VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÁ APLIKACE PRO EDITACI A BIOMETRICKÉ PODEPISOVÁNÍ DOKUMENTŮ

WEB APPLICATION FOR EDITING AND BIOMETRIC SIGNING OF DOCUMENTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ARAM DENK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

156349

Ústav: Ústav informačních systémů (UIFS)
Student: **Denk Aram**
Program: Informační technologie
Název: **Webová aplikace pro editaci a biometrické podepisování dokumentů**
Kategorie: Informační systémy
Akademický rok: 2023/24

Zadání:

1. Prostudujte principy tvorby webových aplikací, práci s dokumenty ve formátu PDF a problematiku biometrického podepisování dokumentů. Seznamte se také s existujícími řešeními v oblasti biometrického podepisování a zhodnoťte jejich nedostatky.
2. Analyzujte požadavky na webovou aplikaci umožňující podepisování dokumentů pomocí podpisového tabletu, editaci a zobrazení podpisových polí v dokumentu a export finálního dokumentu.
3. Navrhněte aplikaci dle požadavků, návrh konzultujte s vedoucím.
4. Implementujte navrženou aplikaci.
5. Otestujte vytvořené řešení.
6. Zhodnoťte dosažené výsledky a další možnosti pokračování tohoto projektu.

Literatura:

- Smejkal, V.: Kryptografický a dynamický biometrický podpis dle platné právní úpravy. Beck-Online, 2019. Dostupné na: https://www.beck-online.cz/bo/chapterview-document.seam?documentId=nrptembrhfpxa4s7geyf6427gm2dglttnm3a
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9.
- Welling, L., Thomson, L.: PHP a MySQL: Kompletní průvodce vývojáře. CPress, 2017.

Při obhajobě semestrální části projektu je požadováno:
Body 1-3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1.11.2023
Termín pro odevzdání: 9.5.2024
Datum schválení: 30.10.2023

Abstrakt

Bakalářská práce se zabývá návrhem a vývojem webové aplikace pro editaci a biometrické podepisování dokumentů ve formátu PDF použitím podpisového tabletu. Práce se věnuje studiu principů tvorby webových aplikací jehož součástí je srovnání JavaScriptových frameworků, také se věnuje studiu práce s dokumenty ve formátu PDF a problematice biometrických podpisů. Jsou zde analyzovány existujících řešení biometrického podepisování dokumentů. Na základě této analýzy jsou stanoveny požadavky na novou webovou aplikaci a proveden její návrh. Následuje popis samotné implementace, při níž bylo použito převážně nástroje React v jazyce JavaScript. Po implementaci je aplikace testována a výsledky jsou zhodnoceny.

Abstract

The bachelor thesis deals with the design and development of a web application for editing and biometric signing of PDF documents using a signature tablet. The thesis focuses on the study of the principles of web application development which includes a comparison of JavaScript frameworks, it also studies manipulation with PDF documents and the issues surrounding biometric signatures. Existing solutions for biometric signing of documents are analyzed. Based on this analysis, the requirements for a new web application are determined and its design is carried out. This is followed by a description of the actual implementation, which is mainly done in JavaScript using the framework React. After implementation, the application is tested and the results are evaluated.

Klíčová slova

Webové aplikace, Biometrické podepisování, Bezpečnost dat, Digitální podpisy, Biometrické technologie, Autentizace

Keywords

Web applications, Biometric signing, Data security, Digital signatures, Biometric technologies, Authentication

Citace

DENK, Aram. *Webová aplikace pro editaci a biometrické podepisování dokumentů.* Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Webová aplikace pro editaci a biometrické podepisování dokumentů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Další informace mi poskytl Ing. Miloš Prokop. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....

Aram Denk

9. května 2024

Obsah

Seznam obrázků

Kapitola 1

Úvod

V dnešní době rozmachu moderních technologií a všudypřítomné digitalizace se stále více odkláníme od tradičních papírových dokumentů k jejich digitálním ekvivalentům. Digitalizace s sebou přináší mnoho výhod, ale také nové výzvy. Jednou z nich je digitalizace procesu podepisování dokumentů. I přes mnohé výhody, které digitalizace přináší, se digitální podpisy dosud nestaly plnohodnotnou náhradou za podpisy tradiční. Ačkoliv certifikované digitální podpisy poskytují velmi vysokou úroveň ověření identity, jejich složitost brání jejich širšímu využití v případech, kdy se běžně používá tradičních podpisů. Navíc, ne každý člověk vlastní digitální certifikát, který je pro certifikovaný digitální podpis nutný, zatímco vytvoření tradičního podpisu pomocí psacího pera je běžnou dovedností. Převod tradičního podpisu do digitálního formátu ovšem není jednoduchý proces. Běžná praxe vkládání podpisu do digitálního dokumentu zahrnuje tisk, ruční podepsání, a následné skenování dokumentu, což je časově i finančně náročné.

S těmito problémy jsme se setkali i ve firmě Seacomp a to například v situacích, kdy je nutné aby v rámci námi vyvíjeného webového ambulantního informačního systému pacient podepsal informovaný souhlas. Cílem této práce je proto vytvořit webovou aplikaci, která umožní podepisování digitálních dokumentů, tak intuitivně, jako by se jednalo o tradiční papírové dokumenty. Pro zachytávání biometrických podpisů vytvořených tahem ruky aplikace využívá specializované podpisové tablety. Tento způsob vytváření podpisů je intuitivní a může být snadno používán i osobami s nízkou digitální gramotností.

Na trhu existuje několik podobných řešení, která jsou analyzována v části 2.2. Všechny ale mají z našeho pohledu nedostatky, které nás vedly k rozhodnutí vyvinout vlastní aplikaci s lepší integrací do našeho systému. Tato práce se nejprve zabývá studiem problematiky spojené s podepisováním dokumentů ve webovém prohlížeči. Získané poznatky jsou popsány v kapitole 2. Jsou zde rozebrány aspekty biometrických podpisů, funkce a historie formátu PDF a také moderní principy vývoje webových aplikací. Na základě tohoto výzkumu a požadavků na aplikaci stanovených v kapitole 3, bylo v kapitole 4 navrženo uživatelské rozhraní a architektura aplikace. V kapitole 5 je pak popsán postup implementace. Aplikace prošla testováním, jehož průběh a výsledky jsou uvedeny v kapitole 6. Snímky realizace aplikace jsou k dispozici v příloze B.

Kapitola 2

Teoretická část

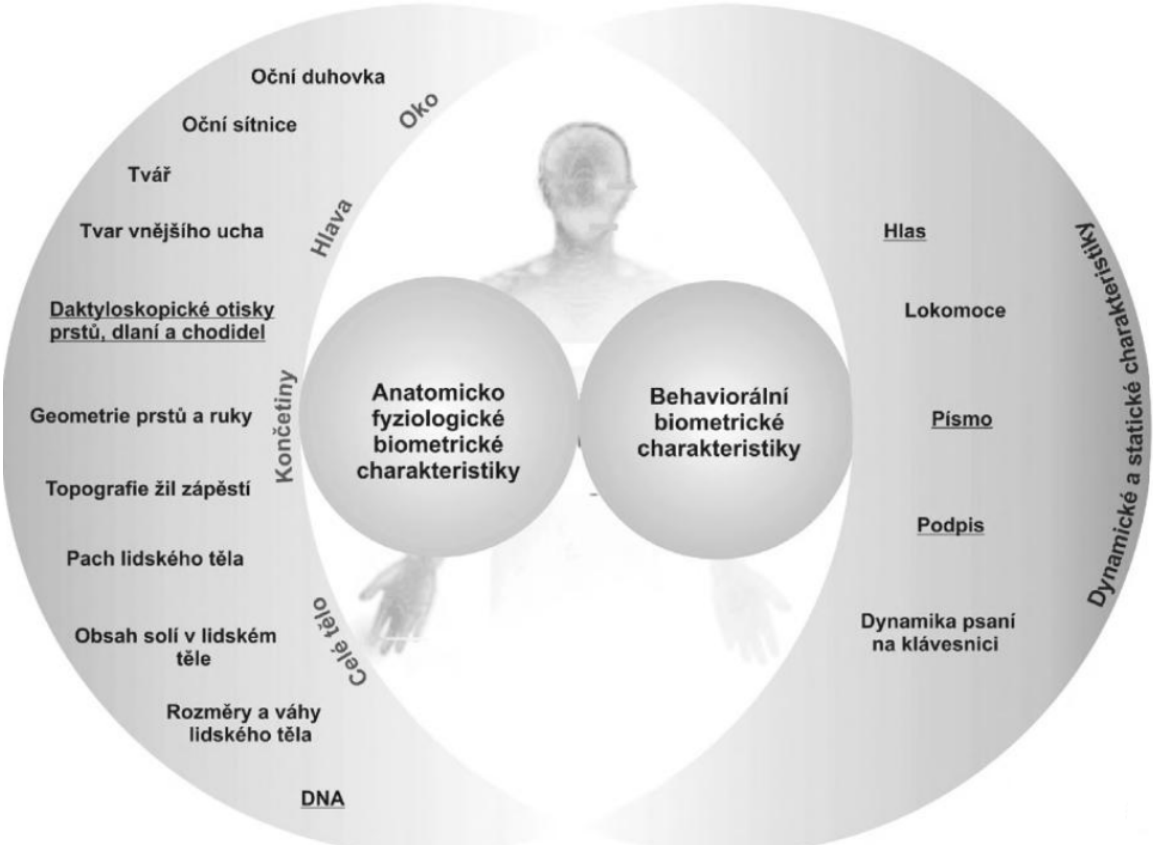
Teoretická část této práce se věnuje problematice spojené s tvorbou webové aplikace pro biometrické podepisování dokumentů. Nejdříve je přiblížena problematika samotných biometrických podpisů, a to v kapitole 2.1. V této kapitole je popsáno dělení a jednotlivé typy biometrických charakteristik a jejich vhodné použití. Problematikou biometrického podepisování dokumentů se zabývá již několik existujících komerčních řešení, v kapitole 2.2 jsou tato řešení podrobněji zkoumána a jsou zde vypsány jejich slabé a silné stránky, je zde také odůvodněno, proč bylo vytvořeno řešení nové. Protože výstupem praktické části této práce je aplikace schopná podepisovat dokumenty v běžně používaném formátu PDF, je v kapitole 2.3 popsána historie tohoto formátu a princip jeho fungování. V kapitole 2.4 je pak popsáno, jak se dokumenty ve formátu PDF pracuje, a to jak z pohledu běžného uživatele, tak i z pohledu programátora. Poslední část teoretické části (2.5) se věnuje principům tvorby webových aplikací, jsou zde popsány dvě hlavní technologie, pomocí kterých je možné webové aplikace vytvářet a poté jsou porovnány nejvíce používané JavaScriptové frameworky pro vytváření webových aplikací.

2.1 Biometrický podpis

V této kapitole bych chtěl přiblížit problematiku biometrického podepisování. Obecně se jedná o problém identifikace osoby. K identifikaci osoby může být použito vlastnictví, a to znalostní i fyzické. V případě znalostního vlastnictví se může jednat například o znalost rodného čísla, PIN kódu či hesla. K identifikaci pomocí fyzického vlastnictví se používá například identifikačních karet, nebo také klíčů, čipů a podobných zařízení. Další možností principu identifikace osoby je na základě biologických vlastností, a to fyzických i psychických, což se nazývá právě biometrickou identifikací. Biometrická identifikace má výhodu v tom, že je pro člověka přirozená. Člověk si nemusí nic pamatovat a uchovávat, biometrická identita je s ním totiž spojena již od narození. V oblasti policejně-soudní je možné používat široké spektrum biometrických znaků, pro praktické použití k podepisování dokumentů je výběr znaků značně omezen. Biometrické charakteristiky je možné rozdělit na dvě základní kategorie, a to anatomicko-fyziologické a behaviorální [29].

2.1.1 Anatomicko-fyziologické charakteristiky

Jsou založeny na časově stálých fyzických vlastnostech lidského těla. Jde o identifikaci například na základě:



Obrázek 2.1: Členění biometrických charakteristik. (Převzato z knihy [29])

- **Oční duhovky**: Barevná část kolem panenky umožňuje velmi rychlou a přesnou identifikaci osoby.
- **Oční sítnice**: Dříve velmi často používaný způsob identifikace. Kvůli vysokým nákladům se od něj nyní upouští [13].
- **Tváře**: Analyzují se rysy obličeje, jako jsou tvary a umístění očí, nosu, rtů a brady.
- **Stavbě vnějšího ucha**: V bezpečnostních systémech zřídka používaný princip identifikace. Častěji se používá ve forenzní vědě a také v oblasti zdravotnictví, například pro identifikaci pacientů.
- **Otisků prstů, dlaní a chodidel**: Jeden z nejrozšířenějších způsobů biometrické identifikace osob. Používá se v oblastech bezpečnosti, kriminalistiky, zdravotnictví i forenzní vědy. Metody identifikace pomocí otisků prstů, dlaní a chodidel stojí na analýze papilárních linií, jejichž tvary jsou unikátní pro každou osobu.
- **Topografii žil v zápěstí**: Identifikace na základě topografie žil v zápěstí je užitečná zejména v kriminalistice a forenzní vědě. Pachatelé trestných činů si nezřídka zakrývají identifikovatelné části těla, jakými jsou například tvář, či otisky prstů, často ale opomíjí zakrytí právě zápěstí. Zápěstí je často viditelné u teroristů a nebo výtržníků při gestech triumfu nebo pozdravu, zatímco pedofilové je obvykle ukazují, když se dotýkají obětí [21].

- **Skladby DNA**: Biometrická identifikace na základě analýzy skladby DNA je soubor metod, které umožňují určit totožnost z biologického vzorku. Analyzují se vybrané úseky DNA. Tento způsob je široce využíván jak v trestních případech, tak ve sporech občanskoprávních. K identifikaci za účelem autentizace oprávnění přístupu k datům, nebo k podepisování dokumentů není tohoto způsobu hojně využíváno, a to hlavně z důvodu nutnosti odebrání biologického vzorku a relativně drahého procesu analýzy [29].

Tyto charakteristiky jsou často využívány pro autentizaci osoby k přístupu do zařízení nebo k datům. Například otisku prstu a stavbě tváře je dnes běžně využíváno k odemykání mobilních zařízení. Pro podepisování dokumentů je z této skupiny nejvhodnější použití otisků prstů. Velkou výhodou této skupiny charakteristik je, že na jejich základě je možné identifikovat i osoby negramotné, nebo ze zdravotních důvodů neschopné použít běžného podpisu tahem ruky. Nevýhodou je možnost získání nefalšovaného originálu i po smrti dané osoby [20].

2.1.2 Behaviorální charakteristiky

Tyto charakteristiky jsou založeny na chování a akcích jednotlivce. Příklady behaviorálních charakteristik zahrnují:

- **Oční pohyby**: Každý jednotlivec má charakteristické vzory očních pohybů, které mohou být sledovány a analyzovány. Tato charakteristika zahrnuje rychlost a směr pohybu očí, jež jsou často spjaté s myšlenkovými procesy [2].
- **Hlas**: Hlasové charakteristiky jednotlivce zahrnují rychlost řeči, tón, frekvenci a další parametry.
- **Chůze**:Každý člověk má svůj vlastní specifický způsob chůze, který je determinován anatomickými, fyziologickými a biomechanickými charakteristikami. Systémy sledující chůzi mohou identifikovat osobu na základě různých faktorů, jako je délka kroků, tempo, symetrie a dalších vlastností chůze.
- **Psaní na klávesnici**: Rozlišovat osoby lze i na základě rychlosti stisku kláves na klávesnici, intervalů mezi stisky a dalších faktorů.
- **Psaní perem**: V této kategorii se rozlišuje rychlost, tlak a tvar tahů. Nejčastěji se analyzuje osobní podpis. Podpis je stále stejné slovo, často s přidanými charakteristickými prvky, které se při normální psaní neobjevují.

Nejběžněji analyzovanou behaviorální charakteristikou je psaní perem. Behaviorální charakteristiky mohou být ovlivněny různými faktory. Nejsou tak časově stálé jako Anatomicko-fyziologické biometrické charakteristiky. Ovlivňující faktory zahrnují únavu, stres, zranění či snížení kognitivních schopností [29].

Existuje mnoho biometrických charakteristik, jež lze využít k identifikaci osoby. Velkou výhodou identifikace pomocí biometrických charakteristik je to, že identifikovaná osoba si nemusí nic pamatovat ani uchovávat, nemůže tedy o svou biometrickou identitu přijít (s výjimkou specifických zdravotních problémů např. amputace končetin, ztráta hlasu). Některé biometrické metody jsou vhodné pro využití pouze ve forenzní vědě a kriminalistice, některé zase pouze v medicíně. Vhodných charakteristik pro podepisování dokumentů je

jen několik, řadí se mezi ně mimo jiné otisk prstu a psaní perem. Nejčastěji využívanou charakteristikou pro podepisování dokumentů je psaní perem. Využívá se tzv. osobního podpisu, což je ručně psané a často stylizované vypsání jména nebo jiného identifikačního znaku (přezdívky, monogramu, erbu) osoby. Tento způsob potvrzování souhlasu s obsahem dokumentu a jeho pravosti je využíván po staletí a je lidem z naší kultury přirozený.

2.2 Analýza existujících řešení

Problematiku připojování biometrických podpisů k elektronickým dokumentům řeší již několik existujících komerčních produktů. Všechny mají své výhody a samozřejmě i nevýhody. V této kapitole jsou tato řešení porovnána. Je zde také odůvodněno, proč byla zvolena cesta vytvoření nové aplikace.

Všechna existující řešení splňují základní premisu, a to že zvládají zachytit podpis vytvořený tahem ruky a následně jej vložit do dokumentu. Zároveň ale mají určité vlastnosti, které je činí více či méně vhodné pro použití. V případě integrace do existující webové aplikace, běžící současně na mnoha stanicích, jsou tyto produkty nevhodné. Problémy existujících řešení je možné shrnout do třech následujících skupin:

- Drahá zařízení pro zachytávání podpisu
- Programy pro Microsoft Windows
- Kompletní závislost na cizí firmě

Drahá zařízení pro zachytávání podpisu

V této skupině jsou aplikace, jež zachytávají podpis pomocí tabletů nebo chytrých telefonů s operačními systémy Android, IOS nebo iPadOS. Jedná se například o tyto produkty: *MobbSign, Docuten*. Tato řešení neumí komunikovat se specializovanými podpisovými tablety. Integrace do existující webové aplikace by možná byla, a to buď přímo spouštěním celé existující webové aplikace na dotykových zařízeních, nebo posíláním dokumentu přes internet mezi počítačem a zařízením s dotykovou obrazovkou [23] [4]. Hlavním problémem a důvodem, proč tyto řešení nejsou vhodná pro použití v mém případě, je právě nutnost nákupu relativně drahého zařízení na každé místo, kde by probíhalo podepisování dokumentů.

