UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2024 Vladislav Volkov

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

SaaS aplikace pro zpracování dat z PDF souborů

Vladislav Volkov

Bakalářská práce
2024

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Vladislav Volkov**
Osobní číslo:	**I20292**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**SaaS aplikace pro zpracování dat z PDF souborů**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem práce je tvorba aplikace zaměřující se na vyhledání konkrétních dat zároveň ve více PDF souborech a vyhodnocení nalezené množiny dat v kontextu zadaných vstupních podmínek. Řešení má za cíl usnadnit extrakci, analýzu a agregaci informací obsažených ve více PDF souborech. Aplikace tak bude také sloužit pro hledání spojitosti a kontextu informací. Součástí závěrečné práce je i analýza současných řešení a jejich konkrétních využití. Předpokládané použité technologie jsou Next.js, React a dále využití API rozhraní AI.

Rozsah pracovní zprávy: **min. 30 s.,dop. rozsah 40 s.SA**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

PENDYALA, V. S. *Machine Learning for Societal Improvement, Modernization, and Progress*, 2022, Spojené státy americké: IGI Global, ISBN 978-1668440476.
SILGE, J. *Text Mining with R: A Tidy Approach*, 2017, O'Reilly Media, ISBN 978-1491981658

Vedoucí bakalářské práce: **Ing. Monika Borkovcová, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. prosince 2023**
Termín odevzdání bakalářské práce: **10. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2024

Prohlašuji:

Práci s názvem SaaS aplikace pro zpracování dat z PDF souborů jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 10. 05. 2024

Vladislav Vokov v.r.

PODĚKOVÁNÍ

Rád bych poděkoval své rodině za finanční a morální podporu po celou dobu mého studia. Další obrovské poděkování patří vedoucí bakalářské práce Ing. Monice Borkovcové Ph.D. bez Vás by nebylo možné tuto práci začít psát a úspěšně ji dokončit v stanoveném termínu.

ANOTACE

Tato bakalářská práce se zabývá návrhem a implementací aplikace SaaS pro zpracování dat ze souborů PDF. Cílem práce je tvorba aplikace, která uživatelům umožní efektivně vyhledávat, extrahovat, analyzovat a agregovat informace z více souborů PDF souborů současně. Aplikace bude využívat aplikační programovací rozhraní OpenAI pro zpracování textu a vyhledávání vztahů mezi daty. Součástí závěrečné práce je i analýza stávajících řešení pro zpracování souborů PDF a zaměřuje se na problémy spojené s extrakcí dat a jejich zpracováním.

KLÍČOVÁ SLOVA

Node.js, Angular CLI, PDF, AI, Firebase, analýza obsahu souborů

TITLE

SaaS application for data processing from PDF files

ANNOTATION

This bachelor thesis deals with the design and implementation of a SaaS application for processing data from PDF files. The goal of this thesis is to create an application that allows users to efficiently search, extract, analyze and aggregate information from multiple PDF files simultaneously. The application will use the API of OpenAI for text processing and searching for relationships between data. This thesis includes analysis of existing solutions for processing PDF files and focuses on the challenges associated with data extraction and its processing.

KEYWORDS

Node.js, Angular CLI, PDF, AI, Firebase, analysis of content file

OBSAH

SEZNAM OBRAZKŮ 11
SEZNAM TABULEK 12
SEZNAM ZDROJOVÝCH KÓDŮ 13
SEZNAM ZKRATEK 14
ÚVOD 15
1. Vývojové frameworky 16
1.1. Node.js 16
1.1.1. Architektura řízení událostí a neblokující I/O 16
1.1.2. Ekosystém npm a klíčové moduly 17
1.1.3. Express.js 18
1.1.4. Zpracování dat pomocí Node.js 20
1.2. Angular 21
1.2.1. Adresářová struktura 21
1.2.2. Základní vlastnosti 22
1.3. Alternativní frontend frameworky 27
1.3.1. React 27
1.3.2. Vue.js 27
1.3.3. Ember.js 27
1.3.4. Srovnání frameworků 28
2. Práce s PDF soubory 30
2.1. Úvod do PDF 30
2.1.1. Struktura a vlastnosti PDF souborů 30
2.2. Technologie pro práci s PDF 31
2.2.1. PDF knihovny a nástroje 31
2.2.2. Extrakce textu z PDF 31
2.2.3. Analýza a zpracování PDF dokumentů 31
2.3. PDF v kontextu webových aplikací 32
2.3.1. Integrace PDF funkcionalit do webových aplikací 32
2.3.2. Případové studie: Využití PDF v praxi 32
3. Umělá inteligence 34
3.1. Úvod do umělé inteligence 34

3.1.1 Hlavní oblasti výzkumu 34
3.2 Algoritmy AI 34
3.2.1 Strojové učení 34
3.2.2 Hluboké učení 36
3.3 Aplikace umělé inteligence v analýze dat 38
3.4 Integrace umělé inteligence do webových aplikací 39
4. SaaS aplikace 41
4.1. Seznámení s prostředím SaaS 41
4.2. Základní úvahy o webové aplikaci SaaS 41
4.3. Potenciál pro budoucí vývoj a komercializaci 42
4.4. Zaměření na potřeby trhu a sladění s trendy 42
5. Aplikační řešení 43
5.1. Technologie použité při vývoji 43
5.1.1. OpenAI API 43
5.1.2. Postman 44
5.1.3. Webstrom 44
5.1.4. Firebase 44
5.2. Požadavky 45
5.2.1. Funkční požadavky 45
4.2.2 Nefunkční požadavky 45
5.3. Návrh databáze 46
5.3.1. Firebase Authentication 47
5.3.2. Firebase Cloud Sotrage 48
5.3.3. Firebase Firestore Database 48
5.4. Implementace aplikace 49
5.4.1. Grafické uživatelské rozhraní 49
5.4.2. Navigace v aplikaci 49
5.4.3. Tvorba a připojení k REST API 62
5.4.4. Testování endpointů 69
ZÁVĚR 70
POUŽITÁ LITERATURA 71
PŘÍLOHY 74

SEZNAM OBRAZKŮ

Obrázek 1: Vztahy mezi částmi NgModule (zdroj: [9]) 24
Obrázek 2: Statistika použití OpenAI modelů (zdroj: vlastní) 43
Obrázek 3: Navigace mezi vstupní, registrační a přihlašovací stránkou (zdroj: vlastní) 50
Obrázek 4: Navigace mezi možnými stránkami pro neautorizovaného uživatele (zdroj: vlastní) 51
Obrázek 5: Navigace mezi stránkami přihlášení, registrace a hlavního panelu (zdroj: vlastní) 52
Obrázek 6: Navigace mezi stránkami po autorizaci uživatele a vstupní stránkou (zdroj: vlastní) 53
Obrázek 7: Navigace mezi stránkami pro resetování hesla (zdroj: vlastní) 53
Obrázek 8: Navigace k nahrávání souborů (zdroj: vlastní) 54
Obrázek 9: Navigace k mazání souborů (zdroj: vlastní) 55
Obrázek 10: Navgiace mezi stránkami a modulárními okny (zdroj: vlastní) 56
Obrázek 11: Navigace v modulárních oknech k vytvoření vlastní podminky (zdroj: vlastní) 57
Obrázek 12: Navigace mezi modulárními stránkami pro práci se vzory (zdroj: vlastní) 58
Obrázek 13: Navigace mezi modulárními stránkami pro vytvoření vzoru (zdroj: vlastní) 59
Obrázek 14: Navigace mezi modulárními stránkami pro editaci vzoru (zdroj: vlastní) 60
Obrázek 15: Navigace mezi modulárními stránkami pro mazání vzoru (zdroj: vlastní) 61
Obrázek 16: Navigace mezi modulárními stránkami pro analýzu souboru (zdroj: vlastní) 62
Obrázek 17: Kolekce koncových bodů v programu Postman (zdroj: vlastní) 69

SEZNAM TABULEK

Tabulka 1: Tabulka porovnání funkcí frameworků frontend (zdroj: předpracováno podle [12]) 28
Tabulka 2: Funkční požadavky (zdroj: vlastní) 45
Tabulka 3: Nefunkční požadavky (zdroj: vlastní) 45

SEZNAM ZDROJOVÝCH KÓDŮ

Zdrojový kód 1: server.js - Koncový bod „/api/analyze-pdf-firebase“ (zdroj: vlastní).....44
Zdrojový kód 2: Stromové zobrazení databáze (zdroj: vlastní).....47
Zdrojový kód 3: Firebase Storage pravidlo (zdroj: vlastní).....48
Zdrojový kód 4: Firestore Database pravidlo (zdroj: vlastní).....49
Zdrojový kód 5: server.js – Konfigurace serveru Node.js pro poskytování aplikací Angular a ověřování uživatelů (zdroj: vlastní) 63
Zdrojový kód 6: server.js - Middlewarová funkce pro ověřování uživatelů na základě tokenů Firebase ID (zdroj: vlastní) 64
Zdrojový kód 7: server.js - Koncový bod API pro načítání souborů PDF s metadaty, ověřený přes Firebase (zdroj: vlastní).....65
Zdrojový kód 8: openai.service.ts - Služba Angular pro načítání dat PDF, zpracování ověřování a požadavků HTTP (zdroj: vlastní).....66
Zdrojový kód 9: server.js - Koncový bod API pro zahájení analýzy PDF s kontrolou platnosti (zdroj: vlastní).....67
Zdrojový kód 10: server.js - Fragment kódu pro stahování a analýzu obsahu PDF a jeho rozdělení podle stránek (zdroj: vlastní) 68
Zdrojový kód 11: server.js - Funkce pro analýzu textu ze stránky PDF pomocí modelu GPT od OpenAI s vlastními podmínkami zahrnutými v požadavku (zdroj: vlastní).....68
Zdrojový kód 12: server.js - Fragment kódu pro správu souběžných požadavků na zpracování více stránek PDF v dávkách (zdroj: vlastní) 69

SEZNAM ZKRATEK

HTTP	Hypertext Transfer Protocol
SaaS	Software as a Service
I/O	Input/Output
NPM	Node Package Manager
LTS	Long Term Support
NVM	Node Version Manager
CLI	Command Line Interface
API	Application Programming Interface
JSON	JavaScript Object Notaiton
MVC	Model-View-Controller
JS	JavaScript
DI	Dependency Injection
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
SPA	Single Page Application
URL	Uniform Resource Locator
DOM	Document Object Model
PDF	Portable Document Format
SDK	Software Development Kit
AI	Artificial Intelligence
NLP	Natural Language Processing
ML	Machine Learning
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
IE	Information Extraction
REST	Representational State Transfer Application
XML	Extebsible Markup Language
SOAP	Simple Object Access Protocol
UID	Uniqie Identifier

ÚVOD

Se současným rozvojem umělé inteligence (Artifical Inteligence – AI) a stále větším počtem modelů dostupných uživatelům lze očekávat jejich častější využívání. Modely AI přispívají k rychlejšímu a efektivnějšímu způsobu využití různých zdrojů dat a ke zvyšující se produkci výstupů v podobě textů, obrázků, zdrojových kódů, apod. S touto myšlenkou je tvořena i tato práce, která se bude zabývat vývojem softwarové aplikace, která má za cíl zvýšit rychlost získávání a analýzy dat z několika dokumentů ve formátu PDF. Hlavním úkolem této práce je vytvoření systému, který bude schopen získat požadované informace ze vstupních dokumentů a tyto údaje vyhodnotit na základě předem stanovených vstupních parametrů. K realizaci tohoto cíle aplikace využívá webové technologie, jako jsou Node.js a Angular, spolu s rozhraním OpenAI API, a klade si za cíl usnadnit a zefektivnit proces extrakce a analýzy dat z PDF souborů.

V práci bude uvedena oblast zkoumání, naznačen proces jejího vývoje, použité technologie a provozní možnosti. Rovněž bude uveden důvod výběru aplikace, z níž vyplývá rostoucí poptávka po aplikaci, která by zvládala složitosti spojené se zpracováním dat z dokumentů PDF. Jako analytické podklady výstupního řešení byl proveden přehled současných dostupných řešení s cílem zhodnocení přesnosti a užitečnosti každého z nich pro stejný druh problému. Tento druh analýzy je klíčový, protože poukazuje na současné nedostatky a upozorňuje na další oblasti, které by bylo možné rozvíjet pomocí moderních softwarových frameworků a technologií umělé inteligence. V rámci řešené oblasti je práce zaměřena i na technické aspekty zpracování souborů PDF, průzkum integrace umělé inteligence pro kvalitnější analýzu dat a použití těchto technologií v rámci aplikace.

První část práce se bude věnovat vývojovým frameworkům, které budou použity pro praktický výstup práce včetně srovnání s ostatními frameworky. Další část práce se bude soustředit na práci s PDF soubory, kde bude zmíněna struktura a vlastnosti PDF souborů a využití PDF souborů v kontextu webových aplikací. V třetí části budou představeny způsoby a možnosti využití AI včetně používaných frameworků, které s touto oblastí souvisí. Následující část bude popisovat prostředí SaaS, které bude v práci také využito a závěrečná kapitola bude shrnovat celý proces vývoje aplikačního řešení.

1. Vývojové frameworky

1.1.Node.js

Node.js je multiplatformní open source prostředí JavaScriptu. Je to oblíbený nástroj pro různé projekty a řešení. Jeho velkou výhodou pro frontendové webové vývojáře je programovací jazyk JavaScript na backendu, díky čemuž odpadá nutnost učit se nový jazyk.

1.1.1. Architektura řízení událostí a neblokující I/O

Node.js, který je považován za efektivní a škálovatelný zejména při vývoji webových aplikací, dokáže využívat kombinaci architektury řízené událostmi a neblokujících I/O. Taková metoda je v případě typických vícevláknových serverů výjimečná, a může tak výrazně odlišit Node.js a dát mu schopnost poskytovat kvalitní výkon, pokud jde o zpracování souběžných požadavků s minimálním využitím zdrojů. [1]

Jednovláknová událostní smyčka

Na rozdíl od běžného webového serveru, který pro každý požadavek vytváří nové vlákno, pak často dochází k výkonnostnímu omezení, když je obsazen. Namísto přístupu založeného na více vláknech je Node.js spuštěn na jediném vlákně. Toto jediné vlákno efektivně provádí více asynchronních požadavků pomocí smyčky událostí a funkce zpětného volání. [1]

Postup by mohl být popsán:

- Fronta událostí: Fronta událostí zpracovává příchozí požadavky a události, jako je síťový vstup/výstup nebo vypršení časovače.
- Smyčka událostí: Doba trvání smyčky událostí spočívá v nepřetržitém sledování a výběru další události z fronty ke zpracování.
- Provedení zpětného volání: Každá událost je spojena s určitou funkcí zpětného volání a související funkce je volána při každém výskytu události. Tyto funkce zpracovávají konkrétní operace související s událostí, například vyhledávání v databázi, zpracování souboru nebo odeslání reakce klientovi.

Navíc je důležité, aby v době, kdy je spuštěna funkce zpětného volání, událostní smyčka nezůstávala neaktivní. Poté pokračuje ve zpracovávání dalších událostí ve frontě, čímž se zajistí, že pracuje pouze jedno vlákno a nedochází k čekání na další. Když spuštěná funkce zpětného volání dokončí svůj zadaný úkol, předá oznámení smyčce událostí a jako takové se zpracovávají akce, které z procesu vyplývají. [1]

Neblokující I/O

Vstupní/výstupní operace, například operace s interakcí s databází nebo přístupem k systému souborů, bývají časově náročné. Tradiční blokační modely mají tu nevýhodu, že celá vlákna čekají na dokončení I/O operace, což prodlužuje dobu dokončení a zhoršuje výkon, zejména v případě mnoha souběžných požadavků.

Node.js naopak běží na modelu neblokujících I/O operací. Vzhledem k tomu, že je zahájena operace I/O, a proto, že nebude nutné čekat na její dokončení, Node.js okamžitě pokračuje zpracováním dalších událostí, které jsou přítomny ve frontě. Jakmile je I/O operace dokončena, provede se jako další funkce zpětného volání, která má zpracovat výsledky. Díky tomu, že algoritmus není blokovací, umožňuje Node.js zpracovávat více I/O operací bez blokování hlavního vlákna, což přispívá k vyšší odezvě a efektivitě aplikace. [1]

1.1.2. Ekosystém npm a klíčové moduly

Node.js je dodáván se správcem balíčků známým jako Node Package Manager. Npm je nástroj, který vývojáři používají k nalezení mnoha opakovaně použitelných balíčků JavaScriptu. Tyto sady pro vývoj softwaru, známé jako moduly s předpřipraveným kódem, pokrývají širokou škálu funkcí, které pomáhají vývojářům ušetřit čas a úsilí při vytváření aplikací. [2]

Npm: rozsáhlé uložiště open-source bloků

Npm poskytuje přístup k obrovské a neustále se rozšiřující sbírce balíčků zdarma, která zahrnuje téměř všechny aspekty vývoje softwaru. [2]

Tyto balíčky, nazývané moduly, lze použít k mnoha účelům, patří mezi ně:

- Webové frameworky: Nejoblíbenější framework Express.js má výkonný základ pro vytváření webových aplikací a rozhraní API. Poskytuje funkce, jako je směrování, middleware a šablonování. [3]
- Zpracování dat a nástroje: Množství knihoven se věnuje úlohám, jako je zpracování dat a manipulace s nimi, připojení k databázím, provádění matematických operací a práce se souborovými systémy.
- Nástroje pro testování a ladění: Npm pomáhá při vývoji tím, že nabízí širokou škálu testovacích frameworků a ladicích nástrojů, které zajišťují kvalitu kódu a také usnadňují řešení chyb.
- Front-End knihovny: Npm spravuje například React, Angular a Vue.js, což umožňuje hladkou integraci s Node,js.

Zjednodušení vývoje pomocí npm

Npm je nejen zodpovědný za poskytování rozsáhlé databáze balíčků, ale také umožňuje efektivně spravovat tyto balíčky pomocí těchto nástrojů v rámci softwarových projektů.

- Instalace a správa balíčků: Npm CLI je nástroj, který umožňuje rychle instalovat, aktualizovat a odstraňovat balíčky v projektech. Provádí také správu závislostí, čímž odstraňuje nutnost samostatného vyhledávání potřebných balíčků a jejich konkrétních verzí. [2]

- Verzování a sémantické verzování: Npm používá sémantické verzování, což je standardizovaná metoda přiřazování čísel verzí balíčkům způsobem, který zaručuje návaznost a přesnost, pokud jde o správu závislostí.

Významné moduly vývoje SaaS

Výběr konkrétních modulů závisí na specifických potřebách každého projektu, ale některé klíčové kategorie jsou pro vývoj aplikací SaaS zvláště důležité.

- Webové frameworky: Představují základ pro vývoj rozhraní RESTful API schopných zpracovávat požadavky a odpovědi HTTP.

- Přístup k datům a manipulace s nimi: Za nezbytné je třeba považovat knihovny pro interakci s databází, zpracování datových formátů, jako je PDF, a provádění analýz.

- Ověřování a autorizace: Autentizace uživatelů, autorizace a mechanismy řízení přístupu jsou nezbytnými součástmi pro přístup software jako služba.

