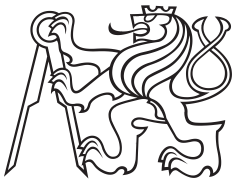
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Verzovací systém pro BPMN modely

David Mikulík

Vedoucí: Ing. Jan Vanke
Obor: Softwarové inženýrství a technologie
Květen 2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Mikulík** Jméno: **David** Osobní číslo: **511127**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

Specializace: **Business informatics**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Verzovací systém pro BPMN modely

Název bakalářské práce anglicky:

Versioning system for BPMN models

Pokyny pro vypracování:

Analyzujte požadavky na verzovací systém pro správu BPMN modelů a porovnejte existující řešení.
Na základě analyzovaných požadavků proveďte technický návrh tohoto systému a vyberte vhodné technologie pro jeho implementaci.
Vytvořte prototyp aplikace dle navržené architektury.
Navrhněte testovací strategii a scénáře, systém podrobte uživatelskému testování.

Seznam doporučené literatury:

1. DUMAS, Marlon; ROSA, Marcello La; MENDLING, Jan; REIJERS, Hajo A. Fundamentals of business process management. Druhé vydání. Berlin, Germany: Springer Berlin, 2018. ISBN 9783662565087
2. FREUND, Jakob a Bernd RÜCKER. Real-Life BPMN (4th edition): Includes an introduction to DMN. 4th edition. Camunda, 2019. ISBN 9781086302097
3. FORD, Neal; RICHARDS, Mark; SADALAGE, Pramod J. a DEHGHANI, Zhamak. Software architecture: the hard parts : modern trade-off analyses for distributed architectures. Beijing: O'Reilly, 2021. ISBN 978-1492086895.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jan Vanke Centrum znalostního managementu FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **26.02.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **15.02.2026**

Ing. Jan Vanke podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. | Datum převzetí zadání

Podpis studenta

Poděkování

Chtěl bych vyjádřit poděkování mému vedoucímu Ing. Janu Vankemu za věcné připomínky, vstřícnost a odborné vedení mojí práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 24. května 2024

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací systému pro verzování BPMN 2.0 diagramů. Cílem je vytvořit webovou aplikaci, která umožní uživatelům spravovat, verzovat a spolupracovat na BPMN diagramech. Na základě analýzy požadavků na verzovací systém pro správu BPMN modelů a porovnání existujících řešení byly definovány požadavky a proveden technický návrh systému. Systém využívá technologie React pro frontend, Spring Boot pro backend a PostgreSQL pro databázi. Byl vytvořen prototyp aplikace, který byl podroben průběžnému a závěrečnému testování. Po vyhodnocení testů byly navrženy možné změny a vylepšení aplikace pro její další rozvoj.

Klíčová slova: BPMN, verzovací systém, podnikové procesy, modelování, správa verzí, React, Spring Boot

Vedoucí: Ing. Jan Vanke

Abstract

This bachelor thesis deals with the design and implementation of a system for versioning BPMN 2.0 diagrams. The goal is to create a web application that allows users to manage, version, and collaborate on BPMN diagrams. Based on an analysis of the requirements for a versioning system for managing BPMN models and a comparison of existing solutions, the requirements were defined and the technical design of the system was carried out. The system uses React for the frontend, Spring Boot for the backend, and PostgreSQL for the database. A prototype application was created and subjected to continuous and final testing. After the evaluation of the tests, possible changes and improvements to the application were proposed for its further development.

Keywords: BPMN, versioning system, business processes, modeling, version control, React, Spring Boot

Title translation: Versioning system for BPMN models

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Nejprve si představíme motivaci, hlavní cíle a strukturu této práce.

1.1 Motivace

Procesní řízení je nezbytnou součástí každého podniku a je nutné jej umět správně uchopit. Schopnost efektivně spravovat procesní modely a využít jejich potenciál v plné výši je tedy nezbytnou součástí každé vyvíjející se entity. Aby bylo možné efektivně spravovat procesní modely, je nutné je verzovat. To umožňuje se vracet ke starším verzím modelu a předejít tak chybám. Z tohoto důvodu je nezbytné mít speciální nástroj, který zvládne spravovat procesní diagramy a zároveň je verzovat, aby uživatelé měli k dispozici veškerá data z vývoje daného procesu v čase.

1.2 Cíle práce

Cílem této práce je vyvinout prototyp systému pro správu a verzování BPMN 2.0 diagramů. V rámci toho porovnáme existující řešení na trhu, zanalyzujeme požadavky na takový systém a následně jej navrhneme, implementujeme a důkladně otestujeme dle současných standardů.

1.3 Struktura práce

Práce je rozdělena do několika kapitol reprezentující softwarový cyklus. V první kapitole si představíme základní teorii k BPMN notaci k nutnému pochopení toho, o čem se budeme celou dobu bavit. Následně na to navážeme průzkumem existujících řešení a jejich porovnání. Na základě těchto znalostí si nadefinujeme požadavky pro nový systém určený k verzování procesních BPMN 2.0 modelů v kapitole zabývající se analýzou a tuto analýzu přetavíme v návrh a implementaci takového systému v kapitolách navazujících. Prototyp takového systému poté podrobíme testování a o něm si řekneme v předposlední kapitole. A na závěr, v poslední kapitole, zhodnotíme splnění cílů a budoucí potenciální vývoj.

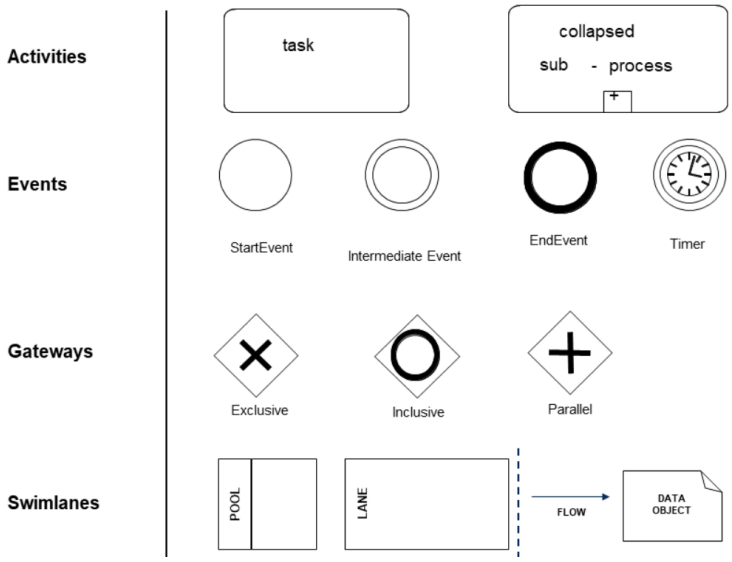
Kapitola 2

Teoretický základ

2.1 BPMN

Business Process Model and Notation, zkráceně BPMN, je grafická notace určená k modelování podnikových procesů, jejich analýze a specifikaci. Účel této notace je poskytnout jednotný způsob znázornění obchodních procesů pomocí vizuální posloupnosti činností a toků ve správném pořadí za sebou. Díky tomu je možné znázornit jakýkoliv proces pochopitelnou a jasnou formou pro jeho uživatele bez ohledu na technické znalosti nebo pochopení daného problému. To má za následek zlepšení komunikace mezi všemi zainteresovanými stranami. Řeší se tím tedy nejednoznačnost textových specifikací procesů. Nejnovější verze (BPMN 2.0.2) byla oficiálně zveřejněna organizací ISO jako norma pod číslem ISO/IEC 19510.[4] [5] [6]

BPMN 2.0 v dnešní době obsahuje velkou škálu elementů, které lze využít k modelování diagramů. My si nyní představíme tyto základní, které nalezneme v téměř každém diagramu, a přiblížíme si je podrobněji: [1]



Obrázek 2.1: Příklad BPMN notace [12]

- **Události (Events)** - znázorněny kruhem, můžeme je dále rozdělit na startovací (Start Event), mezilehlé (Intermediate Event) nebo koncové události (End Event).
- **Činnosti (Activities)** - znázorněny obdélníkem s oblými rohy, můžeme je dále rozdělit na úkoly (Task), podprocesy (Sub-Process) nebo volanou činnost (Call Activity).
- **Brány (Gateways)** - znázorněny kosočtvercem, můžeme je dále rozdělit na exkluzivní (Exclusive Gateway), paralelní (Parallel Gateway), inkluzivní (Inclusive Gateway) nebo založenou na událostech (Event-based Gateway).
- **Spojovací prvky (Connecting Objects)** - znázorněny čarou, můžeme je dále rozdělit na sekvenční (Sequence Flow), zprávové (Message Flow) nebo asociace (Association).
- **Plavecké dráhy (Swimlanes)** - znázorněny obdélníkovým boxem, můžeme je dále rozdělit na bazény (Pool) a dráhy (Lane).

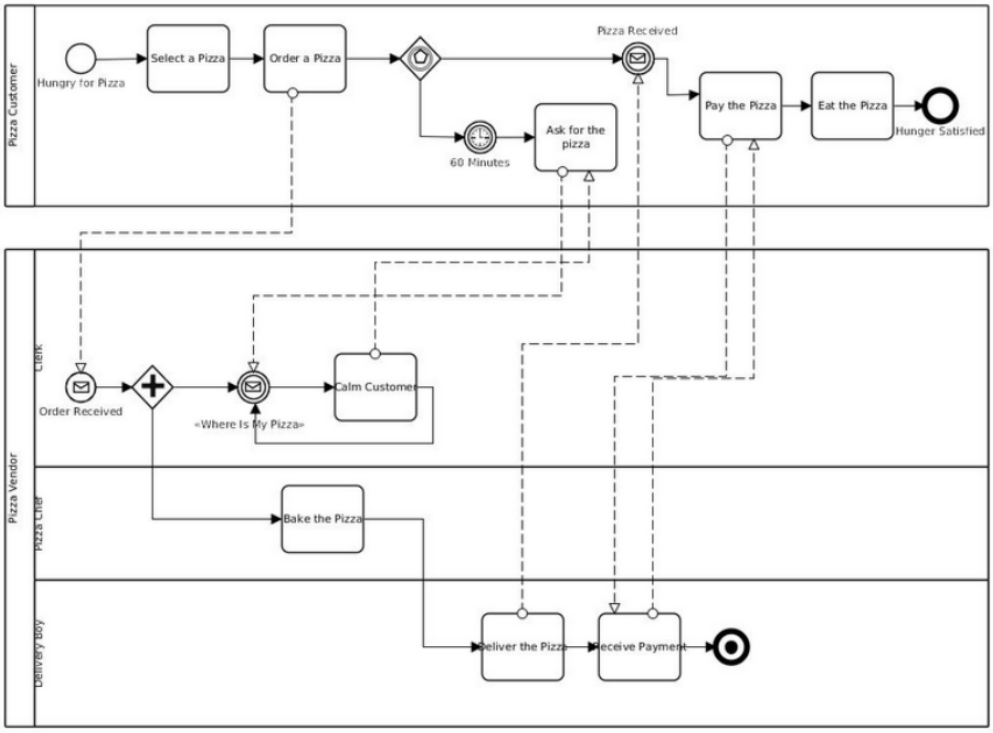
Každý proces začíná vždy **událostí**. Událost reprezentuje to, že došlo k něčemu na začátku, v průběhu nebo na konci procesu, a my na to reagujeme. To znamená, že nám například doručili pizzu, což může být začátek našeho konkrétního procesu, a my poté pizzu nakrájíme a zkonzumujeme.

Jednotlivé úkony, které následují po doručení pizzy, jsou **činnosti**, které vyžadují konkrétní akci. Jedná se konkrétně o **úkoly** nebo o **podprocesy**. Činnost si tedy můžeme představit jako nakrájení pizzy nebo její konzumaci po doručení.

V průběhu procesu může dojít k situaci, kdy je nutné se rozhodnout mezi více variantami nebo reagovat na nějakou událost. K tomu slouží **brány**. Tyto brány se používají k řízení a modifikaci sekvenčního toku a mohou buď rozdělovat, nebo slučovat toky. Existuje několik druhů bran podle jejich využití. V našem procesu s pizzou bychom bránu mohli využít například při rozhodování, jestli pizzu nakrájíme na čtyři dílky nebo jen na dva dílky, podle počtu osob, které ji plánují zkonzumovat.

Aby všechny tyto události, činnosti a brány na sebe správně navazovaly, musíme využít **spojovací prvky**, které slouží k vedení toku a specifikaci vztahů mezi jednotlivými prvky. Díky spojovacím prvkům můžeme napojit událost na počátku procesu s činnostmi, které chceme vykonat, a specifikovat jejich pořadí. To nám zajišťuje správné navázání činností a událostí a jejich posloupnost v čase.

Části procesu můžeme také rozdělit do tzv. **plaveckých drah**, které reprezentují jednotlivé účastníky procesu. To znamená, že v případě pizzy bychom mohli rozdělit jednotlivé účastníky procesu na restauraci, doručovatele a zákazníka. Restaurace pizzu upeče, doručovatel ji doručí a zákazník ji následně zkonzumuje. V každé plavecké dráze by byly jednotlivé úkony prováděné konkrétním účastníkem, což nám pomůže lépe reprezentovat skutečný proces, který se odehrává při objednání pizzy zákazníkem. [2]



Obrázek 2.2: Příklad BPMN diagramu [25]

2.2 Historie BPMN

BPMN byla původně vyvinuta v roce 2004 organizací Business Process Management Initiative (BPMI). O rok později se BPMI spojila s Object Management Group (OMG), což bylo konsorcium pro standardy v počítačovém průmyslu, a společně vydaly dokument, který popisoval notaci a sémantiku BPMN, včetně osvědčených postupů pro tvorbu BPMN diagramů. To umožnilo lepší koordinaci a rozvoj BPMN. V roce 2011 poté OMG vydala specifikaci BPMN 2.0, která nabídla podrobnější standard pro návrh obchodních procesů. Ta zahrnovala bohatší sadu symbolů a notací pro tvorbu diagramů obchodních procesů, standardizaci XML formátu pro ukládání a výměnu BPMN diagramů a přidání prvků pro podporu webových služeb, což napomohlo k celosvětové adaptaci a standardizaci v rámci ISO norem. [13]

2.3 Verzování

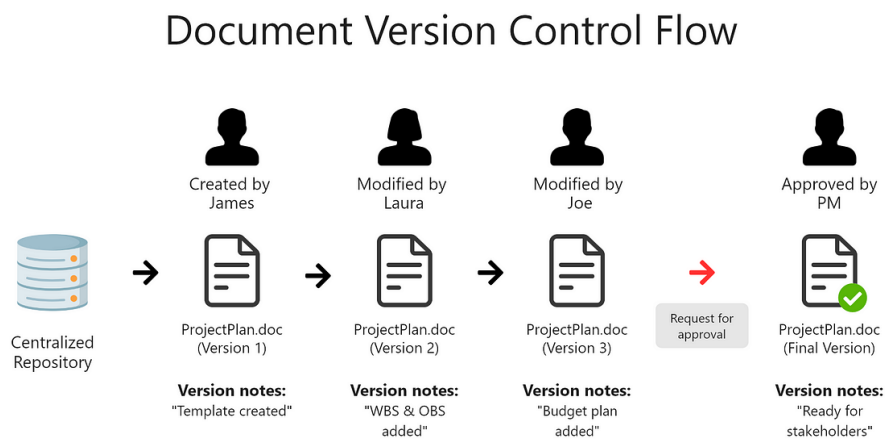
Verzování je klíčovým konceptem ke spravování dat a dokumentů tak, aby bylo možné sledovat jejich změny v průběhu času. Účel verzovacích systémů je zaznamenávat všechny provedené změny a ukládat je do jednotlivých verzí,

ke kterým můžeme následně přistupovat a využívat je v budoucnu. [16]

2.3.1 Principy verzování

Verzování v softwarovém inženýrství stojí na několika základních principech: [18]:

- Každá změna provedená v dokumentu nebo modelu je zaznamenána jako nová verze.
- Každá verze je unikátně identifikována pomocí verzovacích čísel nebo hashů.
- Verzovací systém uchovává historii všech verzí.
- Uživatelům je umožněno vizuálně porovnávat různé verze a identifikovat změny mezi nimi.
- Je možné se vrátit se k předchozím verzím v případě potřeby.



Obrázek 2.3: Průběh řízení verzí [19]

2.3.2 Význam verzování v BPMN modelech

Verzování u BPMN diagramů slouží k zajištění větší kontroly nad správou procesů, protože umožňuje organizacím efektivně sledovat a řídit změny v jejich obchodních procesech. Verzování u procesů tedy v ideálním případě je řízeno těmito principy: [1]

- Verzování umožňuje sledovat změny diagramu v čase, což zajišťuje, že všechny úpravy jsou zaznamenány a mohou být zpětně analyzovány.

- Verzování pomáhá udržet konzistenci mezi různými verzemi diagramu, což je zásadní pro zachování integrity.
- Verzovací systémy uchovávají kompletní historii změn, což zamezuje ztrátě jakýchkoliv předešlých dat
- Vytvářejí možnost se vrátit k předchozím verzím bez zbytečné prodlevy.

