Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

Bakalářská práce

Vývoj webové aplikace pro podporu výuky slovní zásoby cizích jazyků

Vypracoval: Roman Zavalishin
Vedoucí práce: Mgr. Radim Remeš, Ph.D.

České Budějovice 2024

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Roman ZAVALISHIN**
Osobní číslo:	**E21165**
Studijní program:	**B0688A140010 Podniková informatika**
Téma práce:	**Vývoj webové aplikace pro podporu výuky slovní zásoby cizích jazyků**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování

Cílem práce je vytvořit webovou aplikaci pro podporu výuky slovní zásoby cizích jazyků. Aplikace bude podporovat nahrání vlastního textu ze souboru (epub, pdf apod.) a následné označení neznámých slovíček pro automatický překlad, jejich uložení a trénování. Aplikace bude naprogramována s využitím technologie ASP.NET v jazyku C#.

Metodický postup:

1. Studium odborné literatury.
2. Popis terminologie, postupů a nástrojů.
3. Návrh a popis vývoje a implementace výsledné aplikace.
4. Zhodnocení použitelnosti aplikace pro nasazení v reálném prostředí.
5. Závěry a doporučení.

Rozsah pracovní zprávy:	**40 – 50 stran**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování bakalářské práce:	**tištěná**

Seznam doporučené literatury:

1. Lock, A. (2021). *ASP.NET Core in Action.* Second Edition. Manning.
2. Meloni, J., & Kyrnin, J. (2018). *Teach Yourself HTML, CSS, and JavaScript All in One.* Pearson Education.
3. Michaelis, M. (2020). *Essential C# 8.0.* 7th Edition. Addison-Wesley.
4. Price, M. J. (2021). *C# 10 and .NET 6 – Modern Cross-Platform Development: Build apps, websites, and services with ASP.NET Core 6, Blazor, and EF Core 6 using Visual Studio 2022 and Visual Studio Code.* 6th Edition. Packt.
5. Sharp, J. (2018). *Microsoft Visual C# Step by Step.* Microsoft.

Vedoucí bakalářské práce: **Mgr. Radim Remeš, Ph.D.**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **20. ledna 2023**
Termín odevzdání bakalářské práce: **12. dubna 2024**

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13
370 05 České Budějovice

doc. RNDr. Zuzana Dvořáková Líšková, Ph.D.
děkanka

doc. RNDr. Jana Klicnarová, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 9. března 2023

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

Datum 12. 04. 2024 Podpis studenta

Abstrakt a klíčová slova

Tato bakalářská práce zahrnuje vývoj webové aplikace pro podporu výuky cizích jazyků a její nasazení na server, tak aby byla zajištěna dostupnost pro studenty i veřejnost na internetu. Aplikace využívá technologii ASP.NET ve spojení s frameworkem Blazor.

Aplikace umožňuje uživatelům nahrávat digitální verze tištěných knih ve formátech souborů EPUB, PDF nebo TXT. Předpokládá se, že jsou tyto knihy pro uživatele v cizím jazyce. Aplikace nabízí funkci, která uživatelům umožňuje kliknout na jakékoli slovo v textu knihy a získat okamžitý překlad tohoto slova. Překlad je generován překladačem Microsoft, který je založený na umělé inteligenci. S aplikací je propojen pomocí Application Programming Interface (API).

Aplikace nabízí uživateli možnost ukládat neznámá slova do slovníku pro jejich budoucí procvičení. Při procvičování neznámých slov aplikace uživateli ukazuje kartu se slovem a několik možných odpovědí, z nichž každá obsahuje překlad. Uživatel si musí vybrat variantu se správným překladem.

Klíčová slova: ASP.NET, API, poskytování překladů, vývoj programu, webová aplikace.

Abstract and key words

This bachelor’s thesis involves the development of a web application for supporting the learning of foreign languages and its deployment on a server, ensuring accessibility for both students and the public on the Internet. The application uses ASP.NET and Blazor frameworks.

The application allows users to upload digital versions of printed books in EPUB, PDF, or TXT file formats. It is assumed that these books are in a foreign language for the user. The application offers a feature allowing users to click on any word in the book’s text, immediately providing some translations of that word. The translation is generated by a Microsoft Translator based on artificial intelligence. It is connected with the application by an Application Programming Interface (API).

The application has the capability to save unfamiliar words in a dictionary for their future practice. When practicing these words, the application shows the user a card with a word and multiple variants to answer, each containing a translation. The user is required to choose the variant with the correct translation.

Key words: ASP.NET, API, translation provision, program development, web application.

Poděkování

Chtěl bych poděkovat všem, kteří mi pomáhali a podporovali mě během mého studia a při psaní této bakalářské práce. Zvláštní poděkování patří vedoucímu mé bakalářské práce, Mgr. Radimu Remešovi, za cenné rady a trpělivost. Jeho odborné vedení mi velmi pomohlo a umožnilo dokončit tuto práci. Chtěl bych také poděkovat všem členům katedry aplikované matematiky a informatiky Ekonomické fakulty za jejich odbornou pomoc a užitečné lekce během celé doby studia. Děkuji své rodině a přítelkyni za neocenitelnou podporu a porozumění. Děkuji také Mgr. Michaele Procházkové a svému blízkému kamarádovi Martinu Pučilovi za pomoc při hledání gramatických a lexikálních nedostatků v mé bakalářské práci.

Obsah

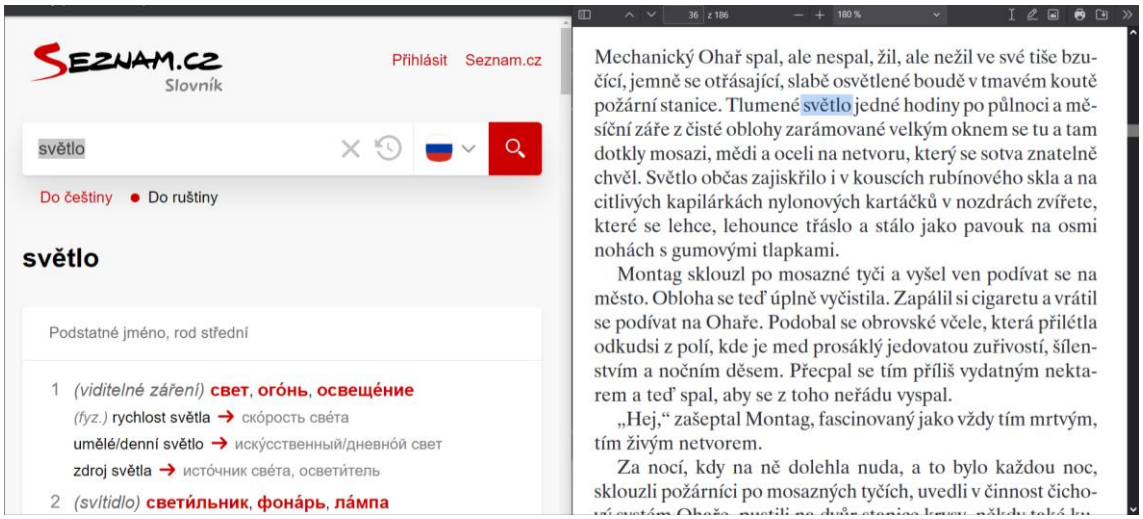
1 Úvod a motivace

Tato bakalářská práce se zabývá procesem vývoje a implementace webové aplikace určené k podpoře výuky cizích jazyků. Hlavním cílem aplikace je pomoci uživatelům lépe porozumět textům v cizím jazyce a zároveň rozšířit jejich slovní zásobu. Aby byl proces vývoje rychlejší a efektivnější, byly použity moderní technologie.

V teoretické části této práce je popsána má motivace a jsou představeny příklady podobných aplikací na trhu. Dále jsou stanoveny cíle a požadavky nutné pro vývoj aplikace a je popsána metodika vývoje softwaru. V téže části jsou detailně představeny použité programovací jazyky, knihovny a nástroje, je vysvětlen model architektury aplikace a demonstrován příklad komunikace mezi klientem a serverem. Praktická část práce je reprezentována kapitolami 6 a 7, ve kterých jsou podrobně popsány procesy vývoje a nasázení aplikace. Na konci práce je uvedeno zhodnocení použitelnosti aplikace na základě zpětné vazby uživatelů a jsou stanovena možná zlepšení aplikace.

Myšlenka vytvořit takovou aplikaci mě napadla ještě v době, kdy jsem se připravoval na studium na univerzitě v České republice. Během přípravy jsem se učil češtinu, která je pro mě cizím jazykem. Protože rád čtu knihy, na doporučení svého učitele jsem začal číst knihu v češtině, aby se zvýšila efektivita mého studia. Často jsem narážel na neznámá slova, která jsem musel ručně z e-knihy kopírovat a vkládat do překladače, což bylo časově náročné. Abych tento proces zautomatizoval, vytvořil jsem skript v programovacím jazyce Python, který jsem vždy spouštěl před čtením knihy na počítači.

Fungovalo to následovně: na obrazovce byly otevřené dva webové prohlížeče. V prohlížeči na levé straně byl otevřen slovník Seznam.cz. Na pravé straně byla zobrazena kniha a její text. Stačilo dvakrát kliknout na libovolné slovo v knize, skript je automaticky zkopíroval a vložil do slovníku, který zobrazil překlad slova. Použití tohoto skriptu je ukázáno na obrázku 1.



Obrázek 1: Ukázka použití Python skriptu

Zdroj: translator-script (2023)

Hlavním důvodem, proč jsem si vybral toto téma je, že jsem již sám nějakou dobu ke čtení využíval jednodušší verzi této aplikace, a tudíž jsem měl představu, jak by bylo možné aplikaci dále vylepšit.

2 Přehled řešené problematiky

Na trhu existuje mnoho různých druhů aplikací pro podporu výuky cizích jazyků, například Duolingo, Rosetta Stone, Babbel nebo Memrise. Tyto aplikace umožňují učit se základy mnoha různých jazyků, mají snadno použitelný design a lekce jsou prezentovány logickým způsobem. Umožňují výuku cizích jazyků prostřednictvím kurzů a lekcí. (Rash, 2024)

Svůj průzkum jsem zaměřil především na aplikace, které mají užší a shodnou funkcionalitu s aplikací, kterou navrhuji. Jejich hlavním účelem je čtení textů a knih v cizích jazycích a ukládání slov z těchto textů pro následné procvičení a opakování.

2.1 Podobné aplikace na trhu

Proč je čtení knih v cizím jazyce důležité? Pro ty, kteří se chtějí zlepšit v jazykových dovednostech, představuje právě čtení knih v cizím jazyce vhodný nástroj. Je to efektivní metoda, jak zlepšit schopnost porozumět textu a naučit se nová slova, se kterými se čtenář během čtení často setkává. Neméně důležité jsou také slovní a gramatické konstrukce, které při čtení člověk poznává a učí se, jak se jazyk v praxi používá. Čtení také pomáhá lépe se naučit větné struktury na základě příkladů. (Telf.cz, ©2022)

Ačkoli jsem sám vymyslel způsob, jak zjednodušit čtení knih v cizích jazycích poskytnutím možnosti jednoduše překládat a ukládat slova, nebyl jsem první, kdo přišel s řešením v této oblasti. Podobné aplikace již existují, ale teprve nabírají na popularitě. Není jich mnoho a na internetu jsem nenašel žádný seznam, ve kterém by byly takové aplikace uvedeny. Níže jsou uvedeny příklady tří aplikací, které mají funkcionalitu nejpodobnější mé aplikaci:

- EWA nabízí způsob učení cizích jazyků pomocí filmů, knih a her. Pomáhá rozšiřovat slovní zásobu čtením knih nebo poslechem audioknih. Nabízí knihovnu s tisíci knihami, které obsahují překlad, přepis a výslovnost každého slova. EWA je především mobilní aplikace, použití webové verze je také možné, ale téměř všechny knihy jsou k dispozici na základě předplatného, což omezuje její použití. Při zkoumání této aplikace jsem nenašel možnost nahrát vlastní knihu. Po kontaktování zákaznické podpory mi potvrdili, že takovou funkci neposkytují. (Lithium Lab Pte Ltd, 2024)

- Readlang poskytuje funkcionality pro čtení textů v původním jazyce, umožňující rychle porozumět novým slovům a frázím díky překladům, slovníkům a vysvětlením založeným na umělé inteligenci. Je možné učit se slovíčka v kontextu pomocí tzv. flashkaret[1]. Nicméně aplikace neumožňuje nahrávání textů z PDF souborů. Tvůrcem Readlangu je Steve Ridout, dříve softwarový inženýr ve společnosti Duolingo. Readlang vytvořil, protože nenašel jednoduchý nástroj, který by mu umožnil učit se španělský jazyk čtením románů a dalšího nativního obsahu. (Readlang, [2013])
- Linga je mobilní aplikace, která pomáhá učit se jazyky čtením knih v cílovém jazyce. Tuto aplikaci vyvinul Andrej Savchuk a je dostupná ke stažení zdarma pro Android a iOS. Má vlastní knihovnu a zároveň umožňuje uživatelům nahrávat jejich knihy a procvičovat označená slova. Nicméně není možné nahrát knihy v českém jazyce, učit se češtinu jako cizí jazyk a neumožňuje nahrávání kratších textů a TXT souborů. Použití aplikace bez předplatného je možné, ale omezuje funkcionalitu. Například ve verzi zdarma lze nahrát pouze 5 vlastních knih za měsíc a není možné překládat fráze a věty. O této aplikaci jsem se dozvěděl při zkoumání na trhu dostupných podobných aplikací v rámci výběru tématu bakalářské práce. Při vývoji mého projektu nebyla webová verze aplikace Linga k dispozici a v současnosti se testuje. (LingaIO, [2022])

Podle popisů aplikací výše je zřejmé, že mají své výhody, ale i nedostatky. Většina z těchto aplikací je vyvíjena pro mobilní platformy, avšak čtení na malé obrazovce mobilního telefonu může být pro některé uživatele nepohodlné. U aplikace EWA chybí možnost nahrávání vlastních knih, zatímco v Readlang se nedají nahrávat PDF soubory. V aplikaci Linga nelze nahrávat kratší texty ve formátu TXT, a navíc neposkytuje možnost učit se češtinu jako cizí jazyk. Většina aplikací je dostupná pro bezplatné používání, avšak jejich funkčnost bývá v takovém případě často omezena, nebo je pozornost uživatelů odváděna reklamami.

[1] Více informací o kartách pro memorování informací: https://bobo.cz/blog/metoda-uceni-pomoci-memorovacich-karticek

3 Úvod k vývoji

V první kapitole byl popsán Python skript, který je předchůdcem mé aplikace a který jsem chtěl zdokonalit. Zaznamenal jsem malý počet existujících aplikací na trhu a jejich nedostatky, které jsem popsal v předchozí kapitole. Tyto aspekty, společně s mým přáním vyvinout užitečný nástroj pomáhající lidem ve studiu cizích jazyků, mě přivedly k rozhodnutí vyvinout vlastní aplikaci, kterou jsem pojmenoval LinguaReader. Její cíle a požadavky jsou popsány v této kapitole.

3.1 Popis vyvíjené aplikace

Primárním cílem této bakalářské práce je vyvinout aplikaci, která nabídne uživatelům tři základní funkce:

- Základní vlastností aplikace bude podpora čtení textů z knih nahraných uživatelem.
- Další důležitou funkcionalitou je možnost kliknout na slovo v textu během čtení a okamžitě zobrazit jeho překlad, s možností uložení slova a jeho překladu do vlastního slovníku.
- Aplikace uživatelům umožní procvičování slov uložených ve vlastním slovníku prostřednictvím flashkaret s možnými variantami odpovědí.

Cílem aplikace LinguaReader není nahradit aplikace pro memorování informací, které používají speciální algoritmy intervalového opakování[2]. Takových aplikací je už na trhu mnoho a každý uživatel si může zvolit tu, která mu bude nejvíce vyhovovat. LinguaReader umožní export slov uložených uživatelem do souboru formátu CSV. Ten lze importovat do jiných, pokročilejších aplikací pro memorování informací. Přesto pro ty uživatele, kteří takové aplikace nepoužívají, bude LinguaReader nabízet jednoduchý systém procvičení uložených slov s možnými variantami odpovědi.

3.2 Požadavky na aplikaci

Funkční požadavky na aplikaci jsou založeny jak na mé vlastní zkušenosti se čtením knih v cizích jazycích, tak na diskusích s přáteli. Tyto diskuse mi poskytly nové pohledy pro stanovení funkčních a nefunkčních požadavků pro aplikaci, které jsou dále popsány.

[2] Další informace o algoritmech intervalového opakování: https://faqs.ankiweb.net/what-spaced-repetition-algorithm.html

3.2.1 Funkční požadavky

FP 1: Webová aplikace – bude zpřístupněna na internetu pomocí webového prohlížeče.

FP 2: Podstatná část aplikace bude běžet na klientském počítači a využívat jeho výpočetní prostředky. Tento přístup sníží zatížení serveru, což sníží náklady a umožní nabízet aplikaci LinguaReader zdarma.

FP 3: V aplikaci bude pouze jedna role s oprávněním na úrovni *uživatel*.

FP 4: Uživatel si bude moci samostatně vytvořit účet a přihlásit se do aplikace.

FP 5: Aplikace umožní uživatelům obnovit své heslo v případě, že ho zapomenou.

FP 6: Uživatel bude moci nahrát do aplikace elektronické knihy ve formátech PDF, EPUB a TXT.

FP 7: Aplikace zajistí možnost otevřít nahranou knihu, čímž uživateli umožní přístup k textu knihy pro čtení.

FP 8: Kliknutím na slovo v textu knihy se uživateli zobrazí několik možných překladů tohoto slova do jazyka vybraného v nastavení účtu.

FP 9: Uživatel bude moci v nastavení svého účtu zvolit jazyk, do kterého se budou slova překládat.

FP 10: Během čtení knihy bude mít uživatel možnost označit slova, kterým nerozumí, a uložit je do vlastního slovníku pro pozdější procvičení.

FP 11: Aplikace nabídne možnost procvičovat uložená slova. Ta budou zobrazena na kartách a uživatel bude muset vybrat jejich správný překlad do mateřského jazyka.

3.2.2 Nefunkční požadavky

NP 1: Všichni uživatelé budou moci používat aplikaci LinguaReader v plném rozsahu zdarma a bez reklam.

NP 2: Aplikace bude dostupná na internetu a její využití bude možné pouze s internetovým připojením.

NP 3: Uživatelské rozhraní by mělo být moderní a uživatelsky přívětivé. Nezkušený uživatel by se měl v aplikaci snadno zorientovat a měl by být schopen ji používat.

NP 4: Kód aplikace by měl být dobře strukturovaný a srozumitelný pro další rozvoj a údržbu. Metody a proměnné by měly být srozumitelně pojmenovány.

NP 5: V případě, že lze naprogramovat stejnou funkcionalitu pomocí různých jazyků, například C# nebo JavaScript, mělo by být provedeno srovnání možností a vybrán nejvhodnější jazyk.

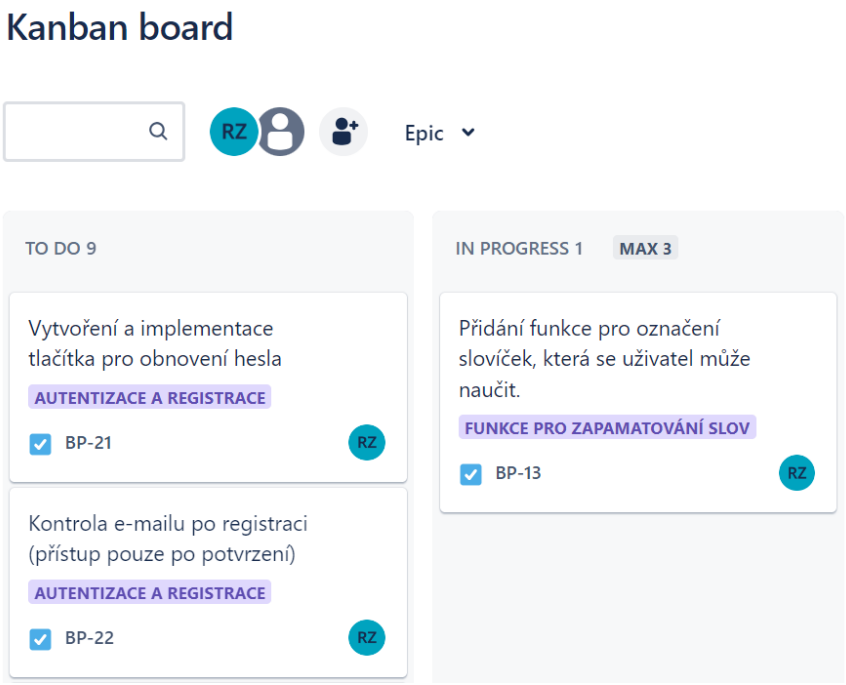
4 Metodiky vývoje softwaru

Metodika je obor, který se zabývá metodami nebo pracovními postupy. Zahrnuje doporučené kroky, přístupy, zásady, techniky, nástroje, dokumenty a metody řízení, které společně přispívají k dosažení určitého cíle. V oblasti vývoje softwaru představuje metodika soubor praktik a postupů, které pokrývají celý životní cyklus vytvářených aplikací. Metodiky pak uplatňují konkrétní metody pro řešení jednotlivých problémů. Tyto metodiky lze rozdělit podle různých přístupů k tvorbě softwaru, jako jsou například metodiky založené na vodopádovém modelu a agilní metodiky. (Martinů & Čermák, 2018, s. 29-118)

Vyvíjel jsem aplikaci LinguaReader sám, aniž bych pracoval v týmu nebo vyvíjel aplikaci pro konkrétního zákazníka, jak to předpokládá většina metodik. Proto jsem pro řízení úkolů vybral Kanban, která je jednou z agilních metodik a kterou jsem používal omezeně a výhradně pro své potřeby.

