FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Bakalářská práce

Výuková aplikace využívající algoritmy Spaced Repetition

Studijní program: B0613A140005AI – Aplikovaná informatika
Studijní obor: B0613A140005 – Informační technologie
Autor práce: **Jiří Halbych**
Vedoucí práce: Ing. Jan Kolaja, Ph.D.

Liberec 2024

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání bakalářské práce

Výuková aplikace využívající algoritmy Spaced Repetition

Jméno a příjmení:	**Jiří Halbych**
Osobní číslo:	M21000108
Studijní program:	B0613A140005 Informační technologie
Specializace:	Aplikovaná informatika
Zadávající katedra:	Ústav nových technologií a aplikované informatiky
Akademický rok:	2023/2024

Zásady pro vypracování:

1. Proveďte rešerši aktuálních aplikací se zaměřením na originální verzi autora Piotra Woźniaka.
2. Porovnejte Spaced Repetition algoritmy mezi jejich verzemi a typy.
3. Vyberte vhodné programovacího nástroje včetně nástrojů pro uchování a správu dat.
4. Vytvořte aplikaci, která bude splňovat následující požadavky:
   a) Umožní správcům vkládání a editaci různých druhů výukových aktivit.
   b) Výukové aktivity lze uspořádat do větších celků – balíčků.
   c) Umožní uživatelům (studentům) zpracovávat tyto aktivity.
   d) Aplikace nechť vybírá úlohy pro uživatele pomocí Spaced Repetition algoritmů.
   e) Umožní uživatelům zobrazit si statistiky svých studijních výsledků.
5. Aplikaci otestujte na reprezentativním vzorku uživatelů.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30 – 40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	čeština

Seznam odborné literatury:

[1] WOZNIAK, Piotr A. a Edward J. GORZELANCZYK. Optimization of repetition spacing in the practice of learning. Acta Neurobiologiae Experimentalis. 1994, 54, 59–62.

[2] PIMSLEUR, Paul. A Memory Schedule. Online. The Modern Language Journal. 1967, roč. 51, č. 2, s. 73-75. ISSN 00267902. Dostupné z: https://doi.org/10.1111/j.1540-4781.1967.tb06700.x. [cit. 2023-10-03].

[3] YE, Junyao; SU, Jingyong a CAO, Yilong. A Stochastic Shortest Path Algorithm for Optimizing Spaced Repetition Scheduling. Online. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. New York, NY, USA: ACM, 2022, s. 4381-4390. ISBN 9781450393850. Dostupné z: https://doi.org/10.1145/3534678.3539081. [cit. 2023-10-03].

[4] KARPICKE, J. D. a ROEDIGER, H. L. Is expanding retrieval a superior method for learning text materials? Online. Memory & Cognition. 2009, roč. 38, č. 1, s. 116-124. ISSN 0090-502X. Dostupné z: https://doi.org/10.3758/MC.38.1.116. [cit. 2023-10-03].

Vedoucí práce:	Ing. Jan Kolaja, Ph.D. Ústav nových technologií a aplikované informatiky
Datum zadání práce:	12. října 2023
Předpokládaný termín odevzdání:	14. května 2024

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Josef Chaloupka, Ph.D.
garant studijního programu

V Liberci dne 19. října 2023

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.
Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.
Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.
Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.
Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS STAG se shoduje s textem tištěné podoby práce.
Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.
Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

10. 5. 2024 Jiří Halbych

Výuková aplikace využívající algoritmy Spaced Repetition

Abstrakt

Práce se zabývá možnostmi využití algoritmů spaced repetition ve výukovém softwaru. Představuje současné dostupné algoritmy tohoto typu a příklady aplikací, které je využívají. Následně je představena struktura a detaily vývoje aplikace, jež vznikla jako hmotný důkaz tohoto využití v rámci práce. Na závěr jsou představeny příklady výsledků, jež používání této aplikace umožňuje.

Klíčová slova: Spaced repetition, paměť, interval, opakování, aplikace

Educational application using Spaced Repetition algorithms

Abstract

The the thesis describes the possible ways to apply spaced repetition in educational software. It introduces the available algorithms of this type and examples of applications that use them. Additionally it introduces the structure and details of development of an application, that was created as a proof of this concept. At the end of the thesis are showcased examples of possible results of using this application.

Keywords: Spaced repetition, memory, interval, review, application

Poděkování

Rád bych poděkoval všem, kteří přispěli ke vzniku tohoto dílka.

Obsah

1 Spaced repetiton **12**
1.1 Historie a základ . . . 12
1.2 Algoritmy . . . 14
1.2.1 SuperMemo . . . 14
1.2.2 Leitnerův systém . . . 15
1.2.3 FSRS . . . 15
1.3 Efektivita spaced repetition . . . 16
1.4 Současné využití . . . 16
1.4.1 Anki . . . 16
1.4.2 Programy SuperMemo . . . 16
1.4.3 Duolingo . . . 17
1.5 Inkrementální čtení . . . 17

2 Cíle práce **18**
2.1 Vytvoření aplikace . . . 18
2.2 Vytvoření editoru . . . 18
2.3 Propojeni editoru s aplikací . . . 18

3 Použité nástroje **19**
3.1 Flutter . . . 19
3.2 Ruby on Rails . . . 19
3.3 PostgreSQL . . . 20
3.4 Redis . . . 20
3.5 Vue.js . . . 20
3.6 Nginx . . . 20
3.7 DHU . . . 21
3.8 IDE . . . 21

4 Výukový software **22**
4.1 Algoritmus . . . 22
4.2 Struktura . . . 22

5 API **23**
5.1 Modely dat . . . 23
5.1.1 User se seznamem vah . . . 23
5.1.2 RefreshToken . . . 24

5.1.3 Course . . . 24
5.1.4 Deck . . . 24
5.1.5 ActivityCard . . . 24
5.1.6 UserCard . . . 24
5.1.7 Revlog . . . 25
5.1.8 Requirement . . . 25
5.2 Endpointy API . . . 26
5.2.1 Autentizace . . . 27
5.2.2 Uživatelské informace . . . 27
5.2.3 Získávání aktivit . . . 27
5.2.4 Výuka . . . 28
5.2.5 Úprava dat . . . 28
5.2.6 Výběr záznamů . . . 28
5.2.7 Audio . . . 29
5.3 Příklady formátů komunikace . . . 29
5.4 Use Case . . . 30
5.4.1 Výpočet intervalu opakování . . . 30
5.4.2 Výběr aktivit k opakování . . . 30

6 Editor **31**
6.1 Use Case . . . 32

7 Aplikace **34**
7.1 Use Case . . . 35
7.1.1 První spuštění . . . 35
7.1.2 Výuka . . . 36
7.1.3 Opakování . . . 37
7.2 Typy aktivit . . . 37
7.3 Android Auto . . . 38
7.4 Google Play . . . 39

8 Testování a výsledky **41**
8.1 Příklady výpočtu intervalu opakování . . . 41

Použitá literatura **45**

A Přílohy **47**
A.1 Odkazy na zdrojový kód aplikace . . . 47

Seznam tabulek

1 Tabulka intervalu opakování po dvou opakováních. 42
2 Tabulka optimistické scénáře výuky 42
3 Tabulka pesimistického scénáře výuky 43

Seznam obrázků

1 Ebbinghausova křivka zapomínání 13
2 Efekt spaced repetition na křivku zapomínání[3] 14

3 Diagram modelů dat API 26

4 Prostředí editoru 32

5 Hlavní nabídka aplikace 35
6 Aplikace propojená s Android Auto 39
7 Stránka aplikace na Google Play 40

Úvod

Práce se zabývá algoritmy spaced repetition, jejich rolí ve vývoji vzdělávacích aplikací a ilustruje vývoj takové aplikace. I přesto že algoritmy spaced repetition původně vznikly pro papír, nabývají významné efektivity ve spojení se softwarem. Optimalizace procesu učení pomocí výpočetní techniky je stále rozvíjející se oblastí a to potvrzují i přibývající různorodé algoritmy spaced repetition. Pro samotný vývoj vzdělávacích aplikací je tak nejen potřeba najít smysluplnou formu implementace těchto algoritmů, ale i výběr samotného algoritmu.

Motivací práce je tak vyvinout výukovou aplikaci, která bude efektivně a smysluplně těchto algoritmů využívat. Většina existujících řešení na trhu strádá buď z hlediska uživatelské přívětivosti, dostupnosti obsahu, technickým zpracováním či implementací spaced repetition. V rámci práce by tak měla vzniknout aplikace taková, která zcela nezanedbá ani jednu z těchto vlastností. K jejímu vytvoření by měly být použity pouze veřejně dostupné nástroje, které umožní lehkou distribuci aplikace uživatelům. Měla by umožňovat spravovat její obsah profesionály, kteří nemusejí mít technické vzdělání a tím zajistit možnost kvalitních výukových aktivit. Zároveň je třeba charakterizovat dostupné spaced reptition algoritmy a vybrat z nich ten nejvhodnější pro tento typ aplikace.

Výstup tohoto vývoje by měl být otestován skupinou uživatelů a z pomocí zpětné vazby z jejich strany laděn. Po dokončení tohoto procesu by měla být výstupem této práce, funkční vzdělávací aplikace, která využívá algortimů spaced repetition pro optimalizaci učení.

1 Spaced repetiton

Klíčovým faktem, na kterém metoda spaced repetition staví, je existence Ebbinghausovy křivky. To je funkce, která přibližně popisuje rychlost zapomínání lidského mozku (viz obrázek 1). Obecně je tento popis pouze přibližný, protože křivka zapomínání individuálního člověka se bude odlišovat dle jeho vlastních schopností. Tato funkce na ose x vyjadřuje množství uběhlého času od naučení se nějakého pro mozek nového pojmu (většinou vyjádřen ve dnech) a na ose y míru pravděpodobnosti, že si mozek tento pojem stále dokáže vybavit (většinou uváděna v procentech). Tato křivka odhaluje poměrně negativní fakt pro proces učení - každý člověk do pár dnů zapomene většinu toho, co se naučil. Lidstvo se však přirozeně naučilo s touto situací bojovat a odolat delší dobu míře zapomínání pomocí opakování naučeného pojmu.

S opakováním však přišla i otázka, na kterou se snaží odpovědět spaced repetition algoritmy - za jak dlouhou dobu, by si měl člověk pojem zopakovat, tak aby jeho učení bylo co nejefektivnější. Tyto algoritmy zjednodušeně pracují na bázi prodlužování intervalu opakování po každém správném vybavení si pojmu při opakování a zkracování toho intervalu po každé špatné odpovědi. Uživatel, tak pouze musí v daných intervalech provádět opakování pojmů do doby, než je pojem zcela stabilně zasazen do jeho paměti.

