FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Bakalářská práce

Webový informační systém pro registraci členů veřejného spolku běží v cloudu

Studijní program: B0613A140005 Informační technologie
Studijní obor: Aplikovaná informatika

Autor práce: **Vladimir Stankov**
Vedoucí práce: Ing. Roman Špánek, Ph.D.
Ústav mechatroniky a technické informatiky

Konzultant práce: Ing. Petr Adámek
EmbedIT s.r.o.

Liberec 2024

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání bakalářské práce

Webový informační systém pro registraci členů veřejného spolku běží v cloudu

Jméno a příjmení:	**Vladimir Stankov**
Osobní číslo:	M20000018
Studijní program:	B0613A140005 Informační technologie
Specializace:	Aplikovaná informatika
Zadávající katedra:	Ústav mechatroniky a technické informatiky
Akademický rok:	2023/2024

Zásady pro vypracování:

1. Nastudujte agilní metodiky vývoje flexibilních webových aplikací běžících v cloudovém prostředí.
2. Seznamte se s doménou konkrétní aplikace pro správu členů veřejného spolku, s problematikou uchování osobní dat a zabezpečením takovéto aplikace.
3. Navrhněte a implementujte řešení této aplikace jak z pohledu frontendu, tak i backendu. Používejte CI/CD, připravte a implementujte flexibilní postupy pro nasazení a verzování.

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

Seznam odborné literatury:

[1] KOUCKÁ, Michaela. UX jako pokročilá strategie při tvorbě webu: UX as an advanced strategy in web design. Diplomové práce, vedoucí Jan Skrbek. Liberec: Technická univerzita v Liberci. 2017.

[2] DOMINGUS, Justin a ARUNDEL, John. Cloud native devOps with Kubernetes: building, deploying, and scaling modern applications in the cloud. Second edition. Beijing: O'Reilly, 2022. ISBN 978-1-0981-1682-8.

Vedoucí práce:	Ing. Roman Špánek, Ph.D. Ústav mechatroniky a technické informatiky
Konzultant práce:	Ing. Petr Adámek EmbedIT s.r.o.
Datum zadání práce:	12. října 2023
Předpokládaný termín odevzdání:	14. května 2024

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

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

V Liberci dne 12. října 2023

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

13. dubna 2024 Vladimir Stankov

Webový informační systém pro registraci členů veřejného spolku běží v cloudu

Abstrakt

Tato práce popisuje vývoj webového informačního systému pro registraci členů veřejného spolku, jehož provoz je zajištěn cloudovými technologiemi. Hlavním cílem práce je analýza moderních metodik vývoje aplikací a služeb. Práce také upozorňuje na problematiku uchovávání osobních údajů a bezpečnosti aplikace, což je přímo jeden z klíčových prvků vývoje jakékoli aplikace v moderním světě.
V rámci práce jsou zkoumány výhody použití metodiky Scrum pro správu vývojových procesů a detailně rozebrány technologie použité na frontendu a backendu systému, konkrétně React, Java 17 s frameworkem Spring Boot a databáze PostgreSQL.Kromě toho byly pro zajištění kontejnerizace a orchestrace aplikací použity Docker a Kubernetes, což umožnilo zvýšit efektivitu nasazení a provozu systému v cloudovém prostředí. Speciální pozornost je věnována otázkám bezpečnosti dat a bezpečnostních mechanismů jako OAuth 2.0 a JWT pro autentizaci a autorizaci uživatelů.
Výsledkem práce je plně funkční systém, který demonstruje, jak lze efektivně využít moderní technologie a metody pro zajištění vysoké úrovně bezpečnosti a škálovatelnosti aplikací.

Klíčová slova: Informační systém, agilní metodiky, Spring boot

Web-based information system for registration of members of a public association runs in the cloud

Abstract

This work describes the development of a web-based information system for the registration of members of a public association, operated with cloud technologies. The main goal of the thesis is the analysis of modern methodologies for developing applications and services. The work also addresses issues related to the storage of personal data and application security, which are directly one of the key elements in the development of any application in the modern world.
Within the work, the benefits of using the Scrum methodology for managing development processes are examined, and the technologies used in the system's frontend and backend are detailed, specifically React, Java 17 with the Spring Boot framework, and the PostgreSQL database. Additionally, Docker and Kubernetes were used to ensure the containerization and orchestration of applications, which enhanced the efficiency of deployment and operation of the system in a cloud environment. Special attention is given to data security issues and security mechanisms such as OAuth 2.0 and JWT for user authentication and authorization.
The result of the work is a fully functional system that demonstrates how modern technologies and methods can be effectively used to ensure a high level of security and scalability of applications.

Keywords: Information system, agile methodologies, Spring Boot

Poděkování

Rád bych poděkoval Ing. Romanu Špánkovi Ph.D. za velkou pomoc s vypracováním bakalářské práce. Dále bych velice rád poděkoval panu Ing. Petrovi Adámkovi za konzultaci a veškerou pomoc během vývoje aplikace.