1.1.3. Express.js

Express.js je populární a minimalistický webový framework postavený na Node.js. Expess je uznáván pro svou flexibilitu a nenáročnost, což je výhodné v tom, že lze vytvářet webové aplikace a API, které nejsou svázány pevnými strukturálními limity. [3]

Vyhody Express

- Minimalismus a flexibilita: Express uplatňuje přístup „hands-off“, který zahrnuje svobodu vývojářů řídit strukturu svých aplikací podle svých preferencí. Tento typ flexibility umožňuje přizpůsobit aplikaci na míru a předejít případným nedostatkům způsobeným silněji formulovanými frameworky. [5]

- Výkonnost: Architektura Express je postavena na asynchronním, neblokujícím I/O modelu Node.js. Díky tomu dokáže zpracovávat množství požadavků najednou s velkou efektivitou. Takže mít škálovatelnou aplikaci není pro Express problematické. [3] [5]

- Ekosystém middlewaru: Express se chlubí obrovskou a rozsáhlou komunitou, která neustále doplňuje různé druhy middlewaru o nové a zajímavé nápady. Díky opakovaně použitelným komponentám lze tyto úlohy, jako je směrování, šablonování, ověřování a další, využívat ve zkrácené podobě, což v konečném důsledku šetří čas potřebný na tyto činnosti. [4]

- Snadné používání: Rozhraní Express API je skvěle zdokumentované a intuitivní, takže se ho snadno naučí i začátečník, zejména pokud již ovládá základy JavaScriptu a Node.js. [4] [5]

Middleware

Funkce middlewaru tvoří zásadní část Express.js, a tím poskytují řadu možností, jak přeměnit aplikaci na sofistikovaný systém správy. Tyto funkce se nacházejí v cyklu požadavek-odpověď aplikace, mají přístup k objektům požadavku *(req)* a odpovědi *(res)* a také mohou přecházet na další middlewarovou funkci. Prostřednictvím této postupné operace lze zajistit principy modularizace logiky webového serveru, což poskytuje základ pro opakovanou použitelnost a udržovatelnost. [6] [7]

Kličové funkce middleware

- Modifikace požadavku/odpovědi: Moduly middlewaru jsou schopny kontrolovat příchozí požadavky a odpovídat na odchozí požadavky, díky čemuž lze s minimálním úsilím provádět úlohy, jako je ověřování, validace vstupů a konverze obsahu.
- Spuštění kódu: Přes middleware se velmi dobře hodí pro účely jakéhokoli kódu, který je třeba spustit během cyklu požadavek-odpověď. Sledují se tak procesy, jako je logování, interakce s databází nebo obchodní logika.
- Ukončení požadavku/odpovědi: Funkce middlewaru umožňuje ukončit celý cyklus požadavek-odpověď, kdykoli k tomu dostane příležitost, odesláním odpovědi klientovi. Často se tak děje v případě zpracování chyb nebo rychlé zpětné vazby v podobě odpovědí, které neprocházejí dalšími vrstvami směrování.
- Navázání řetězce: Volání funkce *next()* umožňuje posunout provádění o krok vpřed k další middlewarové funkci v řetězci. Umožňuje posunout řetězec akcí, které lze realizovat pro určité požadavky. [6] [7] [10]

Typy middlewaru

- Middleware na aplikační úrovni: Provádějí své činnosti na základě každého příchozího požadavku na server. Příkladem je logování požadavků a také rozbor cookies.

- Middleware na úrovni směrovače: Při aplikaci na konkrétní cesty vysoce regulují přesnost v rámci instance směrovače. Příkladem middlewaru na úrovni směrovače jsou autentizační kontroly před chráněnými cestami.

- Middleware pro ošetření chyb: Znamená zpracování chyb aplikace, které mohly způsobit zastavení serveru, a zobrazení přívětivé odpovědi na chybu klientovi.

- Integrovaný middleware: Kromě toho, že Express poskytuje built-in middleware, jako je *express.static* nebo *express.json*, pro běžné úlohy obsluhy statických souborů nebo parsování užitečných souborů JSON, umožňuje také middleware definovaný uživatelem.

- Middleware třetích stran: Kromě pokročilého ekosystému Express existuje také spousta možností middlewaru třetích stran, které se doporučují hlavně při řešení autentizace, správy relací a mnoha dalších úloh. [4] [5] [6] [7]

Integrace s metodami HTTP

Funkce middlewaru lze obvykle navrhnout tak, zda vyhovují jednotlivým typům požadavků HTTP, jako jsou GET, POST, PUT a DELETE. Tato funkce umožňuje vývojářům integrovat chování specificky přizpůsobené různým částem aplikace reprezentovaným prostřednictvím cest. [7]

1.1.4. Zpracování dat pomocí Node.js

Node.js jako hlavní část aplikace SaaS určuje, jakým způsobem zpracovává data získaná ze souborů PDF, za které je zodpovědný. Díky začlenění asynchronnosti a událostního řízení systému Node.js a ekosystému specializovaných knihoven je Node.js schopen efektivně transformovat neorganizovaná data PDF do strukturované a použitelné podoby dat.

Extrakce textu a struktury pomocí nástroje „pdf-parse“

Při zpracování dat se první krok dotýká operace extrakce textového obsahu a strukturálních informací z dokumentů zaslaných v nahraném formátu PDF. V tomto ohledu přijde vhod knihovna npm „pdf-parse“ - zařazená do kategorie *devDependencies*. Poskytuje funkce pro:

- Extrakci textu: Získání přístupu k textu ze souborů PDF s ohledem na různé styly písma, kódování a rozvržení.

- Analýza metadat a struktury: Získání údajů o struktuře dokumentu, např. čísel stránek, nadpisů, tabulek, které se budou hodit v dalších fázích zpracování a analýzy.

- Zpracování složitých rozvržení: „pdf-parse“ si poradí s různými typy souborů PDF a jejich rozvržením, včetně souborů se složitým formátováním nebo vloženými obrázky.

Text a strukturální informace tvoří podstatu pro další kroky zpracování a analýzy.

Transformace a zpracování dat

Po extrakci hrubého obsahu může být nutná komplexní příprava, například jeho formátování do strukturovaného a použitelného materiálu. To může zahrnovat:

- Čištění a normalizaci textu: Operace, jako je odstranění přebytečných znaků, oprava chyb a zachování jednoty formátovaného textu.
- Regulární výrazy: Použití regulárních výrazů jako nástroje pro detekci a extrakci vzorů/fragmentů, jako jsou data, jména a číselné hodnoty.
- Mapování a strukturování dat: Převod dat ze surového formátu do strukturovaného formátu, např. JSON nebo CSV, pro snadnější manipulaci a analýzu.

V důsledku toho lze integrovat externí zdroje dat nebo rozhraní API a rozšířit tak získané informace. Například geolokační údaje by mohly doplnit extrahované adresy a seznamy jmen pomocí externích databází.

1.2. Angular

Angular je produktem společnosti Google a tým Google je zodpovědný za jeho neustálý vývoj a údržbu. Jedná se o výkonný frontendový framework založený na TypeScriptu, který pomáhá při vytváření a testování jednostránkových aplikací se standardizovanou architekturou a komponentami díky pevně organizované a správné struktuře. Pomáhá modularizaci aplikací realizovat obousměrné datové vazby pro synchronizaci modelů a vazby zobrazení, což dále zvyšuje modularitu, testovatelnost a možnost opakovaného použití. Sbírka rozhraní API v aplikaci Angular je poměrně široká a obsahuje obsah pro navigaci, zpracování formulářů, komunikaci HTTP a různé další funkce. Angular doprovází také prosperující ekosystém knihoven třetích stran. Angular CLI urychluje vývoj automatizací rutinních operací a umožňuje také open source vykreslování na straně serveru. Angular Universal zrychluje operace a zvyšuje viditelnost webových stránek ve webových vyhledávačích. [8]

1.2.1. Adresářová struktura

Struktura aplikace Angular využívá rozhraní Angular CLI k vytvoření předdefinované struktury, která obsahuje klíčové komponenty a soubory nezbytné pro spuštění a vývoj aplikace. [8] Základní struktura obsahuje:

- *src/* - Zdrojový adresář
- *app/* - Hlavní adresář aplikace obsahující komponenty, moduly a služby.
    - *app.component.{ts,html,css,spec.ts}* – Kořenová komponenta aplikace, která slouží jako startovací bod, z kterého se začíná budovat uživatelské rozhraní aplikace. Soubory *.ts, .html, .css* představují logiku jazyka TypeScript, šablonu a styly komponenty. Soubor *.spec.ts* obsahuje testy komponenty.
    - *app.module.ts* – Kořenový modul aplikace, který deklaruje komponenty a importuje potřebné Angular moduly.
- *assets/* **-** Adresář pro statické zdroje, jako jsou obrázky, ikony nebo globální styly.
- *environments/* **-** Obsahuje konfigurační soubory pro různá prostředí, např. vývojové a produkční.
- *angular.json* - Konfigurační soubor pro Angular CLI. Definuje, jak jsou projekty sestaveny a nasazeny, včetně konfigurací pro testy.
- *package.json* - Seznam závislostí a skriptů pro správu projektu. Obsahuje také informace o projektu.
- *tsconfig.json* - Konfigurační soubor pro TypeScript, určuje pravidla a možnosti pro kompilátor TypeScriptu.
- *node_modules/* - Adresář, kde npm ukládá všechny externí balíčky a knihovny, na kterých projekt závisí.

1.2.2. Základní vlastnosti

Moduly

Dle [8] [9] jsou moduly Angular jsou hlavními stavebními kameny každé aplikace Angular. Určují, které komponenty, směrnice a služby se v systému kombinují, a propagují přesný výsledek. Moduly poskytují možnost pracovat s oddělenými, ale neporušenými částmi aplikace tím, že každé jednotce dávají její rozsah závislostí.

NgModule je anotátor s nadefinovanými schopnostmi, který poskytuje seznam komponent, direktiv a pipe obsažených v modulu spolu s poskytovanými službami a dalšími moduly, na kterých závisí. Každý program Angular má alespoň jeden kořenový modul, známý jako AppModule, který slouží jako bezpečnostní brána aplikace. Struktura NgModule je rozčleněna do specifických sekci, které mají následující funkce:

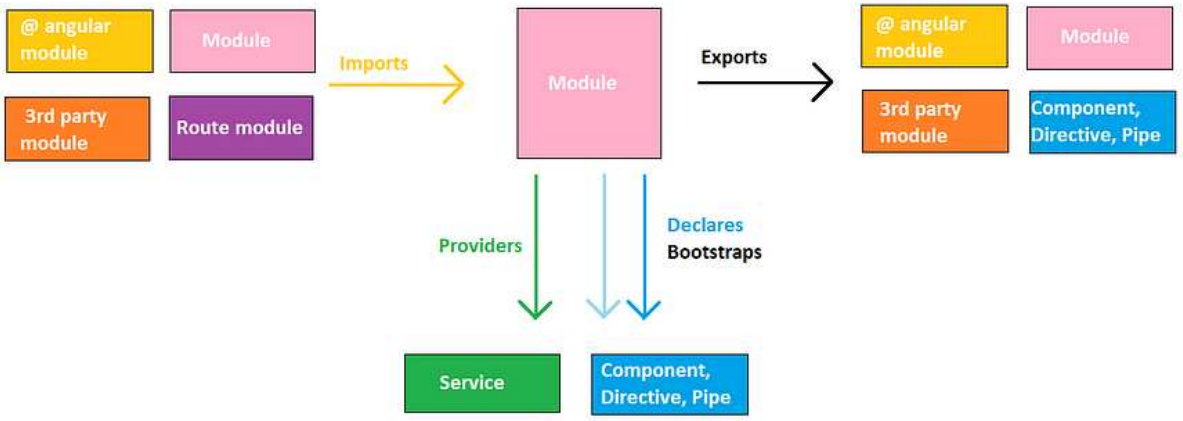
- *declarations* – Obsahuje komponenty, direktivy a pipe přidružené k modulu. Tyto entity mohou být použity pouze v rámci modulu, pokud nejsou explicitně exportovány.
- *imports* – Importuje funkcionality z jiných modulů, jako jsou *RouterModule* nebo moduly třetích stran, a umožňuje tak používání jejich exportovaných deklarací.
- *exports* – Umožňuje zpřístupnění komponent, direktiv a pipe z modulu pro jiné moduly.
- *providers* – Definuje služby, které mají být vytvořeny injektorem závislostí Angularu.
- *bootstrap* – Definuje komponentu, která má být jako první načtena a zobrazena.

Moduly tvoří jádro služby DI systému Angular. Zprostředkovatelé v NgModule mohou být injektováni jako komponenty, direktivy, pipy v konkrétním modulu. Výsledkem je, že vývojáři mohou definovat závislosti, které lze snadno opakovaně používat a spravovat.

Další vlastností NgModule je „lazy loading“ modulů, který je však také podporován. Tato technika umožňuje aplikaci načítat moduly podle potřeby, a to tedy může velmi zkrátit dobu načítání aplikace jako celku. Angular použije směrovač k definování líných cest, které mají být načteny. Následující řádky se věnují běžným typům modulů, které hrají účelnou roli v různých aspektech vývoje softwaru:

- *Funkční moduly* **-** Definují specifickou funkčnost, která není přímo související s uživatelským rozhraním, jako jsou služby a utility.
- *Komponentové moduly* **-** Skupina komponent, direktiv a pipů, které jsou související a sdílejí společný kontext.
- *Sdílené moduly* **-** Moduly, které exportují komponenty, direktivy a pipy tak, že jsou dostupné v mnoha různých částech aplikace.
- *Core moduly* **-** Moduly poskytující služby, které mají být jedinečné v celé aplikaci a obvykle se importují pouze v kořenovém modulu.
- *Router moduly* **-** Konfigurují směrování na úrovni modulu a často jsou spojené s lazy loadingem.

Obrázek 1. ilustruje základní vztahy mezi různými částmi NgModule:



Obrázek 1: Vztahy mezi částmi NgModule (zdroj: [9])

Komponenty

Komponenty představují rozumovou část a vizuální prvky uživatelského rozhraní a jsou nezbytné pro vývoj modulárních a dobře strukturovaných webových aplikací. Komponenty v Angularu jsou třídy napsané v jazyce TypeScript s umístěným dekorátorem *@Component*, který jim dává metadata definující prvky pro zobrazení uživatelského rozhraní a styly CSS. Dekorátor *@Component* má vestavěnou možnost selektoru, která umožňuje využití komponenty v kódu HTML, jedná se o:

- *selector* - Unikátní identifikátor komponenty, používaný v HTML.
- *templateUrl* - Cesta k souboru s HTML šablonou komponenty.
- *styleUrls* - Pole cest k souborům se styly CSS specifickými pro komponentu.

Šablony komponent nesou hlavní odpovědnost za to, jakým způsobem bude struktura HTML zachycena po vykreslení do komponenty. [8] Angular umožňuje specifikovat vazební hodnoty standardních prvků HTML s dynamicky vytvářenými hodnotami dat, funkcemi pro přihlášení k událostem a směrnicemi, které usnadňují konstrukci uživatelského rozhraní.

Styly lze do komponenty přidávat pomocí *styles* nebo prostřednictvím externích *styleUrls*, což umožňuje co nejjemnější izolaci stylů specifických pro komponentu. [8] Data binding je funkce Angularu, která umožňuje synchronizaci dat mezi šablonou komponenty a její třídou.

Angular podporuje několik typů vazeb:

- Interpolace: *{{ value }}* – pro vložení hodnot do HTML.

- Property binding: *[property]="value"* – pro nastavení vlastností elementů.
- Event binding: *(event)="handler()"* – pro poslouchání událostí.
- Two-way binding: *[(ngModel)]="property"* – pro dvousměrnou vazbu dat.

Komponenty v Angularu mají životní cyklus, který umožňuje vykonat kód v určitých fázích existence komponenty, například při jejím vytvoření, změně dat nebo zničení. Angular poskytuje různé „lifecycle hooks“ jako jsou *ngOnInit, ngOnChanges, ngOnDestroy* atd., které umožňují zasahovat do životního cyklu komponenty. [8] [9]

Dependency Injection

Angular DI je vzor, který vyniká tím, že umožňuje poskytovat služby dalším částem aplikace. DI v Angularu je funkce, která externalizuje definice závislostí komponent a služeb, čímž poskytuje flexibilitu, protože není nutné vytvářet přímo z komponent a služeb.

Jádrem systému Angular DI je injektor a provider. Injektor závislostí slouží k vytváření a udržování vztahů uvnitř kontejneru. Každá aplikace má obvykle alespoň jeden kořenový injektor, ale může existovat i několik injektorů dále v hierarchii komponent. [8] Provider může pomoci nastavit položky, které má injektor poskytnout příslušným třídám nebo funkcím. Provider může být registrován na různých úrovních aplikace Angular, což znamená, že pro různé části aplikace mohou existovat různé instance služeb. [8] [9]

Služby

Služby Angular mají v architektuře aplikace významnou důležitost, protože umožňují opětovné použití kódu a dat mezi jednotlivými komponentami. To vede k lepšímu oddělení kódu a logiky.

Typické využití služeb zahrnuje:

1. Sdílení dat mezi komponentami.
2. Implementace aplikační logiky.
3. Správa API volání.
4. Poskytování funkcionality přístupu k datům.

Služby v Angularu jsou napsány v jazyce TypeScript stejně jako komponenty. Pro identifikaci služby se používá dekorátor *@Injectable()*, který umožňuje Angularu vytvořit její instanci jako závislost pro komponenty nebo jiné služby. Dekorátor *providedIn: 'root'* znamená, že služba

bude dostupná globálně v aplikaci. Služby jsou injektovány do komponent nebo jiných služeb prostřednictvím konstruktoru. [8] [9]

Routování

Routování lze považovat za jádro každého SPA, protože umožňuje uživatelům procházet různými díly aplikace bez nutnosti načítat stránku a zachovávat všechny uživatelské změny. K vyřešení navigačního procesu dojde, jakmile Angular Router zachytí konkrétní požadovanou adresu URL, určí odpovídající cestu a zobrazí požadovanou komponentu, což nejen usnadňuje navigaci uvnitř aplikace, ale také přispívá k manipulaci s historií prohlížeče, línému načítání modulů pro účely výkonu a hlídání ochrany cest. Obecně se routování v Angular konfiguruje pomocí modulu *RouterModule*. Angular nejenže usnadňuje navigaci bez tradičních spojení, ale také poskytuje dvě směrnice: *routerLink*, která zobrazuje dynamický odkaz, a *routerLinkActive*, která označuje vybraný a aktivní odkaz. Angular router poskytuje guardy, které lze použít k ochraně cest, čímž umožňuje funkce, jako je ověřování uživatelů. [8] [9]

HttpClient

HttpClient je snadná, odolná a univerzální knihovna pro provádění požadavků HTTP, která je součástí modulu @*angular/common/http*, je k tomu výkonným nástrojem.

HttpClient v aplikaci Angular poskytuje sadu vyvinutých funkcí pro odesílání http požadavků a interakci odpovědí v aplikacích Angular. Mezi tyto funkce patří podpora typů odpovědí, interceptorů, pozorovatelností a zpracování chyb. [8] [10]

Především je nutné importovat modul *HttpClientModule* do hlavního modulu aplikace, ve kterém se má HttpClient používat. Tento krok umožňuje injektovat službu HttpClient do middlewaru v celé aplikaci. [10]

Mezi funkce a vlastnosti HttpClient patří [10]:

- Typové odpovědi: HttpClient umožňuje specifikovat očekávaný typ odpovědi, což usnadňuje práci s daty.
- Observables: HttpClient metody vracejí Observable, což umožňuje efektivně pracovat s asynchronními datovými toky a reagovat na odpovědi v reálném čase.
- Interceptory: Poskytují možnost zachytávat a manipulovat s HTTP požadavky a odpověďmi globálně.

- Error Handling: HttpClient podporuje snadné zacházení s chybami pomocí operátorů RxJS, jako je catchError.

1.3. Alternativní frontend frameworky

Tato kapitola je věnována rešerši alternativních frontend frameworků a jejich základnímu srovnání dle vybraných kategorií, které mohou pomoci s výběrem frameworku z různých hledisek.