2.4 Závěr

V úvodní kapitole jsme si představili pojmy a koncepty týkající se notace BPMN a verzování. Představili jsme si základní prvky BPMN a význam tohoto standardu v podnikových procesech. Také jsme si přiblížili základní principy verzování, což by mělo přinést přehled o tom, čím se bakalářská práce bude zabývat. Když jsme si stanovili teoretický základ, můžeme přistoupit k porovnání existujících řešení pro správu BPMN digramů.

Kapitola 3

Porovnání existujících nástrojů

3.1 Přehled nástrojů pro BPMN modelování

V této kapitole se zaměříme na porovnání existujících nástrojů pro modelování a verzování BPMN modelů. Zhodnotíme dostupné nástroje s důrazem na jejich funkce, schopnosti verzování, cenové plány.

3.1.1 Camunda BPM

Přehled

Camunda BPM je open-source nástroj s placenou enterprise verzí. Tento nástroj je zaměřen na orchestraci procesů, automatizaci workflow a optimalizaci podnikových procesů. [54]

Funkce

Camunda BPM nabízí širokou škálu funkcí, včetně podpory pro BPMN 2.0, CMMN pro správu případů, DMN pro automatizaci rozhodování a real-time monitoring procesů. Díky svému open-source jádru je možné jej integrovat s různými nástroji a systémy.

Verzování

Camunda poskytuje základní verzování BPMN modelů, umožňuje ukládání různých verzí modelů, sledování změn a návrat k předchozím verzím. Pokročilé funkce, jako vizuální porovnávání verzí, nejsou k dispozici v základní verzi.

Výhody a nevýhody

- **Výhody:**
  - Robustní podpora pro BPMN
  - Široká integrace a otevřené zdrojové jádro
  - Aktivní komunita

- **Nevýhody:**
  - Strmější křivka učení
  - Pokročilé funkce dostupné pouze v placené verzi

Cena

Camunda je dostupná zdarma v open-source verzi, ale enterprise verze, která nabízí více pokročilých funkcí a podporu, je placená. Ceny závisí na konkrétních požadavcích a rozsahu implementace.

3.1.2 Signavio Process Manager

Přehled

Signavio Process Manager je placený nástroj, který nabízí pokročilé funkce pro modelování, analýzu a verzování BPMN procesů. Tento nástroj je vhodný pro organizace, které potřebují silné analytické a verzovací schopnosti produktu. [55]

Funkce

Signavio umožňuje spolupráci více uživatelů, nabízí nástroje pro optimalizaci procesů a podporuje pokročilé funkce pro analýzu a vizualizaci procesů.

Verzování

Signavio Process Manager vyniká ve verzování, nabízí vizuální porovnávání verzí, detailní historii změn a možnost návratu k předchozím verzím. Uživatelé mohou snadno sledovat a analyzovat změny v BPMN modelech.

Výhody a nevýhody

- **Výhody:**
  - Pokročilé analytické nástroje
  - Uživatelsky přívětivé rozhraní
  - Silné verzovací funkce
- **Nevýhody:**
  - Vysoké náklady, které mohou být překážkou pro menší společnosti.

Cena

Signavio Process Manager je dostupný pouze jako placený nástroj, s různými cenovými plány závislými na rozsahu použití a požadovaných funkcích.

3.1.3 Bizagi Modeler

Přehled

Bizagi Modeler je nástroj, který je k dispozici zdarma s placenými prémiovými funkcemi. Je známý svými intuitivními vizuálními nástroji pro BPMN modelování. [56]

Funkce

Bizagi Modeler umožňuje vytváření, správu a dokumentaci BPMN modelů. Nabízí základní nástroje pro mapování procesů a jejich verzování.

Verzování

Bizagi poskytuje základní verzovací funkce, jako je ukládání a sledování verzí modelů. Pokročilé funkce jako vizuální porovnávání verzí nejsou dostupné v bezplatné verzi.

Výhody a nevýhody

- **Výhody:**
  - Snadné použití
  - Základní funkce zdarma
  - Vhodné pro začátečníky
- **Nevýhody:**
  - Omezené pokročilé funkce v bezplatné verzi
  - Prémiové funkce jsou dostupné pouze za poplatek.

Cena

Bizagi Modeler je zdarma, ale prémiové funkce, které zahrnují verzovací schopnosti a integrace, jsou dostupné v placené verzi.

3.1.4 Sparx Systems Enterprise Architect

Přehled

Enterprise Architect od společnosti Sparx Systems je placený nástroj, který nabízí širokou škálu funkcí pro modelování a správu BPMN modelů. [57]

Funkce

Tento nástroj umožňuje detailní modelování procesů, správu verzí a podporuje spolupráci mezi uživateli. Poskytuje také nástroje pro analýzu a vizualizaci procesů.

Verzování

Enterprise Architect poskytuje robustní verzovací funkce, včetně detailní historie verzí, vizuálního porovnávání a možnosti návratu k předchozím verzím. Uživatelé mohou snadno sledovat změny a analyzovat jejich dopad na modely.

Výhody a nevýhody

- **Výhody:**
  - Široká škála modelovacích schopností
  - Robustní verzování
  - Podpora spolupráce
- **Nevýhody:**
  - Vyšší cena
  - Strmější křivka učení

Cena

Sparx Systems Enterprise Architect je dostupný jako placený nástroj, s licencemi začínajícími od $229 na uživatele, v závislosti na verzi a funkcích.

3.1.5 IBM Blueworks Live

Přehled

IBM Blueworks Live je placené cloudové řešení pro modelování a správu BPMN procesů, které podporuje verzování a spolupráci v reálném čase. [58]

Funkce

Blueworks Live umožňuje modelování, analýzu a správu BPMN procesů. Poskytuje nástroje pro kolaborativní práci a integraci s dalšími nástroji a systémy.

Verzování

IBM Blueworks Live nabízí pokročilé verzovací schopnosti, včetně sledování změn, vizuálního porovnávání verzí a možnosti návratu k předchozím verzím. Uživatelé mohou snadno sledovat vývoj modelů a analyzovat změny.

Výhody a nevýhody

- **Výhody:**
  - Cloudové řešení
  - Silná podpora spolupráce
  - Pokročilé verzovací funkce
- **Nevýhody:**
  - Vysoká cena
  - Komplexnost implementace

Cena

IBM Blueworks Live je dostupný jako placený nástroj, s měsíčním předplatným začínajícím od $56 na uživatele.

3.1.6 Cawemo

Přehled

Cawemo byl SaaS nástroj vyvinutý společností Camunda pro spolupráci při modelování BPMN procesů. Tento nástroj byl navržen tak, aby umožnil týmům obchodních analytiků, vývojářů a produktových manažerů společně vytvářet, recenzovat a modelovat obchodní procesy na jedné platformě zdarma. [62]

Funkce

Cawemo nabízelo širokou škálu funkcí pro BPMN modelování, včetně:

- Možnosti spolupráce uživatelé v reálném času a možnosti komentovat jednotlivé BPMN modely.
- Integrování dalšími nástroji z Camunda stacku, jako je Camunda Modeler a Camunda Runtime Platform, což umožňovalo synchronizaci modelů mezi těmito nástroji.
- Export a import modelů mnoha způsoby, včetně možnosti přímého nahrání z Camunda Modeleru.

Verzování

Cawemo podporovalo verzovací funkce, které umožňovaly sledování změn a návrat k předchozím verzím modelů. Tato funkcionalita byla zásadní pro týmy, které potřebovaly udržovat přehled o vývoji svých BPMN modelů.

Historie a ukončení provozu

Cawemo bylo spuštěno v roce 2017 a rychle si získalo popularitu. V roce 2024 však společnost Camunda oznámila, že ukončí podporu pro Cawemo k 30. dubnu 2024. Tento krok byl součástí strategického zaměření společnosti na integraci klíčových funkcionalit Cawema přímo do Camunda Platform 8. Tento krok umožnil uživatelům využívat pokročilé kolaborační funkce Cawema přímo v rámci širšího ekosystému Camunda, čímž se zjednodušila správa a nasazení BPMN modelů pro organizace, avšak s omezeními v rámci Camundy pro neplatíci uživatele původního Cawema. [63]

3.2 Závěr

Na základě porovnání nástrojů jsme prošli širokou škálu řešení pro různé potřeby a rozpočty organizací. Porovnání nástrojů pro BPMN modelování ukázalo širokou škálu řešení pro různé potřeby a rozpočty organizací. Zatímco Camunda BPM a Bizagi Modeler nabízejí základní funkce zdarma, pokročilé nástroje jako Signavio Process Manager, Sparx Systems Enterprise Architect a IBM Blueworks Live jsou vhodné pro větší organizace s robustními požadavky. Každý nástroj má své výhody a nevýhody: Camunda BPM vyniká integrací, ale pokročilé funkce jsou placené; Signavio Process Manager je silný a výkonný nástroj, ale drahý; Bizagi Modeler je uživatelsky přívětivý, ale omezený v pokročilých funkcích zdarma. IBM Blueworks Live a Sparx Systems Enterprise Architect nabízejí pokročilé funkce, ale jsou nákladné a složité na implementaci. Cawemo, významný nástroj pro kolaborativní modelování a verzování moderlů, byl v roce 2024 ukončen a jeho klíčové funkce byly integrovány do Camunda Platform 8. Tato situace vytváří prostor pro vývoj nové aplikace s moderním a efektivním řešením pro BPMN modelování a verzování, které splní potřeby současných uživatelů a nabídne platformu s možností optimalizace obchodních procesů s minimálními náklady pro koncové uživatele.

Kapitola 4

Analýza

Analýza je první fází softwarového procesu, kde se klade důraz na pochopení požadavků a potřeb uživatelů. Tato fáze je klíčová pro úspěšné plánování a vývoj softwarového projektu, neboť definuje základní rámec, na kterém bude projekt postaven. Důkladná analýza pomáhá identifikovat doménu, klíčové funkční a nefunkční požadavky, příklady užití a zároveň poskytuje jasný směr pro návrh a implementaci řešení.

V předchozí části této práce jsme provedli podrobné porovnání existujících nástrojů pro BPMN modelování a verzování. Identifikovali jsme klíčové funkce, výhody a nevýhody jednotlivých nástrojů a zhodnotili jejich vhodnost pro různé typy organizací. Na základě tohoto porovnání a poptávky uživatelů jsme schopni definovat požadavky pro nový verzovací systém pro BPMN modely, který nahradí ukončený nástroj Cawemo a bude alternativou pro verzování BPMN diagramů organizacím s minimálními náklady.

4.1 Specifikace požadavků

Specifikace požadavků je klíčovým krokem analýzy. Cílem je definovat veškeré funkční a nefunkční požadavky, které musí systém splňovat. Tato fáze je zásadní pro zajištění toho, že konečný produkt bude vyhovovat potřebám uživatelů a poskytovat očekávané funkce a vlastnosti. Bylo by totiž nežádoucí vyvíjet produkt, který nesplňuje požadavky koncových uživatelů, protože takový produkt by neměl žádné opodstatnění.

4.1.1 Účel aplikace

Účelem aplikace je poskytovat robustní a intuitivní nástroj pro verzování BPMN 2.0 modelů s minimálními náklady pro koncové uživatele.

4.1.2 Základní funkce aplikace

Aplikace bude umožňovat:

- Tvorbu a správu BPMN diagramů.
- Správu verzí BPMN diagramů.

- Kolaborativní práci na BPMN diagramech.
- Import a export BPMN diagramů.
- Uživatelsky přívětivé rozhraní pro správu projektů a procesních diagramů.

4.1.3 Budoucí uživatelé

Aplikace je určena pro byznys analytiky, procesní inženýry, projektové manažery a IT specialisty, kteří potřebují modelovat, upravovat nebo spravovat BPMN procesy.

Tito uživatelé mohou využít aplikaci k vytváření detailních procesních diagramů, analyzování procesních toků a optimalizaci podnikových operací. Byznys analytici budou moci aplikaci využít k dokumentování a vylepšování procesů, zatímco procesní inženýři a projektoví manažeři budou schopni sledovat a řídit změny v procesech. IT specialisté budou mít nástroj, který jim umožní integrovat procesní modely do technických implementací a zajišťovat jejich správnou funkčnost.

4.1.4 Uživatelské rozhraní

Aplikace bude realizován jako webová aplikace přístupná z webového prohlížeče. Uživatelské rozhraní musí být intuitivní, snadno použitelné a poskytovat všechny potřebné funkce pro správu BPMN modelů a verzí.

4.1.5 Komunikační rozhraní

Aplikace bude komunikovat standardními dotazy mezi frontendem a backendem, kromě části obsahují modelování procesů. Tam bude umožněno uživatelům kolaborativně spolupracovat s ostatními členy pomocí komunikace v reálném čase.

4.1.6 Byznysová pravidla

Aplikace bude implementovat řadu byznysových pravidel, jako je správa uživatelských rolí (projektový admin, editor, nečlen projektu) a omezení přístupů k projektům a diagramům na základě těchto rolí.

4.1.7 Funkční požadavky

Funkční požadavky definují specifické chování nebo funkce systému. V této časti si popíšeme klíčové funkční požadavky, které byly identifikovány na základě analýzy konkurenčních platforem, poptávky uživatelů a vývoje ve světě modelování BPMN modelů. Funkční požadavky jsou rozděleny podle zaměření do následujících kategorií

- **byznysové** – požadavky týkající se činností zadavatele,
- **uživatelské** – požadavky z pohledu budoucího uživatele systému,

- **systémové** – požadavky na technologie a implementaci.

Pro specifikaci priorit jednotlivých požadavků využijeme metodu MoSCoW [64]:

- **must have** – (musí mít),
- **should have** – (měl by mít),
- **could have** – (mohl by mít).

FR-1: Registrace nového uživatele

Systém umožní registraci nového uživatele.
Priorita: musí mít
Typ: uživatelský

FR-2: Přihlášení uživatele

Systém umožní přihlášení uživatele.
Priorita: musí mít
Typ: uživatelský

FR-3: Vytváření nových projektů

Systém umožní vytváření nových projektů.
Priorita: musí mít
Typ: uživatelský

FR-4: Přejmenování projektů

Systém umožní přejmenování projektů.
Priorita: měl by mít
Typ: uživatelský

FR-5: Mazání projektů

Systém umožní mazání projektů.
Priorita: musí mít
Typ: uživatelský

FR-6: Odchod uživatele z projektu

Systém umožní odchod uživatele z projektu.
Priorita: měl by mít
Typ: uživatelský

FR-7: Přidávání spolupracovníků do projektu

Systém umožní přidání spolupracovníka do projektu.
Priorita: měl by mít
Typ: uživatelský

FR-8: Vytváření BPMN diagramů

Systém umožní vytváření BPMN diagramů v rámci projektů.
Priorita: musí mít
Typ: uživatelský

FR-9: Přejmenování BPMN diagramů

Systém umožní přejmenování BPMN diagramů.
Priorita: měl by mít
Typ: uživatelský

FR-10: Mazání BPMN diagramů

Systém umožní mazání BPMN diagramů.
Priorita: musí mít
Typ: uživatelský

FR-11: Stahování diagramů

Systém umožní stahování BPMN diagramů jako BPMN XML nebo SVG obrázek.
Priorita: měl by mít
Typ: uživatelský

FR-12: Duplikování diagramů

Systém umožní duplikování BPMN diagramů.
Priorita: měl by mít
Typ: uživatelský

FR-13: Rozšířený modeler

Systém poskytne rozšířený BPMN modeler s rozšiřujícím panelem.
Priorita: musí mít
Typ: uživatelský

FR-14: Vytváření revizí diagramů

Systém umožní vytváření revizí BPMN diagramů.
Priorita: musí mít
Typ: uživatelský

FR-15: Zobrazení historie revizí

Systém poskytne stránku historie revizí s možností prohlížení, přejmenování, mazání a porovnávání revizí.
Priorita: musí mít
Typ: uživatelský

FR-16: Obnova revize BPMN diagramu

Systém umožní obnovu revize do modeleru nebo jako nový diagram.
Priorita: měl by mít
Typ: uživatelský

FR-17: Uživatelský profil a změna hesla

Systém umožní zobrazit a upravit uživatelský profil a změnit heslo.
Priorita: měl by mít
Typ: uživatelský

FR-18: Automatické odhlášení po neaktivitě

Systém automaticky odhlásí uživatele po třech hodinách neaktivity.
Priorita: měl by mít
Typ: systémový

FR-19: Uživatelsky přívětivé rozhraní

Systém poskytne uživatelsky přívětivé a intuitivní rozhraní s navigačním panelem.
Priorita: musí mít
Typ: uživatelský

FR-20: Kolaborativní práce na BPMN diagramu

Systém umožní kolaborativní práci více uživatelů na stejném BPMN diagramu.
Priorita: musí mít
Typ: uživatelský

FR-21: Chat k BPMN diagramu

Systém poskytne chatovou funkci k BPMN diagramu pro snadnou komunikaci mezi uživateli.
Priorita: měl by mít
Typ: uživatelský

FR-22: Specifikace k BPMN diagramu

Systém umožní uživatelům přidávat a upravovat specifikace k BPMN diagramu.

Priorita: měl by mít
Typ: uživatelský

4.1.8 Nefunkční požadavky

Nefunkční požadavky popisují kritéria, která lze použít k posouzení provozu systému, spíše než konkrétní chování. Tyto požadavky jsou klíčové pro zajištění celkové kvality systému a jeho schopnosti vyhovět očekáváním uživatelů a technickým omezením. Následující část popisuje nefunkční požadavky identifikované na základě analýzy a specifických potřeb uživatelů.