Programy pro Microsoft Windows

Další skupinou jsou programy pro Microsoft Windows (*signoSign/2, Wacom signature*). Tyto řešení umí komunikovat se specializovanými podpisovými tablety. Oproti předchozí skupině snižují potřebnou vstupní investici na nákup zařízení pro zaznamenávání podpisů. Problém nastává při jejich integraci do existující webové aplikace, protože běží jako samostatné programy pod operačním systémem Microsoft Windows. Nejsou určené k připojování do jiných systémů, obzvláště těch, které běží v internetovém prohlížeči [33] [41].

Kompletní závislost na cizí firmě

Reprezentativním zástupcem této skupiny je například produkt *signoSign/Universal*. Tohoto produktu se již netýkají problémy drahého zařízení pro zachytávání podpisu ani omezení na určitý operační systém. Problematickou vlastností je způsob distribuce. Toto řešení

musí běžet na serveru dodavatele a přistupuje se k němu pomocí API, z čehož plyne kompletní závislost na cizí firmě. Například pokud by firma zkrachovala, přestane podepisování dokumentů fungovat. Dalším důvodem, proč nebylo toto řešení zvoleno, jsou relativně vysoké poplatky. Poplatek je za prvotní nákup licence k produktu, ale také i periodický, za každé zařízení, na kterém je produkt využíván [34]. Při plánování v rozmezí desítek let se z finančního hlediska více vyplatí investovat do vývoje vlastního řešení.

V této kapitole jsem se zaměřil na analýzu existujících komerčních řešení, které řeší problémy spojené se zachytáváním podpisů. Všechny tyto produkty mají určitou vlastnost, která je činí nevhodné pro použité v mém případě. Z tohoto důvodu jsem se rozhodl vytvořit novou aplikaci, která bude lépe vyhovovat potřebám mé firmy a nebude závislá na cizí firmě.

2.3 Dokumenty ve formátu PDF

PDF (portable document format) je široce používaný formát souborů, využívá se pro vytváření a zobrazování dokumentů nezávisle na aplikačním softwaru, hardwaru a operačním systému. Soubor ve formátu PDF může obsahovat text, obrázky, vektorovou grafiku, hypertextové odkazy, videa a další. Obsah může být zabezpečen šifrováním a nebo i digitálním podpisem. Tento formát zajistí, že se dokument zobrazí na všech platformách stejně [3]. PDF byl vyvinut firmou Adobe, nyní je otevřeným standardem spravovaným Mezinárodní organizací pro normalizaci (ISO) [14]. V této kapitole se budu zabývat zevrubným popisem historie a fungování tohoto formátu a práce s ním.

2.3.1 Historie

Historie formátu PDF sahá až do 70. let minulého století. V následujících několika odstavcích se zaměřím na vývoj PDF od jeho počátku až po dnešek.

U počátku PDF stál problém Johna E. Warnocka (spoluzakladatele Adobe) s tiskem na laserových tiskárnách, se kterými pracoval ve společnosti *Xerox PARC*. Tyto tiskárny používaly rozlišení 240 bodů na palec (dpi), zatímco počítačové monitory, se kterými v té době pracoval, používaly rozlišení 72 bodů na palec. Aby tento problém vyřešil, vyvinul Warnock se svým týmem (zvaným *Camelot*) jazyk pro popis stránek, který nazvali *PostScript*. *PostScript* původně nesl jméno *JaM*, tento název vznikl spojením prvních písmen křestních jmen dvou členů týmu Johna Gaffneyho a Martina Newella [42]. Tento systém nahradil dřívější manuální systém, který byl zdlouhavý a musel být pro každý nový typ písma, jeho velikost a nebo nové zařízení prováděn zvlášť a opakovaně. Použití PostScriptu umožnilo zobrazovat dokumenty na různých zařízeních jednotným způsobem, ale pouze za předpokladu stále poměrně zdlouhavých přípravných prací a znalosti specifikací zařízení, na kterých má být dokument zobrazován. Dalším problémem tohoto jazyka byla jeho složitost. Tento jazyk je totiž Turingovsky kompletní. Turingovská úplnost by se mohla zdát jako výhoda, to ale není v tomto případě pravda. Jazyk PostScript byl pro svůj účel příliš pomalý, měl bezpečnostní rizika a umožňoval i vznik nekonečných smyček.

Jako řešení těchto problémů s jazykem PostScript vznikl formát PDF. PDF staví na jazyku PostScript, zjednodušuje jej, omezuje jeho možnosti a tím zlepšuje jeho výkonnost. Přenositelnost mezi zařízeními byla také oproti PostScriptu vylepšena. Zpočátku nebyl nový formát plošně adoptován [6]. Raketový nárůst uživatelů tohoto formátu proběhl až po roce 1993, kdy Adobe umožnilo volné stažení programu Adobe Reader. S postupem času se stalo PDF standardem pro sdílení dokumentů po internetu.

V roce 2008 se z PDF stal otevřený formát standardizovaný Mezinárodním úřadem pro normalizaci (ISO) a dostal označení ISO 32000-1 [14]. Tento krok vedl k nezávislosti, v této době již plošně používaného formátu, na firmě Adobe a tím k ještě větší adopci. Pod správou Mezinárodního úřadu pro standardizaci nadále probíhá další vývoj tohoto formátu. Nejnovějším a důležitým milníkem PDF, bylo vydání nového standardu PDF 2.0 v roce 2020, ten dále rozšiřuje a vylepšuje formát pro přenositelné dokumenty [15].

2.3.2 Interní struktura

Každý dokument ve formátu PDF je reprezentován binárním souborem. Většina bytů souboru ve formátu PDF představuje ASCII znaky, jde jej tudíž zobrazit v běžném textovém editoru. Ovšem nemělo by se zapomínat na to, že PDF soubor může obsahovat i nějaké byty, které nereprezentují ASCII znaky. Příklad kódu jednoduchého PDF dokumentu naleznete v příloze A. Vnitřní struktura souboru je hierarchie objektů. Kořenem této hierarchie běžně bývá katalog stránek dokumentu, reprezentovaný pomocí objektu typu slovník. Každá stránka je popsána pomocí stránkového slovníku, což je objekt odkazující na objekty nacházející se na stránce. Následuje výpis vybraných základních objektů formátu PDF.

- **Řetězec** – Konečná posloupnost písmen.
- **Číslo** – Reprezentuje číselnou hodnotu. Existují dva typy: celá a reálná čísla.
- **Booleovská hodnota** – Vyjadřuje pravdivostní hodnotu. Její hodnota je reprezentována klíčovými slovy *true* a *false*.
- **Slovník** – Asociativní tabulka, složená z dvojic klíč–hodnota. Je základním stavebním blokem PDF souborů. Používá se k propojení ostatních objektů. Například k propojení řetězců s fonty, katalogu stránek s kořenem souboru a tak podobně.

Všechny tyto objekty se mohou vyskytovat jako záznamy v polích, slovnících a dalších objektech.

Vnitřní struktura dokumentu je složena ze 4 částí: hlavičky, těla, tabulky křížových odkazů a patičky.

Hlavička

Nachází se hned na začátku dokumentu. Obsahuje verzi použitého PDF a případně na dalším řádku může popisovat, jestli soubor obsahuje binární data. Příklad hlavičky pro PDF verze 1.7:

```
%PDF-1.7
```

Tělo

Obsahuje veškerý obsah dokumentu, který je zobrazován uživateli. Skládá se z posloupnosti objektů. Prvním objektem v posloupnosti je většinou katalog odkazující na objekt popisující stránky dokumentu. Následuje příklad těchto dvou objektů:

```
1 0 obj  % entry point
<<
  /Type /Catalog
  /Pages 2 0 R
```

Obrázek 2.2: Dělení pdf dokumentu

```
>>
endobj

2 0 obj
<<
  /Type /Pages
  /MediaBox [ 0 0 200 200 ]
  /Count 1
  /Kids [ 3 0 R ]
>>
endobj
```

Tabulka křížových odkazů

Začíná klíčovým slovem *xref*. Obsahuje umístění objektů v dokumentu. Umožňuje přistupování k jednotlivým částem dokumentu v náhodném pořadí, čímž umožňuje efektivnější práci s dokumentem. Příklad tabulky křížových odkazů velmi jednoduchého dokumentu:

```
xref
0 6
0000000000 65535 f
0000000010 00000 n
0000000079 00000 n
0000000173 00000 n
0000000301 00000 n
0000000380 00000 n
```

Patička

Je označena klíčovým slovem *trailer* a končí značkou označující konec souboru (*%%EOF)*. Nachází se na úplném konci souboru. Zobrazovače PDF čtou dokumenty od konce, patička je proto první věc, na kterou narazí. Patička obsahuje informace o tom, kde se nachází tabulka křížových odkazů a který objekt je kořenem dokumentu. Patička je tedy další část souboru umožňující efektivnější načítání dokumentu. Zde je příklad patičky dokumentu:

```
trailer
<<
  /Size 6
  /Root 1 0 R
>>
startxref
492
%%EOF
```

Veškeré příklady kódu v této kapitole je možné nalézt v příloze A i s kompletním kontextem. Kompletní popis formátu PDF, ze kterého bylo v této kapitole čerpáno, je v ISO standardu [15].

2.4 Práce s PDF dokumenty

Práce s PDF dokumenty je nedílnou náplní mnoha povolání. Z pohledu běžné neprogramátorské kancelářské práce je obrovský výběr produktů a nemalá část z nich má příjemné a přehledné uživatelské prostředí. Nejznámější z nich je program přímo od firmy, která stála u zrodu PDF, a to *Adobe Creator*. Dalšími příklady programů pro tvorbu a úpravu PDF dokumentů jsou *Sajda*, *PDFElement* a *Preview*. Z pohledu programátora je práce s PDF značně problematičtější než z pohledu běžného uživatele. Na problematiku práce s PDF z pohledu programátora se zaměřím v následujícím textu.

2.4.1 Problematika z pohledu programátora

Práce s PDF je z pohledu vývojáře softwaru relativně obtížná. Problémy při tvorbě softwarových řešení, které vytváří, upravují nebo jen zobrazují PDF dokumenty jsou mimo jiné způsobeny těmito důvody:

- **Komplexita**: PDF je velmi komplexní formát, který umožňuje práci s velkým množstvím typů dat, jako je například text, rastrové obrázky, vektorová grafika, anotace, formuláře, videa a další. Také podporuje různorodé funkce, jako jsou křížové odkazy, formuláře, digitální podpisy atd.
- **Nedostatečná standardizace**: Přestože specifikace PDF je standardizovaná Mezinárodní organizací pro normalizaci (ISO), existuje mnoho rozšíření a variací ve způsobu vytváření a enkódovaní dokumentů.
- **Omezené množství a nízká kvalita nástrojů**: Knihoven a rozšíření pro vytváření a editaci PDF není dostupné velké množství. Většina z nich je také značně omezena a dobře umí pracovat jen s úzkou částí funkcionality formátu. Často je také potřeba hlubšího pochopení formátu pro jejich plné využití.

- **Výpočetní náročnost**: Práce s velkými a nebo komplexními PDF dokumenty může být relativně hodně výpočetně náročná. Je nutné na to při vývoji softwarových řešení brát ohled, obzvlášť pokud mají běžet v prostředí s omezenými výpočetními prostředky, jako například na mobilních zařízeních nebo v internetových prohlížečích.

Práce s PDF skrývá mnoho obtíží, které by na první pohled programátora nemusely napadnout. Existují nástroje, které tuto práci ulehčují, mnoho z nich má, ale úzké zaměření a nebo jsou výpočetně velmi náročné. Porovnáním knihoven pro práci s PDF v jazyce JavaScript se zaobírám v části 2.4.2

2.4.2 JavaScriptové knihovny pro práci s PDF

Existuje několik různých knihoven, které umožňují zobrazovat, vytvářet nebo modifikovat PDF dokumenty. V této části se zaměřím výhradně na ty dostupné zadarmo. Placené knihovny pro práci s PDF zahrnují mimo jiné: *PDF.js Express*, *ComPDFKit*, *PSPDFKit* a *PDFTron*. Následuje seznam a krátký popis knihoven pro zobrazování, vytváření a editaci PDF dokumentů v prostředí jazyka JavaScript. Bude také zhodnocena vhodnost jejich využití pro vytvoření aplikace k biometrickému podepisování dokumentů.

- **PDFKit**: Knihovna určená primárně pro Node.js, ale je možné ji využívat i v klientském JavaScriptu pomocí nástroje *Webpack*. Má rozsáhlou dokumentaci a mnoho příkladů použití. Zaměřuje se primárně na vytváření nových dokumentů [27]. Editace existujících dokumentů není pomocí této knihovny v současné době možná, ani není jejími tvůrci plánované tuto možnost přidat [11]. Pro problematiku přidávání biometrických podpisů je nejdůležitější práce s existujícími dokumenty, z tohoto důvodu není PDFKit vhodnou knihovnou pro tento projekt.

- **pdfmake**: Stojí na knihovně PDFKit a vylepšuje ji. Oproti PDFKit zjednodušuje některé procesy a také je lépe adaptovaná pro běh přímo v klientském internetovém prohlížeči [28]. Je ale stále zatěžována nedostatky knihovny PDFKit, a to primárně jejím zaměřením na generování nových pdf dokumentů, z tohoto důvodu není v mé problematice nápomocná.

- **jsPDF**: Profesionálně vyvíjená knihovna s otevřeným zdrojovým kódem. Vývoj vede firma Parallax. Má velmi kvalitní dokumentaci a mnoho uživatelů. Je jedna z nejoblíbenějších knihoven pro práci s PDF (dle počtu hvězd GitHub repozitáře). Umožňuje pouze generování nových dokumentů.

- **diegomura/react-pdf**: Knihovna uzpůsobená pro React. Umožňuje vytváření nových dokumentů za použití interaktivních komponentů. Vhodná pro softwarové produkty, umožňující vytváření dokumentů pomocí manipulace HTML elementů. Tato knihovna neumožňuje editaci existujících dokumentů [31].

- **pdf-lib**: Knihovna psaná v čistém JavaScriptu bez dalších závislostí, díky tomu funguje v jakémkoliv JavaScriptovém prostředí, včetně prohlížečů, Node, Deno i Reactu. Je unikátní v tom, že umožňuje editaci již existujících dokumentů. Jejím použitím je možné v JavaScriptovém prostředí přidávat do existujících dokumentů nové stránky, rozdělovat existující stránky do více nových stránek, vyplňovat formuláře, či dokonce přidávat nové prvky formulářů. Avšak primárním zaměřením této knihovny je vytváření nových dokumentů a funkčnost editace dokumentů je stále značně omezená,

například přidávání nových nevyplněných podpisových polí není možné [26]. Tuto knihovnu v tomto projektu používám k editaci existujících podpisových polí, specificky ke vkládání dat podpisu do dokumentu.

- **PDF.js**: Dle metriky hvězdiček GitHub repozitáře se jedná o nejoblíbenější JavaScriptovou knihovnu pro práci s PDF. Existuje okolo ní relativně velká komunita vývojářů, díky čemu je možné najít velké množství příkladů použití a návodů. Je vyvinutá společností Mozilla, má otevřený zdrojový kód a nemá žádné další závislosti. Umožňuje pouze zobrazování dokumentů v prohlížeči bez jejich editace či vytváření dokumentů nových. Na rozdíl od jiných podobných řešení umožňuje modifikaci zobrazovaní dokumentu [24]. Této knihovny jsem v tomto projektu využil pro zobrazování podepisovaného dokumentu a jednotlivých podpisových polí.

V této kapitole jsem se zabýval několika vybranými knihovnami pro zobrazování, vytváření a editaci dokumentů ve formátu PDF. Existuje relativně široký výběr knihoven vhodných pro tvorbu nových dokumentů a zobrazení existujících, přičemž některé z nich mají velkou komunitu vývojářů. Nabídka knihoven pro editaci již existujících dokumentů v prohlížeči je výrazně menší. Pokud jde o komplexnější úpravu dokumentů, jako je například vkládání obrázků, anotací, nebo formulářů existuje jen jedna knihovna (pdf-lib) a i její možnosti jsou značně omezené.

Obecně lze říci, že kvalita knihoven pro vytváření a zobrazování PDF je dobrá a existuje dostatek podkladů pro práci s nimi. Existuje několik komerčních produktů i několik nekomerčních knihoven, přičemž okolo knihoven s otevřeným zdrojovým kódem je relativně aktivní komunita vývojářů zapojena do jejich dalšího rozvoje. Knihoven pro úpravu již existujících dokumentů je velmi málo, během mého průzkumu jsem našel pouze jednu a její schopnosti jsou značně omezené.

2.5 Principy tvorby webových aplikací

Webové aplikace jsou aplikace, ke kterým je přistupováno přes internet pomocí webového prohlížeče. Za poslední dekádu se internet transformoval z repozitáře stránek obsahující statické převážně vědecké informace na platformu používanou k vývoji a nasazování rozsáhlých interaktivních aplikací [16]. Webové aplikace se stávají stále více oblíbené, protože jsou nezávislé na prostředí. To znamená, že umožňují použití kýmkoliv, kdo má přístup k internetu a zařízení, které je schopné spustit internetový prohlížeč. Při vývoji webových aplikací nemusí vývojář brát v potaz operační systém uživatelů, ani vyvíjet stejnou aplikaci opakovaně s různými technologiemi, aby rozšířil okruh lidí, kteří ji mohou používat, jak je obvyklé například u mobilních aplikací [9]. V této kapitole se zaměřím na principy jejich vývoje. Shrnu dvě základní technologie používané k jejich vývoji: **JavaScript (JS)** a **WebAssembly (WASM)**. Poté se více zaměřím na frameworky používané v JavaScriptovém prostředí.