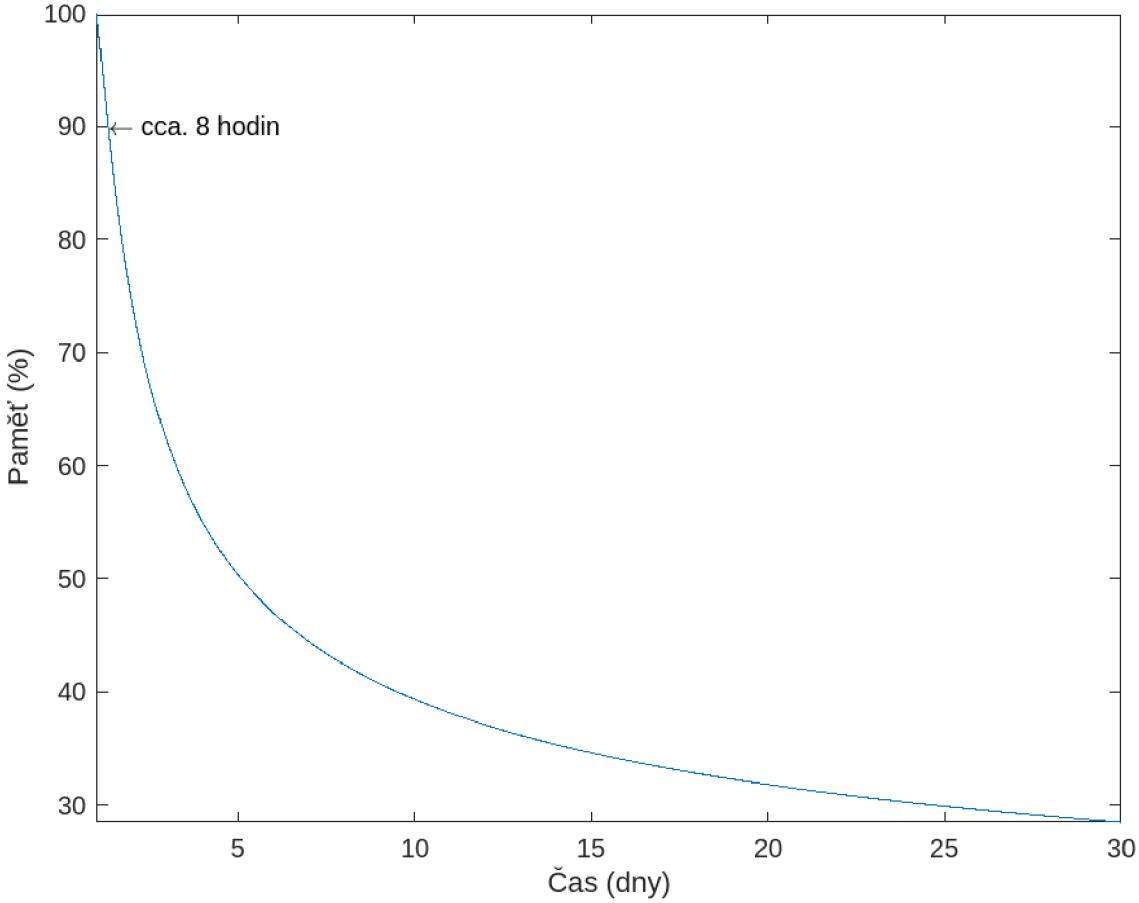
1.1 Historie a základ

Základy spaced repetition začínají s vytvořením Ebbinghausovy křivky na přelomu dvacátého století. Tu sestavil empirickým pozorováním sama sebe. Sám sebe učil nesmyslné slabiky, které opakoval v různých časových intervalech a data o opakování a zapomínání si zapisoval. Tyto data potom následně převedl do rovnice, která se stala funkcí křivky zapomínání:

$$b = \frac{100k}{(log(t))^c + k} \qquad \text{(Rovnice 1: Křivka zapomínání)}$$

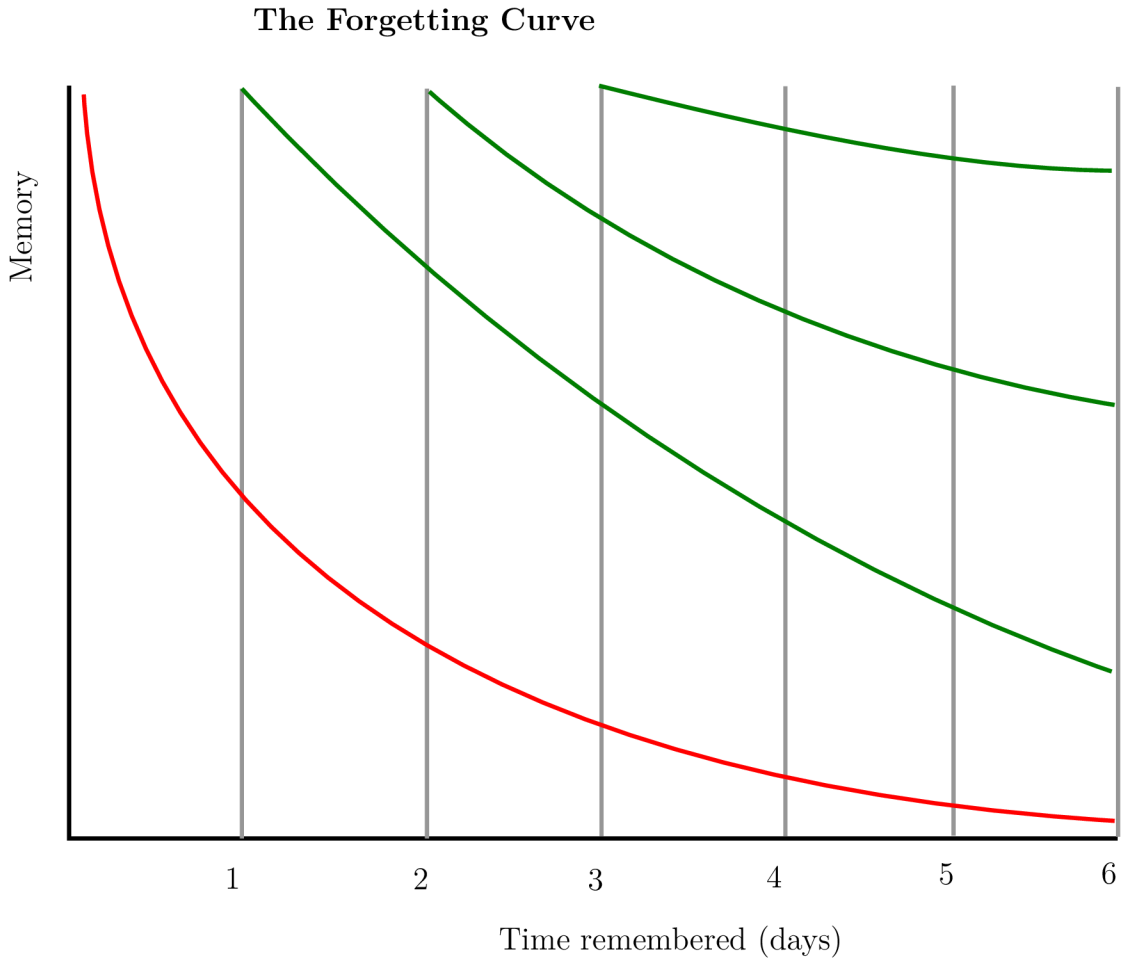
Kde b vyjadřuje míru schopnosti vybavit si daný termín, t reprezentuje čas, c a k jsou konstanty - $c = 1,25$, $k = 1,84$. Základní nastavení všech spaced repetition algoritmů se snaží udržet pravděpodobnost vybavení nad hranicí 90%. Tato hranice většinou nastává během několika hodin od naučení, v Ebbinghausově křivce je to zhruba 8 hodin.[1]

Obrázek 1: Ebbinghausova křivka zapomínání



Moderní chápání spaced repetition vyjádřil až Paul Pimsleur v roce 1967. Kdy znázornil efekt opakování na křivku zapomínání. Po každém opakování se stabilita vzpomínky vrací na své maximum (100% šance vybavit si pojem), zároveň se však zapomínání tohoto pojmu zpomalí. Po každém opakování se tak křivka zapomínání stává stále méně konvexní (viz obr. 2). Z grafu rovněž vyplývá, že tento efekt, může být zresetován, pokud uplyne opravdu velmi dlouhá doba od opakování - uživatel se přestane učit.[2]

Obrázek 2: Efekt spaced repetition na křivku zapomínání[3]



Prvním digitálním využitím spaced repetition byl algoritmus SM-2 vytvořený v roce 1987 Piotrem Wozniakem. Jednalo se o zdigitalizovanou verzi algoritmu SM-0, který byl určený pro tužku a papír. Algoritmus byl součástí první verze programu SuperMemo, který byl napsaný v turbo pascalu a běžel na DOSu, později byl přepsán i pro Windows.[4]

1.2 Algoritmy

1.2.1 SuperMemo

V současnosti se mezi nejpopulárnější algoritmy spaced repetition řadí skupina algoritmů SuperMemo. Ty jsou neustále vyvíjeny a od papírové verze SM-0 se postupně dostaly až k verzi SM-18. Paradoxně jsou však populárnější algoritmy dřívější, vzhledem k jejich jednoduchosti a tím i jednodušší implementaci, zároveň také nabízejí open source licenci. Rovněž v současné době není dostupný dostatečný objektivní výzkum potvrzující, že novější algoritmy jsou efektivnější (většina dostupného výzkumu je prováděna samotnou společností SuperMemo). Všechny algoritmy SM

pracují s hodnocením, vetšinou je škála hodnocení 0-5, ale lze ji změnit, vzhledem k tomu, že nemá velký vliv na algoritmus (důležité je pouze, zda je známka pozitivní nebo ne). Algoritmy SM, se snaží vyjádřit 2 komponenty paměti (stabilita a vytěžitelnost paměti) pomocí matematických vztahů mezi různými faktory učení. Tato snaha se stává více patrnější s každou vydanou verzí.

- SM-2 - Postaven na atributu easiness factor (E-faktor), který se snižoval při špatných odpovědích a zvyšoval při správných a vyjadřoval obtížnost daného pojmu pro uživatele.[4]
- SM-5 - K E-faktoru, přibývá matice O-faktorů. O-faktory, určují velikost prodloužení dalšího intervalu.[5]
- SM-6 - Snaha zpřesnit křivku zapomínání uživatele už od dřívějších opakování.[6]
- SM-8 - Změna matematických vzorců pro vyšší přesnost, odhalení chyb způsobených empirickým pozorováním.[7]
- SM-15 - V potaz jsou nyní brány všechny předchozí známky předmětu při každém přepočítání atributů. Velmi vzrostla komplexita celého algoritmu a dle Wozniakova pozorování i efektivita učení.[8]
- SM-17 - Kladen ještě vyšší důraz na kompletní historii učení daného termínu. První přímé využití dvousložkového modelu paměti.[9]

1.2.2 Leitnerův systém

Dalším důležitým algoritmem je Leitnerův systém. Ten pracuje s krabicemi kartiček k učení. Každá krabice má vlastní interval opakování a krabice jsou seřazeny dle velikosti této periody. Podle daných intervalů uživatel tahá karty a snaží se vybavit si pojem, v případě úspěchu přesouvá kartu do krabice s delší periodou, v případě neúspěchu přesouvá kartu do krabice s kratší periodou. Jedná se o velmi jednoduchý algoritmus, který nebyl určen pro počítače, několik z prvních výukových aplikací však tento algoritmus implementovalo.[**leitner**]

1.2.3 FSRS

Současně nejnovější rodinou algoritmů jsou algoritmy FSRS. Jde o open source algoritmy, které jsou stále ve velmi aktivním vývoji. FSRS zvolilo podobný přístup k modelu paměti jako SuperMemo. Samotné matematické popisy jsou však mnohem jednodušší než novější SM algoritmy. Efektivitu, tak FSRS nabírá zcela jiným způsobem. Jednoduché matematické vzorce, kromě hodnot z předchozích výsledků uživatele, obsahují i váhy. Tyto váhy mají své základní hodnoty a s nimi tento algoritmus funguje obdobně jako jednodušší spaced repetition algoritmy. Jeho efektivita se zvyšuje až s jeho schopností optimalizovat váhy algoritmu dle specifik paměti uživatele. Váhy lze přepočítat z předchozích výsledků uživatele, k tomu je určen model

neuronových sítí. Následující vzorec je příkladem výpočtu šance na vybavení pojmu spočítaného z t - počtu dnů od posledního opakování a S - stability pojmu, která je určena za pomoci zmíněných vah uživatele.[10]

$$R(t, S) = (1 + \frac{t}{9 * S})^{-1} \qquad \text{(Rovnice 2: Výpočet šance na vybavení pojmu)}$$

