulka 3.2: Srovnání populárních SQL databází [1]

Pro ukládání dat byl zvolen SQL databázový systém PostgreSQL, pro jeho shodnost účelu, efektivitu a poměr cena/výkon. Toto rozhodnutí vyplývá z provedené analýzy, zadání práce i zkušeností autora.

3.4.2 Serverová část

Vývoj webových aplikací se logicky dělí na dva celky - klientskou a serverovou část. Obě části mají příslušnou sadu technologií a nástrojů. Tato podkapitola se zaměřuje na technologie používané na straně serveru.

Node.js

Node.js představuje framework, který je schopen spouštět kód napsaný v JavaScriptu v serverovém prostředí. Technologie je postavena na JavaScript enginu prohlížeče Chromium a obsahuje stejný základ, jako ve jmenovaném prohlížeči, tudíž zpracování kódů je velmi rychlé. Jádro celého Node.js tvoří tzv. smyčka událostí (event loop). Do ní vstupují všechny uživatelské požadavky jako události, které jsou poté přiděleny jednotlivým nezávislým vláknům.

Express

Express je rychlý a minimalistický framework, který funguje jako nadstavba pro dříve zmíněný Node.js [25].
Nabízí paletu základních funkcí pro vývoj webových aplikací, aniž by zastiňoval výhody Node.js. Umožňuje snadné vytvoření robustního API a velké množství dostupných rozšíření.
Existuje také mnoho dalších frameworku, které jsou na Expressu postaveny, příkladem může být: Feathers, ItemsAPI, KeystoneJS, Kraken, Sails, Poet a další.

Na frameworku Express běží například webové stránky firem Uber, IBM nebo Accenture [26].

Laravel

Laravel je PHP framework určený pro vývoj webových aplikací poprvé vydaný v roce 2011 [27].

Co se týče vývoje na PHP, Laravel obsahuje téměř vše, co je potřeba pro vytvoření robustní serverové strany: PHPUnit je framework pro testování. Artisan jako rozhraní příkazového řádku pro migraci databáze a vytváření modelů. Dále obsahuje vlastní systém pro směrování, obdobně jako Spring Boot poskytuje kontejnery Dependency Injection pro propojení jakýchkoliv tříd, služeb či modulů. Laravel také nabízí vlastní cache a session jádra připravené ihned k použití [28].

Laravel je jedním z nejoblíbenějších PHP frameworků, a protože PHP má majoritní podíl na všech webových stránkách [29], jistě stojí za zmínku.

Na frameworku Laravel běží populární webové portály, jako jsou Laracast, Deltanet Travel, World Walking, Neighborhood Lender, MyRank a další [30].

Java Spring Boot

Spring Boot [31] je framework pro vývoj backendů, založený na jazyce Java. Využívá se jak pro monolitické, tak i microservices aplikace libovolných rozměrů, založených na frameworku Spring.

Na frameworku Spring Boot běží populární webové portály, jako jsou Trivago, Via Varejo a Intuit.

Klady:

- Podporuje HTTP servery Tomcat nebo Jetty, což usnadňuje nasazení aplikace a odpadá nutnost programátora zabývat se spouštěním aplikačního serveru.
- Usnadňuje použití softwarového patternu známého jako „Convention over Configuration“ [32].
- Snadná integrace s ekosystémem Spring, který obsahuje balíčky Sprig Data, Spring Security, Spring ORM, Spring JDBC a další.
- Je možné nakonfigurovat a vygenerovat spustitelný projekt s minimálním úsilím.
- Velkou výhodou je i předchozí zkušenost autora s frameworkem díky předmětům na fakultě, zabývající se jazykem Java a frameworkem Spring Boot.

Nevýhody:

- Složitý převod starých či existujících projektů ze Spring Framework na Spring Boot Application.
- Jedná se o rozsáhlý framework, může tedy zabrat nějaký čas se v něm zoriantovat.

Shrnutí

Dle rešerše a zkušeností byly pro implementaci serverové části použity následující technologie. Jako framework Java Spring Boot. Pro ukládání dat do databáze se využije Hibernate jako ORM. Komunikace s klientskou částí bude zajištěna vystavením REST API za použití úsporného formátu dotazů JSON, který má díky struktuře klíč-hodnota úspornou velikost a je podporován jak technologií Java Spring Boot, tak i technologiemi pro tvorbu frontendového rozhraní.

3.4.3 API protokol

Pro komunikaci serverové a klientské části aplikace je potřeba definovat nějaký soubor pravidel, funkcí či procedur, podle kterých se budou předávat data, vykonávat operace, atd.
K tomuto účelu slouží předem definované **Application Programming Interface (API)**.
API je abstrakcí, která definuje rozhraní pro interakci s nabízenými funkcemi serverové strany.

K přístupu na webové stránky se využívá protokol HTTP, proto i komunikace s backendem bude zajišťovat některé z existujících Webových API, které využívají protoklu HTTP [33].

K implementaci API lze přistoupit třemi různými způsoby:

SOAP

SOAP neboli Simple Object Access Protocol je dobře definovaný protokol, který byl navržen především jako API pro rozsáhlé korporátní systémy. Je určený pro volání procedur za využití formátu zpráv XML s cílem snadného sdílení informací z různých aplikací, které běží v různých prostředí nebo jazycích.

Tento druh API se tedy snaží odstranit bariéry mezi odlišnými platformami, ale na současné poměry je příliš těžkopádný a kvůli využití "upovídaného"XML formátu jsou zprávy zbytečně velmi objemné. Nehodí se tedy příliš pro menší či moderní projekty.

V současné době je využíván zejména bankami a pojišťovnami.

Ukázku jednoduché komunikace můžete vidět níže v ukázce 3.1 a 3.2 : [34]

Listing 3.1: SOAP dotaz

```
<soapenv:Envelope
xmlns:soapenv="http://sch.xmlsoap.org/soap/envelope/"
xmlns:sch="http://www.soapexample.com/xml/users">
 <soapenv:Header/>
 <soapenv:Body>
  <sch:UserDetailsRequest>
   <sch:name>John</sch:name>
  </sch:UserDetailsRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Listing 3.2: SOAP odpověď

```
<soapenv:Envelope
xmlns:soapenv="http://sch.xmlsoap.org/soap/envelope/">
 <soapenv:Header/>
 <soapenv:Body>
  <ns2:UserDetailsResponse
  xmlns:ns2="http://www.soapexample.com/xml/users">
   <ns2:User>
    <ns2:name>John</ns2:name>
    <ns2:age>5</ns2:age>
    <ns2:address>Greenville</ns2:address>
   </ns2:User
  </ns2:UserDetailsResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

GraphQL

GraphQL je dotazovacím jazykem pro serverové API, jehož prioritou je posílat jen ta data, o která jste si v dotazu napsali [35].
Tento druh API je díky komunikaci HTTP dotazy platformově nezávislé a oproti REST, kde pro každou funkcionalitu je vyžadován endpoint, GraphQL staví pouze na jediném přístupovém URI bodě.
Tím je /graphql, na který se následně posílají dotazy typu mutation, query nebo subscription.

Frontend je tedy jednodušší co se týče dotazování serveru, ale tato logika je přesunuta na backend, kde naopak přidělává práci vývojáři. Složitější je také využívat cache pro dotazy. Přesto se ale jedná o velkého konkurenta REST API, ne však ve smyslu, že by ho mohl nahradit, ale jedná se o alternativu.

Ukázku jednoduché komunikace můžete vidět níže v ukázce 3.3 a 3.4 : [34]

Listing 3.3: GraphQL dotaz

```
{
  person {
    name
    height
    age
  }
}
```

Listing 3.4: GraphQL odpověď

```
{
  "person": {
    "name": "John Doe",
    "height": 1.72,
    "age": 33
  }
}
```

REST

REST, nebo REpresentational State Transfer, je sada architektonických stylů a pravidel pro tvorbu API rozhraní [36].
Pokud API dané aplikace dodržuje REST principy, jedná se o tzv. RESTful API.
Tyto pravidla [37][38] jsou:

- **Klient-server** - Rozdělení aplikace na dvě části (klient a server), které je možné vyvíjet nezávisle a zajistit tak možnost dalších aplikací skrze platformy využívající stejný server.
- **Bezstavová komunikace** - Každý dotaz vedený na server musí obsahovat veškeré potřebné informace. Server nemůže využívat dříve uložený kontext informací. Pro zajištění nejnutnějších stavů komunikace se udržuje stav sezení (session).
- **Kešovatelnost dat** - Pokud lze odpověď dotazu kešovat, umožňuje to serveru znovupoužití dříve vytvořené odpovědi pro ekvivalentní dotaz bez nutnosti využití zdrojů.
- **Jednotné rozhraní** - Rozhraní musí umožňovat klientovi pracovat s různými zdroji stejným způsobem.
- **Vrstevnatost aplikace** - Komponenty aplikace lze na sebe napojovat. Tyto vrstvy mohou interagovat pouze o jednu úroveň dále, s žádnou další nesmí.
- **Kód na vyžádání** - Umožnění stáhnout a vykonat kód v prostředí klienta (nepoviné).

Rozhraní typu REST komunikuje protokolem HTTP dotazů, kde využívá metod GET, POST, UPDATE a DELETE, které jsou mapovány na příslušné CRUD[5] operace. Jedná se také o platformně nezávislé řešení.

Formát komunikace je nejčastěji JSON, XML, popřípadě plain text (obyčejný text).
Oproti SOAP je REST orientován datově, nikoliv procedurálně. Na rozdíl od GraphQL je zde pro každý endpoint vyžadována odpovídající URL.

Ukázku jednoduché komunikace s JSON formátováním můžete vidět níže v ukázce 3.5 a 3.6 :

Listing 3.5: REST dotaz

```
1 GET https://rest-api.test.com/places/1/employees/32
```

[5] Čtyři základní datové operace: Create, Read, Update, Delete.

Listing 3.6: REST odpověď

```
{
  "employee": {
    "name": "John Doe",
    "height": 1.72,
    "age": 33
  }
}
```

Zvolené řešení

Pro komunikaci mezi serverovou a klientskou stranou byl zvolen přístup REST API s JSON formátováním, který nám přináší větší volnost při tvorbě klientské části. JSON formátování především pro svoji jednoduchost a úspornost.

3.4.4 Klientská část

Pro realizaci uživatelského webového rozhraní je nejdříve nutné zvolit technologii, kterou se frontend aplikace bude vyvíjet. V případě webové aplikace flashcards se bude jednat o některý z dostupných webových frameworků, konkrétně si shrneme několik nejpoužívanějších JavaScript (JS) knihoven, frameworků či celých jazyků, které se budou nejvíce hodit pro vývoj Uživatelské rohraní (UI).

React.js

React.js je open-source framework pro vytváření uživatelských rozhraní pro webové služby vyvíjený společností Meta (dříve Facebook) [39].

Základním stavební kámen zde tvoří tzv. komponenty (components), které představují různé, znovupoužitelné HTML elementy se zapouzdřenou funkcionalitou, jejichž skládáním vzniká komplexní UI aplikace. Tyto komponenty pak mají své vlastnosti (props) a spravují svůj vnitřní stav (state). Tento deklarativní způsob práce s daty aplikace vede k více předvídatelnému chování i lehčímu ladění. Pro renderování nevyužívá jen klasické DOM rozhraní, ale vytváří tzv. virtuální DOM, díky které umožňuje vysoký výkon a okamžité reakce na požadavky uživatele.

Klady

- Podle výzkumů [40] se může pyšnit vřelým přijetím mezi vývojáři.
- Představuje skvělou volbu pro frontend aplikace v JavaScriptu.
- Schopnost bezproblémové integrace s jinými frameworky poskytuje velkou výhodu pro ty, kteří chtějí ve svém kódu flexibilitu.

- Komponentový model, který kód zpřehledňuje a umožňuje upravovat pouze vybrané části stránky.
- Vysoký výkon díky virtuální DOM.
- Nepřeberné množství pluginů a závislostí usnadňující vývoj.

Zápory

- Netypovanost jazyka JavaScript (lze vyřešit kontrolou PropTypes či změnou jazyka na TypeScript).

Vue.js

Podle oficiálních stránek [41] je Vue.js progresivní framework pro tvorbu uživatelských rozhraní. Jedná se o Open source software, který může být do projektu začleněn buď jako běžná JS knihovna nebo jako webový aplikační framework. Zároveň jde o nejmladší z uvedených technologií pro Front-end (FE). Nejčastější využití nachází u tvorby Single-Page aplikací za pomoci vytváření znovupoužitelných komponent a dalších knihoven [42].

Klady

- Vestavěné podmíněné renderování či cykly přímo v atributech pomocí atributů v-xxx.
- Komponentový model, který kód zpřehledňuje a umožňuje upravovat pouze vybrané části stránky.
- Vysoký výkon díky virtuální DOM obdobně jako React.js.
- Podobná syntax jako má React.js či Angular.

Zápory

- Netypovanost jazyka JavaScript

Angular

Angular je nejstarším z výše uvedených frameworků. Má však stále dobrou podporu.

Klady

- Podpora frameworku.
- Dokumentace

Zápory

- Nutnost použití jazyka TypeScript.

Zvolené řešení

Pro klientskou část aplikace byl zvolen framework React.js pro jeho popularitu mezi programátory, široké palety doplňujících balíčků. Autor navíc o frameworku již má povědomí. S ohledem na předchozí zkušenosti bude kód v React.js psán v jazyce JavaScript.

3.4.5 Server hosting

V této podkapitole je popsána analýza dostupných řešení pro nasazení aplikace na hosting.
Výběr vhodné hostingové služby pro nasazení byl proveden na základě předchozí analýzy.

K dispozici existuje celá řada řešení, které usnadňují celý proces nasazení do produkčního prostředí. Jsou jimi například Heroku, Netlify, OpenShift a další [43].

Při výběru byly ale brány v potaz pouze ty, které umožňují nasazení aplikace zdarma.

Heroku

Heroku je PaaS[6] služba společnosti Salesforce, která umožňuje nasazení a spouštění Full Stack[7] aplikací [45].