Kanban je metodika, která zvyšuje efektivitu správy projektů a projektových úloh, pomáhá optimalizovat procesy a brání přetížení týmu. Umožňuje především lépe vizualizovat tok práce a kontrolovat množství úkolů. Pro vizualizaci tato metodika využívá nástěnku a karty. Nejjednodušší nástěnka má tři sloupce: *To Do*, *In Progress* a *Done*. Karty nových úkolů se zadávají do sloupce *To Do* a obsahují popis úkolu a jméno přiřazeného vykonavatele. Proces plnění úkolu spočívá v posouvání karet s úkoly z jednoho sloupce do druhého. (Laba, 2022)

Během vývoje své aplikace jsem používal Kanban tabulku pro vizualizaci práce a úkolů. Vytvářel jsem karty s úkoly, plánoval a seskupoval je podle většího úkolu, kterému se říká *Epic*, a který zahrnuje dílčí úkoly. Měl jsem tak například úkoly *Epic* jako *Autentizace a registrace* a *Vytažení textu z PDF* pro zahrnutí různých aspektů vývoje. Nástěnka Kanban se kterou jsem pracoval během vývoje aplikace má 4 sloupce: *To Do*, *In Progress*, *Test* a *Done*. Protože jsem pracoval samostatně a bez externí kontroly, bylo pro mě užitečné nastavit i omezení Work in Progress (WIP), abych během vývoje nezačínal další úkoly, dokud nedokončím ty, na kterých jsem pracoval. Nastavil jsem omezení pro sloupec *In Progress* na 3 úkoly. To znamená, že v tomto sloupci nemohu mít více než 3 úkoly současně. Jak vypadají první dva sloupce mé Kanban tabulky je vidět na obrázku 2.



Obrázek 2: První dva sloupce Kanban tabulky projektu

Zdroj: Atlassian (b. r.)

Díky zvolené metodice jsem měl přehled o úkolech a jejich aktuálním stavu. Pomohla mi také neztratit se v množství rozsáhlých úkolů a efektivněji pracovat na projektu.

5 Technologická infrastruktura

V této kapitole jsou popsány technologie, jako jsou programovací jazyky, frameworky a knihovny, vývojová prostředí a nástroje, které jsem použil pro vývoj aplikace. Na konci kapitoly jsou uvedeny modely architektury aplikace a je ukázán příklad komunikace klienta se serverem.

5.1 Použité technologie

Když mluvíme o webové aplikaci, je těžké si představit vývoj bez použití jazyků HTML, CSS a JavaScript. Podle statistiky firmy W3Techs (©2009-2024) je HTML používán na 95,8 % webových stránek, jejichž značkovací jazyk známe.

Aplikaci jsem se rozhodl vyvíjet v programovacím jazyce C# s použitím souvisejících technologií. Ačkoli C# není základním jazykem pro vývoj webových stránek a aplikací, můj výběr byl ovlivněn osobními zkušenostmi. Tento programovací jazyk jsem se učil během studia na univerzitě a následně jsem s nabytými znalostmi začal pracovat jako .NET vývojář v softwarové firmě. V rámci své práce jsem se například podílel na vývoji webového informačního systému s využitím technologie Blazor Server. Při vývoji aplikace LinguaReader jsem používal podobné přístupy, frameworky a některé knihovny jako na projektech, na nichž jsem se podílel.

5.1.1 Programovací jazyky

1) C# verze 11.0

V době zahájení projektu byla použitá aktuální verze C# 11.0, která se běžně používá v .NET frameworku verze 7.0. C# je moderní, objektově orientovaný programovací jazyk, který pochází z rodiny jazyků C. Tento jazyk podporuje generické metody a typy, které poskytují vyšší typovou bezpečnost a výkon. Umožňuje vytvářet mnoho typů aplikací, jež běží v prostředí .NET. (Microsoft, 2023a)

V tomto projektu je v jazyce C# naprogramována celá serverová část aplikace a většina logiky na straně klienta.

2) HTML a Razor Pages

Jazyk HTML (HyperText Markup Language) je základním stavebním prvkem webu. Definuje strukturu obsahu webu pomocí značení textu, obrázků a dalšího obsahu pro zobrazení ve webovém prohlížeči. Značky HTML zahrnují speciální prvky, jako jsou *head*, *title*, *body*, *header*, aj. (Mozilla Foundation, 2024a)

Většina webových stránek potřebuje dynamický obsah, což znamená webovou stránku, která je generována za běhu spuštění kódu. Framework Razor Pages umožňuje kombinovat příkazy kódu C# se značkami HTML, aby byla generována dynamická webová stránka. Používá příponu souboru cshtml. (Price, 2021, s. 586)

Ve svém projektu přímo nepoužívám jazyk HTML, ale komponenty z knihovny Radzen.Blazor, které jsou naprogramované s využitím HTML. Tyto komponenty vytváří strukturu uživatelského rozhraní a její dynamickou přeměnu pomocí syntaxe Razor.

3) CSS

CSS (Cascading Style Sheets) jsou jazykem stylů, který se používá k popisu prezentace dokumentu napsaného v jazyce HTML a definuje způsob, jak mají být prvky zobrazeny na obrazovce. CSS patří mezi základní jazyky otevřeného webu a je standardizován ve všech webových prohlížečích podle specifikací W3C. (Mozilla Foundation, 2024b)

Podobně jako u HTML, ve svém projektu využívám předdefinované komponenty z knihovny Radzen.Blazor, které již zahrnují CSS styly. V některých případech jsem provedl menší úpravy těchto stylů nebo vytvořil vlastní.

4) JavaScript

JavaScript vyvinula společnost Netscape Communications Corporation, výrobce webového prohlížeče Netscape. JavaScript byl prvním webovým skriptovacím jazykem podporovaným prohlížeči a stále je nejpoužívanějším. Zde je několik možností, které JavaScript nabízí.

- Úprava celé webové stránky nebo její části, aniž by uživatel musel stránku znovu načíst.
- Zobrazování a interakce s daty získanými ze vzdáleného serveru.
- Zobrazování zpráv uživateli jako součást webové stránky, ve stavovém řádku prohlížeče nebo v dialogových oknech.

(Meloni, 2011, s. 66-67)

V projektu je použit JavaScript k doplnění funkcionalit C# a frameworku Blazor WebAssembly, například pro rozdělení textu knihy na stránky a jeho zobrazování, nebo také pro identifikaci vybraného slova při kliknutí. Tyto funkce vyžadují práci s Document Object Model (DOM) HTML dokumentu v prohlížeči. Jak uvádí Brind (2023) Blazor WebAssembly v současné době nemá přístup k DOM prohlížeče.

5.1.2 Frameworky

5) ASP.NET Core 7.0

Aplikace je vyvinuta pomocí ASP.NET Core verze 7.0, což je multiplatformní framework s otevřeným zdrojovým kódem pro webové aplikace, který umožňuje rychlé vytváření dynamických aplikací vykreslovaných na straně serveru. Lze ho použít pro vytváření HTTP Application Programming Interface (API), kterou je možné propojit s mobilními, jednostránkovými nebo jinými back-end aplikacemi. ASP.NET Core je moderní technologie odpovídající současným trendům ve vývoji webových aplikací, jako jsou aplikace na straně klienta a nasazení v cloudových prostředích. (Lock, 2021, s. 4-7)

6) Blazor

Blazor je webový framework založený na jazycích HTML, CSS a C#, umožňující psát webové aplikace v C# místo JavaScriptu, a je součástí ekosystému ASP.NET Core. Pomáhá vytvářet webové aplikace pomocí opakovaně použitelných komponent, které lze spouštět jak na straně klienta, tak na serveru. Je podporován ve všech moderních prohlížečích. (Microsoft, ©2024a)

Blazor framework, společně s ASP.NET Core frameworkem, tvoří základ mé aplikace.

7) Entity Framework Core 7.0

Entity Framework Core 7.0 je multiplatformní verze původní technologie mapování objektů na datová úložiště. Entity Framework, nástroj pro objektově-relační mapování, a byl poprvé vydán jako součást .NET Framework 3.5 v roce 2008. Umožňuje programátorům spojovat sloupce v databázových tabulkách s vlastnostmi v třídách, což zjednodušuje práci s databázemi. (Price, 2021, s. 407-408)

Při vývoji jsem použil Entity Framework Core pro přístup k datům v prostředí .NET, mapování entit a aktualizaci struktury tabulek v databázi pomocí databázových migrací.

5.1.3 Knihovny

8) Radzen.Blazor

Radzen.Blazor nabízí sadu více než 80 komponentů Blazor, které jsou implementovány v jazyce C# a mají otevřený zdrojový kód (Radzen, ©2018-2024). Tato knihovna je zásadním prvkem v mé aplikaci. Pro uživatelské rozhraní jsou použity její komponenty, jako jsou formuláře a jejich prvky, rozevírací seznamy, tlačítka, karty, aj. Příklad použití komponent řádků a sloupců je popsán v kapitole 6.2.

9) VersOne.Epub[3]

Tato knihovna umožňuje práci s obsahem a metadaty souborů EPUB, zajišťuje přístup k jeho sekcím a k textu každé sekce. V kapitole 6.3.1 je ukázáno, jak je použita pro přístup k datům souboru nahraného do aplikace EPUB. Díky VersOne.Epub aplikace LinguaReader získává informace o jazyce knihy a cestu k obrázku na obalu knihy.

10) iText7

Při vývoji své aplikace jsem použil knihovnu iText ve verzi 7, která byla tehdy nejnovější. Knihovna iText pro .NET je vyvinuta v jazyce C#, což usnadňuje její integraci do mé aplikace, a nabízí mnoho funkcionalit pro práci s PDF dokumenty, například možnost extrakce textu z PDF souborů. Kromě standardních metod extrakce textu také umožňuje definování vlastních způsobů pro extrakci, což mi umožnilo přizpůsobit knihovnu specifickým požadavkům mého projektu, jak je ukázáno v kapitole 6.3.2. (Apryse, b. r.)

11) JSInterop

Knihovna JSInterop podle dokumentace Microsoftu (2024a) umožňuje volání funkcí JavaScriptu z metod .NET a obráceně, což je známé jako interoperabilita JavaScriptu. V aplikaci je použita pouze pro volání funkcí JavaScriptu z .NET. Například, jak je ukázáno v kapitole 6.5, je kliknutí v textu zachyceno v C#, který následně pomocí knihovny JSInterop volá JavaScriptovou funkci k detekci kliknutého slova.

12) AngleSharp

AngleSharp je knihovna .NET pro parsování HTML, SVG a MathML, která vytváří DOM podle oficiálních W3C specifikací a podporuje parsování CSS (Rappl, b. r.). V aplikaci LinguaReader je využita pro parsování HTML, identifikaci obrázků a stylů a jejich převod na *data URL*, což je podrobněji popsáno v kapitole 6.3.1.

13) SixLabors.ImageSharp

Tato multiplatformní 2D grafická knihovna pro .NET, navržená společností Six Labors, je určena pro zjednodušení zpracování obrázků (Six Labors, b. r.). V rámci svého projektu jsem tuto knihovnu využil pouze pro změnu rozlišení obrázků na obalech knih. Pro úpravu rozměrů obrázků v textech knih je použit JavaScript.

14) ASP.NET Core Identity

Tento systém správy identit pro webové aplikace zahrnuje kromě základních funkcí, jako jsou správa údajů o identitách a přihlášení, také řadu pomocných funkcí usnadňujících

[3] Další informace o knihovně VersOne.Epub: https://github.com/vers-one/EpubReader

správu uživatelů (Spasojević, 2022). V kapitole 6.8 je popsáno, jak jsem na základě této knihovny vyvíjel funkcionality pro registraci a přihlášení uživatelů do aplikace.

15) MailKit

MailKit je multiplatformní knihovna, která umožňuje sestavení a odesílání emailů v jazyce C# (Stedfast, b. r.). V aplikaci je využita pro zasílání emailů pro potvrzení emailové adresy nebo pro obnovení hesla v případě jeho zapomenutí.

5.2 Vývojová prostředí a nástroje

V této podkapitole jsou popsány vývojová prostředí a nástroje, které byly použity během celého procesu vývoje a nasazení projektu. Většinu z těchto nástrojů jsem již dříve využíval a jejich kombinace mi umožnila na projektu pracovat efektivně.

16) Visual Studio Enterprise 2022

Integrované vývojové prostředí Visual Studio[4] bylo základním nástrojem pro napsání kódu, jeho kompilaci, spouštění a ladění. Také toto prostředí bylo využito pro vytváření a organizaci souborů obsahujících kód. Pro vyhledávání, instalaci a správu knihoven byl použit správce balíčků NuGet[5], který je integrovanou součástí Visual Studio.

17) Microsoft SQL Server Management Studio 19

Podle společnosti Microsoft (2024b), která nabízí tento nástroj, poskytuje rozhraní pro správu a konfiguraci SQL databází, databázových schémat a dat. Toto prostředí jsem používal pro testování, přehledné prohlížení a správu dat v databázi, například vymazání položek.

18) Microsoft Azure

Cloudová platforma Microsoft Azure[6] nabízí více než 200 produktů a cloudových služeb. Aplikace komunikuje se službou Azure AI Translator 3.0: Dictionary Lookup[7] za účelem překladu slov. Nasazení databází a aplikace bylo provedeno na této cloudové platformě a je popsáno v kapitole 7.

[4] Další informace o vývojovém prostředí Visual Studio: https://visualstudio.microsoft.com/cs/#vs-section

[5] Více informací o správci balíčků NuGet: https://www.nuget.org

[6] Cloudová platforma Microsoft Azure je dostupná z https://portal.azure.com

[7] Další informace o službě AI Translator 3.0: Dictionary Lookup: https://learn.microsoft.com/en-us/azure/ai-services/translator/reference/v3-0-dictionary-lookup

19) Jira Kanban

Jira Kanban je systém správy úkolů od firmy Atlassian, který nabízí přehlednou vizualizaci pracovních postupů a usnadňuje sledování a organizaci úkolů (Atlassian, ©2024). V kapitole 4 bylo popsáno, jak jsem během vývoje stanovoval a spravoval úkoly pomocí tohoto systému.

20) GitLab[8]

Platformu pro správu verzí GitLab jsem využíval pro účely zálohování kódu a záruku toho, že neztratím výsledky své práce v případě selhání počítače nebo jiných neočekávaných událostí.

21) Google a internet

Pomocí Googlu jsem vyhledával na internetu informace pro řešení chyb v kódu a postupy pro vývoj určitých funkcí. Hledal jsem například vhodné knihovny pro svůj projekt a čerpal jsem z jejich dokumentací.

22) ChatGPT-3.5[9]

Tento nástroj založený na umělé inteligenci, který generuje odpovědi na dotazy, jsem používal pro podobné účely jako vyhledávač Google – hledání řešení a vhodných přístupů k vývoji aplikace. V některých případech nabízí rychlejší nalezení řešení problému než vyhledávání pomocí internetového prohlížeče.

5.3 Model architektury aplikace

V této podkapitole je podrobněji popsaná architektura aplikace LinguaReader, která je založena na frameworku Blazor. Tento framework nabízí dva základní modely hostingu: Blazor Server a Blazor WebAssembly (WASM). Model Blazor Server zpracovává kód na serveru a aktualizuje uživatelské rozhraní prostřednictvím SignalR[10]. Výhodou tohoto modelu je, že kód aplikace se zpracovává na serveru, což zvyšuje bezpečnost a snižuje zátěž na straně klienta, ale zároveň zvyšuje nároky na síť a serverové zdroje. Model Blazor WebAssembly naopak spouští kód aplikace přímo ve webovém prohlížeči klienta, což snižuje zátěž serveru, umožňuje rychlejší reakci uživatelského rozhraní a nabízí lepší

[8] Nástroj pro správu verzí GitLab je dostupný z https://gitlab.com

[9] Nástroj pro komunikaci s umělou inteligenci ChatGPT je dostupný z https://chat.openai.com

[10] Více informací o technologii SignalR https://learn.microsoft.com/cs-cz/aspnet/core/signalr/introduction

výkon pro uživatele. Tento přístup však vyžaduje více času na stažení zdrojů a načtení aplikace. (Microsoft, 2023b)

Původně jsem plánoval použít výhradně Blazor WebAssembly, aby celá aplikace běžela na straně klienta, což by snížilo zátěž serveru a umožnilo nabízet aplikaci zdarma. Nicméně, kvůli bezpečnostním omezením webových prohlížečů týkajících se přístupu k souborům na klientském počítači, aplikace nemohla ukládat uživatelské soubory. Proto jsem se rozhodl rozšířit svou aplikaci o serverovou část. Využil jsem architekturu klient-server a kombinoval modely hostingu Blazor WebAssembly a Blazor Server. Tento přístup mi umožnil rozdělit logiku a kód mezi klientem a serverem. Splnil také mé původní představy, že většina výpočetní práce bude probíhat na klientské straně. Serverová část funguje jako API, které zpracovává dotazy klienta pro ukládání dat do databáze, odesílání dotazů do Azure AI Translator API pro překlad slov, a také zajišťuje registraci a přihlašování uživatelů. Komunikace mezi klientem a serverem je zajištěna prostřednictvím HTTP metod, jako jsou GET, POST, PUT a DELETE.

5.3.1 Příklad komunikace klienta a serveru

V tomto projektu je komunikace mezi klientem a serverem zajištěna pomocí služeb na straně klienta a API controllerů[11] na straně serveru. Služba na straně klienta odesílá dotaz na server, kde příslušný controller tento dotaz přijímá, zpracovává jej a následně odesílá požadované informace nebo data zpět klientovi. Pro odesílání dotazů klient používá třídu HttpClient z .NET frameworku, kterou injektuji do služeb pomocí návrhového vzoru Dependency Injection (DI). Podle Locka (2021, s. 292-298) tento přístup zvyšuje modularitu aplikací a usnadňuje jejich správu díky možnosti injektování závislostí do tříd. Příklad injektování třídy HttpClient do služby pro překlad slova je ukázán na výpisu kódu 1.

```
private readonly HttpClient _httpClient;

public TranslatorService(HttpClient httpClient)
{
    _httpClient = httpClient;
}
```

Výpis kódu 1: Injektování třídy HttpClient

Zdroj: vlastní zpracování (2024)

[11] Vice o vytváření webových API pomocí ASP.NET Core: https://learn.microsoft.com/cs-cz/aspnet/core/web-api

V aplikaci LinguaReader představuje každá služba zvláštní třídu a obsahuje sadu metod pro odeslání dotazů na server. Pokud je v dotazu třeba odeslat objekt, metoda ho převede do formátu JSON (JavaScript Object Notation). Následně pomocí injektované třídy HttpClient odesílá HTTP dotaz na příslušnou adresu požadavku serveru. Metoda pak čeká na odpověď od serveru a v závislosti na této odpovědi provádí další činnosti, jako je například informování o úspěchu nebo neúspěchu provedené akce. Příklad kódu pro odesílání GET dotazu na překlad slova je ukázán na výpisu kódu 2.

```
WordWithTranslations? wordWithTranslations = await
_httpClient.GetFromJsonAsync<WordWithTranslations?>($"api/Translator/Translate-
Word?word={word}&sourceLang={sourceLang}");
```

Výpis kódu 2: Odeslání GET dotazu na server

Zdroj: vlastní zpracování (2024)

API controllery na serveru očekávají dotazy od klienta. Pro každou metodu služby na straně klienta existuje odpovídající metoda v controlleru na serveru. Například na výpisu kódu 3 je ukázána definice metody pro zpracování dotazu na překlad slova.

```
[HttpGet]
[Route("api/Translator/TranslateWord")]
public async Task<ActionResult<WordWithTranslations>?> TranslateWord(string word,
string sourceLang)
```

Výpis kódu 3: Definice metody pro zpracování dotazu na serveru

Zdroj: vlastní zpracování (2024)

6 Vývoj aplikace

V této kapitole je popsána praktická část práce, jsou vysvětleny postupy a je ukázáno, jak jsem dosahoval cílů aplikace stanovených v kapitole 3. Na začátku jsou uvedeny části, potřebné pro vývoj aplikace, jako jsou databáze a uživatelské rozhraní. Poté je vysvětleno, jak je implementována první část aplikace, která zahrnuje zpracování souborů nahrávaných uživatelem a zobrazení knihy v prohlížeči. Tato část aplikace realizuje první cíl – umožnit čtení textů knih nahraných uživatelem. Podkapitoly 6.5 a 6.6 se týkají identifikace kliknutí na slovo a jeho přeložení, což odpovídá druhému stanovenému cíli – zajistit možnost kliknutí na slovo v textu a okamžitého zobrazení jeho překladu. Implementace posledního cíle spadá do kapitoly 6.7. V poslední podkapitole je popsáno, jak je zajištěna registrace a přihlášení uživatelů.

