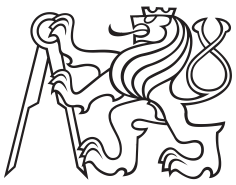
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Návrh a vývoj web aplikace „FlashCards“

Vyacheslav Tsay

Vedoucí: Ing. Božena Mannová, Ph.D.
Studijí program: Softwarové inženýrství a technologie
Specializace: Enterprise systémy
Květen 2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Tsay** Jméno: **Vyacheslav** Osobní číslo: **501252**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

Specializace: **Enterprise systémy**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Návrh a vývoj web aplikace „FlashCards“

Název bakalářské práce anglicky:

Design and development of the "FlashCards" application

Pokyny pro vypracování:

Cílem projektu je návrh a vývoj webové aplikace "FlashCards". Kartička (flash card) má na jedné straně má pojem na obrácené straně definici. Aplikace umožní vytvářet sety kartiček a systematizovat výuku. Aplikace bude sloužit jako pomůcka pro výuku a přípravu ke zkouškám.
1. Seznamte se s problematikou využití testů ve výuce.
2. Proveďte analýzu dostupných existujících řešení podobných aplikací, proveďte jejich porovnání a vyhodnocení.
3. Na základě provedené analýzy navrhněte základní funkcionality navrhované aplikace.
4. Zvolte architekturu aplikace a vyberte nejvhodnější technologie pro implementaci. Výběr technologií zdůvodněte.
5. Aplikaci implementujte a otestujte včetně uživatelských testů.
6. Zhodnoťte výsledky a navrhněte případné další funkcionality nebo jiná zlepšení.
7. Při řešení využívejte vhodných prostředků softwarového inženýrství..

Seznam doporučené literatury:

[1] Roger S. Pressmann Bruce Maxim: Software Engineering: A Practitioner's Approach ,
ISBN-10: 9780078022128
[2] Kornell, Nate. (2009). Optimising learning using flashcards: Spacing is more effective than cramming. Applied Cognitive Psychology, 23(9), 1297-1317. Dostupné z: https://onlinelibrary.wiley.com/doi/abs/10.1002/acp.1537
[3] Owen, Michael. (2023). Active Recall: The Most Effective High-Yield Learning Technique. Dostupné z: https://www.osmosis.org/blog/2022/02/21/active-recall-the-most-effective-highyield-learning-technique

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: **15.02.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **21.09.2025**

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

Rád bych poděkoval své vedoucí bakalářské práce, Ing. Boženě Mannové, Ph.D., za její cenné rady a konzultace. Dále bych chtěl vyjádřit vděčnost svým rodičům a přátelům za jejich neustálou podporu.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a využil pouze zdroje uvedené v seznamu literatury, v souladu s Metodickým pokynem č. 1/20.

V Praze, 24. května 2024

Abstrakt

Tato bakalářská práce je zaměřena na návrh a vývoj webové aplikace "FlashCards", která bude slouží jako pomůcka pro výuku a přípravu ke zkouškám s využitím intervalového opakování.

Na základě principu intervalového opakování pro flash karty a analýzy existujících řešení, byly definovány požadavky a případy užití pro aplikaci. Dále na základě těchto požadavků byly navrženy architektura aplikace, relační model, výpočty pro intervalové opakování, koncové body pro API a uživatelské rozhraní. Nakonec byla aplikace implementována a otestována pomocí uživatelských testů.

Klíčová slova: flash karty, Java, Spring Boot, React, SQL, webová aplikace, výuka, aktivní pamatování, intervalové opakování

Vedoucí: Ing. Božena Mannová, Ph.D.
Katedra počítačů

Abstract

This bachelor thesis focuses on the design and development of the web application "FlashCards", which will serve as an tool for teaching and exam preparation using spaced repetition.

Based on the principle of spaced repetition for flashcards and analysis of existing solutions, requirements and use cases for the application were defined. Further, based on these requirements, the application architecture, relational model, calculations for spaced repetition, API endpoints and user interface were designed. Finally, the application was implemented and tested using usability tests.

Keywords: flashcards, Java, Spring Boot, React, web application, education, active recall, spaced repetition

Title translation: Design and development of the "FlashCards" application

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

1.1 Téma a cíl práce

Téma bakalářské práce se zaměřuje na návrh a vývoj webové aplikace "FlashCards", která bude sloužit jako efektivní a interaktivní pomůcka pro výuku a přípravu ke zkouškám. Aplikace bude představovat digitální alternativu tradičním papírovým kartám a bude poskytovat uživatelům nástroj pro systematické studium různých témat a současně zlepšení jejich schopnosti pamatovat si důležité informace. Důraz bude kladen na jednoduchost použití, což umožní uživatelům vytvářet a spravovat své vlastní sady karet s jednoduchostí a efektivitou.

1.2 Motivace

Technologie a digitalizace se v dnešní době stávají nedílnou součástí každodenního života a významně ovlivňují mnoha jeho aspektů. Klasické papírové karty pro učení a memorizace jsou sice osvědčeným prostředkem[1], ze zkušenosti však lze identifikovat následující nevýhody:

- **Časová náročnost**: vytváření papírových karet ručně může být časově náročné. Psaní nebo tisk každé karty a jejich ruční organizace může trvat delší dobu, zejména při práci s velkým množstvím informací.
- **Omezenost obsahu**: papír jako médium má své omezení, která brání v plném využití některých moderních prvků, které by pomohly při studiu, jako jsou audio záznamy, složité obrázky a videa.
- **Poškození a ztráta**: papírové karty se mohou být snadno ztraceny nebo poškozeny, což vede k ztrátě informací a nutnosti vytvořit je znovu.

- **Omezená flexibilita**: papírové karty neumožňují úpravy obsahu. Pokud se informace změní nebo bude potřeba přidat nové informace, je nutné vytvořit nové karty, což zvyšuje spotřebu papíru a vytváří se odpad.
- **Spotřeba papíru**: vytváření a používání papírových flash karet zvyšuje spotřebu papíru. Tisk a výroba papíru zahrnují těžbu dřeva, což vede ke zhoršování životního prostředí.

Vzhledem k těmto nevýhodám je digitalizace jedním z prostředků pro jejich minimalizace. Zejména formát webové aplikace pro flash karty bude nabízet efektivnější, flexibilnější a udržitelnější alternativu pro učení a memorizace. Kromě toho, existující flashcard aplikace obsahují reklamy, které mohou odvádět pozornost uživatele během učení, a některé užitečné funkce jsou dostupné pouze za předplatné.

Kapitola 2

Flash karty

Active recall (nebo practice testing, testing effect, retrieval practice, česky aktivní pamatování) je studijní technika, která spočívá v aktivním získávání informací z paměti prostřednictvím otázek a testů, které si sami vytváříte. Namísto pasivního prohlížení nebo opětovného čtení materiálu se záměrně dotazujete na materiál, který se snažíte naučit. Tato studijní technika je jedním z nejefektivnějších způsobů, jak se mozek člověka učí a ukládá informace [2][3]. S aktivním pamatováním je často spojováno používání flash karet, které umožňují strukturované opakování informací prostřednictvím otázek a odpovědí.

Flash karty jsou učební karty, které slouží k učení a opakování informací. Většinou se vyrábějí z papíru a na jedné straně obsahují otázku nebo pojem, zatímco na druhé straně je odpověď nebo vysvětlení. Jsou považovány za jedny z klasických studijních nástrojů a to s dobrým důvodem, neboť podporují studium prostřednictvím aktivního pamatování [1]. Kromě toho lze uvést následující výhody:

- **Levnost**: flash karty jsou cenově dostupným nástrojem pro studium. Poznámkové bloky nebo jednoduché kousky papíru, z nichž se dá vyrobit karty, jsou pro studenty ekonomickým řešením.

- **Všestrannost**: flash karty lze vytvářet a používat pro jakýkoliv předmět a nejčastěji se používají při přípravě na zkoušky. Neexistují žádná omezení, k čemu mohou být použity.

- **Přenositelnost**: oproti učebnicím a sešitům flash karty se vyznačují snadnou přenositelností, i když jde o rozsáhlé sady. Vzhledem k tomu, že studenti často studují více předmětů současně, mohou s sebou pohodlně přenášet více sad karet a zbavit se tak hromady těžkých učebnic.

- **Přizpůsobitelnost**: nabízí se možnost přizpůsobení. Sady pro různé předměty lze například různě vybarvit nebo doplnit obrázky pro snadné odlišení.

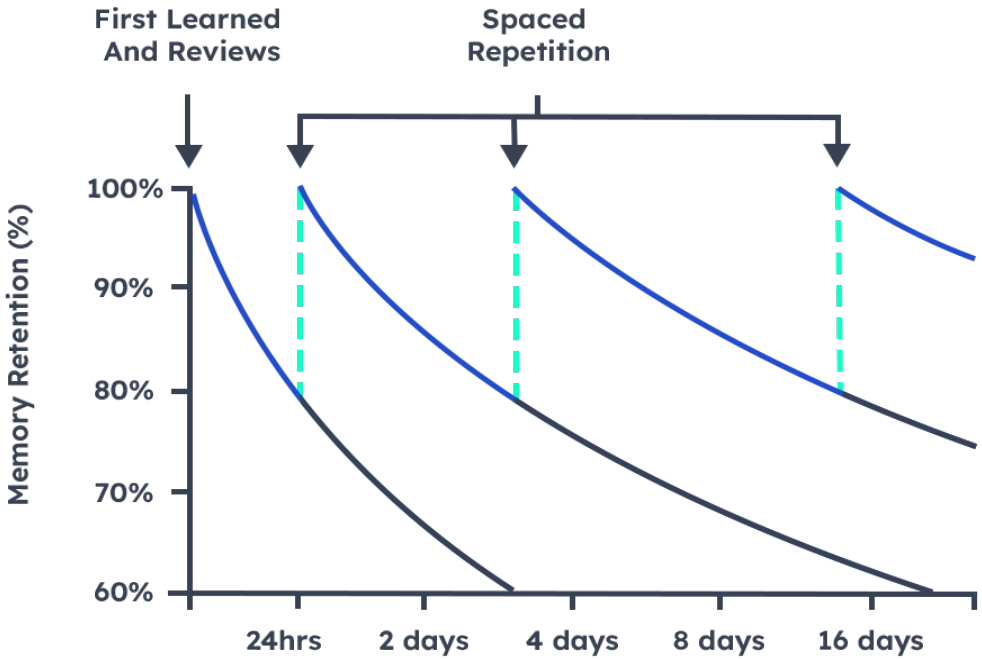
Jsou také populárním a univerzálním nástrojem, který se používá v různých oblastech pro efektivní studium. Například studenti medicíny vytvářejí je pro memorizaci anatomických struktur, léků a léčebných postupů. V oblasti matematiky a přírodních věd lze flash karty využít k procvičování vzorců, klíčových definic a principů. V jazykovém studiu jsou efektivním prostředkem pro učení nových slov, frází a gramatiky. Díky své jednoduché koncepci a flexibilitě se flash karty staly oblíbeným nástrojem nejen pro individuální učení, ale také pro skupinovou přípravu na zkoušky a kolektivní zdokonalování znalostí ve výuce.

2.1 Leitnerův systém

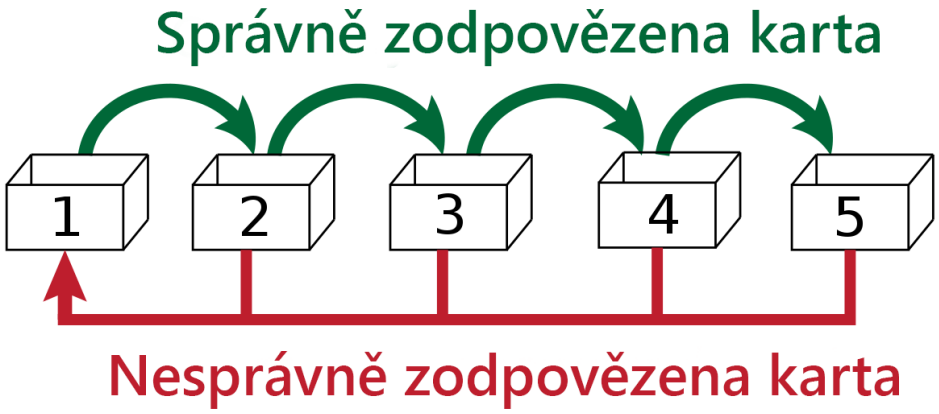
Intervalové opakování (anglicky spaced repetition) je technika založená na poznatcích německého psychologa Hermanna Ebbinghausa, která napomáhá efektivnímu zapamatování informací, obzvláště při využívání učení s flash kartami. Tato metoda spočívá v častějším zobrazování nově zavedených a náročnějších flash karet, zatímco starší a jednodušší jsou zobrazovány méně často [4]. Na obrázku 2.1 je zobrazena křivka zapomínání při intervalovém opakování, která popisuje postupné snižování pravděpodobnosti zapomnění informace v závislosti na čase od posledního opakování.

Souvisejícím pojmem je Leitnerův systém, který v roce 1972 vytvořil německý novinář a vydavatel Sebastian Leitner. Tento systém využívá principy intervalového opakování a byl navržen tak, aby optimalizoval opakování informací v závislosti na tom, jak dobře jsou dané informace již známy. Díky tomu se dosahuje maximální účinnosti učení a podporuje dlouhodobé uchování informací [6]. Základní myšlenkou systému Leitnera je rozdělení sady flash karet do tří až pěti krabic, přičemž každá krabice má vlastní časový interval pro opakování. Například, když je flash karta umístěná v první krabici s intervalem jednoho dne a student správně odpoví na otázku, flash karta je posunuta do následující krabice s delším intervalem. Naopak, pokud je otázka zodpovězena nesprávně, vrátí se zpět do první krabice jak je zobrazeno na obrázku 2.2 [7].

Tímto způsobem systém Leitnera využívá intervalové opakování k posílení paměti. Čím lépe student zná otázku, tím delší intervaly jsou mezi opakováními. Naopak, pokud má potíže, otázka se opakuje častěji.



Obrázek 2.1: Křívka zapomínání. Zdroj: [5].



Obrázek 2.2: Fungování systému Leitnera.

Kapitola 3
Analýza řešení

Tato kapitola se zabývá rešerší již existujících flashcard aplikací a následně definováním funkčních i nefunkčních požadavků a případů užití.