Obsah

Seznam obrázků

Seznam zkratek

DSDM	Dynamic systems development method
FDD	Feature driven development
BDD	Behavior-driven development
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
SOA	Service-oriented architecture
ORM	Object-Relational Mapping
JPA	Java Persistence API
SQL	Structured Query Language
JSON	JavaScript Object Notation
XML	Extensible Markup Language
DOM	Document Object Model
API	Application Programming Interface
GKE	Google Kubernetes Engine
CSS	Cascading Style Sheets

1 Agile

Flexibilní metodiky vývoje softwaru, známé jako agile vývoj, představují souhrnný termín pro řadu přístupů a praktik založených na hodnotách Manifestu [3] agilního vývoje a 12 principech, které jsou jeho základem. Klíčovými zástupci agilních metodik jsou extrémní programování, DSDM, Scrum [5], Kanban [6], FDD, BDD a další, které jsou zaměřeny na minimalizaci rizik prostřednictvím realizace vývoje v sérii krátkých cyklů, nebo iterací, trvajících obvykle dva až tři týdny. Každá iterace představuje plnohodnotný softwarový projekt v miniaturách, zahrnující plánování, analýzu požadavků, návrh, programování, testování a dokumentaci, což umožňuje projektu být potenciálně připraveným k vydání na konci každé iterace.

Agilní metody zdůrazňují význam přímé komunikace a spolupráce, často v rámci jedné kanceláře, což umožňuje účastníkům projektu, včetně zákazníků, vývojářů, testerů a dalších specialistů, úzce spolupracovat mezi sebou. Hlavní metrikou úspěchu v agile je pracující produkt, což vede ke snížení objemu písemné dokumentace a urychlení vývojového procesu. Přes kritiku za nedostatek disciplíny, agilní metodologie prokázaly svou efektivitu v operativním a flexibilním řízení projektů různého rozsahu a složitosti.

V roce 2001 [10] 17 vývojářů softwaru publikovalo Manifest agilního vývoje softwaru, sdílející své zkušenosti v hledání efektivních metod vývoje. Agile transformuje velké projekty na malé, spravovatelné části, nazývané iterace, každá z nich přináší určitý výsledek a produkt, připravený k použití a zpětné vazbě od uživatelů. Tento přístup se liší od vodopádového [4] modelu řízení projektů, předpokládající paralelní a současnou práci designérů, vývojářů a obchodníků, což zajišťuje flexibilnější a adaptivnější rozvoj projektů.

1.1 Agilní principy

- Spokojenost zákazníka prostřednictvím brzkého a průběžného dodávání cenného softwaru.
- Vítejte změněné požadavky, i v pozdních fázích vývoje.
- Často dodávejte funkční software (týdny místo měsíců).
- Úzká, denní spolupráce mezi obchodními lidmi a vývojáři.
- Projekty jsou postaveny kolem motivovaných jednotlivců, kterým by mělo být důvěřováno.

- Osobní konverzace je nejlepší formou komunikace (společné umístění).
- Funkční software je primárním měřítkem pokroku.
- Udržitelný vývoj, schopný udržet stálé tempo.
- Nepřetržitá pozornost technické výbornosti a dobrému designu.
- Jednoduchost – umění maximalizovat množství práce, která není vykonána – je zásadní.
- Nejlepší architektury, požadavky a návrhy vznikají z autoorganizačních týmů.
- Pravidelně tým reflektuje, jak být efektivnější, a podle toho se přizpůsobuje.

1.2 Scrum

Podle statistiky [27] scrum se vyznačuje jako vedoucí a vysoce efektivní metodologie agilního vývoje softwaru, která ztělesňuje principy Agile tím, že nabízí iterativní a inkrementální přístup k řízení projektů [5]. Tento přístup zdůrazňuje flexibilitu, spolupráci a neustálé zlepšování. Scrum usnadňuje flexibilní pracovní prostředí v týmech, kde vývoj probíhá v iteracích s jasně definovanými cíli pro každou iteraci a konkrétními úkoly pro každého člena týmu. To umožňuje firmám efektivně aplikovat principy a hodnoty agilního řízení projektů.

Původně navržený pro optimalizaci procesů ve firmách zabývajících se vývojem softwaru a řízením produktů. Scrum byl od té doby přijat v různých odvětvích, včetně marketingu, brandingu a designu, čímž prokázal svou efektivitu jako vynikající rámec pro řízení dynamicky se vyvíjejících projektů. Podporuje autonomní práci na projektech namísto vykonávání úkolů diktovaných shora dolů.