6.1 Databáze

Pro uložení dat a přístup k nim je použit relační model databáze. Rozhodl jsem se vytvořit 2 databáze. První, aplikační databáze, slouží k ukládání uživatelských dat a knih. Texty knih se ukládají ve formátu HTML, což usnadňuje a urychluje jejich načítání a zobrazování. Klient po obdržení dat od serveru integruje hotový HTML obsah do příslušného kontejneru, což je podrobně popsáno v kapitole 6.4.1. Proces převodu obsahu knihy do HTML formátu při každém otevření uloženého na serveru souboru by byl časově mnohem náročnější.

Druhá databáze slouží jako slovník a uchovává všechna kliknutá slova a jejich vícejazyčné překlady. Vytvořil jsem ji kvůli měsíčnímu limitu 2 000 000 přeložených znaků zdarma v Azure AI Translatoru. Když uživatel potřebuje přeložit slovo, které je již uloženo v databázi, server místo opakovaných dotazů na Azure AI Translator poskytne slovo s jeho překlady přímo z databáze. Takový přístup šetří dostupné znaky a umožňuje nabízet aplikaci zdarma.

Tyto dvě databáze jsem rozdělil zvlášť, abych mohl používat druhou databázi jako nezávislý slovník s překlady pro jiné účely nebo v jiných projektech. Nenavrhoval jsem celé databáze na začátku vývoje, ale použil jsem přístup Code First společně s nástrojem Entity Framework. To znamená, že vývoj databází probíhal současně s vývojem aplikace, a pomocí databázových migrací jsem přidával tabulky, pole a upravoval strukturu databáze. Diagram entitních vztahů obou databází je ukázán v příloze 1.

6.1.1 Code First přístup

Code First přístup znamená, že struktura databáze je vytvořena a spravována prostřednictvím programového kódu, nikoliv pomocí grafických rozhraní. Během vývoje jsem tvořil třídy v kódu, které reprezentují tabulky v databázi. Tyto třídy obsahují vlastnosti, které reprezentují sloupce v tabulkách. Pro správu změn struktury databáze jsem používal migrace, které představují kód, měnící strukturu databáze a umožňující vytvářet, upravovat a odstraňovat tabulky a jejich sloupce. (T Mahachi, 2023)

O Entity Frameworku jsem slyšel už dříve a chtěl jsem se naučit, jak ho používat v praxi. Tento přístup byl vhodný i z toho důvodu, že nebyl jsem jistý, jestli se mi podaří naprogramovat určitou funkcionalitu a zda bude fungovat přesně tak, jak jsem předpokládal. Proto jsem nejprve vytvářel třídy a implementoval metody a až poté pomocí Entity Frameworku mapoval tyto třídy a jejich vztahy do databáze. Metoda pro mapování aplikační databáze je zobrazena na výpisu kódu 4.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);
    modelBuilder.Entity<AbstractBook>(entity => { entity.HasKey(ab => ab.Id); });
    modelBuilder.Entity<BookCover>(entity =>
    {
        entity.HasKey(bc => bc.Id);
        entity.HasOne<AbstractBook>()
            .WithOne(ab => ab.BookCover)
            .HasForeignKey<BookCover>(bc => bc.BookId);
    });
    modelBuilder.Entity<EpubBook>().ToTable("EpubBooks");
    modelBuilder.Entity<PdfBook>().ToTable("PdfBooks");
    modelBuilder.Entity<TxtBook>().ToTable("TxtBooks");

    modelBuilder.Entity<EpubBook>()
        .Property(b => b.Id)
        .HasColumnName("Id");
    modelBuilder.Entity<PdfBook>()
        .Property(b => b.Id)
        .HasColumnName("Id");
    modelBuilder.Entity<TxtBook>()
        .Property(b => b.Id)
        .HasColumnName("Id");
}

public DbSet<AbstractBook> AbstractBooks { get; set; }
public DbSet<EpubBook> EpubBooks { get; set; }
public DbSet<PdfBook> PdfBooks { get; set; }
public DbSet<TxtBook> TxtBooks { get; set; }
public DbSet<BookCover> BookCovers { get; set; }
public DbSet<BookSection> BookSections { get; set; }
public DbSet<SavedWord> SavedWords { get; set; }
```

Výpis kódu 4: Mapování aplikační databáze

Zdroj: vlastní zpracování (2024)

Metoda pro mapování databáze slovníku je zobrazena na výpisu kódu 5.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);

    modelBuilder.Entity<WordWithTranslations>(entity =>
    {
        entity.HasKey(twr => twr.Id);
        entity.HasMany(wt => wt.Translations)
        .WithOne()
        .HasForeignKey(wt => wt.WordId);
    });

    modelBuilder.Entity<WordTranslation>(entity =>
    {
        entity.HasKey(wt => wt.Id);
    });
}

public DbSet<WordWithTranslations> Words { get; set; }
public DbSet<WordTranslation> Translations { get; set; }
```

Výpis kódu 5: Mapování databáze slovníku

Zdroj: vlastní zpracování (2024)

6.1.2 TPT mapovací schéma

Použití mapovacího schématu Table-per-type (TPT) znamená, že každý typ entity je mapován do své individuální tabulky. Vlastnosti specifické pro základní nebo odvozené typy jsou uloženy v tabulce, která odpovídá tomuto konkrétnímu typu. Tabulky reprezentující odvozené typy také ukládají cizí klíč pro spojení se základní tabulkou. (Microsoft, 2023c)

Použití TPT schématu je zobrazeno na diagramu entitních vztahů aplikační databáze v příloze 1. Na tomto diagramu jsou zobrazeny samostatné tabulky *EpubBooks*, *PdfBooks*, *TxtBooks* a základní tabulka *AbstractBooks*, což odpovídá vzoru TPT, ve kterém má každý typ knihy svou vlastní tabulku s unikátními poli a abstraktní třída má svoji tabulku. Nicméně, místo propojení základní a odvozené tabulky pomocí cizího klíče, je použit sdílený primární klíč.

6.2 Uživatelské rozhraní

Jak bylo popsáno v kapitole 5.1.3, pro uživatelské rozhraní je použita knihovna Radzen.Blazor, která poskytuje hotové komponenty pro framework Blazor. Tato knihovna nabízí základní motivy, z nichž jsem vybral motiv s názvem *Material*, který má fialovou barvu jako základní. V této podkapitole je ukázáno použití komponent knihovny

Radzen na příkladu rozložení záhlaví, které je rozděleno na 3 části – levou, pravou a střední.

V levé části je umístěno logo Ekonomické fakulty Jihočeské univerzity jako odkaz na webovou stránku fakulty. Pravá část obsahuje odkaz na nápovědu, nastavení účtu a možnost odhlášení z aplikace. Do středu jsou umístěny odkazy na dvě hlavní části aplikace, pojmenované jako *Reading* a *Learning*. V záložce *Reading* může uživatel nahrát knihy a otevřít je. V sekci *Learning* může uživatel spravovat svůj seznam uložených slov a procvičovat je. Jak vypadá kompletní záhlaví aplikace, je ukázáno na obrázku 3.

Obrázek 3: Záhlaví aplikace LinguaReader

Zdroj: LinguaReader (2024)

Záhlaví se skládá ze sloupců, které obsahují řádky. Protože samotný sloupec nelze rozdělit, je nutné do něj vložit řádek, který poté lze rozčlenit na 12 menších sloupců. Rozložení sloupců ve střední části záhlaví je ukázáno na výpisu kódu 6. Řádek je reprezentován komponentou *RadzenRow* a je rozdělen na dva stejně velké sloupce, reprezentované komponentou *RadzenColumn*. Tyto sloupce jsou vložené do komponenty řádku a mají nastavenou vlastnost *Size* na hodnotu 6. Každý sloupec obsahuje odkaz – komponentu *RadzenLink*, která má vlastnost *Path* definující cestu a vlastnost *Text*, určující zobrazený text odkazu. Cesty jsou uloženy jako konstanty ve statické třídě *Routes*.

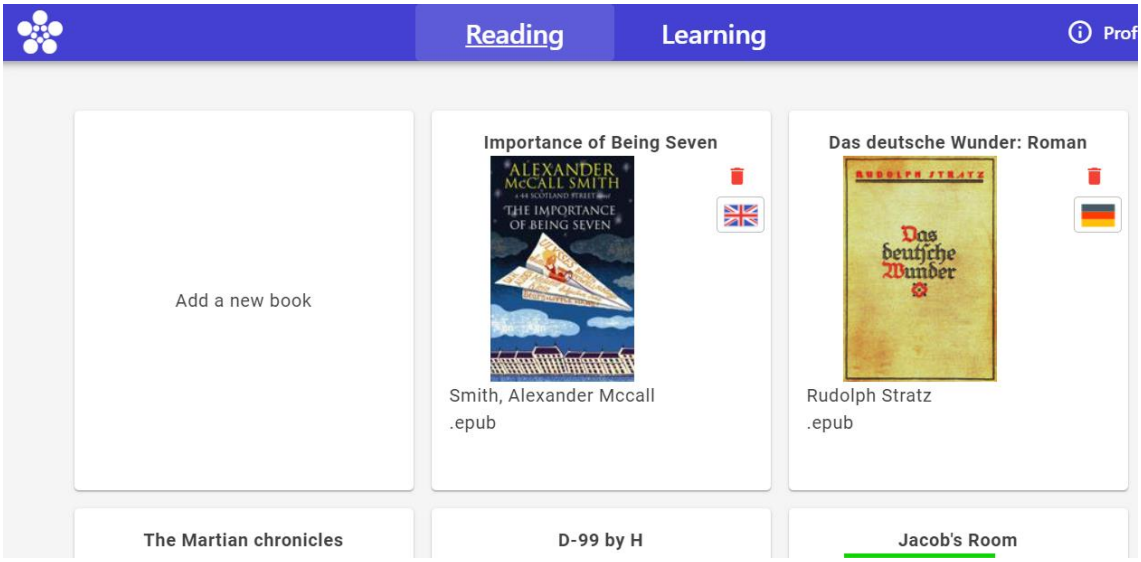
```
<RadzenRow Gap="0" AlignItems="AlignItems.Center" class="h-100 w-100">
    <RadzenColumn Size="6" class="h-100">
        <NavLink>
            <RadzenLink class="h-100 flex-center h4" Path="@Routes.Reading"
Text="Reading" />
        </NavLink>
    </RadzenColumn>
    <RadzenColumn Size="6" class="h-100">
        <NavLink>
            <RadzenLink class="h-100 flex-center h4" Path="@Routes.Learning"
Text="Learning" />
        </NavLink>
    </RadzenColumn>
</RadzenRow>
```

Výpis kódu 6: Kód rozložení prostřední části záhlaví

Zdroj: vlastní zpracování (2024)

6.3 Zpracování nahraných uživatelem souborů

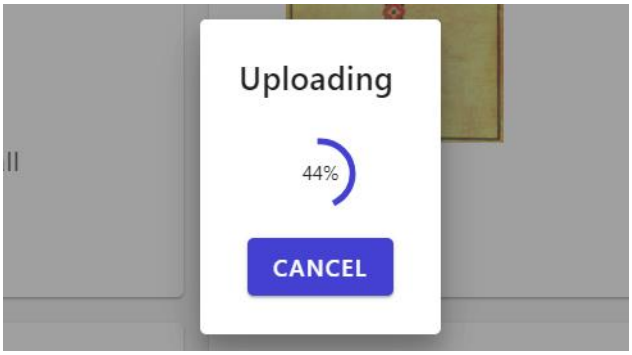
Stránka *Reading* v aplikaci obsahuje Blazor komponentu s názvem *Library*. Tato komponenta zobrazuje uživatelem nahrané knihy a má funkcionalitu pro nahrání dalších knih pomocí tlačítka s textem *Add new book* na začátku seznamu. Proces nahrání knihy představuje extrakci informací a obsahu z knihy a následné uložení získaných dat do databáze jako řádek, který má HTML strukturu. Jak vypadá komponenta *Library*, je ukázáno na obrázku 4. Diagram činností stránky *Reading* je zobrazen v příloze 2.



Obrázek 4: Komponenta Library v LinguaReader

Zdroj: LinguaReader (2024)

Po kliknutí na tlačítko *Add new book* se v aplikaci zobrazí okno, ve kterém uživatel musí vybrat soubor k nahrání. Povolené formáty souborů jsou EPUB, PDF a TXT. Po zvolení souboru aplikace zahájí jeho zpracování a uživateli se zobrazí okno s procesem načítání, které je ukázáno na obrázku 5.



Obrázek 5: LinguaReader: Ukázka procesu načítání knihy

Zdroj: LinguaReader (2024)

Pro přístup k datům z nahraného souboru volá aplikace metodu *GetMemoryStreamFromInput* a předává do ní soubor jako parametr *e*. Metoda ukládá soubor do paměti pomocí třídy *MemoryStream* a vrací soubor bajtů uložený v paměti. Volání této metody a přiřazení vraceného objektu je ukázáno na výpisu kódu 7.

```
MemoryStream? memoryStream = await GetMemoryStreamFromInput(e);
```

Výpis kódu 7: Volání metody pro uložení souboru do paměti

Zdroj: vlastní zpracování (2024)

Počáteční část zpracování, která je popsána výše, je stejná pro všechny formáty souborů, ale proces vytažení dat a obsahu se liší v závislosti na odlišné struktuře formátů EPUB, PDF a TXT. Pro každý formát je implementována zvláštní metoda, kterou aplikace volá na základě zjištěné přípony nahraného souboru. Další podkapitoly popisují tyto metody a způsoby, jakými aplikace přistupuje k datům ze získaného objektu třídy *MemoryStream* pomocí příslušných knihoven. Proces vytažení dat je zobrazen na diagramu činnosti v příloze 2. Diagram tříd, na kterém jsou uvedeny třídy použité v dalších podkapitolách, je ukázán v příloze 3.

6.3.1 EPUB formát

Pokud je přípona nahraného souboru EPUB, ke čtení souboru bajtů je použita knihovna VersOne.Epub. Na výpisu kódu 8 je ukázáno, jak aplikace získává objekt třídy *EpubBookRef*, která představuje knihu s jejími metadaty.

```
EpubBookRef epubBook = await EpubReader.OpenBookAsync(memoryStream);
```

Výpis kódu 8: Použití knihovny VersOne.Epub

Zdroj: vlastní zpracování (2024)

Aplikace na straně klienta dále zpracovává získaná data a přiřazuje je objektu třídy *EpubBook*, která dědí ze třídy *AbstractBook*. Metadata, jako jsou například informace o autorovi a název, se přiřazují objektu třídy *BookCover*, která je vnořena do třídy *AbstractBook*. Obrázek z obalu se neukládá do databáze v původním rozměru, ale jeho rozměry se mění na 200 pixelů šířky a výšky pomocí knihovny SixLabors.ImageSharp.

Text EPUB knihy se skládá z několika sekcí, jež představují jednotlivé kapitoly. Text jedné sekce je HTML souborem, který obsahuje odkazy na soubory obrázků a CSS stylů. Kvůli tomu, že se texty knih ukládají do databáze, bylo nutné před uložením tyto odkazy

upravit. Aby se v textu knihy správně zobrazovaly obrázky a CSS styly, aplikace rozebírá HTML kód pomocí knihovny AngleSharp a každou cestu mění na *data URL* odkaz, jehož syntaxe je uvedena na výpisu kódu 9. Obrázky a soubory se styly jsou uloženy jako *base64* zakódované řetězce přímo uvnitř HTML kódu.

```
data:[<mediatype>][;base64],<data>
```

Výpis kódu 9: Syntaxe data URL

Zdroj: Mozilla Foundation (2023a)

Nakonec klient odesílá objekt naplněný daty na server aplikace, kde knihovna ASP.NET Core Identity zjistí uživatele, který nahrál zpracovanou knihu, a vyplní se poslední parametr v obdrženém objektu EPUB knihy – id tohoto uživatele. Poté server uloží nahranou knihu do databáze a vrátí informace o výsledku operace. Pokud vše proběhlo v pořádku, klient přidá obal uložené knihy do knihovny na klientském počítači.

6.3.2 PDF formát

Vytažení textu z PDF souborů se zachováním struktury – odstavců, mezer a velikosti písma bylo pro mě největší výzvou během celé doby vývoje této aplikace. Vyzkoušel jsem množství knihoven zaměřených na vytažení textu z PDF souboru. Mým cílem nebyla jen extrakce textu, ale i zachování jeho formátování, alespoň odstavců a tučného písma. Při hledání řešení na internetu jsem často narazil na názory, že to je nemožné nebo neexistuje optimální řešení, které bude fungovat pro všechny soubory. Hledal jsem knihovny nejen v jazyce C#, ale i v JavaScriptu a Pythonu, a nakonec jsem vybral dvě možnosti: C# knihovnu iText7 a JavaScriptovou knihovnu pdf.js.

Knihovna pdf.js, vyvinutá společností Mozilla v jazyce JavaScript, je součástí webového prohlížeče Firefox, kde zajišťuje zobrazení PDF dokumentů. Funguje tak, že PDF stránku rozdělí do dvou vrstev: první vrstva představuje obrázek stránky PDF dokumentu, druhá vrstva je text této stránky, překrývající obrázek. Text je umístěn pomocí pozicování v CSS a pro definici řádků se používají HTML prvky *span*. Tato knihovna nemůže měnit rozložení textu, ale pouze zobrazuje stránku dokumentu tak, jak je. Někdy algoritmus neidentifikuje řádky a fráze správně, což vede k problémům s uživatelským rozhraním, jako jsou přeskočené řádky nebo odstavce při výběru textu. Na fóru podpory komunity pdf.js na GitHubu je otevřeno více než 90 otázek týkajících se pouze výběru textu. (Apryse, 2020)

Vzhledem k těmto nedostatkům a potřebě umístit obsah na stránce jako text, ne jako obrázek, jsem zvolil knihovnu iText7. Tato knihovna nabízí možnost vytahovat obsah PDF souboru jako prostý text a umožňuje vytvořit vlastní strategii pro zpracování toho textu.

Aplikace používá knihovnu iText7 v případě, že přípona nahraného souboru je PDF. Pomocí třídy *PdfReader* této knihovny budou přečteny bajty z objektu *MemoryStream* a následně bude vytvořen objekt třídy *PdfDocument*, který představuje nahraný PDF soubor. Tento proces je ukázán na výpisu kódu 10.

```
PdfReader reader = new PdfReader(memoryStream);
PdfDocument pdfDoc = new PdfDocument(reader);
```

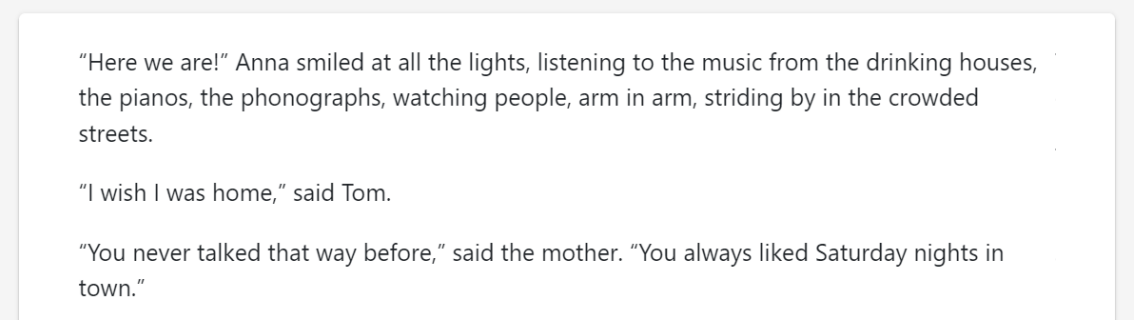
Výpis kódu 10: Použití knihovny iText7

Zdroj: vlastní zpracování (2024)

Dále se pro zpracování textu z instance třídy *PdfDocument* používá vlastní strategie v rámci knihovny iText7. Tento přístup je založen na původní strategii, která identifikuje symboly a řádky pomocí vektorů, a je rozšířen o podmínky, které doplňují vytažený text o HTML prvky *p* a *b* pro označení odstavců a tučného písma v textu.

Pro rozdělení textu na odstavce se podle tohoto přístupu přidává na začátek prázdného řetězce HTML prvek *p*. Poté funkcionalita původní strategie zpracovává několik znaků textu knihy a ukládá je do řetězce. Tento proces se opakuje, dokud strategie nezjistí, že řádek končí a na jeho konci je tečka, otazník nebo vykřičník, v tomto případě se předpokládá, že odstavec skončil a do řetězce se přidává uzavírací HTML prvek *p*. V tomto kontextu se kontrolují poslední 3 symboly, protože řádek může končit uvozovkami a hledaný symbol bude před nimi. Při zpracování dalšího řádku se přidává znovu otevírací HTML prvek *p* do řetězce, který už obsahuje vytažený odstavec. Tímto způsobem se celý text upravuje a dělí na odstavce.