NFR-1: Výkon systému

Systém musí být schopen zpracovávat požadavky uživatelů v reálném čase s minimální latencí.

Priorita: musí mít
Typ: technický

NFR-2: Bezpečnost dat

Systém musí zajišťovat bezpečnost uživatelských dat pomocí zabezpečeného přístupu.

Priorita: musí mít
Typ: technický

NFR-3: Uživatelské rozhraní

Systém musí poskytovat intuitivní a uživatelsky přívětivé rozhraní, které je snadno použitelné pro nové i zkušené uživatele.

Priorita: musí mít
Typ: technický

4.2 Případy užití

Příklady užití představují konkrétní scénáře, ve kterých uživatelé interagují se systémem k dosažení konkrétních cílů. Tyto scénáře pomáhají ilustrovat funkční požadavky a ověřit, zda systém splňuje očekávání uživatelů. Jeden funkční požadavek je obvykle dosažen mnoha případy užití. Aktéry jsou například registrovaní uživatelé a členové projektu, což je buď Project Admin, nebo Editor.

UC-1 Registrace nového uživatele

Nový uživatel se může zaregistrovat do systému, aby získal přístup k aplikaci.
Aktér: nový uživatel
Hlavní scénář:

1. Uživatel vyplní registrační formulář s požadovanými údaji.
2. Systém zkontroluje správnost vyplněných údajů.
3. Systém vytvoří nový uživatelský účet.
4. Systém odešle potvrzovací e-mail uživateli.
5. Uživatel potvrdí registraci kliknutím na odkaz v e-mailu.

Alternativní scénář:

1. Uživatel vyplní registrační formulář s požadovanými údaji.
2. Systém zjistí, že některé údaje jsou neplatné nebo chybí.
3. Systém zobrazí uživateli chybovou zprávu a požádá o opravu údajů.
4. Uživatel opraví údaje a odešle formulář znovu.
5. Systém zkontroluje správnost vyplněných údajů a pokračuje podle hlavního scénáře.

UC-2 Přihlášení uživatele

Uživatel se přihlásí do systému pomocí svých přihlašovacích údajů.
Aktér: registrovaný uživatel
Hlavní scénář:

1. Uživatel vyplní přihlašovací formulář (e-mail a heslo).
2. Systém ověří správnost přihlašovacích údajů.
3. Systém přihlásí uživatele do aplikace.
4. Systém zobrazí hlavní stránku aplikace.

Alternativní scénář:

1. Uživatel vyplní přihlašovací formulář (e-mail a heslo).
2. Systém zjistí, že přihlašovací údaje jsou nesprávné.
3. Systém zobrazí uživateli chybovou zprávu a požádá o opravu údajů.
4. Uživatel opraví údaje a odešle formulář znovu.
5. Systém ověří správnost přihlašovacích údajů a pokračuje podle hlavního scénáře.

UC-3 Vytvoření projektu

Uživatel může vytvořit nový projekt pro práci s BPMN diagramy.
Aktér: registrovaný uživatel
Hlavní scénář:

1. Uživatel klikne na tlačítko pro vytvoření nového projektu.
2. Systém zobrazí formulář pro zadání názvu projektu.
3. Uživatel vyplní název projektu a potvrdí vytvoření.
4. Systém vytvoří nový projekt a zobrazí jej uživateli.

UC-4 Přejmenování projektu

Uživatel může přejmenovat existující projekt.
Aktéři: Project Admin
Hlavní scénář:

1. Uživatel vybere projekt, který chce přejmenovat, z přehledu projektů, otevře projektové menu a zvolí možnost přejmenovat projekt.
2. Systém zobrazí pole pro zadání nového názvu.
3. Uživatel zadá nový název projektu a potvrdí změnu.
4. Systém uloží nový název projektu a zobrazí jej uživateli.

UC-5 Smazání projektu

Uživatel může smazat existující projekt.
Aktéři: Project Admin
Hlavní scénář:

1. Uživatel vybere projekt, který chce smazat, z přehledu projektů, otevře projektové menu a zvolí možnost smazat projekt.
2. Systém zobrazí potvrzovací dialog s dotazem, zda uživatel opravdu chce smazat projekt.
3. Uživatel potvrdí smazání projektu.
4. Systém smaže projekt a informuje uživatele o úspěšném smazání.

UC-6 Opuštění projektu

Uživatel může opustit projekt, ve kterém je jako spolupracovník.
Aktéři: Editor
Hlavní scénář:

1. Uživatel vybere projekt, který chce opustit, otevře projektové menu a zvolí možnost opustit projekt.

2. Systém zobrazí potvrzovací dialog s dotazem, zda uživatel opravdu chce opustit projekt.
3. Uživatel potvrdí odchod z projektu.
4. Systém odstraní uživatele ze seznamu spolupracovníků projektu a informuje jej o úspěšném odchodu.

UC-7 Vytvoření BPMN diagramu

Uživatel může vytvořit nový BPMN diagram v projektu.
Aktéři: Project Admin, Editor
Hlavní scénář - Vytvoření nového diagramu:

1. Uživatel klikne na tlačítko pro vytvoření nového diagramu.
2. Systém zobrazí formulář pro zadání názvu diagramu.
3. Uživatel vyplní název diagramu a potvrdí vytvoření.
4. Systém vytvoří nový diagram a zobrazí jej uživateli.

UC-8 Vytvoření BPMN diagramu

Uživatel může nahrát nový BPMN diagram ve formě souboru.
Hlavní scénář:

1. Uživatel klikne na tlačítko pro nahrání diagramu.
2. Systém zobrazí dialog pro výběr BPMN souboru nebo rozpozná přetažený soubor.
3. Uživatel vybere BPMN soubor ze svého zařízení a potvrdí nahrání.
4. Systém nahraje vybraný soubor, vytvoří nový diagram na základě nahraného souboru a zobrazí jej uživateli.

UC-9 Přejmenování diagramu

Uživatel může přejmenovat existující diagram.
Aktéři: Project Admin, Editor
Hlavní scénář:

1. Uživatel vybere diagram, který chce přejmenovat, z přehledu diagramů, otevře menu diagramu a zvolí možnost přejmenovat diagram.
2. Systém zobrazí pole pro zadání nového názvu.
3. Uživatel zadá nový název diagramu a potvrdí změnu.
4. Systém uloží nový název diagramu a zobrazí jej uživateli.

UC-10 Smazání diagramu

Uživatel může smazat existující diagram.
Aktéři: Project Admin, Editor
Hlavní scénář - Smazání diagramu:

1. Uživatel vybere diagram, který chce smazat, z přehledu diagramů, otevře menu diagramu a zvolí možnost smazat diagram.
2. Systém zobrazí potvrzovací dialog s dotazem, zda uživatel opravdu chce smazat diagram.
3. Uživatel potvrdí smazání diagramu.
4. Systém smaže diagram a informuje uživatele o úspěšném smazání.

UC-11 Stahnutí diagramu

Uživatel může stáhnout diagram ve formátu XML nebo SVG.
Aktéři: Project Admin, Editor
Hlavní scénář:

1. Uživatel má otevřený diagram, který chce stáhnout.
2. Systém zobrazí možnosti stahování (XML nebo SVG soubor).
3. Uživatel vybere požadovaný formát stahování.
4. Systém vygeneruje a stáhne diagram v požadovaném formátu.

UC-12 Duplikování diagramu

Uživatel může duplikovat existující diagram.
Aktéři: Project Admin, Editor
Hlavní scénář:

1. Uživatel vybere diagram, který chce duplikovat, otevře menu diagramu a zvolí možnost duplikovat.
2. Systém vytvoří kopii diagramu se stejným obsahem.
3. Systém zobrazí nový duplikát diagramu v seznamu diagramů.

UC-13 Vytváření revizí diagramu

Uživatel může vytvořit novou revizi diagramu.
Aktéři: Project Admin, Editor
Hlavní scénář:

1. Uživatel má otevřený diagram, který chce přejmenovat.
2. Systém zobrazí aktuální název diagramu v horní části obrazovky s možností otevření menu diagramu.

3. Uživatel klikne na název diagramu, otevře menu diagramu a zvolí možnost vytvořit revizi.
4. Systém uloží aktuální stav diagramu jako novou revizi.
5. Systém zobrazí potvrzení o úspěšném vytvoření revize.

UC-14 Zobrazení historie revizí

Uživatel může zobrazit historii revizí konkrétního BPMN diagramu.
Aktéři: Project Admin, Editor
Hlavní scénář:

1. Uživatel má otevřený diagram, u kterého chce zobrazit historii.
2. Systém zobrazí aktuální název diagramu v horní části obrazovky s možností otevření menu diagramu.
3. Uživatel klikne na název diagramu, otevře menu diagramu a zvolí možnost zobrazit historii revizí.
4. Systém zobrazí stránku s historií revizí, kde zobrazí seznam všech revizí pro vybraný diagram.
5. Uživatel si může zobrazit detaily jednotlivých revizí.

UC-15 Porovnání revizí diagramu

Uživatel může porovnat dvě po sobě jdoucí revize BPMN diagramu a zobrazit změny mezi nimi.
Aktéři: Project Admin, Editor
Hlavní scénář:

1. Uživatel vybere revizi z historie revizí a klikne na tlačítko zobrazit změny.
2. Systém zobrazí rozdíly mezi aktuálně vybranou revizí a jejím předchůdcem pomocí barevného zvýraznění změn.

UC-16 Obnovení revize diagramu

Uživatel může obnovit diagram do stavu vybrané revize.
Aktéři: Project Admin, Editor
Hlavní scénář:

1. Uživatel vybere revizi z historie revizí, kterou chce obnovit.
2. Uživatel klikne na tlačítko pro zobrazení menu revize a klikne na obnovení revize jako nejnovější.
3. Systém obnoví diagram do stavu vybrané revize a zobrazí ho.

UC-17 Přidání spolupracovníka do projektu

Uživatel může přidat spolupracovníka do existujícího projektu.
Aktéři: Project Admin
Hlavní scénář:

1. Uživatel otevře projekt do kterého chce přidat spolupracovníka a klikne na přidat spolupracovníka.
2. Systém zobrazí formulář pro zadání e-mailové adresy spolupracovníka.
3. Uživatel zadá e-mailovou adresu spolupracovníka a potvrdí přidání.
4. Systém odešle pozvánku na zadanou e-mailovou adresu.
5. Spolupracovník přijme pozvánku a systém jej přidá do projektu jako Editora.

Kapitola 5
Návrh

Návrh systému je klíčovým krokem v procesu vývoje softwaru, který určuje, jak bude systém implementován a jak bude splňovat všechny definované požadavky. Tato kapitola se zaměřuje na specifikaci a zdůvodnění volby technologií, architektury, komunikace mezi jednotlivými částmi systému, datového modelu a uživatelského rozhraní.

Jednotlivé části aplikace budeme, dle standardů softwarového inženýrství, rozdělovat ve zbytku této práce hlavně na dvě části, a to na frontend a backend. Frontend je ta část aplikace, která je viditelná pouhým okem a obsahuje veškeré viditelné části aplikace s logikou zpracování jednotlivých viditelných prvků. Backend je naopak část aplikace, která viditelná není a stará se o zpracování dat, aplikační logiku a interakci s databázemi a externími službami. [34]

5.1 Technologie

5.1.1 Frontend

Frontend aplikace bude realizován pomocí knihovny React. React je populární javascriptová knihovna pro tvorbu uživatelských rozhraní, kterou vyvinula a udržuje společnost Facebook. [20]

React

React přináší mnoho výhod, které z něj činí ideální volbu pro vývoj moderní webové aplikace a to jsou zároveň důvody volby této technologie:

- React umožňuje vývoj aplikací pomocí opakovaně použitelných komponent, což zjednodušuje správu a údržbu kódu. [22]
- React používá virtuální Document Object Model (DOM), který zvyšuje výkon aplikace. Virtuální DOM minimalizuje množství změn v reálném DOM, což vede k rychlejšímu vykreslování a lepšímu výkonu. [21]
- React je kompatibilní s BPMN.js knihovnou, kterou budeme hojně využívat a představíme si ji v následující části. [35]

- React nabízí široký ekosystém, který zahrnuje další nástroje a knihovny, jako je React Router pro správu směrování nebo Redux pro správu stavů aplikace. [23]

BPMN.js

BPMN.js je JavaScriptová knihovna, která umožňuje zobrazení a editaci BPMN 2.0 diagramů přímo v prohlížeči. Tato knihovna poskytuje řadu funkcí, které jsou klíčové pro naši aplikaci, zejména v kontextu modelování a verzování BPMN diagramů. [24]

BPMN.js přináší následující výhody:

- Plná podpora standardu BPMN 2.0. [36]
- Flexibilnost a možnost přizpůsobení a rozšíření funkcí podle specifických potřeb aplikace. To zahrnuje možnost přidávání vlastních prvků nebo úprav existujících. [38]
- Nástroje pro validaci BPMN diagramů, což pomáhá uživatelům identifikovat a opravovat chyby v modelech. [37]
- Aktivní vývoj samotné knihovny a podpora komunity vývojářů. [39]

React a BPMN.js dohromady tvoří silný základ pro frontend naší aplikace na kterém můžeme postavit samotnou webovou aplikaci. Tvorba vlastního řešení pro zobrazování a editaci BPMN 2.0 by byla náročná úloha a není to předmětem této práce, proto byla zvolena tato knihovna.

5.1.2 Backend

Backend aplikace bude realizován pomocí Java Spring Boot. Spring Boot je framework pro vývoj aplikací v jazyce Java, který zjednodušuje tvorbu robustních a škálovatelných aplikací. Pro správu závislostí a sestavení projektu použijeme nástroj Maven, což je nejpopulárnější nástroj pro automatizaci a sestavování projektů v jazyce Java. [40] [41]

Spring Boot

Spring Boot přináší mnoho výhod, které z něj činí ideální volbu pro vývoj backendu moderní webové aplikace, a to jsou zároveň důvody volby této technologie:

- Množství automatických konfigurací a přednastavení, což zrychluje a usnadňuje vývoj aplikací. [42]
- Pokročilé bezpečnostní funkce. [44]
- Podpora RESTful API s jednoduchou nastavitelností a rozšiřitelností [45]

- Podpora WebSocketů k zajištění obousměrné komunikace v reálném čase.[46]
- Rozsáhlá komunita vývojářů, která přispívá k neustálému vylepšování frameworku a poskytování podpory a zdrojů. [43]

Dependencies a nástroje

Pro usnadnění vývoje backendu a zajištění robustnosti a efektivity aplikace budeme využívat následující knihovny a nástroje:

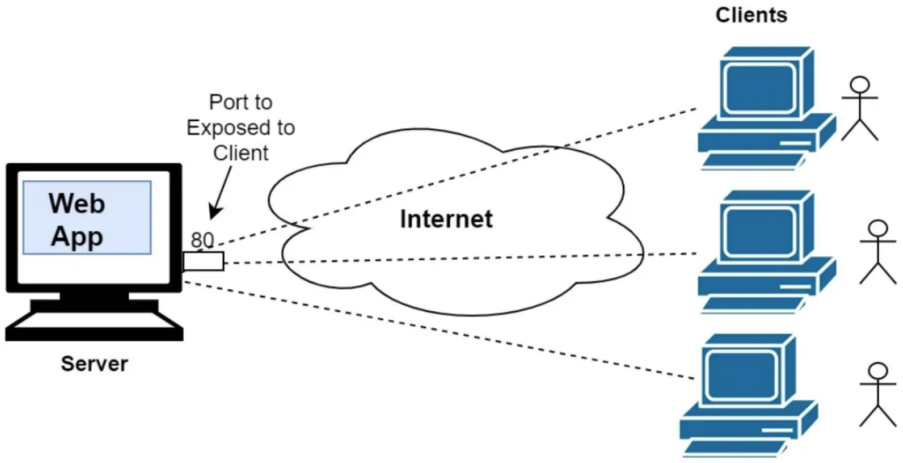
- Spring Data JPA - knihovna poskytuje snadný způsob, jak pracovat s databázemi pomocí JPA (Java Persistence API). Spring Data JPA zjednodušuje implementaci přístupu k datům pomocí repository patternu. [47]
- Spring Security - knihovna poskytuje pokročilé bezpečnostní funkce, jako je autentizace a autorizace uživatelů, ochrana proti CSRF útokům a další. [44]
- Lombok - knihovna Lombok usnadňuje práci s Java objekty tím, že automaticky generuje boilerplate kód, jako jsou gettery, settery, konstruktory a další. [48]
- Spring Web - knihovna umožňuje snadnou tvorbu RESTful webových služeb a zajišťuje komunikaci mezi frontendem a backendem. [50]
- Java JWT - tato knihovna umožňuje bezpečné vytváření a validaci JSON Web Tokenů pro autentizaci a autorizaci uživatelů. [51]
- Spring WebSocket - knihovna umožňuje použití WebSocketů k zajištění obousměrné komunikace v reálném čase. [46]
- Spring Boot DevTools - knihovna zlepšuje vývojářskou produktivitu tím, že poskytuje automatický reload aplikace při změnách kódu a další užitečné nástroje pro vývoj. [49]
- JUnit - JUnit je framework pro psaní a spouštění testů v jazyce Java, který umožňuje snadné a efektivní testování aplikace. [32]

5.1.3 Databáze

Pro naši aplikaci jsme zvolili relační databázi PostgreSQL. PostgreSQL je otevřená a vysoce výkonná databáze, která nabízí robustní sadu funkcí pro správu dat, včetně transakční integrity, podpory ACID (Atomicity, Consistency, Isolation, Durability) a pokročilých dotazovacích možností. Využití PostgreSQL zajišťuje, že naše aplikace bude schopna bezpečně a efektivně ukládat a spravovat data. [33]

5.2 Architektura

Architektura našeho systému bude založena na principu klient-server, kde frontend a backend budou oddělené, což je v dnešní době standard u webových aplikací.