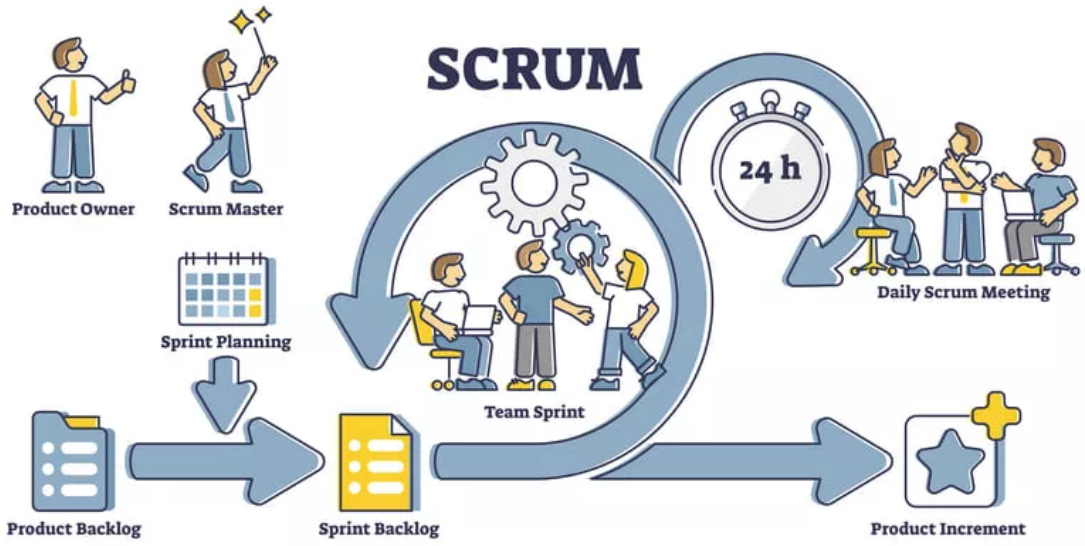
Scrum vede týmy k rozdělení práce na cíle, které mají být dosaženy během časově omezených iterací, známých jako sprinty, trvajících nejvýše jeden měsíc a obvykle dva týdny. Pro hodnocení pokroku se konají krátká, časově omezená setkání známá jako denní stand-upy, trvající nejvýše 15 minut. Na konci každého sprintu tým provádí dvě další schůzky: přezkum sprintu pro demonstraci práce zainteresovaným stranám a sběr zpětné vazby, a retrospektivu sprintu pro interní reflexi.

1.2.1 Role

V centru procesu Scrum stojí tři hlavní role [5] : Vlastník Produktu, Scrum Master a Vývojový tým. Vlastník Produktu nese odpovědnost za definování vize produktu a prioritizaci Produktového Backlogu, aby se ujistil, že tým pracuje na nejcennějších funkcích z hlediska byznysu. Scrum Master, na druhé straně, působí jako facilitátor a ochránce procesu Scrum, pomáhá týmu odstraňovat překážky a zlepšovat jejich pracovní procesy pro efektivnější práci. Vývojový tým, jakožto samoorganizující se a multidisciplinární, přebírá úkol vytváření produktu, pracuje na úkolech z Backlogu Sprintu během každého Sprintu.

1.2.2 Artefakty

Artefakty [5] ve Scrumu jsou určeny k zajištění transparentnosti klíčových aspektů projektu pro všechny zainteresované strany. Produktový Backlog [5] obsahuje seznam všech známých požadavků na produkt, řazených podle priority, což zajišťuje viditelnost toho, na čem bude pracováno. Backlog Sprintu je podmnožinou Produktového Backlogu, vybranou pro realizaci v aktuálním Sprintu, a představuje konkrétní plán práce týmu. Inkrement, nebo součet všech produktových vylepšení dodaných na konci Sprintu, demonstruje pokrok ve vývoji produktu a jeho připravenost k vydání.