Po zpracování celé knihy se přiřazuje její text a všechna data objektu třídy *PdfBook*. Dále klient odesílá objekt naplněný daty na server, který přiřazuje vlastníka knihy a ukládá ji do databáze. Jak vypadá zpracovaný text při otevření PDF knihy, je vidět na obrázku 6.



Obrázek 6: LinguaReader: Zpracovaný text z PDF souboru

Zdroj: LinguaReader (2024)

6.3.3 TXT formát

Pokud je přípona nahraného souboru TXT, vytvoří aplikace prázdný řetězec a na jeho začátek přidá HTML značku *div* s id *txt-content*. To umožní při zobrazení textu vybrat všechny prvky uvnitř *div* kontejneru a upravit jejich CSS styly, aby text na stránce vypadal lépe. Dále aplikace zpracovává soubor po řádcích. Pokud je řádek prázdný, přidá HTML značku *br*, což odpovídá prázdnému řádku. V případě, že řádek obsahuje text, umístí aplikace tento text do HTML značek *p*. Tento přístup zajišťuje, že text na stránce bude zobrazen co nejpodobněji tomu, jak by vypadal v textovém editoru. Každý řádek bude po zpracování přidán do řetězce a bude aktualizován postup načítání. Po dokončení vytažení textu přidá aplikace na konec řetězce zavírací značku *div*, přiřadí název nahraného souboru a pošle na server objekt naplněný daty. Na serveru se zjistí uživatel, který nahrál soubor, a data se uloží do databáze.

6.4 Zobrazení knihy v prohlížeči

Po nahrání knihy uživatel potřebuje vybrat jazyk, ve kterém je kniha napsaná, a až pak aplikace umožní tuto knihu otevřít a zobrazí její text. Text vytažený z nahraného souboru je uložen v databázi ve formátu HTML, a v tuto chvíli se pouze načte na stránku. Aby to dobře vypadalo a připomínalo čtení knihy, je text rozdělen na stránky, aby uživatel viděl jednu stránku knihy a mohl se přesunout na následující nebo případně na předchozí stránku, pomocí tlačítek. Pro dosažení tohoto cíle byl použit JavaScript a CSS styl *column-count*, který dělí text na sloupce, kde každý sloupec představuje jednu stránku.

6.4.1 Zobrazení textu knihy

Po otevření knihy v libovolném formátu se v C# pomocí knihovny JSInterop volá JavaScriptová metoda, které se jako parametr předává text knihy pro zobrazení. Rozdíl v zobrazení textu pro formáty EPUB, PDF a TXT spočívá v tom, že u EPUB knihy se

kód metody volá pro každou sekci, zatímco pro PDF a TXT soubory pouze jednou při otevření knihy. Nejprve metoda v JavaScriptu připraví HTML prvek *div* s id kontejneru, který slouží k zobrazení textu knihy. Poté aplikace vytvoří z obsahu knihy kompletní HTML dokument a vloží ho do tohoto *div* kontejneru. Dalším krokem je zjištění šířky a výšky kontejneru a na základě těchto rozměrů se upraví velikost obrázků. Nakonec rozdělí text, který je dosud vložen jako jedna dlouhá stránka, na sloupce a vrátí počet stránek. JavaScriptová metoda, která zpracovává text pro zobrazení je ukázána na výpisu kódu 11.

```
container = shadow.getElementById("container");
container.innerHTML = "";
var bookDocument = await addBookOnPage(htmlString);
if (bookDocument) {
    await container.appendChild(bookDocument);
    await adjustImages(container, container.clientHeight, container.clientWidth);
    pagesCount = separateBookDocument(container);
}
return pagesCount;
```

Výpis kódu 11: Metoda pro zpracování textu knihy

Zdroj: vlastní zpracování (2024)

Text se dělí na stránky následujícím způsobem. Nejprve zjistí celkovou výšku vloženého textu a rozměry *div* kontejneru. Poté vydělí výšku celého textu výškou tohoto *div* kontejneru, čímž získá počet stránek, na který je třeba ten text rozdělit. Dále nastaví potřebné CSS styly a hodnotu stylu *column-count* na zjištěný počet stránek. Metoda pro rozdělení textu na sloupce je ukázána na výpisu kódu 12.

```
async function separateBookDocument(container) {
    body = container.querySelector("html body");
    var totalHeight = body.offsetHeight;
    var containerHeight = host.clientHeight;
    var containerWidth = host.clientWidth;
    var pagesCount = Math.floor(totalHeight / containerHeight) + 1;
    body.style.margin = 0;
    body.style.width = (containerWidth * pagesCount) + "px";
    body.style.columnCount = pagesCount;
    body.style.position = "relative";
    body.style.columnGap = 0 + "px";
    body.style.columnFill = "balance";
    return pagesCount;
}
```

Výpis kódu 12: Rozdělení textu na sloupce

Zdroj: vlastní zpracování (2024)

Každá sekce EPUB knihy je zvláštní HTML dokument se svými CSS styly. Jejich zobrazení na stránce způsobilo problém, který spočíval v tom, že tyto styly ovlivňovaly

všechny HTML na stránce. Tento problém byl vyřešen pomocí Shadow DOM, který podle společnosti Mozilla Foundation (2024c) umožňuje připojit k HTML prvku další strom DOM a skrýt tak jeho vnitřní části před JavaScriptem a CSS styly na stránce. V aplikaci je *div* kontejner s textem sekce EPUB knihy se svými CSS styly umístěn uvnitř Shadow DOM, zatímco mimo něj jsou HTML prvky stránky, které nebudou ovlivněny CSS kódem uvnitř. Výsledkem je, že CSS styly *div* kontejneru ovlivňují pouze HTML prvky textu knihy.

Jako alternativa k Shadow DOM bylo možné použít iframe[12], což jsem také vyzkoušel, ale tento přístup způsobil další problémy jako omezené použití JavaScriptu, neschopnost identifikovat kliknutí na slovo v C# kódu a problémy při rozdělení textu na stránky.

6.4.2 Optimalizace – porovnání C# a JS

Kód popsaný v předchozí podkapitole je naprogramován v JavaScriptu, ale mohl by být naprogramován i v C#. Zobrazení textu knihy je důležitá funkcionalita, která se v aplikaci odehrává velmi často. V souladu s nefunkčním požadavkem číslo 5 byla vyvinuta shodná metoda, naprogramovaná v jazyce C#. Tato podkapitola se zabývá změřením a porovnáním času potřebného pro rozdělení textu na jednotlivé stránky v těchto dvou programovacích jazycích.

Na rozdíl od JavaScriptu neumí C# přímo pracovat s HTML obsahem, proto je potřeba pro přístup k prvkům HTML a zjištění jejich rozměrů obsah předem takzvaně rozparsovat. Tento rozbor HTML obsahu pomocí knihovny AngleSharp je ukázán na výpisu kódu 13.

```
var parser = new HtmlParser();
var doc = await parser.ParseDocumentAsync(content);
if (document.Head is not null && doc.Head is not null)
{
    document.Head.InnerHtml = doc.Head.InnerHtml;
    document.Body.InnerHtml = doc.Body.InnerHtml;
}
```

Výpis kódu 13: Rozbor HTML obsahu v C#

Zdroj: vlastní zpracování (2024)

Kód v jazyce C# také v některých případech potřebuje volat doplňující JavaScriptové metody pomocí knihovny JSInterop. To se děje například při měření výšky HTML prvku na stránce, pro následný výpočet počtu stránek nebo při vkládání zpracovaného HTML

[12] Další informace o HTML prvku iframe: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe

kódu do určeného prvku na webové stránce. V ostatních případech je kód podobný. Výpočet počtu stránek a nastavení stylů pro rozdělení textu na sloupce v C# jsou ukázány na výpisu kódu 14.

```
actualSectionPagesCount = (int)Math.Ceiling((double)offsetHeight / maxHeight);
document.Body.SetStyle($"padding: 10; margin: 0; width: {900 * actualSectionPages-
Count}; height: {maxHeight}; column-count: {actualSectionPagesCount}");
```

Výpis kódu 14: Rozdělení obsahu na sloupce v C#

Zdroj: vlastní zpracování (2024)

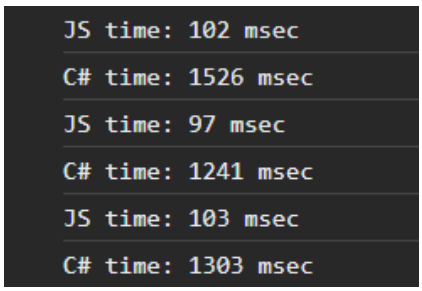
Pro měření času byla použita knihovna System.Diagnostics a její třída *Stopwatch*. Výsledky se vypisují do konzole v milisekundách a na základě změřeného času je možné se rozhodnout, který z jazyků plní úlohu rychleji. Spouštění stopek a volání metod pro oba programovací jazyky je ukázáno na výpisu kódu 15.

```
//JS
Stopwatch stopwatch = Stopwatch.StartNew();
actualSectionPagesCount = await JS.InvokeAsync<int>("divideAndSetHtml", Secti-
ons[2].Text);
stopwatch.Stop();
Console.WriteLine("JS time:" + stopwatch.ElapsedMilliseconds + " msec");
//C#
Stopwatch stopwatch2 = Stopwatch.StartNew();
await DivideAndSetHtml(Sections[2].Text);
stopwatch2.Stop();
```

Výpis kódu 15: Testování rychlosti C# a JavaScript

Zdroj: vlastní zpracování (2024)

Výsledky testování jsou ukázány na obrázku 7. JavaScript plní tuto úlohu mnohem rychleji, což dává smysl, protože není potřeba zvláště rozebírat HTML obsah ani se přepínat mezi C# a JavaScriptem pomocí knihovny JSInterop. Na základě výsledků testování je vidět, že použití jazyka JavaScript pro rozdělení textu knihy na stránky bylo správným rozhodnutím.



Obrázek 7: Výsledky testování v konzoli prohlížeče Google Chrome

Zdroj: Google Chrome (b. r.)

6.4.3 Přepínání stránek

Pro správné zobrazení textu získaného z PDF a TXT souborů a textu rozděleného na sekce z EPUB souboru bylo vytvořeno rozhraní *IDisplayBook*. Obsahuje základní vlastnosti a metody, které musí obsahovat každá třída implementující zobrazení knihy v libovolném formátu. Rozhraní *IDisplayBook* je ukázáno na výpisu kódu 16.

```
public interface IDisplayBook
{
    public string BookId { get; set; }
    public string BookLanguage { get; set; }
    public int CurrentPageNumber { get; set; }
    public int PagesCount { get; set; }
    void NextPage();
    void PreviousPage();
    void JumpToPage(int? pageNumber);
}
```

Výpis kódu 16: Rozhraní IDisplayBook

Zdroj: vlastní zpracování (2024)

Pro text získaný z PDF a TXT souborů jsou používány pouze metody a vlastnosti rozhraní *IDisplayBook*. Implementace metody *NextPage* v tomto případě znamená posunutí obsahu *div* kontejneru o hodnotu rovnající se jeho šířce.

Pro text z EPUB souboru rozdělený na sekce je navíc potřeba tyto sekce přepínat. Kontrola probíhá pomocí vlastností *FirstPage* a *LastPage* třídy *BookSection*, které udávají, na jaké stránce sekce začíná a končí. Při posunutí na další stránku se kontroluje, zda je číslo stránky vyšší než hodnota vlastnosti *LastPage* aktuální sekce. Pokud ano, volá se metoda *NextSection*, která vloží do *div* kontejneru další sekci knihy, jak už bylo popsáno v kapitole 6.4.1. V případě, že další stránka je součástí aktuální sekce, posune se obsah kontejneru o hodnotu odpovídající počtu již přečtených stránek v této sekci. Proto se od čísla aktuální stránky odečítá číslo stránky, kterou sekce začíná. V JavaScriptu se výsledné číslo vynásobí šířkou kontejneru a text knihy se posune doleva na tuto hodnotu. Implementovaná metoda *NextPage* pro EPUB je ukázána na výpisu kódu 17.

```
public async void NextPage()
{
    if (CurrentPageNumber != PagesCount)
    {
        CurrentPageNumber++;
        if (CurrentPageNumber > Sections[currentSectionNumber].LastPage)
            NextSection();
        else
            await JS.InvokeVoidAsync("nextPage", CurrentPageNumber -
Sections[currentSectionNumber].FirstPage);
    }
}
```

Výpis kódu 17: Metoda NextPage pro EPUB knihu

Zdroj: vlastní zpracování (2024)

Tímto způsobem aplikace zobrazuje obsah knihy stránku po stránce, což uživatelům umožňuje přejít na další nebo předchozí stránku, jako kdyby uživatel četl klasickou knihu.

6.5 Identifikace kliknutí na slovo

Při kliknutí na slovo v textu je pomocí kódu v jazyce C# zachyceno kliknutí na celý *div* kontejner, který se předává jako parametr JavaScriptové metodě. Tato metoda určuje konkrétní slovo a jeho pozici v textu pro následné zobrazení okna s překlady přímo nad slovem. Po zjištění těchto údajů vrací JavaScript tyto informace zapouzdřené v objektu třídy *WordInfo,* zpět do C#. V tomto případě je použit programovací jazyk JavaScript z důvodu, že pro výběr kliknutého slova v textu je nutné pracovat s metodou *getSelection*[13], která je součástí API prohlížeče. Jak uvádí Brind (2023), Blazor WebAssembly ve výchozím nastavení nemá přístup k této funkcionalitě prohlížeče. Volání JavaScriptové metody v C# kódu pomocí JSInterop je ukázáno na výpisu kódu 18.

```
wordInfo = await JS.InvokeAsync<WordInfo>("getSelectedWord", host);
```

Výpis kódu 18: Volání JavaScriptové metody ze C# kódu

Zdroj: vlastní zpracování (2024)

Pro identifikaci slov v různých jazycích jsou definovány regulární výrazy, které zahrnují speciální písmena těchto jazyků. Metoda *getSelectedWord* určuje kliknuté slovo na základě regulárního výrazu pro jazyk otevřené knihy. Nejprve metoda ověří, zda byl kliknut text. Pokud ano, zjistí přesný bod kliknutí, a poté začne identifikovat slovo. Od tohoto bodu se posouvá doleva po jednom textovém symbolu, dokud není na začátku řádku, a zároveň označený rozsah odpovídá regulárnímu výrazu – neobsahuje mezeru.

[13] Další informace o Web API metodě: https://developer.mozilla.org/en-US/docs/Web/API/Window/getSelection

Po dokončení *while* cyklu, metoda pravděpodobně označí o jeden symbol navíc, proto ještě jednou zkontroluje označený rozsah, a pokud tento neodpovídá regulárnímu výrazu, posune začátek rozsahu o jeden symbol zpět. Nalezení začátku slova je ukázáno na výpisu kódu 19. Poté se podobným způsobem nalezne konec rozsahu a označí se slovo kliknuté v textu.

```
//Finds the start point of a clicked word
while ((range.startOffset > 0) && range.toString().match(wordRegexp)) {
    range.setStart(node, (range.startOffset - 1));
}
if (!range.toString().match(wordRegexp)) {

    range.setStart(node, range.startOffset + 1);
}
```

Výpis kódu 19: Nalezení začátku kliknutého slova

Zdroj: vlastní zpracování (2024)

Metoda *getSelectedWord* rovněž zjišťuje pozici slova v textu pro následné zobrazení okna s překlady přímo nad kliknutým slovem. Toho dosahuje pomocí CSS vlastností *top* a *left*, které udávají vzdálenost slova od horního a levého okraje stránky. Pozici od horního okraje metoda získá odečtením výšky záhlaví a mezery 5 pixelů mezi slovem a oknem s překlady od hodnoty vlastností *top*. K této hodnotě se přičte hodnota vlastnosti *scrollTop*, která říká, o kolik je stránka posunutá v případě, že se obsah nezobrazí správně a bude mít posuvník. Pro určení pozice zleva se k hodnotě *left* přičte hodnota šířky označeného slova rozdělená napůl, aby střed okna s překlady odpovídal středu slova a aby bylo okno správně umístěno. Nakonec metoda vrací zjištěné slovo a jeho souřadnice *top* a *left*. Kód pro zjištění pozice je ukázán na výpisu kódu 20.

```
//Gets start position to draw translator window (right upon the clicked word)
var rangePosition = range.getBoundingClientRect();
var top = rangePosition.top - rzHeader.clientHeight + rzBody.scrollTop - 5;
var left = rangePosition.left + (rangePosition.width / 2);
resolve({ word, top, left });
```

Výpis kódu 20: Zjištění pozice slova a vracení objektu

Zdroj: vlastní zpracování (2024)

Po dokončení metody *getSelectedWord* je kliknuté slovo označeno červeně a nad ním se zobrazí kostra budoucího okna s překlady. Dokud aplikace nezíská data od serveru, slouží kostra jako náhrada okna s překlady, bliká a naznačuje uživateli, že probíhá načítání. Komponenta kostry okna je zobrazena na obrázku 8.

Obrázek 8: LinguaReader: Element kostry okna s překlady

Zdroj: LinguaReader (2024)

6.6 Přeložení slova

Metoda popsaná v předchozí podkapitole získává slovo, na které uživatel klikl, a vrací ho zpět do kódu v C#, který následně odesílá toto slovo na server a čeká na vrácení objektu třídy *WordWithTranslations*. Pro překlad slov byla použita služba Azure AI Translator a její slovníkový modul. Diagram vztahu entit pro databázi slovníku je zobrazen v příloze 1. Třídy pro překlad slov, použité v této podkapitole jsou uvedeny v příloze 4.

Pokud už v minulosti nějaký uživatel klikl na stejné slovo, bylo po přeložení uloženo do databázové tabulky *Words*. Překlady pro toto slovo byly uloženy v tabulce *Translations* s cizím klíčem odpovídajícím uloženému slovu. Server nejprve zkusí nalézt obdržené slovo k přeložení v databázi, a pokud se mu to podaří, vrátí nalezené slovo s jeho překlady. Tento proces je ukázán na výpisu kódu 21.

```
WordWithTranslations translatorWordResp = await GetTranslationFromDb(word, bo-
okLang, targetLang);
if (translatorWordResp is not null)
    return Ok(translatorWordResp);
```

Výpis kódu 21: Získání slova z databáze a jeho vrácení

Zdroj: vlastní zpracování (2024)

V případě, že slovo v databázi není, server odešle dotaz pro přeložení do Azure AI Translatoru. Dotaz obsahuje slovo, jazyk, v němž je slovo napsáno, a cílový jazyk, který server získal z nastavení v profilu uživatele. Protože služba Azure AI Translator není veřejná, je nutné v dotazu poslat také tajný klíč a lokaci. Tyto informace jsem obdržel po zřízení služby v účtu v Azure cloudové platformě. Proces vytvoření a odeslání dotazu je ukázán na výpisu kódu 22.

```
private async Task<HttpResponseMessage> ConstructAndSendQuery(string word, string
sourceLang, string targetLang)
{
    string route = $"dictionary/lookup?api-version=3.0&from={sourceLang}&to={tar-
getLang}";
    object[] body = new object[] { new { Text = word } };
    var requestBody = JsonConvert.SerializeObject(body);
    using (var request = new HttpRequestMessage())
    {
        request.Method = System.Net.Http.HttpMethod.Post;
        request.RequestUri = new Uri(_endpoint + route);
        request.Content = new StringContent(requestBody, Encoding.UTF8, "applica-
tion/json");
        request.Headers.Add("Ocp-Apim-Subscription-Key", _key);
        request.Headers.Add("Ocp-Apim-Subscription-Region", _location);
        return await _httpClient.SendAsync(request).ConfigureAwait(false);
    }
}
```

Výpis kódu 22: Vytvoření a odeslání dotazu do Azure AI Translatoru

Zdroj: vlastní zpracování (2024)

Během zjišťování překladů slova může nastat situace, kdy toto slovo již existuje v databázové tabulce *Words*, avšak překlady v požadovaném jazyce pro něj chybí. Kontroluje se to pomocí proměnné *existedWordId*, které server přiřadí hodnotu při pokusu nalézt slovo s překlady v databázi. Pokud již slovo existuje, server ho znovu neukládá, ale místo toho propojí nově zjištěné překlady s existujícím záznamem slova v databázi pomocí cizího klíče *WordId*. V případě, že slovo v databázi není, uloží celý objekt třídy *WordWithTranslations* obsahující slovo a jeho překlady. Proces uložení překladů je ukázán na výpisu kódu 23.