2.5.1 JavaScript

JavaScript je vysokoúrovňový programovací jazyk, používaný především pro vývoj webových stránek a aplikací. Většinou se používá ve spojení s HTML a CSS. HTML a CSS zajišťují strukturu a styl webu, JavaScript umožňuje přidávat interaktivitu. Tento jazyk není omezen na určitý operační systém a podporuje událostmi řízený, funkcionální i imperativní styl programování [35]. JavaScript byl vyvinut v roce 1995 Brendanen Eichem pro

společnost Netscape, která spravovala v té době nejpoužívanější internetový prohlížeč pojmenovaný Netscape Navigator [32]. Brendan Eich vyvinul prvotní prototyp pojmenovaný Mocha za pouhých 10 dní. Podnětem k jeho vývoji byl konkurenční boj mezi Netscape s jejich prohlížečem Netscape Navigator a Microsoftem s jejich prohlížečem Internet Explorer. Netscape chtěl přijít s novinkou, která by primárně umožnila urychlit práci s formuláři na webových stránkách, před JavaScriptem se totiž musel každý formulář validovat na serveru, což v době vytáčeného připojení k internetu mohlo trvat nepříjemně dlouho. Netscape ve spojení se Sun Microsystems proto přišli se skriptovacím jazykem běžícím přímo v prohlížeči, umožňujícím validaci formulářů u klienta. Tento nový jazyk byl nejprve pojmenován LiveScript a později přejmenován na JavaScript. Jméno JavaScript bylo zvoleno hlavně jako marketingový tah. Firma Sun Microsystems spolupracující na vývoji JavaScriptu dříve vyvinula i jazyk Java. Zvolením jména podobnému již existujícímu jazyku chtěli přilákat existující vývojáře k adaptaci tohoto nového jazyka. Reakcí Microsoftu na tento úspěšný tah ze strany Netscape, bylo vydání vlastní implementace skriptovacího jazyka běžícího v prohlížeči, který pojmenovali JScript. Syntaxe JScriptu byla velmi podobná JavaScriptu, nebyla však zcela stejná a programy psané pro Netscape Navigator nefungovaly v prohlížeči Internet Explorer a naopak. Aby se ukončil tento problémový stav, rozhodla se firma Netscape jazyk JavaScript standardizovat, za tímto účelem kontaktovala v roce 1997 Evropskou asociaci výrobců počítačů (anglicky European Computer Manufacturers Association zkráceně ECMA). ECMA vytvořilo standard ECMA-262, který definoval nový skriptovací jazyk pojmenovaný ECMAScript, jehož implementace se začala používat ve všech hlavních internetových prohlížečích, včetně Internet Exploreru. Vývoj jazyka ECMAScript, jež je běžně nazýván jeho starým jménem JavaScript pokračuje dodnes [17].

JavaScript je dle průzkumu W3Techs používán na 97,6 % webových stránek [40], i přes to má v podvědomí mnoha vývojářů spíše negativní podtón. Problematické vlastnosti JavaScriptu vyplývají už z podmínek, za jakých byl vytvořen a to hlavně z faktu, že základ tohoto jazyka byl vyvinut za pouhých 10 dní. Některé operace se chovají zvláštně, neočekávaně nebo nekonzistentně. Problematická je také naprostá absence typové bezpečnosti a přílišná závislost na knihovnách. Dle statistiky vytvořené z více než 45000 aktivních GitHub repozitářů vyplývá, že průměrný projekt používající JavaScript má 10 přímých a neuvěřitelných 683 nepřímých závislostí. Takto velký počet závislostí způsobuje nízké zabezpečení stránek používající JavaScript, protože je velká pravděpodobnost, že alespoň v jedné ze závislostí existuje zranitelnost potenciálně využitelná útočníkem [19]. I přes všechny tyto problémy zůstává JavaScript hlavním jazykem pro vývoj webových stránek a aplikací. Díky své všudypřítomnosti, univerzálnosti a dobré manipulaci s DOM (objektově orientovaná reprezentace HTML dokumentu) je vývoj v JavaScriptovém prostředí rychlejší než např. s pomocí WebAssembly. Pro efektivní vývoj kvalitní webové aplikace nebo stránky je stěžejní dobrý výběr frameworku a pomocných knihoven, jejich přehledu a porovnání se věnuji v kapitole 2.5.3.

2.5.2 WebAssembly

WebAssembly (zkráceně WASM) je otevřený webový standard, který definuje binarní formát pro přenositelný strojový kód spustitelný v internetovém prohlížeči. Na rozdíl od JavaScriptu, který je vysokoúrovňovým skriptovacím jazykem, pracuje WebAssembly na nižší úrovni, bližší strojovému kódu. WebAssembly byl vydán v roce 2017 skupinou spojující giganty informačních technologií a to W3C, Mozzila, Microsoft a Google. Díky spojení všech hlavních dodavatelů prohlížečů je WebAssembly podporováno na všech běžně používaných

prohlížečích. WASM je podporován v prohlížečích Google Chrome, Microsoft Edge, Mozilla Firefox, Safari i Opera. Tento standard má jeden hlavní cíl a to aby jeho kód byl parsován i prováděn výrazně rychleji než JavaScript, toho by mělo být dosaženo bez ohrožení kompatibility a bezpečnosti [18]. Výsledný WebAssembly kód může být kompilován z jiných běžně používaných jazyků, jako jsou C, C++, C#, Rust a dalších, díky čemu umožňuje mnoha vývojářům začít vyvíjet webové aplikace bez nutnosti učit se nový jazyk. I přes tuto vlastnost není WASM příliš rozšířený. Z dotazníku z roku 2022 vyšlo najevo, že WebAssembly někdy použilo pouze 17,5 % dotazovaných vývojářů [37]. Pomocí WebAssembly není v současné době možné přistupovat k DOM a manipulovat s ním [22], proto zatím není přímou konkurencí JavaScriptu. Z vlastností WebAssembly vyplývá, že použití této technologie je spíše vhodné pro výpočetně náročné aplikace, jako jsou hry, aplikace zpracovávající obraz či video, CAD aplikace a podobné. Velkou výhodou WASM je bezpečnost. Oproti JavaScriptu má WebAssembly menší komunitu vývojářů, což dělá vývoj s pomocí WASM náročnější, z důvodu menšího počtu návodů a pomocných nástrojů.

2.5.3 JavaScriptové frameworky

Při vytváření nového JavaScriptového projektu je jedním z prvních kroků volba frameworku. Správná volba frameworku je velmi důležitá, framework totiž určuje architekturu i styl programování a může silně ovlivňovat délku vývoje. V dnešní době je k dispozici velké množství frameworků a každým dnem přibývají nové. Na obrázku 2.3 je zobrazena oblíbenost vybraných frameworků, vycházející z průzkumu společnosti StackOverflow. V této části se zaměřím na nejvíce používané JavaScriptové frameworky a popíšu jejich výhody, nevýhody a vlastnosti, díky nimž jsou unikátní.

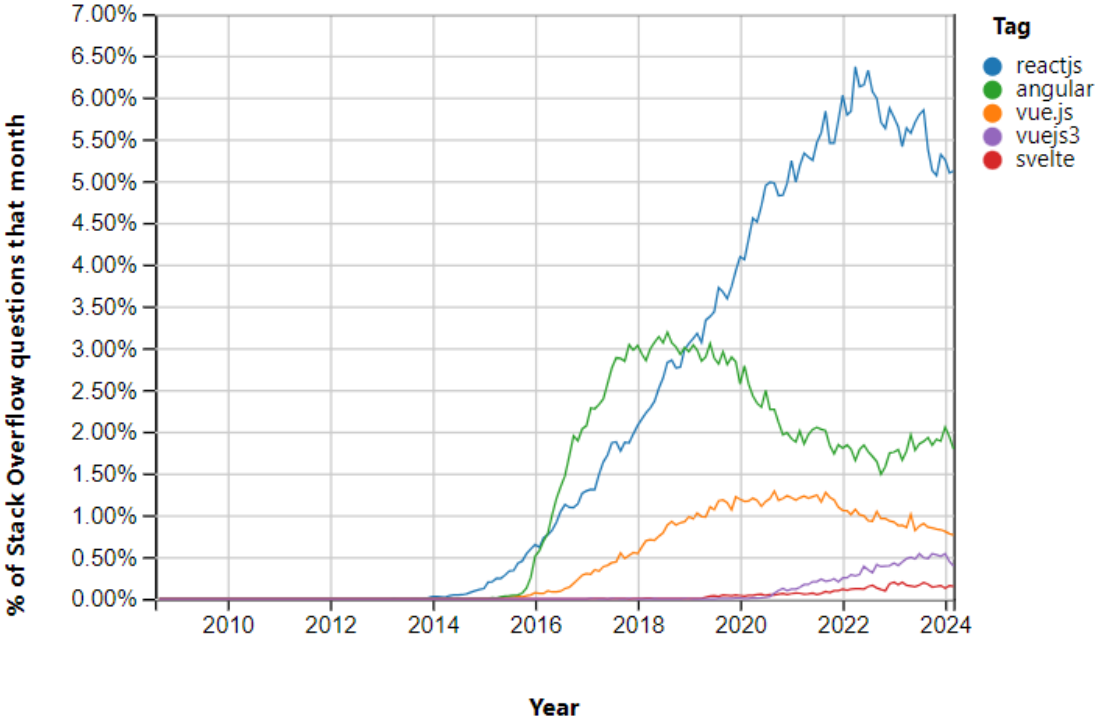
React

React je framework s otevřeným zdrojovým kódem vyvíjený společností Meta. Pro veřejnost byl vydán v březnu 2013 a prvotní ohlasy na něj byly spíše negativní. React totiž přišel s mnoha novými paradigmaty a změnil dosavadní status quo v JavaScriptovém vývoji. Postupem času byl ale React vylepšován a stále více používán [8]. Dnes se bezkonkurenčně jedná o nejpopulárnější JavaScriptový framework. Obrovská oblíbenost tohoto frameworku je jeho předností, díky ní totiž existuje velké množství návodů a nástrojů třetích stran. Návody třetích stran jsou pro tento framework důležité, protože oficiální dokumentace není moc kvalitní, avšak Meta na jejím vylepšení usilovně pracuje. Struktura React kódu je založena na komponentách, což jsou nezávislé, znovupoužitelné části kódu. Každá komponenta může mít vzhled i chování zcela nezávislé na zbytku projektu. Díky implementaci virtuálního DOM je React efektivnější a rychlejší než starší frameworky. Virtuální DOM je odlehčená JavaScriptová reprezentace stromové struktury HTML dokumentu, která umožňuje rychlejší úpravy než skutečný DOM. Při obnovení stránky se provede porovnání virtuálního a skutečného DOM, zaznamenají se rozdíly a ve skutečné struktuře dokumentu jsou poté překresleny pouze změněné části. V Reactu se také používá JSX, což je rozšíření syntaxe JavaScriptu umožňující vkládat HTML značky přímo do JavaScriptového kódu. Následuje krátká ukázka syntaxe s použitím JSX:

```
const element = <h1>JSX</h1>
```

a nyní kód se stejným významem, bez použití JSX:

```
const element = React.createElement('h1', null, 'JSX');
```



Obrázek 2.3: Počet otázek položených na stránce StackOverflow souvisejích s JS frameworky v postupu času. (Převzato z literatury [36])

Z ukázek lze vidět, že použití JSX umožňuje vytváření čitelnějšího a stručnějšího kódu. Nevýhodou Reactu je, že často zahrnuje psaní velkého množství šablonového kódu (tzv. boilerplate code). Velké množství šablonového a často nepřehledného kódu vzniká zejména pro obsluhu stavů. Výsledná rozsáhlost kódu může způsobovat těžkopádnost vývoje a údržby. Vysoká popularita Reactu vedla k vývoji velkého množství podpůrných knihoven a nástrojů, což na jednu stranu přináší hodně možností, jak v Reactu pracovat, ale na druhou stranu způsobuje horší orientaci v jeho ekosystému a fragmentaci komunity.

Angular

Angular je framework a platforma pro strukturovaný vývoj jednostránkových webových aplikací založený na TypeScriptu. Vývoj tohoto frameworku vede společnost Google, ale i Angular má otevřený zdrojový kód, takže kdokoliv se může zúčastnit jeho vývoje. Angular je napsaný v TypeScriptu [1], což umožňuje, aby byly chyby odhaleny již při kompilaci a ne až při běhu aplikace. TypeScript je programovací jazyk postavený na JavaScriptu, přidává do něj typy, rozhraní a mnoho dalších funkcí, ale nakonec se zkompiluje do prostého jazyka JavaScript, který lze spustit ve všech běžných prohlížečích [38]. Aplikace psané v Angularu používají principy modulárního programování, což způsobuje, že jednotlivé části aplikace jsou rozděleny do logicky nezávislých a zaměnitelných celků. Ovšem některé menší aplikace jsou složené pouze z jednoho kořenového modulu. V Angularu je narozdíl od Reactu oddělena logika od vzhledu. Jednou z klíčových vlastností je obousměrná vazba dat, která umožňuje, aby se změny v modelu automaticky promítly do zobrazení a naopak. An-

gular také poskytuje další výkonné funkce, jako je zřejmé předávání závislostí (dependency injection), routování, zabudovaný HTTP klient pro práci s API voláními a další [1]. Komunita Angularu je jedna z největších komunit spojených s JavaScriptovým frameworkem (dle průzkumu StackOverflow druhá největší viz obrázek 2.3), tato komunita přispívá vytvářením pomocných knihoven, pluginů a dalších pomocných materiálů. Oficiální dokumentace je velice kvalitní a dobře zorganizovaná. Problém při vytváření aplikací v Angularu je optimalizace pro webové vyhledávače (SEO), protože aplikace vytvořené v tomto frameworku jsou jednostránkové. Angular aplikace mohou trpět na horší výkon a pomalejší načítání, nejvíce problematické bývá úplně první načtení stránky. Dalším problémem tohoto frameworku je poměrně pozvolná křivka učení, Angular má totiž mnoho komplexních vlastností, konceptů a zvyklostí, i díky této komplexitě je ale velmi vhodný pro vývoj rozsáhlých, dlouho udržovaných a rozvíjených systémů.

Vue.js

Vue.js je progresivní JavaScriptový framework, který vyvinul Evan You a první verzi vydal v roce 2014. Vue.js slouží k vytváření uživatelských rozhraní a jednostránkových aplikací. To, že je Vue progresivní framework znamená, že je možné jej postupně přidávat do již existující aplikace, může být použit jen na jeden prvek uživatelského rozhraní, ale i na celou aplikaci úplně od začátku. Touto vlastností se liší od ostatních frameworků zmiňovaných v této práci, u kterých se počítá s tím, že budou použity od samotného začátku vývoje aplikace. Progresivita v popisu tohoto frameworku také odkazuje na to, že je možné ho začít používat jen s velmi nízkým pochopením jeho funkcí a s postupem času využívat více jeho částí a schopností, díky tomu je velmi dobře přístupný i pro méně zkušené vývojáře. Vývoj aplikací ve frameworku Vue.js se řadí k deklarativnímu programování. Stejně jako React i Vue.js používá virtuální DOM, navíc má implementovaný kompilátor, který kompiluje šablony uživatelského rozhraní na vykreslovací funkce virtuálního DOM, díky tomuto kroku mají aplikace v něm napsané lepší výkon než aplikace psané ve frameworku React. Vue.js se velmi dobře kombinuje s dalšími frameworky a knihovnami [39]. Nevýhodou pro anglicky mluvící vývojáře může být fakt, že tvůrce i velká část komunity tohoto frameworku pochází z Číny a to způsobuje, že některé materiály a diskuze jsou dostupné pouze v čínštině. Velikost komunity okolo Vue.js je srovnatelná s velikostí komunity frameworku Angular, takže i přes částečný problém s jazykovou bariérou existuje dostatečné množství pomocných materiálů třetích stran. Oficiální dokumentace je velmi rozsáhlá, detailní a dobře zorganizovaná.

Svelte

Svelte je relativně nový inovativní framework, umožňující vytváření komponent a celých webových aplikací. Svelte byl vytvořen Richem Harrisem a vydán v roce 2016. Při jeho vývoji se Rich Harris díval na současný stav JavaScriptových frameworků a snažil se řešit jejich problémy a nedostatky inovativními způsoby. Svelte je popisován spíše jako kompilátor než jako framework, protože používá kompilátor ke kompilaci kódu napsaného v jeho specifické syntaxi, která rozšiřuje JavaScript a HTML do vysoce efektivního imperativního kódu, který přímo aktualizuje DOM, díky tomu má velice dobré výkonnostní charakteristiky. Aplikace psané v tradičních frameworcích zmiňovaných v této práci posílají klientovi jak kód specifický pro funkcionalitu aplikace, tak i kód samotného frameworku a až klient interpretuje kód stránky za pomoci kódu frameworku, Svelte je v tomto unikátní a už při kompilaci vytvoří jediný soubor v čistém JavaScriptu, který je pak přenášen klientovi, už

bez specifik Svelte. Díky tomuto kroku jsou finální balíčky aplikace velmi malé v porovnání s klasickými frameworky a mohou, tak být spolehlivě a rychle přenášeny i přes pomalé internetové připojení. Svelte nepoužívá virtuální DOM, Rich Harris argumentuje, že virtuální DOM přináší zbytečné zpomalení. Svelte provede logiku, kterou by aplikace psané v tradičních frameworcích prováděli za běhu v prohlížeči s pomocí virtuálního DOM, již při kompilaci. V tomto frameworku je minimalizováno množství šablonového (boilerplate) kódu, díky tomu je vývoj aplikací v něm rychlejší než v konkurenčních frameworcích [12]. Okolo Svelte se rychle vytvořila početná a aktivní komunita vývojářů, ale v porovnání s ostatními frameworky zmiňovanými v této práci je stále velmi malá. Oficiální dokumentace je kvalitní a obsahuje i dobré návody pro úplné začátečníky. Kvůli relativně nízkému věku a malé komunitě má tento framework méně pomocných nástrojů a horší integraci s IDE než ostatní frameworky.

Závěrem lze říci, že porovnání frameworků React, Angular, Vue.js a Svelte vedlo k zjištění, že každý má své unikátní a specifické charakteristiky. React je díky své flexibilitě, rozsáhlému ekosystému a silné komunitě nejoblíbenějším frontendovým frameworkem. Angular nabízí komplexní řešení s funkcemi, jako je obousměrná vazba dat a zřejmé předávání závislostí, díky tomu je vhodný i pro rozsáhlé projekty, vyžadující robustní strukturu a dobrou škálovatelnost. Vue.js vyniká díky své jednoduchosti, snadné integraci a možností kombinace s jinými frontedovými frameworky, tyto vlastnosti z něj činí oblíbenou volbu jak pro začátečníky, tak pro zkušené vývojáře. Svelte, nejnovější z těchto frameworků, přichází s inovativním přístupem se svým frameworkem založeným na kompilátoru, má bezkonkurenčně nejlepší výkon a osvěžující vývojářský zážitek.

Volba mezi těmito frameworky nakonec závisí na požadavcích vývojáře, jeho schopnostech, velikosti týmu, rozsáhlosti projektu a dalších. Nelze jednoznačně rozhodnout, který z frameworků je obecně nejlepší. Pro projekt, jímž se zabývá praktická část této práce, jsem zvolil framework React hlavně z toho důvodu, že informační systém, do kterého se bude výsledné řešení integrovat je také vytvořen s pomocí tohoto frameworku.