1.3.1. React

React, který se zrodil ve společnosti Facebook, se stal dominantní jednotkou v oblasti knihoven JavaScriptu pro tvorbu uživatelských rozhraní. Základní ideou je architektura založená na komponentách, která vývojářům umožňuje vytvářet opakovaně použitelné prvky uživatelského rozhraní a efektivně spravovat jejich vnitřní stav. [11]

Tento přístup podporuje modularitu, opakovanou použitelnost kódu a snadnou údržbu. Implementace virtuálního DOM v Reactu zajišťuje efektivní vykreslování díky minimalizaci přímé manipulace se skutečným DOM, což vede k optimalizaci výkonu. Flexibilita tohoto řešení umožňuje vývojářům vybírat z rozsáhlého ekosystému knihoven a nástrojů a přizpůsobit tak vývoj konkrétním potřebám projektu. Navíc je díky mírné křivce učení a rozsáhlé aktivní komunitě přístupný vývojářům s různou úrovní dovedností a poskytuje dostatečnou podporu a zdroje. [12]

1.3.2. Vue.js

Vue.js se rychle etabloval jako progresivní framework JavaScriptu, který je známý svou dostupností a rozmanitostí. Používá komponentový přístup podobně jako React, což usnadňuje vytváření modulárních a udržovatelných komponent uživatelského rozhraní. Vue.js se vyznačuje mírnou křivkou učení a intuitivní syntaxí, takže je ideální volbou pro začínající i zkušené vývojáře. Nabízí dobře definovanou strukturu při zachování flexibility, což umožňuje postupnou integraci do stávajících projektů nebo plnohodnotný vývoj jednostránkových aplikací. Vue.js se také může pochlubit funkcemi, jako je vestavěná správa stavů pomocí Vuex a možnosti směrování pomocí Vue Router, což zjednodušuje pracovní postupy při vývoji. Rostoucí komunita a ekosystém poskytují dostatek zdrojů a podpory. [11]

1.3.3. Ember.js

Ember.js se ukazuje jako rozvinutý a funkčně dokonalý framework MVC určený pro tvorbu nákladných webových aplikací. Zdůrazňuje zvyklosti před konfigurací a poskytuje vývojářům strukturované a názorově vyhraněné prostředí. Ember.js obsahuje funkce jako Ember CLI pro

tvorbu kostry projektu a procesů sestavování, Ember Data pro správu perzistence dat a šablony Handlebars pro dynamické vykreslování uživatelského rozhraní. Ember.js má sice ve srovnání s některými jinými frameworky prudší křivku učení, ale jeho důraz na zavedené vzory a osvědčené postupy podporuje konzistenci kódu a dlouhodobou udržovatelnost. Díky tomu je vhodnou volbou pro komplexní projekty vyžadující vysokou míru struktury a organizace. [11]

1.3.4. Srovnání frameworků

Jak si Angular stojí v porovnání s nejznámějšími frameworky pro frontendové programování, ukazuje Tabulka 1. Spolehlivá architektura MVC, integrace jazyka TypeScript a rozsáhlé nástroje systému Angular poskytují silný základ pro vytváření komplexních aplikací s dlouhodobou životaschopností. Pro velké projekty a vývoj aplikací ve firemním prostředí je Angular nejlepší možnou volbou. Přestože Angular může představovat strmější křivku učení, investice do zvládnutí jeho konceptů se promítne do lépe organizované, udržovatelné a snadno škálovatelné kódové základny, což dokonale odpovídá cílům praktického výstupu bakalářské práce.

Tabulka 1: Tabulka porovnání funkcí frameworků frontend (zdroj: předpracováno podle [12])

Funkce	React	Vue.js	Angular	Ember.js
Architektura	Založená na komponentech	Založená na komponentech	MVC	MVC
Křivka učení	Jemná	Jemná	Strmější	Strmější
Komunita a podpora	Velká a aktivní	Rostoucí a aktivní	Velká a aktivní	Zralá a aktivní
Škálovatelnost	Škálovatelný	Škálovatelný	Vysoce škálovatelný	Vysoce škálovatelný
Podpora TypeScriptu	Volitelná	Volitelná	Integrovaná	Integrovaná

Správa stavu	Externí knihovny (Redux)	Vuex (integrovaná)	NgRx (externí)	Ember Data
Routování	React Router (externí)	Vue Router (zabudovaná)	Angular Router (zabudovaná)	Ember Router (zabudovaná)

2. Práce s PDF soubory

Formát PDF je důležitou součástí moderní správy dokumentů a elektronické komunikace a nabízí univerzální formát pro výměnu informací a jejich ukládání na různých platformách. Před zahájením vývoje vysoce specifické aplikace, která umožňuje provádět vyhledávání i analýzu napříč řadou souborů PDF, je třeba proniknout do základních aspektů PDF, zakládají jejich existenci a vývoj. Tento základ je nejen základem pro pochopení toho, co tak často přitahuje používání PDF v online komunikaci, ale co je také důležité, pro získávání dat pomocí jeho složité struktury.

2.1. Úvod do PDF

PDF je všestranný souborový formát, který umožňuje ukládat a sdílet dokumenty nezávisle na softwaru nebo operačním systému, na kterém byly vytvořeny. Nabízí věrné zobrazení textu, obrázků, formátování a rozvržení dokumentu, což z něj činí populární volbu pro sdílení dokumentů a archivaci informací. [13]

2.1.1. Struktura a vlastnosti PDF souborů

Soubor PDF se skládá ze tří hlavních komponent:

1. Objekty: Dokument je rozdělen na jednotlivé objekty, které mohou obsahovat text, obrázky, písma, formátování a další prvky dokumentu.
2. Křížová odkazová tabulka: Tato tabulka ukládá umístění každého objektu v souboru, což umožňuje programu PDF rychle najít a zobrazit potřebné informace.
3. Záhlaví: Záhlaví souboru PDF obsahuje informace o verzi PDF, použité písma a další metadata. [13]

Díky své struktuře nabízí PDF několik výhod oproti jiným formátům dokumentů:

- Universalita: Soubory PDF lze zobrazit na téměř jakémkoli zařízení s nainstalovaným bezplatným prohlížečem PDF.
- Věrnost obsahu: Formát PDF zachovává přesné zobrazení textu, obrázků a formátování dokumentu bez ohledu na použitý software nebo operační systém.
- Zabezpečení: Soubory PDF lze chránit heslem a šifrováním, což pomáhá chránit důvěrné informace.
- Komprese: Formát PDF může komprimovat text a obrázky, což vede k menším velikostem souborů.

2.2. Technologie pro práci s PDF

S narůstající potřebou automatizace a zpracování velkého množství dat je stále důležitější porozumět a efektivně využívat technologie pro práci s PDF soubory. Tato část se věnuje popisu různých nástrojů a knihoven pro efektivní manipulaci s PDF dokumenty, zahrnující extrakci textu, komplexní analýzu a zpracování.

2.2.1. PDF knihovny a nástroje

Pro vývojáře a výzkumníky, kteří se specializují na manipulaci a zpracování PDF dokumentů, jsou klíčové knihovny a nástroje pro práci. Apache PDFBox je jednou z nejpoužívanějších nástrojů v této oblasti. Je to open-source Java knihovna, která umožňuje vytváření, zpracování a extrakci obsahu z PDF dokumentů. Díky široké škále funkcí, jako je zpracování textu, obrázků a metadat, je PDFBox vhodným řešením pro serverové aplikace a velké objemy dat.

Adobe Acrobat SDK je dalším nástrojem, který nabízí rozhraní pro integrování funkcionalit PDF do různých aplikací. Acrobat SDK umožňuje přistupovat k rozsáhlým funkcím pro manipulaci s PDF, včetně konverzí, bezpečnostních nastavení a digitálních podpisů. [14]

2.2.2. Extrakce textu z PDF

Mnoho aplikací si vyžaduje extrakci textu z PDF s cílem automatizovaného zpracování dokumentů a analýzy dat. Vynikající schopnosti extrakce textu v Apache PDFBox umožňují efektivní přístup k textovým obsahům uloženým v PDF souborech. Obsahuje podporu pro rozlišování různých formátů textu a uchovávání struktury dokumentů. [14]

Pro rozšiřující požadavky, jako je práce s komplexními uspořádáním a multimediálními prvky, může být užitečné spojit více nástrojů. Adobe Acrobat SDK může být užitečný pro integraci, protože nabízí pokročilé možnosti zpracování obsahu PDF. [15]

2.2.3. Analýza a zpracování PDF dokumentů

Průzkum PDF dokumentů zahrnuje rozpoznání struktury, obsahu a sémantického významu informací. Nástroje poskytované Apache PDFBox umožňují detailní analýzu dokumentů, včetně extrakce metaúdajů a analýzy bezpečnostních atributů. Pro vývojáře a výzkumníky, kteří potřebují hlubší porozumění interakcím mezi různými prvky dokumentu, je klíčové umět modifikovat a analyzovat obsah dokumentů na úrovni objektů. [14]

Spolu s Adobe Acrobat SDK uživatelé dokáží vytvářet sofistikované aplikace pro zpracování PDF a využívat pokročilé funkce, jako jsou interaktivní formuláře, komentáře a editace dokumentů. [15]

2.3. PDF v kontextu webových aplikací

2.3.1. Integrace PDF funkcionalit do webových aplikací

Webové aplikace hrají v dnešní digitální éře klíčovou roli ve mnoha oblastech, jako je vzdělání, obchod, státní správa a zábava. Integrace funkcionalit pro práci s dokumenty ve formátu PDF je nezbytná pro zajištění komplexního a efektivního fungování webových aplikací. Integrace PDF do webových aplikací přináší mnoho výhod, včetně:

- **Vylepšená uživatelská zkušenost**: Uživatelům je poskytován konzistentní a pohodlný způsob práce s PDF dokumenty bez ohledu na operační systém nebo zařízení, které používají.
- **Rozšířené možnosti sdílení**: Pomáhá s výměnou PDF dokumentů mezi uživateli a umožňuje společnou práci na dokumentech v reálném čase.
- **Zvýšená bezpečnost**: Umí umožnit ochranu citlivých informací v PDF souborech přes hesla, šifrování a další bezpečnostní prvky.

Integrace PDF funkcionalit do webových aplikací může být provedena několika různými způsoby. Mezi nejběžnější patří:

- **Použití knihoven JavaScript**: Pro webové aplikace existuje mnoho knihoven JavaScript, které usnadňují integraci funkcionality PDF. Obvykle tyto knihovny poskytují funkce pro prohlížení, generování, upravování a sdílení dokumentů ve formátu PDF. PDF.js a jsPDF jsou mezi populárními knihovnami pro práci s PDF v JavaScriptu. [16] [17]
- **Použití server-side nástrojů**: Nástroje pro práci s PDF umožňují generovat, upravovat a konvertovat PDF dokumenty na straně serveru. Webové aplikace mohou integrovat tyto nástroje pomocí API nebo webových služeb. [16]

Výběr správné metody integrace funkcí PDF do webové aplikace závisí na konkrétních požadavcích aplikace a technických schopnostech vývojářů.

2.3.2. Případové studie: Využití PDF v praxi

Využití funkcionalit PDF v webových aplikacích je relevantní ve mnoha různorodých oblastech. Níže jsou uvedeny některé příklady praktického využití PDF v webových aplikacích:

- Vzdělávání: PDF mohou webové stránky škol a vzdělávacích institucí použít k distribuci učebních materiálů, studijních materiálů a osvědčení o absolvování kurzů. Také mohou učitelé vytvářet online testy a zadávat úkoly studentům pomocí PDF.
- Obchod: Pro generování faktur, dodacích listů a záručních listů mohou e-commerce webové stránky využít formát PDF.
- Státní správa: Oficiální dokumenty, formuláře a žádosti mohou být zveřejňovány na vládních webových stránkách pomocí formátu PDF.
- Zábava: K distribuci elektronických knih, časopisů a komiksů mohou webové stránky pro sdílení obsahu využívat formát PDF.

3. Umělá inteligence

3.1. Úvod do umělé inteligence

Umělá inteligence (AI) patří k předním technologiím, které utvářejí trend 21. století a určují směr, změny a zlepšování. AI je celkové označení schopnosti strojů imitovat inteligenci člověka. Jedná se o výrobu strojů s překlenovacím systémem, které myslí jako lidé, rozumí řeči, identifikují vzory, rozpoznávají problematiku a také se učí z dat tak, aby se v průběhu času zlepšovaly. [18] [19]

Umělá inteligence má širokou škálu využití a obvykle se dělí do dvou kategorií - úzká umělá inteligence, která je určena k plnění konkrétního úkolu jako je např. rozpoznávání pouze obličeje nebo pouze vyhledávání na internetu či pouze řízení automobilu, a obecná umělá inteligence, která dokáže plnit jakýkoli intelektuální úkol, který dokáže plnit člověk. [20]

3.1.1 Hlavní oblasti výzkumu

Kontext výzkumu umělé inteligence je zastoupen podoblastmi, jako jsou strojové učení, neuronové sítě, robotika, NLP a počítačové vnímání. V algoritmech strojového učení jsou počítače schopny samy provádět rozhodnutí a prognózy na základě dat, s nimiž jsou vyškoleny pracovat. Počítače se tedy učí nejen bez toho, aby byly naprogramovány na konkrétní úkoly, ale také na základě dat, která mají k dispozici. Neuronové sítě, navržené pro napodobení fungování lidského mozku, mají více vrstev a vzájemně propojených uzlů, stejně jako lidská mysl při svých rozhodovacích procesech. Roboti aplikují algoritmy umělé inteligence na strojovou technologii, která autonomně nebo poloautonomně vykonává posloupnost takových úkolů. NLP se tímto omezuje na programování počítačů, které mohou zpracovávat a analyzovat obrovské objemy dat v přirozeném jazyce, zatímco počítačové vidění má ambici vyvíjet algoritmy, které počítačům umožní interpretovat a rozhodovat se na základě vizuálních dat o světě. [18] [19] [20]

3.2 Algoritmy AI

3.2.1 Strojové učení

Strojová inteligence se řídí především algoritmy umělé inteligence, které jsou následně základem systémů umělé inteligence a umožňují těmto strojům vykonávat úkoly, které běžně vykonávají lidé. Algoritmy strojového učení lze rozdělit do různých kategorií, od jednoduchých systémů založených na pravidlech pro automatizaci až po pokročilé modely strojového učení a hlubokého učení, které jsou schopny se samy rozvíjet a zobecňovat. Strojové učení, které je podoborem umělé inteligence, představuje soubor metod, jejichž využitím jsou počítače schopny vyvíjet algoritmy založené na historických datech, které vytvářejí cestu pro tvorbu

těchto modelů. Tato velice dynamická a stále se zdokonalující oblast se stále uplatňuje nejen v podnicích zabývajících se společenskou transformací, ale také v modernizaci kultur v zájmu rozvoje. [22]

Strojové učení pro ekologickou udržitelnost

Mezi aplikacemi strojového učení vyniká environmentální udržitelnost jako oblast, kde strojové učení vykazuje velkou použitelnost. Algoritmy strojového učení mají zásadní význam při simulaci a předvídání změn životního prostředí, což může být užitečné při formulování kvalifikovanějších rozhodnutí v oblasti ochrany životního prostředí. Sady algoritmů byly vytvořeny za účelem sledování velkých souborů dat, například údajů získaných ze satelitních snímků nebo senzorů, aby bylo možné sledovat změny v míře deforestace, množství vody a obecných vzorců znečištění v průběhu časového období. Předvídavost ML pomáhá nejen při včasném vytváření systémů informovanosti, ale také při usměrňování využívání přírodních zdrojů s cílem dosáhnout cílů udržitelného rozvoje. [22]

Předpověď délky života pomocí socioekonomických údajů

Prediktivní schopnost strojového učení se projevuje také v oblasti veřejného zdraví a demografie, kde model ML může předvídat zdravotní výsledky jednotlivců a obcí jako ukazatel socioekonomického typu dat. Operace neuronových sítí jako analýza velkých objemů demografických dat, kde se ML klasifikace uplatňují k významné identifikaci rozdílů v délce života spojených se socioekonomickými faktory. Proto aplikace ML při identifikaci a spojování rizikových faktorů chronických onemocnění s lidským chováním, chudobou, znečištěním ovzduší a nezdravou stravou podněcuje rozhodovací orgány k vývoji a realizaci programů kontroly nemocí, které cíleně zaměřují zdroje a intervence s péčí vedoucí ke zlepšení veřejného zdraví a prodloužení délky života. [22]

Strojové učení ve zdravotnictví

Zdravotnictví je další zásadní oblastí, kde se ML výrazně rozvíjí. Algoritmy ML se využívají v diagnostických postupech, při tvorbě léčebných plánů a v systémech monitorování pacientů. Tyto algoritmy mohou analyzovat lékařské snímky, genetické informace a záznamy o pacientech a pomáhat tak při včasné diagnostice a personalizované medicíně, čímž zlepšují výsledky pacientů a efektivitu zdravotní péče. [22] [30]

Strojové učení pro inteligentní dopravní sítě založené na IoT

ML algoritmy hrají nezbytnou funkci v procesu navrhování inteligentního dopravního systému v různých kontextech rozvoje inteligentních měst a urbanizace. Strojové učení je velmi užitečné při zpracování a analýze nezpracovaných dat z různých senzorů zabudovaných do městské

infrastruktury a souvisí s řízením dopravy v reálném čase, prediktivní údržbou a zlepšením veřejné bezpečnosti. Funkce ML v oblasti maximalizace dopravního provozu a minimalizace dopravních omezení může podporovat rychlejší městskou mobilitu i přispívat ke snižování znečištění ovzduší, které je hlavním nebezpečím, kterému města hrozí. [22]

Předpověď podmínek vzniku plynových hydrátů

Novou aplikací strojového učení pro energetiku je předpověď podmínek vzniku hydrátů plynu. ML modely jsou strojově vyškoleny pro předpovídání zón, kde se pravděpodobně vytvoří hydráty plynu, což zase může být nebo je výhodné pro procesy těžby energie a nepřirozené hrozby. Tato případová studie demonstruje širokou škálu aplikací ML, které se zabývají komplikovanými geologickými a geochemickými informacemi a nabízejí užitečné poznatky pro řízení přírodních zdrojů. [22]

Genetické programování a strojové učení

Stejně tak vývoj genetického programování se strojovým učením pro zvýšení rychlosti a flexibility algoritmů. Taková metodika předpokládá vývoj algoritmů s vnuceným výběrovým mechanismem podobným nekonečné evoluci a dává algoritmu šanci využít vlastní efektivitu. Takový přístup je užitečný zejména tehdy, když standardizované ML modely nemají dostatečné možnosti vyladění nebo působí při řešení složitých problémů. [22]

3.2.2 Hluboké učení

Jelikož hluboké učení obsazuje v technologiích umělé inteligence nejvýznamnější pozici, jednou z oblastí, kde má obrovský dopad, je skenování dokumentů. Tato část se zaměřuje na vývoj a působení hlubokého učení pro extrakci textu ze souborů PDF, což je jedna z klíčových funkcí pro zpřístupnění dat a chytrá podnikatelská řešení.