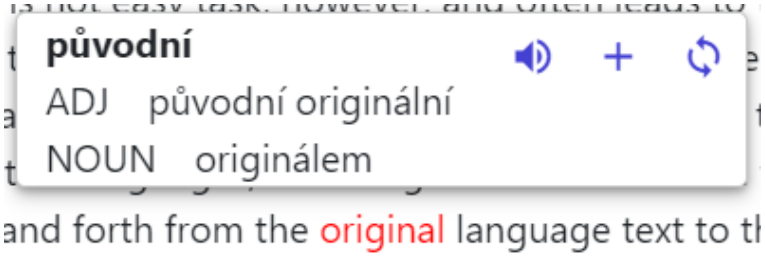
```
foreach (var translation in wordWithTranslations.Translations)
{
    translation.Language = targetLang;
    if (existedWordId != 0)
    {
        translation.WordId = existedWordId;
        _dictionaryDbContext.Translations.Add(translation);
    }
}
if (existedWordId == 0)
    _dictionaryDbContext.Words.Add(wordWithTranslations);
```

Výpis kódu 23: Uložení překladů do databáze

Zdroj: vlastní zpracování (2024)

Po obdržení odpovědi od serveru klient uživateli zobrazí okno s překlady, umístěné přímo nad kliknutým slovem, jak je zobrazeno na obrázku 9. V pravém horním rohu okna se nachází 3 tlačítka. Tlačítko s reproduktorem aktivuje vyslovení kliknutého slova, což je

zajištěno rozhraním *SpeechSynthesisUtterance*[14], dostupným ve většině moderních prohlížečů. Tlačítko s plusem umožní uložení slova a jeho překladů do slovníku uživatele pro pozdější procvičení. Po kliknutí se ikona plusu změní na zaškrtávací symbol, což znamená, že slovo bylo uloženo. Kliknutím na zaškrtávací symbol lze slovo odstranit ze slovníku uživatele. Tlačítko se symbolem synchronizace umožňuje znovu odeslat slovo Azure AI Translatoru pro přeložení a aktualizaci jeho překladů v databázi.



Obrázek 9: LinguaReader: Okno s překlady

Zdroj: LinguaReader (2024)

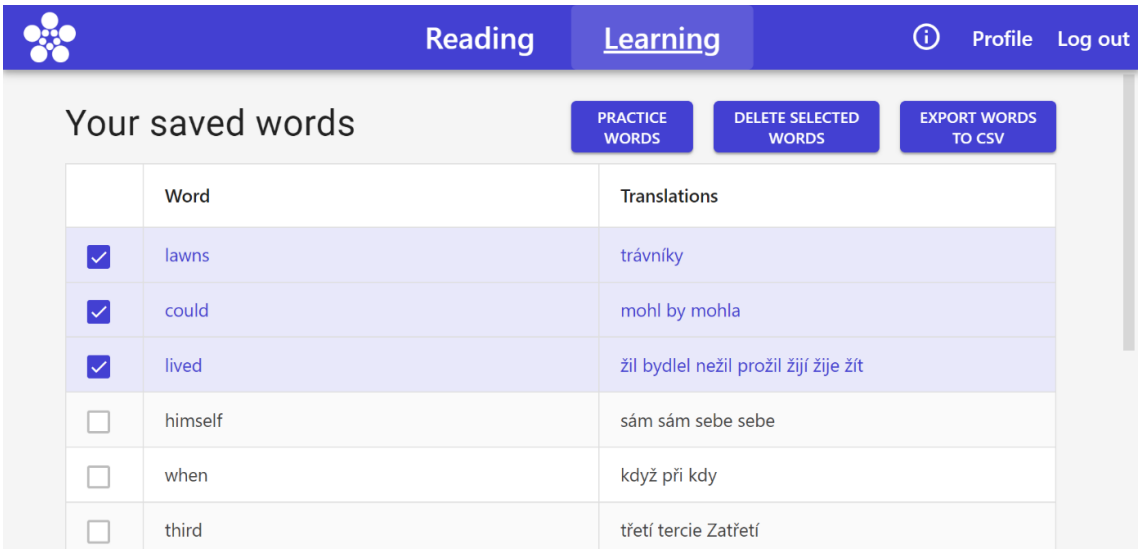
6.7 Slovník uživatele a procvičení slov

Před zahájením implementace této části aplikace byl vytvořen vývojový diagram a navrženy potřebné třídy. Tento přístup usnadnil a urychlil implementaci ve srovnání s ostatními funkcionalitami aplikace. Třídy použité v následujících podkapitolách pro zobrazení a procvičování slov jsou uvedeny na diagramu tříd v příloze 4. Diagram činností pro stránku *Learning* je ukázán v příloze 5.

6.7.1 Slovník uživatele

Stránku se slovníkem obsahujícím slova uložená uživatelem lze otevřít kliknutím na záložku *Learning* v záhlaví aplikace. Klientská část aplikace odešle dotaz na server pro získání všech uložených slov přihlášeného uživatele, jejichž ID jsou uchovávána v aplikační databázi v tabulce *SavedWords*. Po načtení dat se zobrazí seznam, kde první sloupec obsahuje slova v cizím jazyce pro uživatele a druhý sloupec jejich překlady. Uživatel může označit položky v tomto seznamu, nad kterým jsou umístěna tři tlačítka. Stránka *Learning* se seznamem slov je ukázána na obrázku 10.

[14] Další informace o rozhraní SpeechSynthesisUtterance: https://developer.mozilla.org/en-US/docs/Web/API/SpeechSynthesisUtterance



Obrázek 10: Stránka Learning se seznamem slov v LinguaReader

Zdroj: LinguaReader (2024)

Uživatel může vymazat označená slova ze svého slovníku pomocí tlačítka *Delete selected words*. Před kliknutím tlačítka musí označit slova, která chce smazat. Pokud nejsou označena žádná slova, objeví se upozornění a nedojde k žádnému smazání.

Kliknutím na tlačítko *Export words to CSV* bez předchozího označení slov aplikace exportuje celý seznam slov do CSV souboru a stáhne jej na počítač uživatele. Uživatel má také možnost vyexportovat pouze označená slova. Stažený CSV soubor s vlastními slovy může uživatel následně importovat do jiných aplikací pro procvičení. Tento postup byl úspěšně vyzkoušen s aplikací AnkiApp, kde byly na základě importovaného seznamu automaticky vytvořeny flashkarty.

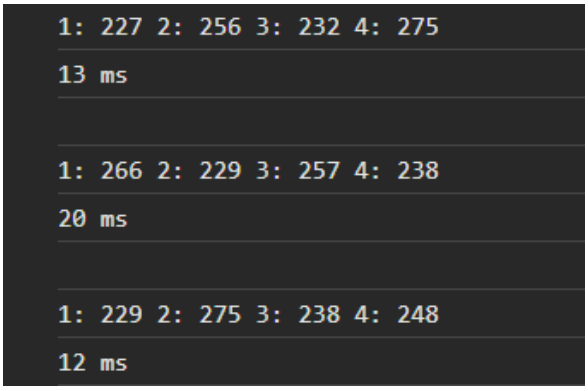
6.7.2 Procvičení slov

Tlačítko s nápisem *Practice Words* umístěné nad seznamem uložených slov spustí cvičení. Po stisknutí tohoto tlačítka aplikace nejprve zkontroluje, zda je ve slovníku uživatele dostatečný počet slov – 10 a více. Pokud je počet uložených slov menší, aplikace zobrazí dialogové okno s informací, že uživatel nemá minimální počet slov potřebný pro cvičení. Pokud je počet slov dostatečný, aplikace odešle dotaz na server, který náhodně vybere 10 slov s překlady ze slovníku uživatele. Server zároveň vybere 3 nesprávné varianty překladu každého vybraného slova, které jsou vybírány ze všech překladů dříve uložených v databázi v tabulce *Translations*.

Pokud by byla slova pro nesprávné odpovědi zvolena náhodně, bylo by nalezení správné varianty pro uživatele příliš jednoduché. Proto do dotazu pro výběr nesprávných odpovědí

z databáze byla přidána podmínka, která zajišťuje, že zvolená slova budou stejného slovního druhu jako správná varianta odpovědi. To znamená, že pokud je správná varianta odpovědi podstatné jméno, nesprávné varianty budou také podstatná jména.

Server vrací klientu seznam naplněný objekty třídy *WordToLearn*. Varianty odpovědí jsou v každém objektu reprezentovány vlastností *VariantsToAnswer*. Správná odpověď je vždy na prvním místě, proto je potřeba tento seznam zamíchat. Provedl jsem testování náhodnosti a rychlosti implementace zamícháním 1000 seznamů. Tento test jsem spustil třikrát. Výsledky tohoto testování jsou uvedeny na obrázku 11. Na prvním řádku jsou zobrazeny pozice správné varianty v seznamu a jejich četnost výskytu na dané pozici po zamíchání, což naznačuje dostatečnou náhodnost. Na druhém řádku je zobrazen čas potřebný pro zamíchání 1000 seznamů.



Obrázek 11: Konzole prohlížeče Google Chrome: výsledky testování zamíchaní

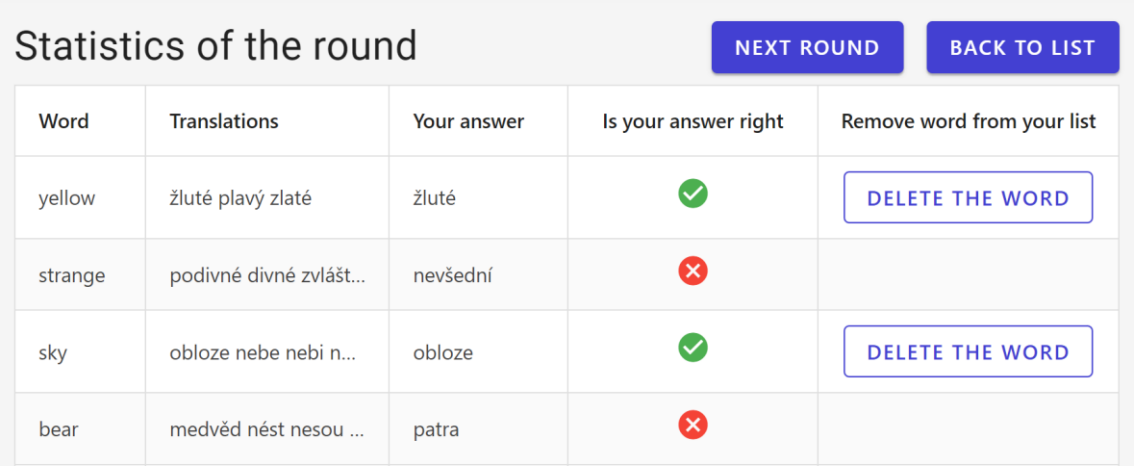
Zdroj: Google Chrome (b. r.)

Jakmile je seznam slov připraven k procvičení, zobrazí aplikace první kartu se čtyřmi možnými variantami odpovědí. Na každé kartě je napsáno slovo v cizím jazyce, který se uživatel učí, a pod kartou jsou ve vybraném mateřském jazyce uživatele zobrazeny 4 možné varianty odpovědí, z nichž 1 je správná. Karta s možnými odpověďmi je ukázána na obrázku 12.

Obrázek 12: Karta s cizím slovem a odpovědí v LinguaReader

Zdroj: LinguaReader (2024)

Po kliknutí na správnou variantu odpovědi ji aplikace označí zelenou barvou. V případě, že vybraná varianta není správná, označí ji červeně a zároveň správnou variantu zeleně. Po obarvení variant čeká aplikace 2 vteřiny, aby si uživatel mohl uvědomit svoji chybu. Poté aplikace zobrazí další kartu a celý proces se opakuje až do konce kola, které se skládá z 10 karet. Jakmile je vybrána odpověď na poslední kartě, vypíše se uživateli statistika celého kola, která je ukázána na obrázku 13.



Obrázek 13: Stránka se statistikou kola cvičení v LinguaReader

Zdroj: LinguaReader (2024)

V seznamu se statistikami uživatel vidí slova, která byla zobrazena na kartách, jejich překlady, svoji odpověď a informaci, zda byla jeho odpověď správná. Slova, na která uživatel odpověděl správně, může ihned odstranit ze svého slovníku pomocí tlačítka umístěného v posledním sloupci tabulky. Nad seznamem se nacházejí dvě tlačítka: jedno pro spuštění dalšího kola cvičení a druhé pro návrat k seznamu slov uživatele.

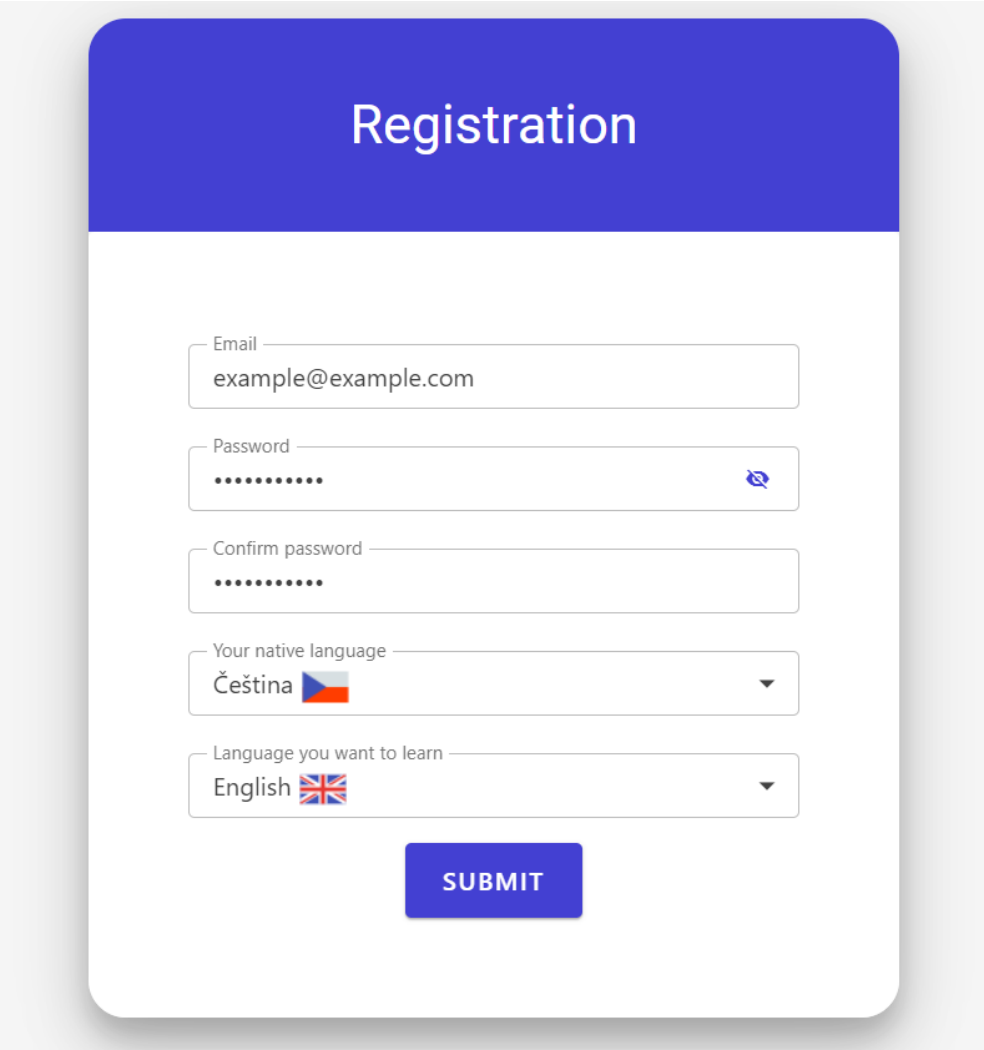
6.8 Registrace a přihlášení uživatelů

Pro identifikaci uživatelů byla nutná registrace a autentizace uživatelů, aby bylo možné propojit jejich účet s knihami, které uživatel nahrává, a se slovy, která si uživatel ukládá pro následné procvičení. Specifická autorizace nebyla potřebná, protože všichni uživatelé mají stejnou roli. Nalezl jsem dva návody na internetu, které doporučují, jak registraci a autentizaci implementovat. Rozhodl jsem se postupovat podle obou návodů a propojit je. Třídy použité v této podkapitole jsou uvedeny v příloze 6.

Z návodu „Blazor WebAssembly Series“ na webu Code Maze (2022a) jsem postupoval podle části „Authentication with Blazor WebAssembly and ASP.NET Core Identity“ od autora Marinka Spasojeviće. Díky tomuto návodu jsem pochopil základní logiku a implementoval funkcionality registrace a autentizace. Pomocí druhého návodu s názvem „ASP.NET Core Identity Series“ na webu Code Maze (2022b), jejíž autorem je také Marinko Spasojević, jsem implementoval obnovení hesla, potvrzení emailu a zablokování uživatele po 5 nesprávných pokusech o zadání hesla. Tyto dva návody nejsou navzájem propojeny, na rozdíl od jejich jednotlivých částí.

6.8.1 Implementace procesu registrace a přihlášení

Registrace slouží pro přidání nových uživatelů do databáze a vyžaduje vyplnění emailové adresy uživatele, hesla a potvrzení hesla. Pro úspěšnou registraci musí heslo obsahovat alespoň 8 znaků, jeden nealfanumerický znak a jednu číslici. Další pole je mateřský jazyk uživatele, do kterého budou překládána slova při čtení knih. Posledním polem je jazyk, který se uživatel chce naučit, v němž bude probíhat procvičování uložených slov. Tyto jazyky je po registraci možné změnit v nastavení profilu. Formulář pro registraci je ukázán na obrázku 14.



Obrázek 14: LinguaReader: Registrační formulář

Zdroj: LinguaReader (2024)

Při kliknutí na tlačítko *Submit* odesílá klient na server dotaz, který obsahuje objekt s údaji, vyplněnými v registračním formuláři a je reprezentován třídou *UserForRegistrationDto*. Server dotaz přijímá a zpracovává údaje z obdrženého objektu. Nejdříve zkontroluje, jestli je objekt validní a obsahuje všechny údaje. Pokud ne, vrátí HTTP odpověď s kódem 400, což znamená chybu.

V případě, že je objekt validní, server vytvoří nový objekt třídy *ApplicationUser* z knihovny ASP.NET Core Identity, kterému přiřadí náhodné uživatelské jméno a data z registračního formuláře, která obdržel od klienta. Dále se server pokusí pomocí služby správce uživatelů, připojené pomocí technologie Dependency Injection, popsané v kapitole 5.3, vytvořit nového uživatele s heslem. Pokud se mu to nepodaří, například protože heslo nesplňuje stanovené podmínky, vrátí sadu chyb, které budou vypsány uživateli nad registračním formulářem. V případě, že uživatel bude vytvořen v databázi,

výsledek této operace bude úspěšný a server odešle na email uvedený uživatelem při registraci odkaz pro potvrzení této emailové adresy.

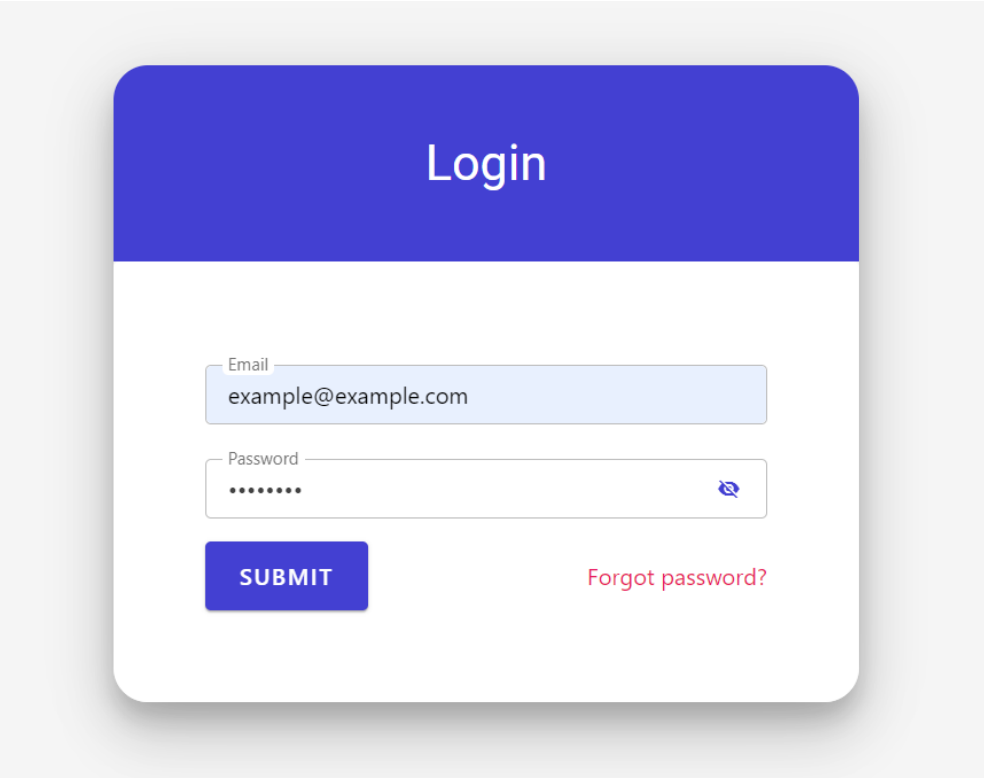
Při úspěšném provedení registrace nového uživatele vrátí server klientovi HTTP stavový kód 201, který podle společnosti Mozilla (2023b) sděluje, že byl požadavek úspěšný a došlo k vytvoření nového záznamu. Metoda na straně serveru, která zpracovává dotaz pro registraci nového uživatele, je ukázána na výpisu kódu 24.

```
[HttpPost]
[Route("api/accounts/Registration")]
public async Task<IActionResult> RegisterUser([FromBody] UserForRegistrationDto
userForRegistration)
{
    if (userForRegistration == null || !ModelState.IsValid)
        return BadRequest();
    var user = new ApplicationUser {
        UserName = "user_" + Guid.NewGuid().ToString(),
        Email = userForRegistration.Email,
        NativeLanguage = userForRegistration.NativeLanguage,
        DesiredLanguage = userForRegistration.DesiredLanguage };
    var result = await _userManager.CreateAsync(user, userForRegistra-
tion.Password);
    if (!result.Succeeded)
    {
        var errors = result.Errors.Select(e => e.Description);
        return BadRequest(new RegistrationResponseDto { Errors = errors });
    }
    await SendAddressConfirmationEmail(user);
    return StatusCode(201);
}
```

Výpis kódu 24: Kód registrace na straně serveru

Zdroj: vlastní zpracování (2024)

Uživatel se může přihlásit do aplikace a používat ji po potvrzení emailové adresy. Formulář pro přihlášení obsahuje dvě pole, pro email a heslo, tlačítko *Submit* a odkaz pro obnovení hesla v případě jeho zapomenutí. Jak vypadá formulář pro přihlášení, je ukázáno na obrázku 15.