Kapitola 3

Specifikace a analýza požadavků na aplikaci

V této kapitole specifikuji základní cíle a body, která má výsledná aplikace plnit. V podkapitole 3.1 budou vyjmenovány požadavky, které byly specifikovány na základě diskuze s vedením firmy SeaComp, s. r. o., tyto požadavky budou poté v podkapitole 3.2 analyzovány.

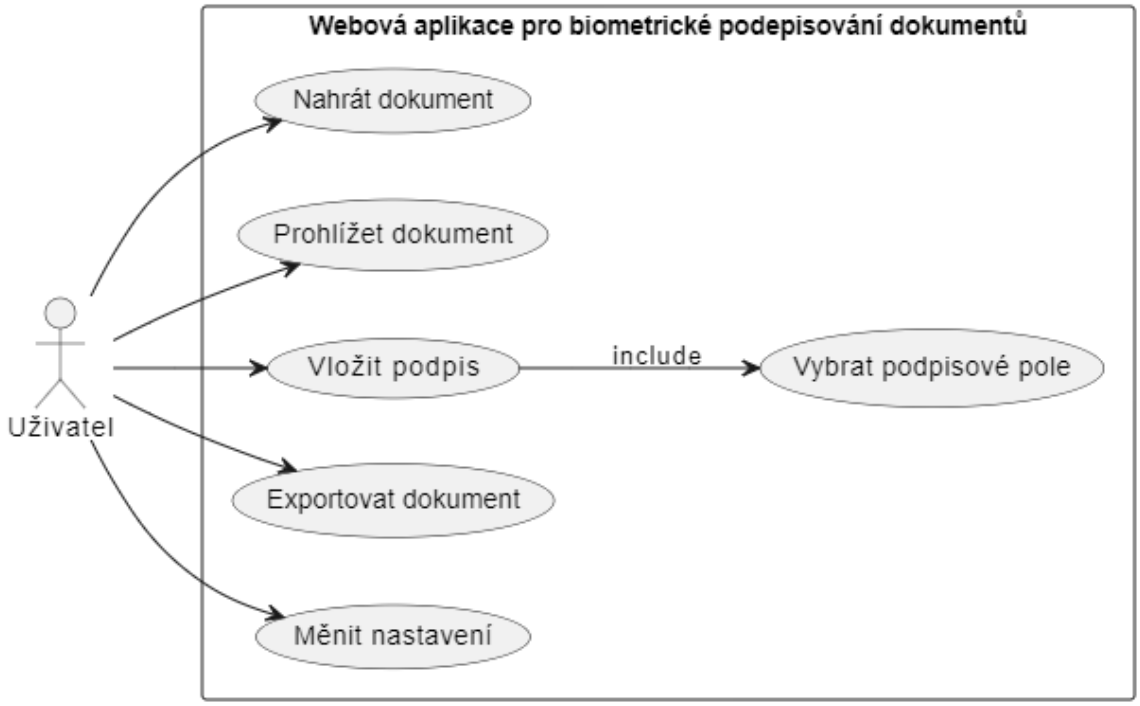
3.1 Specifikace požadavků

V následující části jsou neformálně popsány požadavky na výslednou aplikaci. Požadavky byly specifikovány na základě potřeb firmy SeaComp, s. r. o., a vznikly na základě diskuze jejích zaměstnanců.

Hlavním cílem výsledné aplikace je schopnost podepisovat dokumenty biometrickým podpisem, přesněji podpisem vytvořeným tahem ruky. Aplikace má běžet ve webovém prohlížeči, aby bylo možné ji integrovat do existující webové aplikace ambulančního informačního systému. Podpis, který má být vložen do výsledného dokumentu, je nutné zachytit přímo v aplikaci, protože aplikace bude používaná např. při příjmu nových pacientů do ambulance. Je také požadováno, aby aplikace uměla komunikovat a zachytávat podpisy z podpisových tabletů značky Signotec, kterých již firma několik nakoupila a byly mi poskytnuty za účelem vývoje a testování. Editace dokumentu nad rámec vložení zachyceného podpisu do podpisového pole není požadována. Do prvotních návrhů aplikace jsem přidal i možnost vytváření nových podpisových polí, po diskuzi se zadavatelem bylo stanoveno, že tato funkcionalita není vyžadována. Po zachycení podpisu a jeho vložení do dokumentu musí být možné takto upravený dokument vyexportovat. Z těchto požadavků jsem vytvořil diagram případů užití, který je na obrázku 3.1.

3.2 Analýza požadavků

V následující části budou podrobněji analyzovány požadavky, a přesněji vymezena podoba aplikace. Požadavky, které budou v této části analyzovány, byly specifikovány v podkapitole 3.1.



Obrázek 3.1: Diagram případů užití aplikace

Forma aplikace

Musí se jednat o webovou aplikaci, tedy aplikaci běžící ve webovém prohlížeči. Je také žádoucí, aby byla vyvíjena s pomocí frameworku React, protože i ostatní webové aplikace firmy SeaComp, jsou vytvářeny s pomocí tohoto frameworku. Uživatelské rozhraní by mělo být dostatečně přívětivé a přehledné, aby výslednou aplikaci byli schopni využívat uživatelé bez speciálního vzdělání a se žádným nebo jen minimálním školením. Mělo by se dbát i na zabezpečení aplikace, protože bude pracovat s osobními údaji.

Nahrání dokumentu

Do aplikace bude možné nahrát pouze dokumenty ve formátu PDF. V rámci informačního systému ambulantní péče, do kterého bude tato aplikace později integrována, jsou veškeré dokumenty právě ve formátu PDF, proto není potřeba žádné jiné formáty podporovat.

Zachycení podpisu

Aplikace musí umět zachytit nový podpis. Primární požadavek je, aby bylo možné podpis zachytávat za pomoci podpisového tabletu značky SignoTec. Jako záložní možnost by mělo být možné podpis vytvořit i pomocí počítačové myši. Výsledný podpis by měl být uložen jak v podobě biometrických dat, tak i v podobě obrázku. Obrázek podpisu bude zobrazován ve výsledném dokumentu, měl by proto být graficky kvalitní. Rozlišení obrázku podpisu musí být dostatečně velké a je žádoucí, aby v něm byly reprezentovány i zachycené biometrické charakteristiky, jako je rychlost tahu či přítlak pera k tabletu.

Vložení podpisu do dokumentu

Musí být možné sloučit zachycený podpis s nahraným dokumentem. Pro případ více podpisových polí v dokumentu, musí být možné zvolit specifické podpisové pole, do kterého má být podpis vložen. Do vybraného podpisového pole bude vložen obrázek podpisu. Biometrická data zachycená při vytváření podpisu budou uložena do metadat dokumentu, není potřeba, aby byly přímo graficky znázorněny.

Export dokumentu

Z aplikace musí být možné vyexportovat dokument ve formátu PDF. Ve vyexportovaném dokumentu, musí být uloženy všechny zachycené podpisy, na jim určených místech a v metadatech dokumentu musí být uložena zachycená biometrická data (pokud nějaká byla zachycena).

V této kapitole byly podrobněji zkoumány požadavky specifikované v podkapitole 3.1. Na základě této analýzy bude v kapitole 4 proveden návrh uživatelského rozhraní výsledné aplikace a její vnitřní architektury.

Kapitola 4

Návrh aplikace

Dobrý návrh uživatelského rozhraní a vnitřní architektury je zásadní pro vytvoření kvalitní aplikace. Dobře zpracovaný prvotní návrh urychlí pozdější vývoj aplikace. Návrh je jednoduché upravovat a měnit, zatímco dělat změny v již implementovaných částech aplikace je časově náročné, z tohoto důvodu je důležité, aby vytvoření návrhu bylo jeden z prvních kroků při vytváření nové aplikace. Při navrhování je nutné vycházet ze specifikace a analýzy požadavků. Analýzu a specifikaci požadavků jsem provedl v kapitole 3.

První část (4.1) této kapitoly se zabývá návrhem uživatelského rozhraní (UI), což zahrnuje design a umístění ovládacích prvků, grafické rozložení celé aplikace a způsob, jakým bude uživatel s aplikací interagovat.

V druhé části (4.2) je rozebrán návrh architektury aplikace. Zaměřuji se v ní na strukturu, vnitřní rozložení, použité technologie a podobně. Tato část návrhu ovlivňuje funkcionalitu, výkon a bezpečnost aplikace.

Celkový cíl této kapitoly je vytvořit komplexní návrh zohledňující požadavky, na základě kterého bude možné implementovat výslednou aplikaci.

4.1 Návrh uživatelského rozhraní

Návrh uživatelského rozhraní je klíčovým krokem při vývoji aplikace. Uživatelské rozhraní (UI) je místem, kde uživatelé interagují a komunikují s aplikací, je proto zásadní pro pozitivní uživatelský zážitek a úspěch aplikace. V této kapitole se zaměřím na vytvoření návrhu uživatelského rozhraní na základě požadavků na aplikaci, jež byly specifikovány a analyzovány v kapitole 3.

Z analýzy požadavků vyplývá, že je nezbytné zajistit, aby uživatelské rozhraní bylo intuitivní a snadno ovladatelné. Je to důležité, protože s aplikací budou pracovat uživatelé bez speciálního školení. Aplikaci jsem se rozhodl rozdělit na dvě hlavní zóny: **ovládací panel** a **prohlížeč dokumentu**. Ovládací panel je na horní straně okna a zabírá celou šířku aplikace, podrobněji je ovládací panel popsán v části 4.1.1. Ve zbytku okna je zobrazen dokument samotný a interakcí s ním se otevírá **podpisový editor**, tyto části jsou blíže popsány v 4.1.2 a 4.1.3. Barevně jsem se rozhodl navrhnout aplikaci spíše neutrální, většina prvků bude mít bílou barvu, jediný prvek, který bude nějak obarven bude hlavní menu. V následujícím textu se jim budu věnovat podrobněji.

4.1.1 Ovládací panel

Ovládací panel má formu pruhu zabírajícího celou šířku horní části obrazovky. Jeho výška by neměla přesáhnout 10 % výšky okna. Z ovládacího panelu musí být možné pracovat s těmito funkcemi:

- Nahrání dokumentu
- Změna zobrazované stránky dokumentu
- Export dokumentu
- Přístup do nastavení aplikace

Počet funkcí, které potřebují své vlastní tlačítko, je relativně malý. Vzhledem k tomu, že se veškerá tlačítka pohodlně vejdou vedle sebe do šířky okna, není potřeba mít žádné další rozbalovací nabídky ani skrývat žádné funkce. Tlačítka jednotlivých funkcí mají jednoduchý design. Pro navigaci v dokumentu zde bude textové pole, obsahující číslo aktuálně otevřené stránky, toto číslo bude možné přímo upravovat. Vedle aktuálního čísla stránky a celkového počtu stránek budou dvě tlačítka pro navigaci na předchozí a následující stránku. Toto menu je jedno z vhodných míst, kde je možné aplikovat nějaké barevné schéma. Jako primární barvu jsem se rozhodl použít odstín modré barvy, který používá firma SeaComp. Tuto modrou barvu jsem se rozhodl akcentovat oranžovou barvou, taktéž specifickou pro tuto firmu. Prvotní návrh tohoto menu, bez barevného schématu je na obrázku 4.1.

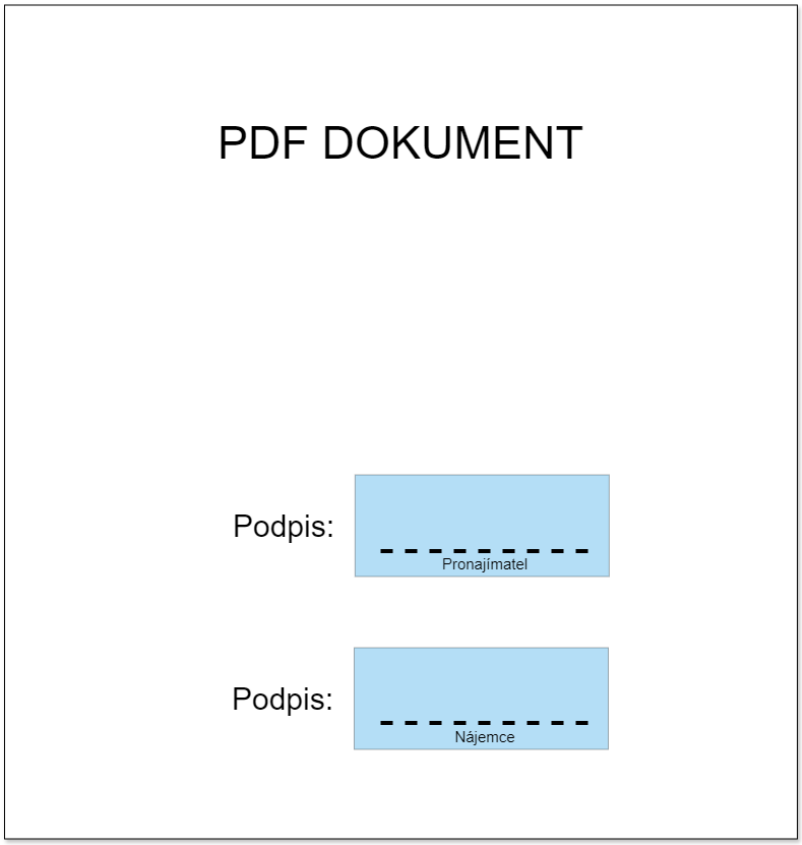
Obrázek 4.1: Grafický návrh ovládacího panelu

4.1.2 Prohlížeč dokumentu

Jedná se o hlavní prvek aplikace, který zabírá většinu prostoru obrazovky. V této části je zobrazován dokument a je to také část aplikace, ze které se otevírá podpisový editor. K navigaci v dokumentu slouží ovládací prvky v ovládacím panelu, popsané v části 4.1.1. V prohlížeči dokumentu se provádí výběr podpisového pole k podepsání, za tímto účelem jsou podpisová pole v dokumentu barevně zvýrazněna. Zvýraznění je provedeno pomocí obdélníku se stejnými rozměry a pozicí jako má dané podpisové pole, obdélník má modrou barvu a je průhledný, tak aby bylo, případně vidět i text, který je podpisovým polem překrytý. Na kterékoliv zvýrazněné podpisové pole je možné kliknout, což otevře podpisový editor, který je popsaný v části 4.1.3. Pokud již byl v podpisovém editoru vytvořen podpis pro dané podpisové pole, bude tento podpis v podpisovém poli zobrazen, ale stále bude možné na podpisové pole kliknout a vložit nový podpis. Grafický návrh prohlížeče dokumentu je na obrázku 4.2.

4.1.3 Editor podpisů

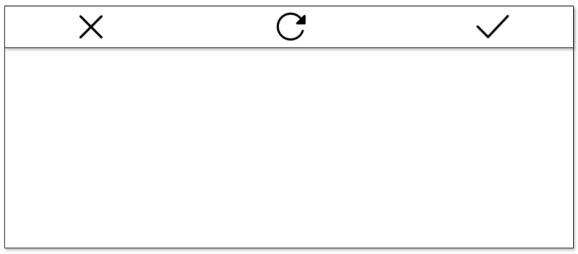
Editor podpisů slouží k vytváření nových podpisů. Editor podpisů nemá svou vlastní stránku, ale je ve vyskakovacím okně překrývajícím zbytek aplikace. Hlavní částí editoru



Obrázek 4.2: Grafický návrh prohlížeče dokumentu

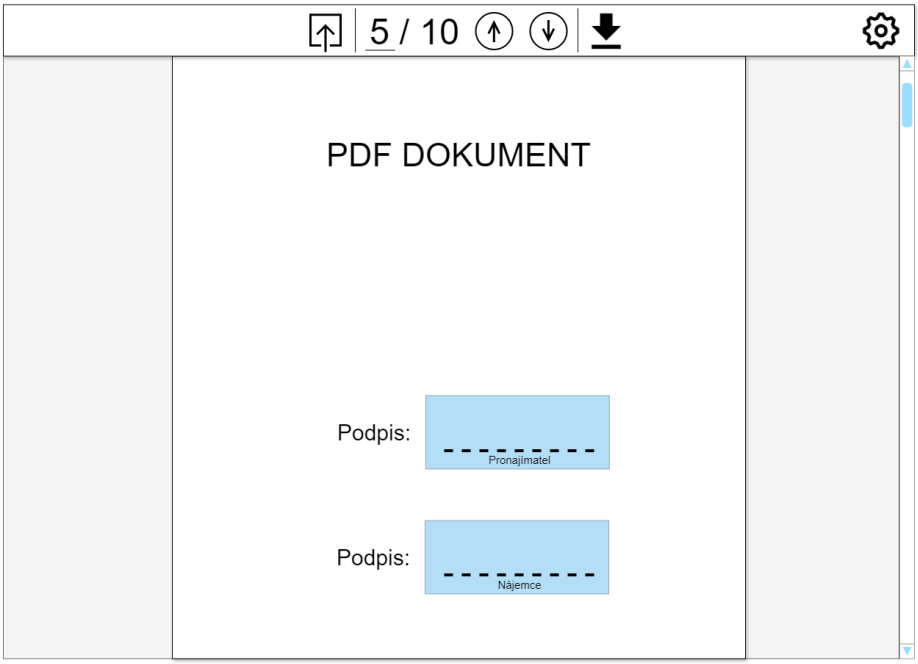
podpisů je plocha pro vytváření podpisu samotného, v ní je možné pomocí myši, nebo připojeného podpisového tabletu vytvářet podpis. Nad plochou pro vytváření podpisu jsou tři tlačítka, které zajišťují tyto funkce:

- **Zrušení podpisu** – zavře podpisový editor bez uložení podpisu.
- **Vymazání pole** – vymaže vytvořený podpis, ale nezavře podpisový editor. Použití v případě, že se uživateli nelíbí podpis, který vytvořil.
- **Přijmutí podpisu** – přijme vytvořený podpis a uloží ho do dokumentu.