Rešerše existujících aplikací bude sloužit k získání relevantních informací o stávajících řešeních a jejich charakteristikách. Důraz bude kladen na jejich funkcionality, uživatelský zážitek a obecné vlastností. Cílem této analýzy je získání přehledu o tom, jakým způsobem existující řešení uspokojují potřeby uživatelů v oblasti učení s flash kartami. Na základě těchto poznatků budou následně definovány funkční požadavky, které stanoví klíčové vlastnosti, jež by nová aplikace měla obsahovat, aby efektivně podporovala učení pomocí flash karet. Současně budou identifikovány nefunkční požadavky, které se týkají aspektů jako výkon, bezpečnost a uživatelsky přívětivé rozhraní. Kapitola rovněž zahrne případy užití, které detailně popíše interakce mezi uživateli a aplikací.

3.1 Existující řešení

Na trhu existuje rozsáhlý výběr aplikací pro práci s flash kartami, zahrnující mobilní, desktopové a webové varianty. Tato řešení nabízejí různé funkcionality, případy užití a cenové možnosti. Cílem této podkapitoly je získat komplexní přehled o těchto aplikacích, detailně popsat jejich přínosy, omezení a shrnout klíčové poznatky. Aplikace byly vybrány na základě předchozích zkušeností s nimi a jejich popularity.

3.1.1 Active Recall

Active Recall je relativně nová flashcard aplikace, která nabízí intuitivní, uživatelsky přívětivé rozhraní pro jednoduché a efektivní využití. Aplikace je

zcela zdarma, neobsahuje reklamy a zahrnuje všechny funkcionality, které se potřebují pro práci s flash kartami.

Active Recall poskytuje dva základní typy flash karet: běžné karty a jazykové karty. Při vytváření jazykové karty aplikace automaticky provede překlad a přidá strojové předčítání slov. To může být užitečné pro poslech a správnou výslovnost. Další zajímavou funkcionalitou je podpora block-based textového editoru[1], která usnadňuje formátování a úpravy obsahu při vytváření běžných karet. Také je možnost přidat obrázky, video a zvukové soubory však není možné. Kromě toho, Active Recall podporuje dva režimy učení: intervalové opakování a tzv. cram[2], přičemž v režimu cram jsou všechny flash karty z sady zobrazovány v pořadí, ve kterém byly přidány [8]. Souhrn výhod a nevýhod je uveden v tabulce 3.1.

Výhody	Nevýhody
Moderní, uživatelsky přívětivé rozhraní jednoduché na použití	Nepodporuje mobilní platformy
Všechny funkce podporované aplikací jsou zdarma	Nepodporuje sdílení sad flash karet
Podpora block-based textového editoru	
Podpora intervalového opakování	

Tabulka 3.1: Výhody a nevýhody aplikace Active Recall.

3.1.2 Brainscape

Brainscape je komplexní flashcard aplikace, obsahující tisíce karet pokrývajících různá témata, včetně jazyků, literatury, matematiky a mnoha dalších. Uživatelé mají možnost procházet existující sady vytvořené jinými studenty a učiteli, nebo si mohou vytvořit své vlastní. K dispozici je pokročilý editor flash karet, umožňující přidávat obrázky a zvukové soubory, ovšem tyto rozšířené funkce jsou přístupné pouze po zakoupení předplatného. Aplikace také podporuje metodu intervalového opakování, která optimalizuje dobu vystavení karet v závislosti na reakcích uživatelů [9][10].

Brainscape dostupná jako web aplikace a pro platformy Android a iOS. Díky tomu zajišťuje také synchronizaci mezi více zařízeními, což umožňuje

[1] uspořádává obsah do samostatných bloků.

[2] z angl. slangu: připravit se na zkoušku zapamatováním si informací v krátkém časovém úseku.[14]

uživatelům snadný přístup ke svým flash kartám, ať už používají počítač, telefon nebo tablet. Tímto způsobem uživatelé mohou studovat a spravovat své materiály kdekoli a kdykoli, což přináší flexibilitu a pohodlí do procesu učení. Souhrn výhod a nevýhod Brainscape je uveden v tabulce 3.2.

Výhody	Nevýhody
Intuitivní uživatelské rozhraní	Nutnost zakoupení předplatného pro využití pokročilých funkcí
Podporuje mobilní platformy	
Sdílení sad flash karet	
Podpora intervalového opakování	

Tabulka 3.2: Výhody a nevýhody aplikace Brainscape.

3.1.3 Cram

Cram je jednoduchá, avšak efektivní flashcard aplikace s několika zajímavými funkcemi pro prohlížení a studium karet. Nabízí různé metody učení, včetně her, které obohacují výukový proces. Jedinečností této aplikace je možnost přidání nápovědy ke každé flash kartě. Kromě toho obsahuje také hodně připravených sad flash karet, které vytvořili jiní uživatelé, pokrývajících různá témata, včetně jazyků, medicíny, práva a mnoha dalších. Nelze však přidávat obrázky do karet, a stávající uživatelské rozhraní potřebuje modernizaci.[9].

Cram je dostupná na desktopu ve formě webové aplikace a nabízí také mobilní aplikaci pro snadný přístup k učení. Synchronizace uživatelské knihovny mezi více zařízeními umožňuje flexibilitu a pohodlnost při studiu bez ohledu na to, kde se uživatel nachází. Celkově je Cram užitečným nástrojem pro efektivní a poutavou výuku prostřednictvím flash karet. Souhrn výhod a nevýhod je uveden v tabulce 3.3.

Výhody	Nevýhody
Podpora mobilních platform	Zastaralé uživatelské rozhraní
Možnost učení pomocí her	Nepodporuje intervalové opakování
Obsahuje mnoha připravených sad flash karet	Nelze přidávat multimédia do flash karet

Tabulka 3.3: Výhody a nevýhody aplikace Cram.

3.1.4 Quizlet

Quizlet je nejpopulárnější aplikace pro samostudium, využívající flash karty a intervalové opakování. Uživatelé mají možnost vytvářet vlastní sady flash karet, přistupovat k sadám vytvořeným jinými uživateli nebo učiteli. Aplikace také umožňuje vkládání obrázků, zvukových záznamů do flash karet, což je však dostupné pouze při zakoupení předplatného [9].

Zajímavým prvkem je využití umělé inteligence, což zvyšuje úroveň personalizace učení. Tato technologie je využívána například v intervalovém opakování nebo v funkci Q-Chat, která je navržena s cílem zefektivnit výuku. Q-Chat není pouze nástrojem na opakování informací, ale skutečným studijním průvodcem. Pomáhá uživatelům pochopit látku, poskytuje rady při učení jazyků, provede obtížnými koncepty a pomůže dosáhnout učebních cílů. Tímto způsobem umělá inteligence aktivně podporuje a obohacuje učební proces uživatelů [11].

Quizlet je k dispozici jako mobilní aplikace, umožňující flexibilitu v učení na cestách, tak i jako webová aplikace pro pohodlné studium na všech obrazovkách. Tím poskytuje uživatelům širokou škálu možností a přístup k materiálům včetně těch, které si vytvoří sami nebo které sdílí s ostatními. Souhrn výhod a nevýhod Quizlet je uveden v tabulce 3.4.

Výhody	**Nevýhody**
Podpora mobilních platform	Nutnost zakoupení předplatného pro využití pokročilých funkcí
Sdílení sad flash karet a obsahuje mnoha připravených sad	
Podpora intervalového opakování	
Intuitivní uživatelské rozhraní	
Využití umělé inteligence	

Tabulka 3.4: Výhody a nevýhody aplikace Quizlet.

3.2 Shrnutí

V podkapitole "Existující řešení" byly představeny čtyři existující flashcard aplikace: Active Recall, Brainscape, Cram a Quizlet. Každá z těchto aplikací má své výhody a omezení a nabízí různé funkcionality a přístupy k výuce pomocí flash karet, a to jak na mobilních zařízeních, tak i na desktopu ve formátu webové aplikace. Souhrnný přehled je zobrazen v tabulce 3.5.

	Active Recall	**Brainscape**	**Cram**	**Quizlet**
Cena předplatného	Není	225 Kč/6 měsíců	Není	159 Kč/měsíčně
Verze zdarma	Ano	Ano	Ano	Ano
Platforma	Webová aplikace	Webová aplikace, mobilní aplikace	Webová aplikace, mobilní aplikace	Webová aplikace, mobilní aplikace
Statistika	Ano	Ano	Ne	Ano
Podpora intervalového opakování	Ano	Ano	Ne	Ano
Několik režimů výuky	Ano	Ano	Ano	Ano
Sdílení sad flash karet	Ano	Ano	Ano	Ano
Předpřipravené flash karty	Ano	Ano	Ano	Ano
Intuitivní rozhraní	Ano	Ano	Ne	Ano
Multimédia ve flash kartách	Ano	Ano; nutné předplatné	Ne	Ano; nutné předplatné
Pokročilý textový editor	Ano	Ano	Ne	Ano; nutné předplatné
Umělá inteligence	Ne	Ne	Ne	Ano; nutné předplatné

Tabulka 3.5: Srovnání existujících flashcard aplikací.

3.3 Analýza a sběr požadavků

Po získání důležitých poznatků z předchozích kapitol, v nichž byla provedena rešerše existujících flashcard aplikací, je tato část je zaměřena na detailní analýzu a sběr požadavků pro webovou aplikaci, která bude sloužit k výuce s flash kartami. Pro strukturování a prioritizaci těchto požadavků bude využita MoSCoW metoda, která poskytuje jasný rámec pro definici funkčních požadavků a současně umožňuje identifikaci klíčových případů užití pro flashcard aplikaci. Požadavky budou rozděleny do následujících čtyř kategorií [12]:

1. **Must have (M)**: základní požadavky nutné pro úspěšné dokončení projektu. Tyto požadavky jsou nezbytné a mají nejvyšší prioritu.
2. **Should have (S)**: požadavky, které jsou důležité, ale mohou být odloženy, pokud je to nezbytné. Tyto požadavky jsou méně kritické než "Must have", stále však mají vysokou prioritu.
3. **Could have (C)**: do této kategorie spadají požadavky, které by bylo příjemné mít, ale nejsou nevyhnutelné. Jejich implementace závisí na dostupném čase a zdrojích.
4. **Will not have (W)**: požadavky, které byly explicitně vyloučeny z aktuálního projektu. Tyto požadavky se mohou stát součástí budoucích verzí projektu.

Kromě toho budou identifikovány případy užití pro lepší pochopení, jak bude aplikace interagovat s uživateli.

3.3.1 Funkční požadavky

Funkční požadavky jsou specifikace, které popisují funkcionalitu aplikace. Tyto požadavky jsou konkrétní a měřitelné vlastnosti, které aplikace musí poskytovat, aby splnila očekávání uživatelů nebo naplnila stanovené cíle projektu. Dále budou označené jako *FRn - Ct - [Název]: [Popis]*, kde *FRn* je identifikace požadavku, *Ct* je kategorie požadavku podle MoSCoW metody.

Po provedení rešerše existujících flashcard aplikací byly stanoveny funkční požadavky:

1. **FR01 - M - Registrace a přihlášení**: aplikace umožní uživatelům snadné vytvoření účtu pomocí registrace a přihlášení prostřednictvím standardního procesu pomocí uživatelského jména a hesla.

2. **FR02 - M - Správa flash karet**: aplikace poskytne uživatelům plnou kontrolu nad svými flash kartami, umožňující přidávání nových, snadné editování obsahu, mazání a pohodlné zobrazení seznamu všech jejich flash karet.

3. **FR03 - M - Správa sad flash karet**: aplikace uživatelům umožní vytvářet, upravovat a odstraňovat sady flash karet, přičemž zároveň umožní snadné přiřazování konkrétních flash karet ke specifickým sadám, což usnadní organizaci a strukturu učebního obsahu.

4. **FR04 - M - Přidávání obrázků**: aplikace umožní přidávat obrázky k flash kartám, což zvýší efektivitu učení.

5. **FR05 - S - Statistika**: aplikace umožní sledovat pokroky uživatele při učení se flash kartami. To zahrnuje například počet absolvovaných výuk s flash karty.

6. **FR06 - S - Pokročilý textový editor**: aplikace bude podporovat block-based textový editor pro detailní úpravy obsahu flash karet.

7. **FR07 - S - Několik režimů výuky**: aplikace bude podporovat dva režimy výuky: s intervalovým opakováním a bez intervalového opakování, při němž všechny flash karty ze sady budou zobrazovány v pořadí, ve kterém byly přidány.

8. **FR08 - C - Předpřipravené sady flash karet**: nabídné hotové sady pro uživatele, které mohou být použité jako testovací flash karty pro ověření funkčnosti aplikace, nebo uživatele upraví je podle svých potřeb.

9. **FR09 - C - Vyhledávání flash karet nebo sad**: aplikace bude podporovat vyhledávání konkrétních flash karet nebo sad.

10. **FR10 - C - Klávesové zkratky**: aplikace bude podporovat klávesové zkratky pro pohodlnější ovládání aplikace.

11. **FR11 - C - Správa uživatelského profilu**: aplikace umožní uživatelům spravovat svůj uživatelský profil s možností editace osobních informací. Kromě toho bude aplikace poskytovat funkcionalitu pro přizpůsobení uživatelského rozhraní, jako je možnost změny pohledu a nastavení velikosti fontu.

12. **FR12 - C - Přihlášení pomocí OAuth2**: uživatelé budou moci přihlásit se pomocí svých účtů v externích službách, jako jsou Google,

Facebook nebo GitHub. Po úspěšném přihlášení budou mít přístup k funkcím aplikace.

13. **FR13 - C - Sdílení sad**: aplikace umožní uživatelům sdílet své vytvořené sady s ostatními uživateli.

14. **FR14 - W - Automatický překlad cizích slov**: aplikace bude automaticky překládat cizí slova v jazykových flash kartách.

15. **FR15 - W - Strojové předčítání slov**: aplikace bude automaticky přidávat strojové předčítání slov do jazykových flash karet.

16. **FR16 - W - Umělá inteligence**: do aplikace bude integrována umělé inteligence, která zefektivnit výuku.

3.3.2 Nefunkční požadavky

Nefunkční požadavky se zaměřují na aspekty, které nejsou přímo spojeny s funkcionalitou aplikace, ale na jeji charakteristiky, kvalitu a omezení. Tyto požadavky se týkají vlastností, které ovlivňují celkový výkon, spolehlivost, bezpečnost a další aspekty aplikace. Dále budou označené jako *NFRn - [Název]: [Popis]*, kde *NFRn* je identifikace požadavku.

Na základě požadavků projektu a rešerše existujících flashcard aplikací byly stanoveny nefunkční požadavky:

1. **NFR01 - Ukladání dat do SQL databáze**: všechna data musí být ukládána do relační databáze s adekvátním schématem pro uchování informací o flash kartách, uživatelích a dalších relevantních entitách.