Obrázek 15: LinguaReader: Formulář pro přihlášení

Zdroj: LinguaReader (2024)

Po vyplnění emailové adresy a hesla uživatelem ve formuláři pro přihlášení a stisknutí tlačítka *Submit* odesílá klient na server dotaz, který obsahuje objekt třídy *UserForAuthenticationDto*. Tento objekt obsahuje údaje, které byly vyplněné ve formuláři pro přihlášení. Server nejprve zkontroluje, jestli takový uživatel je v databázi, a pokud uživatel existuje, provede řadu dalších kontrol. Například server zkontroluje, jestli není uživatelský účet uzamčen a jestli uživatel uvedl správné heslo ve formuláři pro přihlášení. Pokud není heslo správné, zvýší počet neúspěšných pokusů přihlášení tohoto uživatele a vrátí zprávu s chybou, že je ověření neplatné. Část kódu, ve které server provádí kontrolu hesla, je zobrazena na výpisu kódu 25. Poslední kontrolou je ověření, že uživatel po registraci potvrdil emailovou adresu.

```
if (user is null || !await _userManager.CheckPasswordAsync(user, userForAuthenti-
cation.Password))
{
    if (user is not null)
        await _userManager.AccessFailedAsync(user);
    return Unauthorized(new AuthResponseDto { ErrorMessage="Invalid Authentica-
tion"});
}
```

Výpis kódu 25: Kontrola hesla pro přihlášení

Zdroj: vlastní zpracování (2024)

Pokud byly všechny kontroly úspěšné, server resetuje počet neúspěšných pokusů o přístup uživatele a vygeneruje pro něho token s použitím algoritmu HMAC-SHA256[15]. Pro autentizaci uživatelů aplikace používá JSON Web Token (JWT). Jak uvádí Spasojević (2024) na webu Code Maze, JWT tokeny umožňují bezpečný způsob přenosu dat ve formátu JSON a tento přístup je jedním z populárních mechanismů pro webové ověřování. Vymazání počtu neúspěšných pokusů o přístup, vytvoření tokenu a vrácení objektu třídy *AuthResponseDto* klientu, který obsahuje token, je uvedeno na výpisu kódu 26.

```
await _userManager.ResetAccessFailedCountAsync(user);
var signingCredentials = GetSigningCredentials();
var claims = GetClaims(user);
var tokenOptions = GenerateTokenOptions(signingCredentials, claims);
var token = new JwtSecurityTokenHandler().WriteToken(tokenOptions);
return Ok(new AuthResponseDto { IsAuthSuccessful = true, Token = token });
```

Výpis kódu 26: Úspěšné dokončení autentizace

Zdroj: vlastní zpracování (2024)

6.8.2 Obnovení hesla

V případě zapomenutí hesla ho může uživatel obnovit pomocí odkazu ve formuláři pro přihlášení. Po kliknutí na tento odkaz se mu zobrazí formulář, ve kterém uvede pouze svoji emailovou adresu, na kterou aplikace pošle odkaz obsahující token pro obnovení hesla.

Pokud uživatel zadá emailovou adresu, která není v aplikaci registrovaná, bude také vypsáno oznámení, že byl email odeslán. Tento postup je implementován z bezpečnostních důvodů, aby nebylo možné zjistit, které emailové adresy jsou v aplikaci registrované. V případě, že je emailová adresa v databázi nalezena, vygeneruje aplikace na straně serveru pro uživatele token použitím služby správce uživatelů knihovny ASP.NET Core Identity a její metody *GeneratePasswordResetTokenAsync*. Generování tokenu pro uživatele je ukázáno na výpisu kódu 27.

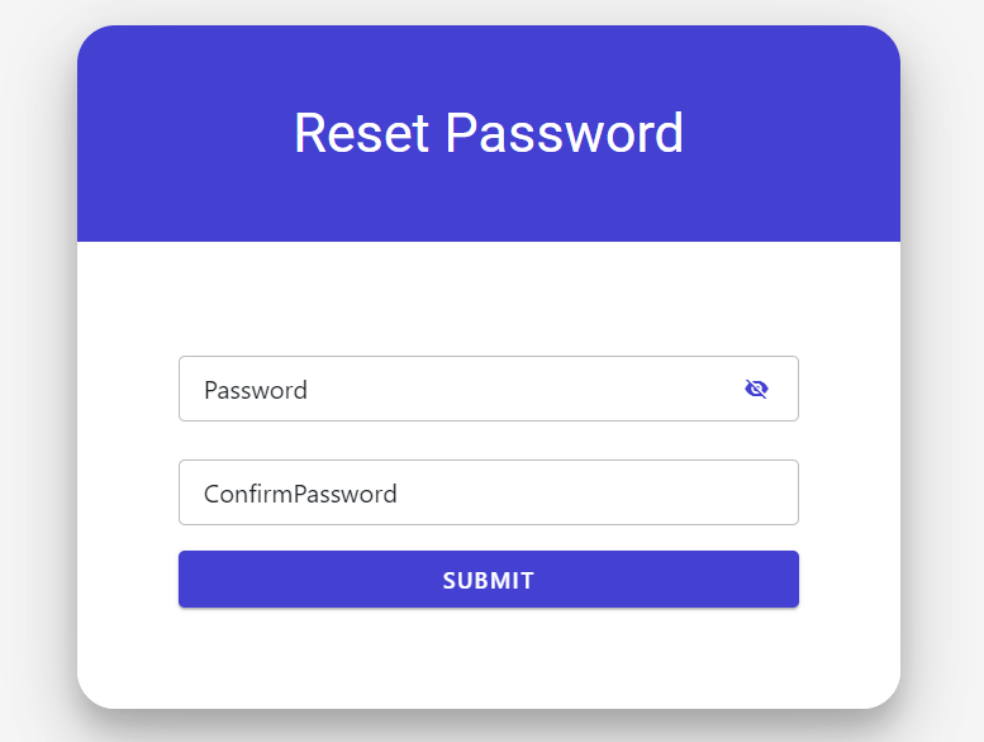
```
var token = await _userManager.GeneratePasswordResetTokenAsync(user);
```

Výpis kódu 27: Volání metody pro generování tokenu

Zdroj: vlastní zpracování (2024)

[15] Další informace o šifrovacím algoritmu Hmac Sha-256: https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography.hmacsha256?view=net-7.0#remarks

Aplikace vytvoří odkaz na stránku pro obnovení hesla, do kterého vloží vygenerovaný token jako parametr, a odešle tento odkaz na emailovou adresu uživatele. Po kliknutí na odkaz v emailu bude uživatel přesměrován na stránku aplikace s formulářem, ve kterém si nastaví nové heslo. Formulář pro obnovení hesla je vidět na obrázku 16.



Obrázek 16: LinguaReader: Formulář pro obnovení hesla

Zdroj: LinguaReader (2024)

Po vyplnění formuláře pro obnovení hesla a kliknutí na tlačítko *Submit* odešle klient na server objekt třídy pro přenos dat *ResetPasswordDto*. Tento objekt obsahuje nově zadané heslo a token, který byl předtím vygenerován a zaslán v odkazu na emailovou adresu uživatele. Pro obnovení hesla server využívá správce uživatelů, kterému předává objekt uživatele, token a nové heslo jako parametry. Proces obnovení hesla serverem je ukázán na výpisu kódu 28.

```
var resetPassResult = await _userManager.ResetPasswordAsync(user,
resetPasswordDto.Token, resetPasswordDto.Password);
```

Výpis kódu 28: Kód pro obnovení hesla

Zdroj: vlastní zpracování (2024)

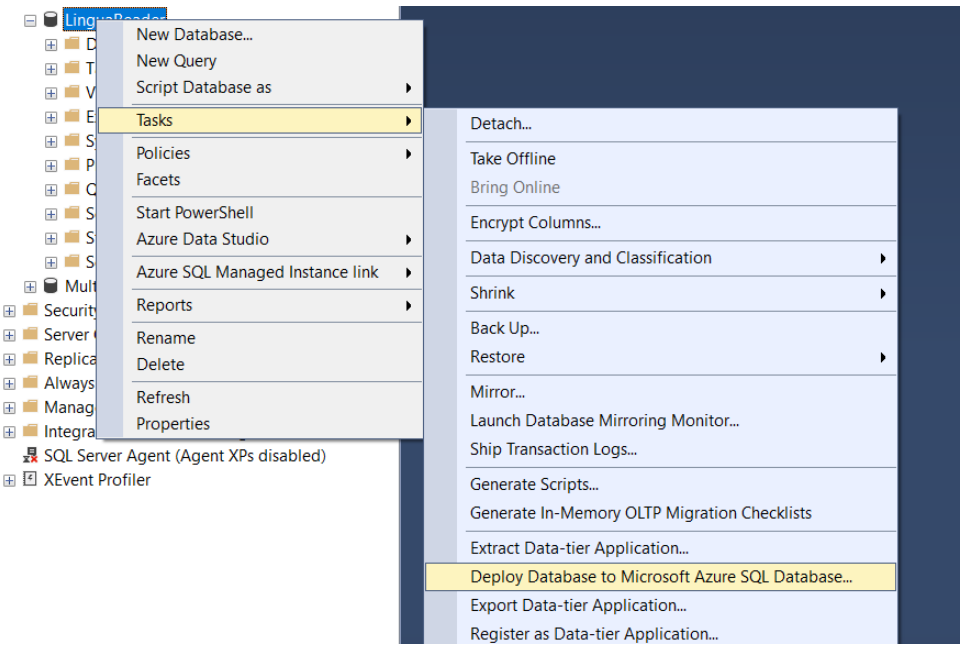
Pokud je heslo úspěšně obnoveno, odesílá server klientu odpověď s HTTP kódem 200 a uživateli se na straně klienta zobrazí odpovídající oznámení o úspěchu. Po dokončení tohoto procesu se může uživatel přihlásit do aplikace s novým heslem.

7 Nasazení databází a aplikace

Nasazení je realizováno v cloudové platformě Microsoft Azure, ve které jsem vytvořil instanci služby Azure AI Translator pro překlad slov. Měl jsem na svém studijním účtu poskytnutý kredit ve výši 100 dolarů, což mi umožnilo nasazení aplikace a databází a zajištění jejich fungování v cloudu po dostatečnou dobu, pro testování, ukázku uživatelům a sběr zpětné vazby.

7.1 Nasazení databází

V procesu nasazení jsem začal s databázemi. V rámci prostředí Microsoft Azure jsem zřídil a nakonfiguroval SQL server, na který byly následně pomocí nástroje SQL Server Management Studio vyexportovány lokální databáze s daty. Spuštění procesu nasazení v SQL Server Management Studio je ukázáno na obrázku 17.



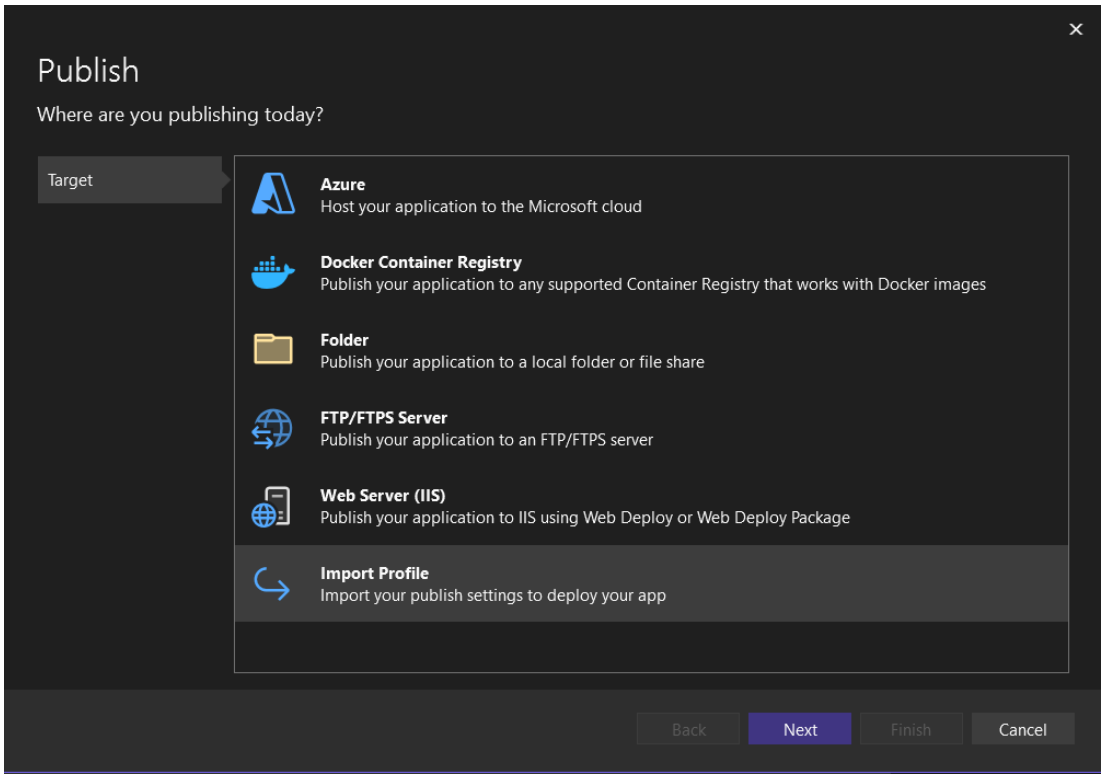
Obrázek 17: Nasazení databáze pomocí SQL Server Management Studio

Zdroj: Microsoft (2024b)

V prostředí Microsoft Azure jsem pro databáze zvolil nákupní model Database Transaction Units (DTU) s cenovou úrovní *Basic*. Tato konfigurace omezuje kapacitu uložených dat na 2 GB a stanovuje náklady na provoz jedné databáze na 5,6 eur měsíčně (Microsoft, ©2024b).

7.2 Nasazení aplikace

V rámci projektu bylo nutné nasadit 2 části aplikace – klientskou a serverovou. Nejprve jsem se pokusil o nasazení pomocí služby Azure Static Web App, což mi umožnilo nasadit klientskou část aplikace. Avšak pro nasazení serverové části byla tato služba nevhodná. Azure Static Web App nabízí možnost využití Azure Functions, což je podle Microsoftu (2023d) služba umožňující spuštění samostatných kusů kódu nebo funkcí v reakci na různé události, včetně HTTP požadavků. Mým cílem však bylo nasadit serverovou část jako samostatnou aplikaci, na což jsem našel vhodnou službu Azure App Service[16]. Vytvořil jsem *App Service Plan* v Azure a provedl základní nastavení pro obě části aplikace. Dále jsem stáhl jejich profily pro publikování a importoval je do vývojového prostředí v programu Visual Studio, jak je vidět na obrázku 18.



Obrázek 18: Importování profilu ve Visual Studio 2022

Zdroj: Visual Studio 2022 (©2024c)

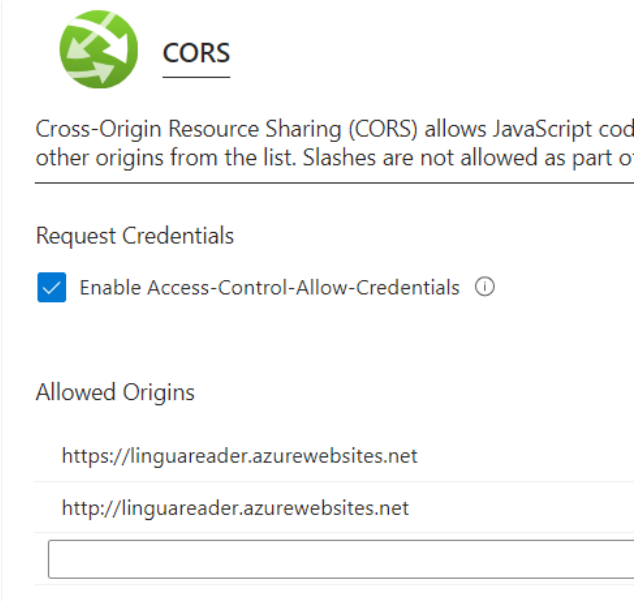
[16] Další informace o službě pro hostování webových aplikací App Service: https://learn.microsoft.com/cs-cz/azure/app-service/overview

Po importu profilu je potřeba stisknout tlačítko *Publish*, aby proběhlo nasazení kódu do aplikačních profilů v prostředí Azure. Pro aktualizaci kódu ve Visual Studiu u vybrané části aplikace je nutné znovu stisknout tlačítko *Publish*, které je ukázáno na obrázku 19.

Obrázek 19: Tlačítko Publish ve Visual Studio 2022

Zdroj: Visual Studio 2022 (©2024c)

Aby mohla klientská část aplikace komunikovat se serverovou částí, je nutné nastavit v prostředí Azure pro serverovou část mechanismus Cross-Origin Resource Sharing (CORS). Toto nastavení umožňuje JavaScript kódu spouštěnému na straně klienta, který má jiný původ, komunikovat se serverem. Nastavení CORS mechanismů je vidět na obrázku 20.



Obrázek 20: Nastavený CORS mechanismus v Azure

Zdroj: Microsoft Azure (b. r.)

7.2.1 Konfigurace a nastavení proměnných prostředí

Pro úspěšné nasazení jsem také musel změnit strategii uložení konfiguračních nastavení aplikace, která byla původně uložena v souboru appsettings.json. Tuto změnu jsem provedl z důvodu rozdělení prostředí ve Visual Studiu na vývojové a produkční, kde se tato nastavení liší. Druhým důvodem bylo najít způsob, jak poskytnout produkčnímu prostředí skrytá data, jako jsou řetězce pro připojení k databázím, hesla atd.

Poskytovatelé konfigurací ukládají nastavení pro aplikaci ve formě klíč-hodnota a poskytují tato data pro *ConfigurationBuilder* při spouštění aplikace. Základními poskytovateli konfigurací jsou:

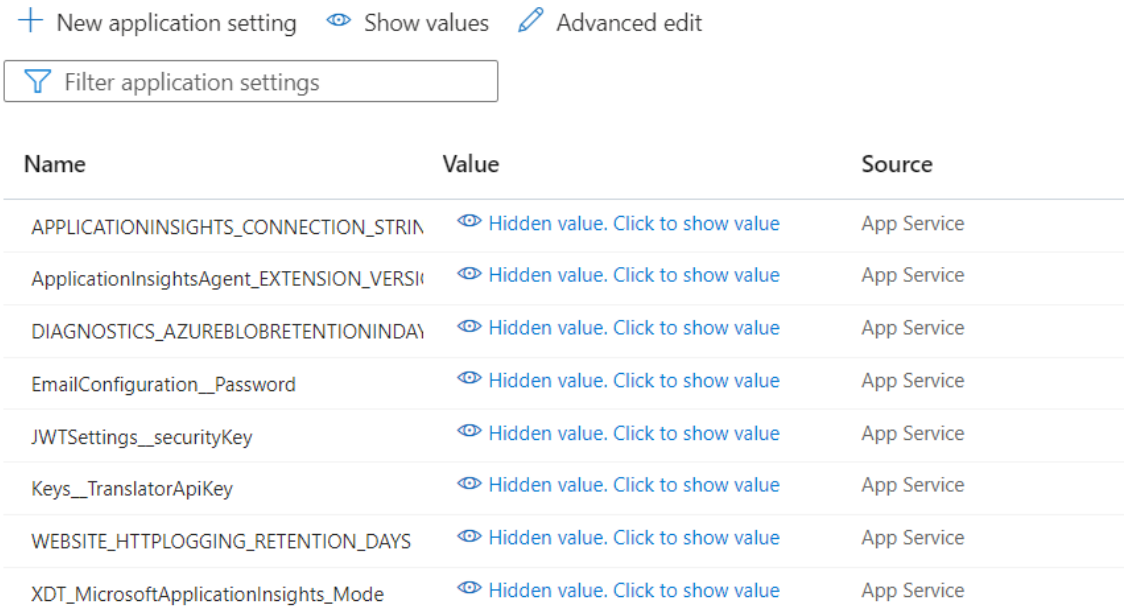
- JSON file provider – načte nastavení z volitelného souboru JSON s názvem appsettings.json. Také načítá nastavení z volitelného souboru JSON specifického pro dané prostředí s názvem appsettings.ENVIRONMENT.json.
- User Secrets – načte skrytá data, která jsou bezpečně uložena během vývoje.
- Environment variables – načte proměnné prostředí jako konfigurační proměnné, které jsou ideální pro ukládání skrytých dat v produkčním prostředí.