1.3 Efektivita spaced repetition

Zkoumání efektivity spaced repetition jako učební metody je problematické. Vzhledem k tomu, že už ze základu spaced repetition, nenastává zlepšení v blízké době od začátku učení. Většina výzkumů však potvrzuje, že spaced repetition přináší velmi dobré výsledky z dlouhodobého hlediska. Síla vzpomínky takto zapamatovaného pojmu je velmi vysoká. Zároveň je odstraněn problém, který nastává u tradičního učení - opakování velkého obsahu dat najednou vede k únavě kognitivních schopností a tím i k zhoršení schopností paměti. Spaced repetition každá data k opakování rozkouskuje dle intervalů, a tak by tento stav měl nastat pouze výjimečně.[11]

Některé výzkumy také zkoumaly efekty spaced repetition na lidské tělo, zejména mozek. Pomocí EEG bylo například zkoumáno napětí některých částí mozku, které souvisí s lexikální pamětí. Tyto výzkumy dospěly k závěru, že reakce mozku ukazuje v těchto oblastech lepších reakci pokud zkoumaný jedinec využíval k učení spaced repetition.[11]

Spaced repetition má zároveň nepřímý dopad na motivaci. Svým přístupem, při kterém se uživatel učí po malých dávkách pojmů, nedochází k vyhoření. Zároveň také vzniká zvyk učit se pravidelně. Spaced repetition také poskytuje záchranou síť - díky přesně naplánovaným intervalům uživatel nezapomíná, jeho výsledky se časem zlepšují a tím roste i jeho motivace.[11]

1.4 Současné využití

1.4.1 Anki

Aplikací, která je plně postavena na spaced repetition je Anki. Jde o výukovou aplikaci pro samouky na bázi kartiček, která je open source. Sama o sobě neobsahuje, žádné kurzy, zato je umožňuje uživatelům vytvářet a to i na velice komplexní úrovni. Hodnocení uvnitř aplikace je provedeno pomocí uživatelského sebehodnocení. Původně byla aplikace postavena na algoritmu SM-2, na konci roku 2023, však umožnila použití FSRS algoritmu. Vybrání algoritmu a nastavení jeho parametrů je tak plně v rukou uživatele, rovněž jako vytváření obsahu.[12]

1.4.2 Programy SuperMemo

Programy SuperMemo jsou proprietárním softwarem a jediným zdrojem novějších SM algoritmů navržených Piotrem Wozniakem. Do nedávna se jednalo pouze o aplikace desktopové, nově však existuje i aplikace pro mobilní zařízení. Desktopové

aplikace, které vycházeli desítky let byly velmi komplexní. To umožňovalo plně využít potenciálu spaced repetition a hyperoptimalizovat výuku. Jejich komplexita však mnoho uživatelů odradila. Aplikace navíc sama o sobě neobsahovala žádné kurzy, uživatel si je musel vytvořit sám a to uvnitř onoho komplexního programu.

Tyto problémy s horším uživatelským požitkem se zřejmě snaží odstranit nová mobilní aplikace SuperMemo. Ta se jeví podobně jako její konkurence. Obsahuje tak i několik připravených kurzů. Změnil se i systém monetizace - desktopové aplikace byly monetizovány jednorázovou platbou, mobilní aplikace však zvolila model měsíčních plateb, nebo možnost jednorázově koupit jeden určitý kurz. Kurzy v mobilní aplikaci jsou vytvořeny různými autory v různých jazycích, v době rešerše se však nepodařilo najít žádný v češtině.

1.4.3 Duolingo

Asi nejvíce populární aplikací pro výuku jazyků je Duolingo. Duolingo je proprietárním softwarem, nelze tak přesně odvodit jeho vnitřní mechanismy. Duolingo velmi dlouhou část svého vývoje spaced repetition implementovalo a stále tvrdí, že jej v nějaké formě využívá. Jeho přesné současné využití není známo a slouží tak zřejmě jako podpůrný mechanismus pro výběr obsahu lekcí a opakování. Jelikož je Duolingo proprietární a komerční software, veškeré kurzy v něm jsou připraveny přímo vývojáři. To přináší vyšší kvalitu kurzů, zároveň je však jejich počet poměrně nízký v porovnání s jinými aplikacemi. Pro česky mluvící uživatele jak tak například dostupný pouze kurz angličtiny.[13]

1.5 Inkrementální čtení

Piotr Wozniak na algoritmech space repetition postavil novou metodu - inkrementální čtení. Jde o metodu jak člověku umožnit zapamatovat si dlouhý text pomocí opakování klíčových otázek. Uživatel tak v textu vyznačí klíčové informace, následně je mu vygenerována série otázek, které cílí na tyto informace. Z těchto otázek se pak sestaví karty, které využívají spaced repetition pro opakování. Inkrementální čtení tak představuje unikátní způsob, jak do běžných výukových aplikací využívajících spaced repetition kromě pojmů a vět přidat možnost učit se efektivně celé souvislé texty. V současnosti je inkrementální čtení dostupné v aplikacích SuperMemo a Anki.[14]

2 Cíle práce

2.1 Vytvoření aplikace

Hlavním cílem práce je vytvoření výukové aplikace, která využívá algoritmů spaced repetition. Toto využití by tak mělo být účelové a vést ke zvýšení efektivity výuky. To zároveň implikuje, že komplexita výsledná komplexita implementace nesmí negativně zasáhnout požitek z používání aplikace, protože by to mohlo vést ke snížení motivace uživatele. Hlavním úkolem spaced repetition, by tedy mělo být počítání intervalů dalšího opakování již naučených karet. Zároveň by aplikace měla umožnit automatické hodnocení odpovědí pro optimální výuku bez zmatení uživatele. Nabízí se možnost vytvoření vlastního spaced repetition algoritmu, ten by se však komplexitou ani efektivitou nepřiblížil již existujícím řešením. Je proto potřeba vybrat vhodný algoritmus z již existujících. Z definice tak nepřipadají v úvahu proprietární algoritmy, kterými jsou například novější verze SuperMemo.

2.2 Vytvoření editoru

Pro výukovou aplikaci je extrémně důležitý její obsah. Ten by se měl průběžně upravovat a expandovat. Aby toto bylo možné zajistit, je potřeba v rámci práce vyvinout editor výukových aktivit, který by dokázal používat člověk specializující se na výuku namísto programování. Aplikace, tak zároveň potřebuje i přehledné schéma pro strukturování výukového materiálu. Výukové aktivity by měly být seskupeny do větších tématických celků, tak aby na sebe mohly plynule navazovat. Toto řazení by mělo být možné vytvářet pomocí daného editoru.

2.3 Propojeni editoru s aplikací

Musí být zajištěna přeprava dat mezi editorem a aplikací. Veškerá data by měla být přepravována mezi částmi systému spolehlivě a bezpečně. Tato data zároveň musí být nějakou formou unifikována a centrálně spravována. Proto je v rámci práce nutno vyvinout software, který je schopný efektivně veškerá potřebná data uchovat. Zároveň by měl tento software provádět veškeré výpočty s nimi, aby bylo zamezeno případným nesrovnalostem vzniklým z nerovnoměrných úprav zbylých částí systému.

3 Použité nástroje

3.1 Flutter

Klíčovým pro samotnou aplikaci byl framework Flutter. Ten využívá programovací jazyk dart a oba tyto nástroje vyvíjí společnost Google. Flutter využívá strukturu takzvaných widgetů. Widgety jsou třídy, které všechny dědí vykreslovací metodu. Tato vykreslovací metoda určuje, co uživatel uvidí při spuštění aplikace. Výhodou a principem widgetů je jejich schopnost vkládat je do sebe a dělat tak stromy widgetů. Tento fakt umožňuje, že každý text i tlačítko v aplikaci je widgetem a potomkem jiného widgetu. Nejdůležitější vlastností Flutteru se v této práci ukázala možnost vygenerování aplikace pro různé platformy z jednoho kódu. Po napsání jednoho kódu je tak možné aplikaci zkompilovat, pro Android i iOS, ale i třeba web a desktopové operační systémy. Tato možnost je pouze omezena případnými importovanými knihovnami, které spoléhají na určité funkce systému. Samotné kompilování mezi platformami nevyžaduje žádnou znalost vývoje softwaru pro tyto platformy, každopádně developer s těmito znalostmi má možnost upravit, jak tato kompilace proběhne. Například u Androidu lze upravit nastavení gradlu, který samotnou aplikaci kompiluje nebo také typické Android soubory, jako je třeba AndroidManifest.xml. Flutter zároveň nabízí velké množství komunitních knihoven. Tyto knihovny obsahují nejen nové funkce, ale někdy i celé widgety, které vývojář může využít a upravit dle potřeby.

3.2 Ruby on Rails

Nejvíce nástrojů vyžadoval vývoj API. Samotný server vyřizující API requesty využívá frameworku Ruby on Rails, který je dostupný pro programovací jazyk Ruby. Ruby on Rails se soustředí na komfort vývojáře, automaticky tak nabízí propojení s libovolnou databází a přehlednou strukturu projektu. Z této struktury jsou pro API důležité 2 části: modely a controllery. Modely definují strukturu databáze a umožňují přístup k tabulkám pomocí objektů, včetně filtrování tabulek. Díky modelům tak lze pracovat s databází i bez znalosti SQL a kód pracující s databází je kompatibilní i s jiným typem databáze. Druhou podstatnou částí struktury projektu v Ruby on Rails jsou controllery. Ty obsahují veškerou logiku API, včetně zpracování requestů a posílání odpovědí na ně. Ruby on Rails zároveň podporuje využití Ruby knihoven (gemů), tyto knihovny přinášejí nové funkce zejména díky podpoře ruby pro C-binding. V aplikaci se tato možnost projevila třeba využitím bezpečnějšího hashovacího

algoritmu Argon2.

3.3 PostgreSQL

Pro API byla vybrána PostgreSQL databáze, zejména pro její efektivní způsob uložení složitějších datovým typů, z těchto typů byl například využit datový typ array nebo speciální datový typ pro JSON. Veškerá trvalá data aplikace jsou tak uložena v této databázi, ke které má přístup pouze API. PostgreSQL navíc nabízí možnost autentizace a autorizaci pomocí socketu, tuto možnost zároveň podporuje i Ruby on Rails. Jelikož se jedná o připojení pomocí lokálního souboru, byla tato metoda vybrána pro autentizaci v rámci aplikace.

3.4 Redis

API také využívá NoSQL databázi Redis. Tato databáze slouží pro uložení asynchronních úloh API do fronty a jejich následné odebrání. Redis byl vybrán, protože data ukládá přímo v operační paměti a tím je zápis a čtení rychlejší. Úlohy z redisové databáze postupně zpracovává ruby knihovna Resque. Tato knihovna je přímo postavena na redisové databázi a její výhodou je, že definuje všechny asynchronní úlohy pomocí objektově orientovaného programování - každá úloha je třída, tyto třídy lze přímo implementovat do aplikace v Ruby on Rails. Díky této implementaci lze z kódu Ruby on Rails přirozeně vkládat úlohy do fronty i přesto, že samotná aplikace nemá přímý přístup do databáze redis.