Obrázek 1.1: Obrázek ilustrující Scrum cykly
https://gb.ru/blog/skrum/

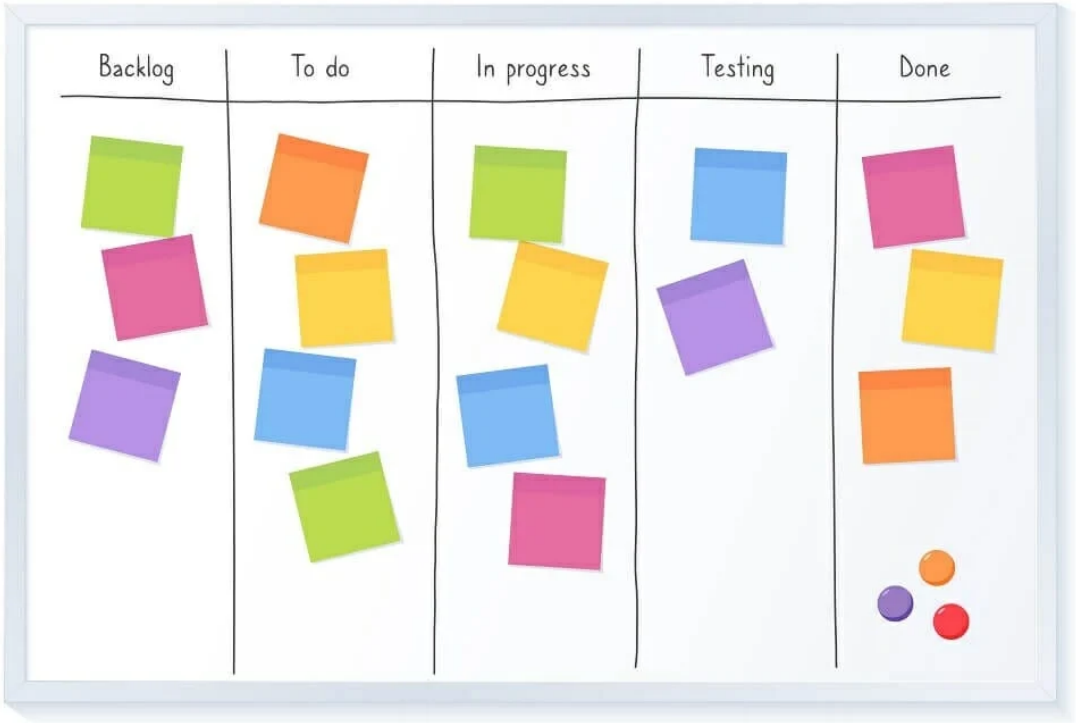
1.3 Kanban

Kanban, metoda původně vyvinutá v Japonsku ve 40. letech pro Toyota Production System [6], byla přizpůsobena pro procesy vývoje softwaru jako prostředek ke zlepšení výrobní efektivity. Založená na principech Lean [6], cílem Kanbanu je eliminovat plýtvání v práci, jako je nečinnost úkolů, nadměrný přenos mezi odděleními nebo vykonávání neaktuálních úkolů. Tato metoda umožňuje vizualizovat celý pracovní proces pomocí Kanban desky, rozdělené do sloupců, které reprezentují různé fáze procesu. Každý úkol nebo pracovní položka je reprezentována kartou, která se přesouvá mezi sloupci od začátku do konce, což usnadňuje sledování pokroku a identifikaci úzkých míst.

Na rozdíl od Scrumu, který je více strukturovaným frameworkem, předpokládajícím práci v iteracích (sprintech) a vyžadujícím vytvoření křížově funkčních týmů s jasně definovanými rolemi, Kanban nabízí flexibilitu a možnost postupného zlepšování stávajících procesů. Kanban nevyžaduje radikální změny ve struktuře týmu

nebo procesu na začátku, což umožňuje integraci do jakéhokoli existujícího pracovního procesu a jeho postupné zdokonalování. To činí Kanban obzvláště atraktivním pro řízení projektů s jasně definovanými úkoly, stejně jako pro servisní týmy, kde je důležité zjednodušit proces přijímání a plnění požadavků.

Kanban je vhodný jak pro řízení osobních úkolů, tak pro koordinaci práce na úrovni týmu nebo strategie společnosti. Umožňuje vytvořit předvídatelný a řiditelný tok práce, minimalizující nečinnost, zpoždění a nadbytečnou práci. Díky svému evolučnímu přístupu k rozvoji procesů Kanban podporuje vytvoření efektivního "výrobního pásu" tvorby hodnoty bez zbytečných nákladů a plýtvání.



Obrázek 1.2: Obrázek ilustrující Kanban desku
https://www.founderjar.com/kanban-board/

2 Cloud

V éře digitální transformace a neustále rostoucího objemu dat se cloudové technologie jeví jako klíčový prvek, umožňující organizacím všech velikostí efektivně škálovat a přizpůsobovat své IT infrastruktury měnícím se obchodním požadavkům. Cloudové výpočty představují nejen technologický posun v poskytování výpočetních zdrojů, ale také radikální změnu v přístupech k IT architektuře [7], vývoji softwaru a správě dat.

Ve světě, kde požadavky na operativnost, flexibilitu a ekonomickou efektivitu organizací neustále rostou, se cloudové technologie stávají nezbytným nástrojem pro dosažení těchto cílů. Cloudové technologie, zejména platformy pro kontejnerizaci, jako jsou Docker [8] a systémy pro orchestraci kontejnerů Kubernetes [9], nabízejí řešení pro snížení nákladů na IT infrastrukturu, zvýšení efektivity práce s daty a urychlení procesů vývoje a implementace nových aplikací a služeb.

2.1 Cloud Architecture

Cloudové architektury jsou obvykle kategorizovány podle modelů poskytování služeb [7], které definují stupeň kontroly, správy a úroveň abstrakce dostupné uživateli. Tyto modely zahrnují infrastrukturu jako službu (IaaS)[2], platformu jako službu (PaaS) a software jako službu (SaaS), nabízející od nízkoúrovňového řízení fyzických zdrojů v IaaS po vysokoúrovňovou abstrakci v SaaS.

Důležitou roli v definování metody poskytování zdrojů a služeb hrají modely nasazení cloudových architektur [7]. Jsou známy čtyři hlavní typy: veřejné, soukromé, hybridní a komunitní cloudu. Veřejné cloudu jsou dostupné široké veřejnosti přes internet, zatímco soukromé cloudu jsou vytvořeny pro interní použití v rámci jedné organizace. Hybridní cloudu představují směs soukromých a veřejných cloudů, zajišťující rovnováhu mezi řízením a flexibilitou, zatímco komunitní cloudu jsou určeny pro skupinu uživatelů se společnými požadavky.

Multicloudové architektury [7] jsou klíčovým prvkem v moderních cloudových strategiích, kombinující služby od různých poskytovatelů pro zlepšení výkonu, zvýšení úrovně spolehlivosti a minimalizaci závislosti na jediném dodavateli. Tento přístup umožňuje organizacím efektivněji spravovat své zdroje a přizpůsobit se měnícím se tržním podmínkám.