Podporuje jazyky Java, PHP, Python, Ruby, Scala, Go a platformu Node.js. Jako primární datové úložiště nabízí systémy PostgreSQL a Redis.

Výhody:

- Možnost nasadit backend i frontend.
- Integrace s GitHub.
- Skvělá oficiální dokumentace.
- Kontejnerová orchestrace.
- Přívětivé i pro začátečníky.
- Snadné kolaborace s více lidmi.
- Snadné a rychlé nasazení.

Nevýhody:

- Aplikační server se při nečinnosti 30-ti minut automaticky vypne. Zapnutí nějakou chvíli trvá.
- V rámci bezplatného používání nelze přidat vlastní doménu.
- Nižší výkon v bezplatné verzi.

OpenShift

OpenShift je PaaS služba společnosti Red Hat, která umožňuje nasazení a spouštění Full Stack aplikací [46].

Podporuje jazyky Java, PHP, Python, Perl, Ruby a platformy Node.js, .NET Core.
Jako datové úložiště nabízí systémy MySQL, PostgreSQL a Redis, MariaDB a MongoDB.

[6] Platforma jako služba (platform as a service)[44].
[7] Pokrývající jak serverovou stranu, tak i klientskou stranu aplikace.

Výhody:

- Možnost nasadit backend i frontend.
- Integrace s GitHub.

Nevýhody:

- Aplikační server se při nečinnosti 30-ti minut automaticky vypne. Zapnutí nějakou chvíli trvá.
- Umožňuje pouze 3 aplikace v rámci jednoho účtu.

Netlify

Netlify je platforma od stejnojmenné společnosti Netlify, která umožňuje vývojářům vytvářet, nasazovat a testovat statické webové stránky s minimální nutnou konfigurací [47].

Podporuje frameworky React.js, Next.js, Vue.js, Angular, Svelte, WordPress a další.

Výhody:

- Dovoluje zobrazení aplikace na vlastní doméně.
- Integrace s GitHub, BitBucket a další.
- Poskytuje Analytiku nasazených webových stránek.
- SSL podpora zdarma.
- Náhled před nasazením.
- Snadné nasazování.
- Drag & Drop funkce.
- Serverless funkce - Node.js, Go.

Nevýhody:

- Neumožňuje nasazení standardní serverové části.
- Limituje šířku pásma a další parametry.

Vercel

Vercel, dříve známý jako ZEIT, je platforma společnosti Vercel, která umožňuje vývojářům vytvářet, nasazovat a testovat statické webové stránky s minimální nutnou konfigurací [48].

Podporuje frameworky React.js, Next.js, Angular, Gatsby a další.

Výhody:

- Snadné nasazení
- Integrace s GitHub, GitLab a další.
- Podpora nasazení skrze git příkazy.
- SSL podpora zdarma.
- Dovoluje zobrazení aplikace na vlastní doméně.
- Serverless funkce - Node.js, Go, Python, Ruby.

Nevýhody:

- Neumožňuje využití databáze či cloudového úložiště.
- Limituje šířku pásma a další parametry.

Vlastní server

Možností je i zprovoznit vlastní platformu či hostující server. To se ale potýká s daleko více negativy než pozitivy.

Podpora libovolných frameworků a jazyků dle instalace.

Výhody:

- Podpora libovolných technologií dle instalace.
- Žádná přímá omezení platformy.
- Při již existujícím serveru velmi jednoduché nasazení
- Není potřeba registrace

Nevýhody:

- Vhodné pouze pro více pokročilé.
- Limitace způsobená Hardwarem.
- Nutnost vlastnit server.
- Potřeba složitější konfigurace.
- Nutnost veřejné IP adresy.
- Spotřeba elektrické energie.

Zvolené řešení

Kromě vyjmenovaných řešení existuje i celá další řada platforem jako například AWS - Amazon Web Service, Firebase a podobně. Ale povětšinou se jedná o placené služby či profesionální nástroje nevhodné pro začátečníky.

Dle předchozí analýzy bylo rozhodnuto použít pro nasazení serverové části aplikace platformu Heroku, kde pro potřeby navrhované aplikace postačí bezplatný plán. Klientská část aplikace bude nasazena na vlastní hosting.

3.5 Shrnutí kapitoly

V této kapitole byla popsána provedená analýza již existujících řešení, ze které vychází požadavky a případy užití na navrhovanou aplikaci. Následovala analýza technologií, na jejímž základě byly vybrány nejvhodnější technologie pro aplikaci Flashcards.

Pro serverovou část byla vybrána kombinace Java a Spring Boot. Pro klientskou část React.js a JavaScript. Pro uchování dat a jejich manipulaci byla zvolena relační databáze PostgreSQL. Pro nasazení byl vybrán jako vhodný hostovací server platforma Heroku.

Kapitola 4

Návrh

Cílem této kapitoly je navrhnout architekturu aplikace včetně uživatelské rozhraní. Poté popsat komunikaci mezi frontendovou a backedovou částí a na závěr ukázat nejdůležitější diagramy průchodů a užití aplikace. Tento návrh je také ovlivněn funkcionalitami existujících řešení rozbírané v předešlé kapitole (viz. chapter 3).

4.1 Architektura

Architektura navržené aplikace je postavena na principu **klient-server**. Komunikace mezi serverem a klientem je uskutečňována pomocí **RESTful** rozhraní skrze dotazy HTTP(s).

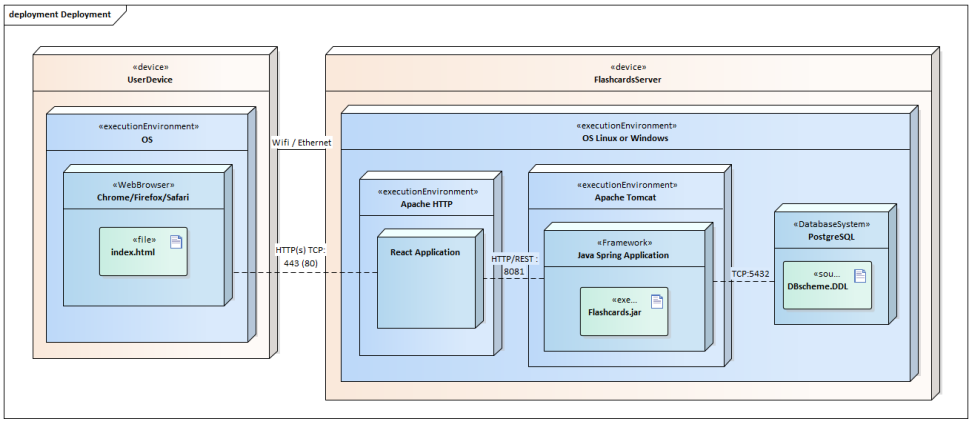
Formát pro přenášená data byl zvolen **JSON** pro jeho platformovou nezávislost a úspornost díky struktuře klíč-hodnota [49].

4.2 Fyzická Architektura

Pro aplikaci Flashcards byla zvolena monolitická architektura.

Monolitická architektura bývá upřednostňována pří vývoji menších, jednodušších a "lehčích"aplikací. Považuje se za tradiční přístup k architektuře aplikace.
Opakem je architektura Microservices, kde se jednotlivé služby (services) nasazují samostatně a nezávisle na sobě [50].
U monolitické architektury se aplikace vyvíjí jako jediný balíček, přestože je vnitřní struktura a logika často rozdělena do jednotlivých vrstev.

V případě aplikace Flashcards je tento monolit rozdělen na 2 balíčky: frontend a backend. Nasazení na server lze vidět na obrázku 4.1.



Obrázek 4.1: Diagram nasazení

4.3 Serverová část

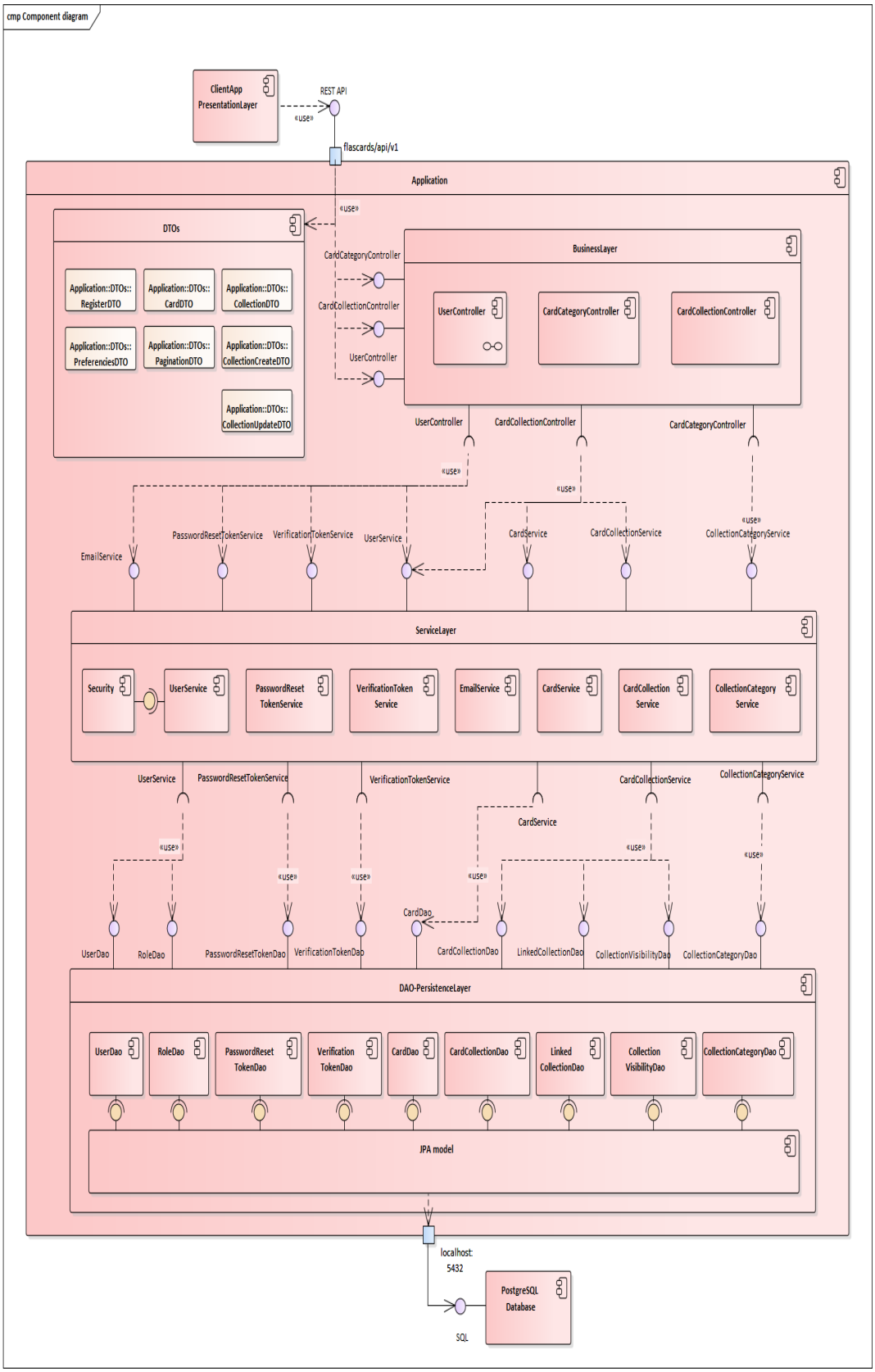
Backendem je myšlena část aplikace umístěná na serveru, s kterou komunikuje klientská část.

Na rozdíl od ní je se tato část stará o hlavní logiku aplikace a úkony, které normální uživatel nevidí. Zpracovává dotazy, data, obstarává spojení s databázovým systémem a předává data klientské části.
Obecně se tedy lze říci, že backend se stará o chod aplikace, ale uživatel s ním přímo do styku nepřijde.

Implementačním jazykem bude Java. Jako aplikační server byl na základě předešlé analýzy a zadání vybrán Apache Tomcat, díky jeho obsažení přímo ve frameworku Java Spring Boot.

Serverová část bude vycházet z třívrstvé architektury, tj. vrstva prezentační, aplikační a persistentní. Dále můžeme počítat databázi jako vrstvu čtvrtou.

Jednotlivé komponenty a vrstvy serverové části lze vidět na obrázku 4.2.

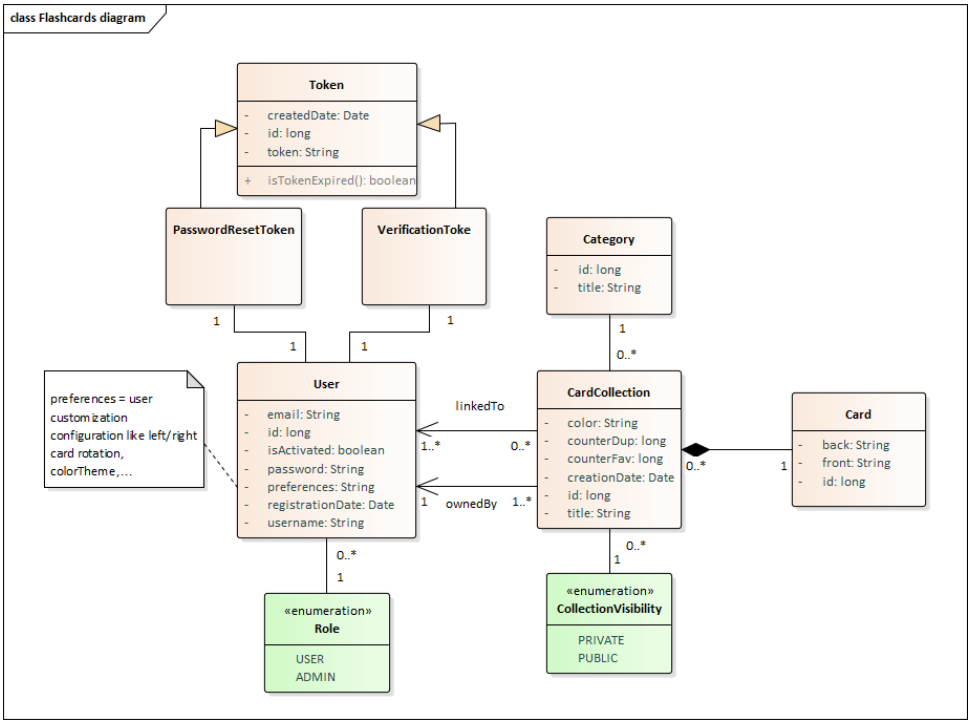


Obrázek 4.2: Diagram komponent

4.3.1 Doménový model

Doménový model pomáhá definovat entity, jejich atributy a vztahy mezi nimi. Popisuje tak data, se kterými v rámci projektu budeme pracovat.