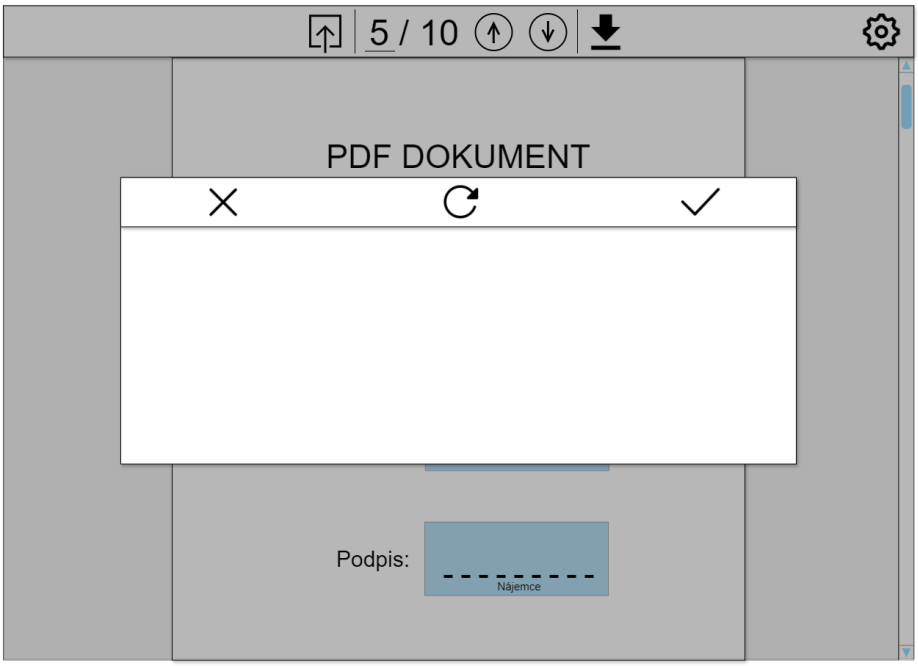


Obrázek 4.3: Grafický návrh editoru podpisů

Spojením výše popsaných elementů vznikne celkový návrh aplikace. Na obrázku 4.4 je celkový návrh aplikace a na obrázku 4.5 je návrh aplikace s otevřeným editorem podpisů.



Obrázek 4.4: Grafický návrh aplikace. Výchozí pohled.



Obrázek 4.5: Grafický návrh aplikace. Pohled na otevřený editor podpisů.

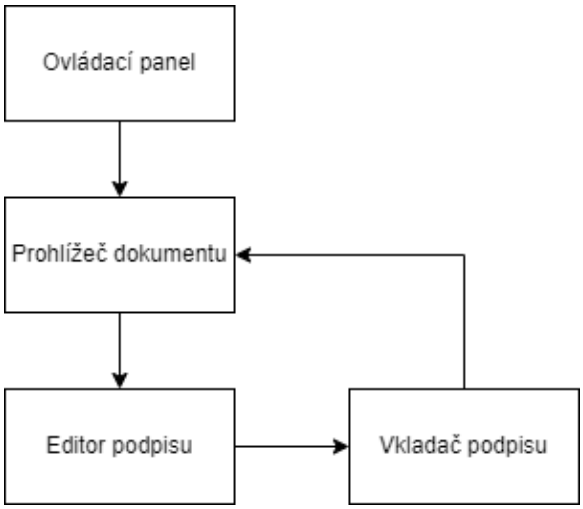
4.2 Návrh architektury

V této podkapitole se zaměřím na návrh vnitřní architektury aplikace. Popíšu použité technologie, vnitřní rozdělení na komponenty a volbu mezi client-side a server-side řešením. Tento návrh ovlivňuje schopnosti aplikace, její rychlost a bezpečnost.

4.2.1 Technologie

Pro vývoj jsem se rozhodl použít framework React. Tato volba ovlivňuje použité programovací paradigma, z velké části i rychlost výsledné aplikace a její velikost. Výchozím programovacím paradigmatem Reactu je deklarativní programování, i já se budu v této práci snažit co nejvíce držet tohoto přístupu. Hlavní roli při výběru frameworku hrálo to, jaký framework používá informační systém, do kterého bude tato aplikace později integrována. Podrobnější popis zvoleného frameworku a porovnání s ostatními frameworky si můžete přečíst v části 2.5.3 této práce. Pro zobrazování PDF dokumentů jsem zvolil knihovnu *pdf-js*, tato knihovna nepodporuje úpravu dokumentů, proto budu používat i knihovnu *pdf-lib* pro přidávání podpisů do dokumentu. Více detailů o těchto knihovnách, jejich schopnostech a podrobnější odůvodnění jejich použití je v části 2.4.2.

4.2.2 Rozdělení komponent



Obrázek 4.6: Graf interakce jednotlivých komponent

V Reactu se projekt dělí na jednotlivé komponenty, správné rozdělení zlepšuje čitelnost, udržovatelnost i škálovatelnost kódu. Měl by se dodržovat princip jedné odpovědnosti, který říká, že každá komponenta by měla mít jedinou odpovědnost a služby, které jsou od tohoto komponentu očekávány, by měly souviset s touto odpovědností [5]. Prvotní návrh rozdělení do komponent, vychází z požadavků na aplikaci (kapitola 3) a následného návrhu uživatelského rozhraní (podkapitola 4.1). Projekt by měl být rozdělen do následujících komponent:

- Ovládací panel – Zajišťuje nahrávání dokumentu, navigaci v dokumentu (ovládá prohlížeč dokumentu) a export dokumentu.
- Prohlížeč dokumentu – Zobrazuje dokument a podpisová pole v něm. Je z něj možné otevřít podpisové pole (editor podpisu).

- Editor podpisu – Slouží k vytváření podpisu. Zajišťuje i komunikaci s podpisovým tabletem.
- Vkladač podpisu – Edituje dokument, vkládá do něj obrázek i biometrická data podpisu.

Na obrázku 4.6 je zobrazena plánovaná interakce a propojení těchto komponent.

4.2.3 Bezpečnostní aspekty a volba mezi client-side a server-side řešením

Tato webová aplikace pracuje s osobními údaji, ty se mohou vyskytovat v nahraném dokumentu a jsou neodlučitelně přítomné při zpracování podpisu, z tohoto důvodu je důležité brát ohled na bezpečnost dat, se kterými se v rámci této aplikace pracuje. Pro vyšší bezpečnost, jsem se rozhodl přenášet jen minimum dat přes internet, protože při přenosu internetem je výrazně jednodušší citlivá data odcizit. Toto platí zejména pro uživatelův podpis. Pokud by byl podpis přenášen na server, a až tam zpracován, je nutné dbát na dobré zabezpečení serveru i zabezpečení přenosu dat na něj. Abych se této problematice vyhnul a snížil možnost zneužití osobních údajů, se kterými se v aplikaci pracuje, rozhodl jsem se provádět veškerou práci pouze v prohlížeči, tedy přímo na počítači klienta (tzv. client-side). Tento krok mi ztížil práci s PDF dokumenty, protože jejich zpracování je výpočetně náročné. Většina knihoven pro úpravu dokumentů je určena pro Node.js, což je serverová odnož JavaScriptu. Možnosti úpravy PDF dokumentů přímo v prohlížeči jsou velmi omezené, kvůli této skutečnosti, jsem musel upustit od původně plánované funkcionality vkládání nových podpisových polí do dokumentu.

V této kapitole jsem prezentoval návrh aplikace. Popsal jsem zde design uživatelského rozhraní, který klade důraz na přívětivost, efektivitu a estetiku. V druhé části se věnuji architektuře a členění kódu, s cílem zajištění jednoduché integrace do existujícího systému, bezpečnosti a dobrého výkonu. Na základě tohoto návrhu jsem aplikaci dále vyvíjel, což je blíže popsáno v kapitole 5.

Kapitola 5

Implementace

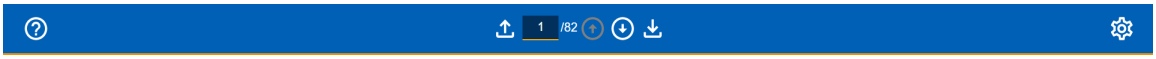
V této kapitole chci čtenáře seznámit s tím, jak konkrétně jsem řešil implementaci webové aplikace pro biometrické podepisování dokumentů. Při implementování jsem vycházel z návrhu aplikace, který jsem popsal v kapitole 4. Celá aplikace je psaná v jazyce JavaScript a frameworku React, více informací o použitém programovacím jazyce je možné nalézt v podkapitole 2.5 a o použitém frameworku pak v části 2.5.3. Z důvodu bezpečnosti údajů, se kterými se v aplikaci pracuje (zejména podpisů) jsou veškeré činnosti prováděny na straně klienta, díky tomu nejsou žádná citlivá data přenášena přes internet, ani ukládána na serveru, kde by bylo jednodušší je odcizit. Z toho vychází, že tato webová aplikace nemá žádnou serverovou logiku ani databázi.

5.1 Rozdíly oproti návrhu

V konečné verzi aplikace bylo implementováno několik funkcí, které se nevyskytují v návrhu. Při návrhu aplikace, se nezdály tyto funkce jako nutné, avšak v procesu vývoje se ukázalo, že jsou potřebné. Do ovládacího panelu bylo přidáno tlačítko pro přístup k nápovědě. Snímky okna s nápovědou lze vidět v příloze B. Při kliknutí na toto tlačítko se otevře vyskakovací okno, kde jsou popsány postupy pro nahrání dokumentu do aplikace, zvláště důležitý je popis způsobu nahrávání dokumentů z internetu pomocí URL, dále je zde popsán postup pro připojení podpisového tabletu. Vylepšení oproti návrhu se dočkal také editor podpisů, kam byl přidán textový element zobrazující název právě podepisovaného podpisového pole, což zlepšuje orientaci uživatele v právě zpracovávaném dokumentu.

5.2 Aplikační jádro

Aplikační jádro spojuje všechny ostatní komponenty aplikace. Jedná se o vstupní bod, ze kterého je aplikace spouštěna. Jsou zde vytvářeny ostatní komponenty a je zajišťována jejich vzájemná komunikace. Také je zde zajišťováno nahrávání PDF dokumentu z internetu a to kvůli způsobu, jakým je dokument z internetu vybírán. Pro nahrání dokumentu z internetu je předáno jeho umístění jako parametr URL této webové aplikace, díky tomu je možné mít v jiné aplikaci nebo informačním systému tlačítko pro podepsání dokumentu, které otevře mou webovou aplikace, kde bude již daný dokument nahrán automaticky bez další akce uživatele. Tento přístup k nahrávání dokumentu umožňuje dobrou integraci do jiných aplikací a systémů. Ostatní komponenty aplikace, které jsou ovládány z aplikačního jádra, jsou popsány v ostatních podkapitolách této kapitoly.



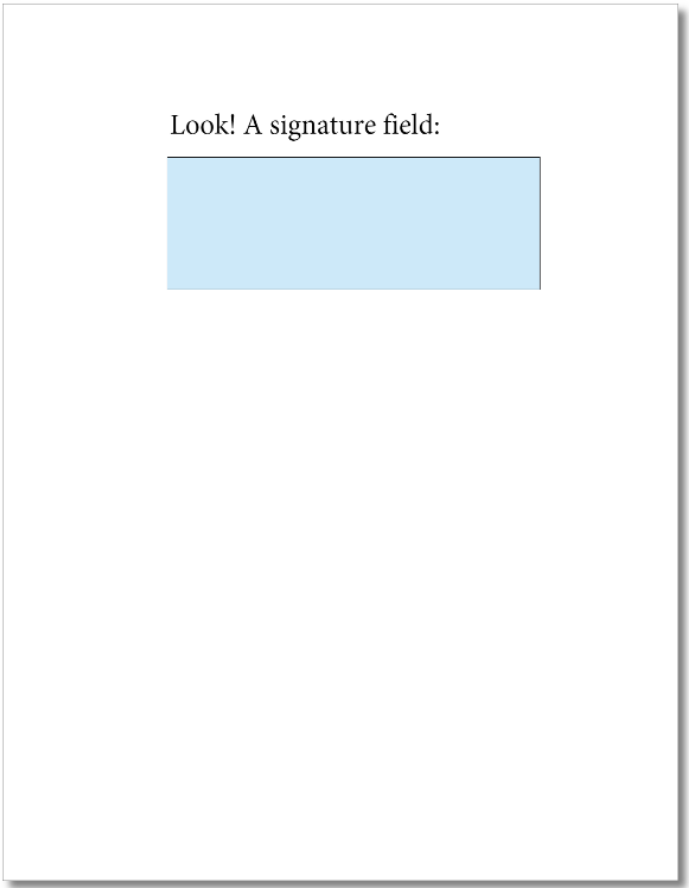
Obrázek 5.1: Snímek ovládacího panelu

5.3 Ovládací panel

K ovládání většiny částí aplikace slouží ovládací panel, v této části ho popíšu z pohledu jeho funkcionality, grafický návrh je popsán v části 4.1.1. S pomocí ovládacího panelu je možné nahrávat dokumenty do aplikace, navigovat v nich, exportovat je a v neposlední řade také měnit nastavení aplikace a zobrazovat nápovědu. Nahrávání dokumentu je možné dvěma způsoby a to buď z vnitřní paměti počítače, nebo z internetu. Pro nahrávání z počítače je v ovládacím panelu tlačítko, které otevře systémové okno pro výběr souboru. Pro vytváření a práci s oknem pro vybírání souboru je použita knihovna *React File Reader*, v ní vybraný a nahraný soubor je předán do aplikačního jádra (5.2), kde je dále zpracován a předán do prohlížeče dokumentu. Nahrávání dokumentů z internetu zcela zajišťuje aplikační jádro (5.2). Pro navigaci v dokumentu je zde textový vstup, do kterého je možné zadat číslo stránky a dvě tlačítka pro přechod na předchozí nebo následující stránku, informace o čísle stránky je předávána do prohlížeče dokumentů. Dále je zde tlačítko pro export, které umožňuje dokument i s přidanými podpisy uložit do počítače. V poslední řade jsou zde tlačítka pro otevření nápovědy a nastavení, tyto tlačítka otevírají vyskakovací okna s jejich příslušným obsahem princip implementace vyskakovacích oken je již popsán v podkapitole popisující implementaci editoru podpisů (5.5). Na obrázku 5.1 je možné vidět naimplementovaný ovládací panel.

5.4 Prohlížeč dokumentu

Tato část aplikace má za úkol zobrazování PDF dokumentů a podpisových polí. Popis návrhu grafického rozhraní tohoto komponentu je v části 4.1.2. K zobrazování dokumentu jsem použil knihovnu PDF.js (blíže popsanou v části 2.4.2), s její pomocí zobrazuji dokument na HTML canvas neboli plátno. Zobrazena je vždy jen jedna stránka dokumentu a k přepínání na jiné stránky slouží ovládací panel (popsaný v části 5.3). Protože je dokument zobrazován na plátno, text v dokumentu není v mé aplikaci textovým elementem, ale obrázkem. Metoda vykreslování na plátno, kterou používám, sice neumožňuje označování textu v dokumentu ani jeho kopírování, ale tato funkce není důležitá pro úkol, který tato webová aplikace plní. Vykreslování na plátno jsem zvolil proto, aby bylo možné zobrazovat nejen běžně zobrazovaný obsah dokumentu, ale i další specializované obrazové prvky, a to konkrétně graficky zvýrazněná podpisová pole, díky tomuto přístupu je také zaručeno, že bude možné pracovat i s dokumenty obsahující netradiční prvky, za cenu že může být snížena jejich interaktivita. Pro každé podpisové pole na zrovna zobrazované stránce je přes dokument vykreslen částečně průhledný obdélník se stejnými rozměry a pozicí jako má podpisové pole. Při pohybu myší po dokumentu je kontrolováno, jestli myš nezasahuje do některého z podpisových polí, pokud ano, je možné stlačením levého tlačítka myši otevřít editor podpisu (popsaný v části 5.5) a přes něj do daného pole vložit podpis. Snímek prohlížeče dokumentu i se zobrazeným podpisovým polem můžete vidět na obrázku 4.1.2.

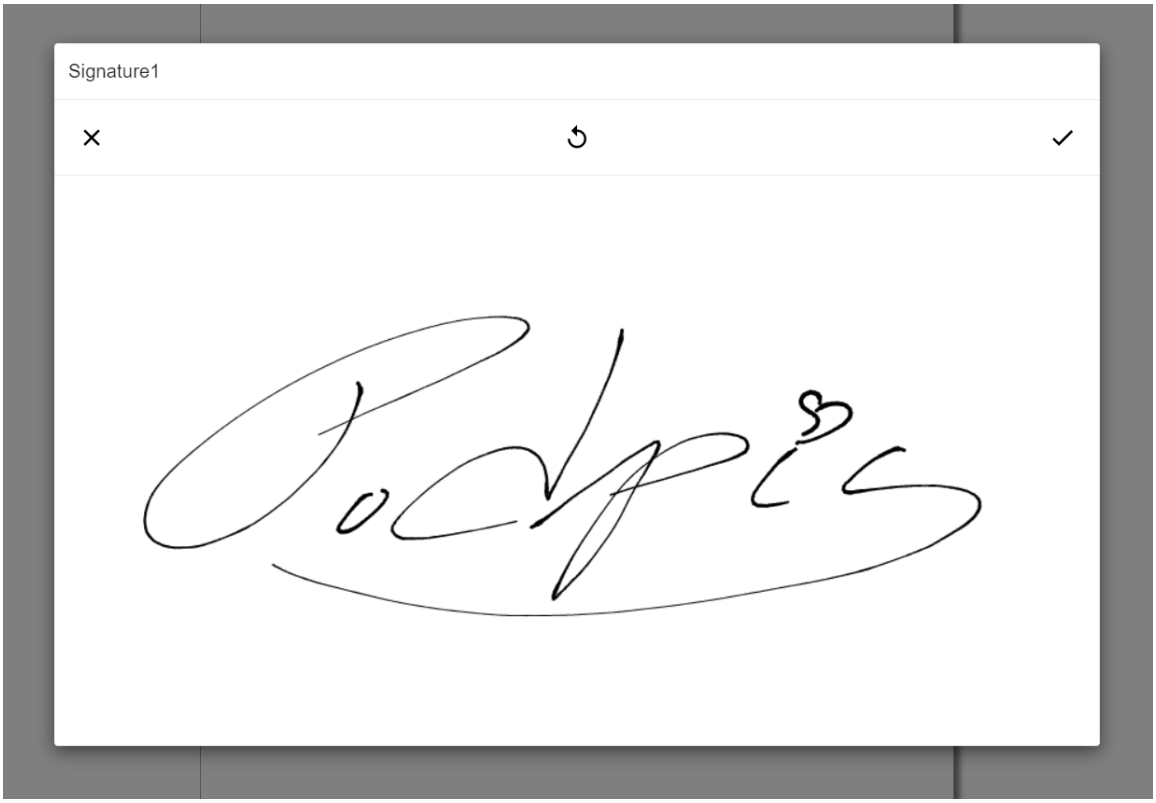


Obrázek 5.2: Snímek prohlížeče dokumentu

5.5 Editor podpisů

V této části popíšu implementaci editoru podpisů. Návrh uživatelského rozhraní tohoto komponentu je popsaný v části 4.1.3. Editor podpisů je vyskakovací okno, ve kterém uživatel vytváří podpisy. Podpis může být vytvářen pomocí myši, dotyku na dotykové obrazovce nebo pomocí podpisového tabletu. Princip komunikace s podpisovým tabletem je blíže popsaný v části 5.6. Vyskakovací okno bylo implementováno s pomocí nástroje *react-overlays* [30]. Tento nástroj ulehčuje implementaci logiky vyskakovacího okna, s jeho pomocí jsem vytvořil vyskakovací okno, které překryje současný stav aplikace. Zavřít vyskakovací okno je možné jak tlačítkem pro to určeným, tak i kliknutím mimo vyskakovací okno. Pokud je vyskakovací okno otevřeno, jsou části aplikace na pozadí deaktivovány, díky tomu je možné se v tlačítkách vyskakovacího okna pohybovat jen pomocí klávesnice, bez nežádoucí interakce s elementy v pozadí. Pokud je připojen podpisový tablet, jsou tlačítka pro ovládání editoru podpisů (zavření editoru bez uložení, smazání vytvořeného podpisu, uložení vytvořeného podpisu) zobrazena i na něm, uživatel tak nemusí v procesu vytváření podpisu pro tyto činnosti používat myš ani klávesnici. V editoru podpisů kreslí uživatel svůj podpis. Digitálně vytvořené podpisy se stálou šířkou křivky nevypadají přirozeně, proto jsem se rozhodl při vytváření podpisu využít i informaci o rychlosti pera, za tímto účelem jsem modifikoval knihovnu *Signature Pad*, jejíž hlavní vývojář je Szymon Nowak. Tato knihovna pracuje s interpolovanými Bézierovými křivkami proměnlivé šířky [25]. Po-

mocí těchto křivek jsou vytvářeny přirozeně vypadající podpisy. Šířka křivky je založena na rychlosti a změně směru pohybu virtuálního pera. Díky mé modifikaci je možné tuto knihovnu využívat i s podpisovými tablety značky Signotec. V jednom podpisu je možné libovolně kombinovat kreslení myší, prstem na dotykové obrazovce nebo perem na podpisovém tabletu. Velikost plátna, na kterém je podpis vytvářen, je dynamická a mění se na základě poměru velikosti stran právě připojeného podpisového tabletu. Po vytvoření podpisu v podpisovém editoru je obrázek ještě oříznut o prázdné místo okolo podpisu, uživatel tak nemusí použít celou plochu pro vytváření podpisu a výsledný podpis bude vždy dostatečně velký a zarovnaný na střed. Takto upravený obrázek podpisu je předán vkladači podpisů (5.7), který s ním dále pracuje. Na obrázku 5.3 je snímek implementace této části i s nakresleným podpisem.