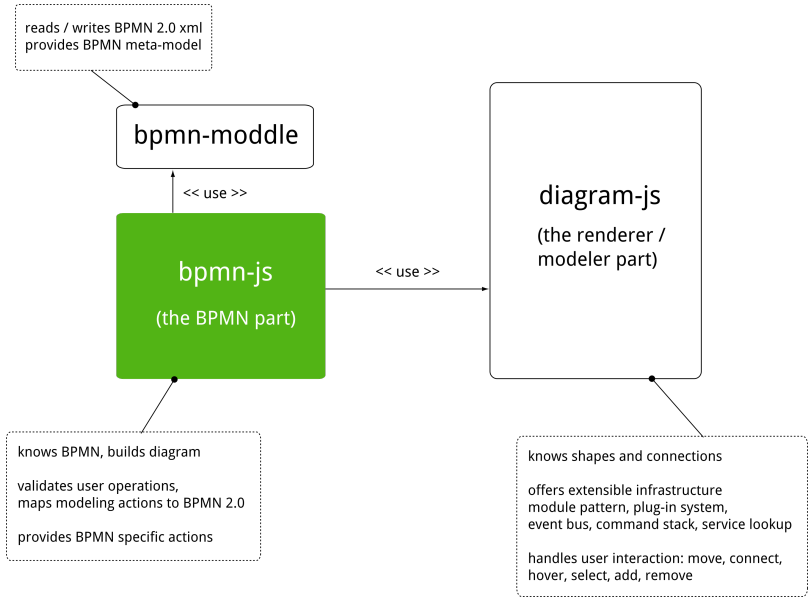


Obrázek 5.1: Klient-server architektura [26]

5.2.1 Frontend

React na frontendu aplikace bude zodpovědný za uživatelské rozhraní, interakce s uživatelem, včetně validace vstupů za účelem minimalizace zátěže serveru, a komunikaci s backendem. Využijeme možnost Reactu na tvorbu dynamických uživatelských rozhraní s komponentami a zjednodušíme si tím vývoj.

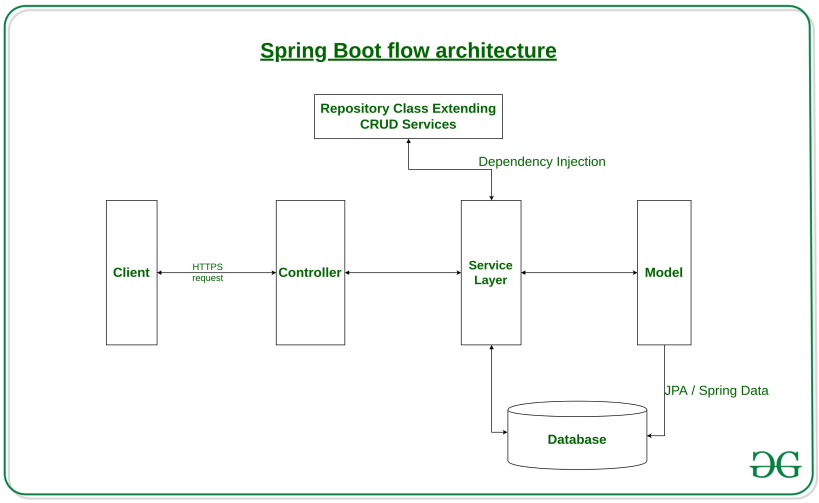
Pro modelování BPMN diagramů budeme používat komplexní knihovnu bpmn-js, která funguje jako nadstavba knihovny diagram-js. Diagram-js knihovna samotná slouží pouze k základní manipulaci s diagrami. V bpmn-js budeme využívat komponenty Modeler a Navigated Viewer, které slouží jako nástroje k modelování a zobrazování BPMN 2.0 diagramů. Na to si napojíme moduly, které nám zajistí rozšíření samotných funkcí jednotlivých komponent, jedná se například o moduly Properties Panel, Minimap nebo Grid. Knihovna bpmn-moddle zajistí validaci vstupních BPMN 2.0 xml souborů, což má i podstatný bezpečnostní rozměr, aby nedocházelo k nahrávání nežadoucích souborů. K provádění HTTP požadavků z frontendu budeme využívat knihovnu Axios, která pomocí promise-based architektury v Javascriptu zaručuje čistý a efektivní způsob zpracování požadových operací.



Obrázek 5.2: Architektura bpmn-js [36]

5.2.2 Backend

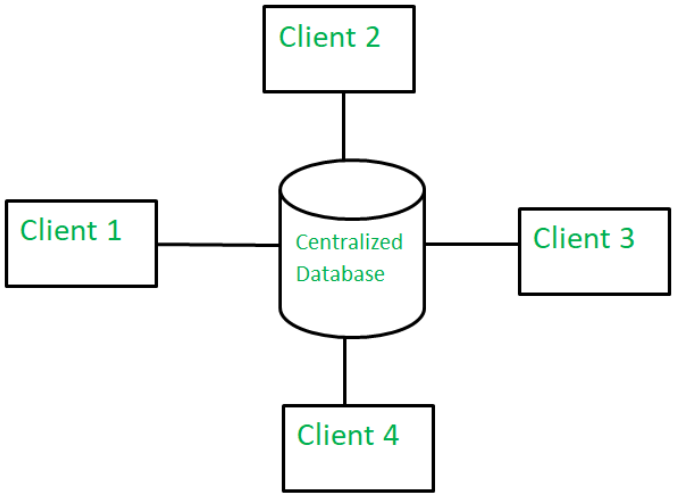
Spring Boot na backendu aplikace bude sloužit primárně k poskytování služeb a zpracování dat od klientů. V rámci služeb se bude jednat hlavně o autentizaci uživatelů a autorizaci přístupů k jednotlivým částem aplikace. S využitím RESTful API pokryjeme CRUD (Create, Read, Update a Delete) operace nad celým systém a pomocí Java Persistence API (JPA) práci s databází. Zároveň zajistíme provoz WebSocketů pro obousměrný datový tok k modelování diagramů v reálném čase, včetně logiky komprese a dekomprese pomocí GZIP funkcí a rozdělení websocket zprávy do jednotlivých částí (chunků) v případě většího objemu dat.



Obrázek 5.3: Architektura Spring Boot [27]

5.2.3 Databáze

V PostgreSQL databázi, kde budeme ukládat veškerá klientská data, využijeme centralizovanou architekturu databáze, která představuje nejpřímočařejší řešení pro webovou aplikaci těchto rozměrů. Databáze bude pokrývat všechny datové aspekty aplikace, včetně BPMN 2.0 xml data pro jednotlivé diagramy. Dle toho bude uzpůsoben datový model. viz. podkapitola 5.3.



Obrázek 5.4: Centralizovaná architektura databáze [59]

5.2.4 Komunikace

Jak už jsem zmiňoval, komunikace mezi frontendem a backendem bude zajištěna pomocí dvou hlavních technologií - RESTful API a WebSocketů.

RESTful API

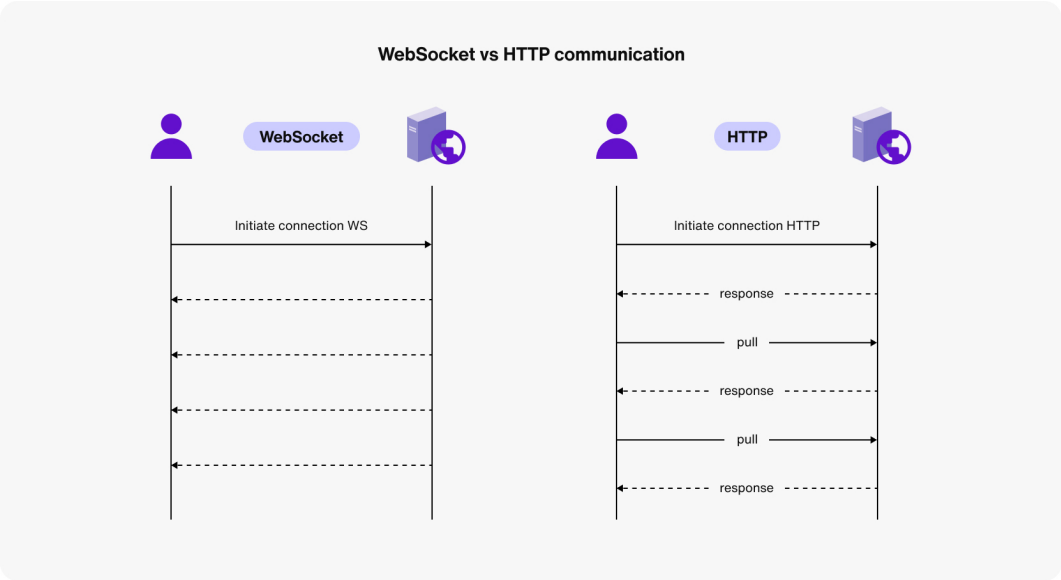
RESTful API bude implementována pomocí Spring Bootu použitím knihovny Spring Web a bude poskytovat endpoints pro všechny potřebné CRUD operace. Každý endpoint bude reprezentovat jednu operaci, jako je vytvoření, čtení, aktualizace nebo smazání entity. Tímto způsobem bude možné spravovat všechny aspekty BPMN diagramů a uživatelských dat. Na frontendu je k provádění HTTP požadavků využita knihovna Axios.

- GET - Získání informací o entitách. Například získání seznamu všech BPMN diagramů nebo detailů konkrétního diagramu.
- POST - Vytváření nových entit. Například vytvoření nového BPMN diagramu.
- PUT - Aktualizaci existujících entit. Například aktualizace existujícího BPMN diagramu.

- DELETE - Mazání entit. Například smazání konkrétního BPMN diagramu.

WebSockety

WebSockety budou použity pro přenos dat v reálném čase mezi frontendem a backendem, což umožní okamžitou synchronizaci dat a spolupráci mezi uživateli na BPMN diagramech. WebSockety budou implementovány pomocí Spring WebSocket knihovny, SockJS a STOMP protokolu. SockJS poskytuje fallback mechanizmy pro případy nedostupnosti WebSocketu a STOMP protokol zajišťuje strukturovanou komunikaci mezi frontendem a backendem. [53] [52]



Obrázek 5.5: Porovnání Websocket vs. HTTP [28]

5.2.5 Zabezpečení

Zabezpečení aplikace bude zajištěno na několika úrovních. Autentizace uživatelů bude řešena pomocí Java JWT, což umožní bezpečnou správu tokenů a autorizaci přístupů k různým částem systému. Poté bude využita knihovna BCryptPasswordEncoder, která je součást Spring Security, k šifrování hesel. Nakonfigurujeme nastavení CORS (Cross-Origin Resource Sharing), abychom omezili přístup pouze na povolené zdroje a detailně nastavíme všeobecnou konfiguraci Spring Security k správné autentizaci endpointů.

5.3 Datový model

Návrh datového modelu je založen na analýze požadavků a reprezentuje rozložení tříd pro pokrytí datových potřeb všech funkcionalit systému. Z důvodu lepšího zabezpečení jsou použity HashId identifikátory u všech entit.

5.3.1 Entity

- **Uživatel (Users)** - reprezentuje uživatele aplikace a obsahuje informace jako jméno, e-mail a role.
  - **UserId**, UUID, unikátní identifikátor uživatele.
  - **FirstName**, VARCHAR, křestní jméno uživatele.
  - **LastName**, VARCHAR, příjmení uživatele.
  - **Email**, VARCHAR, e-mailová adresa uživatele, musí být unikátní.
  - **PasswordHash**, VARCHAR, hash hesla uživatele.
  - **IsActive**, BOOLEAN, stav aktivace uživatele.
  - **Role**, VARCHAR, role uživatele v systému (USER, ADMIN).
- **Projekt (Project)** - Reprezentuje projekt, který obsahuje BPMN diagramy a informace o vlastníkovi a spolupracovnících.
  - **ProjectId**, UUID, unikátní identifikátor projektu.
  - **Name**, VARCHAR, název projektu.
  - **CreatedAt**, TIMESTAMP, datum a čas vytvoření projektu.
  - **UpdatedAt**, TIMESTAMP, datum a čas poslední aktualizace projektu.
- **Uživatel-Projekt (UserProject)** - Reprezentuje vztah mezi uživateli a projekty.
  - **UserId**, UUID, odkaz na uživatele.
  - **ProjectId**, UUID, odkaz na projekt.
  - **Role**, VARCHAR, role uživatele v projektu (OWNER, COLLABORATOR).
- **Diagram (Diagram)** - Reprezentuje BPMN diagram, který je součástí projektu.
  - **DiagramId**, UUID, unikátní identifikátor BPMN diagramu.
  - **ProjectId**, UUID, odkaz na projekt.
  - **Name**, VARCHAR, název BPMN diagramu.
  - **Data**, TEXT, XML obsah BPMN diagramu.
  - **EditedBy**, UUID, odkaz na uživatele, který diagram naposledy upravil.

- **CreatedAt**, TIMESTAMP, datum a čas vytvoření diagramu.
- **UpdatedAt**, TIMESTAMP, datum a čas poslední úpravy diagramu.
- **CreatedBy**, UUID, odkaz na uživatele, který diagram vytvořil.

- **Revize (Revision)** - Reprezentuje jednotlivé revize BPMN diagramu.
  - **RevisionId**, UUID, unikátní identifikátor revize.
  - **DiagramId**, UUID, odkaz na BPMN diagram.
  - **CreatedBy**, UUID, odkaz na uživatele, který vytvořil revizi.
  - **Name**, VARCHAR, název revize.
  - **Data**, TEXT, obsah revize.
  - **VersionNumber**, INT, číslo verze.
  - **CreatedAt**, TIMESTAMP, datum a čas vytvoření revize.
- **Specifikace (Specification)** - Reprezentuje specifikace BPMN diagramu.
  - **SpecificationId**, UUID, unikátní identifikátor specifikace.
  - **DiagramId**, UUID, odkaz na BPMN diagram.
  - **Content**, TEXT, obsah specifikace.
- **Chatová zpráva (ChatMessage)** - Reprezentuje chatové zprávy k BPMN diagramům.
  - **MessageId**, UUID, unikátní identifikátor zprávy.
  - **DiagramId**, UUID, odkaz na BPMN diagram.
  - **UserId**, UUID, odkaz na uživatele, který zprávu odeslal.
  - **Content**, TEXT, obsah zprávy.
  - **CreatedAt**, TIMESTAMP, datum a čas odeslání zprávy.

5.3.2 Integritní omezení

Pro zajištění integrity dat a správnosti vztahů je nutné použít následující omezení:

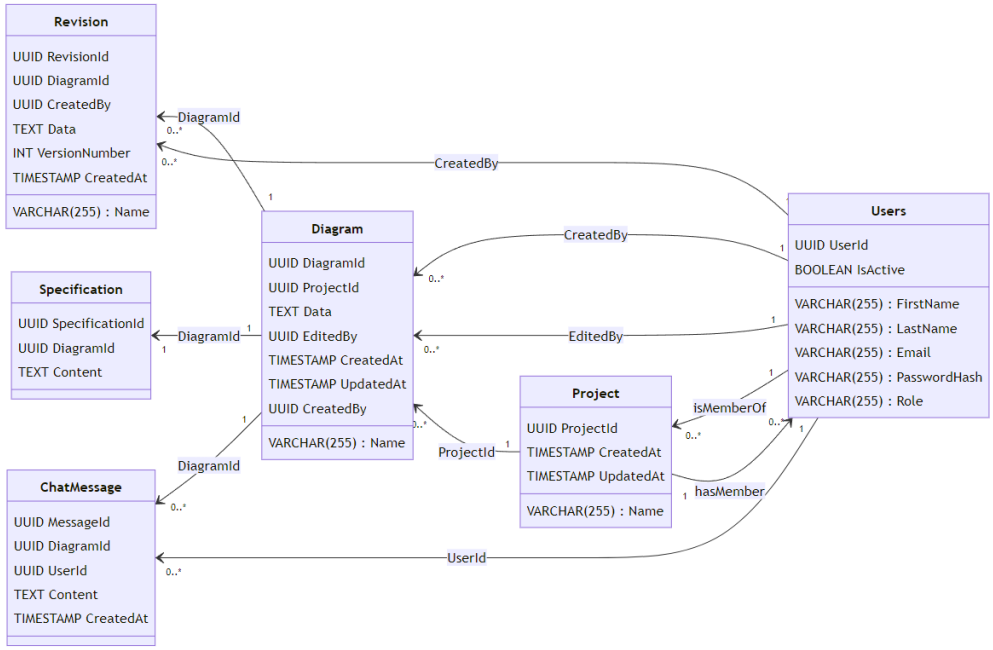
- Každá tabulka bude musí mít primární klíč, který zajišťuje unikátní identifikaci každého záznamu.
- Pro zachování referenční integrity budou použity cizí klíče mezi souvisejícími tabulkami. Když je smazán projekt, všechny přidružené diagramy, specifikace, revize a chatové zprávy budou také smazány díky kaskádovému mazání.
- Pole jako e-mail uživatele budou mít unikátní omezení, aby se zabránilo duplicitám.
- Role uživatelů a projektů budou mít check omezení pro zajištění, že budou obsahovat pouze platné hodnoty.