Hluboké učení v umělé inteligenci

Hluboké učení lze považovat za hlavní součást současné umělé inteligence, která poskytuje nezbytné prostředky pro konstrukci inteligentních systémů, jež lze použít při řešení různých problémů. Příklady jako IBM a Google AI zdůrazňují způsob, jakým je AI schopna napodobit procesy lidské inteligence při provádění hlubokého učení. Díky využití umělých neuronových sítí, které mají schopnost učit se z velkých souborů dat, mohou algoritmy hlubokého učení mimořádně dobře fungovat v oblastech, jako je rozpoznávání obrazu a řeči, zpracování přirozeného jazyka a skládání prognóz. Tato technologie, jak se jí zabývá Pendyala, má zásadní význam pro kulturní rozvoj a moderní svět a pomáhá zlepšovat lékařskou péči, finanční služby a ochranu životního prostředí. Prostřednictvím rozšiřování hranic funkčnosti umělé inteligence

vytváří hluboké učení lepší budoucnost pro lidstvo, přičemž inteligentní systémy ovlivňují výkonnost globálních problémů. [22]

Hluboké učení pro extrakci informací z digitálních dokumentů

Při IE, tedy získávání potřebných informací z digitálních dokumentů, konkrétně z PDF, je jedním z nejzásadnějších problémů doslovné pevné a nelineární nastavení formátu. Problém spočívá v tom, že běžné přístupy k extrakci textu nedokážou vyřešit složitost a rozmanitost obsahu, který nemá žádnou standardní strukturu, jako jsou tabulky, grafy a vícesloupcové texty. Algoritmy hlubokého učení to berou jako velkou výzvu, protože využívají modely, jako jsou CNN a RNN, ke zkoumání vnitřní strukturu typu dokumentu, bez nutnosti spoléhat se na náročné stanovení pravidel. Modely hlubokého učení jsou vyškoleny na velkých souborech dat digitálních dokumentů k rozpoznání různorodé rozvržení a textovou reprezentaci. Tréninkové závazky mají stroji poskytnout schopnost naučit se veškeré informace, které nový dokument nabízí, a snadno přitom extrahovat požadované informace s vysokou přesností. Dalším technologickým pokrokem, který pomáhá zlepšit kredibilitu blockchainů, je NLP, které je již podporováno strojovým učením a hlubokým učením, které do textu vloží další kontext nad rámec sémantického významu. [20] [22]

Použití a dopad

Obrovská škála a mnohostrannost hlubokého učení při zjišťování informací v digitálních dokumentech se ukázala být neomezená na oblast použití. V oblasti podnikání tyto technologie umožňují podnikům automatizovat zpracování obchodních dokumentů, jako jsou faktury, objednávky a zprávy pro zákazníky, což ve výsledku znamená zlepšení reakčních časů a úsporu provozních nákladů. Ve výzkumu a akademické oblasti pomáhá hluboké učení extrahovat informace z vědeckých článků a technických postů, a tak jsou recenze literatury a syntéza dat dokončeny velmi rychle. Vliv, který mají uvedené technologie na lidi na sociální úrovni, je také ohromující. Hluboké učení přináší data a analýzy, které automatizují proces získávání informací. To pomáhá překlenout digitální rozdíly, poskytuje více osobám přístup k informacím a podporuje tak rozhodování založené na datech při tvorbě politických rozhodnutí. [22]

Výzvy a budoucí směry

Přestože má hluboké učení pro extrakci informací z dokumentů své výhody, naráží na některé potíže. Takové modely jsou velmi závislé na množství a kvalitě vstupních dat použitých pro školení. Navíc jsou tyto modely výpočetně náročné, protože potřebují velké množství hardwaru a energie, což může být problematické pro malé organizace a malé rozvojové země. Budoucí vědecký výzkum se zaměřuje na vylepšení schopností a náročnosti těchto algoritmů na zdroje. Kromě toho se některé výzkumné aktivity zaměřují na zlepšení odolnosti modelů vůči změnám

v preferencích formátování dokumentů a vůči vyvíjejícím se předpisům o ochraně údajů, které přináší rostoucí potřeba dodržovat přísnější předpisy o ochraně osobních údajů. [22]

3.3 Aplikace umělé inteligence v analýze dat

Využití sofistikovaných technik umělé inteligence, rozpoznávání obrazu a prediktivní analýza, zvýšuje objem zpracovávaných dat, stejně jako analýzu a generování náhledů.

Komunikace mezi člověkem a počítačem je významným aspektem umělé inteligence a zahrnuje výuku strojů k provádění obrovské množství práce s jazykovými daty z reálného života. Úkolem NLP je podporovat porozumění lidským jazykům s cílem vytvářet význam a užitečnost pro konkrétní účely, jako je strojové učení a algoritmy AI. Aplikace techniky NLP vede například k vývoji systémů provádějících více úkolů, protože provádějí funkce související s analýzou sentimentu, extrakcí témat a automatickou sumarizací textu, což usnadňuje vhled do textových dat v nesrovnatelném měřítku. V kontextu analýzy dat lze NLP použít k automatizaci získávání informací z textových zdrojů dat, včetně e-mailů, příspěvků na sociálních sítích a dokumentů. Pokročilé algoritmy umožňují v těchto datech identifikovat vzory a trendy, které lze využít v rozhodovacích procesech v různých oblastech, jako je marketing, zdravotnictví a veřejná správa. Aplikace NLP nejen zefektivňují organizační procesy, ale také zvyšují schopnost zvládat společenské výzvy prostřednictvím znalostí založených na datech. Rychle se rozvíjející technologie zpracování přirozeného jazyka navíc přináší mnoho nových produktů a technik. Patří mezi ně modely NLP od OpenAI, které jsou schopny porozumět a vytvářet texty, které jsou téměř ekvivalentní těm, které je schopen napsat člověk. Změna v NLP tedy otevírá cestu k dozrávání nástrojů pro analýzu dat založených na umělé inteligenci s intuitivnějšími rozhraními. To umožňuje širšímu spektru uživatelů podílet se na datově řízeném rozhodování i bez hlubších technických znalostí. [22] [23]

Technologie vizuálního rozpoznávání založené na umělé proměňují a automatizují používání obrázků při interpretaci dat. Vydaný software má schopnost určovat a kategorizovat ty položky, které se nacházejí na obrázcích, s cílem automatizovat a následně zvýšit zpracování vizuálních dat. [24] Aplikace rozpoznávání obrazu se liší v lékařské diagnostice, kde algoritmy umělé inteligence podporují lékaře při zjišťování nemocí pomocí lékařských snímků, a v digitální bezpečnosti, která vytváří systém dohledu pomocí rozpoznávání obličejů. Rychlost a přesnost zpracování AI u obrazů, která je na vysoké úrovni, představuje významné zlepšení stávajících metod manuální analýzy obrazu s potenciálem lepšího a většího rozsahu. [22]

Prediktivní analýzu lze definovat jako použití technik umělé inteligence k předvídání budoucích událostí vyplývajících z informací z minulosti. Díky nasazení algoritmů strojového učení může

mít prediktivní analytika schopnost přesně určit věci, které analytičtí výzkumníci nemusí na první pohled odhalit. Taková identifikace trendů má zvláštní význam v oblastech financí, kde se tato technologie využívá pro předpovídání vývoje akcií, a v maloobchodě, kde předpovídá chování spotřebitelů. Primární schopností prediktivní analytiky je poskytovat využitelné poznatky, které mají pomoci při přijímání informovaných rozhodnutí prostřednictvím procesu strategického plánování. [20] [22]

3.4 Integrace umělé inteligence do webových aplikací

Integrace umělé inteligence do webových aplikací představuje obrovský posun ve vývoji digitálních technologií. Umělá inteligence se zapojuje do automatizace úkolů a do poskytování personalizovaných uživatelských zkušeností. Tato integrace se opírá především o využívání rozhraní API, která jsou vytvořena pro výpočty AI, používání různých frameworků a knihoven a také o implementaci silného zabezpečení a ochrany osobních údajů.

Význam rozhraní API je v procesu integrace technologií umělé inteligence s webovými aplikacemi velmi významný. Tímto způsobem mají webové aplikace přístup k účinným funkcím umělé inteligence, na nichž jsou založeny i vzdálené servery, bez potřeby rozsáhlých lokálních výpočetních zdrojů na straně klienta, což umožňuje hluboký rozvoj procesů umělé inteligence. Google mimo jiné nabízí platformu AI s možnostmi rozhraní API pro získávání a zpracování dat od přirozeného jazyka až po rozpoznávání obrazu. [19]

OpenAI navíc poskytuje dobře zdokumentovaná rozhraní API, která pomáhají vývojářům vytvářet různé jazykové a obrazové modely ve webových aplikacích, a tím zlepšovat podstatu aplikací. [23] [24] [25]

Vytváření a integrace umělé inteligence do webových aplikací je možné především zásluhou vzájemně propojených frameworků a knihoven, které tyto úkoly usnadňují. Tyto nástroje sestavují virtuální podpůrné mechanismy, které dále umožňují plynulou integraci AI v rámci jednotlivých produktů. Jednou z populárních knihoven, která umožňuje spouštění modelů strojového učení z prohlížeče, je TensorFlow.js, s jehož pomocí lze provádět implementaci AI bez závislosti na zpětných serverech. Kromě toho se poslední inovace společnosti Google v podobě rozhraní Gemini API ukázala jako značný přínos, který umožňuje vývojářům používat optimalizované modely strojového učení, které lze intenzivně ladit podle úloh, které weboví vývojáři vyžadují, což usnadňuje proces integrace. [28]

Technologické nástroje jako PyTorch a Keras se významně rozvinuly a dnes se tyto užitečné nástroje učí implementovat modely umělé inteligence s vlastní rychlostí, škálovatelností a

flexibilitou tisíce webových vývojářů. Při výběru frameworku obvykle záleží na typu samotné aplikace, míře vyžadovaných funkcí a požadovaném uživatelském dojmu. [29]

S rostoucí integrací technologií umělé inteligence do struktury webu, konkrétně do webových aplikací, se zvyšují i rizika pro bezpečnost a ochranu osobních údajů. Integrace umělé inteligence do systémů nabízí rapidní nárůst provozního výkonu, ale zároveň přináší nové cesty k narušení dat a etické pochybnosti tím, že vyžaduje složité bezpečnostní přístupy. V souvislosti s IBM je třeba říci, že AI používaná ve webových aplikacích by se měla vyznačovat bezpečností takových dat, která se používají v modelech AI. [18]

Pro zvýšení anonymity je možné zavést přístupy, jako je federativní učení. Algoritmy spuštěné prostřednictvím nasazených zařízení nebo serverů tak nebudou muset sdílet soukromé zdroje dat. Navíc je nutné etické využívání umělé inteligence vnímat vážně a zdůraznit, že nejdůležitějšími zásadami při zavádění umělé inteligence jsou transparentnost, spravedlnost a odpovědnost. Spočívá v zajištění jistoty, že systémy AI nebudou přispívat k neobjektivitě a zneužívání dat, k čemuž může přispět podrobné testování a zavedení etických standardů již v rané fázi vývoje. [20]

4. SaaS aplikace

Nejvýznamnějšími inovacemi v oblasti správy dat je rozkvět softwaru jako služby (SaaS). Jedná se o snadno dostupné a škálovatelné řešení. Tato kapitola se zabývá aplikací SaaS pro zpracování dat speciálně vytvořenou z PDF souborů s využitím principu vývoje aplikace SaaS.

4.1. Seznámení s prostředím SaaS

Pro sestavení základu je nesmírně důležité pochopit paradigma SaaS jako součást celého světa cloud computingu.

SaaS je přístupem k poskytování cloudových služeb, kdy je aplikace spuštěna na serveru poskytovatele a uživatelé k nim přistupují prostřednictvím internetu. Tím odpadá nutnost instalace a údržby na lokálních místech, což pomáhá šetřit náklady, umožňuje široké možnosti rozvoje a kvalitnější přístup k využívání služeb pro všechny. [26]

4.2. Základní úvahy o webové aplikaci SaaS

Navazující aplikace má za cíl prozkoumat základní principy vývoje aplikace SaaS pro zpracování souborů PDF. Vzhledem k povaze praktického výstupu a omezením spojeným s využíváním API, jako je OpenAI API, které nejsou bezplatné, je současný stav aplikace zaměřen na ověření konceptu a funkčnosti.

Vývoj úspěšné webové aplikace SaaS vyžaduje přístup orientovaný na uživatele, který klade hlavní důraz na funkčnost, použitelnost a bezpečnost:

- Intuitivní uživatelské rozhraní a uživatelský zážitek: Přijetí a spokojenost uživatelů do značné míry závisí na přehlednosti a intuitivnosti rozhraní. Program musí umožňovat rychlý postup uživatelů při práci se soubory PDF nevyžadující velkou pozornost.
- Robustní schopnosti extrakce dat: Získání by mělo uplatňovat pokročilé techniky extrakce dat, k přesné extrakci textu, tabulek a dalších relevantních datových bloků v různých formátech PDF.
- Možnosti transformace a výstupu dat: Extrahovaná data musí být v takové podobě, kterou lze transformovat do různých formátů, aby pomohla uživatelům s různými potřebami a také integrovat s jinými systémy.
- Bezpečnost a ochrana osobních údajů: Pro ochranu citlivých údajů a dodržení předpisů o ochraně údajů je nutné zavést účinná bezpečnostní opatření, včetně šifrování dat a bezpečných protokolů pro ověřování uživatelů.

- Výkon a spolehlivost: Platforma poskytuje spolehlivý výkon a dobrý rychlostní výkon nepřetržitým provozem s malými výpadky.

4.3. Potenciál pro budoucí vývoj a komercializaci

I když je omezen svým čistě akademickým zaměřením, existuje značný prostor pro budoucí rozvoj do podoby plnohodnotné aplikace SaaS. Po vyřešení problémů s licencemi API a dalším vývoji funkcí může tato aplikace být uvedena na trh a konkurovat na rostoucím trhu nástrojů pro zpracování dat SaaS.

4.4. Zaměření na potřeby trhu a sladění s trendy

Návrh odpovídá rychle rostoucí poptávce po cloudových řešeních pro vytěžování dat a tuto skutečnost lze přičíst rostoucí závislosti na elektronických dokumentech v několika odvětvích. Tendence specializovaných aplikací SaaS poukazuje na to, že se nabízejí řešení specifická pro konkrétní odvětví. [27]

5. Aplikační řešení

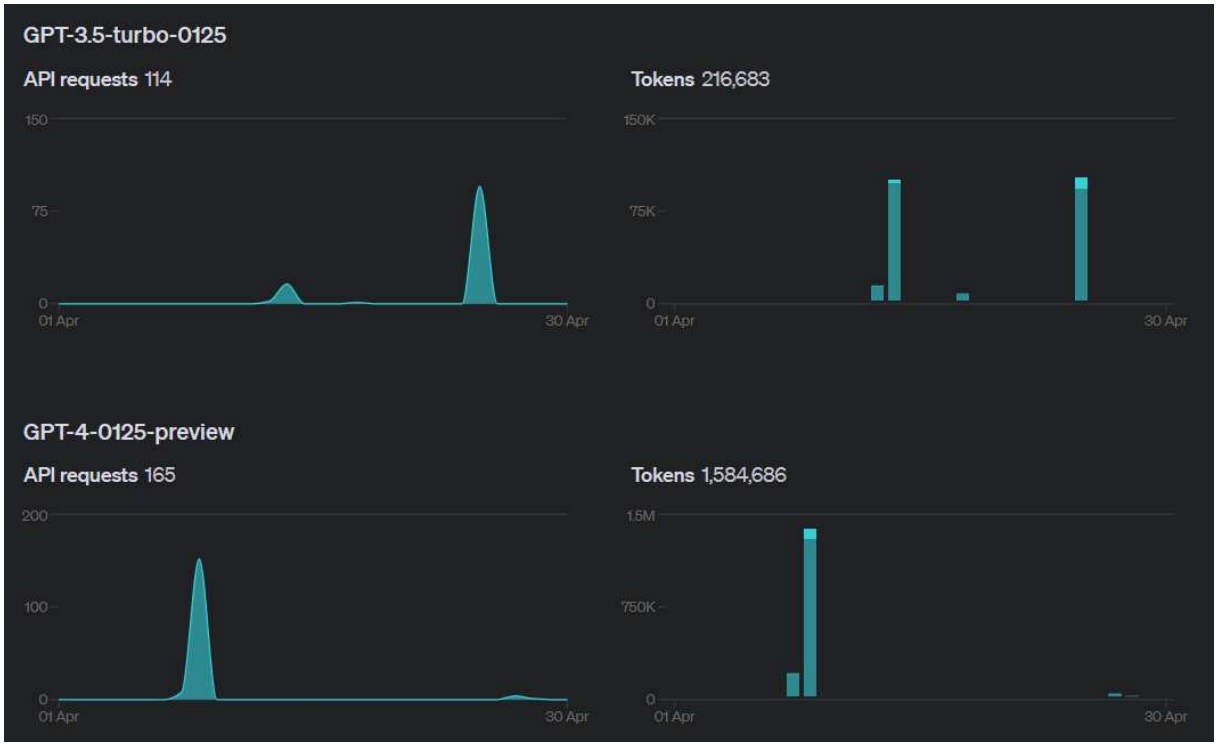
Tato kapitola popisuje návrh vyvíjené aplikace Juggle. Nejprve jsou popsány vybrané technologie použité k vývoji a testování aplikace. Návrh aplikace vychází z funkčních a nefunkčních požadavků pro analýzu PDF dokumentů. Rovněž je popsána klíčová logika aplikace, a to uvedením kódu použitého v aplikaci.

5.1. Technologie použité při vývoji

Kapitola se soustředí na vývojové nástroje, prostředí, webový server a REST API.

5.1.1. OpenAI API

Při vývoji aplikace Juggle je použita umělá inteligence z OpenAI API. Konkrétně, základem funkčnosti aplikace Juggle je model OpenAI GPT, především GPT-4-0125-preview a GPT-3.5-turbo-0125. OpenAI tyto modely představuje prostřednictvím svých API, čímž lze jednoduše integrovat možnosti zpracování přirozeného jazyka do aplikací Model připravuje prostředí pro složitější textové operace a úlohy generování, počet použitých tokenů je 1 584 686, ať už se jedná o model GPT-3.5-turbo-0125 s počtem tokenů rovným 216 683 v období vývoje aplikace. Je známo, že GPT-4 se vyznačuje vyšší přesností než jeho předchůdci a schopností poskytovat přesnější odpovědi na složité otázky než předchozí generace. Tyto údaje jsou čerpány v době psaní této práce. [30] [31]



Obrázek 2: Statistika použití OpenAI modelů (zdroj: vlastní)

Integrace aplikace Juggle s těmito modely je provedena pomocí strukturovaného požadavku API. To je znázorněno v níže přiloženém kódu JavaScriptu, který ilustruje, jak aplikace využívá Axios k zaslání požadavků do rozhraní OpenAI API:

```
const response = await
axios.post('https://api.openai.com/v1/chat/completions', {
  model: "gpt-4-0125-preview", messages: messages, temperature: 0.7,
max_tokens: 500,
}, {
  headers: {
    'Content-Type': 'application/json', 'Authorization': `Bearer
${process.env.OPENAI_API_KEY}`
  }
});
```

Zdrojový kód 1: server.js - Koncový bod „/api/analyze-pdf-firebase“ (zdroj: vlastní)

5.1.2. Postman

Aplikace Postman vznikla jako rozšíření prohlížeče pro systémy Windows, OS X a Linux a slouží k vytváření a údržbě rozhraní API JSON, XML, SOAP či REST API. Mezi jeho funkce patří rozhraní API pro automatizaci testování a požadavků API. Tento nástroj je vybaven jednoduchým a snadno použitelným rozhraním příkazového řádku, kde lze vybrat požadavek HTTP, zadat URL adresu webové aplikace a potřebné parametry. Ve dolní části obrazovky aplikace se nachází také odpověď serveru doplněná příslušným stavovým kódem HTTP. [32]

5.1.3. Webstrom

Pro vývoj aplikace je jako hlavní kódovací nástroj zvoleno vývojové prostředí WebStorm IDE od společnosti JetBrains. WebStorm je přesně navržen pro JavaScript a technologie, které jej pokrývají, bez ohledu na moderní technologie vývoje webových stránek, které podporuje, včetně HTML5, Node.js, CSS, Sass, React, Angular a Vue.Js Jeho schopnosti se značně liší tím, že obsahuje funkci doplňování kódu, která zajišťuje bezproblémovou a bezchybnou navigaci, a možnosti refaktoringu, které výrazně zvyšují produktivitu a kvalitu kódu. Další výhodou používání Web Stormu je, že se bezproblémově integruje s populárním systémem pro správu verzí Git, což značně usnadňuje správu verzí. [33]

5.1.4. Firebase

Firebase se používá k vytvoření webové aplikace, komplexní platformu Backend-as-a-Service, kterou nabízí společnost Google. Firebase díky své široké škále funkcí zjednodušuje proces vývoje, takže je pro tuto aplikaci vhodnou volbou. Vzhledem k tomu, že aplikace ve velké míře zahrnuje práci se soubory PDF. Integrované řešení pro ověřování uživatelů, správu úložišť a

databází umožňuje časovou úsporu. Snadná integrace Firebase s různými frontendovými technologiemi a její velkorysá bezplatná varianta jsou důvodem v rozhodnutí použít ji pro účely výstupní aplikace. [34]

- Autentizace: Služba Firebase authentication se používá k implementaci bezpečné registrace uživatelů, přihlašování a správy relací. Tato funkce je nezbytná pro zajištění integrity a důvěrnosti uživatelských dat.
- Úložiště: Pro ukládání souborů PDF nahraných uživateli je využita služba Firebase Storage. Škálovatelné a spolehlivé úložiště souborů této služby zjednoduší správu potenciálně velkých dokumentů PDF.
- Databáze Firestore: K uložení metadat spojených s nahranými soubory PDF a dalších potřebných dat aplikace je používána databáze Firebase Firestore Database. Flexibilní struktura databáze Firestore byla zvolena i z důvodu usnadnění modelování vztahů mezi těmito entitami. [34]

5.2. Požadavky

5.2.1. Funkční požadavky

Tabulka 2: Funkční požadavky (zdroj: vlastní)

F1	Aplikace musí mít integrovanou funkcionalitu pro ověření identity pomocí emailu.
F2	Aplikace musí umožňovat změnu hesla.
F3	Aplikace by měla umožnit při načítání spojit více souborů PDF do jednoho.
F4	Aplikace umožňuje uživatelům stáhnout nebo zobrazit výsledek analýzy.
F5	Načtení, permanentní uložení a prezentace předdefinovaných podmínek a vzorů pro každý soubor PDF
F6	Pro analyzování obsahu PDF by měl být používán koncový bod rozhraní OpenAI.
F7	Načtení, permanentní uložení a prezentace souborů PDF vztahujících se k uživateli.
F8	Aplikace musí informovat uživatele o stavu analýzy.
F9	Aplikace musí zobrazovat obsah PDF souborů.