(Lock, 2021, s. 338)

Rozdělil jsem JSON soubory podle prostředí na *Development* a *Production*, a do nich jsem uložil veřejná data. Soubor appsettings.json obsahuje data, která jsou stejná pro všechna prostředí.

Soubor appsettings.Development.json obsahuje veřejná data pro vývojové prostředí. Pro skrytá data používám ve Visual Studiu *User Secrets Manager*. Tento nástroj umožňuje efektivní přidání proměnných prostředí, které jsou uloženy v jiném umístění než samotná aplikace (Lock, 2021, s. 344). Tímto způsobem jsou uchovávána skrytá data, jako jsou připojovací řetězce k databázím, klíč pro použití Azure AI Translatoru, heslo pro email a další citlivé informace, které by se neměly vyskytnout v systému pro správu verzí GIT.

Soubor appsettings.Production.json obsahuje veřejná data pro produkční prostředí. Liší se například řádky s odkazy na klienta – zatímco ve vývojové fázi se používá localhost, v produkční fázi je to skutečná URL adresa. Skrytá data pro toto prostředí jsou uložena jako proměnné přímo v Azure v rámci konfigurace nasazené aplikace. Tyto proměnné prostředí uložené v Azure jsou ukázány na obrázku 21.



Obrázek 21: Proměnné prostředí uložené v Azure

Zdroj: Microsoft Azure (b. r.)

8 Zhodnocení použitelnosti aplikace

Po nasazení aplikace jsem poslal odkaz na ni několika lidem, aby ji vyzkoušeli a abych získal zpětnou vazbu. Ptal jsem se, zda narazili na nějaké chyby nebo nedokonalosti během používání aplikace a jestli se podařilo nahrát všechny požadované soubory. Další dotazy se týkaly funkčnosti, například, zda překlad slov zvýšil jejich porozumění textu v cizím jazyce, jestli byla funkce procvičování uložených slov užitečná, nebo jestli si slova vyexportovali do jiné aplikace. Zajímalo mě také, zda je uživatelské rozhraní aplikace přívětivé a jestli bylo jednoduché se v něm vyznat.

Uživatel 1 je specialistka na lidské zdroje a hodnotí aplikaci pozitivně. Podařilo se jí nahrát soubor a přidat slovíčka do seznamu pro další zapamatování a procvičování. Aplikace jí umožnila číst knihu v cizím jazyce bez využití slovníku. Ocenila funkci vyslovování slov a líbila se jí herní mechanika zapamatování slovíček. Ocenila by, kdyby si aplikace pamatovala, na které stránce se uživatel zastavil, pokud přejde do sekce učení a pak se znovu vrátí ke čtení. Uživatelské rozhraní je podle ní jednoduché a srozumitelné.

Uživatel 2 má zkušenosti s programováním webových aplikací. Nahrávání souborů ve formátu EPUB pro něj bylo bezproblémové. Menší problémy byly se zobrazením souborů ve formátu PDF – text a větší obrázky občasně přetékaly vymezené pole. Při používání aplikace ocenil, že slova, které neznal, si mohl uložit a později procvičovat. Procvičování slov v aplikaci podle jeho názoru funguje bezchybně. Aplikace LinguaReader je podle něj velmi užitečná na čtení knih v cizím jazyce a velmi dobře ovladatelná a přehledná.

Uživatel 3 má zkušenosti jako specialista technické podpory. Aplikace podle něho funguje výborně a na žádné chyby nenarazil. Překlad slov ocenil jako velmi dobrý a aplikace mu pomohla lépe porozumět textu v cizím jazyce. Doporučil změnit způsob upozornění o nastavení jazyka knihy před jejím otevřením, aby to nevypadalo jako chyba. Existující návod k použití doporučil zobrazit uživateli hned po prvním přihlášení.

Na základě zpětné vazby uživatelů a vlastních zkušeností s aplikací bych ji ohodnotil jako dobře použitelnou a vyžadující několik drobných vylepšení. Převážnou většinu knih se podařilo nahrát a aplikace správně zobrazila jejich text. Překlad slov kliknutím v textu funguje správně a přispívá porozumění textům v cizích jazycích, avšak přesnost poskytnutých překladů by mohla být vyšší. Ukládání slov a jejich následné procvičování fungují bezchybně. Uživatelské rozhraní je velmi dobře ovladatelné, ale obsahuje nepřesnosti a mohlo by být zlepšeno.

9 Vývoj do budoucna

Následující kroky jsou důležité pro rozvoj a zdokonalení aplikace. V předchozí kapitole byla popsána zjištěná zpětná vazba a požadavky od uživatelů. Zároveň jsem analyzoval kód aplikace a použité přístupy během vývoje. Na základě těchto informací byly identifikovány oblasti pro optimalizaci a stanoveny možná zlepšení aplikace.

- Zlepšení kvality překladů slov nebo přeložení slova s ohledem na kontext, ve kterém se vyskytuje. K tomu je možné rozšířit již používaný přístup o další služby nebo využít jiné služby pro překlad slov. Dalším vylepšením by mohlo být zobrazení a uložení překladů v základních tvarech – sloves v infinitivu a podstatných jmen v nominativu.
- Přidání ukazovacích zájmen přeloženého slova. Služba Azure AI Translator má v JSON odpovědi klíč s názvem *prefixWord*, který by podle dokumentace[17] měl obsahovat ukazovací zájmena k přeloženému slovu. Nicméně při obdržení odpovědi od překladače je hodnota tohoto klíče vždy prázdným řetězcem.
- Zlepšení a úpravy uživatelského rozhraní aplikace a jeho lokalizace do dalších jazyků. Zobrazení návodu pro nově registrované uživatele a zajištění možnosti zvětšení textu při čtení knihy, aby byl text responzivní a přizpůsobil se rozměru stránky. Přidání statistiky o všech pokusech procvičování karet uživatelem, nikoliv pouze po dokončení kola. Oprava chyb, které se vyskytují při zobrazení knih, například přetékání textů a obrázků přes vymezené pole.
- Uložení HTML souborů na server místo ukládání textu knih ve formátu HTML do databáze. Pro EPUB knihy by mohla být uložena celá složka se zachovanou strukturou. Knihy, které obsahují celý text v jednom souboru, mohou být uloženy jako HTML soubor společně s vytaženými obrázky a dalšími soubory.

Stanovení těchto bodů pro zlepšení mé aplikace je pro mě důležité, protože plánuji i nadále pokračovat ve vývoji a optimalizaci aplikace LinguaReader. Některá z těchto vylepšení nejsou náročná a lze je implementovat poměrně rychle. Většina důležitých změn, jako je zlepšení kvality překladů a optimalizace strategie vytažení textu z PDF souborů, vyžadují více času a zvážení, jak je správně provést.

[17] Další informace o službě překladače AI Translator 3.0: Dictionary Lookup: https://learn.microsoft.com/en-us/azure/ai-services/translator/reference/v3-0-dictionary-lookup

10 Závěr

Aplikace LinguaReader je zdokonalením mého původního Python skriptu pro překlad slov při čtení digitálních textů. Při návrhu aplikace jsem vycházel z vlastních zkušeností a byl motivován jak omezeným počtem a nedostatky existujících řešení na trhu, tak i přáním vytvořit užitečný nástroj, který pomůže lidem ve studiu cizích jazyků. Cílem této práce bylo vytvořit webovou aplikaci pro podporu výuky cizích jazyků a rozšiřování slovní zásoby. Tento cíl považuji za splněný.

Aplikace byla vyvinuta v jazyce C# s využitím technologie ASP.NET a je dostupná[18] na internetu pro veřejnost. Umožňuje nahrání vlastního textu ze souborů ve formátech EPUB, PDF a TXT. Dále nabízí funkci označení neznámých slov pro automatický překlad, který je realizován pomocí služby překladače v cloudovém systému Microsoft Azure. Podle požadavků aplikace umožňuje ukládání a trénování slov označených v textu knihy. Během trénování se uživateli zobrazuje slovo v cizím jazyce a několik variant překladů, ze kterých musí vybrat tu správnou. Funkční požadavky stanovené v kapitole 3 jsou implementovány v plném rozsahu. Stejně tak byly dodrženy nefunkční požadavky. Například v kapitole 6.4.2 bylo provedeno porovnání programovacích jazyků C# a JavaScript pro zobrazení textu knihy.

Aplikace je nasazená a podle zpětné vazby uživatelů je použitelná a pomáhá jim v porozumění textům v cizích jazycích a při učení se slovní zásoby. Podle zpětné vazby uživatelů disponuje aplikace také uživatelsky přívětivým rozhraním a je v něm snadné se zorientovat, což splňuje nefunkční požadavek č. 3. Od uživatelů jsem obdržel zpětnou vazbu, podle které plánuji zlepšit kvalitu překladů slov, ještě lépe upravit uživatelské rozhraní a přidat další funkce.

Vývoj projektu v rámci mé bakalářské práce mi umožnil získat praktické zkušenosti s novými technologiemi a knihovnami, zejména v oblasti tvorby webových aplikací klient-serverové architektury s použitím programovacího jazyku C# a frameworku ASP.NET Core, a při práci s databází pomocí Entity Frameworku. Získal jsem také mnoho dalších zkušeností, což mi zajisté pomůže v mých budoucích projektech a profesionálním rozvoji.

[18] Aplikace LinguaReader je dostupná z https://linguareader.azurewebsites.net

11 Summary

The LinguaReader application is an improvement of my original Python script for translating words while reading digital texts. When designing the application, I was inspired by my own experiences and driven both by the limited number and deficiencies of existing solutions on the market, as well as by the desire to create a valuable tool that will assist people in studying foreign languages. The aim of this thesis was to create a web application to support the learning of foreign languages and their vocabulary. I consider this goal achieved.

The application was developed in C# using ASP.NET framework and is available online to the public. It allows users to upload their own text from EPUB, PDF, and TXT files. The application has a feature to mark unknown words for automatic translation, which is done using the Microsoft Azure cloud translation service. According to the requirements, the application allows to storage and practice of words marked in the text of a book. During practice, users are shown a word in a foreign language with several translation options, from which they need to choose the correct one. The functional requirements set in Chapter 3 have been fully implemented, as well as the non-functional requirements. For example, in chapter 6.4.2, a comparison was made between the programming languages C# and JavaScript for displaying the text of the book.

The application is deployed and according to user feedback, it is usable and helps them understand texts in foreign languages and learn vocabulary. Users also say the application has a user-friendly interface and is easy to navigate, which meets the non-functional requirement 3. I have received feedback from users based on which I plan to improve the quality of word translations, further refine the user interface, and add additional features.

Developing this project for my bachelor's thesis allowed me to gain practical experience with new technologies and libraries especially in the field of creating web applications with client-server architecture using the C# programming language and the ASP.NET Core framework, and in working with databases using Entity Framework. I also gained many other experiences, which will help me in my future projects and professional development.

12 Seznam použitých zdrojů

Apryse. (1. červen 2020). *How PDF.js Works*. Získáno 19. leden 2024, z PDF.js Express: https://pdfjs.express/blog/how-pdf-js-works

Apryse. (b. r.). *itext-dotnet*. Získáno 28. březen 2024, z GitHub: https://github.com/itext/itext-dotnet

Atlassian. (©2024). *Continuous delivery starts on the Jira kanban board*. Získáno 30. březen 2024, z Jira Software: https://www.atlassian.com/software/jira/features/kanban-boards

Atlassian. (b. r.). *Jira Software*. Získáno 10. duben 2024, z Atlas: https://id.atlassian.com/login

Brind, M. (15. únor 2023). *JavaScript Interop in Blazor - Executing JavaScript From C#*. Získáno 30. březen 2024, z Learn Blazor: https://www.learnblazor.com/javascript-interop

Code Maze. (2022a). *Blazor WebAssembly Series*. Získáno 22. březen 2024, z Code Maze: https://code-maze.com/blazor-webassembly-series/

Code Maze. (2022b). *ASP.NET Core Identity Series*. Získáno 22. březen 2024, z Code Maze: https://code-maze.com/asp-net-core-identity-series/

Google. (b. r.). *Google Chrome*. Získáno 10. duben 2024, z Chrome: https://www.google.com/chrome/

Hasvy. (4. prosinec 2023). *translator-script*. Získáno 10. duben 2024, z GitHub: https://github.com/Hasvy/translator-script

Laba. (3. leden 2022). *Kanban: základní principy a výhody*. Získáno 14. březen 2024, z Laba: https://l-a-b-a.cz/blog/12-kanban-zakladni-principy-a-vyhody

LingaIO. ([2022]). *Linga*. Získáno 12. březen 2024, z Linga: https://linga.io/

Lithium Lab Pte Ltd. (2024). *EWA*. Získáno 10. březen 2024, z EWA: https://appewa.com

Lock, A. (2021). *ASP.NET Core in Action* (Second Edition). United States of America: Manning.

Martinů, J., & Čermák, P. (2018). *Metodiky Vývoje Software.* Studijní opora, Moravská vysoká škola Olomouc, Olomouc. Získáno 14. březen 2024, z https://dl1.cuni.cz/pluginfile.php/864918/mod_resource/content/1/Metodiky-v%C3%BDvoje-software-studijn%C3%AD-text.pdf

Meloni, J. (2011). *Sams Teach Yourself HTML, CSS, and JavaScript All in One.* Indianapolis, USA, Indiana: Pearson Education. Získáno 19. únor 2024, z https://www.doc-developpement-durable.org/file/Projets-informatiques/cours-&-manuels-informatiques/htm-html-xml-ccs/Sams%20Teach%20Yourself%20HTML,%20CSS,%20and%20JavaScript%20All%20in%20One.pdf

Microsoft. (©2024a). *Build beautiful web apps with Blazor*. Získáno 30. březen 2024, z .NET: https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor

Microsoft. (©2024b). *Azure SQL Database pricing*. Získáno 30. březen 2024, z Microsoft Azure: https://azure.microsoft.com/en-us/pricing/details/azure-sql-database/single/#pricing

Microsoft. (©2024c). *Visual Studio 2022*. Získáno 10. duben 2024, z Microsoft: https://visualstudio.microsoft.com/cs/#vs-section

Microsoft. (4. květen 2023a). *A tour of the C# language*. Získáno 10. březen 2024, z Microsoft Learn: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

Microsoft. (30. listopad 2023b). *modely hostování ASP.NET Core Blazor*. Získáno 18. leden 2024, z Microsoft Learn: https://learn.microsoft.com/cs-cz/aspnet/core/blazor/hosting-models?view=aspnetcore-7.0

Microsoft. (12. leden 2023c). *Inheritance*. Získáno 19. leden 2024, z Microsoft Learn: https://learn.microsoft.com/en-us/ef/core/modeling/inheritance

Microsoft. (24. květen 2023d). *Azure Functions overview*. Získáno 10. březen 2024, z Microsoft Learn: https://learn.microsoft.com/en-us/azure/azure-functions/functions-overview?pivots=programming-language-csharp

Microsoft. (1. březen 2024a). *ASP.NET Core Blazor JavaScript interoperability (JS interop)*. Získáno 3. duben 2024, z Microsoft Learn: https://learn.microsoft.com/en-us/aspnet/core/blazor/javascript-interoperability/?view=aspnetcore-7.0

Microsoft. (19. březen 2024b). *Download SQL Server Management Studio (SSMS)*. Získáno 30. březen 2024, z Microsoft Learn: https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16

Microsoft. (b. r.). *Microsoft Azure*. Získáno 10. duben 2024, z Microsoft Azure: https://portal.azure.com/

Mozilla Foundation. (29. srpen 2023a). *Data URLs*. Získáno 30. březen 2024, z MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Data_URLs

Mozilla Foundation. (10. duben 2023b). *201 Created*. Získáno 19. březen 2024, z MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/201

Mozilla Foundation. (5. březen 2024a). *HTML: HyperText Markup Language*. Získáno 30. březen 2024, z MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/HTML

Mozilla Foundation. (5. březen 2024b). *CSS: Cascading Style Sheets*. Získáno 30. březen 2024, z MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/CSS

Mozilla Foundation. (24. leden 2024c). *Using shadow DOM*. Získáno 30. březen 2024, z MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/API/Web_components/Using_shadow_DOM

Price, M. (2021). *C# 10 and .NET 6 – Modern Cross-Platform Development: Build apps, websites, and services with ASP.NET Core 6, Blazor, and EF Core 6 using Visual Studio 2022 and Visual Studio Code* (Sixth Edition). Birmingham, UK: Packt.

Radzen. (©2018-2024). *Radzen Blazor Components*. Získáno 29. březen 2024, z Radzen Blazor Components: https://blazor.radzen.com/

Rappl, F. (b. r.). *AngleSharp*. Získáno 28. březen 2024, z GitHub: https://github.com/AngleSharp/AngleSharp

Rash, W. (29. prosinec 2023). *Best language learning apps in 2024, tested by our editors*. Získáno 12. březen 2024, z CNN Underscored: https://edition.cnn.com/cnn-underscored/reviews/best-language-learning-apps

Readlang. ([2013]). *About Readlang*. Získáno 12. březen 2024, z Readlang Blog: https://blog.readlang.com/about

Six Labors. (b. r.). *ImageSharp*. Získáno 28. březen 2024, z GitHub: https://github.com/SixLabors/ImageSharp

Spasojević, M. (6. říjen 2022). *Introducing Identity to the ASP.NET Core Project*. Získáno 30. březen 2024, z Code Maze: https://code-maze.com/identity-asp-net-core-project/

Spasojević, M. (31. leden 2024). *JWT Authentication in ASP.NET Core Web API*. Získáno 30. březen 2024, z Code Maze: https://code-maze.com/authentication-aspnetcore-jwt-1/

Stedfast, J. (b. r.). *MailKit*. Získáno 28. březen 2024, z GitHub: https://github.com/jstedfast/MailKit

T Mahachi, J. (5. březen 2023). *Entity Framework Code First: with Migrations*. Získáno 16. březen 2024, z Medium: https://medium.com/@josiahmahachi/entity-framework-code-first-with-migrations-8d1a197d2bd4

Telf.cz. (©2022). *Cizojazyčné knihy nejsou strašák, ale pomocník. Víte, jak vybrat tu správnou?* Získáno 30. březen 2024, z Telf: https://telf.cz/cizojazycne-knihy-nejsou-strasak-ale-pomocnik-vite-jak-vybrat-tu-spravnou/

W3Techs. (©2009-2024). *Usage statistics of HTML for websites*. Získáno 26. březen 2024, z W3Techs - World Wide Web Technology Surveys: https://w3techs.com/technologies/details/ml-html_any

Zavalishin, R. (2024). *LinguaReader*. Získáno 10. duben 2024, z Azurewebsites: https://linguareader.azurewebsites.net