5.3.3 Trigger

Pro automatizaci aktualizace času u projektů po aktualizace vnořeného diagramu využijeme funkce triggeru, ktera nám zajistí aktualizaci časového razítka v tabulkách rovnou na úrovni databáze a není potřeba tuto logiku implementovat jinde.

5.3.4 Diagram datového modelu

Vizuální reprezentace datového modelu se znázorněním vztahů mezi entitami:



Obrázek 5.6: Diagram datového modelu

5.4 Uživatelské rozhraní

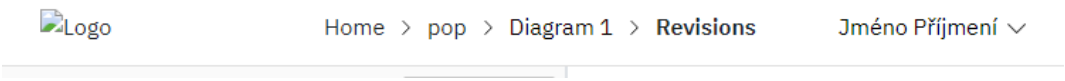
Navrhneme si základní uživatelské rozhraní jednotlivých komponent a stránek a vytvoříme dle nich wireframy.

5.4.1 Navigační panel

Navigační panel je klíčovým prvkem uživatelského rozhraní, který umožňuje uživatelům snadný přístup k hlavním sekcím aplikace. Komponenta navigačního panelu zobrazuje prvky v navigační baru dle vnoření uživatele ve webové aplikace.

- Dashboard - úvodní strana aplikace, kde uživatelé mohou vidět přehled svých projektů a aktivit, v navigačním baru reprezentována jako "Home".

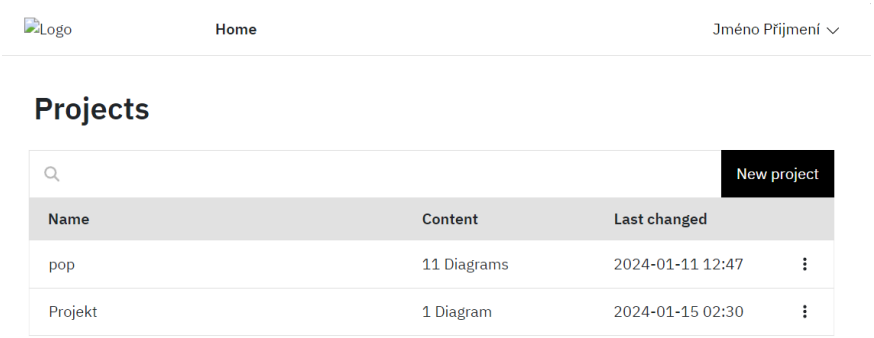
- Projekt - sekce pro správu projektů, kde mohou uživatelé vytvářet nové projekty, přejmenovávat je, mazat a spravovat členy projektu, v navigačním baru se vždy zobrazí konkrétní název projektu.
- Diagram - sekce pro správu BPMN diagramů, kde mohou uživatelé vytvářet, upravovat, mazat a verzovat diagramy, v navigačním baru se vždy zobrazí konkrétní název diagramu.
- Revize - sekce pro správu revizí BPMN diagramů, kde mohou uživatelé zobrazit historii revizí, porovnávat různé verze a obnovovat diagramy, v navigačním baru reprezentována jako "Revisions".
- Profil - sekce pro správu uživatelského profilu, kde mohou uživatelé aktualizovat své osobní informace a měnit heslo, v navigačním baru se vždy zobrazí konkrétní jméno a příjmení přihlášeného uživatele.



Obrázek 5.7: Návrh header komponenty Navigace

5.4.2 Dashboard

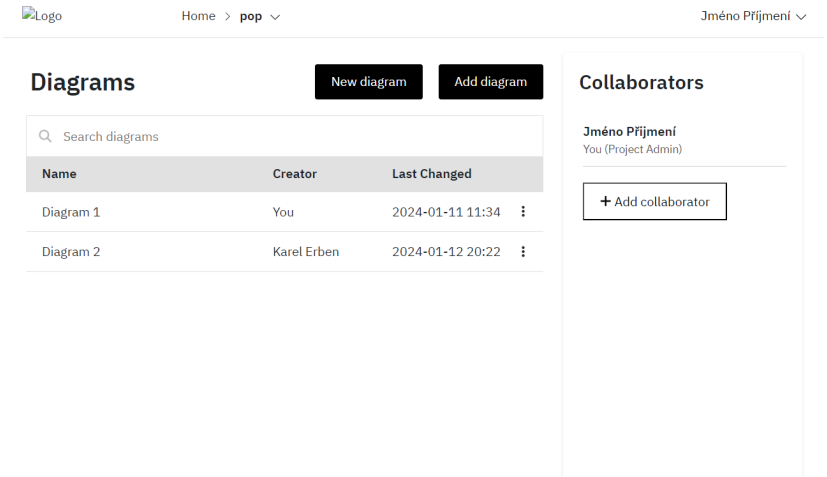
Dashboard poskytuje uživatelům přehled o aktuálních projektech uživatele. Zobrazuje seznam projektů včetně jejich ovládacích panelů k manipulaci daných vlastností. Stránka bude umožňovat uživatelům vytvářet nové projekty, přejmenovávat existující projekty nebo mazat/opouštět projekty (dle rolí uživatelů).



Obrázek 5.8: Návrh stránky Dashboard

5.4.3 Projekt

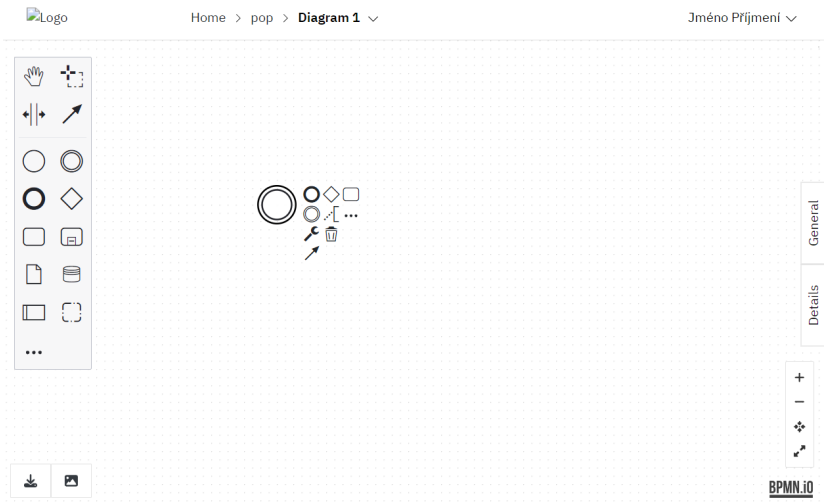
Projekt poskytuje uživatelům přehled o aktuálních diagramech v projektu. Zobrazuje seznam projektů včetně jejich ovládacích panelů. Stránka bude umožňovat uživatelům nahrávat nebo vytvářet nové diagramy, přejmenovávat existující diagramy nebo je mazat.



Obrázek 5.9: Návrh stránky Projekt

5.4.4 Modeler

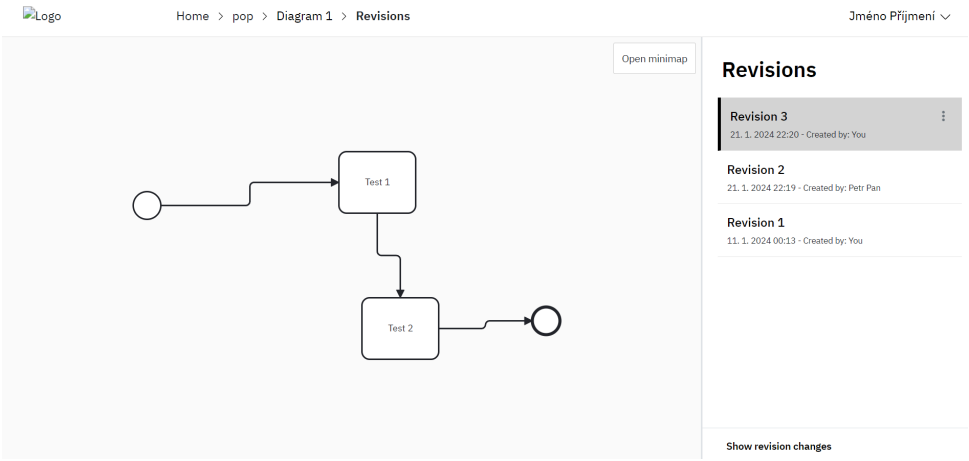
Modeler se otevře pokaždé, když budou chtít uživatelé zobrazit nebo editovat diagramy. Zde se bude odehrávat nejvíce uživatelské práce a jedná se o středobod aplikace. Uživatelé kromě možnosti chatování, specifikace popisu diagramu nebo kolaborativní práce na diagramu, mohou také vytvářet revize a přecházet na jejich historii.



Obrázek 5.10: Návrh stránky Modeler

5.4.5 Revize

Stránka s revizemi BPMN diagramů je klíčovým prvkem pro správu a sledování změn v diagramech. Stránka zahrnuje zobrazování historie revizí přehlednou formou, porovnání jednotlivých revizí mezi sebou grafickým způsobem a možnost obnovovat, přejmenovávat nebo mazat revize.



Obrázek 5.11: Návrh stránky Revize

Kapitola 6

Implementace

Tato čast nyní implementuje návrh z předchozí kapitoly do reálných kulis. Popíšeme si vývojové nástroje, který byly využity k implementaci, strukturu frontendu a backendu aplikace, a jednotlivé implementační řešení relevantní položek.

6.1 Vývojové prostředí

6.1.1 Nastavení vývojového prostředí

Pro vývoj frontendové části aplikace jsme využili knihovnu React. Vývojové prostředí bylo nastaveno pomocí Node.js a npm, což jsou standardní nástroje pro správu balíčků a běhové prostředí pro JavaScript.

Backendová část byla vyvíjena pomocí Spring Boot. Pro správu závislostí a sestavení projektu jsme použili Maven.

6.1.2 Vývojové nástroje

Pro vývoj jsme využili tato integrovaná vývojová prostředí (IDE):

- Webstorm - IDE od společnosti JetBrains, specializované na vývoj v JavaScriptu, které bylo využito pro vývoj frontendové části v Reactu. [60]
- IntelliJ IDEA - IDE pro vývoj v Javě, také od společnosti JetBrains. Z tohoto prostředí byla spravována také databáze. [61]

6.2 Frontend

6.2.1 Struktura projektu

Frontendová část aplikace je strukturována modulárně. Hlavní složkou je `src`, která obsahuje implementaci.

- **src/api:** Obsahuje služby pro komunikaci s backendem, např. pomocí knihovny Axios.

- `axiosConfig.js` - konfigurace Axios instance.
- `config.js` - základní konfigurace URL pro Axios a WebSocket.

- **src/assets:** Obsahuje statické soubory jako obrázky.
  - `diagramImage.png` - obrázek diagramu.
  - `procesqaLogo.png` - logo aplikace.
- **src/auth:** Obsahuje autentizační a autorizační logiku.
  - `AuthContext.js` - kontext pro správu autentizace uživatelů.
  - `PrivateRoute.js` - komponenta pro řízení přístupu k privátním stránkám.
  - `PublicRoute.js` - komponenta pro řízení přístupu k veřejným stránkám.
- **src/components:** Obsahuje všechny React komponenty používané v aplikaci.
  - `Header.js` - komponenta pro zobrazení jednotného záhlaví a navigace na všech stránkách.
  - `ErrorPopup.js` - komponenta pro zobrazení chybových hlášení.
- **src/pages:** Obsahuje jednotlivé stránky aplikace.
  - `BpmnModelerPage.js` - stránka pro modelování BPMN diagramů.
  - `DashboardPage.js` - přehledová stránka s projekty uživatele.
  - `ForgotPasswordPage.js` - stránka pro obnovení hesla.
  - `HomePage.js` - hlavní úvodní stránka aplikace.
  - `LoginPage.js` - stránka pro přihlášení uživatelů.
  - `NotFoundPage.js` - stránka zobrazená při nenalezení stránky.
  - `ProjectPage.js` - stránka pro správu konkrétního projektu.
  - `RevisionPage.js` - stránka pro správu revizí BPMN diagramů.
  - `SignupPage.js` - stránka pro registraci nových uživatelů.
  - `TokenSimulatorPage.js` - stránka pro simulaci tokenů v BPMN diagramech.
  - `UserDetailsPage.js` - stránka pro správu uživatelských údajů.
- **src/styles:** Obsahuje globální styly a styly pro jednotlivé komponenty.
  - `BpmnModelerPage.css` - styly specifické pro stránku s BPMN modelářem.
  - `DashboardPage.css` - styly pro přehledovou stránku.
  - `ForgotPasswordPage.css` - styly pro stránku s obnovou hesla.
  - `Header.css` - styly pro záhlaví.

- `HomePage.css` - styly pro úvodní stránku.
- `LoginPage.css` - styly pro přihlašovací stránku.
- `NotFoundPage.css` - styly pro stránku nenalezení.
- `ProjectPage.css` - styly pro stránku projektu.
- `RevisionPage.css` - styly pro stránku revizí.
- `SignupPage.css` - styly pro registrační stránku.
- `UserDetailsPage.css` - styly pro stránku uživatelských údajů.

- **src/utils:** Obsahuje užitečné utility funkce a helpery.
  - `bpmnDiffUtil.js` - nástroje pro porovnání BPMN diagramů.
  - `change-handler.js` - handler změn v diagramech.
  - `differ.js` - porovnávací nástroj.
- `App.js` - hlavní soubor aplikace, který definuje routování pomocí `react-router-dom`.
- `index.js` - vstupní bod aplikace, který renderuje hlavní komponentu `App`.

6.2.2 Interakce s backendem

Frontend komunikuje s backendem pomocí Axiosu, který je nakonfigurován v souboru `axiosConfig.js`. Všechny API požadavky jsou směrovány na `/api` endpoint.

- `axiosConfig.js` - obsahuje základní konfiguraci Axios instance a interceptory pro přidání JWT tokenu do každého požadavku. Tato konfigurace zahrnuje nastavení základních URL, timeoutů, a hlaviček pro každý požadavek.
- `config.js` - obsahuje základní konfiguraci URL pro Axios a WebSocket, což umožňuje snadnou změnu konfiguračních parametrů bez nutnosti zásahu do ostatních částí kódu. Tento soubor definuje klíčové URL pro komunikaci s backendem a WebSocket serverem.

Každý API požadavek je vybaven autentizačním tokenem, který je automaticky přidán do hlavičky požadavku pomocí interceptoru v Axiosu. Toto zabezpečení zajišťuje, že všechny interakce mezi frontendem a backendem jsou chráněny a ověřeny.

```
1 import axios from 'axios';
2
3 const axiosInstance = axios.create({
4 baseURL: '/api',
5 timeout: 10000,
6 headers: { 'Content-Type': 'application/json' }
7 });
8
9 axiosInstance.interceptors.request.use(config => {
```

```
const token = localStorage.getItem('token');
if (token) {
config.headers['Authorization'] = Bearer ${token};
}
return config;
}, error => {
return Promise.reject(error);
});

export default axiosInstance;
```

Listing 6.1: Inicializace a konfigurace Axiosu

Interakce s backendem je také realizována pomocí WebSocketu, což umožňuje obousměrnou komunikaci v reálném čase. Data jsou při odesílání komprimována a v případě potřeby odesílána v chuncích (po částech). Níže je ukázka kódu pro inicializaci WebSocket spojení a odesílání dat v chuncích:

```
const socket = new SockJS(config.websocketUrl);

socket.onopen = () => {
console.log('WebSocket connection established');
};

socket.onmessage = (event) => {
const message = JSON.parse(event.data);
console.log('Message from server:', message);
};

socket.onclose = () => {
console.log('WebSocket connection closed');
};

socket.onerror = (error) => {
console.error('WebSocket error:', error);
};
```

Listing 6.2: Inicializace WebSocket spojení a obsluha událostí

```
const autosaveDiagram = async (xml) => {
 if (stompClient && stompClient.connected) {
 const compressedData = compressData(xml);
 const base64Data = btoa(String.fromCharCode.apply(null, new
     Uint8Array(compressedData)));
  const totalChunks = Math.ceil(base64Data.length / CHUNK_SIZE);
    for (let i = 0; i < totalChunks; i++) {
        const chunk = base64Data.slice(i * CHUNK_SIZE, (i + 1) *
            CHUNK_SIZE);
        stompClient.publish({
            destination: `/app/updateDiagram/${diagramId}`,
            body: JSON.stringify({ diagramId, chunk, totalChunks
                , index: i })
            });
        }
    }
};
```

Listing 6.3: Odesílání dat pomocí WebSocket komprimované po částech

Tento kód inicializuje WebSocket spojení, které umožňuje aplikaci přijímat zprávy od serveru v reálném čase a reagovat na ně okamžitě. Odesílání dat v chuncích zajišťuje, že i velké BPMN modely mohou být efektivně přeneseny přes WebSocket.