Architektura orientovaná na služby (SOA) [7] hraje důležitou roli ve vývoji cloudových aplikací, nabízí framework pro integraci a opětovné použití komponent, což je kriticky důležité pro vytváření škálovatelných a flexibilních cloudových řešení.

V kontextu cloudových výpočtů SOA podporuje vytváření modulárních aplikací, které mohou být snadno adaptovány a škálovány pro splnění obchodních potřeb.

2.2 Docker

Docker [8], platforma pro kontejnerizaci aplikací, umožňuje vývojářům balit aplikace společně s jejich závislostmi do standardizovaných jednotek pro vývoj softwaru, známých jako kontejnery. To zajišťuje konzistenci prostředí na všech etapách vývoje, testování a nasazení, což je kriticky důležité pro zajištění spolehlivosti a bezpečnosti cloudových aplikací. Kontejnerizace tak přímo přispívá k realizaci modelu cloudových výpočtů, poskytujíc flexibilitu a škálovatelnost, které jsou základními aspekty cloudových architektur.

2.3 Kubernetes

Kubernetes [9], na druhé straně, představuje systém pro automatizaci nasazování, škálování a správy kontejnerizovaných aplikací. Pracující v úzké integraci s Dockerem, Kubernetes usnadňuje orchestraci kontejnerů na rozsahu, který je vyžadován pro moderní cloudové aplikace. To umožňuje automaticky spravovat stav aplikací, zajišťující jejich dostupnost a škálovatelnost bez nutnosti zásahu operátora. Takto se Kubernetes stává klíčovým prvkem ve výstavbě cloudových architektur, podporující dynamické řízení zdrojů a optimalizaci provozu aplikací v cloudových prostředích.

3 Technologie

Výběr správných technologií je jedním z hlavních předpokladů každého úspěšného projektu, správně zvolené technologie společně s dobrým porozuměním základům zajišťují polovinu veškerého úspěchu. V této části práce budou prezentovány nástroje, které byly využity pro realizaci daného projektu. Bude pojednáno o jazycích, které byly použity, o frameworkách a externích technologiích, jež usnadňovaly vývoj.

3.1 Maven

Pro správu projektu a automatizaci sestavení aplikace byl zvolen Apache Maven, což je nástroj pro správu projektů a sestavování softwaru využívající Java. Maven poskytuje jednoduchý způsob, jak konfigurovat a sestavit Java projekty, spravovat závislosti, dokumentaci, distribuci a další aspekty vývoje softwaru.

3.2 Java

Java, široce uznávaná za svou robustnost, výkon a schopnost přenositelnosti mezi různými operačními systémy [11], představuje ideální volbu pro vývoj mnoha typů aplikací, včetně webových aplikací. Na základě principu "write once, run anywhere" (WORA) Java zajišťuje vysokou míru přenositelnosti kódu, což je dosaženo použitím Java Virtual Machine (JVM), která abstrahuje vykonávání kódu od konkrétního operačního systému.

Jako přísně objektově orientovaný jazyk Java napomáhá vytváření modulárního, flexibilního a snadno rozšiřitelného kódu, což je obzvláště ceněno při řízení velkých projektů a podpoře softwaru. Navíc Java disponuje rozsáhlou standardní knihovnou, která poskytuje vývojářům mnoho předem implementovaných funkcí pro vykonávání různorodých úkolů.

Z hlediska bezpečnosti byla Java navržena s ohledem na ochranu před různými hrozbami, včetně mechanismů pro ověřování byte kódu, správu paměti k zabránění únikům a ochranu před spuštěním škodlivého kódu. [11] Tyto bezpečnostní opatření, implementovaná na úrovni JVM, hrají klíčovou roli v zajištění bezpečného provozu programů.

3.3 Back-end framework

Pro strukturování a usnadnění vývoje backendu aplikace byl zvolen framework Spring Boot 3 [12]. Spring Boot je součástí širšího ekosystému Spring a poskytuje vývojářům rozsáhlé možnosti pro rychlý vývoj, konfiguraci a nasazení webových aplikací. Oproti jiným Java frameworkům, jako je Java EE (Jakarta EE) nebo Micronaut, Spring Boot vyniká svou vysokou mírou automatizace a konfigurace "out-of-the-box", která vývojářům umožňuje soustředit se přímo na business logiku aplikace, místo na konfiguraci prostředí a závislostí.

Spring Boot 3 přináší aktualizace, které jsou plně kompatibilní s Java 17 a využívají její nové funkce pro zlepšení výkonu a bezpečnosti aplikací. Mezi hlavní výhody Spring Boot patří:

- Rychlý start a snadná konfigurace: Spring Boot nabízí řadu předkonfigurovaných nastavení, které usnadňují rychlý start nových projektů a snižují potřebu manuální konfigurace.
- Široká podpora pro databázové operace: Integrace s JPA a Hibernate umožňuje efektivní práci s databázemi a snadné mapování objektů.
- Bezpečnost: Spring Security poskytuje robustní řešení pro autentizaci a autorizaci, ochranu proti běžným bezpečnostním hrozbám.
- Mikroslužby: Podpora pro vývoj mikroslužeb s možností snadného nasazení a škálování.