Na základě obecné definice karet flashcards, zadání práce a dříve vydefinovaných požadavků, které by měla navrhovaná aplikace splňovat, výsledný diagram tříd by mohl vypadat jako na obrázku 4.3.



Obrázek 4.3: Class diagram aplikace

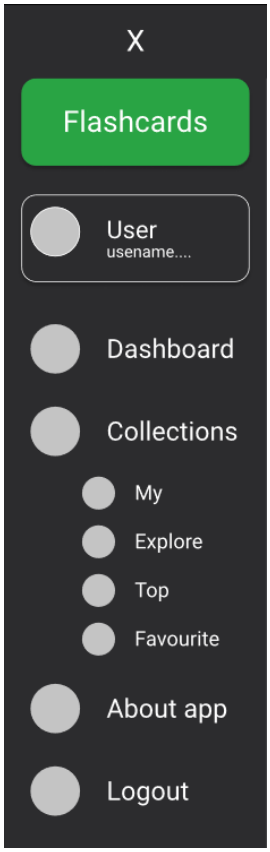
4.4 Klientská část

Klientská část neboli FE je část aplikace, která komunikuje se serverovou částí a uživatelem pomocí UI. V případě webové aplikace flashcards se bude jednat o framework React.JS, který byl vybrán během analýzy technologií klientské části.

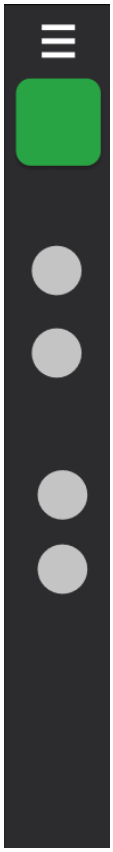
4.4.1 Návrh Uživatelského rozhraní

Při navrhování klientské části se zaměříme především na dříve vydefinované požadavky a případy užití viz. section 3.2. Pro tyto potřeby byl využit online nástroj Figma [51].

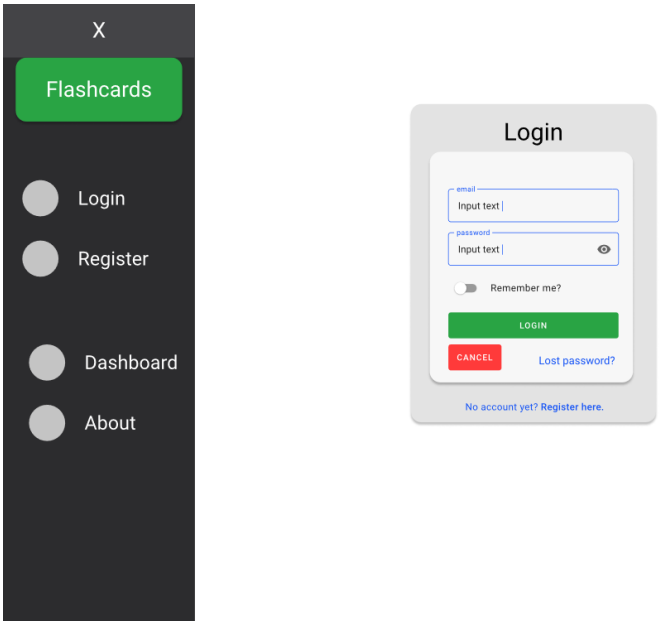
Obrazovky UI



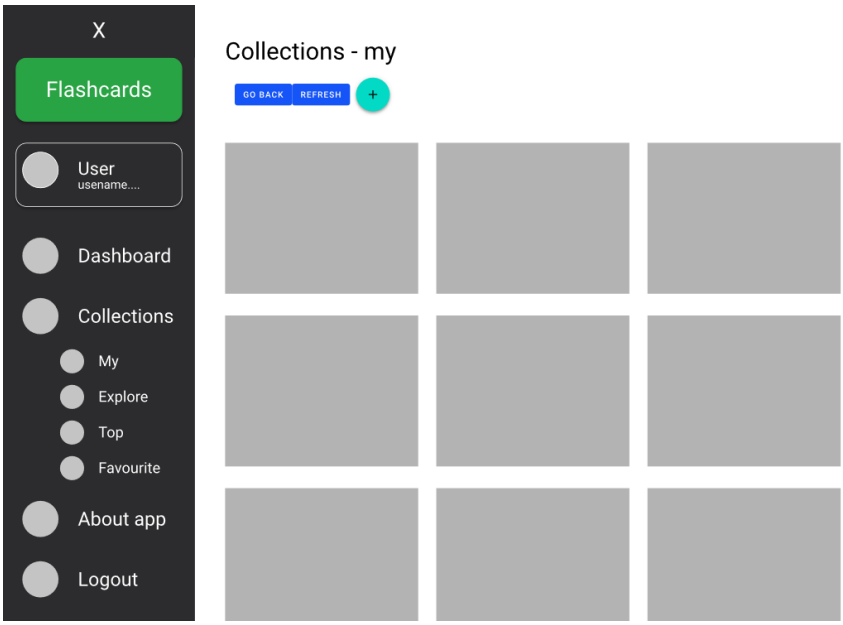
Obrázek 4.4: Návrh UI - Navigační panel - celý



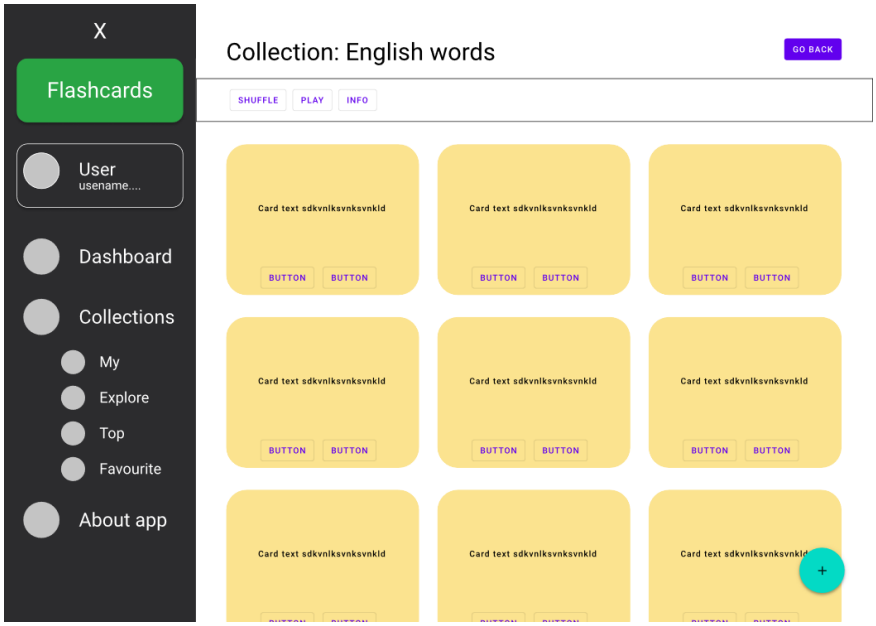
Obrázek 4.5: Návrh UI - Navigační panel - minimalizovaný



Obrázek 4.6: Návrh UI - přihlašovací stránka



Obrázek 4.7: Návrh UI - seznam kolekcí



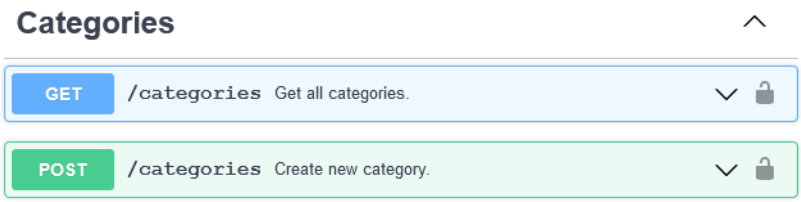
Obrázek 4.8: Návrh UI - seznam karet v kolekci

4.5 Komunikace REST API

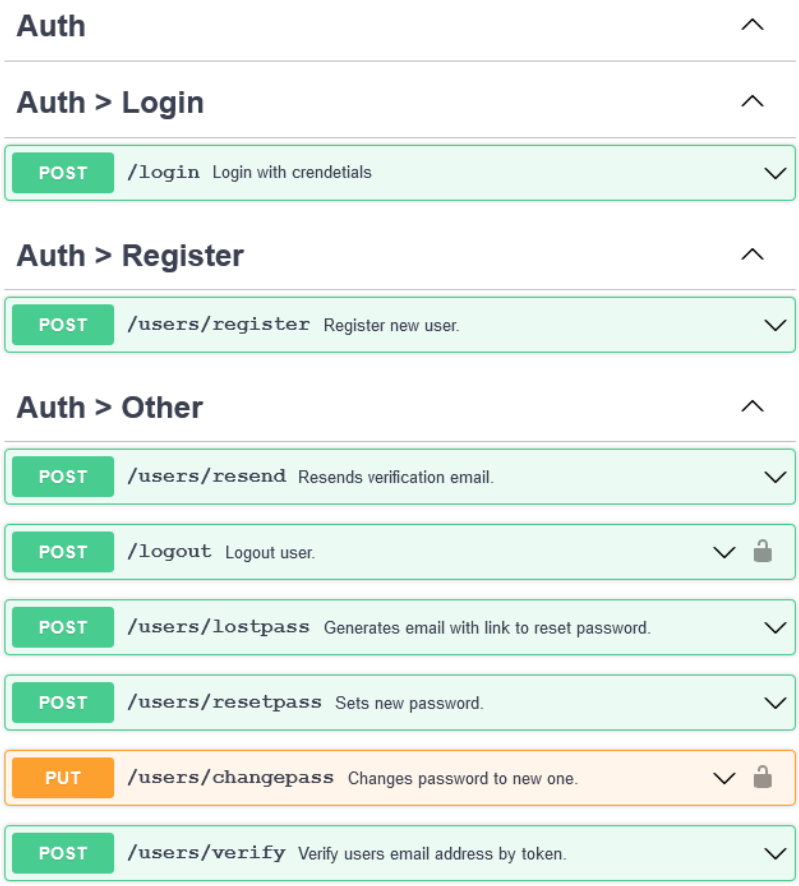
Rest API a komunikace s ním byla popsána pomocí specifikace OpenAPI 3.0.0.

Specifikace OpenAPI, dříve známá jako Specifikace Swagger, představuje nástroj pro strojově čitelný popis, vizualizaci a testování webových API [52]. Také umožňuje automatické generování části kódu API pro serverovou či klientskou stranu na základě předem vytvořené specifikace ve formátu JSON nebo YAML.

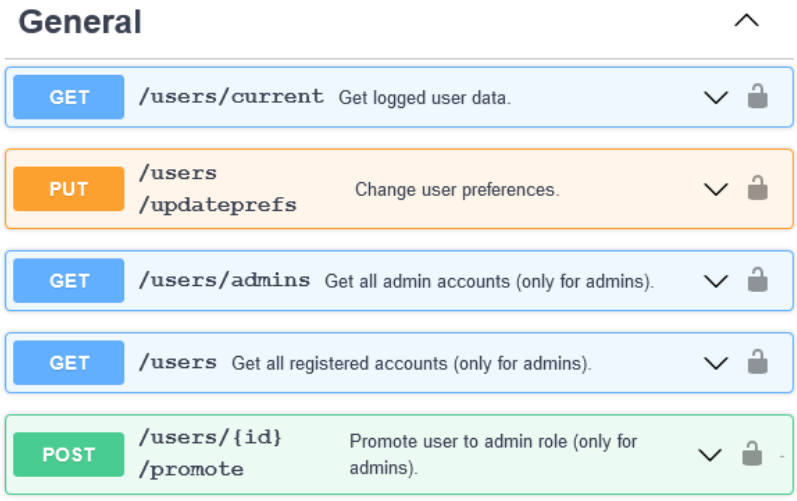
Tato dokumentace se nachází ve zdrojovém kódu serverové části.
Cesta k souboru: $/src/main/resources/flashcards_beopenapi3_spec.yaml$ Ve stejné složce se nachází i generovaná specifikace OpenAPI, popřípadě po spuštění backendu je k dispozici na adrese v příloze Seznam Odkazů Ukázka vizualizace dokumentace:



Obrázek 4.9: OpenAPI - Kategorie



Obrázek 4.10: OpenAPI - Autentizace a uživatelské funkce



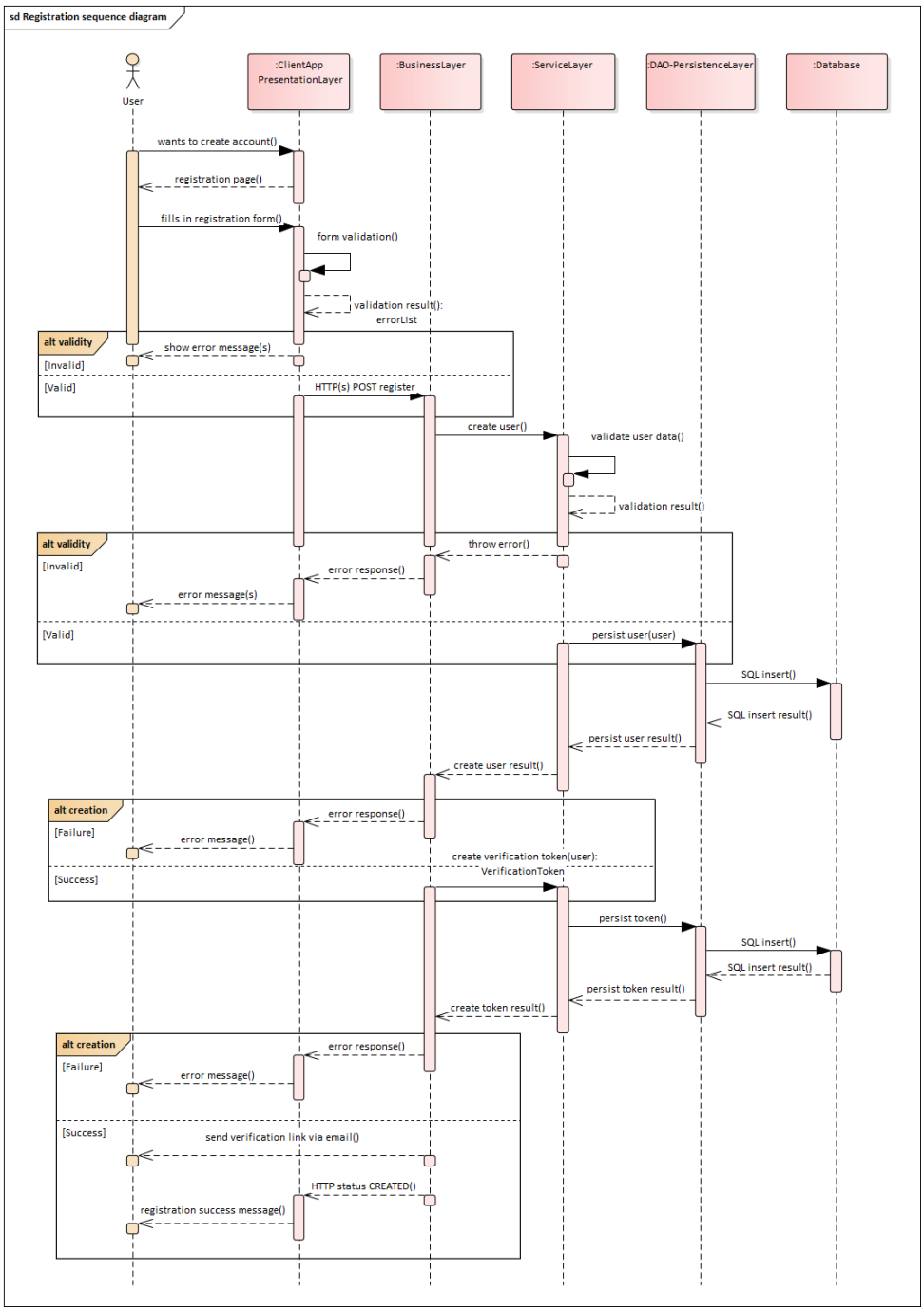
Obrázek 4.11: OpenAPI - Další

Obrázek 4.12: OpenAPI - Kolekce

4.6 Sekvenční diagramy

Sekvenční diagramy byly vytvořeny v nástroji Enterprise Architect 14 [53] a popisují některé důležité mechanismy, jak by aplikace měla fungovat při používání.

4.6.1 Registrace

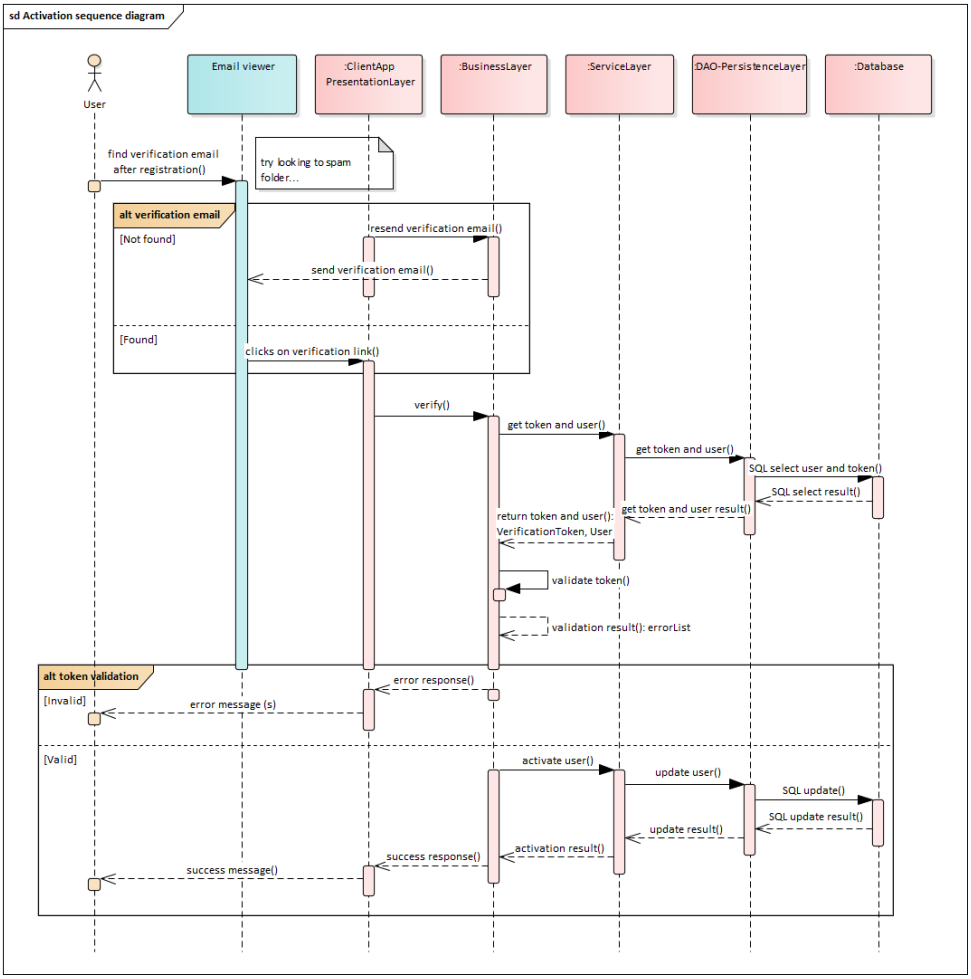


Obrázek 4.13: Sekvenční diagram registrace

4.6.2 Ověření účtu

Po registraci je nutné účet, respektive emailovou adresu ověřit. To se děje zasláním ověřovacího emailu na registrovanou adresu, ve kterém se nahází

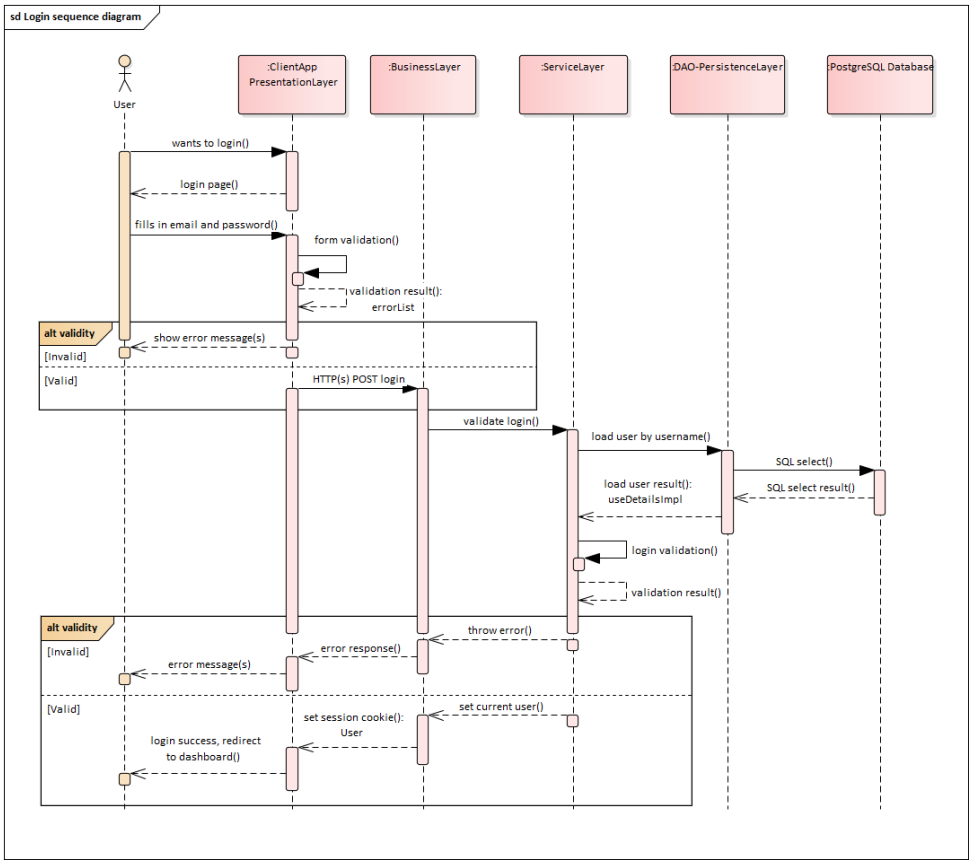
odkaz s unikátním tokenem. Po otevření odkazu je token validován a pokud byl odkaz platný, účet je ověřen a uživateli se zobrazí hlášení o úspěšném ověření a nabídka k přesměrování na přihlašovací formulář.
Tento proces popisuje obrázek 4.14



Obrázek 4.14: Sekvenční diagram aktivace účtu

4.6.3 Přihlášení

Po úspěšném ověření registrovaného účtu je možné se přihlásit. Pro přihlášení je nutné vyplnit přihlašovací údaje (uživatelské jméno a heslo). Pokud uživatel zadá správné údaje, serverová část nastaví session cookie, která drží stav relace a vrátí informace o úspěšném přihlášení společně se základními údaji o uživateli. Klientská strana si tyto údaje zpracuje a uživatele přesměruje na stránku Dashboard.
Tento proces popisuje obrázek 4.15



Obrázek 4.15: Sekvenční diagram přihlášení

4.7 Shrnutí kapitoly

V této kapitole byla navržena architektura a uživatelské rozhraní aplikace. Dále se definovala komunikace mezi frontendem a backendem. Nakonec byly graficky znázorněny sekvence některých důležitých funkčních bloků aplikace.

Kapitola 5

Implementace

Kapitola implementace popisuje použité prostředí, nástroje a technologie v průběhu vývoje a samotný postup při psaní zdrojového kódu serverové a klientské části aplikace.

5.1 Vývojové prostředí a použité nástroje

Pro potřeby vývojových prostředí nebyla žádná analýza provedena. S veškerými potřebnými nástroji se autor seznámil během studia či vlastní praxe.

5.1.1 JetBrains IntelliJ IDEA

IntelliJ IDEA je komerční vývojové prostředí (IDE) pro programování v jazycích Java, Kotlin, Groovy a dalších [54]. Také nabízí vestavěné nástroje pro ladění, zvýraznění syntaxe, kontextový našeptávač, podporu Git, refactorizaci, testování, atd. Podporuje i velké množství rozličných rozšíření.

Tímto nástrojem byla implementována serverová strana aplikace.
Použitá verze: 2021.2.3 Použitá verze Java JDK: 17

5.1.2 Sparx Enterprise Architect

Enterprise Architect od společnosti Sparx Systems je kompletní CASE nástroj pro systémovou analýzu a návrh, který kompletně pokrývá cyklus vývoje systému, tj. od zadání požadavků přes analýzu stavu, návrhu modelů, testování a údržbu, to vše s využitím diagramů v UML [53].

Tímto nástrojem byly kresleny veškeré vlastní diagramy.
Použitá verze: 14

5.1.3 VS Code

Visual Studio Code je bezplatný editor zdrojového kódu vyvíjený společností Microsoft pro operační systémy Windows, Linux a macOS. Obsahuje podporu pro Git (a pro GitHub), zvýraznění syntaxe, kontextový našeptávač a podporu pro ladění a refaktorizaci [55]. V rámci široké palety rozšíření lze editor obohatit o podporu dalších jazyků či nástrojů.

Tímto nástrojem byla implementována klientská strana aplikace.
Použitá verze: 1.65.1

5.1.4 Postman

Postman je multiplatformní aplikace, která zvládá širokou škálu kroků od návrhu, přes testování, až po monitoring HTTP API [56].

Tímto nástrojem bylo navrženo a testováno API serverové části aplikace, kdy ještě nebyla implementována klientská část.
Použitá verze: 9.14.13

5.1.5 Mozilla Firefox

Browser Firefox je populární bezplatný internetový prohlížeč firmy Mozilla [57].

Tento nástroj byl použit jako hlavní prohlížeč při pozorování, debuggování a testování klientské strany aplikace.
Použitá verze: 98.0.1

5.1.6 PostgreSQL pgAdmin4

PgAdmin je nejpopulárnější open source software pro správu databází na platformě PostgreSQL [58].

Tento nástroj byl použit pro správu databáze aplikace.
Použitá verze: 4

5.1.7 Figma

Figma je cloud-based kolaborační design nástroj [51]. Jedná se o webový vektorový grafický editor a prototypovací nástroj s možností práce offline v desktopových aplikacích pro Windows či iOS [59]. Funkcionalita se zaměřuje hlavně na prototypování UI a design. Aplikace funguje v reálném čase s důrazem na možnost kolaborace více lidí.

Tímto nástrojem byly navrženy hlavní obrazovky aplikace.
Použitá verze: online

5.2 Buildovací nástroj

Mezi nejpopulárnější nástroje pro správu, řízení a automatizaci sestavení aplikace se řadí **Maven** A **Gradle**. Přestože je Gradle má podle statistik rychlostí nárůst v sestavování programu o 37-98% [60] a kratší syntaxi, z osobních zkušeností autor preferuje nástroj Maven.

Proto byl pro tento projekt zvolen nástroj Maven.

5.3 Verzování kódu

Pro verzování kódu byl použit nástroj GitHub, který je open-source službou pro správu kódu nástrojem GIT.