2. **NFR02 - Hibernate jako Object-Relational Mapping**: pro objektově relační mapování aplikace bude použit Hibernate, který bude konfigurován tak, aby efektivně mapoval objekty na tabulky v databázi a umožňoval snadný přístup k datům pomocí objektových modelů.

3. **NFR03 - Spring Boot pro serverovou část**: serverová část aplikace bude implementována pomocí Spring Boot, což zajistí jednoduchou konfiguraci a rychlý vývoj. Spring Boot umožní také snadnou integraci s dalšími moduly a nástroji.

4. **NFR04 - Single Page Application**[3]: aplikace bude navržena jako Single Page Application (SPA) s využitím moderních frontend technologií

[3] dynamické přepisování aktuální stránky namísto načítání nových

(např. React) pro vytvoření plynulého uživatelského rozhraní s minimálním načítáním stránek. To zlepší uživatelskou přívětivost a celkový dojem z aplikace.

5. **NFR05 - Bezpečnost**: aplikace musí zajistit bezpečnostní opatření pro ochranu dat a uživatelských informací. To zahrnuje šifrování přenosu dat mezi klientem a serverem pomocí protokolu HTTPS, ověřování identity uživatelů a správu přístupových práv. Hesla uživatelů by měla být ukládána v bezpečné a hashované formě.
6. **NFR06 - Jazyk uživatelského rozhraní**: uživatelské rozhraní by mělo být v angličtině, aby přilákalo více uživatelů. Nicméně slova by měla být snadno srozumitelná i lidem, jejichž mateřským jazykem není angličtina.
7. **NFR06 - Kompatibilita**: aplikace musí být kompatibilní s různými webovými prohlížeči (Chrome, Firefox, Safari, Edge).
8. **NFR07 - Výkon**: aplikace by měla poskytovat rychlé odezvy na uživatelské interakce a efektivní zpracování požadavků na serverové straně, aby nemuseli čekat na načítání nebo odezvu zejména při práci s flash kartami.
9. **NFR08 - Udržovatelnost**: aplikace musí být navržena s ohledem na udržovatelnost. To zahrnuje jasnou dokumentaci, používání návrhových vzorů, a implementaci standardů kódu, což usnadní rozvoj, údržbu a rozšiřování aplikace.
10. **NFR09 - Dostupnost**: aplikace by měla být dostupná 24/7 s minimálním plánovaným výpadkem služby.

3.3.3 Případy užití

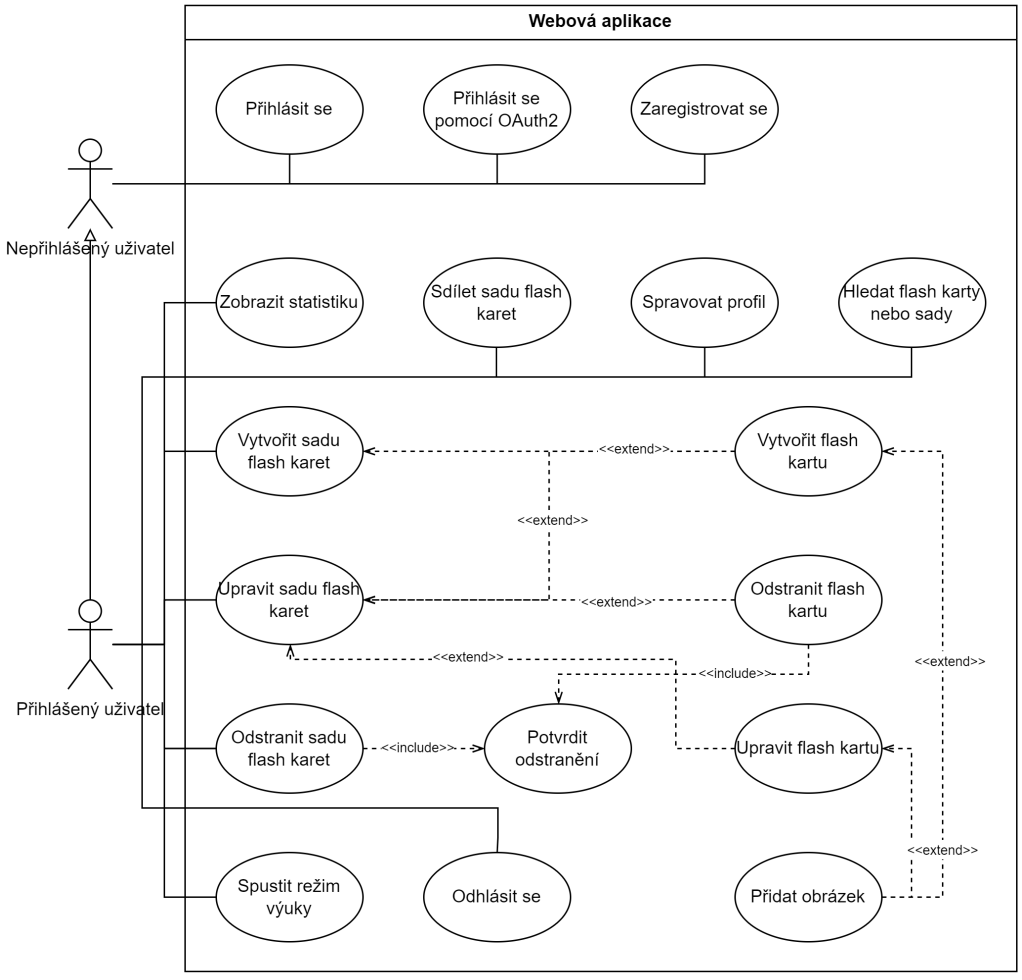
Případy užití, též use case, je metodika v oblasti softwarové analýzy a návrhu, která slouží k detailnímu popisu funkcí a interakcí softwaru z pohledu jeho budoucího využití. Případy užití umožňují vizualizovat, jak bude software interagovat s různými aktéry, ať už jsou to skuteční uživatele, automatizované procesy nebo jiné systémy [13].

Při sběru a analýze požadavků byli identifikováni následující aktéři pro případy užití, kteří budou interagovat s webovou aplikací:

1. Nepřihlášený uživatel – reprezentuje uživatele, který není přihlášen do aplikace a má omezený přístup k funkcím aplikace.

2. Přihlášený uživatel – reprezentuje uživatele, který se přihlásil do aplikace a má plný přístup ke všem jejím funkcím.

Interakce mezi těmito aktéry a aplikací jsou detailně znázorněny na následujícím obrázku 3.1.



Obrázek 3.1: Diagram případů užití.

Kapitola 4
Výběr technologií

V současné době existuje mnoho technologií v oblasti backend a frontend vývoje a není správné tvrdit, že jedna technologie je lepší než druhá, protože se používají v různých situacích. Je důležité vybírat technologie podle konkrétních potřeb a požadavků projektu.

Tato kapitola se zaměří na zkoumání a porovnávání některých z nejpopulárnějších technologií. Na základě požadavků stanovených v předchozí kapitole, jejich podpory, jednoduchosti implementace, výkonu, škálovatelnosti a zkušenosti s těmito technologiemi, budou vybrány vhodné kandidáty pro vývoj webové aplikace "FlashCards".

4.1 Backend

Backend je část softwarové aplikace, která je zodpovědná za správu a zpracování dat a za provádění úkolů, které nejsou přímo viditelné uživatelům. V případě webové aplikace backend zpracovává úlohy, jako jsou dotazy do databáze, autentifikace a autorizace, business logika[1]. S frontendem komunikuje prostřednictvím API[2] pro vyměňování dat a plnění požadavků uživatelů.

Vývoj backendu zahrnuje psaní kódu v jazycích, jako je Python, Java, Javascript, PHP nebo jiné, často s využitím různých frameworků. Tento proces také zahrnuje práci s relačními a nerelačními databázemi.

4.1.1 Databáze

Databáze je kolekce strukturovaných či nestrukturovaných dat, která je organizována tak, aby umožňovala efektivní ukládání, manipulaci a vyhledávání

[1] část kódu, který splňuje požadavky zákazníků/uživatelů
[2] Application Programming Interface

informací. Slouží k uchování velkého množství dat, k nimž se přistupuje pomocí dotazů.

Existují dva základní typy databází:

- **Relační**: často označované jako SQL[3] databáze, používají relační model pro ukládání dat do tabulek s definovanými vztahy mezi nimi. Tento model umožňuje provádět složité dotazy pomocí strukturovaného dotazovacího jazyka SQL a poskytuje konzistenci dat díky transakcí, které zaručují ACID[4] vlastnosti [15]. SQL databáze se využívají např. v internetových obchodech, kde je potřeba předdefinovat vlastnosti pro všechny produkty, jako je cena, název, popis atd.
- **Nerelační**: též NoSQL databáze, navrženy pro ukládání a manipulaci s nestrukturovanými nebo polostrukturovanými daty. Tyto databáze nepoužívají relační model a mohou nabídnout větší flexibilitu a škálovatelnost v určitých situacích, ale nezaručují konzistenci dat. Hlavními typy datových modelů jsou dokument, klíč-hodnota, wide-column a graf [16]. Např. grafová databáze se používá v sociálních sítích k reprezentaci vztahů mezi uživateli, kde každý uživatel je reprezentován jako vrchol a vztahy mezi nimi jsou reprezentovány hranami.

Pro vytváření a správu databáze je potřeba systém pro správu databází zkráceně DBMS[5], který poskytuje mechanismy pro zajištění bezpečnosti, spolehlivosti, souběžnosti, integrity uložených dat. Mezi nejznámější DBMS pro relační databáze patří PostgreSQL, MySQL, Oracle Database a Microsoft SQL Server. Tyto systémy hlavně se liší cenou, výkonností a funkcionalitou. Pro nerelační databáze jsou to MongoDB (dokument), Redis (klíč-hodnota), Apache Cassandra (wide-column) a Neo4j (graf).

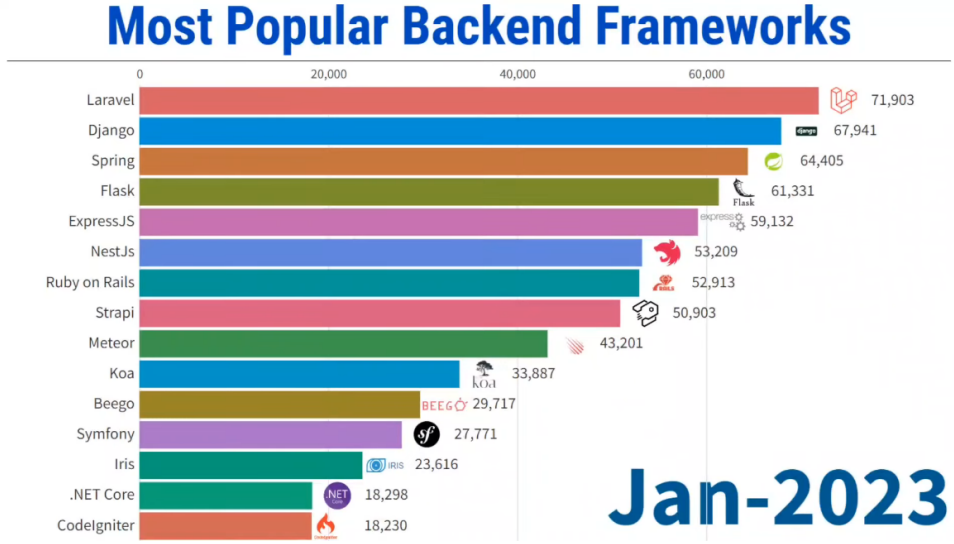
Jelikož je nezbytné zajistit konzistenci a bezpečnost dat pro účely vývoje aplikace "FlashCards", lepším kandidátem je relační databáze s DBMS PostgreSQL. Systém pro správu databází byl vybrán s ohledem na jeho cenu, funkcionalitu a zkušenosti s tímto systémem.

4.1.2 Frameworky

Framework je předpřipravená, opakovaně použitelná sada softwarových nástrojů a knihoven nabízející vývojářům základ, na kterém mohou stavět, a

[3] Structured query language
[4] Atomicity, Consistency, Isolation, Durability
[5] Database management system

zefektivňuje proces vývoje tím, že poskytuje standardizované vzory, komponenty a funkce. Na obrázku 4.1 jsou zobrazeny nejpopulárnější backend frameworky od ledna 2023. Data jsou vypočítány na základě hvězdiček v archivu GitHub [17].



Obrázek 4.1: Nejpopulárnější backend frameworky. Zdroj: [15].

Z dat na obrázku 4.1 vyplývá, že třemi nejpopulárnějšími frameworky jsou Laravel, Django a Spring Boot, které budou dále porovnány.

Laravel

Laravel je open-source backend framework založený na PHP a poprvé vydaný v roce 2011 Otwellem T. [18].

Framework využívá architektonický vzor Model-View-Controller (MVC), který usnadňuje jeho pochopení a práci s ním. Laravel disponuje rozsáhlou dokumentací a vestavěnými funkcemi, jako je například autentizace a autorizace, cachování a rozhraní příkazového řádku Artisan pro správu databází, migrace dat a generování boilerplate kódu[6], což umožňuje vývojářům soustředit se na logiku aplikace a snižuje tak opakující se práci [18][19].

Nicméně, narozdíl od svých výhod, má Laravel i některé nevýhody. Například, vestavěné funkce a abstrakce Laravelu jsou spojeny s výkonnostní zátěží. Zatímco u malých aplikací to nemusí být významný problém, u velkých a složitých aplikací to může mít vliv na výkon. Dalším nedostatkem

[6] části kódu, které se opakují na více místech s malými nebo žádnými změnami.

je krátkodobý cyklus podpory hlavních verzí, což vyžaduje od vývojářů být up-to-date pro získání aktualizací zabezpečení a oprav chyb [19].

Django

Django je open-source webový framework v jazyce Python vyvijen v 2003 Willisonem S. a Holovatym A., který podporuje rychlý vývoj a čistý design. Řídí se architektonickým vzorem Model-View-Template (MVT). Jedním z klíčových principů Django je DRY (Don't Repeat Yourself), což znamená, že vývojáři by se měli snažit psát kód bez zbytečného opakování a co nejvíce omezit duplicity [20].

Framework poskytuje různé vestavěné funkce, jako je ORM (Object-Relational Mapping) pro interakci s databázemi, systém směrování URL, šablonovací engine pro dynamické generování HTML[7], zpracování formulářů, autentizační a autorizační mechanismy a další [21].

Nevýhodou Django je rozsáhlost a požadavek na znalost celé struktury před začátkem vývoje. Django rovněž nepředepisuje žádné konvence, což může některé vývojáře odradit a někdy zpomalit vývoj. I když framework je skvělý pro konstrukci velkých projektů, pro menší projekty je nadbytečný. Těžká, monolitická struktura může být překážkou pro vývojáře hledající vysoce přizpůsobitelné, rychlejší aplikace [20].