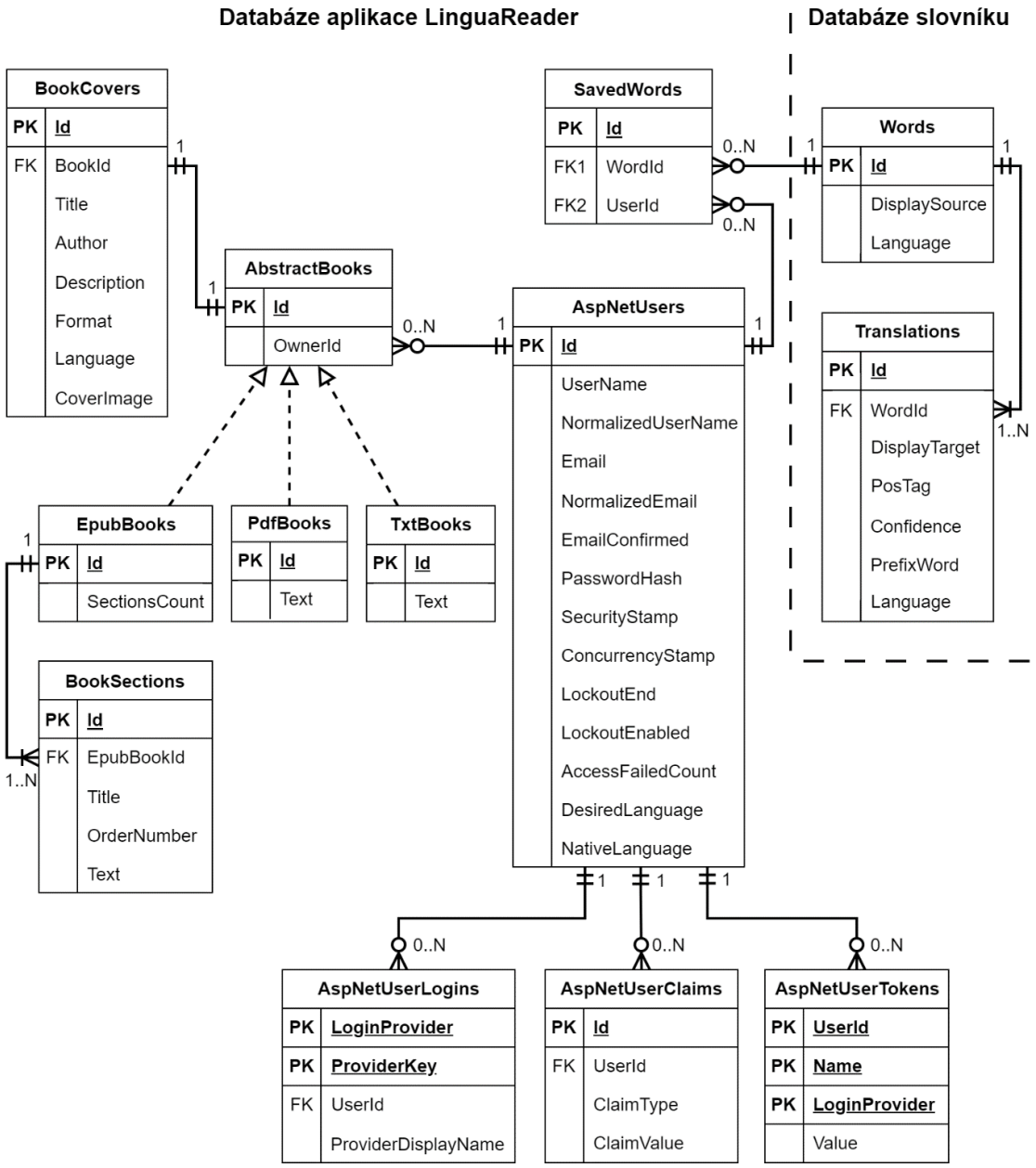
13 Seznam obrázků

14 Seznam výpisů kódu

15 Seznam použitých technologií

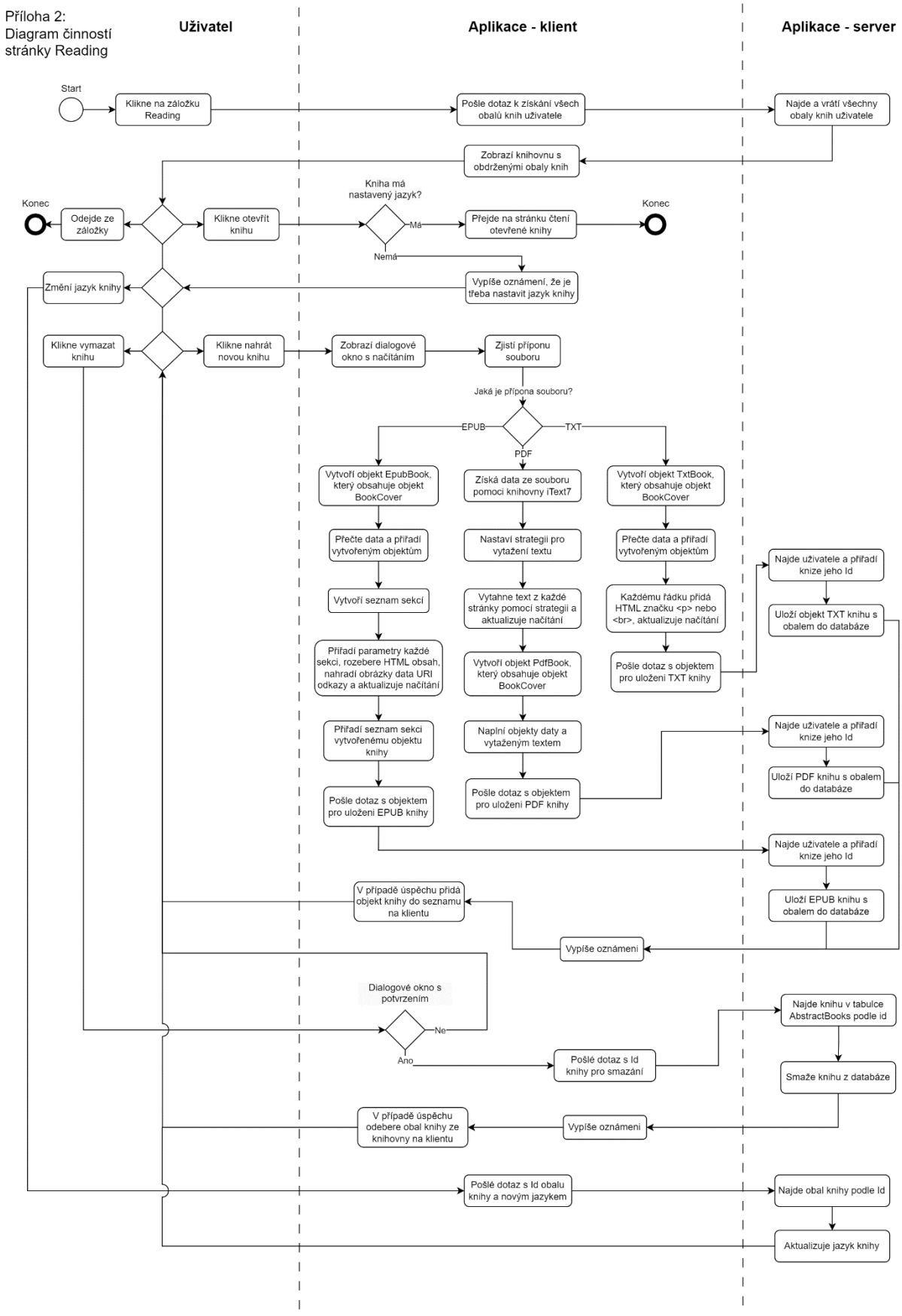
16 Seznam příloh

17 Přílohy



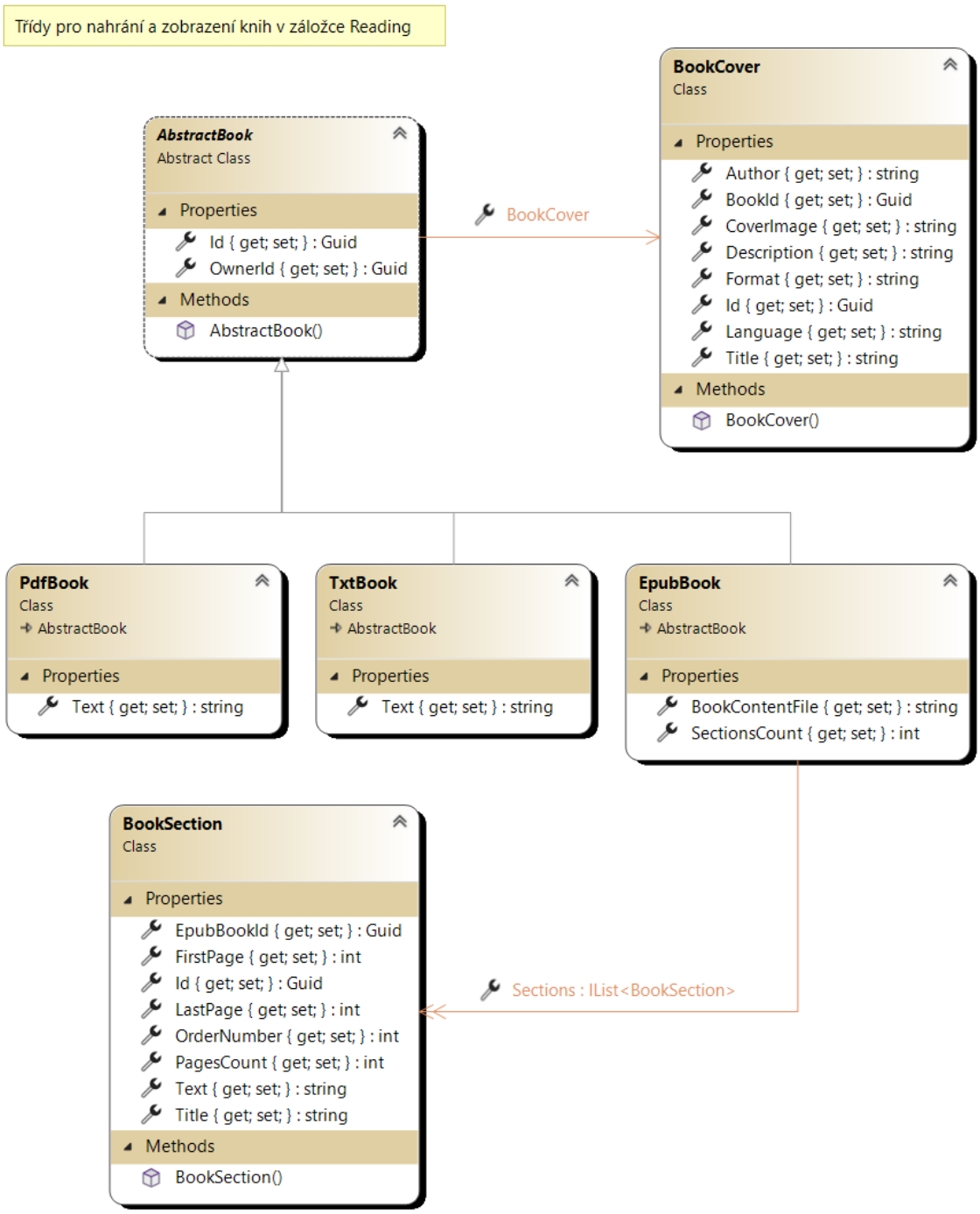
Příloha 1: Diagram entitních vztahů

Zdroj: vlastní zpracování (2024)



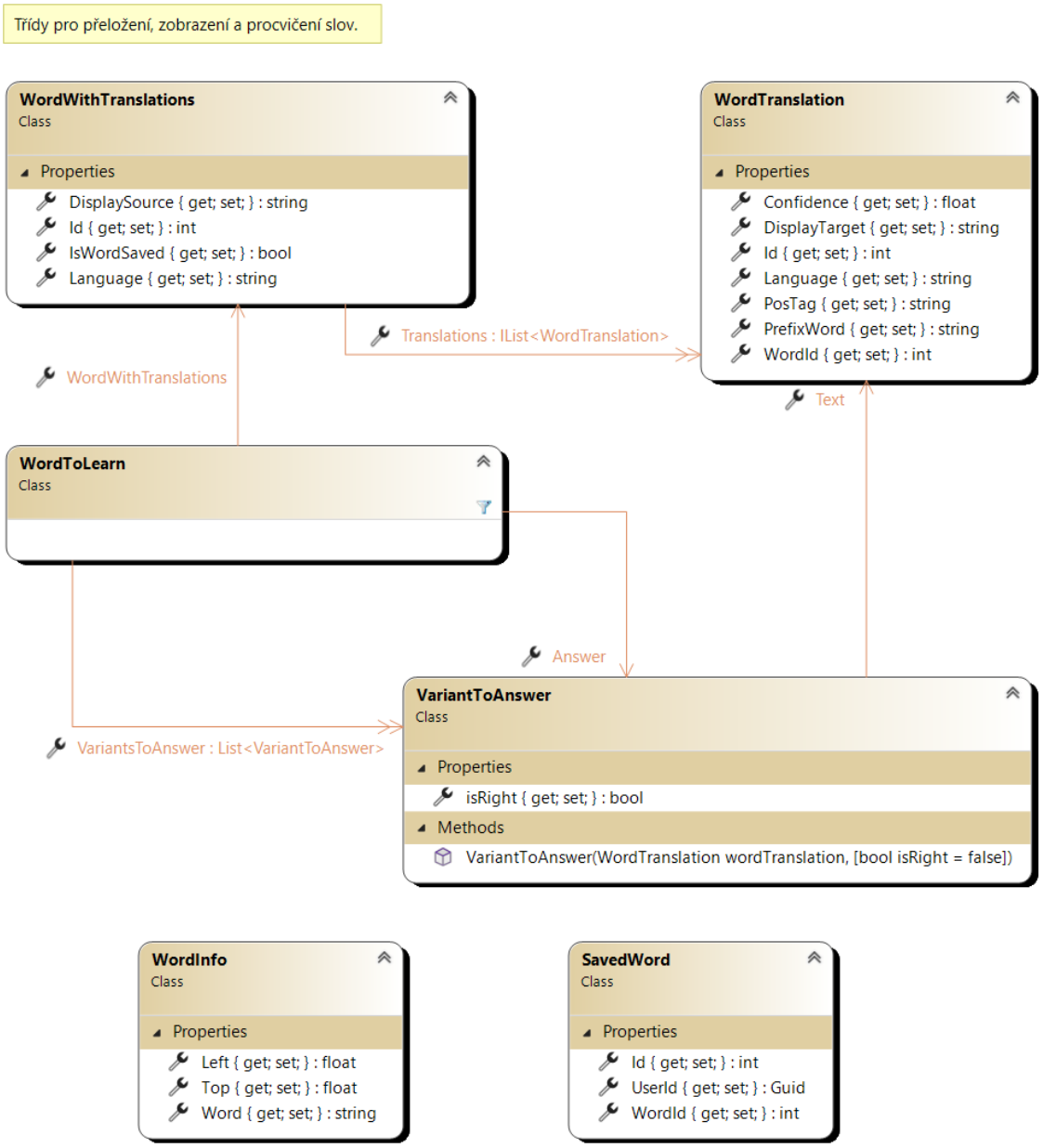
Příloha 2: Diagram činností stránky Reading

Zdroj: vlastní zpracování (2024)



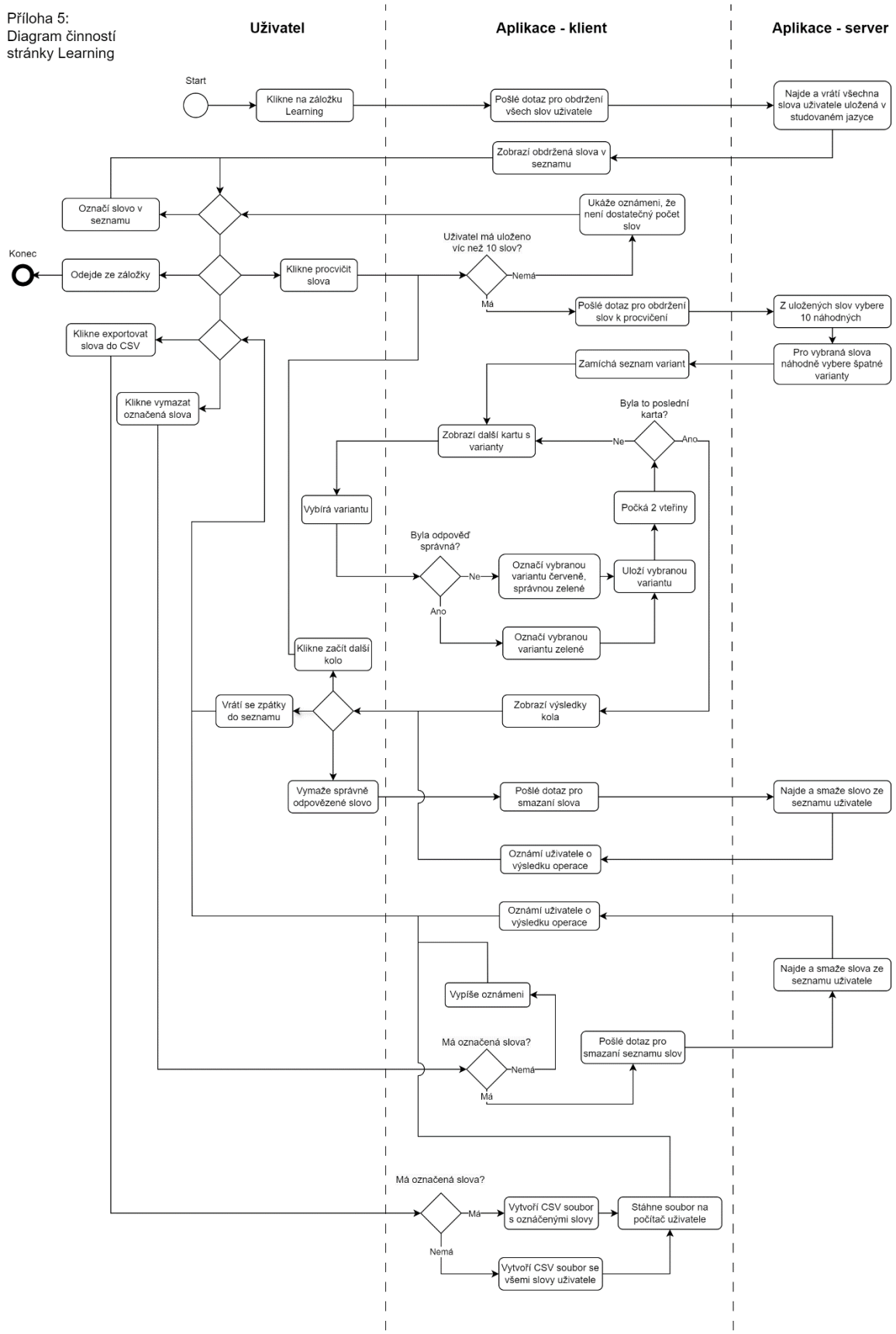
Příloha 3: Diagram tříd pro funkcionalitu čtení

Zdroj: vlastní zpracování (2024)



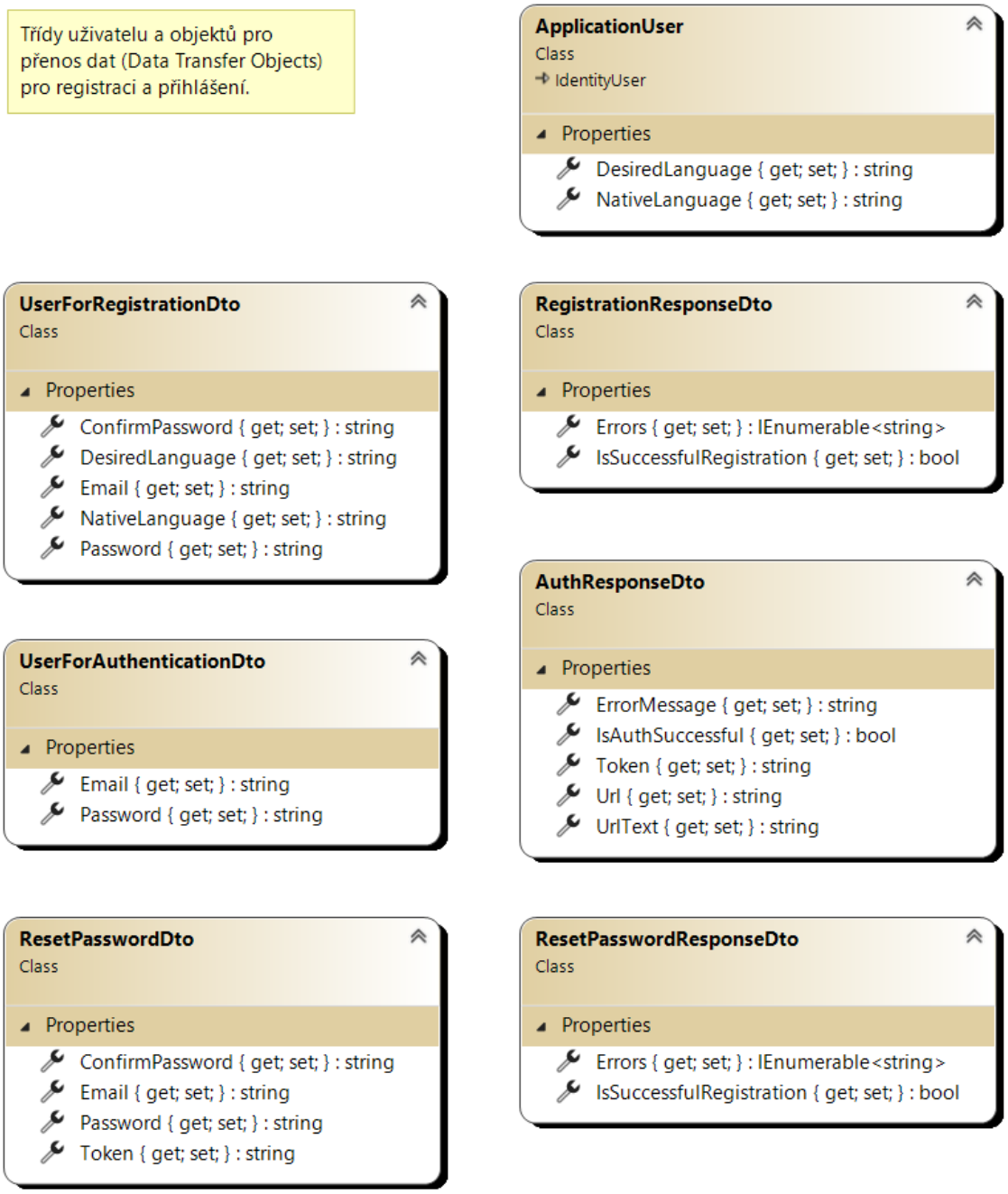
Příloha 4: Diagram tříd pro systém učení

Zdroj: vlastní zpracování (2024)



Příloha 5: Diagram činností stránky Learning

Zdroj: vlastní zpracování (2024)



Příloha 6: Diagram tříd pro registraci a přihlášení

Zdroj: vlastní zpracování (2024)