3.5 Vue.js

Jde o javascriptový framework, který byl využit pro vytvoření editoru. Vue.js se používá zejména na frontendové webové aplikace. Využívá tedy prostředky pro webové stránky jako HTML, CSS a javascript. Samotný editor byl nakonec zkompilován pomocí nástroje Vite, který Vue.js podporuje v základu. Výhodou javascriptových frameworků je množství dostupných knihoven. Samotný Vue.js má vlastní knihovny, které vývojáři zpříjemňují práci s ním, z těchto knihoven byl použit třeba Vue Router, který umožňuje efektivní směrování uživatelských requestů. Pro uživatelské prostředí byla využita knihovna Naive UI, která nabízí komponenty přímo pro Vue.js a zrychluje tak práci s designem stránky.

3.6 Nginx

Slouží jako reverzní proxy před všemi službami. Pro webovou aplikaci a editor zároveň poskytuje SSL certifikáty. Nginx také podporuje velké množství requestů najednou, vzhledem k robustní struktuře výukového softwaru a počtu requestů, které

jsou odeslány, je tato vlastnost klíčová. Nginx je zároveň multifunkční, kromě funkce reverzní proxy je použit i pro hosting statických webů (editor a webová verze aplikace) a spravuje jejich bezpečnostní nastavení.

3.7 DHU

Desktop Head Unit je součástí vývojových prostředků Android SDK. Jedná se o emulátor automobilového systému Android Auto. Lze jej spustit na libovolném operačním systému a propojit s mobilním zařízením pomocí USB. Výhodou je, že emulátor zobrazí i nepodepsáné aplikace a aplikace, které nebyly nahrány na obchod Google Play.

3.8 IDE

- IntelliJ IDEA - Použitá pro vývoj aplikace ve flutteru. Poskytuje doplněk pro jazyk dart i samotný flutter. Dokáže kód spustit pro libovolné zařízení včetně webového prohlížeče. Zároveň nabízí efektivní debugging přímo na mobilní zařízení pomocí propojení přes USB.
- RubyMine - Použit pro vývoj API, protože je to jediné IDE s vysokou podporou přímo pro programovací jazyk Ruby a jeho frameworky. Umožňuje tak spouštět Ruby on Rails s různými konfiguracemi a ladit takto rozsáhlou aplikaci za běhu. Zároveň umožňuje přímo v IDE přistupovat do databází a upravovat je.
- VSCodium - Použit pro vývoj Editoru. Jde o opensource verzi populárního IDE VSCode. Díky velice široké nabídce doplňku jej lze využít s libovolným programovacím nástrojem.

4 Výukový software

4.1 Algoritmus

Vybrat algoritmus pro výukovou aplikaci je poměrně komplexní problém. Jejich efektivita se velmi obtížně objektivně porovnává a navíc efektivita není vše, co je od takového algoritmu potřeba. Například spoléhání na samohodnocení uživatele může přinést nevýhody. Uživatel jako zdroj hodnocení je velmi nespolehlivý, navíc každý uživatel sebe hodnotí různým způsobem - velmi běžné je, že se uživatel hodnotí pouze extrémními známkami (nejnižší a nejvyšší možnou). Důležité je tak implementovat algoritmus takový, u kterého lze hodnocení spočítat automaticky a zároveň je více imunní proti menší škále hodnocení. Z těchto důvodů byl vybrán FSRS algoritmus verze 3.5 (Na začátku práce byl použit algoritmus verze 3, který byl v té době nejnovější verzí, a v průběhu aktualizován). Zároveň je algoritmus FSRS opensource pod licencí MIT a nabízí knihovny pro různé programovací jazyky s poměrně přehlednou dokumentací.

4.2 Struktura

Kompletní výukový software je rozdělen do tří modulů, v kterých každý má svou specifickou funkci. Ve středu stojí API, které zpracovává data. Tato data lze uživatelsky přívětivěji do API zanést editorem. A existující data zprostředkována API zobrazí uživateli samotná výuková aplikace a poskytne zpětnou vazbu daného uživatele.

Systém využívá dvou databázových systému. PostgreSQL databázi pro uložení dat o uživatelích, výukových aktivitách a uživatelskou historii s těmito aktivitami. Druhou databází je Redis, který slouží pouze pro správu asynchronních operací API. Přímý přístup do obou databází má pouze API.

Před všemi částmi systému, stojí reverse proxy (nginx) poskytující každé možnost běžet se SSL certifikátem. Všechny části běží na stejné doméně, ale na různých poddoménách. Všechny části systému běží pouze na jednom stroji s operačním systémem Debian 10 a jsou zcela přenositelné na jiné distribuce.

5 API

API bylo napsáno v programovacím jazyce Ruby a využívá jeho framework pro stavbu webových aplikací Ruby on Rails. Tyto nástroje byly vyhodnoceny jako ideální pro stavbu robustního REST API pro správu a poskytování dat, zejména díky užitečným vestavěným funkcím a struktuře umožňující jednoduché modelování databáze přímo pomocí Ruby datových typů. Rovněž je využívána Ruby knihovna Resque, kterou lze zcela integrovat s Ruby on Rails. Tato knihovna má za úkol vyřizování asynchronních úloh na novém vlákně a tím zabrání čekání na vyřízení operací v hlavním procesu API.

Pro API byla přímo vytvořena i Ruby knihovna obsahující FSRS algoritmus. Jedná se o přepis opensource algoritmu ve verzi 3.5 do Ruby s modifikacemi pro běh ve vlastní výukové aplikaci, vzhledem k tomu že původní knihovny FSRS byly vytvořeny pro běh v aplikaci Anki. Celá knihovna je prakticky tvořena dvěma třídami. Pro použití knihovny je třeba pouze vytvořit instanci jedné z těchto tříd a použít jedinou její metodu. Tato metoda slouží pro přepočítání všech atributů výukové aktivity a vrátí veškeré možné nové hodnoty atributů pro každou možnou dosaženou známku.

5.1 Modely dat

Každá tabulka Postgres databáze je vyjádřena modelem API. Tyto modely jsou klíčové k porozumění celkové funkcionality systému. Lze pro ně zároveň i generovat vlastní metody, které se mohou automaticky spouštět před i po vytvoření záznamu v databázi i po odstranění z databáze. Každý model je vlastní třída, Ruby on Rails zároveň obsahuje speciální makra pomocí, kterých lze efektivně definovat kardinalitu a parcialitu nebo akce po smazání cizího klíče.

5.1.1 User se seznamem vah

K potřebám identifikace uživatele a správě jeho dat slouží model User, který obsahuje informace o zvoleném heslu a emailu uživatele, rovněž také informaci o tom, kdy se uživatel zaregistroval. Speciálním atributem v tomto modelu je seznam vah, které jsou využívané k vypočítání stability zapamatování jednotlivých pojmů a lze je tedy optimalizovat individuálně dle schopností daného uživatele. Na tento model jsou pak napojeny další modely, které obsahují data spojené s výukovými aktivitami. Zároveň tato funkce obsahuje metodu, která před zápisem nového záznamu do databáze,

zašifruje heslo uživatele pomoci hashovacího algoritmu Argon2.

5.1.2 RefreshToken

Pro autentizaci je rovněž důležitý model RefreshToken, který si ukládá informaci o vygenerovaných auth a refresh tokenech. To umožňuje aplikaci zůstat přihlášená a v moment kdy z bezpečnostních důvodů vyprší platnost auth tokenu ho lze pomocí refresh tokenu zregenerovat. Zároveň si také uchovává informaci, kdy auth token expiruje, aby si aplikace mohla v čas zažádat o regeneraci.

5.1.3 Course

Nejvyšším výukovým celkem systému je kurz, jeho strukturu definuje model Course. Sám o sobě má Course pouze dva atributy - název a mezinárodně unifikovanou čtyřpísmenou zkratku jazyka, skládající se z kódu jazyka dle normy ISO 639 a z kódu země dle normy ISO-3166. Název kurzu slouží pouze pro využití v uživatelském rozhraní, veškerá logika systému operuje pomocí zmíněné zkratky pro vyvarování se nepřesností zejména u jazyků a dialektů, které jsou vázané na specifický region. Součástí kurzu je však další menší model, který ho dál člení.

5.1.4 Deck

Tímto modelem je Deck, který umožňuje roztřídit výukový materiál do smysluplných lekcí. Opět sám o sobě obsahuje pouze atributy pro uživatelské rozhraní, těmi jsou název a popis balíčku a jeden speciální atribut typu boolean, který označuje zda balíček obsahuje pouze audio výukové aktivity. Audio balíčky jsou běžně dostupné v aplikaci, jsou takto označeny aby aplikace po spuštění v Android Auto věděla, které balíčky lze přehrát bez nutnosti interakce ze strany uživatele.

5.1.5 ActivityCard

Posledním modelem v tomto řetězci jsou samotné výukové aktivity, ty jsou definovány modelem ActivityCard. ActivityCard obsahuje atributy pro uložení informace v češtině a v cizím jazyce. Zároveň obsahuje údaje o typech aktivit, které lze generovat z těchto informací, tyto údaje jsou zaneseny jako pole stringů, což umožňuje správně vybraný databázový systém. Zbylé atributy obsahují přídavné možnosti pro aktivitu jako okolnosti za kterých se má přehrát audio nebo texty nápověd.

5.1.6 UserCard

Modely User a ActivityCard jsou propojeny mezimodelem (v návaznosti na to i mezitabulkou v databázi) UserCard. Mezimodel tedy obsahuje atributy pro identifikátor modelů User a ActivityCard. Kromě těchto atributů však obsahuje atributy potřebné pro FSRS algoritmus včetně přesného času naplánovaného opakování. Tento

model je tak základním prvkem pro uchování informací pro použití metody spaced repetition.