GIT je distribuovaný systém správy verzí, jehož autorem je Lius Torvalds. Výhodou distribuovaného systému oproti centralizovanému je, že každý uživatel si drží vlastní kopii repozitáře nebo jeho části lokálně a následně se sledují změny. Díky tomu lze spolupracovat na jednom souboru současně s dalšími lidmi. Git také zachovává i předchozí verze, ke kterým se lze kdykoliv vrátit, což činí vývoj pohodlnější a bezpečnější [61].

Pro vývoj celé aplikace byly založeny 2 Git repozitáře: flashcards-backend pro serverovou část. flashcards-frontend pro klientskou část.

Odkazy na repozitáře se nacházejí v příloze Seznam Odkazů

5.4 Normy a standardy

Při implementaci aplikace se budou dodržovat následující normy a standardy:

- EsLint doporučení pro frontend dle pluginů:
  - plugin:react/recommended
  - plugin:jest/recommended
  - plugin:react-hooks/recommended
  - eslint:recommended
- Forma ukládání dat do databáze je normalizovaná
- Rest dotazy budou odpovídat správným HTTP metodám [62]

- Zprávy commitů ve verzovacím nástroji se píší anglicky stručně a srozumitelně
- Hesla budou bezpečně uchována v hashované formě se solí pomocí algoritmu BCrypt

5.5 Zabezpečení

Mezi nejpoužívanější způsoby zabezpečení webových aplikací patří OAuth, JWT token a HTTP Basic.

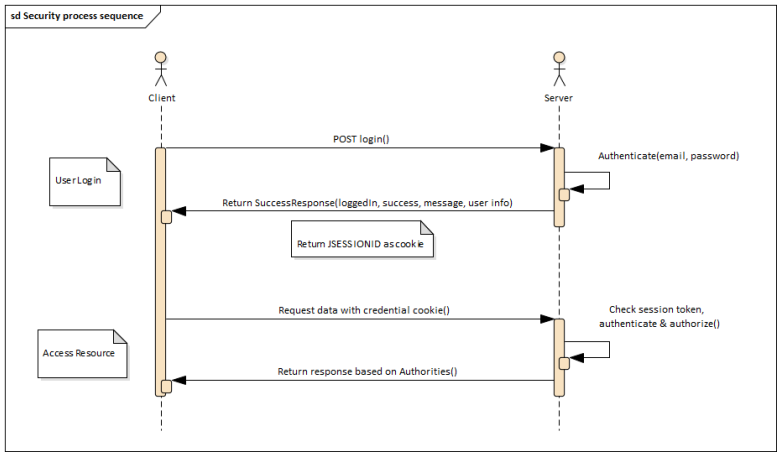
Pro autentizaci a autorizaci v rámci aplikace bylo rozhodnuto využít způsob OAuth.

5.5.1 OAuth

OAuth je protokol, navržený pro bezpečnou API autentizaci. Využívá posílání přístupových tokenů od bezpečnostních autorit spolu s dotazem namísto posílání přihlašovacích údajů. Jeho specifikace popisuje podrobně fungování oné výše popsané výměny tokenů a klíčů, ale nijak nedefinuje vlastní komunikaci s webovou aplikací ani není vázán na konkrétní typ API (pouze je omezeno na HTTP protokol). OAuth lze tedy použít jako metodu ověření uživatele k jakémukoli typu API (SOAP, XML-RPC, REST atd.), a to nejen na webových aplikacích.

Pro autentizační a autorizační účely v rámci specifikace OAuth je využívána implementace Spring modulu **Spring Security**, jehož využití je snazší díky výborné integraci s frameworkem Spring Boot. To bylo hlavní výhodou oproti implementaci zabezpečení pomocí JWT.

Následující diagram 5.1 ukazuje postup, jak je implementován proces přihlášení a autorizace uživatele při dotazu serveru.



Obrázek 5.1: Proces přihlášení a autorizace uživatele

5.6 Použité knihovny

5.6.1 Backend

Hlavní závislosti

Spring Framework. Framework Spring, respektive jeho Java implementace byl již popsán v sekci Java Spring Boot.

Spring Security. Spring Security je Java/JavaEE framework, kter y poskytuje mechanismy pro...

Hibernate. Framework Hibernate je implementací Java Persistence API (JPA) v jazyce Java, která umožňuje tzv. objektově relační mapování. Usnadňuje tím zachování dat objektů i po ukončení aplikace.

Postgresql. Balíček org.postgresql je ovladač databáze platformy PostgreSQL, který je vyžadován pro připojení a správu této databáze.

Doplňující závislosti

Spring Boot Starter Test. Startovací balíček pro testování Spring Boot aplikací. Obsahuje testovací knihovny jako JUnit Jupiter, Mockito a další.

Lombok. Lombok je Java knihovna, sloužící k minimalizaci často používaného primitivního kódu pomocí anotací. Zvyšuje tak přehlednost a snižuje rozsah zdrojového kódu.

SpringDoc OpenAPI. Knihovna SpringDoc OpenApi pomáhá automatizovat generování dokumentace vystaveného API zkoumáním konfigurace, struktury tříd a anotací. Tuto dokumentaci dokáže automaticky generovat ve formátu JSON/YAML a následně ji zpřístupnit ve formátu HTML stránky.

5.6.2 Frontend

Pro klientskou část aplikace byl využit balíčkový systém dependencí **NPM**, který umožňuje kromě správy, spouštění a sestavení frontendu i snadné přidávání jeho závislostí.

Tyto podpůrné balíčky byly rozděleny do dvou kategorií podle důležitosti, nicméně se nejedná o všechny použité balíčky. Celý seznam spolu s verzemi je k dispozici ve zdrojovém kódu frontendu v souboru package.json pod atributem *dependencies*.

Hlavní závislosti

React.JS. Framework React.js byl již popsán v sekci 3.4.4. Balíček react a jeho dependence (react-dom, react-scripts) byly nainstalovány během vytváření projektu klienta NPM příkazem *npx create-react-app nazev-aplikace*, který vytvoří základní strukturu aplikace. Použitá verze

React Query. Knihovna React Query zajišťuje získávání, kešování, synchronizaci a aktualizování stavů v React aplikaci. Umožňuje automatické občerstvování dat, paralelní dotazy, podporuje konstrukty async/await i Promise, SSR, integraci s vlastním http klientem a mnoho dalšího [63].

Na rozdíl od knihovny React Redux [64] zde není potřeba vytvářet objekt store a asynchroní thunk metody, ale vše funguje s minimální konfigurací rovnou po instalaci balíčku.

Axios. Axios je spolehlivý HTTP klient pro komunikaci se serverem [65]. V projektu je použit integrován do React Query dotazů.

Material UI. Material UI je populární knihovnou pro tvorbu uživatelského rozhraní webových stránek využívající technologii React.js [66].

Balíček obsahuje předpřipravené komponenty ve stylu Material Designu od společnosti Google [67], který se využívá například v telefonech Android. Knihovnu tvoří balíčky: @mui/icons-material, @mui/material

React Redux. Redux je populární knihovna pro správu a centralizaci stavu aplikací [64].
Hlavními konstrukty reduxu jsou: **store**, **akce** a **reducer**, kdy store udržuje stav aplikca a je propagován napříč aplikací, akce je popis nějaké změny, kterou chceme provést, a reducer danou změnu bezpečně provede.

Nejčastěji se používá spolu s frameworky React nebo Angular.

Redux Persist. Redux Persist je rozšiřující balíček pro React Redux, umožňující trvalé uložení a následné obnovení stavu aplikace, respektuve redux storu [68].

Formik. Formik je flexibilní knihovna, která se stará o sledování hodnot, chybových hlášek, navštívených polí, validaci a zpracování webových formulářů. Lze používat s HTML vstupními poli, předpřipravenými či vlastními. Pro validaci polí lze kombinovat s knihovnou Yup [69].

V projektu se formik používá ve všech formulářích pro jeho organizovanost a užitečnost spolu s validací nástroje Yup.

Yup. Yup je javascriptový nástroj na tvorbu schémat pro analýzu a ověřování hodnot. Umožňuje definovat jednoduchá i složitá schémata, transformovat

hodnotu tak, aby odpovídala, ověřovat tvar existující hodnoty nebo vše najednou [70].

connected-react-router. Connected React Router umožňuje propojit klasický React Router pro směrování mezi stránkami v aplikaci a Redux store. To umožňuje přístup k hodnotám url či historii přímo přes propagovaný store v kterékoliv komponentě [71].

node-sass. Knihovna Node-sass poskytuje překlad pro Node.js na LibSass, oblíbený preprocesor stylů Sass (Syntactically awesome Style Sheets). Umožňuje nativně kompilovat .scss soubory do css a to i automaticky přes spojovací middleware [72].

Doplňující závislosti

react-toastify. React-Toastify je knihovna založení na knihovně React, která umožňuje snadné zobrazování notifikací v React aplikacích. Na výběr je z několika předdefinovaných pop-up notifikací, ale lze si vytvořit libovolný vlastní styl [73].

material-ui-confirm. Balíček Material-UI Confirm poskytuje jednoduché potvrzovací dialogové okno založené na Material-UI knohovně. Díky React hooks je použití jednoduché a účinné [74].

V projektu je knihovna využita především u destruktivních operací jako mazání, ke kterému by mohlo dojít neúmyslně.

react-error-boundary. Knihovna poskytující jednoduchý obal pro komponenty, který zachytává propagované chyby při renderování. Ty pak následně umožňuje zpracovávat a řešit [75].

react-pro-sidebar. React Pro Sidebar je přizpůsobitelný a responzivní postranní panel s rozevíracími nabídkami [76].

V projektu je využit jako obal postranní navigace pro jeho responzivní design.

react-spring. React Spring reprezentuje přístup k animacím, přechodům, interpolacím apod. skrze moderní react přístupy jako jsou dedikované komponenty nebo hooks. Poskytuje možnosti deklarativního i imperativního přístupu [77].

V projektu je react-spring využit k animaci otáčení karet v sekci učení/hry pro možnost resetování animace a snadného ovládání.

react-cookie-consent. Všem uživatelům, kteří využívají konkrétní službu ukládající soubory cookies do počítače, je nutné sdělit informaci, že k ukládání

dochází. Vyplývá to z evropského nařízení o nabízení internetových služeb přibyla všem webovým aplikacím nová povinnost [78].

Balíček react-cookie-consent přináší předpřipravenou komponentu vytvářející souhlasný banner se soubory cookies [79].

Knihovna tedy především ulehčuje implementaci této funkcionality a urychlila tím vývoj důležitých prvků aplikace.

5.7 Realizace Databáze

Realizace samotné databáze spočívalo pouze ve vytvoření nové, prázdné databáze, jejíž název a přístupové údaje se zanesly do konfiguračního souboru backendu nacházející se v *src/main/resources/application.properties*. Schéma databáze se vygeneruje automaticky po spuštění podle modelu díky JPA anotacím.

5.8 Realizace Serverové části

Jak již bylo zmíněno v kapitole Analýza, pro implementaci backendové strany byl zvolen framework Java Spring Boot. Serverem je program, zpracovávající HTTP dotazy přicházející od webového klienta. Architekturu aplikace Spring Boot lze obecně rozdělit na čtyři vrstvy následujícím způsobem: Prezentační vrstva, Aplikační vrstva, Persistentní vrstva a vedle stojící databáze. Detail architetury byl již představen v kapitole Návrh včetně diagramu komponent4.2.

5.8.1 Prezentační vrstva

Prezentační vrstva je vnější vrstvou, která zpracovává přicházející HTTP dotazy. Požadavky samotné však nezpracovává, pouze je deleguje na vrstvu aplikační a výsledná data poté odesílá klientovi. Tyto kontroléry se v projektové struktuře nachází ve složce *src/main/java/...rest*.

5.8.2 Aplikační vrstva

Prostřední částí je aplikační vrstva. Zde leží jádro aplikace, její logika a funkce, výpočty a zpracování dat. V projektové struktuře se nachází především v adresáři *src/main/java/...service*.

5.8.3 Persistentní vrstva

Persistentní vrstva zprostředkovává rozhraní pro komunikaci s databází. Ve zdrojovém kódu je reprezentuje balíček DAO (Data Access Object). Každý modul z tohoto balíčku je součástí návrhového vzoru Dao a dědí od generické abstraktní třídy BaseDao<Class>.

Tato třída rozhraní implementuje veškeré CRUD operace. V některých třídách základní operace nestačily, proto bylo potřeba vytvořit vlastní SQL dotazy skrze knihovnu CriteriaApi, která abstrahuje čisté SQL příkazy do jazyk Java.

5.8.4 Model

Modelem je myšlen balíček tříd, které odpovídají navrženému diagramu tříd, doplněné o JPA anotace, konstruktory a metody. V projektové struktuře se nachází ve složce *src/main/java/.../model*

5.8.5 Přenos dat

Pro přenos objektů mezi serverovou a klientskou stranou byly implementovány speciální třídyDTO[1], které vycházejí z modelových tříd.
Nemají však vnitřní logiku a slouží jen jako kontejner dotazu či odpovědi. Někdy není třeba posílat celý objekt, proto DTO nemusí obsahovat veškeré atributy jako jeho vzorová třída. Pak je ale nutné provést důkladné mapování na reálné instance objektů.

5.9 Realizace Klientské části

Klientská část neboli FE je část aplikace, která komunikuje se serverovou částí a uživatelem pomocí UI. V případě webové aplikace flashcards se bude jednat o framework React.js, který byl vybrán během analýzy technologií klientské části.

Frontend aplikaci v jazyce React.js tvoří soubor index.js kde se nachází všechny aplikační kontexty a volá se hlavní komponenta *App*.

Zde dochází k inicializaci všech potřebných zdrojů a rozpadu aplikace na hlavní komponenty uživatelského rozhraní - Svislá navigační lišta a samotný obsah. Obsah je následně spravován směrovačem, který podle URL adresy zobrazuje správnou komponentu stránky.

Všechny stránky a komponenty, kterými jsou tvořeny, jsou definovány moderní syntaxí - Functional components [80], která umožňuje využití tzv. **hooks**.

[1] Data Transfer Object

Aplikace je dělaná především pro desktopová zařízení, avšak lze ji využívat i na mobilních zařízení díky responsnímu designu.