Spring Boot

Spring Boot je open-source Java framework používaný pro vytvoření robustních a škálovatelných aplikací s minimální konfigurací, což umožňuje vývojářům soustředit se na implementaci logiky aplikace. Je nadstavbou frameworku Spring, který poskytuje komplexní infrastrukturní podporu pro vývoj Java aplikací [40].

Spring Boot se jednoduše integruje s jinými projekty z ekosystému Spring, jako jsou Spring Security, Spring Cloud, Spring Data a další. Tato integrace výrazně usnadňuje vývoj komplexních aplikací. Další výhodou je to, že framework obsahuje vestavěný webový server (například Tomcat, Jetty nebo Undertow), což umožňuje snadné spuštění aplikace a zjednodušuje nasazení bez nutnosti konfigurace externího serveru [41].

Nicméně, je důležité poznamenat, že Spring Boot má své nevýhody. Je rozsáhlý framework a orientace v něm vyžaduje čas a úsilí. V některých případech automatická konfigurace a konvence Spring Bootu nemusí odpovídat požadavkům konkrétního projektu. V takových situacích může být nutné

[7] HyperText Markdown Language

výchozí chování přepsat nebo přizpůsobit, což může být náročné a může to aplikaci připravit o všechny výše uvedené výhody [41].

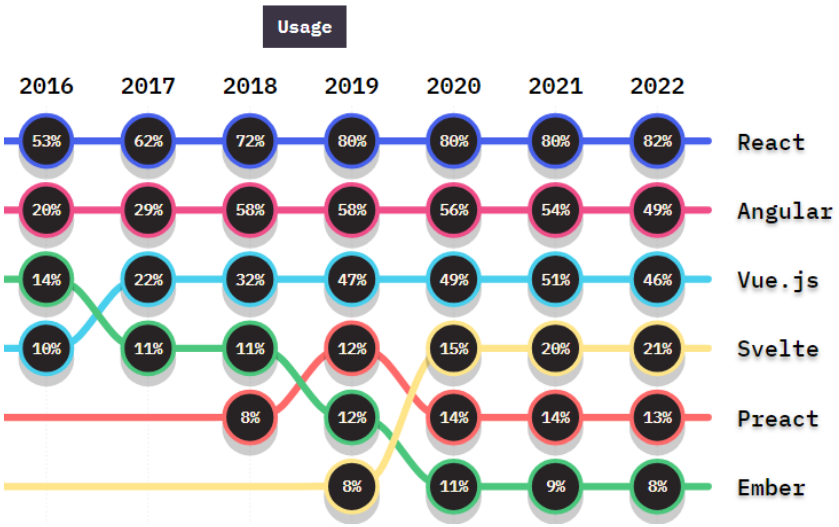
V této podkapitole byly zkoumány nejpopulárnější backend frameworky, jako jsou Laravel, Django a Spring Boot. Na základě požadavků projektu, vyhodnocení a porovnání výhod a nevýhod jednotlivých frameworků byl vybrán nejlepší kandidát pro webovou aplikaci "FlashCards"– Spring Boot. Důležitou roli při výběru hrála také zkušenost s touto technologií.

4.2 Frontend

Frontend je část aplikace nebo webové stránky, s níž uživatelé přímo interagují, zahrnující tlačítka, menu, formuláře, obrázky, text a další vizuální prvky.

Vývoj frontendu zahrnuje tvorbu těchto komponent pomocí HTML, CSS[8] a Javascriptu. HTML definuje strukturu stránky, CSS určuje styl a Javascript dodává interaktivitu a dynamické chování. Pro efektivnější vývoj se často využívají Javascript frameworky, které nabízejí funkce jako architektura založená na komponentách, správa stavů a směrování [22].

Na obrázku 4.2 jsou zobrazeny nejpopulárnější Javascript frameworky a knihovny pro rok 2022, přičemž první tři budou podrobněji zkoumány a porovnány v následujících podkapitolách.



Obrázek 4.2: Nejpopulárnější frontend frameworky a knihovny. Zdroj: [23].

[8] Cascading Style Sheets

4.2.1 React

React není framework, ale open-source Javascript knihovna vyvinutá společností Meta. Jeho hlavním cílem je usnadnit tvorbu interaktivních uživatelských rozhraní pomocí opakovaně použitelných komponent obsahujících svou logiku a ovládací prvky. Každá aplikace v Reactu se skládá z více těchto komponent.

React využívá virtuální DOM[9], což zvyšuje výkon tím, že porovnává předchozí a aktuální stavy komponent a aktualizuje pouze změněné prvky v reálném DOM, místo aby vykresloval vše znovu.

Knihovna může být náročná na pochopení pro začátečníky, protože zavádí pokročilé koncepty jako JSX[10], props, správa stavů, životní cyklus komponent a hooky. Někdy je třeba použít další knihovny pro komplexnější projekty jako Redux nebo React Router.

4.2.2 Angular

Angular, vytvořený společností Google, je kompletní Javascript framework pro vývoj webových aplikací, na rozdíl od Reactu, který je knihovnou. Jeho základními stavebními kameny jsou komponenty, které obsahují logiku a vzhled a mohou být opakovaně použity v aplikaci, podobně jako v React [24].

Jednou z významných vlastností Angular je obousměrná datová vazba, která automaticky synchronizuje data mezi modelem a uživatelským rozhraním, což usnadňuje sledování a aktualizaci stavu aplikace. Poskytuje také širokou škálu funkcí a nástrojů pro správu stavů, směrování, testování a další [25].

Vzhledem k jeho složitosti a široké škále funkcí však může vývojářům trvat dlouho, než se naučí s Angular efektivně pracovat, tím vývoj aplikace může zpomalit [25].

4.2.3 Vue.js

Vue.js je další open-source Javascript knihovna, která umožňuje snadno vytvářet interaktivní uživatelská rozhraní pomocí komponent, podobně jako v React a Angular.

Jednou z klíčových vlastností knihovny je reaktivní přístup k aktualizaci uživatelského rozhraní, který sleduje změny dat a aktualizuje pouze relevantní části uživatelského rozhraní, čímž zvyšuje výkon a efektivitu [26].

Jako relativně nová knihovna, může Vue.js však podléhat častým změnám a aktualizacím, což může způsobit nestabilitu vývoje a údržby aplikací. I

[9] Document Object Model
[10] Javascript XML

když nabízí širokou škálu nástrojů a knihoven, hledání pokročilých nástrojů pro specifické účely může být obtížné v případě komplexnějších projektu [26].

V této podkapitole byly zkoumány a porovnány nejpopulárnější frontend frameworky a knihovny, jako jsou React, Angular a Vue.js. Na základě analýzy vyplývá, že React je flexibilnější a výkonnější pro nevelké projekty než Angular. A s ohledem na předchozí zkušenosti s vývojem frontendu byl vybrán React spolu s Typescript, místo Vue.js pro vývoj webové aplikace "FlashCards".

Typescript je nadstavbou pro Javascript, která přidává typovou kontrolu a statickou analýzu kódu, čímž výrazně zlepšit jeho čitelnost, údržbu a bezpečnost.

4.3 Aplikační programové rozhraní

API, neboli aplikační programové rozhraní, jsou mechanismy, které umožňují dvěma softwarovým komponentám vzájemně komunikovat pomocí sady pravidel a protokolů. API definuje metody a datové formáty, které mohou aplikace používat k vyžádání a výměně informací. Existuje řada různých typů rozhraní, které se používají pro různé účely, jako jsou API knihoven, webová API, API třetích stran, hardwarová API a další.

Pro účely komunikace mezi backendovou a frontendovou částí aplikace "FlashCards"bude použito webové API, k němuž se přistupuje pomocí protokolu HTTP[11]. Mezi nejpopulárnější způsoby implementace API patří REST, SOAP, RPC [27], které budou podrobně analyzovány v následujících podkapitolách.

4.3.1 REST

REST, neboli REpresentational State Transfer, je architektonický styl a soubor principů pro tvorbu škálovatelného, lehkého a snadno použitelného API. Rozhraní, které splňuje následující principy, může být nazváno RESTful API [27] [28]:

1. **Klient-Server**: komunikace musí mít podobu požadavek klienta a následně odpověď serveru. Servery nemohou žádat a klienti nemohou odpovídat.
2. **Bezestavovost**: Server neukládá žádná data z klientských požadavků a nepamatuje si nic z minulých komunikací.

[11] HyperText Transfer Protocol

3. **Ukládání do mezipamětí (cache)**: Pokud je to možné odpověď na požadavek klienta je nutné ukádát do mezipamětí.

4. **Jednotné rozhraní**: všechny požadavky na stejný prostředek by měly vypadat stejně a používat HTTP protokol bez ohledu na to, odkud požadavek přichází, aby byla zajištěna kompatibilita mezi libovolným klientem a libovolným serverem.

5. **Vícevrstvý systém**: rozhraní musí být navrženo tak, aby klient ani server nepoznali, zda komunikují s koncovou aplikací, nebo s prostředníkem.

6. **Kód na vyžádání (volitelný)**: rozhraní v některých případech mohou v odpovědi obsahovat také spustitelný kód. V těchto případech by se kód měl spouštět pouze na vyžádání.

Hlavní výhodou REST API spočívá v tom, že klient a server jsou od sebe zcela odděleni. To umožňuje vytvořit abstraktní vrstvy, které pomáhají zachovat flexibilitu i při růstu a vývoji systému.

4.3.2 SOAP

SOAP (Simple Object Access Protocol) je protokol pro přenos dat, který se použít k vytváření API. Protokol SOAP je standardizován konsorciem W3C (World Wide Web Consortium) a k přenosu dat využívá formát XML [29].

SOAP striktně definuje, jak mají být zprávy přenášeny a co mají obsahovat. Díky tomu je SOAP API bezpečnější než REST API, ačkoli kvůli přísným pokynům jsou také náročnější na kód a obecně se hůře implementují. Z tohoto důvodu se SOAP často implementuje pro interní přenosy dat, které vyžadují vysoké zabezpečení. Další výhodou protokolu SOAP však je, že funguje přes jakýkoli komunikační protokol, nejen přes HTTP [27].

4.3.3 RPC

Protokol RPC (Remote Procedural Call) na rozdíl od REST a SOAP, které usnadňují přenos dat, spouští skripty a procedury na serveru.

Rozhraní může ve svých voláních používat formát buď JSON (protokol JSON-RPC), nebo formát XML (protokol XML-RPC). XML je bezpečnější a vstřícnější než JSON, ale jinak jsou si tyto dva protokoly podobné. I když je protokol RPC přísný, jedná se o poměrně jednoduchý a snadný způsob, jak spouštět kód ve vzdálených sítích [27].

RPC API mají omezené zabezpečení a možnosti, a proto se často používají spíše REST API a SOAP API. Lze je však použít pro interní systémy k provádění základních procesních požadavků, zejména mnoha najednou [27].

Na základě porovnání bylo zjištěno, že REST API nabízí oddělení klienta a serveru, což umožňuje vytvoření abstraktních vrstev a zachování flexibility systému i při jeho růstu a vývoji. REST API také využívá standardní komunikační protokol HTTP a formáty dat jako JSON nebo XML, což usnadňuje jeho použití.

SOAP je sice bezpečnější, ale kvůli svým striktním pravidlům je méně flexibilní a náročnější na kód. RPC může být jednoduchý, ale má omezené zabezpečení a možnosti.

Z tohoto důvodu bylo rozhodnuto, že pro komunikaci mezi backendovou a frontendovou částí webové aplikace "FlashCards"bude použito REST API.

4.4 Shrnutí

V rámci výběru technologií pro webovou aplikaci "FlashCards"bylo rozhodnuto používat následující:

- **Databaze**: byla zvolena relační databáze s použitím DBMS PostgreSQL. Tato volba byla provedena s ohledem na potřebu zajištění konzistence a bezpečnosti dat, a zároveň s ohledem na funkcionality a zkušenosti s tímto systémem.
- **Backend framework**: pro backendovou část aplikace byl vybrán Spring Boot, což je Java framework poskytující robustní a škálovatelné řešení pro vývoj aplikací s minimální konfigurací. Tato volba byla založena na schopnosti Spring Bootu poskytnout komplexní infrastrukturní podporu a integraci s dalšími projekty z ekosystému Spring, což značně usnadní vývoj komplexních aplikací.
- **Frontend framework**: pro frontendovou část aplikace byl zvolen React spolu s Typescript. React je flexibilní a výkonný nástroj pro vytváření interaktivních uživatelských rozhraní, který poskytuje širokou škálu funkcí a nástrojů pro efektivní vývoj. Použití Typescriptu přidává kódovou kontrolu a statickou analýzu, což výrazně zlepšuje čitelnost, údržbu a bezpečnost kódu.
- **Komunikace**: pro komunikaci mezi backendem a frontendem bude použito REST API. REST je architektonický styl a soubor principů

pro tvorbu škálovatelného, lehkého a snadno použitelného API. Využívá standardní komunikační protokol HTTP a jednoduché formáty dat, což usnadňuje jeho použití a integraci v rámci aplikace.

Tato sada technologií byla vybrána na základě požadavků projektu, analýzy, zkušenosti a porovnání nejpopulárnějších možností v každé části aplikace.

Kapitola 5
Návrh

Tato kapitola se bude věnovat návrhu webové aplikace "FlashCards". V následujících sekcích budou popsány architektura aplikace, logika intervalového opakování a důležité API endpointy[1]. Kromě toho zde budou prezentovány klíčové diagramy a pohledy uživatelského rozhraní.

5.1 Architektura

Softwarová architektura je popis komponent softwarového systému a vztahů mezi nimi [13].

Mikroservisní architektura je architektonický styl, který umožňuje rozdělení rozsáhlého softwarového systému na menší, autonomní částí - mikroslužby, z nichž každá má vlastní oblast odpovědnosti. Díky tomuto rozdělení je možné každou mikroslužbu vyvíjet, nasazovat a škálovat nezávisle na ostatních. Další výhodou je, že každá mikroslužba má vlastní datový model a rozhraní, což zjednodušuje správu a umožňuje týmům používat různé technologie.