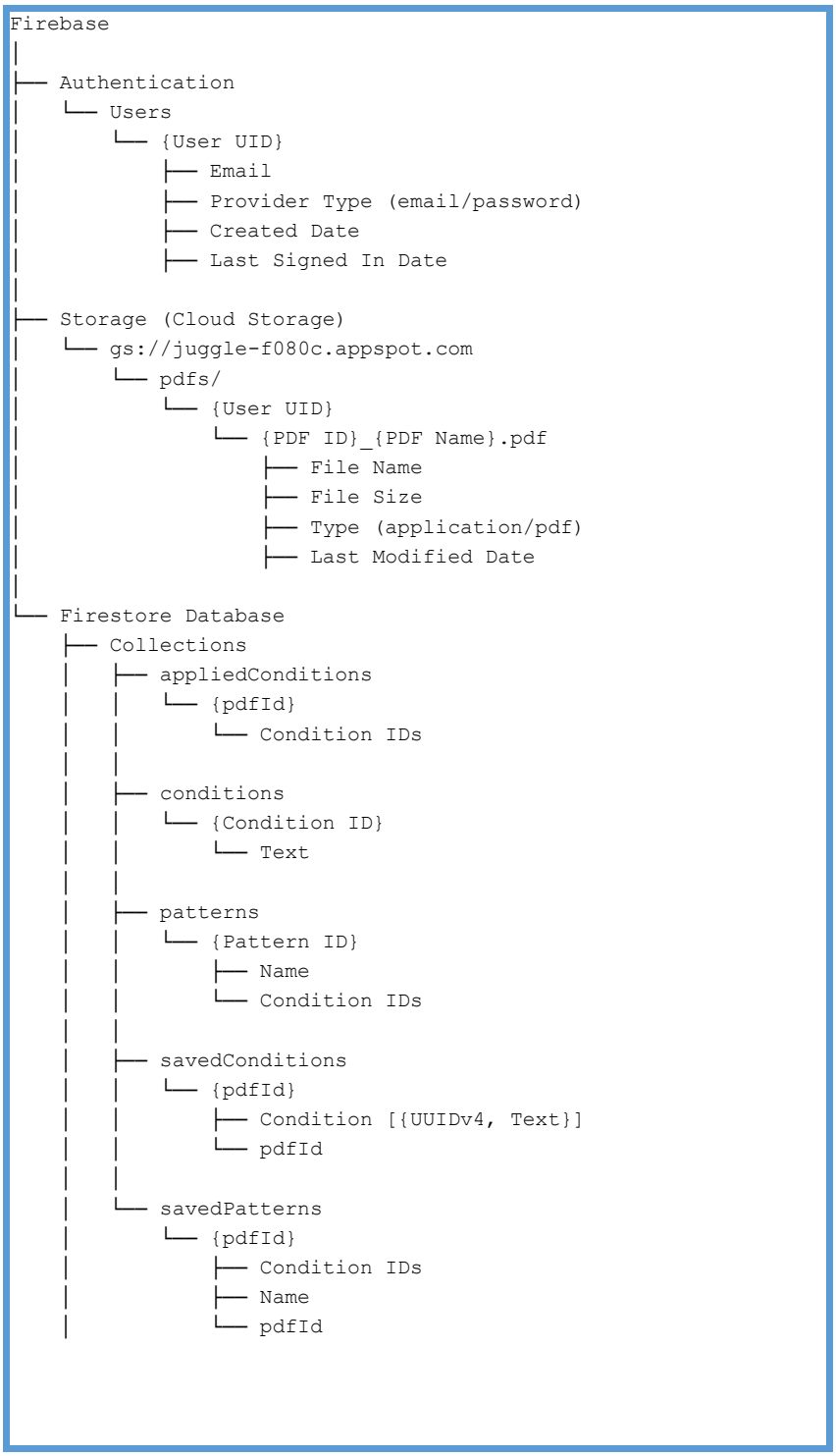
4.2.2 Nefunkční požadavky

Tabulka 3: Nefunkční požadavky (zdroj: vlastní)

NF1	Aplikace bude naprogramována v jazyce JavaScript
NF2	Aplikace musí být responzivní a uživatelsky přívětivá
NF3	K zobrazení obsahu PDF a interakci bude použita externí knihovna ng2-pdf-viewer
NF4	Uživatel musí mít přístup pouze ke svým údajům
NF5	Kompatibilita a optimální integrace použitých frameworků a knihoven
NF6	Implementace důkladného ošetření chyb, které uživatelům poskytuje jasné a informativní chybové zprávy
NF7	Šifrování citlivých dat při přenosu i v klidovém stavu
NF8	Aplikace musí být funkční v nejnovějších verzích hlavních webových prohlížečů

5.3.Návrh databáze

Firebase obsahuje řadu pokročilých funkcí, díky kterým je vývoj aplikací mnohem produktivnější než kdykoli jindy. Za důkaz toho, jak si Firebase vede při vytváření uživatelsky orientované aplikace, se považují následující skutečnosti. Cílem je proniknout do ověřování Firebase, cloudového úložiště a databáze Firestore pro správu uživatelských dat, ukládání souborů a relačních datových struktur. Strukturní návrh databáze Firebase lze popsat pomocí stromového zobrazení, které vyznačuje umístění dat v jednotlivých službách:



Zdrojový kód 2: Stromové zobrazení databáze (zdroj: vlastní)

5.3.1. Firebase Authentication

Pro správu identit uživatelů se používá autentifikace Firebase, kde je každý uživatel identifikován e-mailovou adresou, což dokumentuje podrobné zobrazení ve stromovém přehledu v rámci uzlu „Authentication“. Zdrojový kód 2. uvádí, že systém ověřování Firebase Authentication ukládá e-mailovou adresu uživatele, e-mail a heslo jako typ poskytovatele

ověření, datum vytvoření účtu a datum jeho posledního přihlášení. Každému uživateli je také přiřazeno unikátní ID uživatele.

5.3.2. Firebase Cloud Sotrage

Cloudové úložiště Firebase slouží k ukládání souborů, jak ukazuje uzel „Storage (Cloud Storage)“ ve stromovém zobrazení ve zdrojovém kódu 2. Úložiště je vytvořeno tak, že musí obsahovat cestu gs://juggle-f080c.appspot.com/pdfs/. V tomto adresáři jsou složky pojmenovány podle UID každého uživatele, což zajišťuje, že uživatelská data jsou členěna a zabezpečena. Soubory jsou uloženy s podrobnými metadaty, včetně názvu souboru, velikosti souboru, typu "application/pdf" a data poslední změny každého souboru, což znamená datum jeho nahrání.

V případě úložiště Firebase Storage pravidla zajišťují, že uživatelé mohou přistupovat pouze k souborům ve složce s konkrétním názvem UID, takže přístupová práva jsou přímo přizpůsobena ověřování uživatele:

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /pdfs/{userId}/{allPaths=**} {
      allow read, write: if request.auth != null && request.auth.uid ==
      userId;
    }
  }
}
```

Zdrojový kód 3: Firebase Storage pravidlo (zdroj: vlastní)

5.3.3. Firebase Firestore Database

Databáze Firestore je v aplikaci strukturována do pěti kolekcí: appliedConditions, conditions, patterns, savedConditions a savedPatterns. Tato struktura je označena v uzlu „Firestore Database“ dříve uvedeného stromového zobrazení ve zdrojovém kódu 2. Tyto kolekce jsou vzájemně propojeny.

- kolekce *appliedConditions* uchovává odkazy na podmínky použité na konkrétní dokumenty, identifikované pomocí polí jako *uid*, *conditionIds* a *pdfId*.
- kolekce *conditions* obsahuje neměnné globální podmínky s poli pro *uid* a *text*.
- kolekce *patterns* obsahuje šablony, které obsahují množinu podmínek definovaných pomocí *uid*, *conditionIds* a *name*.

- *savedPatterns* lze odkazovat buď na globální podmínky, nebo na podmínky *savedConditions* vytvořené uživateli, propojené také pomocí *pdfId*.

U Firestore znemožňuje pravidlo provádět jakékoli operace přímo z přístupů na straně klienta, což se může zdát být omezující, ale je koncepcí vynucování validace a provádění operací na serverové straně za účelem zachování integrity a zabezpečení dat.

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if false;
    }
  }
}
```

Zdrojový kód 4: Firestore Database pravidlo (zdroj: vlastní)

5.4. Implementace aplikace

Tato kapitola pojednává o nástrojích používaných k vývoji a na příkladech vysvětluje, jak řešit konkrétní problémy.

Pro vývoj webové aplikace je zvolen framework Angular jako frontendový framework JavaScriptu a framework Express.js jako backendový framework JavaScriptu, který se využívá pro implementaci serverové části aplikace Juggle. Důvodem volby těchto dvou frameworků je možnost kompletní implementace řešení popsaných požadavků.

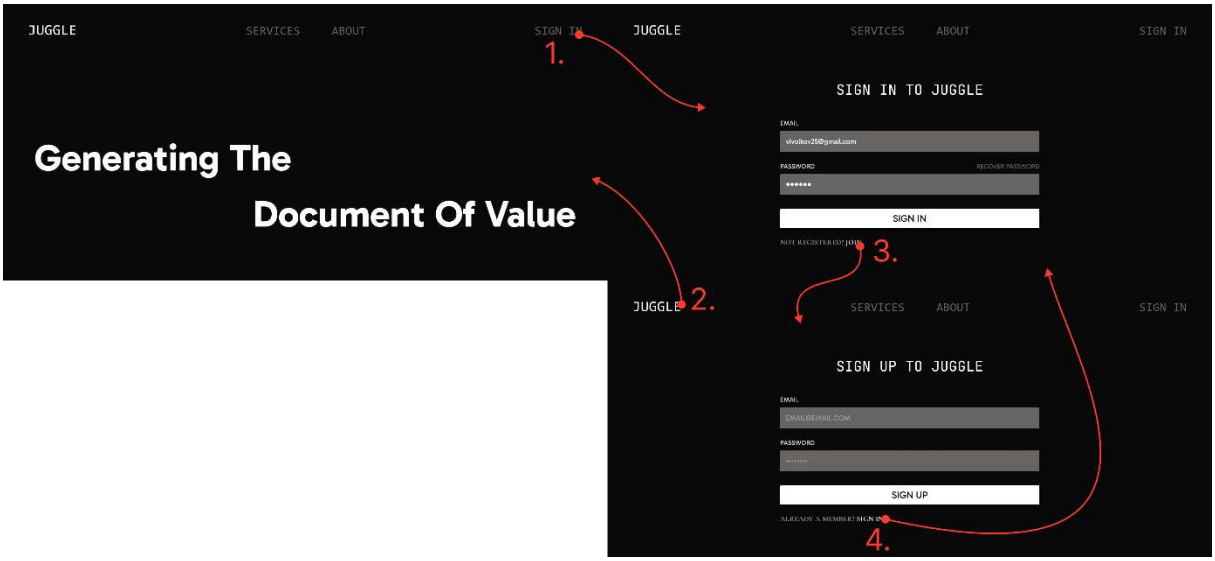
5.4.1. Grafické uživatelské rozhraní

Při vývoji aplikací je velmi významné grafické uživatelské rozhraní, jelikož představuje významný faktor hodnocení úspěšnosti. Jestliže aplikace nevypadá graficky atraktivně, pak ji nebudou uživatelé tak snadno a rádi běžně používat. Dodržení tohoto požadavku je usnadněno použitím TailwindCSS, frameworku CSS zaměřeného na užitečnost, a nastudování aktuálních trendů v oblasti webového designu. Kompletní grafické vykreslení aplikace je poskytnuto v Příloze A.

5.4.2. Navigace v aplikaci

Navigace je realizována vnitřními nástroji Figma. Vizuální přechody mezi stránkami jsou znázorněny červenými čarami. Přechody se provádějí v závislosti na zařízení pomocí tlačítek a kliknutím na obrazovku. Oba typy přechodů jsou úspěšně otestovány.

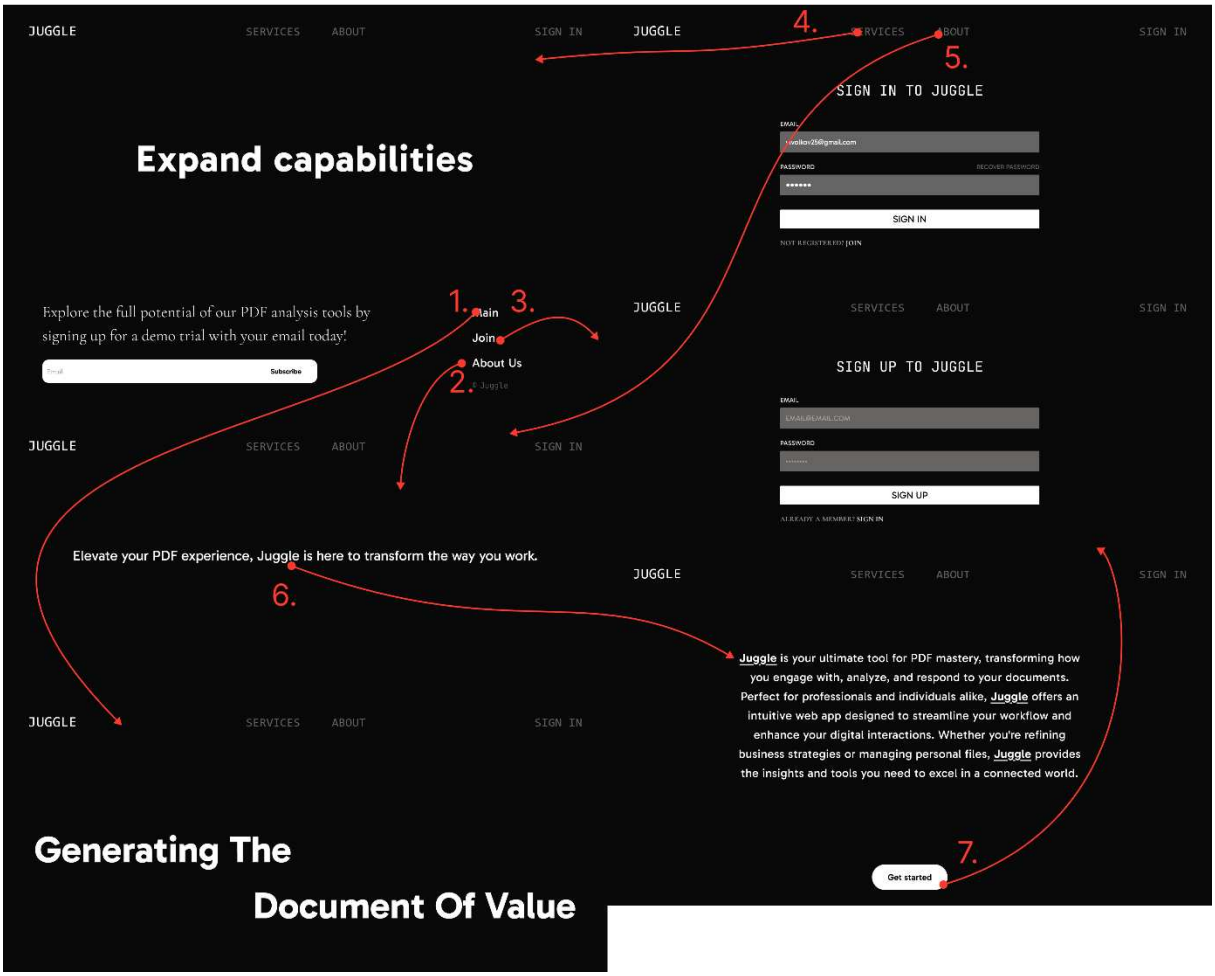
Obrázek 3. znázorňuje navigaci neoprávněného uživatele mezi vstupní, registrační a přihlašovací stránkou. Šipkou 1. je znázorněn přechod na přihlašovací stránku, pokud se chce uživatel vrátit na vstupní stránku, musí jednoduše kliknout na název projektu, jak ukazuje šipka 2. Pokud chce uživatel jednoduše přecházet mezi přihlašovací a registrační stránkou, musí pouze kliknout na odkazy na formulářích těchto dvou stránek, tuto vizuální logiku znázorňují šipky 3. a 4.



Obrázek 3: Navigace mezi vstupní, registrační a přihlašovací stránkou (zdroj: vlastní)

Obrázek 4. znázorňuje navigaci nepřihlášeného uživatele mezi všemi možnými stránkami, které jsou k dispozici uživatelům, kteří nejsou přihlášení.