6.2.3 BPMN modelování

BPMN modelování je implementováno pomocí knihovny `bpmn-js`. Tato knihovna poskytuje funkcionalitu pro vytváření a úpravy BPMN diagramů a je rozšířena o několik modulů.

- `BpmnModelerPage.js` - hlavní komponenta pro modelování BPMN diagramů, která inicializuje a konfiguruje modeler. Tato komponenta je zodpovědná za vykreslení BPMN modeleru na stránce a integraci dalších modulů.
- `BpmnModelerPage.css` - obsahuje styly specifické pro stránku s BPMN modelerem.
- `diagram-js-minimap` - modul pro zobrazení miniatury diagramu, který umožňuje uživatelům rychle navigovat velkými diagramy pomocí malé náhledové mapy.
- `gridModule` - modul pro zobrazení mřížky, která pomáhá uživatelům přesně umístit a zarovnat prvky v diagramu.
- `CreateAppendAnythingModule` - modul pro rozšířené možnosti přidávání prvků do diagramu, umožňující větší flexibilitu při tvorbě BPMN diagramů.
- `BpmnPropertiesPanelModule` - modul pro zobrazení a úpravu vlastností jednotlivých prvků BPMN diagramu. Tento modul poskytuje uživatelům možnost měnit detaily každého prvku, jako jsou ID, název, dokumentace a další specifické vlastnosti.

6.2.4 Inicializace BPMN modeleru

Inicializace BPMN modeleru je klíčovým krokem při vytváření BPMN diagramů. Níže je ukázka kódu pro inicializaci modeleru s několika moduly:

```
1  const newModeler = new Modeler({
2      container: viewmodelRef.current,
3      keyboard: { bindTo: document },
4      propertiesPanel: {
5          parent: '#properties-panel',
6      },
7      additionalModules: [
8          BpmnPropertiesPanelModule,
9          BpmnPropertiesProviderModule,
10         minimapModule,
11         gridModule,
```

```
        CreateAppendAnythingModule
    ]
});
```

Listing 6.4: Inicializace instance Modeler v Reactu

6.2.5 Revize BPMN diagramů

Revize BPMN diagramů jsou podstatnou součástí systému, která umožňuje sledování změn a správu různých verzí diagramů. Níže je popsán přístup k implementaci revizí:

Ukládání revizí

Nová revize je vytvořena po kliknutí na možnost "Vytvořit novou revizi". Po vytvoření dojde k jejímu odeslání přes axiosInstanci pomocí http požadavku POST. První revize se vytvoří automaticky při vytvoření diagramu. Níže je ukázka kódu vytvoření nové revize.

```
const saveRevision = async (diagramId, xml) => {
try {
await axiosInstance.post(/diagrams/${diagramId}/revisions, { xml
    });
console.log('Revision saved successfully');
} catch (error) {
console.error('Error saving revision:', error);
}
};
```

Listing 6.5: Vytvoření nové revize BPMN diagramu

Načítání a porovnávání revizí

Uživatelé mohou načítat starší revize BPMN diagramů a porovnávat je navzájem. Pro porovnávání revizí používáme vlastní implementaci diff nástrojů, které najdou rozdíly v diagramech a následně tyto rozdíly barevně zvýrazníme. Tento proces využívá několik utilit, které pomáhají identifikovat rozdíly mezi verzemi diagramů.

```
import { diff } from '../utils/bpmnDiffUtil';

const applyDiffing = async (selectedIndex = null) => {
    const handler = await BpmnDiffUtil.applyDiff(viewer, nextXML
        , currentXML );

    const elementRegistry = viewer.get('elementRegistry');
    const modeling = viewer.get('modeling');
    const canvas = viewer.get('canvas');

    modeling.setColor(elementRegistry.getAll(), null);

 const applyChanges = (changes, colorConfig) => {
```

```
    Object.keys(changes).forEach(id => {
        const element = elementRegistry.get(id);
        if (element) {
            modeling.setColor([element], colorConfig);
            console.log(`Color applied for element ID: ${id}`);
        } else {
            console.warn(`Element with ID ${id} not found`);
        }
    });
};

applyChanges(handler._removed, { stroke: '#bb163f'});
applyChanges(handler._layoutChanged, { stroke: '#2c97fd'});
applyChanges(handler._added, { stroke: '#69d31f'});
applyChanges(handler._changed, { stroke: '#ff9b00'});

}
```

Listing 6.6: Porovnávání revizí a zvýraznění změn

Obnova revizí

Systém umožňuje uživatelům obnovit diagram na jakoukoliv předchozí verzi. Obnovení je realizováno prostřednictvím API, které načte vybranou revizi a nastaví ji jako aktuální verzi diagramu. Tato funkcionalita zajišťuje, že uživatelé mohou kdykoliv vrátit změny a pracovat s dřívějšími verzemi svých diagramů.

6.2.6 Zabezpečení

Zabezpečení na frontendu je klíčové pro ochranu citlivých dat a zajištění bezpečné komunikace mezi klientem a serverem. Implementace zabezpečení zahrnuje správu autentizačních tokenů, ochranu citlivých údajů a zajištění, že všechny požadavky jsou autorizovány.

Autentizační tokeny jsou ukládány v lokálním úložišti (localStorage) a automaticky přidávány do hlaviček všech odchozích HTTP požadavků pomocí interceptoru v Axiosu. Tím je zajištěno, že pouze autorizovaní uživatelé mají přístup k chráněným zdrojům.

Pro zvýšení bezpečnosti je uživatel automaticky odhlášen po třech hodinách neaktivity. Toto opatření zajišťuje, že opuštěné relace nebudou zneužity třetími stranami.

Níže je ukázka kódu pro správu autentizačních tokenů a ochranu citlivých dat:

```
import axiosInstance from './axiosConfig';

const handleLogin = async (credentials) => {
    try {
        const response = await axios.post('/api/login',
            credentials);
        const { token } = response.data;
        localStorage.setItem('token', token);
```

```
8
9           axiosInstance.defaults.headers['Authorization'] = `
                Bearer ${token}`;
10
11          setTimeout(() => {
12              localStorage.removeItem('token');
13              delete axiosInstance.defaults.headers['Authorization
                    '];
14              console.log('Session expired, user logged out
                    automatically');
15          }, 3 * 60 * 60 * 1000);
16
17          console.log('Login successful, token stored');
18      } catch (error) {
19          console.error('Login error:', error);
20      }
21  };
```

Listing 6.7: Správa autentizačních tokenů a ochrana citlivých dat

Tento kód zajišťuje, že autentizační token je správně uložen a používán pro autorizaci všech budoucích požadavků na server, čímž se chrání citlivá data a zajišťuje bezpečná komunikace.

6.3 Backend

6.3.1 Struktura projektu

Backendová část je strukturována do vrstevnaté architektury (rozložení do několika vrstev dle funkce a odpovědnosti).

- **com.procesqa.backendprocesqa:** Obsahuje hlavní třídu aplikace.
  - `BackendProcesqaApplication.java` - hlavní třída aplikace, která spouští Spring Boot aplikaci.
- **com.procesqa.backendprocesqa.config:** Obsahuje konfigurační soubory.
  - `SecurityConfig.java` - konfigurace zabezpečení aplikace pomocí Spring Security.
  - `WebSocketConfig.java` - konfigurace WebSocket pro real-time komunikaci.
- **com.procesqa.backendprocesqa.controller:** Obsahuje kontroléry pro zpracování HTTP požadavků.
  - `DiagramController.java` - kontrolér pro správu BPMN diagramů.
  - `ProjectController.java` - kontrolér pro správu projektů.
  - `RevisionController.java` - kontrolér pro správu revizí diagramů.
  - `SpecificationController.java` - kontrolér pro správu specifikací.

- `UserController.java` - kontrolér pro správu uživatelů.
- `WebSocketController.java` - kontrolér pro správu WebSocket komunikace.
- `GlobalExceptionHandler.java` - kontrolér pro globální zpracování výjimek.

- **com.procesqa.backendprocesqa.entity:** Obsahuje entity, které reprezentují tabulky v databázi.
  - `User.java` - entita uživatele.
  - `Project.java` - entita projektu.
  - `Diagram.java` - entita diagramu.
  - `Revision.java` - entita revize.
  - `Specification.java` - entita specifikace.
  - `ChatMessage.java` - entita zprávy v chatu.
  - `UserProject.java` - entita spojující uživatele a projekt.
- **com.procesqa.backendprocesqa.repository:** Obsahuje repozitáře pro přístup k databázi.
  - `UserRepository.java` - repozitář pro uživatele.
  - `ProjectRepository.java` - repozitář pro projekty.
  - `DiagramRepository.java` - repozitář pro diagramy.
  - `RevisionRepository.java` - repozitář pro revize.
  - `SpecificationRepository.java` - repozitář pro specifikace.
  - `ChatMessageRepository.java` - repozitář pro zprávy v chatu.
  - `UserProjectRepository.java` - repozitář pro spojení uživatele a projektu.
- **com.procesqa.backendprocesqa.service:** Obsahuje služby pro logiku aplikace.
  - `UserService.java` - služba pro správu uživatelů.
  - `ProjectService.java` - služba pro správu projektů.
  - `DiagramService.java` - služba pro správu diagramů.
  - `RevisionService.java` - služba pro správu revizí.
  - `SpecificationService.java` - služba pro správu specifikací.
  - `ChatService.java` - služba pro správu zpráv v chatu.
- **com.procesqa.backendprocesqa.security:** Obsahuje bezpečnostní konfiguraci a autentizační filtry.
  - `JwtAuthenticationFilter.java` - filtr pro JWT autentizaci.

- `CustomUserDetails.java` - custom implementace `UserDetails` pro autentizaci uživatelů.
- `JwtTokenProvider.java` - poskytovatel JWT tokenů.
- `InvalidTokenException.java` - výjimka pro neplatné tokeny.

- **com.procesqa.backendprocesqa.dto:** Obsahuje DTO (Data Transfer Object) třídy.
  - `DiagramUpdateRequest.java` - DTO pro aktualizaci diagramu.
  - `JwtAuthenticationResponse.java` - DTO pro JWT autentizační odpověď.
  - `UserLoginRequest.java` - DTO pro žádost o přihlášení uživatele.
  - `UserRegistrationRequest.java` - DTO pro žádost o registraci uživatele.
- **com.procesqa.backendprocesqa.util:** Obsahuje utility třídy.
  - `CompressionUtil.java` - utility třída pro kompresi a dekompresi dat.

6.3.2 Inicializace

com.procesqa.backendprocesqa

- `BackendProcesqaApplication.java` - hlavní třída aplikace, která spouští Spring Boot aplikaci.

```
@SpringBootApplication
public class BackendProcesqaApplication {
    public static void main(String[] args) {
        SpringApplication.run(BackendProcesqaApplication.class,
            args);
    }
}
```

Listing 6.8: BackendProcesqaApplication.java

6.3.3 Konfigurace

com.procesqa.backendprocesqa.config

Nezbytné součásti pro nastavení zabezpečení a websocketů.

- `SecurityConfig.java` - konfigurace zabezpečení aplikace pomocí Spring Security.
- `WebSocketConfig.java` - konfigurace WebSocket pro real-time komunikaci.

```java
@Configuration
@EnableWebSecurity
public class SecurityConfig {
    @Autowired
    private CustomUserDetailsService customUserDetailsService;

    @Autowired
    private BCryptPasswordEncoder bCryptPasswordEncoder;

    @Autowired
    private JwtAuthenticationFilter jwtAuthenticationFilter;

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http)
        throws Exception {
        http
            .cors(cors -> cors.configurationSource(
                corsConfigurationSource()))
            .csrf(AbstractHttpConfigurer::disable)
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/api/users/register", "/api/
                    users/login").permitAll()
                .anyRequest().authenticated())
            .addFilterBefore(jwtAuthenticationFilter,
                UsernamePasswordAuthenticationFilter.class);

        return http.build();
    }

    @Bean
    public AuthenticationManager authenticationManagerBean(
        HttpSecurity http) throws Exception {
        AuthenticationManagerBuilder
            authenticationManagerBuilder = http.getSharedObject(
            AuthenticationManagerBuilder.class);
        authenticationManagerBuilder
            .userDetailsService(customUserDetailsService)
            .passwordEncoder(bCryptPasswordEncoder);
        return authenticationManagerBuilder.build();
    }

}
```

Listing 6.9: SecurityConfig.java

```java
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements
    WebSocketMessageBrokerConfigurer {
    @Override
    public void registerStompEndpoints(StompEndpointRegistry
        registry) {
        registry.addEndpoint("/ws").setAllowedOrigins("http://
            localhost:3000").withSockJS();
    }

    @Override
```

```
10      public void configureMessageBroker(MessageBrokerRegistry
            registry) {
11          registry.enableSimpleBroker("/topic");
12          registry.setApplicationDestinationPrefixes("/app");
13      }
14  }
```

Listing 6.10: WebSocketConfig.java

6.3.4 Kontrolery

com.procesqa.backendprocesqa.controller

Kontrolery poskytují rozhraní pro správu různých entit aplikace. Jsou to třídy označené anotací `@RestController`, které zpracovávají HTTP požadavky a vracejí HTTP odpovědi. Každý kontrolér mapuje specifické požadavky na konkrétní metody pomocí anotací jako `@GetMapping`, `@PostMapping`, `@PutMapping` a `@DeleteMapping`.

- `DiagramController.java` - kontrolér pro správu BPMN diagramů.
- `ProjectController.java` - kontrolér pro správu projektů.
- `RevisionController.java` - kontrolér pro správu revizí diagramů.
- `SpecificationController.java` - kontrolér pro správu specifikací.
- `UserController.java` - kontrolér pro správu uživatelů.
- `WebSocketController.java` - kontrolér pro správu WebSocket komunikace.
- `GlobalExceptionHandler.java` - kontrolér pro globální zpracování výjimek.

`DiagramController` například spravuje BPMN diagramy, umožňuje jejich vytváření, načítání a aktualizaci. `createDiagram` metoda přijímá nový diagram a ID projektu jako parametry, vytvoří diagram a vrátí odpověď s HTTP statusem 201 (Created).

```
1   @RestController
2   @RequestMapping("/api/diagrams")
3   public class DiagramController {
4
5       @PostMapping
6       public ResponseEntity<?> createDiagram(@RequestBody Diagram
            diagram, @RequestParam UUID projectId) {
7           Authentication authentication = SecurityContextHolder.
                getContext().getAuthentication();
8           String email = authentication.getName();
9           User user = userService.findByEmail(email);
10          if (user == null) {
```

```
11              log.warn("No authenticated user found with email:
                    {}", email);
12              return ResponseEntity.status(HttpStatus.UNAUTHORIZED
                    ).body("User not authenticated");
13          }
14          if (!projectService.isUserAllowedOnProject(user.
                getUserId(), projectId)) {
15              log.warn("User {} does not have access to project
                    {}", email, projectId);
16              return ResponseEntity.status(HttpStatus.FORBIDDEN).
                    body("User does not have access to this project
                    ");
17          }
18          diagram.setProject(projectService.findProjectById(
                projectId));
19          diagram.setCreatedBy(user);
20          diagram.setCreatedAt(LocalDateTime.now());
21          diagram.setUpdatedAt(LocalDateTime.now());
22          diagram.setData(BPMNdiagramXML);
23
24          Diagram createdDiagram = diagramService.createDiagram(
                diagram, projectId);
25
26          return ResponseEntity.status(HttpStatus.CREATED).body(
                createdDiagram);
27  }
```

Listing 6.11: DiagramController.java

`WebSocketController` zajišťuje správu WebSocket komunikace pro aktualizaci diagramů v reálném čase. Implementace provozu websocketu s kompresí a po chuncích (částech), když je to nutné z důvodu velikosti modelů, zahrnuje následující kroky:

- Nastavení velikosti chunků na 16 300 bajtů. Tato hodnota byla zvolena tak, aby byla dostatečně velká pro efektivní přenos dat, ale zároveň se vlezla do maximálního omezení velikosti zprávy ve websocketu.
- Chunky jsou ukládány do dočasné struktury chunkStorage, která je implementována jako `ConcurrentHashMap`. Klíčem je `UUID` diagramu a hodnotou je mapa indexů chunků a jejich obsahu.
- Po obdržení všech chunků je obsah rekonstruován a dekomprimován pomocí třídy `CompressionUtil`.
- Po dekompresi je aktualizován obsah diagramu v databázi a nastavena aktuální časová značka.
- Aktualizovaný obsah diagramu je znovu komprimován a odeslán zpět klientům v chuncích.

```
@Controller
public class WebSocketController {
    @MessageMapping("/updateDiagram/{diagramId}")
    public void updateDiagram(@DestinationVariable UUID
        diagramId, @Payload Map<String, Object> updateRequest) {
        UUID id = UUID.fromString(updateRequest.get("diagramId")
            .toString());
        int totalChunks = Integer.parseInt(updateRequest.get("
            totalChunks").toString());
        int index = Integer.parseInt(updateRequest.get("index").
            toString());
        String chunk = updateRequest.get("chunk").toString();

        chunkStorage.computeIfAbsent(id, k -> new HashMap<>()).
            put(index, chunk);

        if (chunkStorage.get(id).size() == totalChunks) {
            StringBuilder completeData = new StringBuilder();
            for (int i = 0; i < totalChunks; i++) {
                completeData.append(chunkStorage.get(id).get(i))
                    ;
            }
            chunkStorage.remove(id);

            String decompressedData;
            decompressedData = CompressionUtil.decompress(
                completeData.toString());

            Diagram existingDiagram = diagramService.
                findDiagramById(diagramId);
            if (existingDiagram == null) {
                throw new ResponseStatusException(HttpStatus.
                    NOT_FOUND, "Diagram not found");
            }
            existingDiagram.setData(decompressedData);
            existingDiagram.setUpdatedAt(LocalDateTime.now());
            diagramService.updateDiagram(existingDiagram);
            String compressedResponseData;

            compressedResponseData = CompressionUtil.compress(
                existingDiagram.getData());

            sendChunksBackToSubscribers(diagramId,
                compressedResponseData);
        }
    }
    private void sendChunksBackToSubscribers(UUID diagramId,
        String compressedData) {
        int totalChunks = (int) Math.ceil((double)
            compressedData.length() / CHUNK_SIZE);
        for (int i = 0; i < totalChunks; i++) {
            String chunk = compressedData.substring(i *
                CHUNK_SIZE, Math.min((i + 1) * CHUNK_SIZE,
                compressedData.length()));
            Map<String, Object> message = new HashMap<>();
            message.put("diagramId", diagramId.toString());
```

```
42              ...
43              messagingTemplate.convertAndSend("/topic/
                    diagramUpdate/" + diagramId, message);
44          }
45      }
46  }
```

Listing 6.12: WebSocketController.java

`CompressionUtil` je utility třída, která poskytuje metody pro kompresi a dekompresi dat pomocí algoritmu GZIP. Metoda `compress` přijímá textový řetězec, komprimuje ho a vrací Base64 kódovaný řetězec. Metoda `decompress` přijímá Base64 kódovaný komprimovaný řetězec, dekomprimuje ho a vrací původní textový řetězec.