Oproti tomu, frameworky jako Jakarta EE nabízejí více "enterprise" funkcí, které mohou být v některých případech přínosné, ale často za cenu složitější konfigurace a nižší flexibility [13]. Micronaut nabízí srovnatelnou výkonnost a dobu startu s Spring Boot, ale nemá tak rozsáhlou komunitní podporu a ekosystém, což může být limitující pro vývoj a řešení specifických problémů. [14]

Ve výsledku byl Spring Boot 3 zvolen pro jeho vysokou produktivitu, podporu moderních Java verzí a široké možnosti pro vývoj škálovatelných a bezpečných webových aplikací, což jej činí ideálním nástrojem pro implementaci tohoto systému.

3.4 PostgreSQL

PostgreSQL není jen relační, ale objektově-relační databázový systém. [15] To mu dává některé výhody oproti jiným SQL databázím s otevřeným zdrojovým kódem, jako jsou MySQL, MariaDB a Firebird. Základní charakteristikou objektově-relační databáze je podpora uživatelských objektů a jejich chování, včetně datových typů, funkcí, operací, domén a indexů. [15] To činí PostgreSQL neuvěřitelně flexibilním a spolehlivým. Mimo jiné umí vytvářet, ukládat a načítat složité datové struktury. PostgreSQL prokazuje vysokou výkonnost při zpracování velkých objemů dat, podporuje složité dotazy, transakce a paralelní zpracování dat. [15] Také disponuje vestavěnými mechanismy pro zajištění integrity dat a podporuje širokou škálu

bezpečnostních funkcí, včetně pokročilého systému řízení přístupu, šifrování dat na úrovni sloupců a transportní úrovni. Je důležité poznamenat, že PostgreSQL úzce následuje standardy SQL a podporuje mnoho moderních funkcí, včetně JSON, XML, podpory fulltextového vyhledávání a uložených procedur. [15] To zajišťuje flexibilitu ve vývoji a možnost použití složité obchodní logiky na straně serveru.

3.5 React

React je JavaScriptová knihovna pro vývoj uživatelských rozhraní, kterou představil Facebook v roce 2013. React přinesl do webového vývoje řadu inovativních přístupů, mezi které patří komponentní přístup, virtuální DOM a deklarativní programování, což ho učinilo jedním z nejpopulárnějších nástrojů mezi vývojáři po celém světě. Základním principem Reactu je vytváření rozhraní jako nezávislých komponent, což zajišťuje vysokou míru opětovného použití kódu a usnadňuje údržbu a škálování aplikací. [16] Virtuální DOM implementovaný v Reactu umožňuje optimalizovat aktualizaci zobrazení stránky minimalizací interakce s reálným DOM, což je z hlediska výkonu nákladná operace. Deklarativní programování, které React nabízí, zjednodušuje vývojový proces tím, že umožňuje vývojářům popisovat, jak by komponenty rozhraní měly vypadat v různých stavech, namísto toho, aby popisovali, jak těchto stavů dosáhnout, což činí kód srozumitelnějším a snadněji laditelným.

3.6 Jira

Jira, vyvinutá společností Atlassian, je jednou z nejpopulárnějších a nejmocnějších systémů pro správu projektů a sledování chyb, který je široce využíván v průmyslu vývoje softwaru. Nabízí flexibilní nástroje pro plánování, sledování a správu jak agilních, tak tradičních projektů, což ji činí ideální volbou pro týmy uplatňující agilní metodologii Scrum. Jira umožňuje snadno vytvářet a spravovat produktové backlogy, plánovat a provádět sprinty, stejně jako sledovat pokrok práce v reálném čase.

3.7 Git

Charakteristickým rysem Gitu je používání ”snímků” pro zachycení stavu projektu v určitém okamžiku, což zajišťuje vysoký výkon a pohodlí při používání, a společně s využitím GitHubu, který je webovou službou založenou na Gitu, výrazně rozšiřuje možnosti práce na projektech, poskytující nástroje pro správu úkolů, code review a sledování chyb. Díky své integraci s Gitem se GitHub stal centrálním prvkem v ekosystému vývoje softwaru. Pro udržení strukturovanosti ve vývojovém procesu byl použit model větvení Git flow, který stanovuje přísná pravidla pro vytváření a slučování větví. Tento model usnadňuje správu verzí a přípravu vydání, činí proces vývoje organizovanějším a předvídatelnějším. Git Flow klade důraz na důležitost

oddělení pracovních procesů, což přispívá k zvýšení kvality a stabilitě vyvíjeného softwaru.