Třívrstvá architektura je typ klient-serverové architektury, která organizuje aplikace do tří vrstvy:

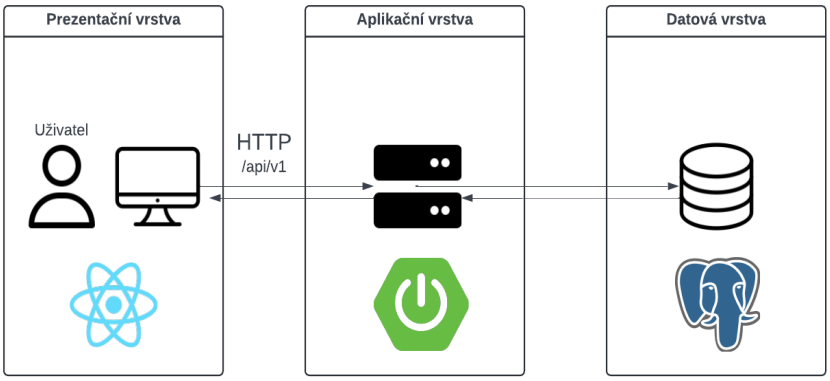
1. **Prezentační vrstva**: zahrnuje uživatelské rozhraní webové stránky nebo mobilní aplikace a veškerou logiku, která je potřebná pro zobrazování dat a přijímání uživatelských vstupů.

2. **Aplikační vrstva**: zpracovává požadavky uživatele z prezentační vrstvy a provádí operace na datech, jako je validace, výpočty nebo přístup k databázi.

[1] neboli koncové body, konkrétní místo v API, které přijímá požadavky a odesílá odpovědi.

3. **Datová vrstva**: obsahuje databázi nebo jiný systém pro ukládání a získávání dat.

Každá z těchto vrstev může být implementována na jednom nebo na více serverech. Obrázek 5.1 ukazuje, jak by mohla vypadat třívrstvá architektura.



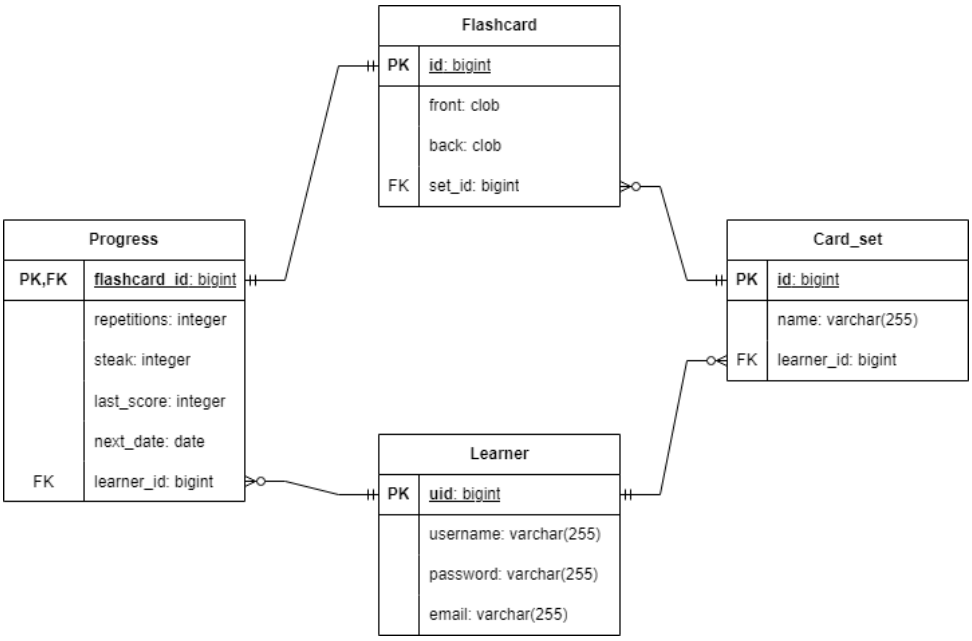
Obrázek 5.1: Třívrstvá architektura.

Pro webovou aplikaci "FlashCards"bude použita třívrstvá architektura místo mikroservisní z několika důvodů. Třívrstvá architektura poskytuje jednodušší implementace pro nevelkou aplikaci jako "FlashCards". Mikroservisní architektura by mohla být příliš komplexní pro tento typ aplikace a zavést zbytečnou složitost. Dalším důvodem je rychlost vývoje: třívrstvá architektura umožňuje rychlejší vývoj a nasazení aplikace díky menší složitosti.

5.2 Relační model

Relační model popisuje data v databázi, včetně toho, jak jsou data uložena v tabulkách a jaké vztahy existují mezi těmito tabulkami, což umožňuje jasně specifikovat, jaké informace budou uloženy v databázi a jak budou propojeny.

Na základě požadavků a případů užití, popsaných v kapitole 3, byl vytvořen relační model pro webovou aplikace "FlashCards", který je zobrazen na obrazku 5.2. Tento model poslouží jako základ pro vytvoření tabulek v databázi, které budou následně namapovány pomocí ORM.



Obrázek 5.2: Relační diagram aplikace "FlashCards".

5.3 Koncové body

API endpoint je koncový bod, obvykle URL adresa, ve webovém API, který definuje specifickou operaci nebo zdroj, ke kterému může klient přistupovat prostřednictvím HTTP protokolu.

Na základě požadavků a případů užití, popsaných v kapitole 3, byly navřene koncové body, které jsou popsány v tabulce 5.1.

"(se stránkováním)"v tabulce 5.1 znamená, že koncové body budou podporovat parametry požadavku *?page={n}* a *?size={n}*, kde *n* je celé číslo, pro stránkování. Výchozí nastavení těchto parametrů je *?page={0}&size={10}*. Stránkování umožňuje serveru poskytovat klientům pouze ty data, která jsou aktuálně potřebná pro zobrazení dané stránky. To snižuje zátěž na server a zlepšuje výkon aplikace, zejména při práci s velkými datovými sadami.

5.4 Sekvenční diagramy

Sekvenční diagram umožňuje vizualizovat, jak objekty komunikují mezi sebou a jakým způsobem probíhají jednotlivé akce v čase.

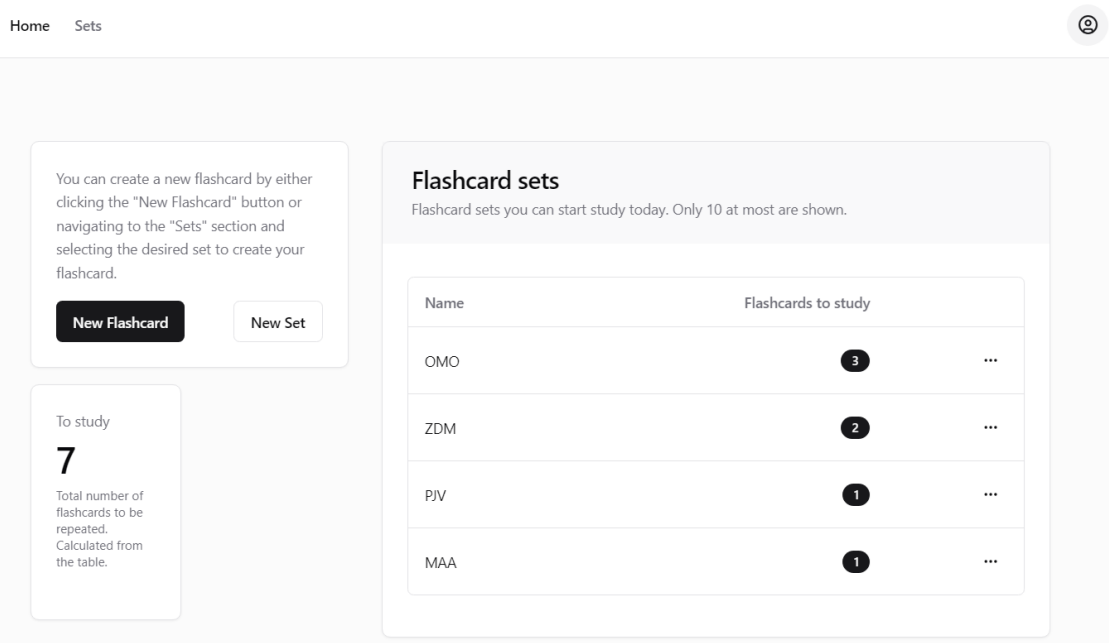
Při návrhu webové aplikace "FlashCards"byly vytvořeny následující sekvenční diagramy:

- **Tvorba flash karet**, obrázek 5.6: pokud uživatel chce vytvořit flash kartu a nemá žádné sady, nejprve musí vytvořit sadu flash karet a teprve poté může pokračovat vytvářením samotných flash karet.
- **Režim Study**, obrázek 5.7: po spuštění režimu Study se uživateli zobrazí pouze ty flash karty, u kterých následující datum opakování odpovídá aktuálnímu datu. Uživatel pak bude odpovídat se skóre, přičemž aktualizace pokroku probíhá asynchronně.
- **Režim Cram**, obrázek 5.8: po spuštění režimu Cram se uživateli zobrazí všechny flash karty z dané sady.

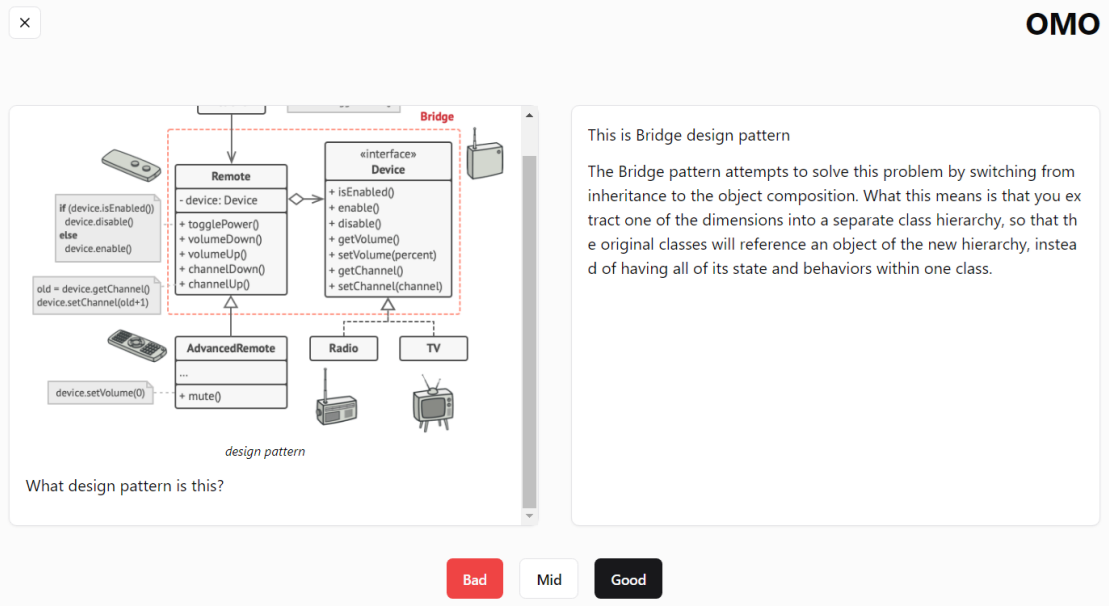
5.5 Uživatelské rozhraní

V rámci návrhu webové aplikace "FlashCards"byly vytvořeny návrhy důležitých stránek v online nástroji Figma pomocí hotových komponent shadcn/ui [31]:

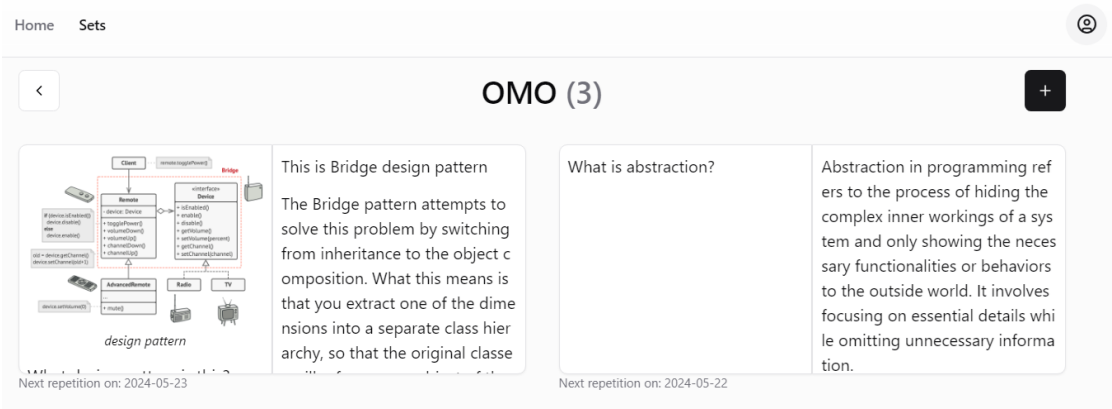
- Domovská stránka obsahuje statistiku uživatele, seznam sad flash karet, které je možné opakovat v aktuálním datu, a tlačítko "New Flashcard"pro vytvoření nové flash karty, "New Set"pro vytvoření nové sady. (viz obrázek 5.3)
- Stránka režimu *Study* zahrnuje tlačítka se skóre "Good", "Mid"a "Bad", která jsou potřebná pro intervalové opakování. (viz obrázek 5.4)
- Stránka sady flash karet, na které jsou zobrazeny flash karty, patřící do aktuální sady. (viz obrázek 5.5)