5.9.1 Hooks

Hook je novinka pro React verze 16.8 a výše, která umožňuje využívat stav (state) a další react funkce bez nutnosti psaní tříd [81].

V aplikaci se vlastní hooks nachází ve složce *src/hooks*, popřípadě jsou tímto způsobem řešeny modální okna, která se nachází v příslušný adresářích podle obsahu.

5.9.2 Udržování dat

Udržování dat aplikace je zajišťováno dvěma způsoby: První je globálním *store* z balíčku Redux, který drží potřebné informace o přihlášeném uživateli. Druhou metodou je knihovna React Query, která po provedeném dotazu na server odpověď drží v interní cache paměti. Pokud nebyly data zneplatněna, při stejném dotazu kdekoliv v aplikaci vrátí knihovna uloženou odpověď a na server požadavek odeslán není.

5.9.3 Popis Aplikace

Aplikaci tvoří celkem 13 různých hlavních komponent / stránek.

Těmi jsou: AboutPage, LoginPage, RegisterPage, HomePage, DashboardPage, ProfilePage, AdminPage, LostPassPage, ResetPassPage, VerifyPage, CollectionsPage, CollectionDetailPage a PageNotFoundPage.

Hlavním rozcestníkem aplikace je postranní svislý panel, obsahující odkazy na jednotlivé stránky. Jeho obsah se mění podle typu uživatele. Pokud by menu příliš rušilo nebo se nehodilo pro malé obrazovky, lze ho plynule sbalit do kompaktní verze bez názvů.

Pro veřejnost, tedy nepřihlášené uživatele, jsou k dispozici pouze domovská stránka, stránka o projektu, přihlášení, registrace, ověření účtu, stránka zapomenutého hesla a resetování zapomenutého hesla.
Po přihlášení se uživateli postranní navigace rozšíří o nové položky a nepotřebné odkazy jako registrace a přihlášení se skryjí.

Standardní uživatel má k dispozici především profilovou stránku s informacemi o účtu a nastavením a pak podmenu kolekcí, kde má přístup k různým druhům kolekcí.

Uživatel typu Administrátor má navíc přístup do speciální sekce, kde může přidávat nové kategorie a administrátory.

Autentizace

Registrace a přihlášení jsou veřejně dostupné formou formulářů, které jsou implementovány pomocí knihovnou Formik pro správu zadaných hodnot a knihovnou Material UI, která zajišťuje stylizaci.

Po registraci je uživatel přesunut na stránku ověření emailové adresy, kde čeká na ověřovací email. Pokud mu žádný nepřišel, může si ho zde vyžádat znovu.

V zaslaném emailu se nachází speciální odkaz s tokenem. Po otevření odkazu je uživatel ověřen a je upozorněn oznámením že se nyní může přihlásit.

Uživatel, který se jednou přihlásí, bude po znovuotevření stránky stále přihlášen díky uloženým údajům v LocalStorage skrze React Redux a Redux Persist. Po dohlášení jsou tyto data odstraněna.

Pokud by uživatel zapomněl své heslo, má možnost si heslo resetovat. Stačí přejít na stránku ztraceného hesla a vyplnit email, na který je účet registrován. Následně se odešle na daný email odkaz s tokenem na formulář s nastavením hesla nového.

Kolekce

Menu kolekcí je přístupné pro libovolného přihlášeného uživatele. Nachází se v něm hned několik druhů kolekcí. Záložka "Moje"kolekce obsahuje sady přímo vlastněné přihlášeným účtem, v "Objevit"kolekce se nachází veškeré veřejně přístupné kolekce a lze v nich vyhledávat podle názvu. Pak tu je záložka "Top", kde jsou seřazené nejlepší, respektive nejoblíbenější kolekce a nakonec "Oblíbené", kde se nacházejí všechny veřejné sady označené jako oblíbené.

Na všech zmíněných stránkách je obsah stránkována a lze přejít k vytváření vlastní nové kolekce pomocí modálního okna, které nabízí veškerá potřebná vstupní pole.

Další akce spojené s kolekcemi karet se nachází u každé zvlášť. Jedná se o funkce dle diagramu případů užití, respektive vstup do kolekce, Přidat / odebrat z oblíbených, úprava kolekce, smazání a vytvoření kopie. Akce jsou ale závislé na druhu sady.

Úprava vlastněných kolekcí probíhá přes stejné modální okno jako její vytváření, s tím rozdílem, že stávající atributy jsou již ve formuláři vyplněna.

Samotné kolekce jsou zobrazovány jako karty umístěné v mřížce. Jejich barvu lze definovat při vytváření. Kromě akčních tlačítek se zde nacházejí atributy jako název, viditelnost, kategorie, počet karet v kolekci, počet uživatelů, jenž si kolekci přidali do oblíbených, a počet, kolikrát byl vytvořen duplikát.

Karty

Sekce karet je přístupná libovolnému přihlášenému uživateli po vstupu do kolekce. Musí se však jednat o jím vytvořenou nebo veřejnou kolekci. Obdobně jako u kolekcí lze karty vytvářet formulářem uvnitř modálního okna. Ty se následně zobrazí v mřížkovém rozložení níže.

Na přední straně je vidět text otázky nebo pojmu a akční tlačítka pro smazání a úpravu karty modálním oknem. Na zadní straně je odpověď či definice. Mezi stranami karty lze přepínat kliknutím na její tělo, které je doprovázeno animací otočení.

Mezi další funkce na stránce patří promíchání karet, zobrazení informací o kolekci, v níž se nacházíme a možnost přepnutí do herního režimu.

Již tato strana lze využít pro učení, další možností je již zmíněný herní režim.

Herní režim

Herní, respektive výukový režim umožňuje procházet karty v kolekci jednotlivě stejným způsobem jako v přehledu kolekce - otáčením. Maximalizuje se tím soustředění na danou otázku a odpověď. Důležitými tlačítky jsou zde Další karta, Předchozí karta a promíchat karty.

5.10 Shrnutí kapitoly

Tato kapitola popisovala, které nástroje a prostředí byly v rámci implementace použity. Přiblížila způsob zabezpečení aplikace a byly popsány hlavní použité závislosti kódu. Nakonec byla popsána realizace částí projektu a samotná aplikace.

5.10.1 Nedokončené funkce a části

V rámci vývoje nebyly dokončeny následující funkce:

- Obsah domovské stránky a design Dashboardu.
- Filtrování kolekcí podle kategorií.

Kapitola 6

Nasazení

Tato kapitola obsahuje proces nasazení serverové a klientské části aplikace do produkčního, respektive testovacího prostředí včetně postupu nasazení a nutných změn v kódu.

6.1 Postup nasazení

Pokyny pro nasazení aplikace do produkčního prostředí vybraného v sekci Server Hosting byly rozděleny do dvou částí - postup pro serverovou a postup pro klientskou stranu.

6.1.1 Serverová strana

Pro nasazení serverové strany je nejprve potřeba program zkompilovat. Sestavení do spustitelného souboru se provede spuštěním životního cyklu nástroje Maven package, který provede jednotkové testy a následně provede kompilaci do balíčku jar.

Protože kompilovaný soubor backendové aplikace již obsahuje Apache Tomcat, lze pro testovací účely aplikaci spustit příkazem `java -jar flashcardsBackend.jar`.

Pro nasazení byl vytvořen bezplatný účet na platformě Heroku, který postačí pro účely aplikace.
Dále byl stažen program HerokuCLI, který umožňuje tvorbu a správu Heroku aplikací přímo z terminálu [82].

Pro potřeby nasazení na platformu Heroku byly provedeny následující úpravy kódu:

Úpravy:

- Změna cookie SamSite Policy ze *strict* na *none* kvůli rozdílné doméně frontendu a backendu.
- Separace citlivých nebo důležitých konfiguračních hodnot backendu do souboru, se kterým je spouštěno prostředí Apache Tomcat. Ten tyto hodnoty následně předává Spring aplikaci.
- Změna nastavení portu mail účtu.

Postup nasazení byl převzat z doporučeného postupu nasazení Java Spring Boot aplikace pro Heroku z oficiálních stránek [82].

Kroky pro nasazení backendu na platformu Heroku:

- Založení účtu na Heroku.
- Vytvoření projektu na Heroku.
- Databáze - Přidání pluginu Heroku Postgres jako databázové úložiště.
- Přihlášení k heroku v HerokuCLI
- Propojení s Git repositářem.
- Konfigurace nasazení.

  Nastavení buildpacku na heroku/jvm

  Instalace pluginu java

  Vytvoření konfiguračního souboru Procfile

  Obsah: `web: java -Dserver.port=\$PORT \$JAVA_OPTS -jar flashcards_backend-1.0.0.jar`

  Vytvoření konfiguračního souboru system.properties s verzí použité Javy

  Obsah: `java.runtime.version=17`

  Vytvoření konfiguračního souboru application.properties

  Obsahuje přístupové údaje k databázi a emailovému účtu.
- Samotné nasazení.
- Otevření stránky a kontrola nasazení.

  Stav (logy) aplikace lze sledovat příkazem `heroku logs --tail`

Po nasazení je serverová část aplikace dostupná na adrese v příloze Seznam Odkazů

6.1.2 Klientská strana

Pro nasazení klientské strany je nejprve potřeba webovou aplikaci zkompilovat do produkční verze. Provedeme to skrze nástroj NPM příkazem *npm run build*, který sestaví produkční verzi do složky *./build*.

Pro nasazení je pak obsah této složky nutný nahrát na požadovaný hostovací server. Vzhledem k použití statické architektury a směrování v aplikaci, není potřeba klienta "servírovat"skrze server.

Kroky pro nasazení frontendu na osobní server:

- Připojení pomocí FTP ke vzdálenému serveru.
- Vytvoření adresáře flashcards
- Přesunutí buildu do adresáře.
- Otevření stránky a kontrola nasazení.

Obsah adresáře zkompilované react aplikace byl nahrán na osobní hosting do adresáře */flashcards*, který odpovídá definované cestě v konfiguračním souboru.
Následně je klientská část aplikace dostupná na adrese v příloze Seznam Odkazů

6.2 Shrnutí kapitoly

V této kapitole byl popsán způsob nasazení serverové a klientské části aplikace včetně postupu.
Kvůli zvoleným platformám byly pro správnou funkčnost nutné i drobné změny v kódu či konfiguraci.

Výsledkem je nasazená funkční aplikace, jejíž serverová část leží na platformě Heroku a klientská část, která je hostována na osobním webovém serveru.

Kapitola 7

Testování

Tato kapitola se zabývá testováním aplikace Flashcards. Hlavní náplní je uživatelské testování doplněné o popis programátorského testování.

7.1 Programátorské testování

Aplikace byla testována manuálním prováděním logických scénářů podle případů užití. Pro některé části serverové části byly napsány i jednotkové testy, které se prováděli při každé kompilaci. Z tohoto testování vyplynulo několik chyb, které bylo třeba opravit.

Integrační testy byly prováděny během vývoje ručně nástrojem Postman [56], který umožňuje posílat libovolné HTTP dotazy a zobrazit jejich odpovědi.

Dále byla aplikace testována, jak se zobrazuje v populárních webových prohlížečích. Vyskytly se problémy s nepodporovanými hodnotami stylů napříč prohlížeči, které byly opraveny. Aplikace se testovala v prohlížečích **Mozilla Firefox** (verze 98.0.1), **Google Chrome** (verze 99.0.4844.74) a **Microsoft Edge** (verze 99.0.1150.39).
Ve všech těchto prohlížečích se došlo k závěru, že aplikace funguje správně.

7.2 Uživatelské testování

Účelem uživatelského testování je především ověřit použitelnost uživatelského rozhraní a funkcí aplikace. Provádějí ho reální uživatelé, to znamená potenciální či stávající uživatelé dané aplikace, protože vývojáři či testeři nedokáží věrně napodobit chování nového reálného návštěvníka webu.

Uživatelské testování obvykle probíhá tak, že tester (myšleno koncový uživatel) dostane úkol či sadu úkonů (scénář), které se následně pokouší splnit. K testerovi je přidělen pozorovatel, který zapisuje chování a postup uživatele včetně uvažování. Po skončení testu ještě tester odpovídá na otázky o průběhu

testování, jeho dojmech a pod. z testované aplikace.

7.2.1 Testovací subjekty

Protože cílovou skupinou aplikace jsou především studenti, 85% dotázaných bylo vybráno z kategorie 15-26 let pro zahrnutí jak středoškolských, tak vysokoškolských studentů. Minoritně byly zahrnuty i kategorie do patnácti let a 26-50 let.

V testovacím vzorku lidí byla potřeba zastoupení alespoň 25% leváků kvůli možnosti levého a pravého otáčení karet, které se v rámci předběžného přístupu ukázalo jako žádoucí.

7.2.2 Testování

Po odstranění nalezených chyb během programátorského testování byla tedy aplikace zpřístupněna k používání šesti lidem, kterým byl předložen testovací scénář, který ilustruje běžné použití aplikace, a následně dotazník o práci s aplikací. Poté následoval volný režim, kde tester procházel aplikaci dle vlastního uvážení. Během testování autor zaznamenával postup a chování subjektů.

Scénáře

Následují předpřipravené úkoly pro testery, které simulují primární využití aplikace uživatelem.

Scénář č. 01

1. Zaregistrovat se do webové aplikace.
2. Ověřit si registrovaný email.
3. Přihlásit se do aplikace.
4. Vytvořit novou osobní kolekci.
5. Vytvořit 3 karty uvnitř této kolekce.
6. Vytvořit kartu **s překlepem** uvnitř této kolekce.
7. Opravit kartu s překlepem.
8. Spustit výukový/herní režim pro tuto kolekci.
9. Zveřejnit tuto kolekci.
10. Odhlásit se z aplikace.

Scénář č. 02

1. Zadat špatné heslo při přihlášení.
2. Obnovit zapomenuté heslo.

3. Přihlásit se do aplikace.
4. Zobrazit uživatelský profil.
5. Změnit si heslo.
6. Vyhledat libovolnou veřejnou kolekci, které nejste autorem.
7. Přidat kolekci do oblíbených.
8. Vytvořit privátní kopii oblíbené kolekce a upravit ji.
9. Odhlásit se z aplikace.