3.8 Github Copilot

Tento nástroj představuje revoluční řešení založené na umělé inteligenci, určené k urychlení procesu vývoje prostřednictvím automatického generování kódu na základě komentářů a částečně zadaného kódu vývojářem. GitHub Copilot [17] byl představen v roce 2021 a od té doby si získal značnou pozornost v softwarovém průmyslu. Založen na výkonném modelu GPT-3 od OpenAI, GitHub Copilot je schopen analyzovat existující kód a poskytovat doporučení k jeho doplnění, což umožňuje vývojářům výrazně snížit čas strávený psaním rutinního kódu a soustředit se na složitější aspekty projektu. Nástroj je integrován přímo do vývojového prostředí.

4 Rešení

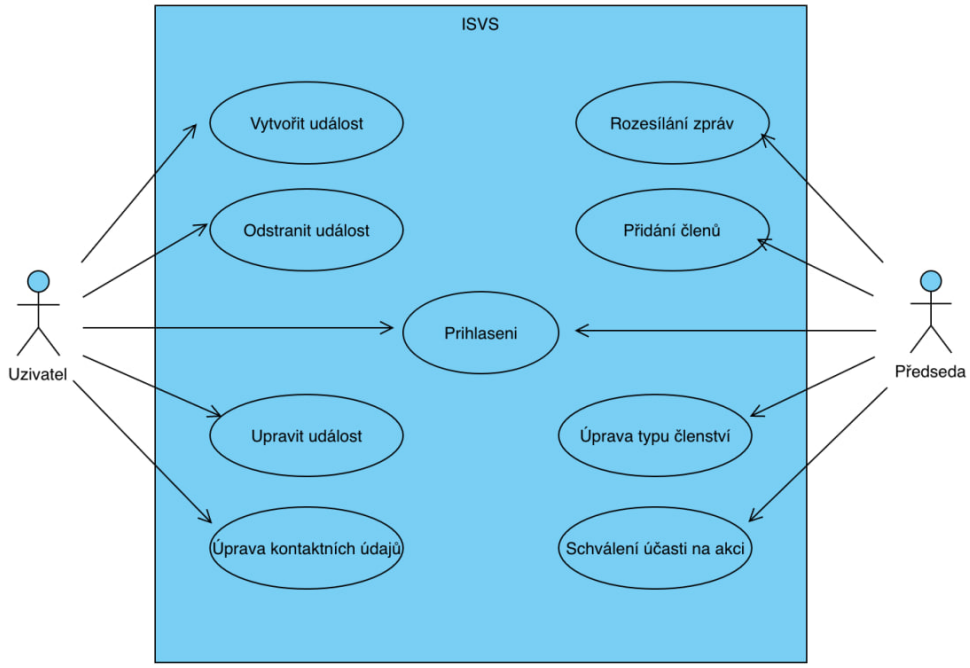
V této kapitole bude řeč o implementaci řešení a využití technologií a metodik z předchozích kapitol. Budou probrány klíčové funkce aplikace a popsána technologická realizace projektu vycházející z požadavků.

4.1 Požadavky

Informační systém je navržen tak, aby podporoval komplexní správu členství, autentizaci a autorizaci uživatelů, evidenci členských příspěvků a dalších závazků, organizaci akcí a událostí, a efektivní komunikaci mezi členy. Systém umožňuje evidenci členů spolku, kde členem může být jak fyzická, tak právnická osoba. Podporuje různé typy členství, jako jsou důchodci, studenti nebo právnické osoby, a umožňuje jedné osobě být členem více spolků. Členy do systému přidává administrátor. V oblasti autentizace a autorizace systém umožňuje přihlášení přes Google account a nezávislou registraci. Každý uživatel může zobrazovat a upravovat své kontaktní údaje. V rámci evidence akcí a událostí systém umožňuje členům spolku přihlašovat se na různé akce, jako například zájezdy, a může zahrnovat možnost přihlášení i pro nečleny. Přihlášení určitého typu člena může podléhat dodatečnému schválení. Pro efektivní komunikaci systém umožňuje hromadné rozesílání zpráv členům spolku nebo účastníkům konkrétní akce. Zprávy lze posílat jako SMS, e-mail nebo vyexportovat jako PDF pro tisk. Systém by měl obsahovat mechanismus pro definování formátu zprávy, zejména pro tisk.

4.1.1 Use-case diagram

Pro lepší pochopení a vnímání klíčových funkcí, které toto řešení nabízí, byl vytvořen use case diagram. Ilustruje možnosti, které může využívat běžný uživatel, stejně jako možnosti, které jsou dostupné předsedovi. Dá se říci, že předseda je administrátorem systému z hlediska rozhraní a dostupných funkcí v aplikaci. Jinými slovy, předseda má v aplikaci rozšířenou funkcionalitu.



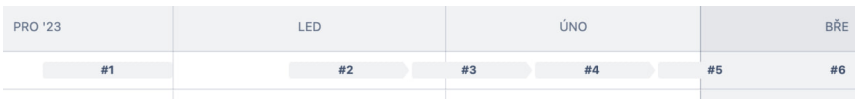
Obrázek 4.1: Use-Case diagram

4.2 Vývojový proces

Jak bylo zmíněno v předchozí kapitole o použitých technologiích, agilní vývoj byl podporován využitím Jira, což umožnilo efektivní správu a sledování dvoutýdenních sprintů. Každý sprint začínal výběrem úkolů, které byly pečlivě vybrány z Jira backlogu na základě priorit, které byly stanoveny panem konzultantem.