6.3.5 Služby

com.procesqa.backendprocesqa.service

Služby implementují logiku aplikace a zajišťují komunikaci s databází přes repozitáře. Jsou označeny anotací `@Service` a obsahují metody pro práci s daty.

- `UserService.java` - služba pro správu uživatelů.
- `ProjectService.java` - služba pro správu projektů.
- `DiagramService.java` - služba pro správu diagramů.
- `RevisionService.java` - služba pro správu revizí.
- `SpecificationService.java` - služba pro správu specifikací.
- `ChatService.java` - služba pro správu zpráv v chatu.

```
1   @Service
2   public class DiagramService {
3       @Autowired
4       private DiagramRepository diagramRepository;
5
6       public List<Diagram> findAllDiagrams() {
7           return diagramRepository.findAll();
8       }
9
10      public Diagram createDiagram(Diagram diagram, UUID projectId
            ) {
11          diagram.setProject(new Project(projectId));
12          return diagramRepository.save(diagram);
13      }
14  }
```

Listing 6.13: DiagramService.java

`DiagramService` například poskytuje metody pro vyhledávání a vytváření diagramů. Metoda `createDiagram` nastaví projekt pro nový diagram a uloží jej do databáze.

6.3.6 DTO

com.procesqa.backendprocesqa.dto

DTO (Data Transfer Objects) třídy slouží k přenosu dat mezi frontendem a backendem. Zajišťují, že pouze relevantní data jsou přenášena mezi klientem a serverem.

- `DiagramUpdateRequest.java` - DTO pro aktualizaci diagramu.
- `JwtAuthenticationResponse.java` - DTO pro JWT autentizační odpověď.
- `UserLoginRequest.java` - DTO pro žádost o přihlášení uživatele.
- `UserRegistrationRequest.java` - DTO pro žádost o registraci uživatele.

```
1 @Data
2 public class UserRegistrationRequest {
3     private String email;
4     private String firstName;
5     private String lastName;
6     private String password;
7 }
```

Listing 6.14: UserRegistrationRequest.java

`UserRegistrationRequest` například obsahuje údaje potřebné pro registraci nového uživatele.

6.3.7 Entity

com.procesqa.backendprocesqa.entity

Entity reprezentují tabulky v databázi a používají se k mapování dat pomocí JPA. Každá entita je anotována jako `@Entity` a obsahuje atributy, které se mapují na sloupce v databázi.

- `User.java` - entita uživatele.
- `Project.java` - entita projektu.
- `Diagram.java` - entita diagramu.
- `Revision.java` - entita revize.
- `Specification.java` - entita specifikace.
- `ChatMessage.java` - entita zprávy v chatu.
- `UserProject.java` - entita spojující uživatele a projekt.

```java
@Entity
@Data
@Table(name = "users")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private UUID userId;

    private String firstName;
    private String lastName;
    private String email;
    private String passwordHash;
    private Boolean isActive;
    private Role role;

    public enum Role {
        USER, ADMIN
    }
}
```

Listing 6.15: User.java

`User` například reprezentuje uživatele aplikace. Obsahuje atributy jako `userId`, `firstName`, `lastName`, `email` a `passwordHash`.

6.3.8 Repozitáře

com.procesqa.backendprocesqa.repository

Repozitáře poskytují rozhraní pro přístup k databázi a podporují CRUD operace (Create, Read, Update, Delete). Jsou označeny anotací `@Repository` a rozšiřují rozhraní `JpaRepository`.

- `UserRepository.java` - repozitář pro uživatele.
- `ProjectRepository.java` - repozitář pro projekty.
- `DiagramRepository.java` - repozitář pro diagramy.
- `RevisionRepository.java` - repozitář pro revize.
- `SpecificationRepository.java` - repozitář pro specifikace.
- `ChatMessageRepository.java` - repozitář pro zprávy v chatu.
- `UserProjectRepository.java` - repozitář pro spojení uživatele a projektu.

6.3.9 Security

com.procesqa.backendprocesqa.security

Bezpečnostní konfigurace a autentizační filtry jsou klíčové pro zabezpečení aplikace. Používáme JWT (JSON Web Token) pro autentizaci a autorizaci

uživatelů.

- `JwtAuthenticationFilter.java` - filtr pro JWT autentizaci.
- `CustomUserDetails.java` - Custom implementace `UserDetails` pro autentizaci uživatelů.
- `JwtTokenProvider.java` - poskytovatel JWT tokenů.
- `InvalidTokenException.java` - výjimka pro neplatné tokeny.

```
1  public class JwtAuthenticationFilter extends
       OncePerRequestFilter {
2      @Autowired
3      private JwtTokenProvider jwtTokenProvider;
4
5      @Autowired
6      private CustomUserDetailsService customUserDetailsService;
7
8      @Override
9      protected void doFilterInternal(HttpServletRequest request,
           HttpServletResponse response, FilterChain filterChain)
10             throws ServletException, IOException {
11         String token = jwtTokenProvider.resolveToken(request);
12         if (token != null && jwtTokenProvider.validateToken(
               token)) {
13             Authentication auth = jwtTokenProvider.
                   getAuthentication(token);
14             SecurityContextHolder.getContext().setAuthentication
                   (auth);
15         }
16         filterChain.doFilter(request, response);
17     }
18 }
```

Listing 6.16: JwtAuthenticationFilter.java

`JwtAuthenticationFilter` zajišťuje, že každý příchozí HTTP požadavek obsahuje platný JWT token, který je následně validován a použit pro autentizaci uživatele.

`JwtTokenProvider` obsahuje metody pro vytvoření, validaci a extrakci informací z JWT tokenu.

6.3.10 Util

com.procesqa.backendprocesqa.util

Utility třídy poskytují pomocné funkce a metody, které jsou použity napříč aplikací. Zde máme pouze utilitu ke kompresi a dekompresi dat

- `CompressionUtil.java` - utility třída pro kompresi a dekompresi dat.

Kapitola 7

Testování

Testování je klíčovou součástí softwarového vývoje, zajišťující, že systém funguje podle očekávání a splňuje všechny požadavky. Neprve si zadefinujeme testy, které budeme využívat:

- Jednotkové testy - testování jednotlivých částí implementované logiky kódu pomocí knihovny JUnit přímo ve Spring Bootu. [29]
- Manuální testy - testování manuálně bez automatizovaných nástrojů dle předpřipravených scénářů, vhodné pro uživatelské rozhraní, kde je třeba posoudit vzhled a chování aplikace z pohledu uživatele. [30]
- Uživatelské testy - zahrnují skutečné uživatele, kteří pracují s aplikací a poskytují zpětnou vazbu na její použitelnost, funkčnost a výkon. Tento typ testování ověřuje, že aplikace je uživatelsky přívětivá a splňuje očekávání koncových uživatelů. [31]

Poté si navrhneme testovací strategii:

1. Nadefinujeme si testovací scénáře.
2. Vytvoříme testovací data.
3. Automatizujeme jednotkové testy.
4. Provedeme manuální testy.
5. Provedeme uživatelské testování.
6. Vyhodnotíme testy.

7.1 Manuální testy

Manuální testy hrají svoji roli hlavně při ověřování správné funkčnosti aplikace z pohledu uživatele. Tyto testy jsou prováděny podle předem připravených testovacích scénářů, které popisují jednotlivé kroky a očekávané výsledky.

Testovací scénáře jsou detailní popisy kroků, které tester musí provést, aby ověřil správné chování systému. Každý scénář zahrnuje:

- Název scénáře - stručný popis toho, co se bude testovat.
- Předpoklady - podmínky, které musí být splněny před začátkem testování.
- Kroky - detailní popis jednotlivých kroků, které tester provádí.
- Očekávaný výsledek - popis toho, co by se mělo stát po provedení každého kroku.

7.1.1 Testovací scénáře

V rámci manuálního testování jsme si nadefinovali základní testovací scénáře k pokrytí těch nejdůležitějších součástí aplikace, které nám zároveň v budoucnu budou sloužit jako regresní testy k otestování fukncionalit, které by mohli být v budoucím vývoji narušeny.

Registrace a přihlášení nového uživatele

Test ID: TC-1
Priorita: Vysoká
Předpoklady: Žádný existující účet s použitým e-mailem
Kroky:

1. Uživatel vyplní registrační formulář s požadovanými údaji.
2. Systém zkontroluje správnost vyplněných údajů.
3. Systém vytvoří nový uživatelský účet.
4. Systém odešle potvrzovací e-mail uživateli.
5. Uživatel potvrdí registraci kliknutím na odkaz v e-mailu.
6. Uživatel vyplní přihlašovací formulář (e-mail a heslo).
7. Systém ověří správnost přihlašovacích údajů.
8. Systém přihlásí uživatele do aplikace.
9. Systém zobrazí hlavní stránku aplikace.

Očekávaný výsledek: Nový uživatel úspěšně zaregistrován a přihlášen do aplikace.

Vytvoření, přejmenování a smazání projektu

Test ID: TC-2
Priorita: Vysoká
Předpoklady: Registrovaný uživatel přihlášen do aplikace
Kroky:

1. Uživatel klikne na tlačítko pro vytvoření nového projektu.

2. Systém zobrazí formulář pro zadání názvu projektu.
3. Uživatel vyplní název projektu a potvrdí vytvoření.
4. Systém vytvoří nový projekt a zobrazí jej uživateli.
5. Uživatel vybere projekt z přehledu projektů a zvolí možnost přejmenovat projekt.
6. Systém zobrazí pole pro zadání nového názvu.
7. Uživatel zadá nový název projektu a potvrdí změnu.
8. Systém uloží nový název projektu a zobrazí jej uživateli.
9. Uživatel vybere projekt z přehledu projektů a zvolí možnost smazat projekt.
10. Systém zobrazí potvrzovací dialog s dotazem, zda uživatel opravdu chce smazat projekt.
11. Uživatel potvrdí smazání projektu.
12. Systém smaže projekt a informuje uživatele o úspěšném smazání.

Očekávaný výsledek: Projekt úspěšně vytvořen, přejmenován a smazán.

Správa BPMN diagramů v projektu

Test ID: TC-3
Priorita: Vysoká
Předpoklady: Registrovaný uživatel přihlášen do aplikace a má přístup k projektu
Kroky:

1. Uživatel klikne na tlačítko pro vytvoření nového BPMN diagramu.
2. Systém zobrazí formulář pro zadání názvu diagramu.
3. Uživatel vyplní název diagramu a potvrdí vytvoření.
4. Systém vytvoří nový diagram a zobrazí jej uživateli.
5. Uživatel vybere diagram z přehledu diagramů a zvolí možnost přejmenovat diagram.
6. Systém zobrazí pole pro zadání nového názvu.
7. Uživatel zadá nový název diagramu a potvrdí změnu.
8. Systém uloží nový název diagramu a zobrazí jej uživateli.
9. Uživatel vybere diagram z přehledu diagramů a zvolí možnost smazat diagram.

10. Systém zobrazí potvrzovací dialog s dotazem, zda uživatel opravdu chce smazat diagram.
11. Uživatel potvrdí smazání diagramu.
12. Systém smaže diagram a informuje uživatele o úspěšném smazání.

Očekávaný výsledek: BPMN diagram úspěšně vytvořen, přejmenován a smazán.

Revize a historie BPMN diagramů

Test ID: TC-4
Priorita: Střední
Předpoklady: Registrovaný uživatel přihlášen do aplikace a má přístup k projektu s BPMN diagramy
Kroky:

1. Uživatel má otevřený diagram, který chce přejmenovat.
2. Systém zobrazí aktuální název diagramu v horní části obrazovky s možností otevření menu diagramu.
3. Uživatel klikne na název diagramu, otevře menu diagramu a zvolí možnost vytvořit revizi.
4. Systém uloží aktuální stav diagramu jako novou revizi.
5. Systém zobrazí potvrzení o úspěšném vytvoření revize.
6. Uživatel má otevřený diagram, u kterého chce zobrazit historii.
7. Systém zobrazí aktuální název diagramu v horní části obrazovky s možností otevření menu diagramu.
8. Uživatel klikne na název diagramu, otevře menu diagramu a zvolí možnost zobrazit historii revizí.
9. Systém zobrazí stránku s historií revizí, kde zobrazí seznam všech revizí pro vybraný diagram.
10. Uživatel si může zobrazit detaily jednotlivých revizí.

Očekávaný výsledek: Revize diagramu úspěšně vytvořena a historie revizí správně zobrazena.

7.2 Jednotkové testy

Jednotkové testy jsou základní formou automatizovaného testování, které se zaměřuje na testování jednotlivých jednotek kódu izolovaně. V našem projektu jsme pro jednotkové testy použili knihovnu JUnit 5, která je de

facto standardem pro testování v Javě. [32] V kodů jsme tedy pokryli ty nejklíčovější třídy pomocí jednotkových testů k zajištění správné funkčnosti nyní i v budoucnu. Níže je ukázka testu k otestování Diagram služby přímo ve Spring Boot aplikaci.

```
@SpringBootTest
public class DiagramServiceTests {

    @Autowired
    private DiagramService diagramService;
    @Test
    void updateDiagram_ShouldReturnUpdatedDiagram() {
        UUID diagramId = UUID.randomUUID();
        Diagram diagram = new Diagram();
        diagram.setDiagramId(diagramId);
        diagram.setName("Old diagram");

        Diagram updatedDiagram = new Diagram();
        updatedDiagram.setDiagramId(diagramId);
        updatedDiagram.setName("New diagram");

        when(diagramRepository.save(any(Diagram.class))).
            thenReturn(updatedDiagram);

        Diagram result = diagramService.updateDiagram(
            updatedDiagram);

        assertNotNull(result);
        assertEquals("New diagram", result.getName());
        verify(diagramRepository, times(1)).save(any(Diagram.
            class));
    }

    @Test
    void deleteDiagram_ShouldInvokeDeleteById() {
        UUID diagramId = UUID.randomUUID();

        doNothing().when(diagramRepository).deleteById(diagramId
            );

        diagramService.deleteDiagram(diagramId);

        verify(diagramRepository, times(1)).deleteById(diagramId
            );
    }
}
```

Listing 7.1: Ukázka JUnit testu

7.3 Uživatelské testy

Uživatelské testy hrají podstatnou roli v ověření toho, že aplikace je uživatelsky přívětivá a splňuje očekávání koncových uživatelů. Takové testy je nutné provádět se skutečnými uživateli, kteří používají aplikaci a mohou poskytnout

zpětnou vazbu.

7.3.1 Metodika testování

Pro uživatelské testování se volí kombinace kvantitativních a kvalitativních metod. Kvantitativní data jsou získávána prostřednictvím vyplnění krátkého dotazníku, který sleduje interakci uživatelů s aplikací. Kvalitativní data se shromažďují prostřednictvím rozhovorů a pozorování s účastníky testování, kde uživatelé podrobně popisují své zkušenosti a identifikují specifické problémy k opravě.

7.3.2 Výběr respondentů

Subjekty pro uživatelské testování byly vybrány ze škály osob neznalých BPMN notace a procesního řízení a osob znalých této notace a procesního řízení. Tím byl zajištěn pohled z různých stran spektra.