Scénář č. 03

1. Přihlásit se do aplikace na mobilním zařízení.
2. Vyzkoušet základní funkce (tj. správa kolekcí, správa karet, spuštění výukového módu, ...)

Dotazník

Po splnění všech úkolů každý testovací subjekt vyplnil závěrečnou anketu, kde odpověděl na tyto otázky:

1. Využili byste aplikaci Flashcards pro vlastní přípravu nebo výuku?
2. Narazili jste během plnění úkolů dle scénáře na nějaké problémy? Pokud ano, uveďte úkol, problém a návrh na vylepšení.

Vyplněné dotazníky jsou uvedené v příloze: B

7.2.3 Vyhodnocení testování

Vyhodnocení testování a výsledků ankety testovacích subjektů se nachází v následujícím seznamu. Pro přehlednost nebyly duplicitní připomínky opakovaně sepisovány.

- Uživatel č. 1 (pravák)
  - Přehled kolekcí je příliš barevný, odchyluje se od stylu aplikace.
    - Styl byl upraven, tak aby kolekce měly barevné pouze ohraničení karty.
  - Karta kolekce na přehledu obsahuje příliš informací a tlačítek najednou.
    - Byl opraven jednotný styl ikon.
    - Tato připomínka byla přidána do seznamu oprav.
  - Na telefonu je možné pro ušetření místa na obrazovce u některých akčních tlačítek zobrazovat pouze ikonu (vytvořit, obnovit, zpět)
    - Tato připomínka byla opravena.

    - Uživatel uvedl, že po vytvoření duplikátu kolekce by kromě oznámení o úspěšnosti mohla aplikace přejít na vytvořenou kolekci nebo alespoň na přehled osobních kolekcí
        - Tato připomínka byla přidána do seznamu možných rozšíření.
    - Uživateli přišel zvláštní uvítací pozdrav v odkazu na profil.
        - Do seznamu možných rozšíření bylo přidáno tento pozdrav po první návštěvě profilu skrýt

- Uživatel č. 2 (levák)
    - Uživatel uvedl, že mu přišlo zvláštní otáčet karty zprava doleva.
        - Není považované za chybu. Otáčení lze nastavit v profilu uživatele.
    - Uživatel ocenil možnost otáčení karet zleva/zprava.
    - Uživatel vytkl časté uspávání serveru a následnou nedostupnost aplikace.
        - Řešením je pouze zakoupení vyššího plánu na Heroku nebo vlastní server. Není předmětem práce.
- Uživatel č. 3 (levák i pravák)
    - Uživatel uvedl, že by ocenil českou lokalizaci aplikace.
        - Tato připomínka byla přidána do seznamu možných rozšíření.
    - Uživatel uvedl požadavek na otevírání kolekcí pomocí kliknutí na kartu místo tlačítka "Enter"
        - Požadavek byl připsán na seznam možných oprav.
    - Uživatel uvedl, že na mobilním zařízení iPhone SE 2019, iOS v. 15.4.1 v prohlížeči Safari aplikace nefunguje.
        - Po prozkoumání problému bylo zjištěno, že problém způsobuje konfigurace Safari prohlížeče, kde Apple blokuje soubory cookies napříč doménami. (cross site tracking protection [83])
- Uživatel č. 4 (pravák)
    - Uživatel ocenil možnost otáčení karet zleva/zprava.
    - Uživateli chyběly toast notifikace při změně hesla
        - Připomínka byla ihned opravena.
- Uživatel č. 5 (pravák)
    - Uživateli při používání aplikace na mobilním zařízení překážela postranní navigace (i zmenšená verze) navrhl přidat možnost skrytí lišty úplně do plovoucího tlačítka.
        - Tato připomínka byla přidána do seznamu možných rozšíření.
- Uživatel č. 6 (pravák)

    - Uživatel ocenil moderní vzhled a sbalitelnost postranní navigace.
    - Uživatel ocenil mobilní vzhled, konkrétně v horizontálním (landscape) režimu.
    - Uživatel uvedl pár drobných gramatických chyb/překlepů.
        - Tyto připomínky byly ihned opraveny.
    - Uživatel uvedl chybu při restartování výukového režimu a přesměrování při resetování hesla.
        - Tyto chyby byly opraveny.
    - Uživatel uvedl návrh, že kolekce by kromě názvu mohla mít i popisek.
        - Návrh byl přidán do seznamu možných rozšíření.
- 50% dotázaných uživatelů nahlásili problémy s responsivitou a UI aplikace. Konkrétně vyhledávací pole kolekcí je příliš malé a text na tlačítku pro vytvoření kolekce je přesahuje.
- Většina dotázaných uživatelů prohlásilo, že by aplikaci pro přípravu využilo, především na svém mobilním zařízení.
    - V dalším vývoji je třeba se více soustředit na podporu pro mobilní zařízení.

7.3 Shrnutí kapitoly

Tato kapitola se zabývala testováním vyvíjené aplikace. Aplikace byla podrobena programátorských testům v rámci vývoje a po nasazení uživatelským testům, které částečně sloužily jako akceptační.

Z testovacích závěrů byly odvozeny především problémy s uživatelského rozhraní, ale jednalo se spíše o drobné připomínky, které nebrání funkčnosti aplikace.

Část z nich byla opravena a znovu otestována, větší problémy nebo nápady na úpravu a vylepšení byly přidány do seznamu možných rozšíření a oprav v sekci Další možný vývoj.

Jedinou připomínkou, která brání funkčnosti aplikace je rozdílná doména frontend a backend části. Řešením je zaplacení vlastní domény v rámci Heroku nebo nasazení serverové části na vlastní server. Po této úpravě je aplikace funkční napříč všemi prohlížeči, definovanými v nefunkčním požadavku č. 1.

Kapitola 8

Závěr

V závěrečné kapitole je popsáno zhodnocení celé práce a možné úpravy, vylepšení či rozšíření.

8.1 Zhodnocení

Cílem práce byl návrh a vývoj webové výukové aplikace formou kartiček flashcards, která by usnadnila samostudium a přípravu na zkoušky či testy.

V práci byla provedena rešerše metody výuky pomocí karet a existujících řešení, která umožňují vytváření vlastních karet.

Na základě provedené analýzy byly vybrány funkční a nefunkční požadavky, podle kterých byl vytvořen diagram případů užití a analytický doménový model tříd.

Poté byla provedena analýza technologií pro implementaci, na jejímž základě byly vybrány nejvhodnější technologie - Java, Spring Boot, PostgreSQL, JavaScript, React.js a Heroku.

Potom co byly zvoleny vhodné technologie, byla popsána implementace aplikace včetně její fyzické architektury, použitých závislostí, nástrojů, zabezpečení, poskytovaného uživatelského rozhraní a nasazení na cloudovou platformu.

Nakonec byla aplikace otestována a ohodnocena potencionálními uživateli.

Zadání práce považuji za splněné. Všechny funkční a nefunkční požadavky jsou splněné, aplikace je implementována, otestována, nasazena a připravena ke skutečnému provozu.

Dále jsou uvedeny možnosti zlepšení a rozšíření aplikace.

Nebyla realizována implementace reklam, která byla rozšířením zadání jako nefunkční požadavek NFR9, ale nemá vliv na funkčnost aplikace.

8.2 Další možný vývoj

Implementovaná aplikace funguje a splňuje funkční i nefunkční požadavky, ale vždy je co vylepšovat, zejména u podobných aplikací. Prní možností jak aplikaci lze vylepšit je zapracovat problémy a připomínky, nalezené během uživatelského testování.

V další řadě lze přidávat nové funkce, které by vylepšily chod aplikace či přidali novou funkcionalitu.

Zapracování připomínek:

- Přesunutí vedlejších akčních tlačítek kolekce (smazat, upravit) do kontextového menu.
- Zobrazovat uvítací pozdrav u profilu jen do první návštěvy profilu.
- Vylepšit responsivitu mobilního zobrazení.
- Možné otevření kolekce při kliknutí na ni bez nutnosti tlačítka "Enter".

Rozšíření požadavků:

- Podpora více jazyků.
- Tmavý režim.
- Rozšíření atributů kolekce o popis.
- Zobrazování reklam mezi kartami/kolekcemi pro podpoření konceptu aplikace jako bezplatná platforma karet flashcards.
- Rozšíření funkcí cílené na administrátory jako správa kolekcí, úprava kategorií, apod.
- Statistika nad odpověďmi.
- Předdefinované druhy karet - ano/ne, multichoice, otázka/odpověď, doplň do věty, ...
- Podpora multimédií - obrázky, zvuky, videa, ...
- Rozšíření režimu výuky o možnost zobrazovat nejdříve odpověď.
- Možnost úplného skrytí navigační lišty do plovoucího tlačítka.
- Sdílení soukromých kolekcí konkrétnímu člověku.
- Změna architektury na client-dispatcher-server, pro balancování náporu na server(y).
- Cachování častých dotazů na úrovni serveru (popřípadě dispatcheru).
- Využití specializovaných technologií na full-textové vyhledávání (např. Elastic search) pro snížení zátěže systému.
- Systém předplatného s pokročilými funkcemi (multimédia/předdefinované druhy karet,...) a vypnutím reklam.

Další rozšíření:

- Vytvoření placeného účtu na platformě Heroku, který umožní mít vlastní doménu a server se nebude každých 30 minut vypínat.

Příloha A

Seznam Odkazů

- **Git repozitář:**

  backend: `https://github.com/TheEvilCount/flashcards-be`

  frontend: `https://github.com/TheEvilCount/flashcards-fe`

- **Adresy nasazené aplikace:**

  backend: `https://poustdan-flashcards-be.herokuapp.com`

  kontext: `/flashcards/api/v1/`

  generovaná OpenApi specifikace: `/flashcards/api/v1/v3/api-docs`

  frontend: `https://danielpoustka.cz/flashcards`

Příloha B

Vyplněné dotazníky

Dále se nacházejí odpovědi testovacích subjektů na dotazník v rámci uživatelského testování. Data byla pro lepší čitelnost transformována do přehledné tabulky B.1.

Dotazníky			
		Otázky	
č. test. uživatele	Dominantní ruka	Využili byste aplikaci Flashcards pro vlastní přípravu nebo výuku?	Narazili jste během plnění úkolů dle scénáře na nějaké problémy? Pokud ano, uveďte scénář, úkol, problém a návrh na vylepšení.
1	pravá	Asi ano, ale nejvíce bych aplikaci využíval v mobilním telefonu.	Na žádné kritické problémy jsem nenarazil. Mám jen malé připomínky k UI. Konkrétně jsem chvíli hledal "osobní"kolekce, přehled kolekcí je příliš křiklavě barevný, karta kolekce je nepřehledná / obsahuje příliš informací najednou.
2	levá	Nejspíše ano.	Kromě časté nedostupnosti z důvodu uspání serveru jsem na žádnou kritickou chybu nenarazil.
3	obě	Ano, pod podmínkou přidání českého jazyka a podpory pro zařízení s iOS.	Aplikace nefunguje na mém mobilním zařízení. Problémy s orientací v aplikaci. Chybí CZ jazyk.
4	pravá	Ano, například při přípravě k maturitě.	Ne.
5	pravá	Ano	Ne, nenarazil.
6	pravá	S flashcards nemám zkušenost, ale zřejmě je vyzkouším k přípravě a opakování na zkoušky.	Nenarazil jsem.

Tabulka B.1: Dotazník - Přehled odpovědí

Příloha C

Seznam použitých zkratek

ACID Čtyři databázové vlastnosti: atomicita (**A**tomicity), konzistence (**C**onsitency), izolovanost (**I**solation), trvalost (**D**urability)

API Application Programming Interface

BE Back-end

CRUD Čtyři základní datové operace: Create, Read, Update, Delete.

CSS Cascading Style Sheets

DAO Data Access Object

DTO Data Transfer Object

FE Front-end

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

JS JavaScript

JSON JavaScript Object Notation

LAMP Kombinace open source technologií pro implementaci webových stránek, nejčastěji: systém GNU/**L**inux, web. server **A**pache, databáze **M**ySQL, jazyk **P**HP[20].

ORM Object-Relational Mapping

PaaS Platforma jako služba (platform as a service)[44].

REST Representational State Transfer

SQL Structured Query Language

UI Uživatelské rohraní

URL Uniform Resource Locator

Příloha D

Slovníček

Full Stack Pokrývající jak serverovou stranu, tak i klientskou stranu aplikace.

framework Softwarová struktura, sloužící jako podpora při programování, vývoji a organizaci softwarových projektů. Může obsahovat podpůrné programy, knihovny, podporu pro návrhové vzory či doporučené postupy.

Open source Počítačový software s veřejným zdrojovým kódem a možností podílení se na jeho tvorbě[9].

Příloha E

Literatura

[1] Jakub Hranáč, "Porovnání přístupů k ukládání dat ve webových aplikacích," Bakalářská práce, ČVUT v Brně, 2008, p. 43, str. 35-35. [Online]. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=116103 (Shlédnuto 2021-11-30).

[2] David Bryson, "Using flashcards to support your learning," *Journal of Visual Communication in Medicine*, vol. 35, no. 1, 2012. [Online]. Dostupné z: https://doi.org/10.3109/17453054.2012.655720 (Shlédnuto 2021-08-23).

[3] Merkul Vasil. (2021) Návrh a vývoj web. aplikací flash cards. [Online]. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/94661/F3-BP-2021-Merkul-Vasil-Navrh%20a%20vyvoj%20web%20aplikaci%20flash%20cards.pdf (Shlédnuto 2021-10-19).

[4] Neznámý autor. (2021) Metoda učení pomocí memorovacích kartiček. BOBO BLOK spol. s r.o. Blog. [Online]. Dostupné z: https://bobo.cz/blog/metoda-uceni-pomoci-memorovacich-karticek (Shlédnuto 2021-10-23).