Obrázek 5.3: Snímek editoru podpisů

5.6 Komunikace s podpisovým tabletem

Jednou z klíčových vlastností této aplikace je její schopnost komunikace s podpisovým tabletem. V této části se zaměřím na to, jak byla tato část implementována. Pravidla komunikace PC a podpisových tabletů nejsou standardizována, každá firma používá svůj vlastní způsob komunikace.

Jako výrobce podpisových tabletů, se kterými bude tato aplikace kompatibilní, byla zvolena Německá firma Signotec. Při implementaci této části, jsem dbal na možnost případného rozšíření o schopnost komunikace s podpisovými tablety jiného výrobce. V současné době, jsou ale podporovány pouze všechny podpisové tablety od výrobce Signotec.

Dále se zaměřím přímo na popis komunikace aplikace s podpisovým tabletem používající komunikační standard firmy Signotec. Aby bylo možné komunikovat s podpisovým tabletem z prohlížeče bez použití zásuvných modulů, vyvinula firma Signotec řešení *signoPAD-API/Web* [10], které slouží jako rozhraní mezi podpisovým tabletem a webovou aplikací. SignoPAD-API/Web pracuje jako lokální server, který může běžet v operačních systémech Windows i Linux, na jedné straně komunikuje s podpisovým tabletem a přes webový server ho zpřístupňuje ostatním aplikacím. Umožňuje jak zobrazovat obsah na tabletu, tak i získávat data z tabletu. Použitím aplikačního programového rozhraní WebSocket umožňuje současnou oboustrannou komunikaci (plný duplex) za použití jedné schránky (socketu). Má aplikace tento server využívá. Po připojení k serveru se načtou informace o serveru samotném a o připojeném podpisovém tabletu. Jakmile je spuštěn proces vytváření podpisu (blíže popsaný v části 5.5) je tabletu poslán signál, který zahájí detekci přiložení pera na plochu tabletu, společně s tímto signálem je mu také zaslán název podepisovaného podpisového pole, který se na podpisovém tabletu zobrazí. Při pohybu pera po displeji podpisového tabletu, je webové aplikaci poslána informace o pozici pera, ta je zpracována editorem podpisů (5.5) a na základě posloupnosti těchto pozic je v editoru podpisů vytvořen obrázek podpisu. Po přijetí podpisu jsou od tabletu získána biometrická data, která jsou poté společně s vytvořeným obrázkem podpisu zpracována ve vkladači podpisů popsaném v části 5.7.

5.7 Vkladač podpisu

V této části popíšu detaily implementace vkladače podpisů. Vkladač podpisů je část aplikace, která edituje PDF dokument. Vkladač podpisů nemá žádné grafické uživatelské rozhraní, uživatel ho používá pouze nepřímo interakcí s editorem podpisů, který je popsaný v části 5.5. Vkladač funguje asynchronně, jeho vstupy jsou PDF dokument ve formátu base64 zakódovaného řetězce, název podpisového pole, obrázek podpisu (také ve formátu base64 řetězce) a biometrická data podpisu. Pro editaci PDF dokumentu je zde použita JavaScriptová knihovna pdf-lib, ta umožňuje editaci podpisových polí v dokumentu, ale to jen ve smyslu vložení obrázku do podpisového pole. Podrobnější popis této knihovny a její porovnání s jinými knihovnami pro práci s PDF dokumenty je v podkapitole 2.4.2. Za pomoci této knihovny je k podpisovému poli specifikovanému jeho názvem přiřazen obrázek podpisu, ten se tak zobrazí ve výsledném dokumentu, právě na pozici daného podpisového pole. Obrázek podpisu je zvětšen tak, aby maximálně využil prostoru podpisového pole, ale vždy je zachován originální poměr stran podpisu. Pokud jsou dodána i biometrická data, jsou uložena do metadat dokumentu, konkrétně jako záznam v tabulce vlastních vlastností dokumentu (anglicky custom document properties). Pro každý vložený biometrický podpis je v této tabulce vytvořen nový záznam, v názvu záznamu je identifikováno podpisové pole, ke kterému biometrická data náleží a hodnotou záznamu jsou biometrická data samotná. Takto upravený dokument je poté vrácen do hlavního komponentu celé aplikace, ze kterého je zobrazen uživateli a je také možné ho exportovat ven z aplikace. Na obrázku 5.4 je zobrazen upravený dokument, do kterého byl vložen podpis, na obrázku je vidět jak vložený obrázek podpisu, tak i otevřené okno zobrazující vlastnosti dokumentu s vloženými biometrickými daty podpisu.

V této kapitole jsem popsal jednotlivé části aplikace a detaily jejich implementace. Popsal jsem použité technologie a knihovny. Je zde také odůvodněno, proč jsem použil některé přístupy a implementační postupy. Výsledkem spojení popsaných částí je aplikace, ve které

Obrázek 5.4: Snímek upraveného a vyexportovaného PDF dokumentu, zobrazen je obrázek podpisu i uložená biometrická data.

je možné vytvářet podpisy, poté je přidávat do PDF dokumentů a takto upravené dokumenty ukládat do zařízení. Dokumenty je možné nahrávat z vnitřního úložiště zařízení, nebo z internetu. Je podporováno velké množství vstupních metod, podpisy je možné vytvářet pomocí myši, specializovaného podpisového tabletu a i pomocí dotykové obrazovky. Výsledná aplikace je responzivní a díky možnosti vytváření podpisů pomocí dotykové obrazovky je možné ji plnohodnotně používat i na mobilních zařízeních. Při vývoji jsem dbal na dobré rozdělení kódu a možnost jeho pozdějšího rozšíření, díky tomu je možné v budoucnu přidávat další funkcionalitu, nebo podporu nových vstupních zařízení. Více snímků z výsledné webové aplikace je možné vidět v příloze B.

Kapitola 6

Testování

Tato kapitola se zaměřuje na jednu z posledních, ale stále velmi důležitých fází vývoje aplikace, a to testování. Testování poskytuje informace o kvalitě aplikace. Zároveň odhaluje špatně fungující části aplikace, neočekávané chování, neintuitivní prvky uživatelského rozhraní a mnoho dalšího. Tento proces je klíčový pro zajištění spolehlivosti a uživatelské přívětivosti aplikace. Na této aplikaci byly provedeny tři základní skupiny testů. První z nich je funkční testování, které ověřuje, že funkcionalita aplikace neobsahuje chyby. V uživatelském testování se získávají informace o kvalitě uživatelského prostředí. Poslední skupinou je testování výkonu, které podrobuje aplikaci výpočetně náročným scénářům a sleduje její výkon.

6.1 Funkční testování

V rámci procesu vývoje webové aplikace je nutné zajistit, aby všechny implementované funkce byly plně funkční a odpovídaly požadavkům stanoveným v kapitole 3.1. Funkční testování je v tomto procesu klíčové. Tato podkapitola popisuje proces funkčního testování, které bylo prováděno manuálně a iterativně.

Primárním cílem funkčního testování je ověřit, že každá funkce aplikace pracuje podle specifikací, plní požadavky uživatelů a funguje bez chyb.

Funkční testování bylo prováděno v následujících fázích:

1. Definice testovacích scénářů: Na základě specifikací a požadavků, byly stanoveny testovací scénáře, které pokrývaly specifickou část funkčních aspektů aplikace
2. Manuální provedení testů: Testovací scénáře byly manuálně provedeny a výsledky testů byly zaznamenány.
3. Analýza a implementace oprav: Identifikované chyby byly podrobeny analýze, na jejímž základě byla navržena a následně implementována jejich oprava.
4. Opětovné testování: Po opravě chyb bylo testování opakováno s cílem ověřit úspěšnost nápravných opatření a vyloučit vznik nových problémů , aby se zajistilo, že opravy byly úspěšné a nevznikly jiné problémy.

Funkčním testováním byly průběžně odhalovány chyby v implementaci aplikace, tyto chyby byly na základě testů opravovány. Díky funkčnímu testování byla vytvořena relativně bezchybová webová aplikace. Po provedení funkčního testování se mohlo přejít na další typy testování, ve kterých se testovala uživatelská přívětivost a výkonnost aplikace.

6.2 Uživatelské testování

Uživatelské testování poskytuje zpětnou vazbu přímo od potenciálních uživatelů. V této sekci se zaměřím na metodiku, provedení a analýzu uživatelského testování, které pomáhá zajistit, že finální aplikace bude nejen funkční, ale i uživatelsky přívětivá.

Testování bylo provedeno na vzorku tří subjektů, přičemž subjekt číslo jedna je studentem fakulty informačních technologií a subjekty číslo dva a tři jsou zaměstnanci firmy Seacomp. Všem subjektům byl prezentován stejný scénář: dostali otevřenou prázdnou webovou aplikaci s úkolem podepsat specifikovaný dokument a následně jej uložit zpět do počítače. Během testování jsem jim neposkytoval žádné instrukce ani rady, pouze jsem pozoroval jejich postup a zaznamenával jejich komentáře a postřehy. Následuje popis průběhu jednotlivých testů:

6.2.1 Subjekt č.1

Subjekt číslo jedna zahájil interakci s aplikací otevřením okna nápovědy. Po prostudování dostupných instrukcí nápovědu zavřel a přešel do sekce nastavení. Zjistil, že žádné změny v nastavení není potřeba provádět, a proto se vrátil k hlavnímu rozhraní. Následně prostřednictvím tlačítka v prostoru prohlížeče dokumentů provedl nahrání dokumentu. Toto tlačítko je zobrazeno pouze v případě, že v aplikaci aktuálně není nahrán žádný dokument. Proces výběru a nahrání dokumentu v tomto případě proběhl bez komplikací.

V dalším kroku subjekt lokalizoval místo pro podpis a kliknutím na jeho pozici otevřel editor podpisů. Tento krok rovněž spustil proces vytváření podpisu na připojeném podpisovém tabletu. Subjekt následně vytvořil podpis pomocí podpisového tabletu, ovšem s prvním pokusem nebyl spokojen. Proto na podpisovém tabletu použil tlačítko pro vymazání vytvořeného podpisu a následně vytvořil nový podpis. S novým podpisem již byl spokojen a pomocí tlačítka na podpisovém tabletu podpis přijal.

Po úspěšném vložení podpisu do dokumentu subjekt přistoupil k exportu dokumentu. Vzhledem k nejistotě ohledně správného postupu subjekt prozkoumal ovládací panel a tlačítka v něm. Subjekt se zmínil, že by bylo užitečné, kdyby byla u tlačítek kontextová nápověda, která by objasnila jejich funkčnost. Po identifikaci odpovídajícího tlačítka pro export subjekt úspěšně dokument uložil.

6.2.2 Subjekt č.2

Subjekt číslo dvě začal přímo plnit zadaný úkol bez předchozího prostudování nápovědy či konfigurace nastavení. Pro lokalizaci dokumentu určeného k podpisu použil standardní prohlížeč souborů operačního systému Windows. Následně se pokusil o nahrání dokumentu do webové aplikace metodou táhni a pusť, tedy „uchopením“ ikony dokumentu z prohlížeče souborů a přetažením do prostoru webové aplikace, což však nevedlo k požadovanému výsledku, neboť dokument nebyl otevřen v testované webové aplikaci, ale v integrovaném prohlížeči dokumentů webového prohlížeče. Tento neúspěch si vyžádal opětovné spuštění webové aplikace. Při dalším pokusu o nahrání dokumentu zvolil subjekt postup pomocí tlačítka umístěného v ovládacím panelu, což splnilo jeho očekávání a dokument úspěšně nahrál

V rámci prohlížeče dokumentů subjekt lokalizoval určené místo pro vložení podpisu a kliknutím na něj aktivoval editor podpisů. S využitím připojeného podpisového tabletu vytvořil podpis, který následně akceptoval kliknutím na příslušné tlačítko v editoru. Po kontrole vzhledu a správného umístění podpisu v dokumentu subjekt přistoupil k exportu

dokumentu, k čemuž využil tlačítko na ovládacím panelu a podepsaný dokument úspěšně uložil do vnitřního úložiště zařízení.

6.2.3 Subjekt č.3

Subjekt číslo tři přistoupil k úkolu velmi přímočaře. Ihned po zahájení testu klikl na tlačítko pro nahrání dokumentu a vybral příslušný soubor. Následně lokalizoval podpisové pole a kliknutím na něj otevřel editor podpisů. V editoru podpisů nejprve vytvořil jednoduchý podpis pomocí počítačové myši, následně se ale rozhodl využít připojený podpisový tablet. Pomocí tlačítka pro stornování podpisu uzavřel editor podpisů, což zřejmě dle reakce nebyl jeho původní záměr. Po opětovném otevření editoru podpisů vytvořil nový podpis s využitím podpisového tabletu. Nově vytvořený podpis akceptoval pomocí příslušného tlačítka na podpisovém tabletu. Posledním krokem byl export dokumentu, ke kterému subjekt využil příslušné tlačítko v ovládacím panelu a podepsaný dokument úspěšně uložil.

Provedené uživatelské testování identifikovalo potenciální nedostatky a oblasti pro možné vylepšení aplikace, rovněž poskytlo zpětnou vazbu ohledně částí aplikace, které jsou již dobře navržené a efektivně implementované.

Jedním ze zjištěných nedostatků byla potřeba implementace kontextové nápovědy, která by popisovala funkce jednotlivých tlačítek, a to jak u tlačítek v ovládacím panelu, tak i v editoru pro vytváření podpisů. Absence kontextové nápovědy nutí uživatele dedukovat funkci tlačítek jen na základě jejich ikon, což může vést ke snížení efektivity práce s aplikaci, zejména pro nové uživatele, a také omezuje přístupnost aplikace pro lidi s tělesným postižením, zejména pro lidi trpící sníženou zrakovou schopností, kteří jsou závislí na technologiích pro čtení obrazovky s hlasovým výstupem. Tato funkcionalita byla na základě provedených testů do aplikace přidána.

Další zjištěný nedostatek, byla absence funkce nahrávání souborů metodou táhni a pusť (tzv. drag and drop). Tato funkce by v určitých situacích zjednodušila proces nahrávání dokumentu pro podepsání. Vzhledem k tomu, že aplikace již obsahuje alternativní způsob pro nahrání dokumentů, který byl na základě uživatelského testování ohodnocen jako intuitivní, a dokonce preferovaný většinou uživatelů, není dodatečná implementace této funkce prioritou.

Uživatelské testování odhalilo několik mírných nedostatků aplikace. Některé z reportovaných nedostatků byly na základě testování odstraněny. Celkově výsledky uživatelského testování ukázaly, že aplikace splňuje všechny stanovené požadavky a je vhodná pro plnění stanovených cílů. Z intuitivního porozumění ohledně volby správného postupu pro dosažení subjekty kýženého výsledku vyplývá, že uživatelské rozhraní aplikace je intuitivní a přehledné.

6.3 Testování výkonu

Jedna z klíčových oblastí aplikace, kterou je potřeba testovat, je výkonnost aplikace. Tato kapitola se zaměřuje na zátěžové testování, jehož cílem je ověřit schopnost aplikace zvládat vyšší množství zpracovávaných dat, aniž by došlo k degradaci výkonu nebo stability aplikace. Zátěžové testování je nezbytné pro identifikaci úzkých míst v architektuře aplikace.

Tato aplikace je plně implementována jako klientská aplikace zcela běžící v internetovém prohlížeči, výkonnost aplikace je tedy přímo závislá na parametrech zařízení, na kterém běží. Aplikace nemá žádnou serverovou část, která by byla sdílena mezi jednotlivými uživateli,

díky tomu není výkonnost ani stabilita aplikace ovlivněna počtem uživatelů používajících aplikaci souběžně. Zátěžové testy se tedy budou zaměřovat pouze na testování interakce aplikace s velkými objemy dat.

Funkční a uživatelské testování ukázalo, že běžné činnosti v aplikaci jsou dostatečně výkonné. Zpomalení aplikace tedy může teoreticky nastat pouze v případě zpracování velkého objemu dat, což se týká především zpracování PDF dokumentů. Zátěžové testy se tedy zaměří na práci s objemnými dokumenty.