7.3.3 Účastníci

Pro uživatelské testování bylo zvoleno pět účastníků dle předchozí specifikace výběru:

- Projektový manažer, 34 let, střední znalost BPMN notace, vysoká úroveň počítačové gramotnosti
- Chemická laborantka, 57 let, nulová znalost BPMN notace, nízká úroveň počítačové gramotnosti
- Studentka politologie, 24 let, nízká znalost BPMN notace, střední úroveň počítačové gramotnosti
- Procesní analytik, 26 let, vysoká znalost BPMN notace, vysoká úroveň počítačové gramotnosti
- Softwarový inženýr, 23 let, vysoká znalost BPMN notace, vysoká úroveň počítačové gramotnosti

7.3.4 Seznam identifikovaných problémů

V průběhu uživatelského testování byly identifikovány tyto problémy:

- Nefunkční stahování do SVG formátu.
- Zobrazování projektového menu v navigační liště z jiné strany, než by mělo být možné.
- Zobrazení informačního pop-up okna při vytvoření revize zabraňuje rychlému překliknutí na historii revizí.
- Problémy s kompatibilitou na mobilních zařízeních.

7.3.5 Výsledky uživatelského testování

Výsledky uživatelského testování ukázaly, že aplikace je celkově uživatelsky přívětivá pro všechny druhy uživatelů. Nebyl pro ně problém registrace, přihlášení ani zakládání projektů nebo diagramů. Objevilo se pouze pár problémů, které vznikly v průběhu vývoje dalších funkcionalit, a bylo tedy nutné původní funkcionality opravit. Co se týče mobilních zařízení, tak u nich není důvod cokoliv měnit, protože mobilní zařízení nejsou schopna modelovat diagram v modeleru, takže by to postrádalo smysl.

7.4 Výsledky testování

Výsledky testování aplikace ukázaly následující:

- Manuální testy - Všechny manuální testy proběhly úspěšně. Proces vytváření a správy revizí také proběhl bez chyb.
- Jednotkové testy - Jednotkové testy pokryly hlavní funkcionality aplikace. Testy úspěšně ověřily správnost operací jako je aktualizace a mazání diagramů. Knihovna JUnit 5 byla efektivně použita pro automatizaci těchto testů.
- Uživatelské testy - Uživatelské testy zahrnovaly pět účastníků s různou úrovní znalostí BPMN notace a počítačové gramotnosti. Všichni účastníci hodnotili aplikaci jako uživatelsky přívětivou a funkční. Několik problémů bylo identifikováno a následně opraveno, včetně nefunkčního stahování do SVG formátu a chybného zobrazení projektového menu.

Celkově lze konstatovat, že aplikace splňuje očekávání a požadavky kladené na její funkcionalitu a uživatelskou přívětivost. Byly identifikovány a opraveny drobné chyby, což přispělo k celkové stabilitě a spolehlivosti systému.

Kapitola 8

Závěr

V rámci této práce, jejímž cílem bylo vytvořit systém pro verzování BPMN modelů, jsme prošli základní teorii o notaci BPMN a blíže jsme si přiblížili samotný koncept verzování. Díky tomuto jsme si mohli porovnat existující řešení pro správu a verzování BPMN diagramů a mohli jsme vytvořit souhrn všech nutných požadavků pro takový verzovací nástroj s minimálními náklady pro koncové uživatele. Na základě této analýzy jsme zvolili technologie a architekturu, které vyhovují našim potřebám. Tento návrh jsme pak přetavili v konkrétní implementaci klientské a serverové části a celý systém důkladně otestovali.

Implementace zahrnovala použití moderních technologií, jako je React pro frontend, Spring Boot pro backend a PostgreSQL jako databázový systém. Díky těmto technologiím jsme vytvořili robustní a uživatelsky přívětivý systém, který umožňuje efektivní správu a verzování BPMN diagramů. Uživatelská rozhraní byla navržena s důrazem na intuitivní ovládání.

Systém umožňuje uživatelům vytvářet, upravovat, mazat a verzovat BPMN diagramy, s možností porovnání různých verzí a obnovy předchozích verzí. Důležitou součástí bylo také implementovat bezpečnostní prvky, včetně uživatelské autentizace a autorizace pomocí JWT tokenů, a správa přístupových práv k projektům a diagramům.

Testování systému probíhalo jak manuálně, tak automatizovaně, což nám umožnilo odhalit a opravit případné nedostatky a zajistit vysokou kvalitu konečného produktu.

Na závěr lze říci, že cíle práce byly splněny a výsledkem je funkční systém pro verzování BPMN diagramů, který je připraven k nasazení a použití v praxi. Tento projekt poskytuje solidní základ pro budoucí rozvoj a integraci dalších funkcionalit dle požadavků uživatelů a nových technologií.

Literatura

[1] Dumas, M., La Rosa, M., Mendling, J. and Reijers, H.A., 2018. *Fundamentals of business process management*. 2nd ed. Berlin, Germany: Springer Berlin. ISBN 9783662565087.

[2] Freund, J. and Rücker, B., 2019. *Real-Life BPMN (4th edition): Includes an introduction to DMN*. 4th ed. Camunda. ISBN 9781086302097.

[3] MacNeil, Caeleigh. Asana. "Why Projects Fail: 12 Common Causes and How to Prevent Them."[online]. 2024 [cit. 2024-04-14]. Dostupné z: `https://asana.com/resources/why-projects-fail`.

[4] International Organization for Standardization (ISO). "ISO/IEC 62652:2021(en)."[online]. 2013 [cit. 2024-03-08]. Dostupné z: `https://www.iso.org/standard/62652.html`.

[5] Object Management Group (OMG). Business Process Model and Notation (BPMN) Version 2.0. [online]. 2011 [cit. 2024-04-10]. Dostupné z: `https://www.omg.org/spec/BPMN/2.0/PDF`.

[6] IBM. Introduction to BPMN. [online]. 2022 [cit. 2024-04-10]. Dostupné z: `https://www.ibm.com/blog/bpmn/`.

[7] Edumax. "What Exactly Are Activities and Events in BPMN."[online]. 2024 [cit. 2024-04-22]. Dostupné z: `https://www.edumax.pro/blog/what-exactly-are-activities-and-events-in-bpmn`.

[8] Lucidchart. "BPMN Activity Types."[online]. n.d. [cit. 2024-04-22]. Dostupné z: `https://www.lucidchart.com/pages/bpmn-activity-types`.

[9] ValueBlue. "What Are BPMN Gateways?"[online]. 2022 [cit. 2024-04-22]. Dostupné z: `https://www.valueblue.com/knowledge-base/what-are-bpmn-gateways`.

[10] Krishnan, Krishna. "BPMN Connecting Objects, Pools, Swimlanes, and Artifacts."LinkedIn. [online]. 2022 [cit. 2024-04-22]. Dostupné z: `https://www.linkedin.com/pulse/bpmn-connecting-objects-pools-swimlanes-artifacts-krishna-krishnan`.

[11] Visual Paradigm. "BPMN 2.0 Tutorial - Swimlanes."[online]. 2016 [cit. 2024-04-22]. Dostupné z: https://www.visual-paradigm.com/tutorials/bpmn2.jsp.

[12] Interfacing. "BPMN 2.0 Symbology."[online]. n.d. [cit. 2024-04-22]. Dostupné z: https://www.interfacing.com/bpmn-2-0-symbology.

[13] Adel, Abbas. "The History of BPMN."Medium. [online]. 2023 [cit. 2024-04-20]. Dostupné z: https://abbasadel.medium.com/the-history-of-bpmn-6b539272ce10.

[14] Gliffy. "What is Business Process Model and Notation (BPMN)."[online]. 2024 [cit. 2024-04-20]. Dostupné z: https://www.gliffy.com/blog/what-is-business-process-model-notation-bpmn.

[15] Next4Biz. "What is BPMN 2.0 (Business Process Modeling Notation)."[online]. 2023 [cit. 2024-04-20]. Dostupné z: https://www.next4biz.com/what-is-bmpn-2-0-business-process-modeling-notation/.

[16] Atlassian. "What is Version Control?"[online]. n.d. [cit. 2024-04-24]. Dostupné z: https://www.atlassian.com/git/tutorials/what-is-version-control.

[17] Copado. "Version Control Principles."[online]. 2023 [cit. 2024-04-24]. Dostupné z: https://docs.copado.com/articles/#!copado-methodology-temp/version-control-principles.

[18] Lerin, Lennart. "Version Control Comprehensive Summary."Medium. [online]. 2024 [cit. 2024-05-03]. Dostupné z: https://lennart-lerin.medium.com/version-control-comprehensive-summary-adb1d60101aa.

[19] Agile Insider. "4 Principles to Follow About Document Version Control."Medium. [online]. 2024 [cit. 2024-04-20]. Dostupné z: https://medium.com/agileinsider/4-principles-to-follow-about-document-version-control-d9116b21f05e.

[20] React. "React – A JavaScript library for building user interfaces."[online]. 2024 [cit. 2024-04-27]. Dostupné z: https://react.dev.

[21] React. "React DOM."[online]. 2024 [cit. 2024-04-27]. Dostupné z: https://legacy.reactjs.org/docs/faq-internals.html.

[22] Rascasone. "React.js pro svižné a moderní weby a aplikace."[online]. 2024 [cit. 2024-04-27]. Dostupné z: https://www.rascasone.com/cs/blog/react-js-pro-svizne-a-moderni-weby-a-aplikace.

[23] ITNEXT. "React Ecosystem Guide."[online]. 2024 [cit. 2024-04-28]. Dostupné z: https://itnext.io/react-ecosystem-guide-4a5f85d17623.

[24] bpmn.io. "BPMN-js Walkthrough."[online]. 2024 [cit. 2024-05-02]. Dostupné z: https://bpmn.io/toolkit/bpmn-js/walkthrough/.

[25] Trisotech. "BPMN Examples."[online]. 2024 [cit. 2024-04-28]. Dostupné z: https://cloud.trisotech.com/bpmnquickguide/bpmn-quick-guide/bpmn-examples.html.

[26] Dilini, Nimesha. "Simple Introduction to Client-Server Architecture Concept."Medium. [online]. 2024 [cit. 2024-04-22]. Dostupné z: https://nimesha-dilini.medium.com/simple-introduction-to-client-server-architecture-concept-7d2979bed31d.

[27] GeeksforGeeks. "Introduction to Spring Boot."[online]. 2024 [cit. 2024-04-29]. Dostupné z: https://www.geeksforgeeks.org/introduction-to-spring-boot/.

[28] Sendbird. "WebSocket vs HTTP Communication Protocols."[online]. 2024 [cit. 2024-04-29]. Dostupné z: https://sendbird.com/developer/tutorials/websocket-vs-http-communication-protocols.

[29] GeeksforGeeks. "Unit Testing - Software Testing."[online]. 2024 [cit. 2024-05-05]. Dostupné z: https://www.geeksforgeeks.org/unit-testing-software-testing/.

[30] Katalon. "Manual Testing."[online]. 2024 [cit. 2024-05-05]. Dostupné z: https://katalon.com/resources-center/blog/manual-testing.

[31] Pojďme testovat. "Manuální testování."[online]. 2024 [cit. 2024-05-05]. Dostupné z: https://www.pojdmetestovat.cz/file/16.

[32] Baeldung. "JUnit 5 - Baeldung."[online]. 2024 [cit. 2024-04-22]. Dostupné z: https://www.baeldung.com/junit-5.

[33] AWS. "What is PostgreSQL?"[online]. 2024 [cit. 2024-04-22]. Dostupné z: https://aws.amazon.com/rds/postgresql/what-is-postgresql/.

[34] AWS. "The Difference Between Frontend and Backend."[online]. 2024 [cit. 2024-04-26]. Dostupné z: https://aws.amazon.com/compare/the-difference-between-frontend-and-backend/.

[35] BPMN-IO. "React BPMN. Github."[online]. 2024 [cit. 2024-05-11]. Dostupné z: https://github.com/bpmn-io/react-bpmn.

[36] BPMN-IO. "BPMN-JS Walkthrough."[online]. 2024 [cit. 2024-05-11]. Dostupné z: https://bpmn.io/toolkit/bpmn-js/walkthrough/.

[37] BPMN-IO. "BPMN Moddle. Github."[online]. 2024 [cit. 2024-05-11]. Dostupné z: https://github.com/bpmn-io/bpmn-moddle.

[38] BPMN-IO. "BPMN-JS Toolkit."[online]. 2024 [cit. 2024-05-11]. Dostupné z: https://bpmn.io/toolkit/bpmn-js/.

[39] BPMN-IO. "BPMN.IO Forum."[online]. 2024 [cit. 2024-05-11]. Dostupné z: https://forum.bpmn.io.

[40] Apache Maven. "What is Maven?"[online]. 2024 [cit. 2024-05-11]. Dostupné z: https://maven.apache.org/what-is-maven.html.

[41] Codex. "Detailed Comparison of Java Build Automation Tools: Maven vs Gradle."Medium. [online]. 2024 [cit. 2024-05-11]. Dostupné z: https://medium.com/codex/detailed-comparison-of-java-build-automation-tools-maven-vs-gradle-7b2ce4090

[42] IBM. "Java Spring Boot."[online]. 2024 [cit. 2024-05-12]. Dostupné z: https://www.ibm.com/topics/java-spring-boot.

[43] Spring.io. "Spring Community."[online]. 2024 [cit. 2024-05-12]. Dostupné z: https://spring.io/community.

[44] Spring.io. "Spring Security."[online]. 2024 [cit. 2024-05-13]. Dostupné z: https://spring.io/projects/spring-security.

[45] Baeldung. "REST with Spring Series."[online]. 2024 [cit. 2024-05-10]. Dostupné z: https://www.baeldung.com/rest-with-spring-series.

[46] Spring.io. "WebSocket Support in Spring."[online]. 2024 [cit. 2024-05-10]. Dostupné z: https://docs.spring.io/spring-framework/reference/web/websocket.html.

[47] Spring.io. "Spring Data JPA."[online]. 2024 [cit. 2024-05-10]. Dostupné z: https://spring.io/projects/spring-data-jpa.

[48] Baeldung. "Project Lombok."[online]. 2024 [cit. 2024-05-10]. Dostupné z: https://www.baeldung.com/intro-to-project-lombok.

[49] Baeldung. "Spring Boot Devtools."[online]. 2024 [cit. 2024-05-10]. Dostupné z: https://www.baeldung.com/spring-boot-devtools.

[50] Javatpoint. "Spring Boot Starter Web."[online]. 2024 [cit. 2024-05-10]. Dostupné z: https://www.javatpoint.com/spring-boot-starter-web.

[51] JWT.io. "Introduction to JSON Web Tokens."[online]. 2024 [cit. 2024-05-10]. Dostupné z: https://jwt.io/introduction.

[52] GeeksforGeeks. "STOMP Protocol."[online]. 2024 [cit. 2024-05-10]. Dostupné z: https://www.geeksforgeeks.org/stomp-protocol/.

[53] Spring Framework. "WebSocket Fallback Options."[online]. 2024 [cit. 2024-05-10]. Dostupné z: https://docs.spring.io/spring-framework/reference/web/websocket/fallback.html.

[54] Camunda. Camunda BPM - Features. [online]. 2024 [cit. 2024-05-01]. Dostupné z: https://camunda.com/products/bpm-platform/.

[55] Signavio. Signavio Process Manager. [online]. 2024 [cit. 2024-04-22]. Dostupné z: https://www.signavio.com/products/process-manager/.

[56] Bizagi. Bizagi Modeler. [online]. 2024 [cit. 2024-04-22]. Dostupné z: https://www.bizagi.com/en/platform/modeler.

[57] Sparx Systems. Enterprise Architect - Features. [online]. 2024 [cit. 2024-04-22]. Dostupné z: https://sparxsystems.com/products/ea/features/.

[58] IBM. IBM Blueworks Live. [online]. 2024 [cit. 2024-04-22]. Dostupné z: https://www.ibm.com/cloud/blueworks-live.

[59] GeeksforGeeks. "Difference Between Centralized Database and Distributed Database."[online]. 2024 [cit. 2024-05-10]. Dostupné z: https://www.geeksforgeeks.org/difference-between-centralized-database-and-distributed-database/.

[60] JetBrains. "WebStorm."[online]. 2024 [cit. 2024-05-12]. Dostupné z: https://www.jetbrains.com/webstorm/.

[61] JetBrains. "IntelliJ IDEA."[online]. 2024 [cit. 2024-05-12]. Dostupné z: https://www.jetbrains.com/idea/.

[62] Cawemo. Cawemo (WaybackMachine). [online]. 2024 [cit. 2024-04-22]. Dostupné z: https://web.archive.org/web/20240425170755/https://www.cawemo.com/.

[63] Camunda. End of Life Notification for Cawemo SaaS Service. [online]. 2024 [cit. 2024-04-22]. Dostupné z: https://camunda.com/blog/2024/02/camunda-retire-cawemo-integrated-solutions/.

[64] Clegg, Dave. "Prioritisation using MoSCoW."Dynamic Systems Development Method Consortium. [online]. 2014 [cit. 2024-05-12]. Dostupné z: https://www.agilebusiness.org/page/MoSCoW.