[5] Jan Kohut. (2021) Anki nebo quizlet? která aplikace vám pomůže nejvíce s učením?! [Online]. Dostupné z: https://jakserychlenaucit.cz/flashcards-metody-uceni/ (Shlédnuto 2021-11-02).

[6] Jill Duffy. (2021) Quizlet review. [Online]. Dostupné z: https://www.pcmag.com/reviews/quizlet (Shlédnuto 2021-11-02).

[7] Anki. (2021) Anki official website. [Online]. Dostupné z: https://apps.ankiweb.net/ (Shlédnuto 2021-11-04).

[8] Ransom Patterson. (2021) These flashcard apps will help you study better in 2021. [Online]. Dostupné z: https://collegeinfogeek.com/flashcard-apps/ (Shlédnuto 2021-11-09).

[9] Neznámý autor. (2021) The open source definition. [Online]. Dostupné z: https://opensource.org/osd (Shlédnuto 2021-11-04).

[10] Neznámý autor. (2021) Brainscape - oficiální stránka. [Online]. Dostupné z: https://www.brainscape.com/spaced-repetition (Shlédnuto 2021-11-09).

[11] Neznámý autor. (2021) Cram - oficiální stránky. [Online]. Dostupné z: https://www.cram.com/ (Shlédnuto 2021-11-09).

[12] (2021) Nfr: Nefunkční požadavky. [Online]. Dostupné z: https://thekomix.cz/blog/nfr-nefunkcni-pozadavky/ (Shlédnuto 2021-11-17).

[13] Oracle. (2021) What is a database? [Online]. Dostupné z: https://www.oracle.com/cz/database/what-is-database/ (Shlédnuto 2021-11-13).

[14] Mark Smallcombe. (2021) Sql vs nosql: 5 critical differences. [Online]. Dostupné z: https://www.xplenty.com/blog/the-sql-vs-nosql-difference/ (Shlédnuto 2021-11-13).

[15] W. A. Higashino, M. A. M. Capretz, K. Grolinger, and A. Tiwari. (2013) Data management in cloud environments: Nosql and newsql data stores. [Online]. Dostupné z: https://journalofcloudcomputing.springeropen.com/track/pdf/10.1186/2192-113X-2-22.pdf (Shlédnuto 2021-11-22).

[16] Neznámý autor. Oracle vs mssql. [Online]. Dostupné z: https://cs.education-wiki.com/1535311-oracle-vs-mssql (Shlédnuto 2021-11-22).

[17] (2021) Oracle cloud price list. [Online]. Dostupné z: https://www.oracle.com/cloud/price-list.html#database-bm (Shlédnuto 2021-11-22).

[18] Toby Wolpe. (2014) Mysql: Why the open source database is better off under oracle. [Online]. Dostupné z: https://www.zdnet.com/article/mysql-why-the-open-source-database-is-better-off-under-oracle/ (Shlédnuto 2021-11-29).

[19] (2021) Mysql - oficiální stránky. [Online]. Dostupné z: https://www.mysql.com/ (Shlédnuto 2021-11-29).

[20] Ben Lutkevich. Lamp (linux, apache, mysql, php). [Online]. Dostupné z: https://whatis.techtarget.com/definition/LAMP-Linux-Apache-MySQL-PHP (Shlédnuto 2021-11-29).

[21] What is postgresql? - oficiální dokumentace. [Online]. Dostupné z: https://www.postgresql.org/docs/current/intro-whatis.html (Shlédnuto 2021-11-29).

[22] Richard Peterson. (2021) What is postgresql? introduction, advantages & disadvantages. [Online]. Dostupné z: https://www.guru99.com/introduction-postgresql.html#6 (Shlédnuto 2021-11-29).

[23] (2019) Postgresql: a closer look at the object-relational database management system. [Online]. Dostupné z: https://www.ionos.com/digitalguide/server/know-how/postgresql/ (Shlédnuto 2021-11-29).

[24] ITNetwork.cz Vita. (2021) Kterou sql databázi použít? [Online]. Dostupné z: https://www.itnetwork.cz/sql/kterou-sql-databazi-pouzit/ (Shlédnuto 2021-11-13).

[25] (2022) Express - oficiální stránka. [Online]. Dostupné z: https://expressjs.com/ (Shlédnuto 2022-01-22).

[26] (2022) Express - produkční využití. [Online]. Dostupné z: https://expressjs.com/en/resources/companies-using-express.html (Shlédnuto 2022-01-22).

[27] (2022) Laravel - oficiální stránka. [Online]. Dostupné z: https://laravel.com/ (Shlédnuto 2022-01-22).

[28] Solomon Eseme. (2020) Top 5 backend frameworks 2021. [Online]. Dostupné z: https://codeburst.io/top-5-backend-frameworks-2021-f3a73a18aa5f (Shlédnuto 2022-02-11).

[29] (2022) Usage statistics of server-side programming languages for websites. [Online]. Dostupné z: https://w3techs.com/technologies/overview/programming_language (Shlédnuto 2022-02-15).

[30] (2014) Laravel - produkční využití. [Online]. Dostupné z: https://codecondo.com/15-websites-built-with-laravel/ (Shlédnuto 2022-01-15).

[31] (2022) Spring boot - oficiální stránka. [Online]. Dostupné z: https://spring.io/projects/spring-boot (Shlédnuto 2022-01-22).

[32] Francisco Sáez. (2022) Convention over configuration. [Online]. Dostupné z: https://facilethings.com/blog/en/convention-over-configuration (Shlédnuto 2021-11-22).

[33] Barbora Koďousková. (2021) Co je to api. [Online]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-api (Shlédnuto 2021-12-09).

[34] (2021) Graphql - oficiální stránka. [Online]. Dostupné z: https://graphql.org/ (Shlédnuto 2021-12-09).

[35] (2019) What is a graphql? [Online]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-graphql (Shlédnuto 2021-12-09).

[36] (2020) What is a rest? [Online]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-a-rest-api (Shlédnuto 2021-12-28).

[37] (2022) What is a rest? [Online]. Dostupné z: https://restfulapi.net/ (Shlédnuto 2022-01-24).

[38] Bc. Jan Bobisud. Automatické generování restful rozhraní. Diplomová práce. [Online]. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/24251/F3-DP-2014-Bobisud-Jan-prace.pdf?sequence=3&isAllowed=y (Shlédnuto 2022-01-25).

[39] (2022) React.js - oficiální stránky. [Online]. Dostupné z: https://reactjs.org/ (Shlédnuto 2022-11-01).

[40] (2021) State of js. front-end frameworks comparison. [Online]. Dostupné z: https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/ (Shlédnuto 2022-11-02).

[41] Neznámý autor. (2021) Vue.js official website. [Online]. Dostupné z: https://vuejs.org/v2/guide/ (Shlédnuto 2021-11-05).

[42] Shaumik Daityari. (2021) Angular vs react vs vue: Which framework to choose in 2021. [Online]. Dostupné z: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/ (Shlédnuto 2021-11-05).

[43] Sai kumaresh. (2020) Top 4 free cloud hosting platforms. [Online]. Dostupné z: https://medium.com/beyondx/top-4-free-cloud-hosting-platforms-ea7715e9823f (Shlédnuto 2021-12-15).

[44] (2022) Co je paas? [Online]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/what-is-paas/ (Shlédnuto 2022-03-15).

[45] (2022) Heroku - oficiální stránky. [Online]. Dostupné z: https://www.heroku.com/ (Shlédnuto 2022-03-15).

[46] (2022) Red hat openshift - oficiální stránky. [Online]. Dostupné z: https://www.redhat.com/en/technologies/cloud-computing/openshift (Shlédnuto 2022-03-15).

[47] (2022) Netlify - oficiální stránky. [Online]. Dostupné z: https://www.netlify.com/ (Shlédnuto 2021-03-15).

[48] (2022) Vercel oficiální stránky. [Online]. Dostupné z: https://vercel.com/ (Shlédnuto 2022-03-15).

[49] (2022) Json, working with json - mozilla documentation. [Online]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON (Shlédnuto 2022-01-15).

[50] Chris Richardson. Pattern: Microservice architecture. [Online]. Dostupné z: https://microservices.io/patterns/microservices.html (Shlédnuto 2022-02-10).

[51] Neznámý autor. (2021) Figma - oficiální stránky. [Online]. Dostupné z: https://www.figma.com/ (Shlédnuto 2021-11-05).

[52] (2022) Swagger - oficiální stránky. [Online]. Dostupné z: https://swagger.io/ (Shlédnuto 2022-03-15).

[53] (2022) Sparx systems - enterprise architect - oficiální stránky. [Online]. Dostupné z: https://sparxsystems.com/ (Shlédnuto 2022-03-15).

[54] (2022) Jetbrains intellij idea - oficiální stránky. [Online]. Dostupné z: https://www.jetbrains.com/idea/ (Shlédnuto 2022-03-15).

[55] (2022) Visual studio code - oficiální stránky. [Online]. Dostupné z: https://code.visualstudio.com/ (Shlédnuto 2022-03-15).

[56] (2022) Postman api client - oficiální stránky. [Online]. Dostupné z: https://www.postman.com/ (Shlédnuto 2022-03-15).

[57] (2022) Browser mozilla firefox - oficiální stránky. [Online]. Dostupné z: https://www.mozilla.org/cs/firefox/new/ (Shlédnuto 2022-03-15).

[58] (2022) pgadmin - oficiální stránky. [Online]. Dostupné z: https://www.postman.com/ (Shlédnuto 2022-03-15).

[59] Šimon Jůn. (2021) Proč je figma dar z nebes? [Online]. Dostupné z: https://www.simonjun.cz/blog/proc-je-figma-dar-z-nebes (Shlédnuto 2021-11-05).

[60] Tom Gregory. (2021) Maven vs. gradle in-depth comparison. [Online]. Dostupné z: https://tomgregory.com/maven-vs-gradle-comparison/#Summary (Shlédnuto 2022-01-12).

[61] (2022) Git - oficiální stránky. [Online]. Dostupné z: https://git-scm.com/ (Shlédnuto 2022-03-15).

[62] (2021) Http methods. [Online]. Dostupné z: https://restfulapi.net/http-methods/ (Shlédnuto 2022-03-21).

[63] Npm package - react query. [Online]. Dostupné z: https://www.npmjs.com/package/react-query (Shlédnuto 2022-03-21).

[64] React redux - oficiální stránky. [Online]. Dostupné z: https://react-redux.js.org/ (Shlédnuto 2022-03-21).

[65] Npm package - axios. [Online]. Dostupné z: https://www.npmjs.com/package/axios (Shlédnuto 2022-03-21).

[66] Npm package - material ui. [Online]. Dostupné z: https://mui.com/getting-started/usage/ (Shlédnuto 2022-03-21).

[67] Material design od google - oficiální stránka. [Online]. Dostupné z: https://material.io/ (Shlédnuto 2022-03-21).

[68] Npm package - redux persist. [Online]. Dostupné z: https://www.npmjs.com/package/redux-persist (Shlédnuto 2022-03-21).

[69] Npm package - formik. [Online]. Dostupné z: https://www.npmjs.com/package/formik (Shlédnuto 2022-03-21).

[70] Npm package - yup. [Online]. Dostupné z: https://www.npmjs.com/package/yup (Shlédnuto 2022-03-21).

[71] Npm package - connected react router. [Online]. Dostupné z: https://www.npmjs.com/package/connected-react-router (Shlédnuto 2022-03-21).

[72] Npm package - node sass. [Online]. Dostupné z: https://www.npmjs.com/package/node-sass (Shlédnuto 2022-03-22).

[73] Npm package - react toastify. [Online]. Dostupné z: https://www.npmjs.com/package/react-toastify (Shlédnuto 2022-03-22).

[74] Npm package - material-ui confirm. [Online]. Dostupné z: https://www.npmjs.com/package/material-ui-confirm (Shlédnuto 2022-03-22).

[75] Npm package - react error boundary. [Online]. Dostupné z: https://www.npmjs.com/package/react-error-boundary (Shlédnuto 2022-03-22).

[76] Npm package - react pro sidebar. [Online]. Dostupné z: https://www.npmjs.com/package/react-pro-sidebar (Shlédnuto 2022-03-22).

[77] Npm package - react spring. [Online]. Dostupné z: https://react-spring.io/#introduction (Shlédnuto 2022-04-03).

[78] UOOU úřad pro ochranu osobních údajů. (2021) Cookies od začátku roku 2022 pouze se souhlasem. [Online]. Dostupné z: https://www.uoou.cz/cookies-od-zacatku-roku-2022-pouze-se-souhlasem/d-53646 (Shlédnuto 2022-03-10).

[79] Npm package - react-cookie-consent. [Online]. Dostupné z: https://www.npmjs.com/package/react-cookie-consent#projects-using-react-cookie-consent (Shlédnuto 2022-03-21).

[80] Function and class components. [Online]. Dostupné z: https://reactjs.org/docs/components-and-props.html (Shlédnuto 2022-01-21).

[81] React - introducing hooks. [Online]. Dostupné z: https://reactjs.org/docs/hooks-intro.html (Shlédnuto 2022-01-21).

[82] (2022) The heroku cli. [Online]. Dostupné z: https://devcenter.heroku.com/articles/heroku-cli (Shlédnuto 2022-03-30).

[83] Cookie status - safari. [Online]. Dostupné z: https://www.cookiestatus.com/safari/ (Shlédnuto 2022-03-08).

[84] Jan Kohut. (2016) Poznejte jednoduchou ale efektivní metodu učení, ověřeno praxí! [Online]. Dostupné z: https://jakserychlenaucit.cz/flashcards-metody-uceni/ (Shlédnuto 2021-10-24).

[85] Roger S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach 8th Edition.* Mcgraw-hill Education, 2015, p. 947, iSBN-13: 978-0078022128. [Online]. Dostupné z: http://ce.sharif.edu/courses/98-99/2/ce474-2/resources/root/Roger%20S.%20Pressman_%20Bruce%20R.%20Maxin%20-%20Software%20Engineering_%20A%20Practitioner%E2%80%99s%20Approach-McGraw-Hill%20Education%20(2014).pdf (Shlédnuto 2022-01-01).