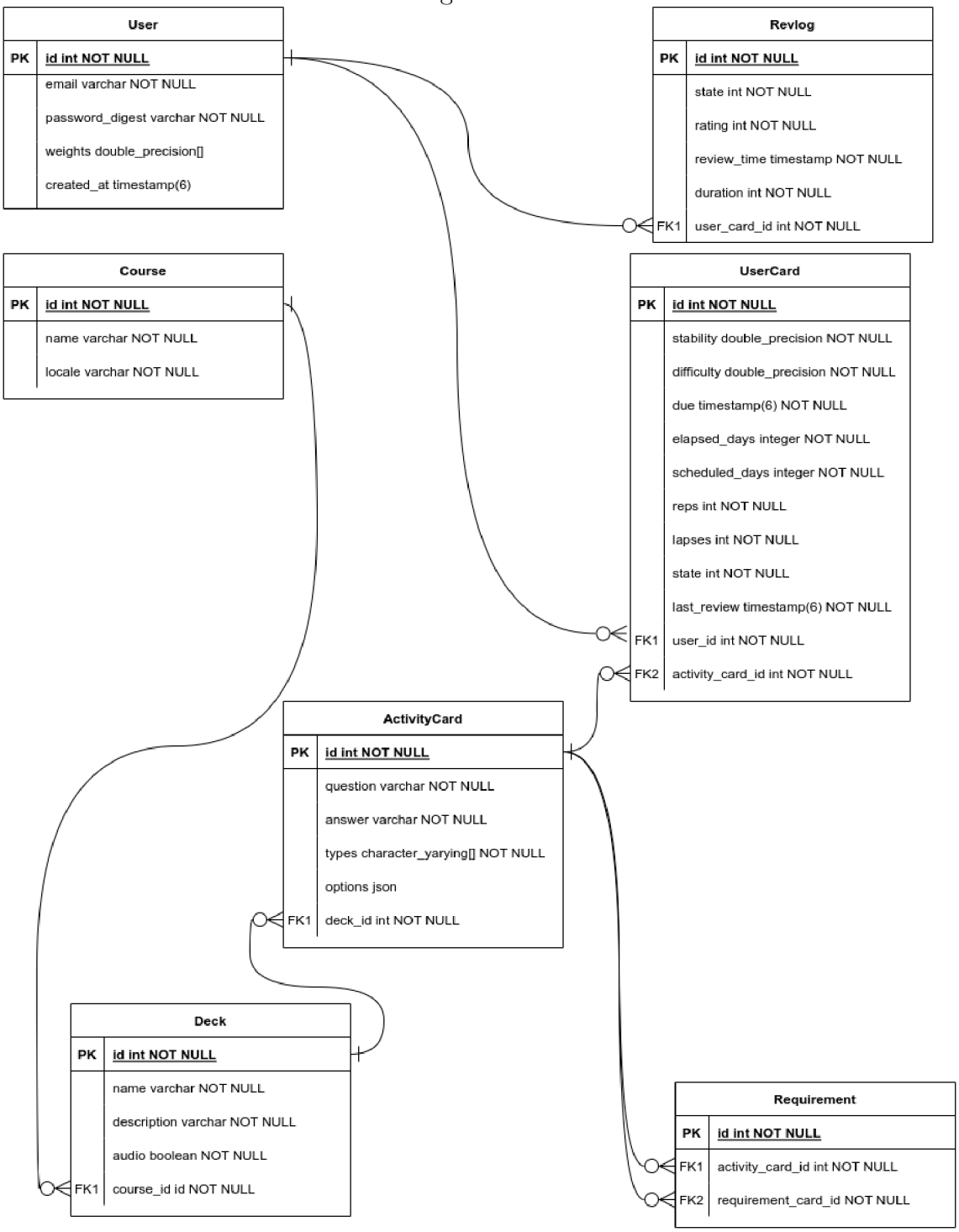
5.1.7 Revlog

Pomyslným rozšířením modelu UserCard je model Revlog, který ukládá historii všech předchozích výsledků uživatele dané aktivity. Tato historie slouží k případné optimalizaci vah algoritmu pro přesnější vystižení schopností uživatele. Historie obsahuje všechny předchozí hodnoty atributů a čas vytvoření, takže pomocí ní lze zcela zrekonstruovat celý proces učení uživatele.

5.1.8 Requirement

Posledním klíčovým modelem je Requirement, který spojuje ActivityCard sám se sebou. Tím umožňuje každé aktivitě definovat, které informace by měl uživatel již umět před začátkem výuky této nové aktivity. Requirementy lze řetězit za sebou v aplikaci se zobrazí všechny, které jsou potomky aktivity, která má logickou hodnotu true pro atribut standalone.

Obrázek 3: Diagram modelů dat API



5.2 Endpointy API

API poskytuje velmi širokou škálu endpointů pro práci s daty. Tyto endpointy představují jedinou možnou formu komunikace dalšího softwaru s API. Jelikož se jedná

o REST API, všechny endpointy jsou URL. Veškeré dotazy a odpovědi jsou ve standardním formátu JSON. Přístup k endpointům je zprostředkován pomocí reverse proxy nástrojem nginx, žádosti na neznámé cesty se tak ani nedostanou na API. Pro zabezpečení a identifikaci uživatele slouží jeho auth token, který lze u všech endpointů zaslat v HTTP hlavičce „Authorization“.

5.2.1 Autentizace

I přesto že se nejedná o klíčovou funkci výukové aplikace, je autentizace důležitou funkcí celého systému. Je totiž třeba uživatele identifikovat pro správu jeho výsledků. Autentizace pracuje na bázi emailu a hesla, kdy heslo je zašifrováno pomocí hashovacího algoritmu Argon2 pro práci v programu a uloženi v databázi. Podporované jsou následující požadavky:

1. POST /users/sign_in - Žádost pro přihlášení uživatele. Po verifikaci zaslaných údajů je uživateli vrácen auth token, refresh token a čas expirace auth tokenu.

2. POST /users/sign_up - Žádost pro registraci uživatele. Pokud uživatelský email nebyl zabrán jiným uživatelem, jsou uživatelské údaje uloženy do databáze a uživateli jsou vráceny auth token, refresh token a čas expirace auth tokenu.

5.2.2 Uživatelské informace

- Endpointy, které vrací informace o uživateli, pro zprostředkování informací o jeho výsledcích během učení. Tyto informace jsou uživateli zobrazeny v profilové části aplikace. Tyto endpointy nejsou používané v logických funkcích a mají pouze informační funkci.

1. GET /user/info - Vrátí informace o uživateli, tím jsou email a údaj o počtu opakování dle algoritmu.

2. GET /user/cards - Vrátí informace o všech kartách, které se uživatel naučil, včetně vnitřních atributů, potřebných pro výpočet dalších intervalů opakování. Z těchto dat, lze uživateli zobrazit informaci jak rychle, která slova zapomíná.

5.2.3 Získávání aktivit

Endpointy pro získávání seznamu aktivit pro učení a opakování. Tyto endpointy používá aplikace v procesu výuky.

1. GET /learn?deck=$id - Žádost o seznam aktivit pro výuku specifického balíčku. Parametr id je unikátní identifikátor daného balíčku.

2. GET /review - Žádost o seznam aktivit pro opakování. Pomocí auth tokenu, který je poslaný v hlavičkách žádosti, se z databáze získají aktivity, které by si měl daný uživatel zopakovat dle FSRS algoritmu.

5.2.4 Výuka

Pro spouštění FSRS algoritmu stačí pouze jeden endpoint. Ten je využit jak při výuce tak při opakování. Tento endpoint je přímo napojen na logiku knihovny FSRS a poskytuje tak veškerou logiku pro přepočítávání atributů jednotlivých pojmů, které se uživatel naučil.

1. POST /repeat - Pošle data o uživatelově výkonu v dané aktivitě. Součástí žádosti je známka - číselné ohodnocení správnosti uživatelova výsledku, délka trvání vyřešení dané aktivity v milisekundách a identifikátor dané aktivity (ActivityCard id).

5.2.5 Úprava dat

Žádosti, pro vytváření a strukturování výukového materiálu aplikace. Odpovídají třem CRUD operacím - create, update, delete.

1. POST /course/create - Žádost pro vytvoření nového kurzu.
2. POST /course/update - Žádost pro úpravu existujícího kurzu.
3. DELETE /course/delete - Žádost pro smazání existujícího kurzu.
4. POST /deck/create - Žádost pro vytvoření nového balíčku.
5. POST /deck/update - Žádost pro úpravu existujícího balíčku.
6. DELETE /deck/delete - Žádost pro smazání existujícího balíčku.
7. POST /card/create - Žádost pro vytvoření nové výukové aktivity.
8. POST /card/update - Žádost pro úpravu existující výukové aktivity.
9. DELETE /card/delete - Žádost pro smazání existující výukové aktivity.

5.2.6 Výběr záznamů

Žádosti pro výběr specifických seznamů záznamů modelů z databáze. Většina se využívá pro interaktivní výběry v editoru. Endpointy s filtrem existují z důvodu menší velikosti datového toku při častějších requestech, které vyžadují detailnější informace pouze o specifických záznamech.

1. GET /courses - Vrátí seznam všech kurzů, které se nacházejí v databázi.
2. GET /deck/all - Vrátí seznam všech balíčků, které se nacházejí v databázi.
3. GET /decks?course=$id - Vrátí seznam všech balíčků, které se nacházejí v daném kurzu, pomocí parametru course. Zároveň pomocí auth tokenu zjistí informaci, které balíčky už se uživatel naučil.

4. GET /decks/audio - Vrátí seznam všech balíčků, které obsahují pouze audio karty.
5. GET /card/all - Vrátí seznam všech výukových aktivit
6. GET /card/course?course=$id - Vrátí seznam všech výukových aktivit v daném kurzu pomocí parametru course.

5.2.7 Audio

Žádosti, které získají informace ze Speechactors API a pro další práci s audiem. Informace ze Speechactors jsou potřebné pro editor aby se dala připravit žádost pro vygenerování audia.

1. POST /upload/audio - Žádost pro nahrání vlastního audio souboru k výukové aktivitě.
2. GET /languages - Vrátí všechny zkratky jazyků, jejichž podporu nabízí Speechactors. Tyto zkratky mají stejný formát jako zkratky jazyků v modelu Course.
3. GET /voices?locale=$zkratka - Vrátí všechny jednotlivé hlasy, které Speechactors nabízí ve vybraném jazyce. Jazyk lze vybrat vyplněním parametru locale zkratkou z endpointu /languages. S hlasy jsou zároveň vráceny další informace o hlasech, včetně genderu hlasu a unikátního identifikátoru hlasu.

5.3 Příklady formátů komunikace

Pro komunikaci s editorem a aplikací API využívá několik formátů žádostí a odpovědí. Všechny tyto žádosti a odpovědi využívají datový formát JSON. JSON byl vybrán pro svou univerzálnost implementace v programovacích nástrojích a jednoduchému přístupu k atributům.

```
1  {
2  "deck": 1,
3  "question": "Dobré ráno",
4  "answer": "Good morning",
5  "types": ["translation", "dictation", "sort"],
6  "standalone": true,
7  "requirements": [2, 3],
8  "play_audio": ["end"],
9  "generate_audio": ["answer"],
10 "voice": "en-US-ElizabethNeural",
11 "options": {
12     "question_hints": {"dobré": "good", "ráno": "morning"},
13     "answer_hints": {"good": "dobrý", "morníng": "ráno"}
14     }
15 }
```

Kód 1: Žádost pro přídání výukové aktivity

5.4 Use Case

5.4.1 Výpočet intervalu opakování

1. Na endpoint /repeat, přijde žádost obsahující id aktivity, čas do odpovědi a výslednou známku za aktivitu
2. API načte existující data z databáze pokud existují, jinak vytvoří nový záznam s defaultními hodnotami
3. Načte se knihovna FSRS, která z předchozích a nových dat spočítá nové atributy, pomocí matematických vzorců a vah uživatele
4. Nové atributy dané aktivity uživatele jsou zapsány do databáze
5. Zároveň je přidán další revlog do databáze, pro sestavení plné historie dané aktivity

5.4.2 Výběr aktivit k opakování

1. Na endpoint /review přijde žádost z aplikace pro získání aktivit, které by si měl uživatel zopakovat
2. Z databáze jsou vybrány veškeré uživatelské karty aktivit, u kterých je atribut due (plánované datum opakování) starší než současný čas.
3. Tento seznam karet je seřazen dle nejstaršího času due atributu.
4. Atributy karet jsou naformátovány do formátu JSON a zaslány zpátky aplikaci.

6 Editor

Editor dat byl vytvořen v javascriptovém frameworku Vue.js. Ten byl zvolen vzhledem k tomu, že se nejedná o robustní aplikaci, ale pouze aplikaci, která slouží jako uživatelské prostředí pro editaci dat v API. Do editoru se lze přihlásit pomocí autentizace přes API. Editor si uloží do místního úložiště prohlížeče auth token, refresh token a přesný čas vypršení auth tokenu. Díky této funkci se uživatel nemusí znovu přihlašovat při každém použití editoru.