Test rychlosti nahrávání dokumentů byl prováděn nahráváním dokumentů různých velikostí z vnitřního úložiště počítače do aplikace. Výkonnost nahrávání dokumentu z internetu nebyla testována, protože až na proces získání dokumentu ze vzdáleného úložiště přes internet je využíváno stejného postupu. Při testování nahrávání dokumentu z internetu by byly výsledky ovlivněny rychlostí internetového spojení a také vytížeností a výkonností serveru, ze kterého by byly dokumenty stahovány.

Běžná velikost dokumentů se pohybuje ve stovkách kilobytů až jednotkách megabytů. Při zátěžovém testování byly použity dokumenty o velikostech 50MB, 100MB a 200MB. V tabulce 6.1 jsou uvedeny průměrné časy nahrávání těchto dokumentů do testované aplikace. Tato data poskytují přehled o výkonnosti aplikace při manipulaci s objemnými soubory.

Tabulka 6.1: Doba za jakou je možné pracovat s nahraným dokumentem

Velikost dokumentu	Průměrný čas nahrávání
50MB	4,30 sekund
100MB	5,29 sekund
200MB	6,61 sekund

Zátěžový test ukázal, že aplikace efektivně zpracovává i dokumenty s nadprůměrnou velikostí. Čas potřebný k načtení dokumentu o velikosti 200MB byl přibližně 6 sekund, což je dobrý výsledek srovnatelný s časy načítání u běžně používaných prohlížečů dokumentů, u kterých se čas načítání stejného souboru obvykle pohybuje v časech přesahujících 10 sekund (Microsoft Edge 14 sekund, Google Chrome 13 sekund). Z testovaných programů vykázal lepší výkon pouze Adobe Acrobat, který tento soubor otevřel za pouhé 3 sekundy. Z výsledků zátěžového testování lze usuzovat, že výkonnost aplikace je adekvátní a v procesu podepisování dokumentů nebude uživatele limitovat.

Testování aplikace popsané v této kapitole bylo nezbytným krokem pro zajištění její dobré funkčnosti, výkonnosti a uživatelské přívětivosti. Díky funkčnímu testování bylo odhaleno a opraveno mnoho chyb, což vedlo k vytvoření robustní aplikace. Uživatelské testování poskytlo zpětnou vazbu od potenciálních uživatelů a ukázalo, že aplikace je intuitivní a snadno použitelná. Testování výkonu ukázalo, že aplikace zvládá i operace s velkými objemy dat, bez negativního dopadu na její použitelnost.

Testování ukázalo, že aplikaci je možné publikovat, avšak i nadále je nutné aplikaci monitorovat a průběžně sbírat zpětnou vazbu od jejích uživatelů.

Kapitola 7

Závěr

Cílem této práce bylo navrhnout a implementovat webovou aplikaci, která umožňuje vytváření biometrických podpisů s využitím specializovaných podpisových tabletů. Aplikace dále měla umožňovat vkládání vytvořených podpisů do existujících PDF dokumentů a následný export takto upravených dokumentů do počítače uživatele. Hlavní motivací tohoto projektu byla potřeba produktu pro podepisování dokumentů v rámci webového ambulančního informačního systému, na jehož vývoji se podílím jako zaměstnanec firmy Seacomp. Návrh aplikace a volba technologií byly provedeny tak, aby bylo možné výslednou aplikaci jednoduše integrovat do zmíněného systému. Cíl práce byl splněn a byla vytvořena intuitivní a stabilní webová aplikace umožňující vytváření podpisů a jejich následné vkládání do dokumentů. Před začátkem samotného vývoje byl proveden průzkum problematiky biometrického podepisování PDF dokumentů ve webovém prostředí. Byly prostudovány aspekty biometrických podpisů, a také fungování a historie formátu pro přenositelné dokumenty. Další zkoumanou a popsanou tématikou byly principy tvorby webových aplikací včetně hlubší analýzy JavaScriptových frameworků. V neposlední řadě byla provedena analýza existujících řešení pro biometrické podepisování dokumentů. Na základě specifikovaných požadavků na aplikaci a poznatků získaných při průzkumu problematiky byl vytvořen návrh aplikace, který zahrnoval jak uživatelské rozhraní, tak i vnitřní architekturu aplikace. Pro vývoj byl zvolen programovací jazyk JavaScript a framework React, což umožnilo vytvořit interaktivní a responzivní webovou aplikaci, kterou je jednoduché integrovat do stávajícího informačního systému. Pro zajištění bezpečnosti osobních dat, se kterými se v aplikaci pracuje, byla aplikace navrhnuta a implementována tak, aby trvale neukládala žádná data a všechny operace probíhaly přímo v počítači uživatele a ne na vzdáleném serveru. Tento přístup zcela znemožňuje odcizení osobních dat uživatelů prostřednictvím napadení serveru nebo pomocí útoků typu člověk uprostřed (man in the middle). Omezení spojená s tímto přístupem vedla k rozhodnutí upustit od implementace funkce pro přidávání nových podpisových polí do dokumentu, protože úpravy tohoto typu jsou vysoce výpočetně náročné a není možné je efektivně provádět v prohlížeči. Podpisy vytvořené v aplikaci, ať už pomocí specializovaného tabletu, či myši, jsou stylizovány tak, aby vizuálně připomínaly tradiční, perem psané podpisy a zároveň byla zachována jejich jedinečnost a autenticita. Aplikace byla podrobena testování, které potvrdilo její funkčnost, intuitivnost uživatelského rozhraní a schopnost zpracovávat velké objemy dat. Uživatelé, kteří aplikaci testovali, se v ní rychle zorientovali a byli schopni samostatně a efektivně provádět požadované operace. V rámci dalšího rozvoje aplikace by bylo vhodné zvážit přidání podpory podpisových tabletů od jiných výrobců. V dlouhodobém horizontu by bylo možné aplikaci vylepšit vytvořením ser-

verové části aplikace, která by umožnila vkládání nových podpisových polí a případně i další modifikace dokumentů.

Literatura

[1] Angular Team. *Angular - What is Angular?* 2023 [cit. 11.4.2024]. Dostupné z: https://angular.io/guide/what-is-angular.

[2] Bayat, A. a Pomplun, M. Biometric Identification Through Eye-Movement Patterns. *Advances in Human Factors in Simulation and Modeling*. Springer, Cham. Červen 2017. Dostupné z: https://doi.org/10.1007/978-3-319-60591-3_53.

[3] Bienz, T., Cohn, R. a Adobe Systems (Mountain View, C. *Portable document format reference manual*. Addison-Wesley Boston^ eMA MA, 1993.

[4] Docuten. *Digital Signature: Human Resources* [online]. Prosinec 2020 [cit. 24.2.2024]. Dostupné z: https://docuten.com/wp-content/uploads/2020/12/Whitepaper_HR_EN.pdf.

[5] Esaias, M. *Single Responsibility Principle in React*. 2024. Dostupné z: https://dev.to/mikhaelesa/single-responsibility-principle-in-react-10oc.

[6] Eve, M. P. New Leaves: Riffling the History of Digital Pagination. *Book History*. Johns Hopkins University Press. 2022, sv. 25, č. 2, s. 479–502. Dostupné z: https://doi.org/10.1353/bh.2022.0017.

[7] Free Software Foundation Inc. . *Introduction to PDF* [online]. Květen 2010 [cit. 23.3.2024]. Dostupné z: https://web.archive.org/web/20141010035745/http://gnupdf.org/Introduction_to_PDF.

[8] Gackenheimer, C. *Introduction to React*. Apress, 2015. ISBN 978-1-4842-1246-2. Dostupné z: https://doi.org/10.1007/978-1-4842-1245-5.

[9] GeeksforGeeks. What is Web App? - Web Application Explained. *GeeksforGeeks* [online]. 2024, [cit. 6.4.2024]. Dostupné z: https://www.geeksforgeeks.org/what-is-web-app/.

[10] GmbH signotec. *Signotec signoPAD-API/Web*. [cit. 22.4.2024]. Dostupné z: https://en.signotec.com/portal/seiten/signotec-signopad-api-web-900000547-10002.html.

[11] Govett, D. *Komentář k PDFKit problému "Edit existing PDF"*. Prosinec 2014 [cit. 14.4.2024]. Dostupné z: https://github.com/foliojs/pdfkit/issues/83#issuecomment-65164832.

[12] Harris, R. Svelte 3: Rethinking Reactivity. *Svelte Blog*. 2019, [cit. 13.4.2024]. Dostupné z: https://svelte.dev/blog/svelte-3-rethinking-reactivity.

[13] Hill, R. *Retina Identification*. Boston, MA: Springer, 1996. ISBN 978-0-306-47044-8. Dostupné z: http://doi.org/10.1007/0-306-47044-6_6.

[14] International Organization for Standardization. *Document management – Portable document format*. ISO 32000-1:2008. Vernier, Geneva, Switzerland: International Organization for Standardization, 2008. Dostupné z: https://www.iso.org/standard/51502.html.

[15] International Organization for Standardization. *Document management – Portable document format – Part 2: PDF 2.0*. ISO 32000-2:2020. Vernier, Geneva, Switzerland: International Organization for Standardization, 2020. Dostupné z: https://www.iso.org/standard/75839.html.

[16] Jazayeri, M. Some trends in web application development. In: IEEE. *Future of Software Engineering (FOSE'07)*. 2007, s. 199–213. ISBN 0-7695-2829-5. Dostupné z: https://doi.org/10.1109/FOSE.2007.26.

[17] Keith, J. A brief history of javascript. *DOM Scripting: Web Design with JavaScript and the Document Object Model*. Springer. 2005. Dostupné z: https://doi.org/10.1007/978-1-4302-3390-9_1.

[18] Lardinois, F. Google, Microsoft, Mozilla And Others Team Up To Launch WebAssembly, A New Binary Format For The Web. *TechCrunch*. 2015, [cit. 9.4.2024]. Dostupné z: https://techcrunch.com/2015/06/17/google-microsoft-mozilla-and-others-team-up-to-launch-webassembly-a-new-binary-format-for-the-web/.

[19] Lemos, R. Dependency Problems Increase for Open Source Components. *Dark Reading*. 2021, [cit. 8.4.2024]. Dostupné z: https://www.darkreading.com/application-security/dependency-problems-increase-for-open-source-components.

[20] Mairs, G. T. Fingerprint Signatures (A Word of Caution Concerning Their Use). *Journal of Criminal Law & Criminology*. Listopad 1936.

[21] Matkowski, W. M., Chan, F. K. S. a Kong, A. W. K. A study on wrist identification for forensic investigation. *Image and Vision Computing*. 2019, sv. 88. ISSN 0262-8856. Dostupné z: https://doi.org/10.1016/j.imavis.2019.05.005.

[22] MDN. WebAssembly Concepts. *MDN Web Docs*. 2024, [cit. 9.4.2024]. Dostupné z: https://developer.mozilla.org/en-US/docs/WebAssembly/Concepts.

[23] MOBBEEL SOLUTIONS. *MobbSign Brochure: Be.* [online]. Březen 2021 [cit. 24.2.2024]. Dostupné z: https://www.mobbeel.com/en/mobbsign-advanced-electronic-signature-biometrics/.

[24] Mozilla. *PDF.js*. 2024 [cit. 14.4.2024]. Dostupné z: https://github.com/mozilla/pdf.js.

[25] Nowak, S. *Signature Pad*. [cit. 23.4.2024]. Dostupné z: https://github.com/szimek/signature_pad.

[26] PDF LIB. *PDF-LIB*. [cit. 14.4.2024]. Dostupné z: https://pdf-lib.js.org/.

[27] PDFKit přispěvatelé. *PDFKit*. [cit. 14.4.2024]. Dostupné z: https://pdfkit.org/.

[28] pdfmake přispěvatelé. *Pdfmake*. 2021. Dokumentace. Dostupné z: https://pdfmake.github.io/docs/0.1/.

[29] Rak, R., Matyáš, V., Říha, Z. a Kolektiv. *Biometrie a identita člověka ve forenzních a komerčních aplikacích*. Grada, 2008. ISBN 978-80-247-2365-5.

[30] React-Bootstrap Team. *Overlay | React Bootstrap*. 2024 [cit. 23.4.2024]. Dostupné z: https://react-bootstrap.github.io/react-overlays/.

[31] React-pdf přispěvatelé. *React-pdf*. 2024. Dokumentace. Dostupné z: https://react-pdf.org/.

[32] Routley, N. *Animation: Internet Browser Market Share (1996-2019)*. Leden 2020 [cit. 7.4.2024]. Dostupné z: https://www.visualcapitalist.com/internet-browser-market-share/.

[33] signotec GmbH.. *Documentation signoSign/2* [online]. červen 2022 [cit. 24.2.2024]. Dostupné z: https://en.signotec.com/portal/seiten/signotec-signosign-2-signature-software-900000603-10002.html.

[34] signotec GmbH.. *Sign electronically Easy. Secure. Digital.* [online]. 2024 [cit. 24.2.2024]. Dostupné z: https://www.signosign-universal.signotec.com/?lang=en.

[35] SoftTeco. *History of JavaScript*. 2024 [cit. 7.4.2024]. Dostupné z: https://softteco.com/blog/history-of-javascript.

[36] *Stack Overflow Trends*. [cit. 10.4.2024]. Dostupné z: https://insights.stackoverflow.com/trends?tags=reactjs%2Cvue.js%2Cangular%2Csvelte%2Cvuejs3.

[37] *State of JavaScript 2022: Other Features*. 2022 [cit. 9.4.2024]. Dostupné z: https://2022.stateofjs.com/en-US/features/other-features/.

[38] TypeScript Team. *TypeScript: JavaScript With Syntax For Types*. 2023 [cit. 11.4.20240]. Dostupné z: https://www.typescriptlang.org/.

[39] Vue.js team. *Introduction. Vue.js*. [cit. 13.4.2024]. Dostupné z: https://vuejs.org/guide/introduction.html.

[40] W3Techs. *CP JavaScript Technology Details*. [cit. 7.4.2024]. Dostupné z: https://w3techs.com/technologies/details/cp-javascript.

[41] Wacom. *Sign pro PDF App* [online]. 2023 [cit. 24.2.2024]. Dostupné z: https://www.wacom.com/en-us/for-business/products/sign-pro-pdf-app.

[42] Warnock, J. E. The Origins of PostScript. *IEEE Annals of the History of Computing*. 2018, sv. 40, č. 3, s. 68–76. Dostupné z: http://doi.org/10.1109/MAHC.2018.033841112.

Příloha A

Kód jednoduchého PDF dokumentu

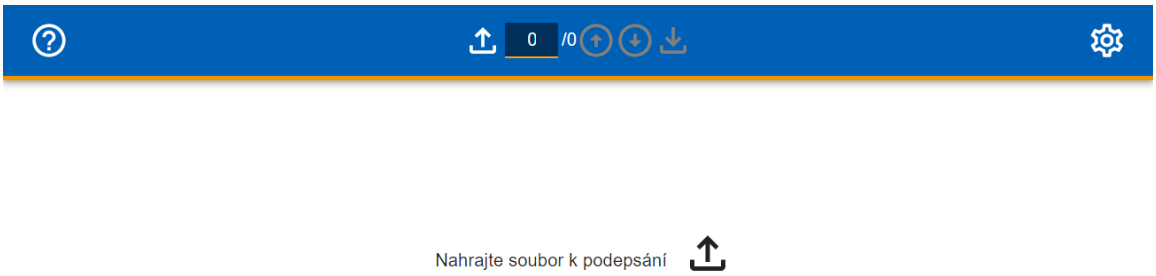
Příklad kódu reprezentujícího jednoduchý pdf dokument obsahující pouze text "Hello, world!". Byty které nereprezentují ASCII znak, byly odstraněny. Převzatý z *gnupdf.org* [7].

```
%PDF-1.7

1 0 obj  % entry point
<<
  /Type /Catalog
  /Pages 2 0 R
>>
endobj

2 0 obj
<<
  /Type /Pages
  /MediaBox [ 0 0 200 200 ]
  /Count 1
  /Kids [ 3 0 R ]
>>
endobj

3 0 obj
<<
  /Type /Page
  /Parent 2 0 R
  /Resources <<
    /Font <<
      /F1 4 0 R
    >>
  >>
  /Contents 5 0 R
>>
```

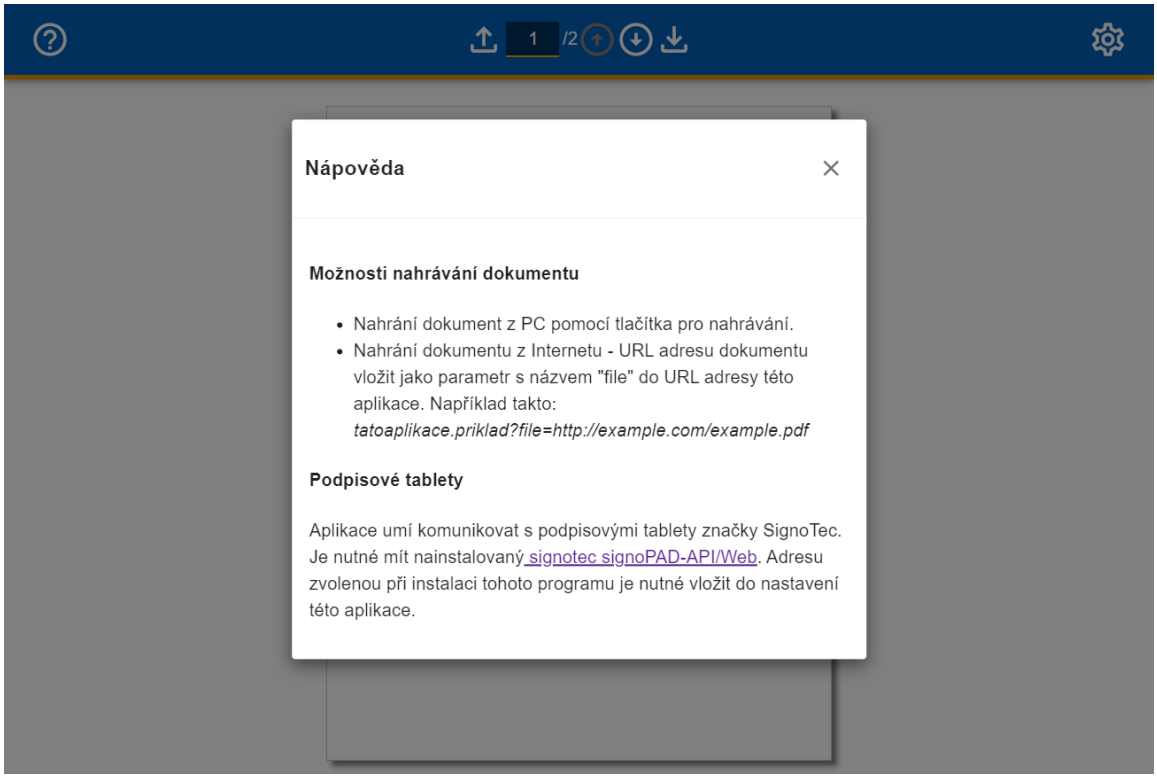
```
endobj

4 0 obj
<<
  /Type /Font
  /Subtype /Type1
  /BaseFont /Times-Roman
>>
endobj

5 0 obj  % page content
<<
  /Length 44
>>
stream
BT
70 50 TD
/F1 12 Tf
(Hello, world!) Tj
ET
endstream
endobj

xref
0 6
0000000000 65535 f
0000000010 00000 n
0000000079 00000 n
0000000173 00000 n
0000000301 00000 n
0000000380 00000 n
trailer
<<
  /Size 6
  /Root 1 0 R
>>
startxref
492
%%EOF
```