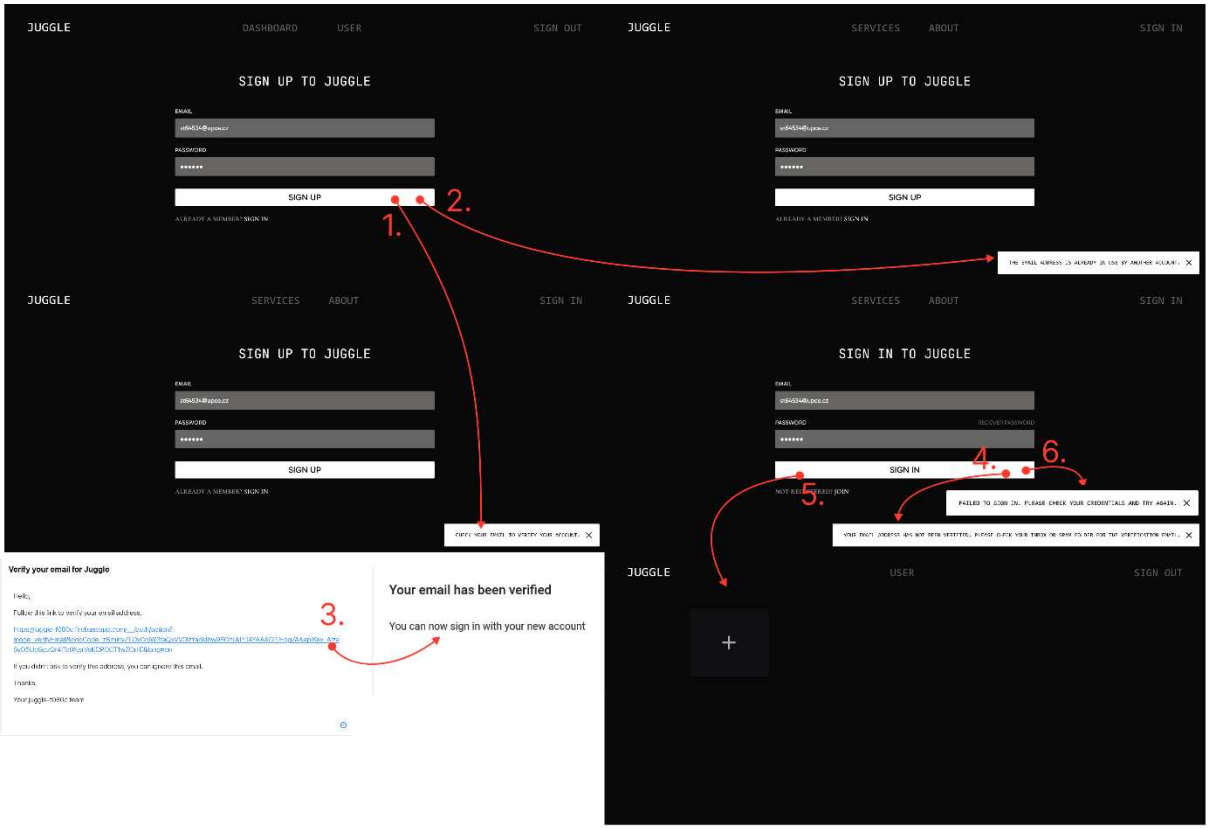
Kromě vstupní stránky je design komponenty footer k dispozici také na stránce Services, kde může uživatel přejít na vstupní stránku, šipka 1., přejít rovnou na registrační stránku, šipka 3., přejít na stránku About, šipka 2. Šipky 4. a 5. zobrazují logiku navigace na panelu pro neautorizovaného uživatele, která vede na stránky Services, resp. About. Na stránce About je zobrazena funkce zobrazení textu po kliknutí na něj, šipka 6. Po zobrazení textu může uživatel snadno přejít na registrační stránku, šipka 7.



Obrázek 4: Navigace mezi možnými stránkami pro neautorizovaného uživatele (zdroj: vlastní)

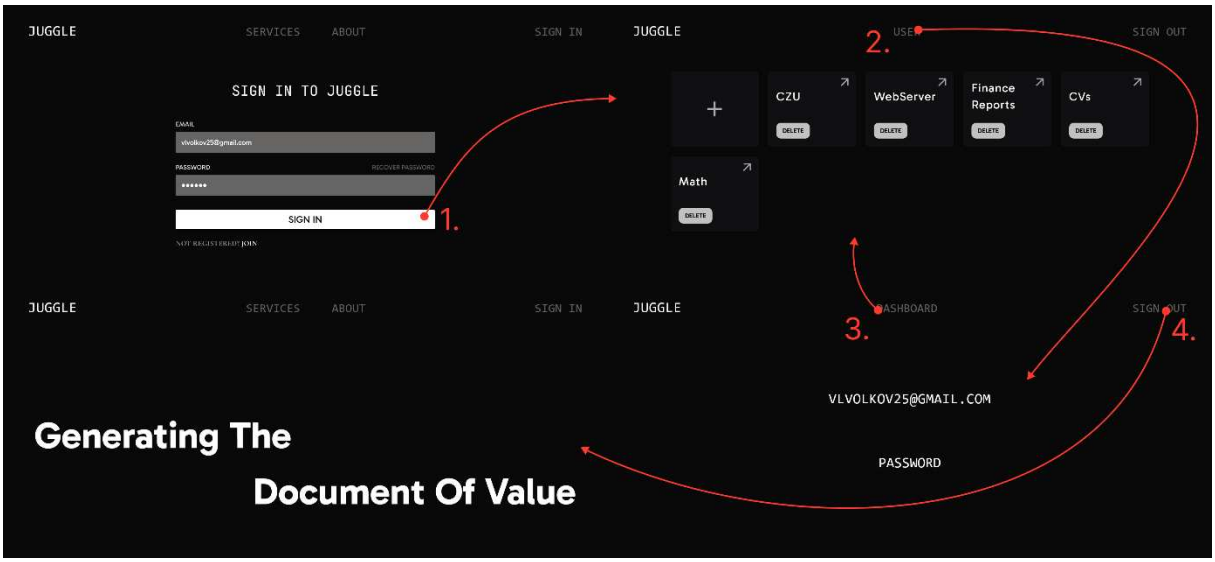
Obrázek 5. znázorňuje navigaci mezi stránkami přihlášení, registrace a ovládacího panelu, které jsou k dispozici při registraci a po přihlášení uživatele. Obrázek rovněž zobrazuje postup ověření identity a ošetření chyb při registraci a přihlášení na straně uživatele. Jakmile uživatel zadá své údaje, e-mail a heslo, zobrazí se zpráva pro potvrzení identity e-mailem, šipka 1. Pokud uživatel s uvedenou e-mailovou adresou již existuje, zobrazí se na registrační stránce odpovídající zpráva, šipka 2. Pro potvrzení identity musí uživatel kliknout na odkaz zaslaný v e-mailu. Po kliknutí na odkaz bude uživatel informován, že pokus o identifikaci je úspěšný, šipka 3. Jestliže uživatel nepotvrdí svoji identifikaci, zobrazí se mu při pokusu o přihlášení zpráva týkající se tohoto aspektu, šipka 4. Pokud uživatel zadá nesprávné údaje, zobrazí se také

zpráva o nesprávných údajích, šipka 6. Pokud jsou údaje správné a identita je potvrzena, uživatel bude přesměrován na stránku hlavního panelu, šipka 5.



Obrázek 5: Navigace mezi stránkami přihlášení, registrace a hlavního panelu (zdroj: vlastní)

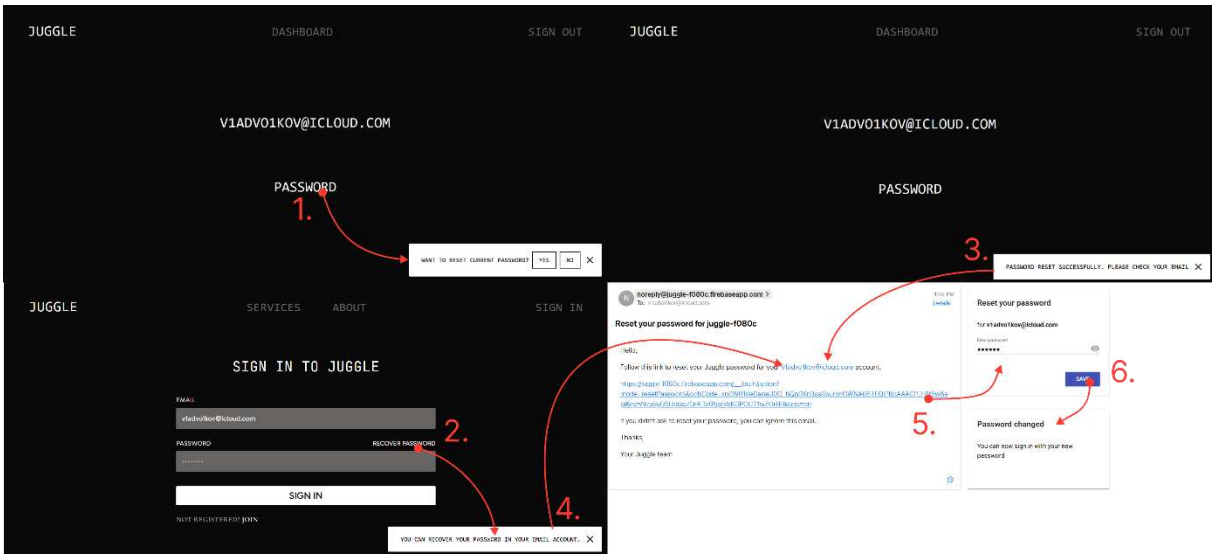
Obrázek 6. znázorňuje navigaci mezi úvodními stránkami, které jsou k dispozici ihned po autorizaci uživatele a vstupní stránku po odhlášení z účtu. Šipka 1. ukazuje přechod na stránku hlavního panelu po úspěšné autorizaci. Šipka 2. ukazuje možnost přechodu do uživatelského profilu, kde se po kliknutí na e-mail zkopíruje e-mailová adresa a po kliknutí na skryté heslo bude možné změnit heslo, funkce je zobrazena na obrázku 15. Šipka 3. zobrazuje přechod na stránku hlavního panelu. Šipka 4. zobrazuje navigaci po odhlášení z účtu.



Obrázek 6: Navigace mezi stránkami po autorizaci uživatele a vstupní stránkou (zdroj: vlastní)

Obrázek 7. znázorňuje navigaci mezi stránkami přihlášení a profilu.

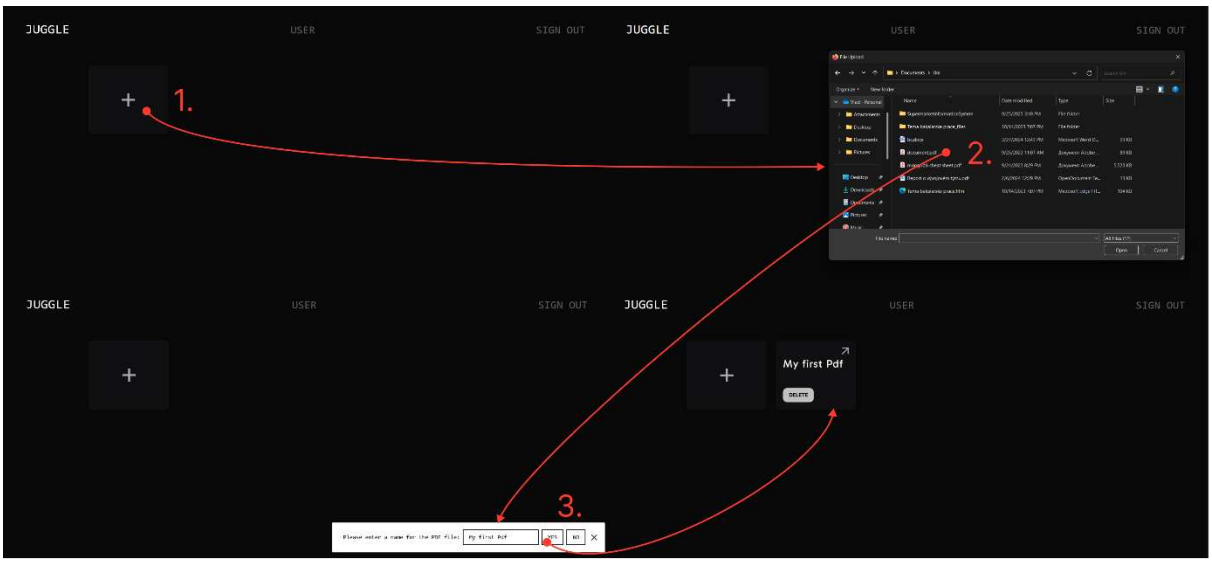
V profilu uživatele po kliknutí na skryté heslo bude možné změnit heslo, šipka 1. Lze to provést také na přihlašovací stránce, šipka 2. Po odsouhlasení obnovení hesla se zobrazí zpráva, že je nutné tuto funkci provést kliknutím na odkaz v zaslaném e-mailu, šipky 3. a 4. Šipky 5. a 6. ukazují navigační logiku pro resetování hesla vně aplikace Juggle.



Obrázek 7: Navigace mezi stránkami pro resetování hesla (zdroj: vlastní)

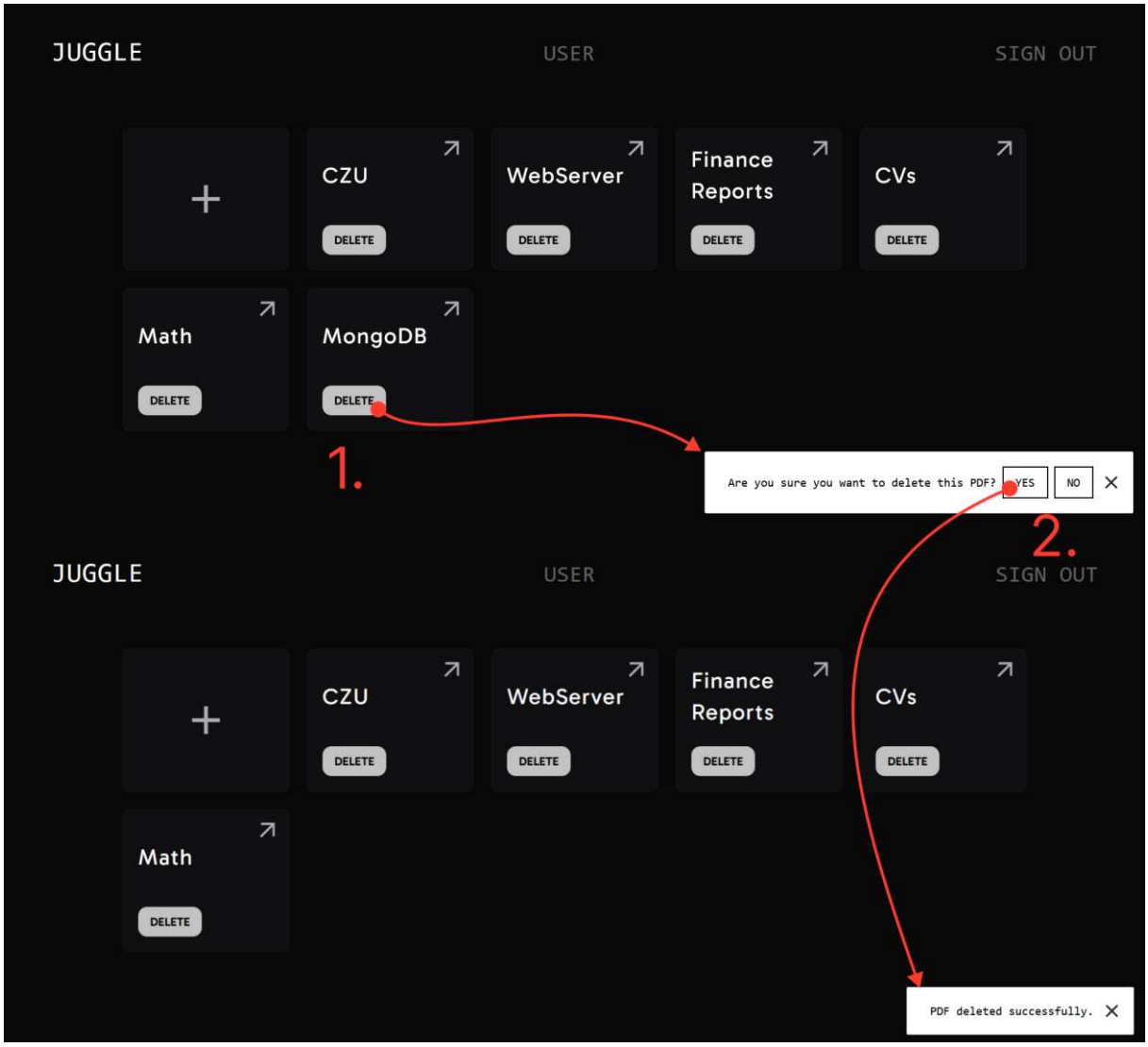
Na obrázku 8. je znázorněna navigace na stránce Dashboard, která zahrnuje práci s nahráváním souborů PDF, kde je k dispozici také funkce pro nahrávání sloučených souborů z pracovního zařízení uživatele, také je k dispozici také funkce pro mazání souborů PDF. Tyto stránky jsou dostupné pouze autorizovaným uživatelům.

Kliknutím na tlačítko na stránce ovládacího panelu se otevře Průzkumník souborů, šipka 1., kde může uživatel vybrat jeden nebo více souborů PDF najednou. Po výběru souborů se zobrazí pole pro zadání názvu stahovaného souboru, šipka 2. Po potvrzení názvu se soubor zobrazí na stránce ovládacího panelu, šipka 3.



Obrázek 8: Navigace k nahrávání souborů (zdroj: vlastní)

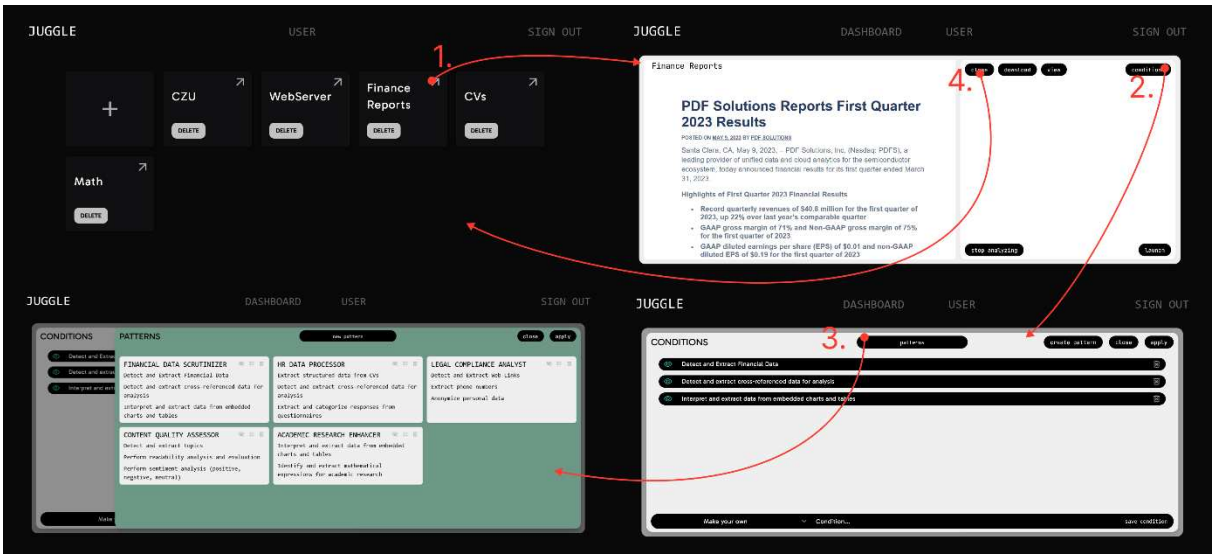
Obrázek 9. znázorňuje navigační logiku pro odstranění souboru. Po stisknutí tlačítka mazání se zobrazí zpráva potvrzující smazání vybraného souboru, šipka 1. Šipka 2 zobrazí zprávu, že soubor je úspěšně odstraněn.



Obrázek 9: Navigace k mazání souborů (zdroj: vlastní)

Obrázek 10. znázorňuje navigaci mezi stránkami a modulárními okny, které zahrnují práci s otevřenými soubory PDF ze seznamu na ovládacím panelu. Tyto stránky jsou dostupné pouze autorizovanému uživateli.

Pro práci se souborem musí uživatel kliknout na tlačítko se symbolem šipky. Vybraný soubor se otevře na nové záložce prohlížeče, šipka 1. Pokud kliknete na tlačítko conditions, zobrazí se modální okno pro práci s podmínkami, šipka 2. Pokud uživatel klikne na tlačítko vzory v modálním okně Conditions (Podmínky), zobrazí se modální okno pro práci se vzory, šipka 3. Každý panel má tlačítko zavření, které uživatele vrátí do předchozího otevřeného modálního okna nebo stránky, šipka 4.