Editor obsahuje dvě kategorie funkcí - vytváření a upravování existujících dat. Každá tato sekce obsahuje jednotlivé interaktivní formuláře pro práci s daným modelem. Sekce k vytváření nových dat tak obsahuje:

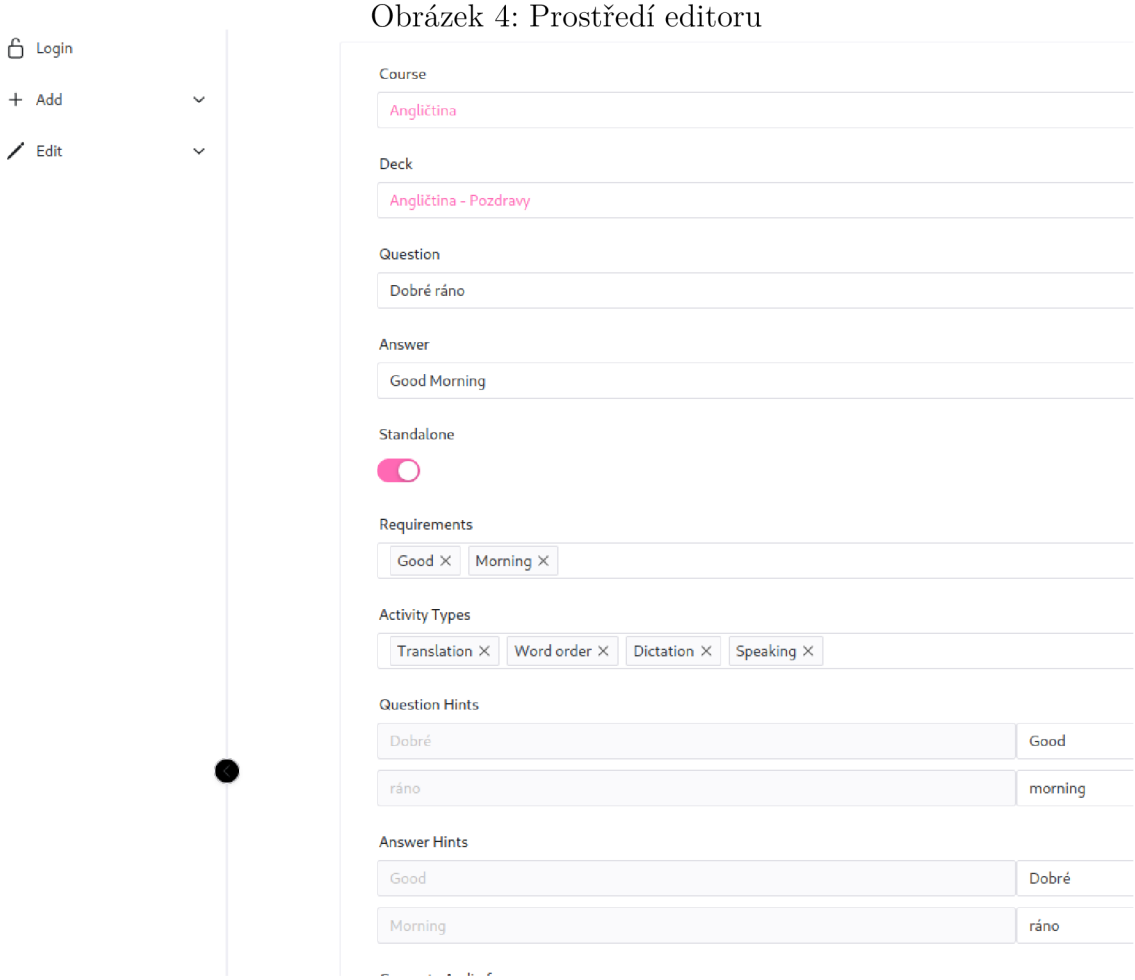
1. Vytváření kurzu - Velice jednoduchý formulář, který pouze obsahuje nutnost vyplnit název kurzu a vložit zkratku jazyku.

2. Vytváření balíčku - Formulář, který umožňuje vyplnit jméno a popis balíčku a přepínač zda se jedná o audio balíček. A zároveň požaduje zařazení balíčku do existujícího kurzu.

3. Vytváření karty aktivity - Nejvíce komplikovaný formulář. Vyžaduje zařazení do balíčku (v návaznosti i do kurzu). Obsahuje možnosti vybrat požadavky na předchozí naučené karty i zda tato karta je pouze požadavkem (například pouze jedno slovo) nebo samostatně stojící informací. Pak je potřeba zadat daný pojem v obou jazycích a vybrat typy aktivit, které lze z těchto informací vyplnit. Po zadání informací se automaticky vygeneruje interaktivní formulář pro zadání nápověd pro jednotlivá slova. Jako poslední obsahuje formulář možnosti audia. Je možné nahrát vlastní audio nebo si nechat vygenerovat audio pomocí umělé inteligence. Generování audia probíhá pomocí služby Speechactors, se kterou komunikuje API - editor tedy nemá žádnou přímou komunikaci s jiným softwarem než API. Pro každý jazyk je tak na výběr z několika typů hlasů.

Sekce úpravy dat, obsahuje prakticky stejné formuláře. Navíc je však potřeba vybrat existující záznam daného modelu. Po výběru se automaticky vyplní ostatní kolonky a lze je tak snadno upravit. Rovněž je po výběru záznamu možné jej kompletně vymazat.

Obrázek 4: Prostředí editoru



6.1 Use Case

V hypotetické situaci, kdy správce chce založit zcela nový kurz je jediným speciálním požadavkem na správce pouze znalost formátu jazykových zkratek kurzu. S touto znalostí pak vypadá tok událostí následovně:

1. Správce otevře webovou aplikaci editoru v libovolném internetovém prohlížeči.
2. Správce se přihlásí zadáním svého emailu a hesla. Tím se odešle žádost s těmito údaji na endpoint /users/sign_in.
3. API vyhodnotí správnost daných údajů.
   (a) Údaje jsou správné, uživateli je tak vrácena odpověď s refresh tokenem, auth tokenem a datem jeho expirace.
   (b) Údaje jsou špatně, API vrací HTTP kód 401 a editor ho vyzývá k zkontrolování údajů.

4. Editor uloží atributy odpovědi do místní paměti prohlížeče zašifrované pomocí rozhraní webcrypto.
5. Správce využije formuláře pro vytvoření kurzu a vytvoří nový kurz.
6. Editor zašle žádost na endpoint /course/create.
7. Správce využije formuláře pro vytvoření balíčků a vytvoří libovolný počet balíčků, které přiřadí do vytvořeného kurzu.
8. Editor zašle žádost na endpoint /deck/create.
9. Správce využije formuláře pro vytvoření karet a vytvoří libovolné množství karet, které přiřadí do vytvořených balíčků.
10. Editor zašle žádost ve na endpoint /card/create.

V případě, že správce chce upravit již existující balíček, dochází k následujícímu toku událostí:

1. Správce otevře webovou aplikaci editoru v libovolném internetovém prohlížeči.
2. Správce se přihlásí zadáním svého emailu a hesla. Tím se odešle žádost s těmito údaji na endpoint /users/sign_in.
3. API vyhodnotí správnost daných údajů.
   (a) Údaje jsou správné, uživateli je tak vrácena odpověď refresh tokenem, auth tokenem a datem jeho expirace.
   (b) Údaje jsou špatné, API vrací HTTP kód 401 a editor ho vyzývá k zkontrolování údajů.
4. Správce otevře formulář pro editaci balíčků, vybere v něm, který balíček chce změnit a změní libovolné atributy.
5. Editor zašle žádost na endpoint /deck/update.

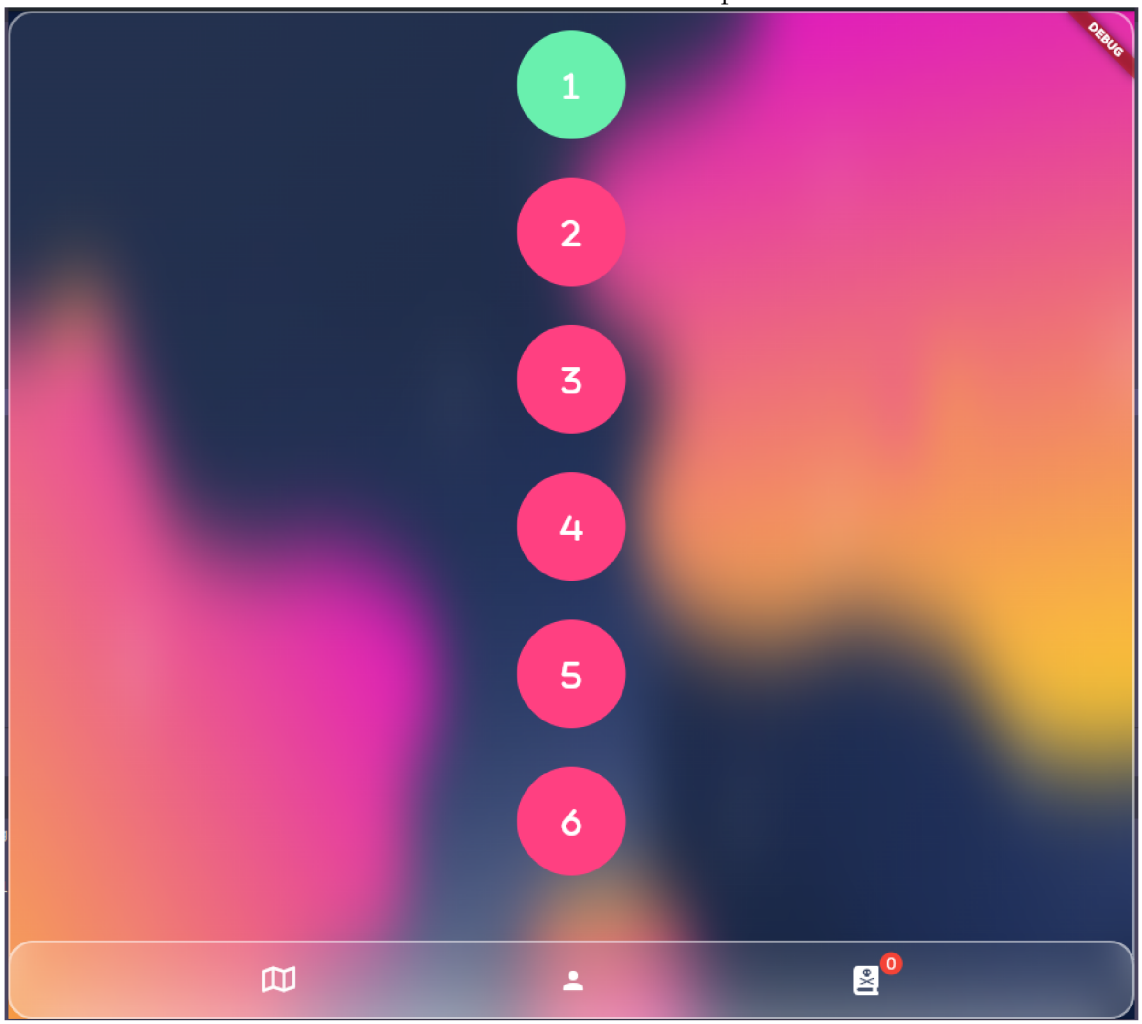
7 Aplikace

Aplikace je klientem využívající API pro zprostředkování funkcí výuky uživateli. Byla zkompilována pro web, kde je dostupná na adrese https://congoro.xyz a lze ji zde využít v plné funkcionalitě. Na mobilním zařízení ji lze i nainstalovat jako progresivní webovou aplikaci - ta je rovněž vygenerována automaticky pomocí frameworku flutter. Mnohem přívětivější je však využití aplikace zkompilované přímo pro Android. Ta je dostupná i z obchodu Google Play. Potencionálně možné je aplikaci zkompilovat pro další platformy například iOS, aplikace však v rámci práce nebyla na těchto platformách otestována.

Design aplikace je minimalistický a obsahuje pouze pro uživatele nutné užitečné funkce. Uživatelské prostředí se snaží být intuitivní za pomoci využití ikon, popisků tlačítek a nadpisů u widgetů. Samotný design je alespoň částečně responzivní z kontextu velikosti obrazovky zřízení a z kontextu platformy spuštění. Aplikace stažená z Google Play rovněž vždy vybere nejvhodnější verzi pro architekturu CPU zařízení a zároveň i pro oficiální kategorii DPI zařízení.