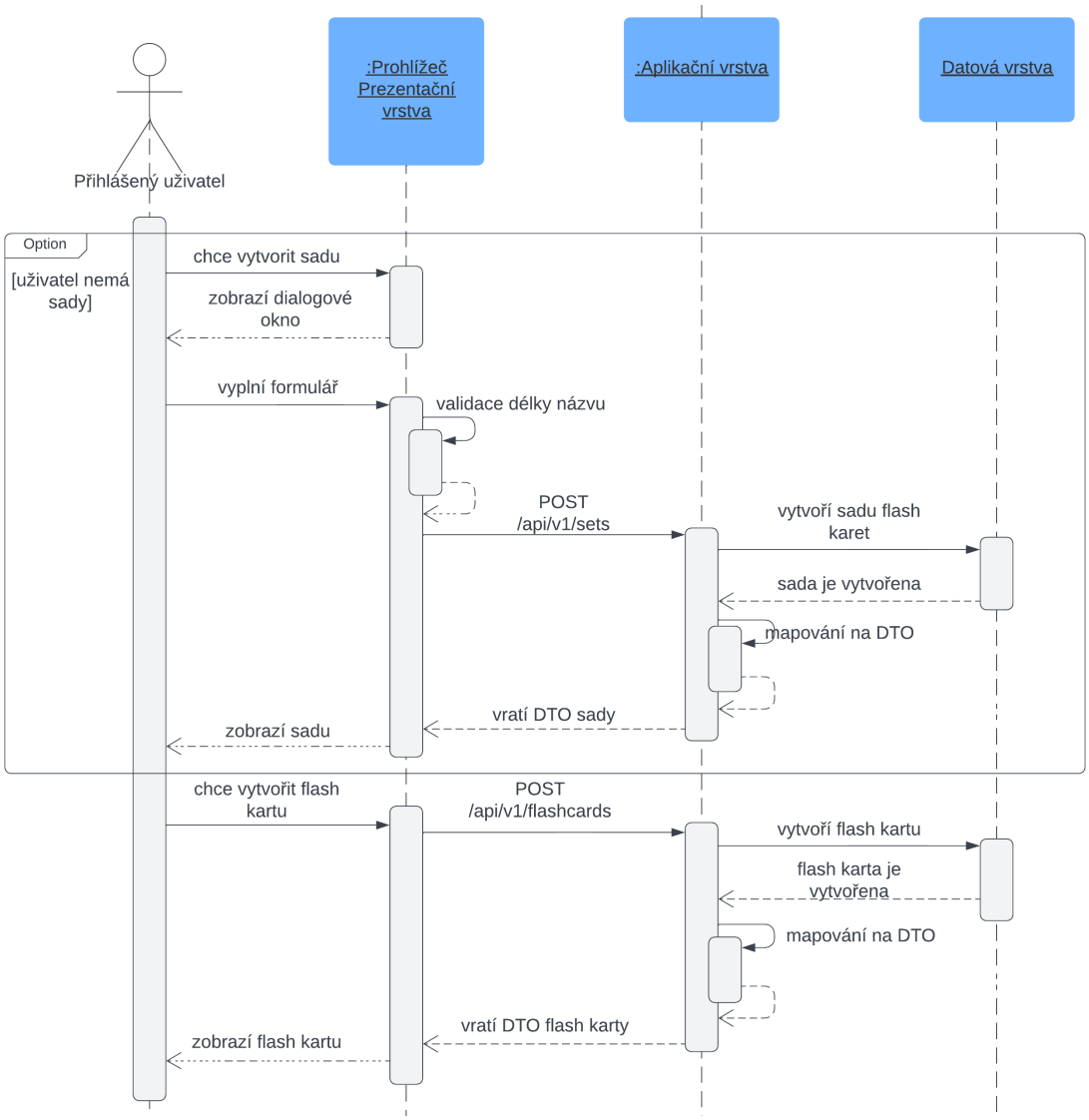
Obrázek 5.3: Návrh domovské stránky aplikace "FlashCards".



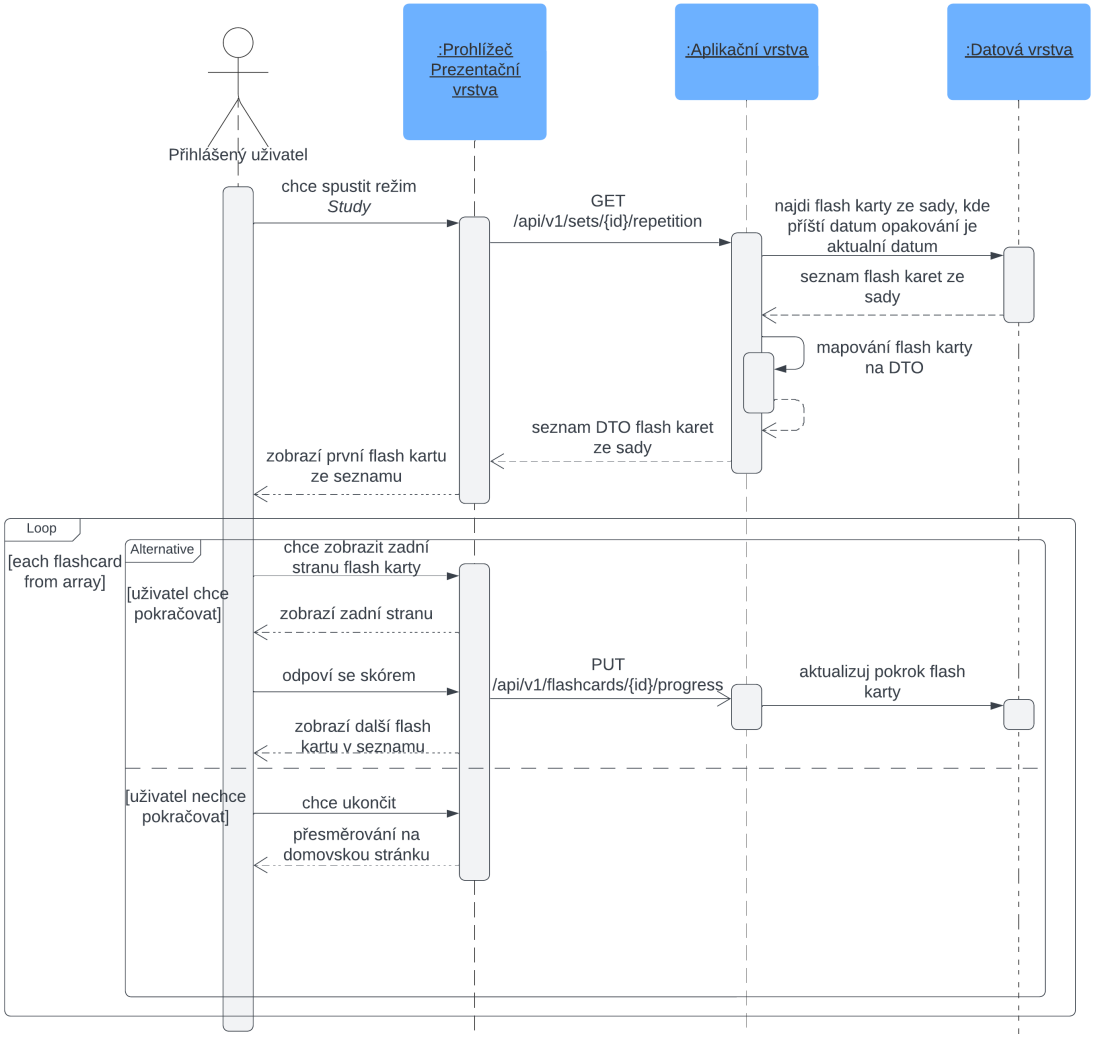
Obrázek 5.4: Návrh stánky režimu *Study* aplikace "FlashCards".



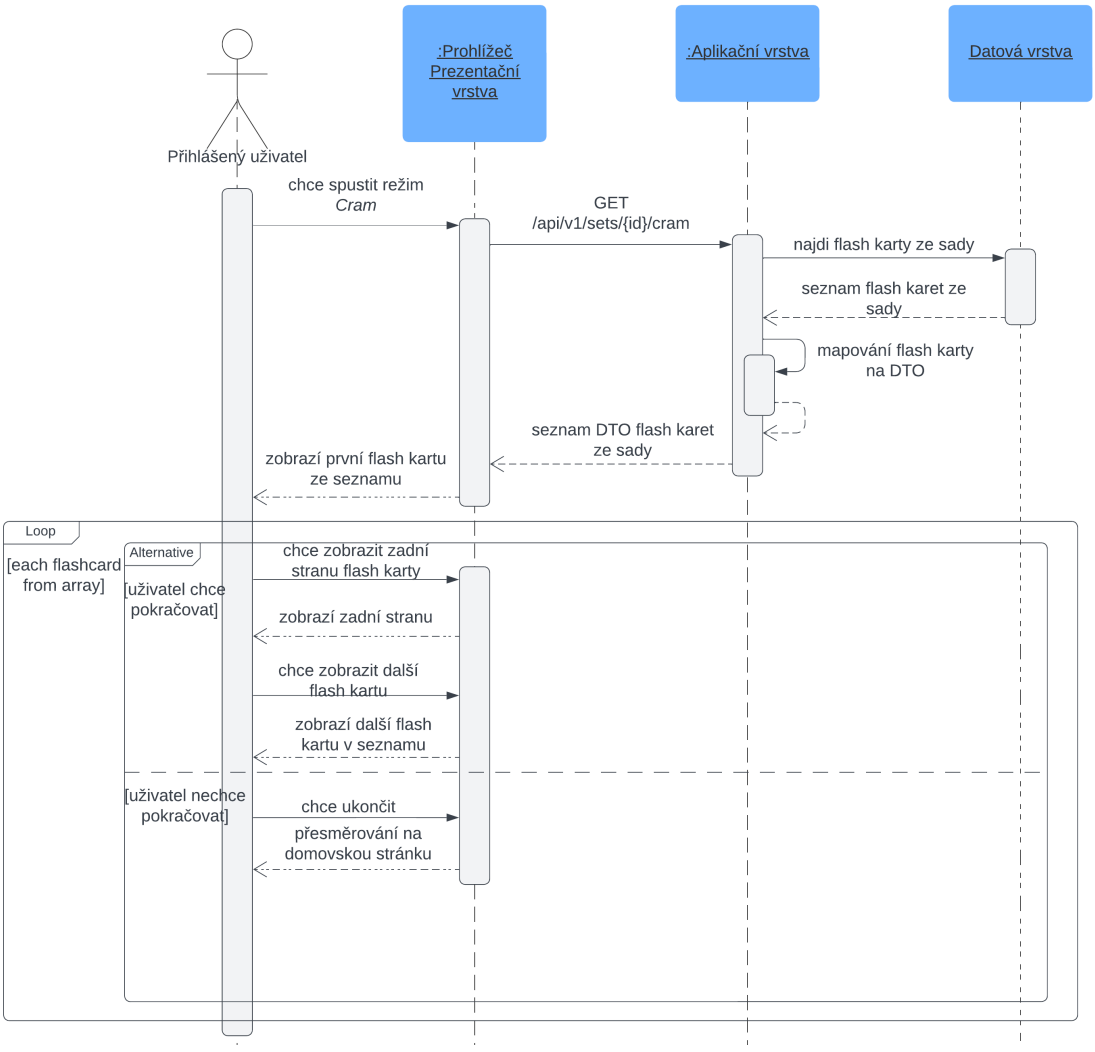
Obrázek 5.5: Návrh stránky sady flash karet aplikace "FlashCards".



Obrázek 5.6: Sekvenční diagram tvorby flash karet.



Obrázek 5.7: Sekvenční diagram režimu *Study*.



Obrázek 5.8: Sekvenční diagram režimu *Cram*.

HTTP method	**URL. /api/v1**	**Popis**
/auth		
POST	/register	Koncový bod pro registrace. Vytváří profil uživatele.
POST	/authenticate	Koncový bod pro autentizace uživatele.
/sets		
POST	/	Vytváří sadu flash karet.
GET	/	Vrátí všechny sady flash karet (se stránkováním).
GET	/rep	Vrátí sady, obsahující flash karty, které je potřeba opakovat (se stránkováním).
GET	/all	Vrátí názvy všech sad flash karet.
GET	/{id}	Vrátí sadu flash karet podle id.
GET	/{id}/flashcards	Vrátí všechny flash karty ze sady podle id (se stránkováním).
GET	/{id}/repetition	Vrátí všechny flash karty, které je potřeba opakovat, ze sady podle id.
GET	/{id}/cram	Vrátí všechny flash karty ze sady podle id.
PATCH	/{id}	Aktualizuje sadu flash karet podle id.
DELETE	/{id}	Smaže sadu flash karet podle id.
/flashcards		
POST	/	Vytváří flash kartu.
GET	/{id}	Vrátí flash kartu podle id.
PATCH	/{id}	Aktualizuje flash kartu podle id.
DELETE	/{id}	Smaže flash kartu podle id.
GET	/{id}/progress	Vrátí pokrok flash karty podle id.
PUT	/{id}/progress	Aktualizuje pokrok flash karty podle id.
/learners		
GET	/	Vrátí informace o přihlášeném uživateli.
GET	/stats	Vrátí statistiku přihlášeného uživatele.

Tabulka 5.1: Koncové body aplikace "FlashCards".

5.6 Shrnutí

V této kapitole byla navržena architektura webové aplikace "FlashCards"- tří-vrstvá architektura. Dále byl vytvořen relační model pro databázi, definovány koncové body API pro komunikaci mezi frontendem a backendem, navrženo uživatelské rozhraní a sekvenční diagramy pro procesy tvorby flash karet, režimu *Study* a *Cram*, které detailně popisují interakce mezi objekty v čase.

Kapitola 6

Implementace

Tato kapitola se zaměřuje na popis realizace webové aplikace "FlashCards"a použitých nástrojů a knihoven. Rovněž podrobněji popisuje fungování autentizace a intervalového opakování.

6.1 Intervalové opakování

Webová aplikace "FlashCards"bude podporovat dva režimy učení: *cram* a *study*. V režimu *cram* budou všechny flash karty z dané sady zobrazovány v pořadí, ve kterém byly přidány. Režim *study* pak představuje učení s intervalovým opakováním.

Pro realizaci intervalového opakování v aplikaci "FlashCards"byla zavedena entita *Progress* (viz obrázek 5.2). Dále je zaveden atribut *streak*, který ukazuje, kolikrát za sebou uživatel odpověděl na otázku se skórem *Good*. Tento atribut bude klíčovým pro implementaci následujících výpočtů:

- Pokud uživatel odpověděl na otázku z flash karty se skórem *Good*, pak se k aktuálnímu datu přičte číslo (dny), které je na pozici *streak* v Fibonacciho posloupnosti (číslo 0 je vyloučeno). Příklady vypočtu: *streak = 1, dny = 1; streak = 2, dny = 1; streak = 3, dny = 2; streak = 4, dny = 5* atd. Maximální číslo dnů ke přičtení je 34.
- Pokud uživatel odpověděl na otázku z flash karty se skórem *Mid*, pak se *streak* podělí 2 a toto číslo přičte k aktuálnímu datu.
- Pokud uživatel odpověděl na otázku z flash karty se skórem *Bad*, pak příští datum opakování je aktuální datum.

Protože Leitnerův systém nestanoví konkrétní intervaly opakování, byla využita Fibonacciho posloupnost pro výpočet intervalů.

6.2 Bezpečnost

Autentizace je proces ověřování identity uživatele a udělení přístupu k chráněným zdrojům.

Ve webové aplikace "FlashCards"autentizace bude realizována pomocí JWT. Uživatelé se budou přihlašovat pomocí svého uživatelského jména a hesla.