Obrázek 10: Navgiace mezi stránkami a modulárními okny (zdroj: vlastní)

Na obrázku 11. je znázorněna navigace v modulárních oknech, která zahrnuje práci s vytvořením podmínky pro analýzu vybraného souboru PDF uživatelem. Tyto stránky jsou dostupné pouze autorizovanému uživateli.

Po zadání textu podmínky do textového pole musí uživatel kliknout na tlačítko pro uložení nové podmínky, šipka 1. Tato podmínka se bude vztahovat pouze na vybraný soubor PDF. Pokud se uživatel pokusí vytvořit stejnou podmínku, zobrazí se zpráva o nemožnosti ukládání duplikátů, šipka 2. Pokud chce uživatel vybrat uloženou podmínku, musí v roletovém menu vybrat možnost Saved (Uloženo), šipka 3. Pozice Saved (Uloženo) se zobrazí pouze v případě, že existují uložené podmínky, šipka 4. Po výběru uložené podmínky ji uživatel kliknutím na tlačítko , šipka 5., přidá do seznamu podmínek k analýze, což je potvrzeno zprávou.

Obrázek 11: Navigace v modulárních oknech k vytvoření vlastní podminky (zdroj: vlastní)

Obrázek 12. znázorňuje navigaci v modulárních oknech, která zahrnuje práci pomocí vzorů, které obsahují podmínky pro analýzu uživatelem vybraného souboru PDF. Tyto stránky jsou přístupné pouze autorizovanému uživateli.

Modální okno Patterns se otevře kliknutím na tlačítko patterns, šipka 1. Každý kontejner vzoru má tři tlačítka, z nichž jedno vypadá jako oko, které je zodpovědné za logiku zaškrtávacího políčka. Jakmile kliknutím na oko označíme vzor, zvýrazní se zaškrtávací políčko jako vybrané, šipka 2. Stejné podmínky ve vzorech nebudou při přidávání do společného seznamu podmínek duplikovány. Vybrané vzory lze použít kliknutím na tlačítko apply (aplikovat) a zobrazí se zpráva, šipka 3. Pro přepnutí do seznamu vzorů stačí kliknout na tlačítko close (zavřít), šipka 4. Pro použití nového seznamu podmínek musí uživatel kliknout na tlačítko apply v modálním okně Conditions, šipka 5.

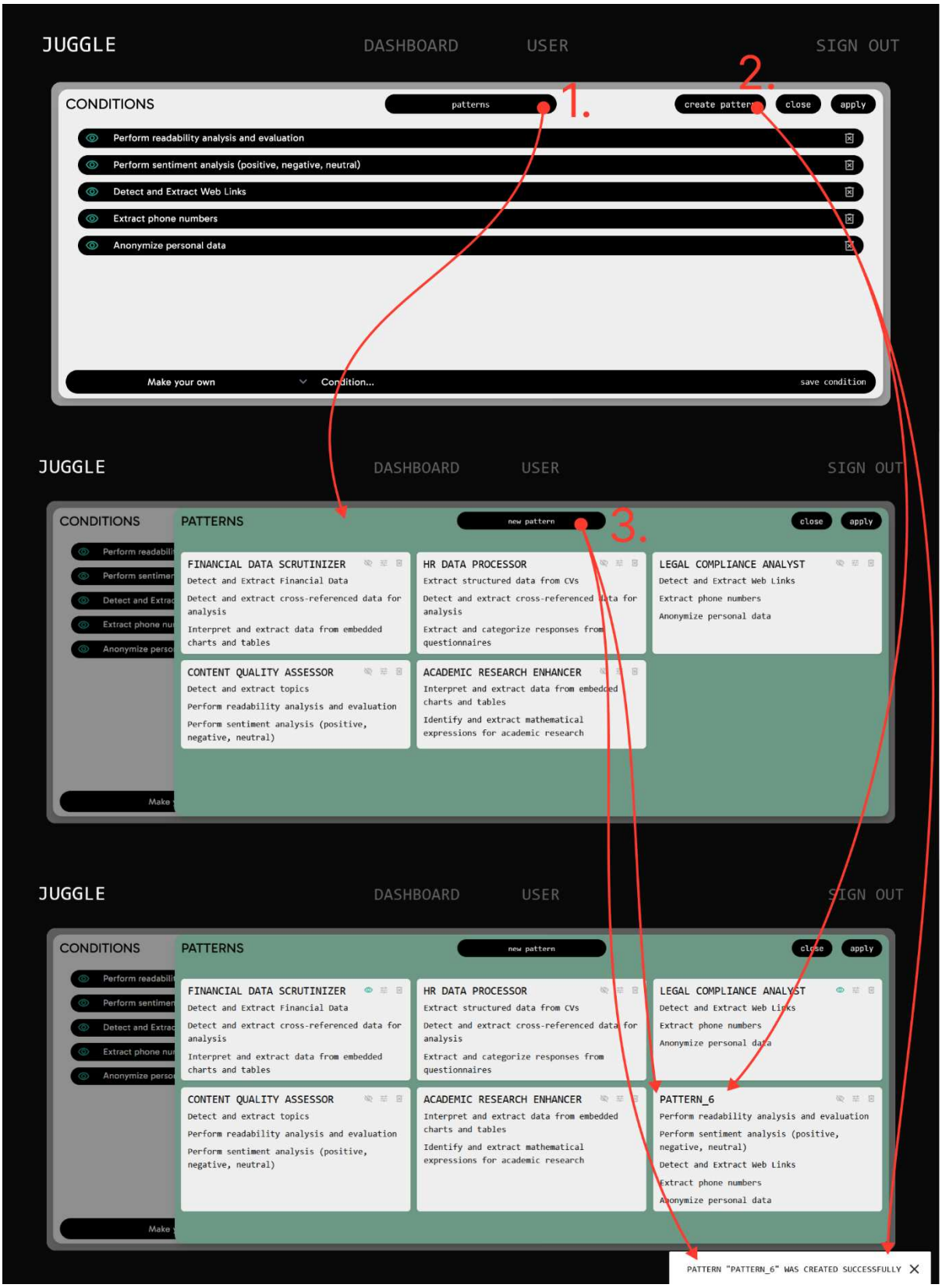
Obrázek 12: Navigace mezi modulárními stránkami pro práci se vzory (zdroj: vlastní)

Obrázky 13, 14, 15 znázorňují navigaci v modulárních oknech, která zahrnuje práci s vytvářením, editací a mazáním vzorů vybraného PDF souboru uživatelem. Tyto stránky jsou přístupné pouze autorizovanému uživateli.

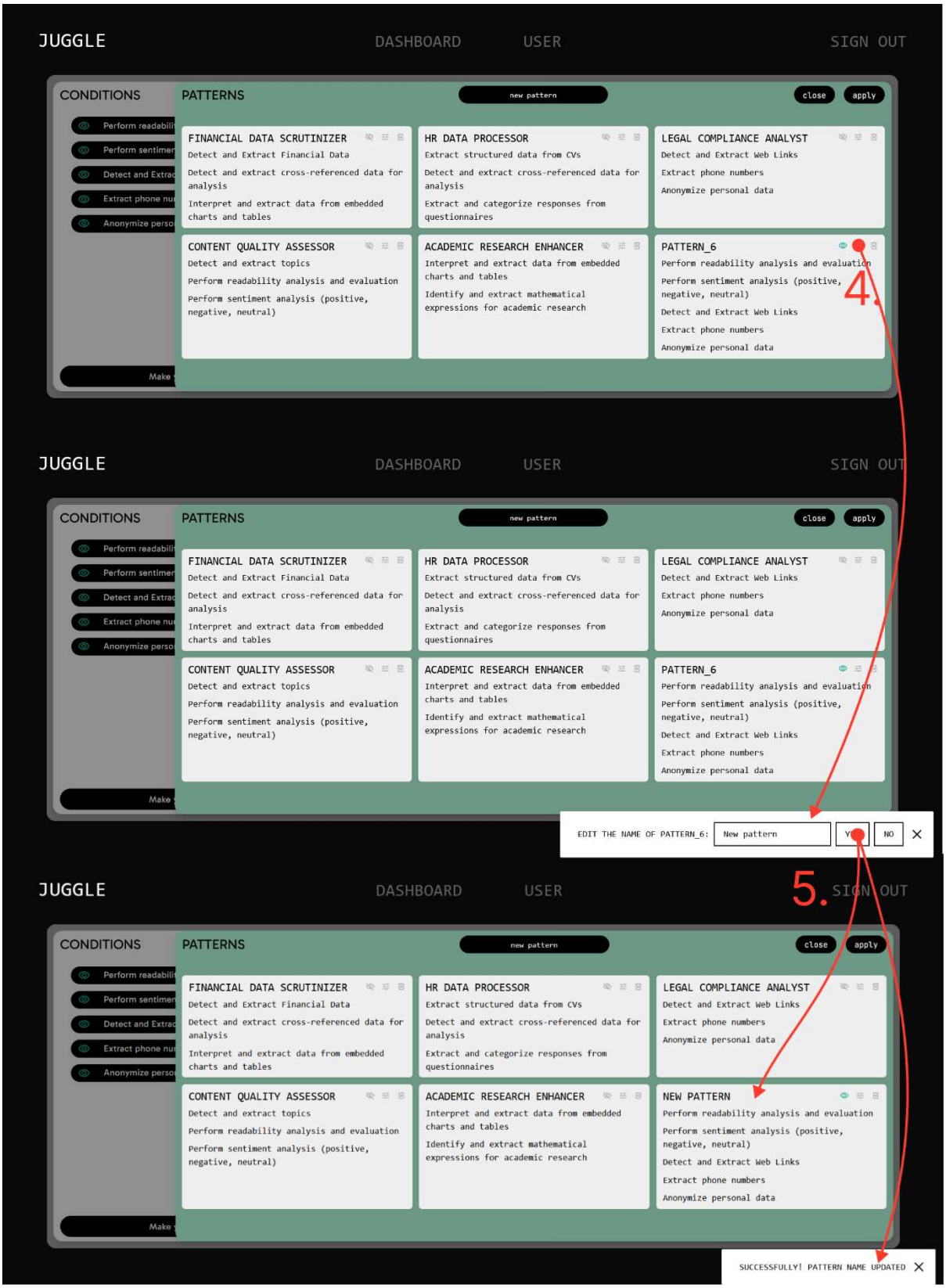
Vzor je vytvořen z aktuálního seznamu podmínek. Pro vytvoření vzoru nabízí aplikace uživateli dvě možnosti: kliknout na tlačítko v modálním okně Conditions, šipka 2., nebo přejít do modálního okna Patterns, šipka 1., a vytvořit nový vzor tam, šipka 3.. Výsledek je stejný.

Pro změnu vytvořeného vzoru je nutné kliknout na prostřední tlačítko v kontejneru vzoru, po jehož stisknutí může uživatel nastavit nový název, šipka 4. Potvrzením nového názvu se ihned zobrazí nové označení vzoru s příslušným hlášením, šipka 5.

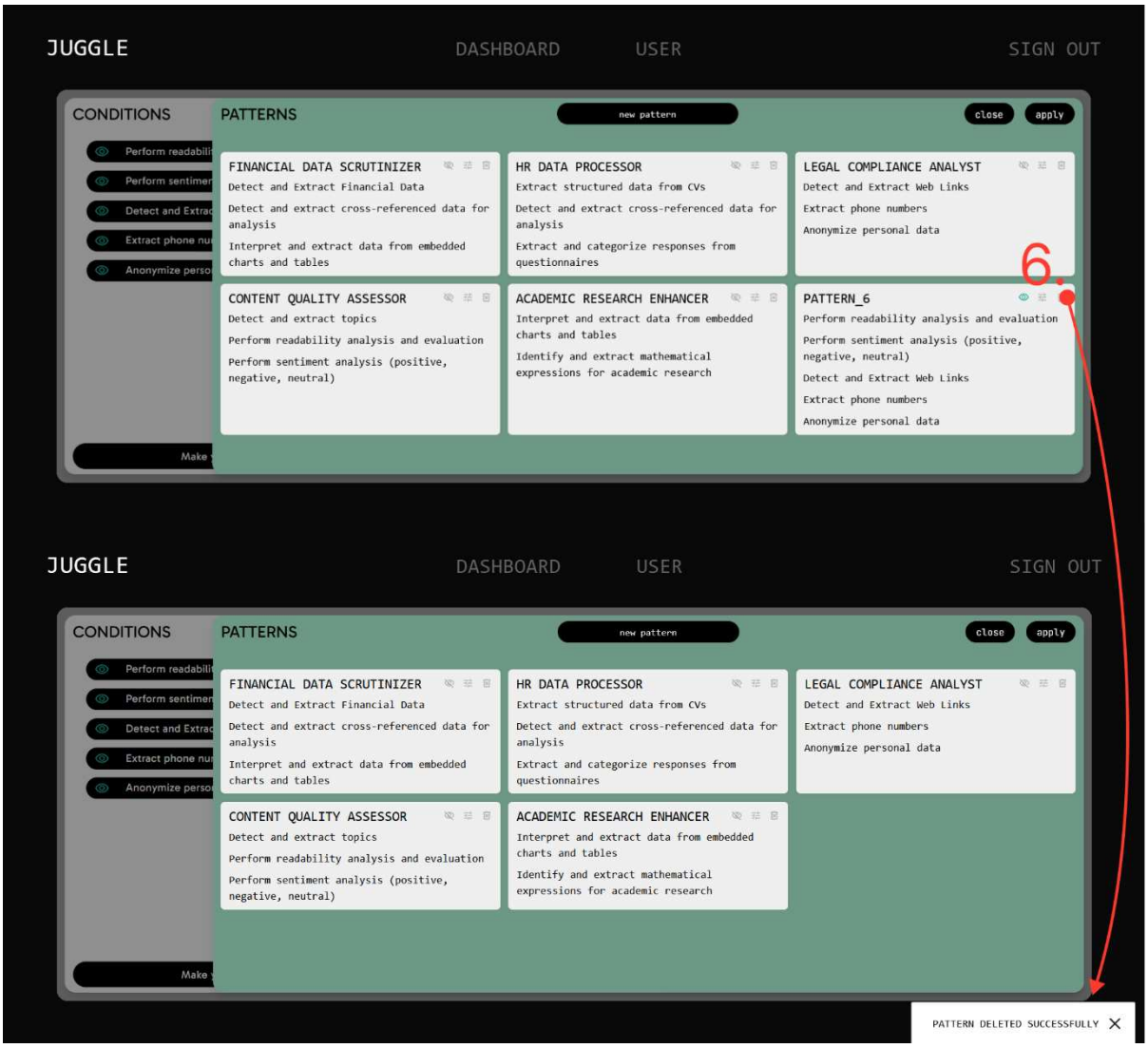
Pro odstranění vzoru stačí kliknout na pravé tlačítko uvnitř kontejneru, čímž bude vzor okamžitě odstraněn a uživatel bude upozorněn, šipka 6.



Obrázek 13: Navigace mezi modulárními stránkami pro vytvoření vzoru (zdroj: vlastní)



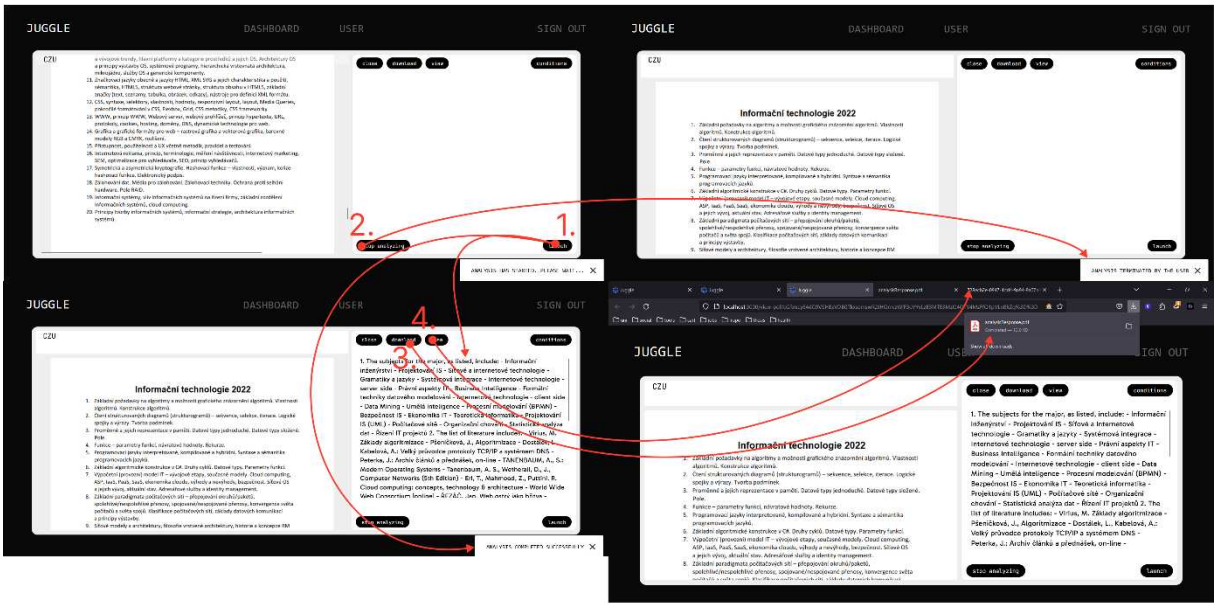
Obrázek 14: Navigace mezi modulárními stránkami pro editaci vzoru (zdroj: vlastní)



Obrázek 15: Navigace mezi modulárními stránkami pro mazání vzoru (zdroj: vlastní)

Obrázek 16. znázorňuje navigaci na stránce aplikace, která zahrnuje práci s analýzou souboru PDF vybraného uživatelem na základě stanovených podmínek pro uvedený soubor. Tyto stránky jsou přístupné pouze autorizovanému uživateli.

Uživatel musí pro analýzu souboru PDF kliknout na tlačítko launch (spustit), po čemž se objeví zpráva, že analýza je zahájena, a po úspěšném dokončení se zobrazí text hotové analýzy a zpráva o provedené analýze, šipka 1. Uživatel může také zastavit analýzu stisknutím tlačítka stop analyzing, šipka 2. Uživatel si může text analýzy stáhnout, šipka 3. nebo otevřít na nové záložce prohlížeče, šipka 4. Oba případy jsou ve formátu PDF.



Obrázek 16: Navigace mezi modulárními stránkami pro analýzu souboru (zdroj: vlastní)

5.4.3. Tvorba a připojení k REST API

Ve webové aplikaci určené ke správě souborů PDF je prvním krokem vytvoření bezpečného a funkčního backendu, který bude řešit aspekt ukládání a zobrazování souborů. Sestavení rozhraní REST API bude provedeno v této části práce pomocí Express.js, frameworku webových aplikací pro Node.js, a jeho spojením s Firebase pro autentifikaci a uchovávání souborů.

Server je zřízen pomocí Express.js. Zdrojový kód 5. ukazuje inicializaci serveru a middlewarovou konfiguraci, zahrnující statický hosting aplikace Angular a middleware pro autentifikaci k zabezpečení cest API.

```
const admin = require('firebase-admin');
const express = require('express');
const app = express();
const path = require('path');
const angularAppPath = path.join(__dirname, '../dist/juggle/browser');

app.use(express.static(angularAppPath));
app.use('/api', authenticateUser);
```

Zdrojový kód 5: server.js – Konfigurace serveru Node.js pro poskytování aplikací Angular a ověřování uživatelů (zdroj: vlastní)

express.static slouží pro frontend systému Angular, který je dostupný z kořenové adresy URL. Middleware *authenticateUser* je aplikován na všechny cesty začínané na /api, čímž se zajistí jejich zabezpečení pomocí ověřování.