Aplikace vyžaduje pro jakoukoliv její funkcionalitu připojení k internetu. Aplikace rovněž požaduje další oprávnění k zařízení, například přístup k mikrofonu kvůli úlohám, jež vyžadují hlasovou interakci ze strany uživatele. Potřeba připojení k internetu je zapříčiněna tím, že aplikace prakticky nepřistupuje k interní paměti zařízení (výjimkou je bezpečné uložení textových hodnot jako auth token). Balíčky kurzu, jejich aktivity ani audia tak nejsou offline vůbec dostupné, zároveň aplikace vůbec neukládá audio nahrané uživatelem při hlasových úlohách.

Obrázek 5: Hlavní nabídka aplikace



7.1 Use Case

7.1.1 První spuštění

Při prvním spuštění aplikace je tok událostí následující:

1. Uživatel spustí samostatnou aplikaci a nebo otevře její webovou verzi.
2. Zobrazí se přihlašovací formulář.
3. (a) Uživatel již má účet z dřívější doby a tak se přihlásí,
   (b) Uživatel ještě nemá účet a tak zvolí možnost registrace a zaregistruje se,
4. (a) Aplikace pošle žádost s údaji na endpoint API /users/sign_in,
   (b) Aplikace pošle žádost s údaji na endpoint API /users/sign_up,
5. API vyhodnotí platnost údajů,

   (a) Údaje jsou správné, uživateli je tak vrácena odpověď obsahující auth token, refresh token a informaci s datem expirace auth tokenu.
   (b) Údaje jsou špatné, API vrací HTTP kód 401 a aplikace ho vyzývá k zkontrolování údajů.

6. Aplikace uloží atributy odpovědi do permanentní paměti v zašifrované podobě (přesná implementaci se liší dle platformy).

7. Aplikace provede žádost na endpoint /courses k získání dostupných kurzů.

8. Uživateli jsou kurzy zobrazeny a je vyzván k výběru.

9. Po výběru kurzu se ID tohoto kurzu uloží do místní paměti zařízení.

10. Aplikace provede žádost na endpoint /decks?course s ID kurzu pro získání všech balíčků tohoto kurzu.

11. Uživatel je nyní v základní nabídce aplikace a vidí všechny dostupné balíčky vybraného kurzu.

7.1.2 Výuka

Pro běžné učení uživatele je tok událostí následující:

1. Uživatel si vybere balíček z nabídky dostupných balíčků pro daný kurz

2. Aplikace provede GET žádost na endpoint /learn?deck s id vybraného balíčku a odpovědí získá výukové aktivity a informace o balíčku.

3. Uživateli jsou zobrazeny informace o balíčku a seznam pojmů, které obsahuje.

4. Uživatel klikne na tlačítko „Učit se“, pro zahájení výuky daných pojmů.

5. Aplikace začne postupně zobrazovat výukové aktivity dle typů, které daný pojem umožňuje.

6. Uživatel postupně řeší dané výukové aktivity dle svých schopností.

   (a) V případě správné odpovědi si aplikace zaznamená známku 3 a čas trvání odpovědi a uživatel pokračuje dál.
   (b) V případě špatné odpovědi si aplikace zaznamená známku 1 a čas trvání odpovědi, uživateli je zobrazena správná odpověď a může pokračovat dál.

7. Po dokončení všech typů aktivit daného pojmu odešle aplikace průměrnou známku a průměrný čas trvání odpovědi na endpoint /repeat.

8. Aplikace stejným způsobem projde všechny pojmy a následně uživatele přesměruje zpět do hlavní nabídky.

7.1.3 Opakování

1. Již po startu aplikace pošle request na endpoint /reviews pro získání počtu aktivit k opakování. Tento počet zobrazí uživateli v hlavní nabídce.
2. Uživatel stiskne tlačítko pro možnost opakování naučených pojmů.
3. Aplikace provede žádost na endpoint /review. Odpovědí je jí vrácen seznam všech aktivit k zopakování, všechny jejich atributy a audio.
4. Uživateli je zobrazena první z těchto aktivit.
5. Po odpovědi je známka zaslána na API pomocí žádosti /repeat.
6. Uživateli je zobrazena další aktivita a tento proces se opakuje dokud nedojdou aktivity k zopakování.
7. Po dokončení všech aktivit k zopakování je uživateli oznámeno, že si zopakoval vše co bylo potřeba.

7.2 Typy aktivit

Aplikace obsahuje několik typů aktivit. Různé typy aktivity jsou nutností pro procvičení látky z několika aspektů. Například u jazyků je důležité kromě správného pravopisu, být schopný dané slovo vyslovit nebo jej rozeznat z řeči cizí osoby. Diverzita cvičení navíc zabraňuje monotónnosti, která by mohla časem poškodit motivaci uživatele. Aplikace nabízí následující typy aktivit:

- Prostý překlad - Uživateli je zobrazen text, ten je buď v češtině nebo ve studovaném jazyce. Nad ním je instrukce, která uživateli říká aby daný text přeložil do druhého jazyka. To lze provést zadáním překladu do textového pole uprostřed obrazovky a odesláním odpovědi tlačítkem „Poslat“. Posléze je odpověď vyhodnocena porovnáním textu uživatele a přednastavené odpovědi, velikost písmen a speciální znaky jsou ignorovány.
- Diktování - Úloha zaměřena na rozeznání slov z řeči. Uživateli je zobrazeno tlačítko, po jehož stisknutí je mu přehrána nahrávka s řečí v cizím jazyce. Jeho úkolem je přepsat tuto řeč do slov do textového pole pod tlačítkem a následně odpověď odeslat tlačítkem „Poslat“. Odpověď je vyhodnocena porovnáním textu uživatele a přednastavené odpovědi, velikost písmen a speciální znaky jsou ignorovány.
- Vlastní výslovnost - Úloha je zaměřena na výslovnost cizího jazyka uživatelem. Uživateli je zobrazen text v cizím jazyku a tlačítko „Začít nahrávat“. Po stisku tohoto tlačítka je úkolem uživatele přečíst danou větu. To může uživatel zopakovat dokud není spokojen. Po každém nahrání je nahrávka vyhodnocena a uživateli je zobrazeno, která slova vyslovil správně. Rozeznání slov využívá Google Speech to Text rozhraní, které je vestavěné v operačním

systému Android. Ve webové aplikace je využito rozhraní Web Speech API, které podporuje většina moderních prohlížečů, které je rovněž napojeno na danou Google službu.

- Doplnění slova - Tato úloha slouží k zvýraznění větných struktur cizího jazyka. Uživateli ji zobrazena věta nebo fráze, kde jedno ze slov je nahrazeno mezerou. Nad větou se nachází český překlad celé věty. Pod větou se nachází tři různá slova. Tyto slova lze přesunout do mezery ve větě. Cílem uživatele je vybrat takové slovo, aby doplněná věta byla překladem vzorové české věty. Možnosti lze měnit do doby, než uživatel klikne na tlačítko „Poslat“. Posléze je vyhodnoceno zda slovo, které uživatel vybral se shoduje s přednastavenou odpovědí.

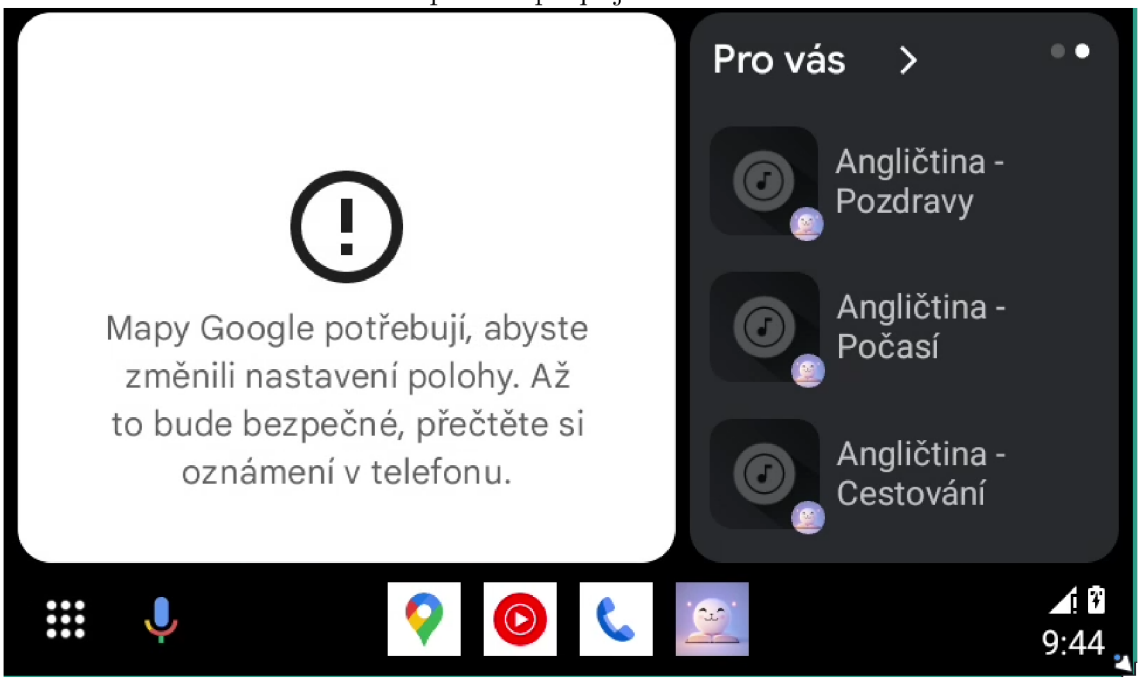
- Skládání slovosledu - Jazyky mají různé slovosledy, proto tato úloha testuje uživatele v jejich znalosti. Před uživatelem leží věta rozdělená na jednotlivá slova. Slova lze přesouvat a tím měnit jejich pořadí. Cílem uživatele je přesunout slova do takového pořadí, aby vytvořil správný slovosled dané věty. Poté lze odpověď odeslat ke zkontrolování pomocí tlačítka „Poslat“. Aplikace následně porovná pořadí slov uživatele s pořadím slov v přednastavené odpovědi.

- Poslech rozhovoru - Jedná se o úlohu, která není interaktivní. Uživatel pouze poslouchá připravený rozhovor, pokud není spuštěn v Android Auto, vidí rozhovor napsaný ve formě zpráv mezi účastníky rozhovoru.

7.3 Android Auto

Aplikace se dokáže propojit s Android Auto a nabídnout audio aktivity. Kvůli limitaci ze strany API Android Auto je nemožné přidat možnost interagovat pomocí hlasu - možné jsou pouze vestavěné hlasové příkazy, samotné psaní hlasem je možné pouze pro chatovací aplikace. Aplikace využívá API pro mediální aplikace. Díky tomu zobrazí všechny balíčky, které jsou v editoru označeny jako audio jako dostupné skladby. Po vybrání skladby začne Android Auto postupně přehrávat všechny audia karet obsažených v daném balíčku.