Příloha B

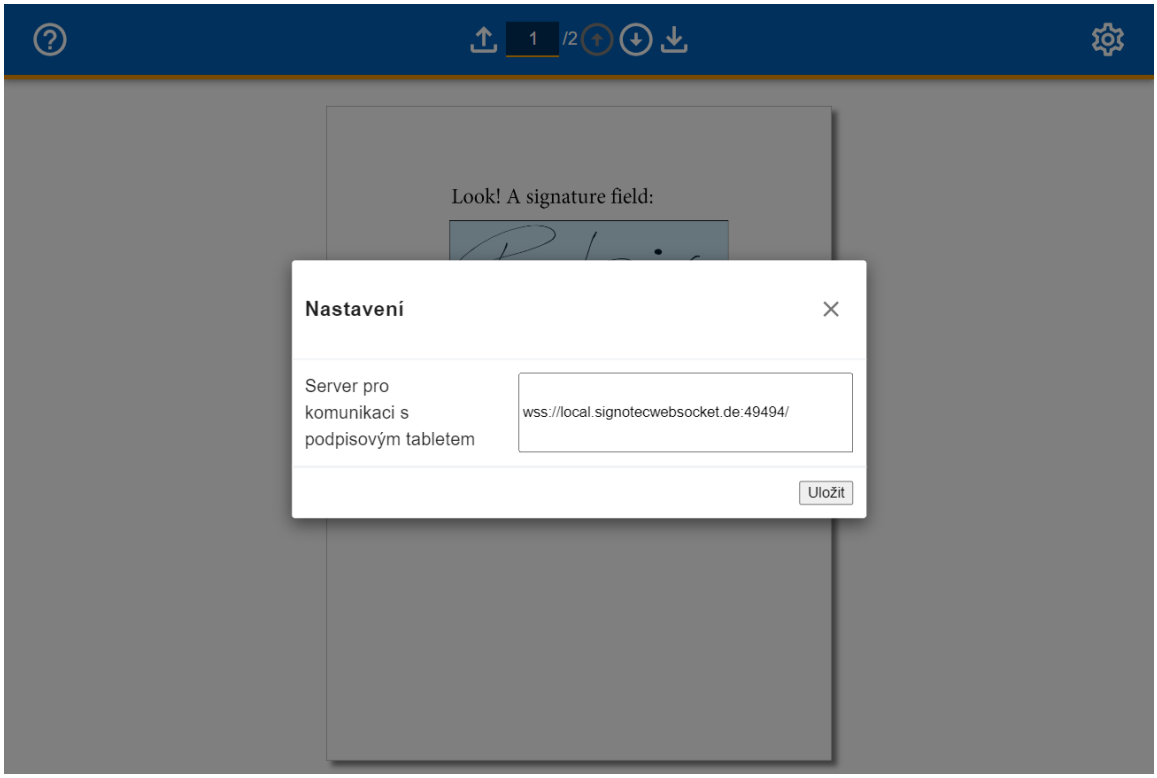
Snímky hotové webové aplikace



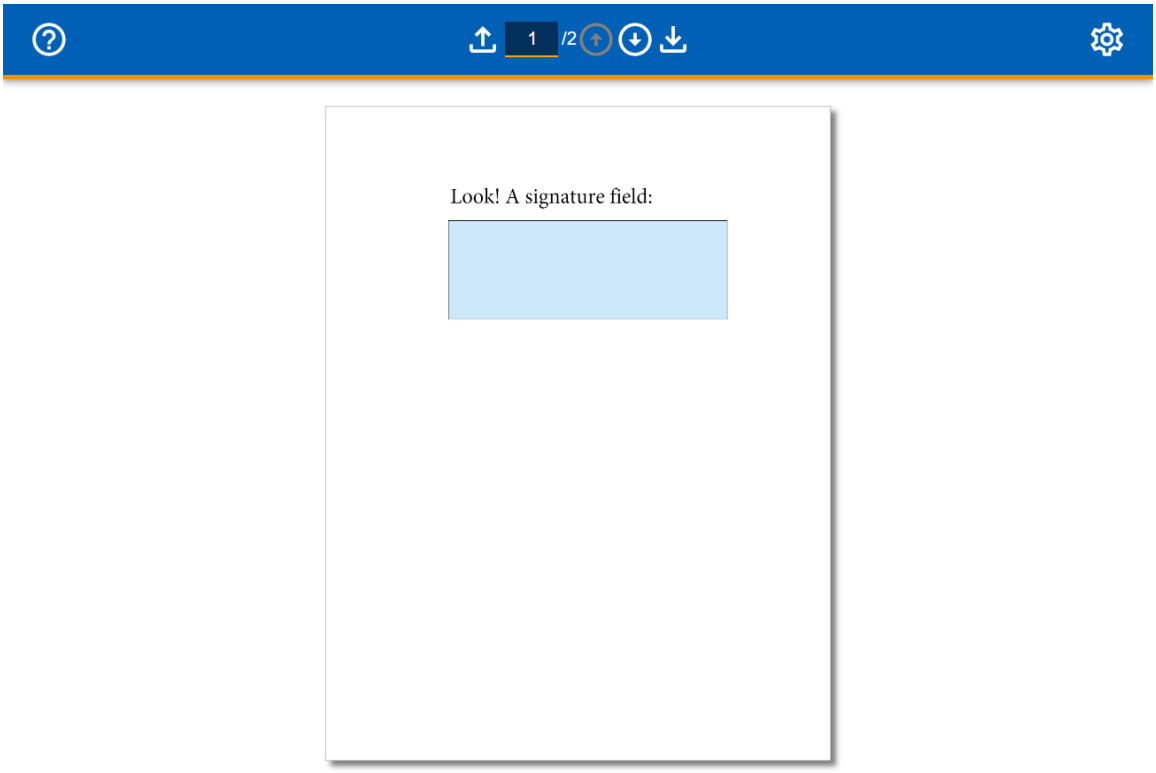
Obrázek B.1: Aplikace bez nahraného dokumentu



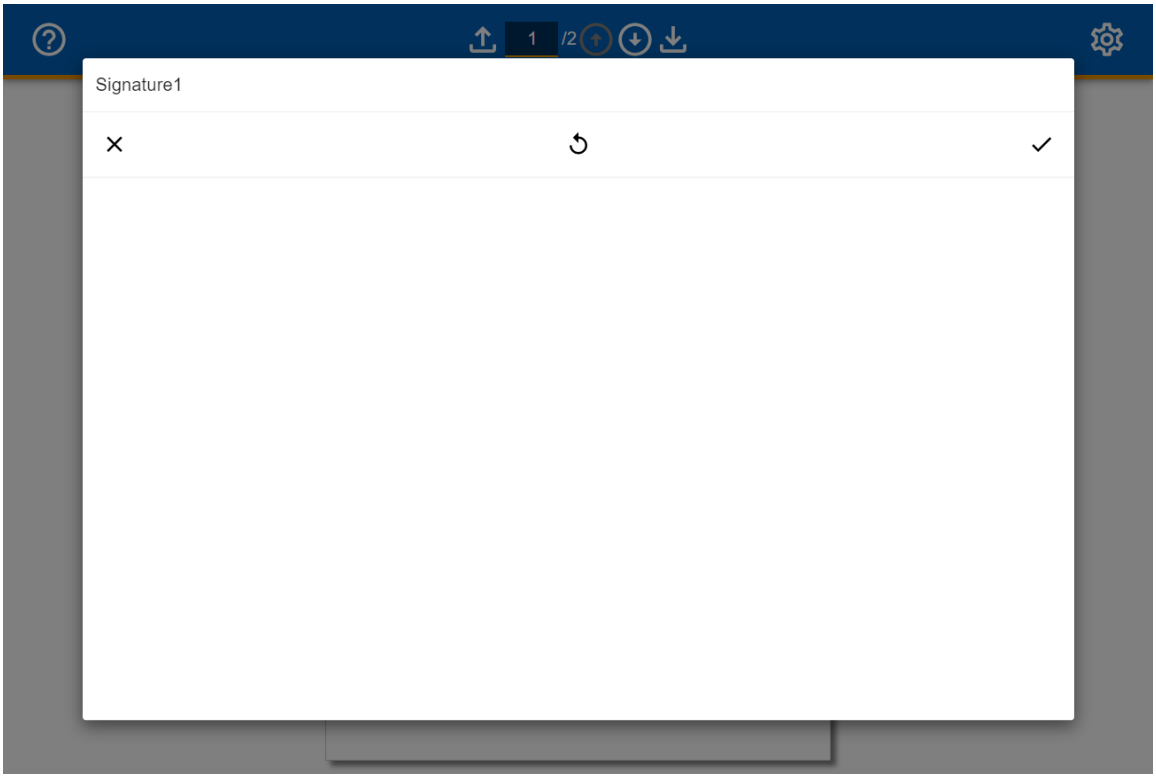
Obrázek B.2: Otevřené okno s nápovědou



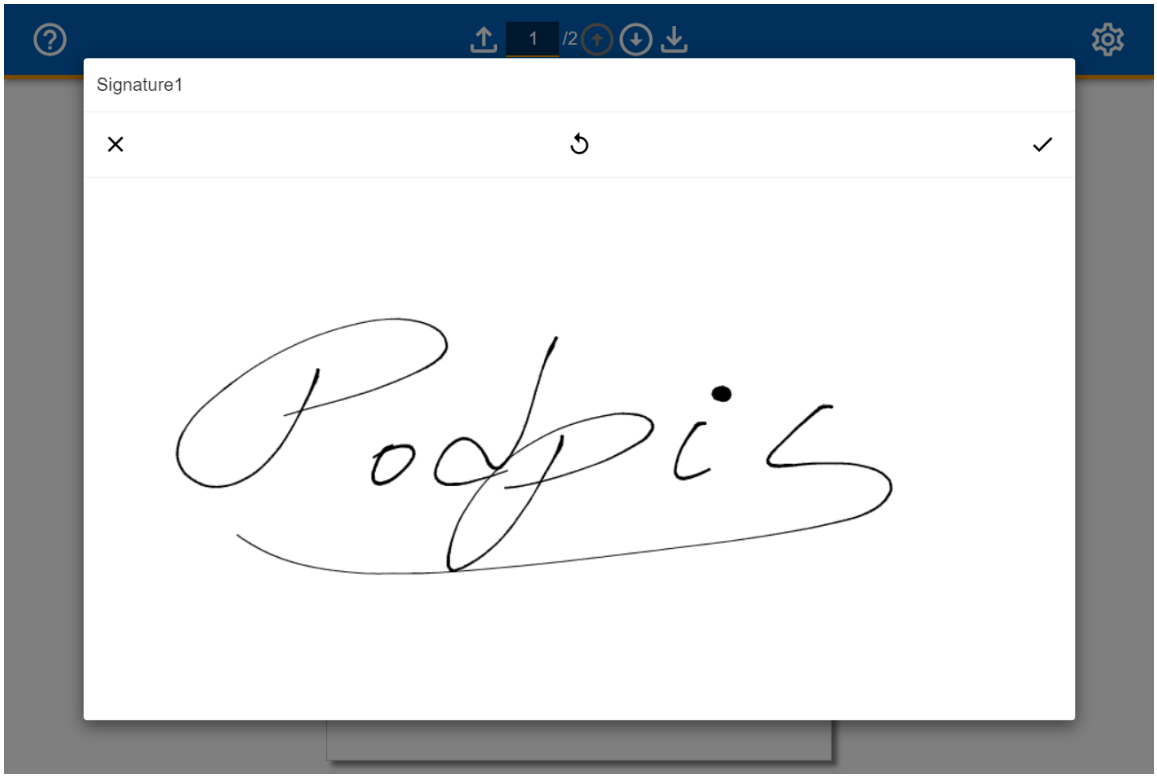
Obrázek B.3: Otevřené okno s nastavením



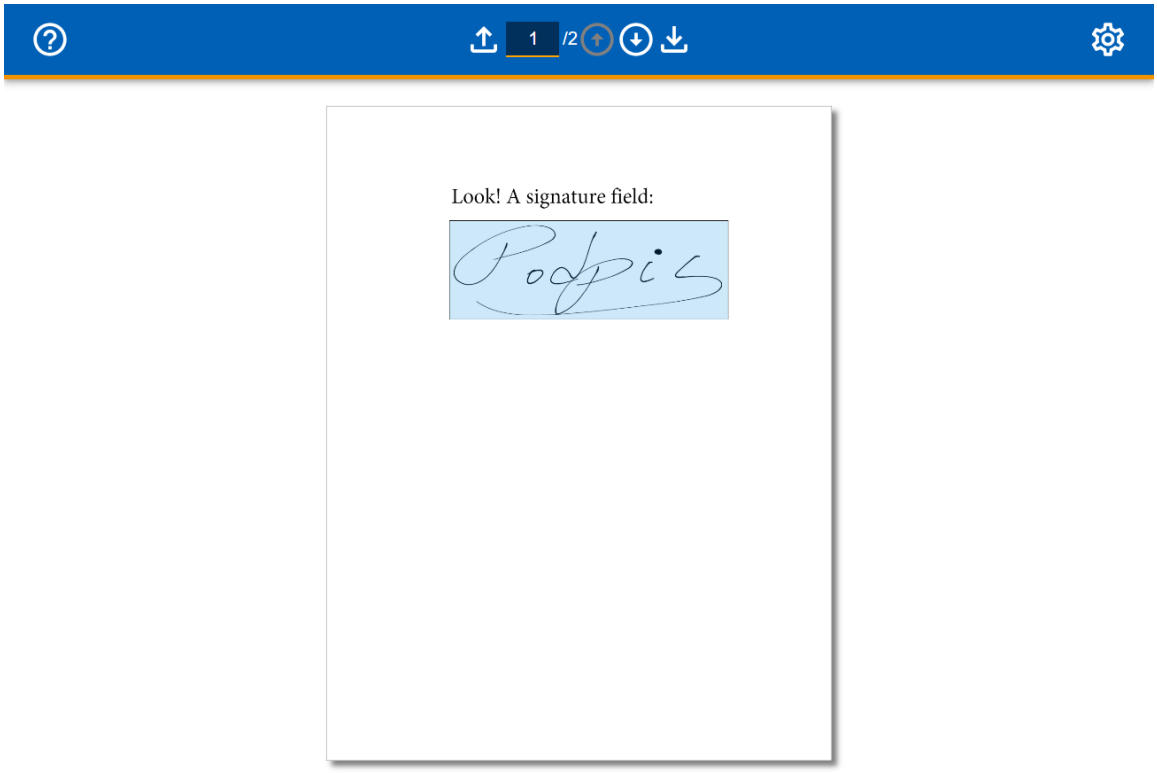
Obrázek B.4: Aplikace s nahraným dokumentem, který obsahuje podpisové pole



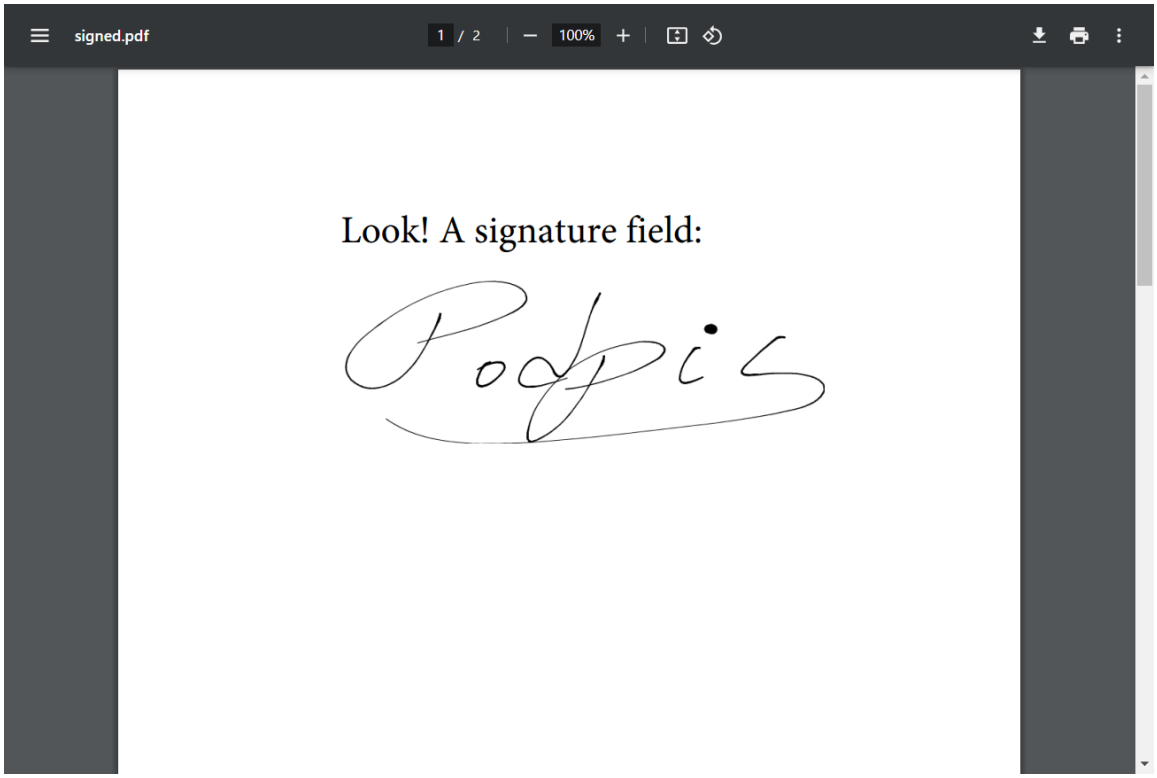
Obrázek B.5: Otevřený podpisový editor



Obrázek B.6: Podpisový editor s vytvořeným podpisem



Obrázek B.7: Aplikace zobrazující upravený dokument



Obrázek B.8: Dokument, který byl upraven pomocí implementované aplikace otevřený v prohlížeči Google Chrome