Pro zabezpečení je klíčový proces autentizace. Jeho prostřednictvím se ověřuje existence a validita Bearer tokenu ve Zdrojovém kódu 6.

```
const authenticateUser = async (req, res, next) => {
  const authorizationHeader = req.headers.authorization;

  if (!authorizationHeader || !authorizationHeader.startsWith('Bearer ')) {
    console.error('No ID token was provided in the Authorization header.');
    res.status(403).send('Unauthorized');
    return;
  }

  const idToken = authorizationHeader.split('Bearer ')[1];
  try {
    req.user = await admin.auth().verifyIdToken(idToken);
    next();
  } catch (error) {
    console.error('Error while verifying Firebase ID token:', error);
    res.status(403).send('Unauthorized');
  }
};
```

Zdrojový kód 6: server.js - Middlewarová funkce pro ověřování uživatelů na základě tokenů Firebase ID (zdroj: vlastní)

Funkce middlewaru zachytí požadavek, extrahuje token, ověřuje jej pomocí Firebase Admin SDK a v případě, že je token neplatný, pokračuje k dalšímu middlewaru nebo požadavek odmítne.

V aplikaci (Zdrojový kód 7) je znázorněn koncový bod API pro načtení všech souvisejících souborů PDF pro uživatele.

```
app.get('/api/fetch-all-pdfs', authenticateUser, async (req, res) => {
  const userUid = req.user.uid;

  try {
    const [files] = await bucket.getFiles({prefix: `pdfs/${userUid}/`});
      const metadataPromises = files.map(file => file.getSignedUrl({action:
      'read', expires: '03-09-2100'})
      .then(url => ({
        id: file.name, url, path: file.metadata.selfLink
      })));

    const filesMetadata = await Promise.all(metadataPromises);
    res.json(filesMetadata);
  } catch (error) {
    console.error("Error fetching PDF metadata:", error);
    res.status(500).json({
      message: "Failed to fetch PDF metadata from Firebase Storage.", error:
      error.message,
    });
  }
});
```

Zdrojový kód 7: server.js - Koncový bod API pro načítání souborů PDF s metadaty, ověřený přes Firebase (zdroj: vlastní)

Tento koncový bod využívá úložiště Firebase Storage k načtení všech souborů PDF patřících ověřenému uživateli a vrací metadata včetně podepsané adresy URL pro každý soubor, čímž se zajistí bezpečný přístup k těmto souborům.

Ve Zdrojovém kódu 8. je znázorněna služba Angular pro přístup k tomuto API.

```
@Injectable({
  providedIn: 'root'
})
export class UploadService {

  constructor(
    private storage: AngularFireStorage,
    private db: AngularFirestore,
    private http: HttpClient,
    private afAuth: AngularFireAuth
  ) {
  }

...

  fetchAllPDFs(): Observable<PdfData[]> {
    return this.afAuth.idToken.pipe(
      switchMap(token => {
        const headers = { 'Authorization': `Bearer ${token}` };
        return this.http.get<PdfData[]>('/api/fetch-all-pdfs', { headers });
      }),
      catchError(error => {
        console.error('Error fetching PDFs:', error);
        return throwError(error);
      })
    );
  }
}

interface PdfData {
  id: string;
  url: string;
  path: string;
}
```

Zdrojový kód 8: openai.service.ts - Služba Angular pro načítání dat PDF, zpracování ověřování a požadavků HTTP (zdroj: vlastní)

Služba *UploadService* využívá *HttpClient* modulu Angular ve spolupráci s *AngularFireAuth* pro ověřování a provedení požadavku HTTP GET. Hlavičku Authorization konstruuje pomocí

tokenu *Bearer* získaného z autorizace Firebase, což zajišťuje, že je požadavek řádně autorizován

OpenAI API

Základem aplikace je funkční požadavek realizovaný prostřednictvím koncového bodu */api/analyze-pdf-firebase*, který zpracovává a analyzuje obsah souboru PDF uloženého v úložišti Firebase Storage pomocí jazykového modelu OpenAI.

Struktura koncového bodu je nastavena tak, že přijímá požadavek POST obsahující název souboru PDF tzv. *pdfFileName* a sadu podmínek tzv. *Conditions*, což jsou požadavky, které chce uživatel provést nebo zjistit na základě PDF souboru.

```
app.post('/api/analyze-pdf-firebase', async (req, res) => {
  let {pdfFileName, conditions} = req.body;

  if (!pdfFileName) {
    return res.status(400).send('PDF file name not provided.');
  }

  if (!conditions || !Array.isArray(conditions) || conditions.length === 0) {
      return res.status(400).send('Conditions are required and must be an
      array.');
  }

  pdfFileName = decodeURIComponent(pdfFileName);
```

Zdrojový kód 9: server.js - Koncový bod API pro zahájení analýzy PDF s kontrolou platnosti (zdroj: vlastní)

Ve Zdrojovem kódu 10. je znázorněna nutné parametry, které musí mít správnou strukturu. Dekóduje také název souboru PDF, který může být zakódován v adrese URL.

Po úspěšném schválení vstupních údajů se server pokusí načíst zadaný soubor PDF z úložiště Firebase.

```
const file = bucket.file(pdfFileName);
const [fileBuffer] = await file.download();
const pdfData = await pdfParse(fileBuffer);

const pages = pdfData.text.split(/(?=Page \d+)/);
```

Zdrojový kód 10: server.js - Fragment kódu pro stahování a analýzu obsahu PDF a jeho rozdělení podle stránek (zdroj: vlastní)

PDF soubor je načten jako buffer a k extrakci textu z PDF je použita knihovna *pdf-parse*. Tento text je poté rozčleněn na stránky a připraví se k analýze, kdy se s každou stránkou pracuje jako se samostatným obsahem. Pro analýzu textu PDF se používá model GPT společnosti OpenAI.

```
const processPage = async (pageText, pageIndex) => {
  const instructions = "Please provide responses based solely on the
  provided text.";
  const combinedConditionsText = conditions.map((condition, index) =>
  `${index + 1}. ${condition.text}`).join("\n");

  const messageContent =
  `${instructions}\n\n${combinedConditionsText}\n\nPage ${pageIndex +
  1}:\n${pageText}`;
  const messages = [{role: "user", content: messageContent}];

  const response = await
  axios.post('https://api.openai.com/v1/chat/completions', {
    model: "gpt-4-0125-preview",
    messages: messages,
    temperature: 0.7,
    max_tokens: 4096,
  }, {
    headers: {
      'Content-Type': 'application/json',
      'Authorization': `Bearer ${process.env.OPENAI_API_KEY}`
    }
  });

  return response.data;
};
```

Zdrojový kód 11: server.js - Funkce pro analýzu textu ze stránky PDF pomocí modelu GPT od OpenAI s vlastními podmínkami zahrnutými v požadavku (zdroj: vlastní)

Funkce *processPage* slouží k tvorbě strukturované zprávy, která kombinuje uživatelské instrukce, podmínky a text stránky. Tato zpráva je následně odeslána rozhraní OpenAI API ke zpracování.

API spravuje vytíženost omezováním celkového počtu současných požadavků na rozhraní API OpenAI, čímž je zajištěna rychlá a efektivní odezva serveru.

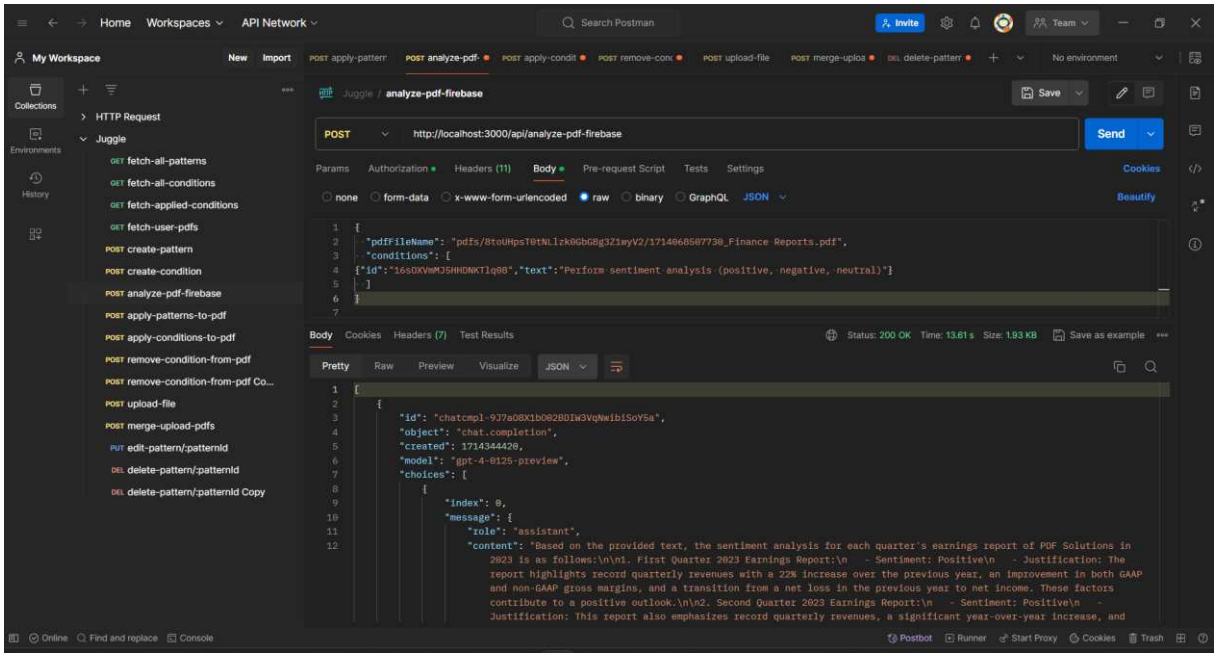
```
const MAX_CONCURRENT_REQUESTS = 5;
const results = [];
for (let i = 0; i < pages.length; i += MAX_CONCURRENT_REQUESTS) {
  const requests = pages.slice(i, i + MAX_CONCURRENT_REQUESTS).map((page,
  index) => processPage(page, i + index));
  results.push(...(await Promise.all(requests)));
}
```

Zdrojový kód 12: server.js - Fragment kódu pro správu souběžných požadavků na zpracování více stránek PDF v dávkách (zdroj: vlastní)

Díky tomuto mechanismu se stránky zpracovávají v sadách, při dodržení limitu *MAX_CONCURRENT_REQUESTS*, což umožňuje vyhnout se zatížení serveru i externího rozhraní API.

5.4.4. Testování endpointů

Pro testování jednotlivých implementovaných koncových bodů se používá osvědčený nástroj Postman. Před použitím každého koncového bodu přímo v aplikaci se tento nástroj používá k vytvoření a otestování serverové části aplikace. Obrázek 17. znázorňuje jeden z koncových bodů, v kolekci pod názvem Juggle lze také nalézt všechny implementované koncové body.



Obrázek 17: Kolekce koncových bodů v programu Postman (zdroj: vlastní)

ZÁVĚR

Tato práce se zaměřila na využití současných modelů umělé inteligence v aplikaci zaměřující se na vyhledání konkrétních dat ve více PDF souborech a vyhodnocení nalezené množiny dat v kontextu zadaných vstupních podmínek. Cílem bylo zvýšit rychlost pro nalezení, zpracování, čtení a agregaci dat z více souborů s využitím webových technologií a umělé inteligence. Využité modely umělé inteligence pomohli s manipulací s daty ve formátu PDF a shrnutí surových dat do strukturovaných a uživatelsky přívětivých výsledků. Pro využití tohoto řešení byly prozkoumány možnosti poskytovaných modelů včetně ukázek jejich použití a tyto poznatky následně umožnili implementaci do aplikace Juggle, přičemž pro její tvorbu bylo také nutné komplexněji pochopit zpracování dat.

Teoretická část práce se zaměřila na vývojové frameworky, a to na Node.js, Angular a alternativní frontend frameworky včetně jejich srovnání. Do kapitoly Práce s PDF soubory byly shrnuty informace o struktuře a vlastnostech těchto souborů a jejich možné využití v kontextu webových aplikací. Stěžejní kapitola Uměla inteligence představuje základní algoritmy AI a popisuje její využití pro analýzu dat. Závěrečná kapitola se zabývá analýzou, návrhem a implementací výsledné aplikace včetně použitých technologií a využití API OpenAI.

Výstupní aplikace představuje prototyp, a pro její širší použití by bylo vhodné jej rozšířit o další funkcionality převážně v oblasti volby vstupních podmínek. Při tvorbě této práce mi získané dovednosti spolu s nabytými znalostmi o využití umělé inteligence podstatně zvýšily schopnosti při vývoji softwaru a nyní jsem se začal zajímat o to, jak tento nástroj v budoucnu vytvářet jako samostatné podnikové řešení v oblasti analýzy dat.

POUŽITÁ LITERATURA

[1] Overview of Blocking vs Non-Blocking. *NodeJS.org* [online]. 2024 [cit. 24.3.2024]. Dostupné z: https://nodejs.org/en/docs/guides/blocking-vs-non-blocking

[2] Express/Node introduction. *Developer.mozilla.org* [online]. 2024 [cit. 11.4.2024]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

[3] Node Hero – Using NPM: Tutorial. *Blog.risingstack.com* [online]. 2022 [cit. 25.3.2024]. Dostupné z:: https://blog.risingstack.com/node-hero-npm-tutorial/

[4] What is Express.js? *Codeacademy.com* [online]. 2024 [cit. 11.4.2024]. Dostupné z: https://www.codecademy.com/article/what-is-express-js

[5] What Is Express.js? Everything You Should Know. *Kinsta.com* [online]. 2023 [cit. 12.4.2024]. Dostupné z: https://kinsta.com/knowledgebase/what-is-express-js/

[6] Writing middleware for use in Express apps. *ExpressJS.com* [online]. 2017 [cit. 12.4.2024]. Dostupné z: https://expressjs.com/en/guide/writing-middleware.html

[7] Using middleware. *ExpressJS.com* [online]. 2017 [cit. 12.4.2024]. Dostupné z: https://expressjs.com/en/guide/routing.html

[8] ARISTEIDIS, B. Learning Angular - Fourth Edition: A no-nonsense guide to building web applications with Angular. 2023 [cit. 13.4.2024]. Packt Pub, ISBN 9781803240602. Dostupné z: https://dl.ebooksworld.ir/books/Learning.Angular.4th.Edition.Aristeidis.Bampakos.Packt.9781803240602.EBooksWorld.ir.pdf

[9] An Introduction To Angular Modules. *Medium.com* [online]. 2022 [cit. 14.4.2024]. Dostupné z: https://medium.com/@kevinmavani/an-introduction-to-angular-modules-db75c3ebf842

[10] Using the Fetch API. *Developer.mozilla.org* [online]. 2024 [cit. 14.4.2024]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

[11] RATHINAM, S. Analysis and Comparison of Different Frontend Frameworks [online]. 2022 [cit. 15.4.2024]. Dostupné z: https://www.suryaanshrathinam.com/static/media/SuryaanshRathinam_ComparisonOfFrontendFrameworks.a6d0c12b1b711c4cc8aa.pdf

[12] BORZI, F. Which one is the most actively developed Front-End framework in 2024? — React vs Angular vs Vue vs Svelte vs Ember. *Medium.com* [online]. 2024 [cit. 15.4.2024]. Dostupné z: https://javascript.plainenglish.io/which-one-is-the-most-actively-developed-front-end-framework-in-2024-d662c9951ecc

[13] Adobe Systems Incorporated. PDF format. *Adobe.com* [online]. 2024 [cit.16.4.2024]. Dostupné z: https://www.adobe.com/acrobat/about-adobe-pdf.html

[14] Apache PDFBox. *Pdfbox.apacge.org* [online]. 2024 [cit. 16.4.2024]. Dostupné z: https://pdfbox.apache.org/index.html

[15] Adobe Acrobat SDK. *Adobe.com* [online]. 2021 [cit. 16.4.2024]. Dostupné z: https://opensource.adobe.com/dc-acrobat-sdk-docs

[16] ZEESHAN, W. The 5 Node.js PDF Libraries Every Developer Must Know. *Dev.to* [online]. 2023 [cit. 16.4.2024]. Dostupné z: https://dev.to/xeshan6981/the-5-nodejs-pdf-libraries-every-developer-must-know-4b39#:~:text=generation%20and%20manipulation.-,Features,for%20creating%20visually%20appealing%20documents

[17] KARAKAYA, H. Complete Guide to PDF.js. *Pspdfkit.com* [online]. 2024 [cit. 17.4.2024]. Dostupné z: https://pspdfkit.com/blog/2023/complete-guide-to-pdfjs/

[18] What is artificial intelligence (AI)? *Ibm.com* [online]. 2023 [cit. 17.4.2024]. Dostupné z: https://www.ibm.com/topics/artificial-intelligence

[19] What is Artificial Intelligence (AI)? *Google.com* [online]. 2024 [cit. 17.4.2024]. Dostupné z: https://cloud.google.com/learn/what-is-artificial-intelligence

[20] LASKOWSKI, N, What is artificial intelligence (AI)? Everything you need to know. *Techtarger.com* [online]. 2024 [cit. 18.4.2024]. Dostupné z: https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-Intelligence

[21] DIAZ, M. What is AI? Everything to know about artificial intelligence. *Zdnet.com* [online]. 2024 [cit. 18.4.2024]. Dostupné z: https://www.zdnet.com/article/what-is-ai-heres-everything-you-need-to-know-about-artificial-intelligence/

[22] PENDYALA, V. S. Machine Learning for Societal Improvement, Modernization, and Progress 2022 [cit. 19.4.2024]. Spojené státy americké: IGI Global, ISBN 978-1668440476.

[23] Text generation models. *Openai.com* [online]. 2024 [cit. 22.4.2024]. Dostupné z: https://platform.openai.com/docs/guides/text-generation

[24] Image generation. *Openai.com* [online]. 2024 [cit. 22.4.2024]. Dostupné z: https://platform.openai.com/docs/guides/images

[25] Vision. *Openai.com* [online]. 2024 [cit. 22.4.2024]. Dostupné z: https://platform.openai.com/docs/guides/vision

[26] What are Iaas, Paas and Saas? *Ibm.com* [online]. 2024 [cit 2.5.2024]. Dostupné z: https://www.ibm.com/topics/iaas-paas-saas

[27] How Fast Growing Companies use SaaS to Increase Growth Rates by 19.6%. *Superoffice.com* [online]. 2023 [cit 2.5.2024]. Dostupné z: https://www.superoffice.com/blog/saas/

[28] Gemini API Overview. *Google.dev* [online]. 2024 [cit. 24.4.2024]. Dostupné z: https://ai.google.dev/gemini-api/docs/api-overview

[29] Top 20 AI Development Frameworks & Libraries in 2024. *Techvify-software.com* [online]. 2024 [cit. 24.4.2024]. Dostupné z: https://techvify-software.com/best-artificial-intelligence-framework/

[30] GPT-4 Turbo and GPT-4. *Openai.com* [online]. 2024 [cit. 25.4.2024]. Dostupné z: https://platform.openai.com/docs/models/gpt-4-turbo-and-gpt-4

[31] GPT-3.5 Turbo. *Openai.com* [online]. 2024 [cit. 25.4.2024]. Dostupné z: https://platform.openai.com/docs/models/gpt-3-5-turbo

[32] About Postman. *Postman.com* [online]. 2024 [cit. 26.4.2024]. Dostupné z: https://www.postman.com/about-postman/

[33] WebStrom Features. *Jetbrains.com* [online]. 2024 [cit. 27.4.2024]. Dostupné z: https://www.jetbrains.com/webstorm/features/

[34] STEVENSON, D. What is Firebase? The complete story, abridged. *Medium.com* [online]. 2018 [cit. 27.5.2024]. Dostupné z: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0

PŘÍLOHY

Příloha A – Webová aplikace - Juggle.....75

PŘÍLOHA A – WEBOVÁ APLIKACE - JUGGLE

Složka Juggle.zip obsahuje:

1. Zdrojový kód aplikace
2. Grafické uživatelské rozhraní
3. Nastavení a spuštění webové aplikace.