Obrázek 6: Aplikace propojená s Android Auto



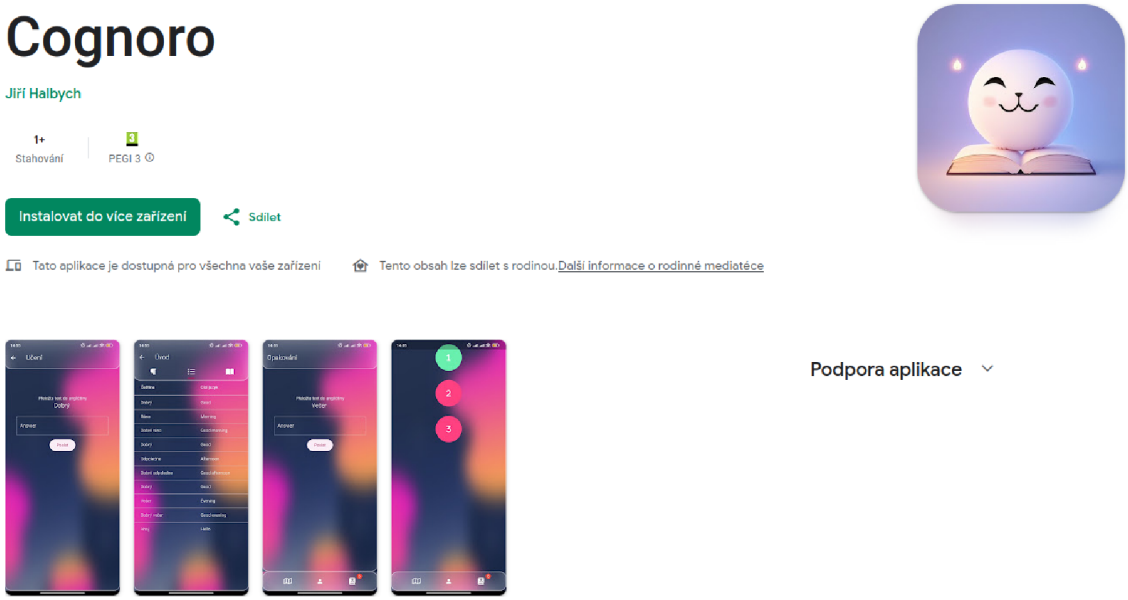
7.4 Google Play

Aplikace byla nahrána na obchod s aplikacemi Google Play. Samotný proces nahrání aplikace je velmi komplexní, ale přináší mnoho výhod. K vytvoření účtu vývojáře je třeba mít ve svém Google účtu platební identitu a následně zaplatit jednorázový poplatek ve výši 25 amerických dolarů. Po vytvoření účtu získá vývojář přístup do webové aplikace Google Play Console. Stále však nemůže aplikaci nahrát. Nejdříve musí vývojář potvrdit, že vlastní zařízení s operačním systémem Android. Toho lze dosáhnout stažením aplikace Play Console a přihlášením se v ní do svého Google účtu. Po potvrzení android zařízení musí ještě vývojář potvrdit svou vlastní identitu. K potvrzení identity je potřeba vyfotit libovolný státem vydaný doklad totožnosti a nahrát obě strany do formuláře. Verifikace těchto dokumentů trvá několik dnů.

Po verifikaci účtu, může vývojář začít s nahráváním své aplikace. To je ze začátku možné pouze v rámci interního testování. Interní testování umožňuj nahrání apk nebo appbundle souboru. Rozdíl v souborech je ve výsledné velikosti stahování. Pokud vývojář použije soubor apk, každý uživatel, jehož zařízení podporuje tento apk soubor, jej bude stahovat v celé jeho velikosti. Pokud však vývojář nahraje appbundle soubor, Google Play Console z něj automaticky vygeneruje stovky apk souborů (různé kombinace podpory CPU architektur, velikostí displeje apod.). Při stahování aplikace z Google Play tak uživatel stahuje pouze vygenerovaný apk soubor vhodný pro jeho zařízení, který má zlomkovou velikost originálního appbundle souboru. V interním testování je aplikace dostupná pouze pro testery, kteří jsou do něj přihlášeni. Pro přihlášení testera do interního testování musí vývojář přidat jeho email do seznamu testerů a následně mu zaslat odkaz pro přihlášení do interního

testování. S tímto odkazem si tester může aplikaci přímo stáhnout z Google Play.

Obrázek 7: Stránka aplikace na Google Play



8 Testování a výsledky

Aplikaci byla otestována v rámci uzavřeného testovaní umožněného platformou Google Play. Díky tomu mohly být odstraněny některé chyby ve funkcionalitě a uživatelském prostředí aplikace. Zároveň byly přidány některé funkce, které mohou uživatelům zpříjemnit používání aplikace. Příkladem takové funkce je například možnost vypnout úlohy, které vyžadují zadávání textu.

8.1 Příklady výpočtu intervalu opakování

Pokud uživatel vidí kartu naprosto poprvé, vypočtený interval dalšího opakování je v řádu minut. Pouze v případě známky 4 (naprostá bezchybnost) je interval 6 dnů, vzhledem k tomu, že známka 4 je prakticky nedosažitelná pokud uživatel tento pojem nikdy předtím neviděl. Podobnost intervalu u nižších známek však neznamená, že první opakování je irelevantní - z výsledné známky jsou odvozeny další atributy, které jsou použity při výpočtech následujících intervalů.

Z následující tabulky je patrný rozdíl mezi špatnými známkami 1 a 2, které většinou znamenají opakování v řádu minut a dobrými známkami 3 a 4, kde je opakování většinou v řádu dní. Zároveň se ukazuje jak nejvíce extrémní známky 1 a 4 dokáží velmi drasticky interval ovlivnit. Například známka 1 znamená zapomenutí pojmu, a tím pádem je třeba okamžité opakování (5minutový interval).

První opakování	První interval	Druhé opakování	Druhý interval
1	1 minuta	1	5 minut
1	1 minuta	2	10 minut
2	5 minut	1	5 minut
2	5 minut	2	10 minut
1	1 minuta	3	1 den
3	10 minut	1	5 minut
2	5 minut	3	1 den
3	10 minut	2	10 minut
3	10 minut	3	2 dny
1	1 minuta	4	2 dny
4	1 den	1	5 minut
2	5 minut	4	2 dny
4	1 den	2	10 dní
3	10 minut	4	3 dny
4	1 den	3	20 dní
4	1 den	4	48 dní

Tabulka 1: Tabulka intervalu opakování po dvou opakováních.

V následujících tabulkách lze pozorovat kompletní učební proces jednoho termínu uživatelem a lze tak i odhadnout obtížnost daného pojmu pro daného uživatele. U pesimistického scénáře je nutno přihlédnout k tomu, že nastává většinou v případě nedodržení času opakování a tak by v realitě časová osa byla mnohem delší. U každého termínu proces učení většinou přibližně následuje jednu z těchto tabulek, existuje však i možnost, že časem přejde jedna tabulka v druhou.

Známka	Vypočtený interval	Skutečný interval
3	10 minut	1 hodina
3	2 dny	2 dny
4	17 dnů	18 dnů
3	21 dnů	22 dnů
3	3 měsíce	

Tabulka 2: Tabulka optimistické scénáře výuky

Známka	Vypočtený interval	Skutečný interval
2	5 minut	2 dny
1	5 minut	1 hodina
3	1 den	3 dny
2	1 den	2 dny
3	1 den	

Tabulka 3: Tabulka pesimistického scénáře výuky

Jak lze z obou tabulek vypozorovat, výsledky opakování drasticky ovlivňují délku dalšího intervalu opakování. Tento výsledek ještě více podtrhuje možné problémy u aplikací, které používají sebehodnocení uživatelů. Zároveň je zřejmá výhoda možnosti optimalizovat algoritmus přepočítáním jeho vah neuronovou sítí z předchozích výsledků uživatele.

Závěr

V rámci práce byly popsány současné algoritmy spaced repetition a jejich míra uplatnění pro implementaci v rámci výukové aplikace. Taková aplikace vznikla a v práci jsou její struktura i technická specifika detailně popsány. Aplikace tak plně umožňuje uživatelům studovat kurzy, které jsou přehledně rozděleny do tématických balíčků. Tyto kurzy a balíčky lze vytvářet a editovat za pomoci editoru, který rovněž vznikl v rámci práce. Tento editor má uživatelské prostředí a mohou jej tedy využít správci z pedagogického prostředí. Tyto dvě částí softwaru byly propojeny za pomocí REST API, které spravuje data a řídí veškerou logiku vybraného algoritmu spaced repetition. Veškeré endpointy tohoto API byly v práci popsány a tak je v budoucnu možné vyvíjet další software, který tento systém bude moci využít. Vybraný algoritmus spaced repetition je jedním z nejmodernějších a nabízí velmi efektivní způsob optimalizace učení pro každého uživatele. Algoritmus byl pouze implementován do procesu opakování, k čemuž byl původně zamýšlen. Samotná aplikace je rovněž dostupná pro uživatele na Google Play, kde probíhá její testování. Aplikace je uživateli testována a jejich zpětná vazba tak přispívá k jejímu dalšímu vývoji.

Použitá literatura

[1] EBBINGHAUS, H. *Memory: A Contribution to Experimental Psychology*. Teachers College, Columbia University, 1913. Columbia University. Teachers College. Educational reprints. ISBN 9780722229286. Dostupné také z: https://books.google.cz/books?id=oRSMDF6y3l8C.

[2] PIMSLEUR, Paul. A Memory Schedule. *The Modern Language Journal*. 1967, roč. 51, č. 2, s. 73–75. Dostupné z DOI: https://doi.org/10.1111/j.1540-4781.1967.tb06700.x.

[3] ICEZ. *ForgettingCurve.svg*. 2007. Dostupné také z: https://en.m.wikipedia.org/wiki/File:ForgettingCurve.svg.

[4] WOZNIAK, Piotr. *Optimization of learning*. 1990.

[5] WOZNIAK, Piotr. *Algorithm SM-5*. 2019. Dostupné také z: https://supermemo.guru/wiki/First_fast-converging_spaced_repetition_algorithm:_Algorithm_SM-5.

[6] WOZNIAK, Piotr. *Employing forgetting curves in spaced repetition (1991)*. 2018. Dostupné také z: https://supermemo.guru/wiki/Employing_forgetting_curves_in_spaced_repetition_(1991).

[7] WOZNIAK, Piotr. *Algorithm SM-8*. 2018. Dostupné také z: https://supermemo.guru/wiki/First_data-driven_spaced_repetition_algorithm:_Algorithm_SM-8.

[8] WOZNIAK, Piotr. *Algorithm SM-15*. 2019. Dostupné také z: https://supermemo.guru/wiki/Algorithm_SM-15.

[9] WOZNIAK, Piotr. *Algorithm SM-17*. 2021. Dostupné také z: https://supermemo.guru/wiki/Algorithm_SM-17.

[10] YE, Jarrett. *FSRS Algorithm*. 2023. Dostupné také z: https://github.com/open-spaced-repetition/fsrs4anki/wiki/The-Algorithm.

[11] XUECHEN, Yuan. Evidence of the Spacing Effect and Influences on Perceptions of Learning and Science Curricula. 1. vyd. 2022. Dostupné z DOI: 10.7759/cureus.21201.

[12] *What spaced repetition algorithm does Anki use? - Frequently Asked Questions*. 2023. Dostupné také z: https://faqs.ankiweb.net/what-spaced-repetition-algorithm.html.

[13] ONSTWEDDER, Elizabeth. *Dear duolingo: Why is spaced repetition so important for learning?* Duolingo Blog, 2023. Dostupné také z: https://blog.duolingo.com/spaced-repetition-for-learning/.

[14] WOZNIAK, Piotr. *Incremental reading*. 2020. Dostupné také z: https://supermemo.guru/wiki/Incremental_reading.

A Přílohy

A.1 Odkazy na zdrojový kód aplikace

- API - https://gitlab.com/jiri.halbych/api
- Editor - https://gitlab.com/jiri.halbych/editor
- Aplikace - https://gitlab.com/jiri.halbych/app