6.2.1 JSON Web Token

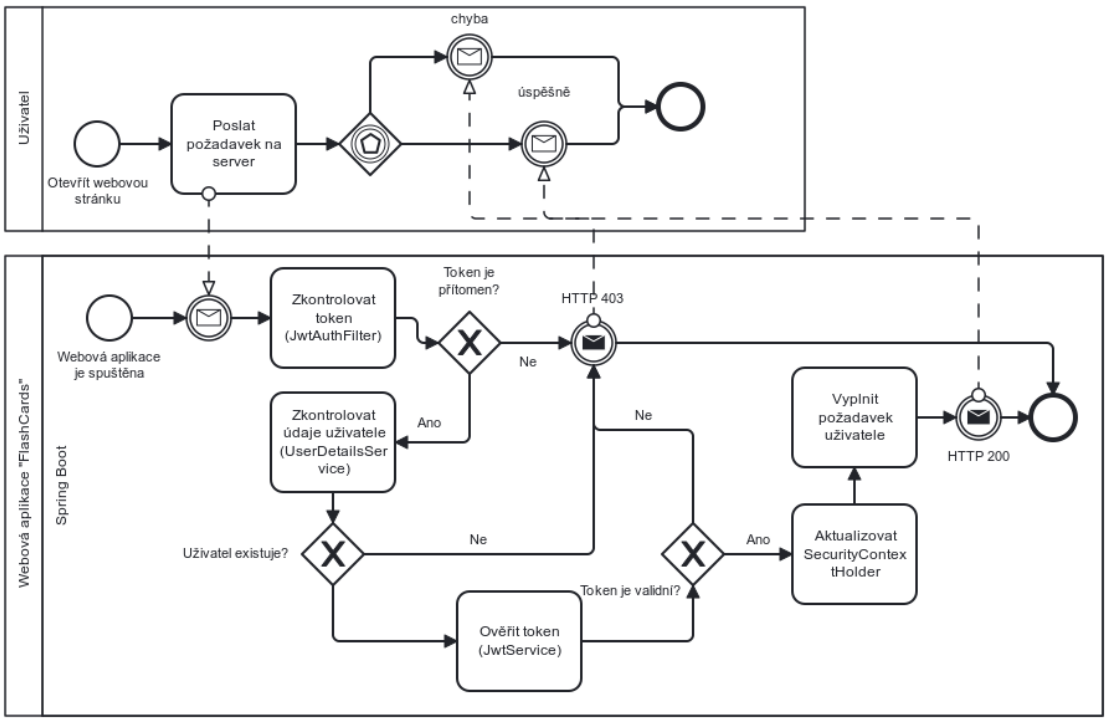
JSON Web Token (JWT) poskytuje kompaktní a samostatný způsob bezpečného přenosu informací mezi stranami ve formátu JSON objektu. Tento token může být podepsán pomocí tajného klíče (pomocí algoritmu HMAC) nebo pomocí páru veřejného a soukromého klíče (pomocí algoritmů RSA nebo ECDSA). JWT se skládá ze tří částí oddělených tečkami [42]:

- **header**: obsahuje informace o typu tokenu a použitém algoritmu pro podepisování dat.
- **payload**: obsahuje užitečné informace, které jsou přenášeny v tokenu. V případě webové aplikace "FlashCards"jsou to uživatelské jméno a heslo. Tato část také může obsahovat tzv. "claims", která udávají nároky spojené s tokenem, jako je platnost, vydavatel a další metadata.
- **signature**: podepsaná část tokenu obsahující kryptografický podpis hlavičky a těla. Tento podpis se vytváří pomocí HMAC, RSA nebo ECDSA.

6.2.2 Proces autentizace

Na obrázku 6.1 je zobrazen zjednodušený proces autentizace, který byl vytvořen pomocí BPMN (Business Process Model and Notation) procesního diagramu [44].

Proces začíná tím, že uživatel odešle požadavek na server. Tento požadavek prochází několika filtry, z nichž jedním je implementovaný filtr *JwtAuthFilter*, který ověřuje token. Filtr nejprve zjišťuje, zda v hlavičce požadavku existuje klíč "Authorization". Pokud klíč neexistuje nebo pokud jeho hodnota nezačíná slovem "Bearer", filtr předá požadavek dalšímu filtru a ukončí svou činnost, což znamená, že není k dispozici žádný JWT token k analýze a použití. Pokud hlavička "Authorization"existuje a její hodnota začíná slovem "Bearer", filtr extrahuje uživatelské jméno z tokenu. Toto uživatelské jméno je následně předáno do *UserDetailsService*, který ověřuje, zda uživatel s tímto uživatelským



Obrázek 6.1: Proces autentizace.

jménem existuje v databázi. Pokud uživatel neexistuje, server vrátí chybu 403, což znamená, že přístup k požadovanému prostředku je zakázán. Pokud uživatel existuje, filtr zkontroluje platnost tokenu. Pokud je token platný, filtr aktualizuje *SecurityContextHolder* s informacemi o uživateli, čímž autentizuje požadavek. Poté je požadavek předán k zpracování [45].

6.3 Backend

Tato podkapitola se zaměřuje na popis struktury, buildovacího nástroje a důležitých závislostí, které byly použité pro realizace backend části aplikace, která je napsána v Javě s využitím frameworku Spring Boot.

6.3.1 Buildovací nástroj

Buildovací nástroje jsou klíčové pro správu závislostí, sestavování, testování kódu a vytváření balíčků pro nasazení aplikací. Dva nejpopulárnější nástroje

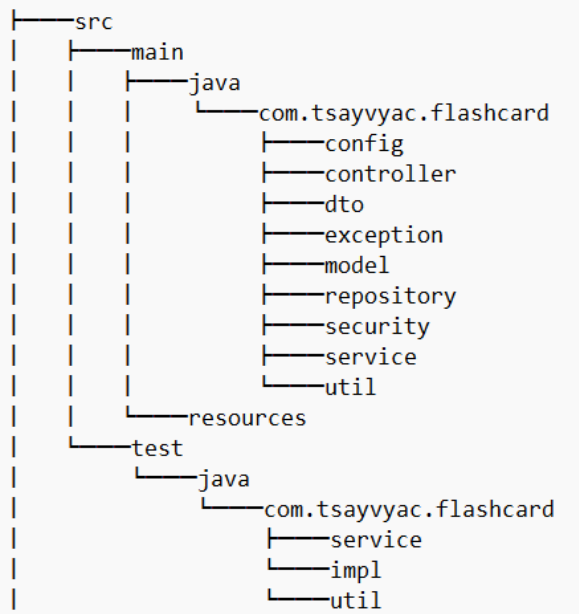
pro Spring Boot projekty jsou Gradle a Maven.

Konfigurace v Gradle je obecně kratší a přehlednější než v Maven. Kromě toho je Gradle mnohem flexibilnější pro sestavování projektu a je obecně považován za výkonnější. Podle měření je Gradle až o 60% rychlejší než Maven[31].

Nicméně, i když Gradle nabízí větší flexibilitu a vyšší výkon, pro tento projekt byl zvolen Maven kvůli zkušenostem s tímto nástrojem.

6.3.2 Popis struktury

Na obrázku 6.2 je znázorněna struktura složek projektu, která bude podrobněji popsána níže.



Obrázek 6.2: Struktura složek projektu.

- **config**: obsahuje konfigurační třídy, ve kterých se konfigurují nastavení aplikace.
- **controller**: obsahuje třídy, které zpracovávají příchozí HTTP požadavky a definují koncové body API.
- **dto**: obsahuje DTO[1]. DTO je návrhový vzor používaný k přenosu dat

[1] Data Transfer Object

mezi různými vrstvami nebo komponentami aplikace. Hlavním účelem DTO je zapouzdřit data a poskytnout jednoduchou datovou strukturu, kterou lze snadno předávat v rámci aplikace. používají se k přenosu dat mezi frontendem a backendem webové aplikace.

- **exception**: obsahuje výjimky pro aplikace a třídu GlobalExceptionHandler, která je označena anotací @ControllerAdvice. Tato třída obsahuje metody, které jsou označeny anotací @ExceptionHandler. Tím je umožněno definovat metodu, která se zavolá vždy, když je v aplikaci vyhozena konkrétní výjimka.
- **model**: obsahuje datové modely, které reprezentují strukturu a chování aplikační domény. Tyto třídy jsou mapovány na databázové tabulky a definují vlastnosti a vztahy dat.
- **repository**: obsahuje třídy odpovědné za přístup k datům a perzistenci. Tyto třídy komunikují s databází a poskytují metody pro CRUD[2] operace.
- **security**: obsahuje třídu JwtAuthFilter pro bezpečnostní účely, které byly popsány v podkapitole 6.2.
- **service**: obsahuje třídy, které implementují business logiku. Kontroléry používají tyto služby k provádění operací s daty.
- **util**: obsahuje pomocné třídy např. třídu Mapper, která se používá k mapování datových modelů na DTO nebo naopak, nebo třídu Constant, do které se ukládá konstanty pro různé účely.
- **test**: obsahuje jednotkové testy, které je popsány v kapitole 7.

6.3.3 Důležité závislosti

Spring Boot poskytuje "starter"závislosti, které navrženy tak, aby zjednodušily konfiguraci projektu tím, že automaticky přidávají nezbytné závislosti. To umožňuje rychle začít s vývojem aplikace, aniž by ručně vyhledávat a konfigurovat jednotlivé závislosti [40].

- **spring-boot-starter-data-jpa**: poskytuje knihovny a konfigurace pro použití Java Persistence API (JPA). JPA umožňuje snadnou integraci s relačními databázemi a práci s daty pomocí objektově-relačního mapování.

[2]Create (zápis), Read (čtení), Update (aktualizace), Delete (mazání)

- **spring-boot-starter-web**: obsahuje knihovny a konfigurace pro vytvoření webové. Zahrnuje nástroje pro vytváření kontrolerů, obsluhu HTTP požadavků a další funkce pro vývoj webových aplikací.
- **spring-boot-starter-security**: poskytuje knihovny a konfigurace pro zabezpečení aplikace. Obsahuje nástroje pro autentizaci, autorizaci a další bezpečnostní funkce, které jsou nezbytné pro ochranu aplikace před neoprávněným přístupem.
- **spring-boot-starter-test**: obsahuje knihovny a konfigurace pro psaní a spouštění testů. Zahrnuje nástroje pro jednotkové testování, integrační testování a automatizované testování REST API pomocí nástrojů jako JUnit, Mockito a Spring Test.

Ostatní závislosti:

- **Lombok**: knihovna založená na anotacích, která umožňuje redukovat boilerplate kód.
- **PostgreSQL JDBC Driver**: umožňuje se připojit k databázi PostgreSQL.

6.3.4 Databáze

Databázové tabulky byly realizovány podle relačního modelu (obrázek 5.2). Poté byly údaje pro přístup k databázi přidány do konfiguračního souboru profilu *pg*, který se nachází v *src/main/resources/application-pg.yml*.

6.4 Frontend

Frontend část webové aplikace "FlashCards"je vytvořena v TypeScriptu s využitím knihovny React a je plně responzivní[3]. Pro sestavení frontendu a provoz vývojového serveru je použit nástroj Vite, který podporuje Hot Module Replacement (HMR). To umožňuje okamžité zobrazení změn v kódu přímo v prohlížeči. Pro produkční sestavení Vite využívá Rollup, který zajišťuje optimalizovaný výstup a snadné nasazení [32]. Pro rychlý vývoj byly také použity komponenty shadcn/ui, které jsou snadno použitelné a přizpůsobitelné [33], a CSS framework Tailwind CSS, který předdefinuje třídy stylů [34].

[3]přístup, podle kterého by uživatelské rozhraní mělo reagovat na velikost obrazovky, platformu a orientaci.

6.4.1 Použité knihovny a nástroje

Pro realizace pokročilejších funkcí frontendu byly použité knihovny. Důležité popsány níže:

- **react-router-dom**: knihovna pro zpracování směrování v React aplikacích. Umožňuje definovat různé routy a vykreslovat různé komponenty na základě adresy URL. Používá se při vytváření SPA aplikace s dynamickou navigací [35].
- **axios**: knihovna sloužící k provádění požadavků. Poskytuje jednoduché a intuitivní rozhraní pro provádění asynchronních operací, jako je načítání dat ze serveru nebo odesílání dat. Axios podporuje funkce, jako jsou interceptory, transformace požadavků/odpovědí a automatické parsování JSON [36].
- **editor-js**: block-based textový editor, který nabízí řadu funkcí, které uživatelům pomáhají efektivně vytvářet a formátovat obsah [39].
- **yup**: knihovna, která se používá pro validaci formulářů. Umožňuje definovat validační schémata pomocí jednoduchého rozhraní. Schémata lze použít k validaci uživatelských vstupů a zajistit, aby data odeslaná prostřednictvím formulářů splňovala zadaná kritéria, jako jsou povinná pole, datové typy, omezení délky a další [43].
- **eslint**: nástroj, který pomáhá udržovat kvalitu a konzistenci kódu tím, že upozorňuje na potenciální chyby, prosazuje standardy kódování a identifikuje oblasti, které je třeba zlepšit [37].
- **prettier**: formátovač kódu, který automaticky formátuje kód tak, aby byla zajištěna jeho konzistence a čitelnost [38].

6.5 Shrnutí

V této kapitole byla podrobně popsána implementace webové aplikace "FlashCards", včetně použitých nástrojů a knihoven. Dále byly detailně popsány procesy realizované autentizace pomocí JWT a intervalového opakování.

Kapitola 7
Testování

Tato kapitola popisuje proces testování projektu. Při testování webové aplikace "FlashCards"byly provedeny jednotkové testy důležitých služeb, testování API, a uživatelské testy, aby byla zajištěna její funkčnost a použitelnost.

7.1 Jednotkové testování

Jednotkové testy je testy, které se zaměřují na ověření správnosti jednotlivých částí kódu, typicky na úrovni metod.

Jednotkové testy, které se nachází ve složce *src/test*, byly navrženy k ověření správnosti důležitých služeb a pomocných metod aplikace a implementovány pomocí testovacích frameworků jako JUnit a Mockito.

7.2 Testování API

Během testování API, prováděného v průběhu vývoje backend části webové aplikace "FlashCards"pomocí nástroje Postman, byly odhaleny různé chyby a nesrovnalosti, jako například nesprávné HTTP status kódy, chyby v validaci dat, nesprávné odpovědi serveru a problémy s autentizací.

Nalezené chyby byly opraveny prostřednictvím úprav backendového kódu, aktualizace validací a zlepšení logiky zpracování dat. Po opravách byly koncové body znovu otestovány, aby bylo zajištěno, že byly problémy úspěšně vyřešeny a že aplikace funguje podle očekávání.

7.3 Uživatelské testování

Uživatelské testování je proces, při němž skuteční uživatelé aplikace interagují s aplikací, aby se zjistilo, jak snadno a efektivně mohou plnit své úkoly. Cílem tohoto testování je identifikovat problémy s použitelností, porozumět uživatelskému chování a získat zpětnou vazbu pro zlepšení aplikace [13].

Jelikož aplikace bude složit jako pomůcka pro výuku a přípravu ke zkouškám, pro uživatelské testování bylo vybráno 6 studentů ve věku 20-25 let. Tito uživatelé plnili úkoly podle předem připraveného scénáře a odpovídali na otázky, které byly položené na konci testování. Odkaz na testovací scénáře je uveden v příloze B.

7.3.1 Dotazník

Uživatelé odpovídali na následující otázky, které byly položené na konci uživatelského testování:

1. Jak hodnotíte přívětivost uživatelského rozhraní? Bylo to intuitivní? Byly pro Vás barvy rozhraní příjemné?
2. Setkali jste se během testovacích scénářů s technickými problémy nebo nejasnostmi? Pokud ano s jakými?
3. Bylo rychlé se naučit pracovat s funkcemi webové aplikace?
4. Bylo pro Vás užitečné používat pokročily textový editor?
5. Používali byste webovou aplikace pro přípravu ke zkouškám nebo pro jiné účely? Pokud byste použili webovou aplikace pro jiné účely, co by to mohlo být?
6. Existuje něco, co by Vám chybělo nebo co by podle Vás vyžadovalo další zlepšení?

Výsledky uživatelských testů jsou uvedeny v příloze B.

7.4 Shrnutí

V této kapitole bylo popsáno, jak projekt byl otestován během vývoje pomocí testování API, jednotkových testů a na konci vývoje pomocí uživatelských testů. Jednotkové testy zajistily správnost důležitých služeb a pomocných metod aplikace, zatímco uživatelské testy ověřily její použitelnost. Tyto testy společně přispěly k celkové stabilitě a kvalitě webové aplikace "FlashCards".

Kapitola 8
Závěr

V závěru této bakalářské práce lze říci, že webová aplikace "FlashCards"úspěšně plní svůj hlavní cíl, kterým bylo vytvoření interaktivního prostředí pro efektivní učení pomocí flash karet.

V průběhu vývoje byly na základě analýzy a porovnání existujících řešení pečlivě navrženy a implementovány klíčové funkcionality práce s flash kartami. Kromě toho, byly implementovány Leitnerův systém pro řízení intervalového opakování a zabezpečení pomocí autentizace využívající JWT token.

Testování aplikace, včetně jednotkových a uživatelských testů, hrálo klíčovou roli při ověřování správnosti implementace a zajištění uživatelské přívětivosti.

Zadání bakalářské práce lze považovat za splněné a aplikace představuje funkční a uživatelsky přivětivé prostředí pro podporu učení pomocí flash karet a intervalového opakování.

Tato bakalářská práce mi umožnila prakticky aplikovat znalosti získané během studia a výrazně rozvinout své schopnosti v oblasti webového vývoje, práce s frameworkem Spring Boot a knihovnou React. Všechny tyto poznatky a dovednosti budu moci plně využít v budoucích projektech a profesním životě.

8.1 Další rozvoj aplikace

Tato kapitola se zaměřuje na možnosti dalšího rozvoje aplikace.

Zlepšení aplikace lze dosáhnout následujícími způsoby:

1. Nasazení aplikace na server za účelem zajištění lepší dostupnosti pro uživatele.

2. Zpracování připomínek od uživatelů, které byly získány během uživatelského testování.

3. Současná verze webové aplikace "FlashCards"je plně funkční, nicméně během implementace nebyly realizovány funkční požadavky z kategorie C a W, které byly popsány v kapitole 3.3.1. Tyto požadavky budou realizovány v budoucích verzích.

4. Vzhledem k tomu, že intervaly v Leitnerovu systému mohou být individuálními pro každého uživatele, plánuje se v budoucích verzích aplikace umožnit jim nastavení vlastních intervalů. Tímto způsobem uživatelé budou moci lépe řídit své učení.

5. S přibývajícími funkcionalitami webové aplikace může být potřeba vytvořit stránku s uživatelskou příručkou, kde budou podrobně popsány všechny funkce a způsob jejich používání. To usnadní uživatelům pochopení a efektivnější používání všech funkcí aplikace.

6. Vytváření mobilní aplikace, což přinese flexibilitu a pohodlí do procesu učení.

Příloha A

Seznam zkratek

SQL Structured query language
SPA Single Page Application
HTTPS Hypertext Transfer Protocol Secure
OAuth2 Open Authorization 2
ACID Atomicity, Consistency, Isolation, Durability
API Application Programming Interface
MVC Model-View-Controller
MVT Model-View-Template
ORM Object-Relation Mapping
DRY Do not repeat yourself
URL Uniform Resource Locator
HTML HyperText Markup Language
CSS Cascading Style Sheets
JSX Javascript XML
HTTP HyperText Transfer Protocol
REST Representational State Transfer
SOAP Simple Object Access Protocol
RPC Remote Procedure Call
JWT JSON Web Token
DTO Data Transfer Object
CRUD Create, Read, Update, Delete
JPA Java Persistence API
HMR Hot Module Replacement

Příloha B

Seznam odkázů

Repozitáře:

- Backend: https://github.com/tsayvyac/flashcard-be
- Frontend : https://github.com/tsayvyac/flashcard-fe-vite

Formuláře:

- Testovací scénáře: https://forms.gle/k3iQ7r8ykPJGeJ41A

Výsledky:

- Výsledky uživatelských testů: https://bit.ly/4dNOvWI

Příloha C

Literatura

[1] KORNELL, Nate. Optimising learning using flashcards: Spacing is more effective than cramming. 19.01.2009. *Applied Cognitive Psychology*, *23*(9), 1297-1317. Dostupné z: https://doi.org/10.1002/acp.1537

[2] OWEN, Michael. *Active Recall: The Most Effective High-Yield Learning Technique* [online]. akt.13.03.2023. [vid. 25-11-2023] Dostupné z: https://www.osmosis.org/blog/2022/02/21/active-recall-the-most-effective-highyield-learning-technique.

[3] DUNLOSKY, John; RAWSON, Katherine A.; MARCH, Elizabeth J.; NATHAN, Mitchell J.; WILLINGHAM, Daniel T. Improving Students' Learning With Effective Learning Techniques: Promising Directions From Cognitive and Educational Psychology. 08.01.2013. *Psychological Science in the Public Interest*, *14*(1), 4-58. Dostupné z: https://doi.org/10.1177/1529100612453266.

[4] KANG, Sean H.K. Spaced Repetition Promotes Efficient and Effective Learning: Policy Implications for Instruction. 13.01.2016. *Policy Insights from the Behavioral and Brain Sciences*, *3*(1), 12-19. Dostupné z: https://doi.org/10.1177/2372732215624708.

[5] COLLINS, Stella. *How to use spaced repetition for learning* [online]. 03.03.2023. [vid. 30-11-2023]. Dostupné z: https://www.stellarlabs.io/resources/spaced-repetition.

[6] SANDER, Tamm. *The Leitner System: What It Is, How It Works* [online]. 22.01.2023. [vid. 20.02.2024]. Dostupné z: https://e-student.org/leitner-system.

[7] GROMADA, Jess. *The Leitner System: How Does it Work?* [online]. 20.01.2021. [vid. 04-11-2023]. Dostupné z: https://www.mindedge.com/learning-science/the-leitner-system-how-does-it-work/.

[8] Recall Labs. *Getting started with Active Recall* [online]. 2022. [vid. 09-11-2023]. Dostupné z: https://activerecall.com/getting-started.

[9] POUSTKA, Daniel. *Webová výuková aplikace Flashcards*. 20.05.2022. Bakalářská práce. České vysoké učení technické v Praze. Dostupné z: http://hdl.handle.net/10467/100932.

[10] PATTERSON, Ransom. *These Flashcard Apps Will Help You Study Better in 2023* [online]. 15.12.2022. [vid. 18-12-2023]. Dostupné z: https://collegeinfogeek.com/flashcard-apps.

[11] DUFFY, Jill a VATU, Gabriela. *Quizlet Review* [online]. 18.11.2023. [vid. 19-12-2023]. Dostupné z: https://www.pcmag.com/reviews/quizlet.

[12] BRUSH, Kate. *MoSCoW method* [online]. 2023. [vid. 23-12-2023]. Dostupné z: https://www.techtarget.com/searchsoftwarequality/definition/MoSCoW-method.

[13] PRESSMAN, Roger S. *Software Engineering: A Practitioner's Approach. Seventh Edition*. New York, McGraw-Hill, 2010.

[14] "Cram". *Dictionary.com* [online]. 2023. [vid.23-12-2023]. Dostupné z: https://www.dictionary.com/browse/cram.

[15] SKOPAL, Tomáš; HOLUBOVÁ, Irena; SVOBODA, Martin. *Databázové transakce* [online]. 23.03.2021. [vid. 08-02-2024]. Databázové systémy. Dostupné z: https://cw.fel.cvut.cz/b212/_media/courses/b0b36dbs/lecture-06-database-transactions.pdf

[16] *Mongodb.com. What is NoSQL?* [online]. 2024. [vid. 08-02-2024]. Dostupné z: https://www.mongodb.com/resources/basics/databases/nosql-explained.

[17] *Statisticsanddata.org. Most Popular Backend Frameworks – 2012/2023* [online]. 2024. [vid. 10-02-2024]. Dostupné z: https://statisticsanddata.org/data/most-popular-backend-frameworks-2012-2023/.

[18] *Laravel.com*. Oficiální webová stránka [online]. 2024. [vid. 10-02-2024]. Dostupné z: https://laravel.com/.

[19] Guru Staff. *The Pros and Cons of Laravel* [online]. 01.12.2021. [vid. 10-02-2024]. Dostupné z: https://www.guru.com/blog/the-pros-and-cons-of-laravel/.

[20] PROTASIEWICZ, Jakub. *What is Django used for?* [online]. akt. 12.01.2024. [vid. 15-02-2024]. Dostupné z: https://www.netguru.com/blog/why-use-django.

[21] *Djangoproject.com*. Oficiální webová stránka [online]. 2024. [vid. 15-02-2024]. Dostupné z: https://www.djangoproject.com/.

[22] TIMBÓ, Rafael. *Front End Frameworks: What They Are, and Best Options* [online]. akt. 23.02.2024. [vid. 02-03-2024]. Dostupné z: https://www.revelo.com/blog/front-end-frameworks.

[23] State of Javascript. *Front-end Frameworks* [online]. 2022. [vid. 02-03-2024]. Dostupné z: https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/.

[24] *Angular.io*. Oficiální webová stránka [online]. 2024. [vid. 10-03-2024]. Dostupné z: https://angular.io/.

[25] RANGNEKAR, Parija. *The Benefits and Downsides of Angular Development!* [online]. 05.12.2023. [vid. 10-03-2024]. Dostupné z: https://www.linkedin.com/pulse/benefits-downsides-angular-development-parija-rangnekar-ska5f/.

[26] *Vuejs.org*. Oficiální webová stránka [online]. 2024. [vid. 11-03-2024]. Dostupné z: https://vuejs.org/.

[27] JUVILER, Jamie. *4 Types of APIs All Marketers Should Know* [online]. 16.01.2023. [vid. 15-03-2024]. Dostupné z: https://blog.hubspot.com/website/types-of-apis.

[28] AUBRECHT, Petr a LEDVINKA, Martin. *HTTP, REST Web Services* [online]. 2022. [vid. 15-03-2024]. Enterprise systems. Dostupné z: https://cw.fel.cvut.cz/b221/_media/courses/b6b36ear/lectures/lecture-06-rest-s.pdf.

[29] MITRA, Nilo a ERICSSON, Yves L. *SOAP Version 1.2 Part 0: Primer (Second Edition)* [online]. 27.04.2007. [vid. 20-03-2024]. W3C. Dostupné z: https://www.w3.org/TR/soap12-part0/.

[30] SCHIRANO, Pietro. *@shadcn/ui - Design System* [online]. 2023. [vid. 25-03-2024]. Dostupné z: https://www.figma.com/community/file/1203061493325953101.

[31] GREGORY, Tom. *Maven vs. Gradle in-depth comparison* [online]. 17.12.2021. [vid. 03-04-2024]. Dostupné z: https://tomgregory.com/gradle/maven-vs-gradle-comparison/.

[32] *Vitejs.dev. Guide. Getting Started* [online]. 2024. [vid. 12-03-2024]. Dostupné z: https://vitejs.dev/guide/.

[33] *Ui.shadcn.com.* Oficiální stránka komponent shadcn/ui [online]. 2024. [vid. 12-03-2024]. Dostupné z: https://ui.shadcn.com/.

[34] *Tailwindcss.com.* Oficiální stránka frameworku Tailwind CSS [online]. 2024. [vid. 12-03-2024]. Dostupné z: https://tailwindcss.com/.

[35] *Reactrouter.com. Feature Overview* [online]. 2024. [vid. 08-04-2024]. Dostupné z: https://reactrouter.com/en/main/start/overview.

[36] *Axios-http.com. Getting Started* [online]. 2024. [vid. 08-04-2024]. Dostupné z: https://axios-http.com/docs/intro.

[37] *Eslint.org. Getting Started with ESLint* [online]. 2024. [vid. 08-04-2024]. Dostupné z: https://eslint.org/docs/latest/use/getting-started.

[38] *Prettier.io. What is Prettier?* [online]. 2024. [vid. 09-04-2024]. Dostupné z: https://prettier.io/docs/en/.

[39] *Editorjs.io.* Oficiální stránka editoru [online]. 2024. [vid. 10-04-2024]. Dostupné z: https://editorjs.io/.

[40] *Spring.io. Starters.* Oficiální dokumentace frameworku Spring [online]. 2024. [vid. 20-04-2024]. Dostupné z: https://docs.spring.io/spring-boot/docs/current/reference/html/using.html.

[41] *Vmsoftwarehouse.com. Spring Framework vs. Spring Boot – pros and cons* [online]. 2024. [vid. 20-04-2024]. Dostupné z: https://vmsoftwarehouse.com/spring-framework-vs-spring-boot-pros-and-cons.

[42] *Jwt.io. Introduction to JSON Web Tokens* [online]. 2024. [vid. 24-04-2024]. Dostupné z: https://jwt.io/introduction.

[43] Quense J. Yup. *Github repository* [online]. akt. 26.04.2024. [vid. 28-04-2024]. Dostupné z: https://github.com/jquense/yup.

[44] *Bpmn.org. Oficiální stránka BPMN* [online]. 2024. [vid. 02-05-2024]. Dostupné z: https://www.bpmn.org/.

[45] GORDADZE, Ioram. *Spring Security With JWT for REST API* [online]. 2021. [vid. 10-05-2025]. Dostupné z: https://www.toptal.com/spring/spring-security-tutorial.