Během sprintu byl každý den využíván Jira board pro sledování pokroku a identifikaci překážek. Pomocí Jira byla snadno prováděna aktualizace stavu úkolů na ”In Progress” nebo ”Done” podle toho, jak probíhal vývoj a přidávání nové funkcionality.

Na konci každého sprintu byl proveden sprint review, kde byla prezentována dokončená práce přímo v Jira. Tato prezentace zahrnovala demonstraci nových funkcí a diskusi o splněných úkolech. Následně byla provedena retrospektiva, během které byly v Jira vytvořeny záznamy o zjištěních a akčních bodech pro další sprint. Tento proces umožnil nejen reflektovat nad tím, co bylo dosaženo, ale také identifikovat oblasti pro zlepšení a plánovat změny pro zvýšení efektivity v dalším sprintu.



Obrázek 4.2: Jira Spring Log

Jak je možné vidět na obrázku, vývoj probíhal od prosince do konce března. Na

kompletní vývoj bylo využito 6 sprintů. V průběhu vývoje byla kladená velká důraz na kvalitu kódu, testování a bezpečnost aplikace.

4.2.1 Verzování kódu

Verzování kódu je důležitým faktorem v procesu agilního vývoje. Jak již bylo řečeno v předchozí kapitole o technologiích, v rámci této práce bylo využito verzování pomocí Gitu a jeho webové služby GitHub. V projektu byla zavedena jasná strategie větvení, která byla založena na modelu Git Flow. Tento model definuje pevnou strukturu pro vytváření a správu větví, což umožňuje udržovat strukturu během procesu vývoje a přípravu nových verzí aplikace k vydání.

Feature branch: Pro každou novou funkci nebo úpravu byla vytvořena samostatná větev odvozená od hlavní vývojové větve (develop). To umožnilo pracovat na nových funkcích bez ovlivnění stabilní verze aplikace.

Develop branch: Sloužila jako hlavní větev pro vývoj, kam byly průběžně začleňovány všechny větve pro nové funkce po dokončení a revizi. Tato větev obsahovala nejaktuálnější vývojovou verzi aplikace.

Master branch: Reprezentovala stabilní verzi aplikace, která byla připravena k nasazení na produkční prostředí. Každé začlenění do této větve znamenalo vydání nové verze aplikace.

4.2.2 CI/CD

Continuous Integration (CI) a Continuous Deployment (CD) jsou nezbytné procesy v době rychlého vývoje softwaru. Přinášejí rychlost v nasazování nových funkcí, provedení oprav a komunikaci s dalšími službami, kterých je spousta v dnešní době. V rámci této práce bylo provádění těchto procesů zajištěno pomocí GitHub Actions. To je automatizační platforma integrovaná přímo do GitHubu. Nabízí flexibilní řešení pro automatizaci různých fází vývojového cyklu, od jednoduchých úkolů, jako je automatické spouštění testů, až po složitější procesy, jako je sestavování a nasazování aplikací. GitHub Actions využívají koncepty pracovních postupů, které jsou definovány v souborech YAML v repozitáři projektu. Tyto pracovní postupy jsou spouštěny na základě určitých událostí, jako například push do repozitáře. Jednou z dalších výhod GitHub Actions je to, že GitHub už má připravené a jednoduché konfigurace pro libovolný cíl.

4.3 Uchování osobních dat

4.3.1 GDPR

Při vývoji jakékoli aplikace, kde je od uživatelů vyžadováno, aby zanechali své osobní údaje, se vynořuje otázka jejich správného zpracování, uchovávání, předávání a mazání. Podle legislativy Evropské unie, zejména Obecného nařízení o ochraně osobních

údajů (GDPR) [18], které nabyl účinnosti 25. května 2018, jsou kladeny přísné požadavky na zpracování osobních údajů. V rámci vývoje této aplikace byly dodrženy hlavní aspekty, které se přímo týkají správného zpracování údajů z hlediska zákona.

Zákonnost, spravedlnost a transparentnost: Článek 5(1)(a) GDPR stanoví, že osobní údaje musí být zpracovávány zákonně, spravedlivě a transparentně ve vztahu k subjektu údajů.

Omezení účelů: Článek 5(1)(b) GDPR vyžaduje, aby osobní údaje byly shromažďovány pro jasně definované, legitimní účely a nesměly být dále zpracovávány způsobem, který není slučitelný s těmito účely.

Minimalizace údajů: Článek 5(1)(c) GDPR uvádí, že osobní údaje musí být adekvátní, relevantní a omezené na to, co je nezbytné pro účely, pro které jsou zpracovávány.

Přesnost: Článek 5(1)(d) GDPR vyžaduje, aby osobní údaje byly přesné a v případě potřeby aktualizované; musí být přijata všechna rozumná opatření k okamžitému vymazání nebo opravě osobních údajů, které jsou nepřesné vzhledem k účelům, pro které jsou zpracovávány.

Omezení uchovávání: Článek 5(1)(e) GDPR stanoví, že osobní údaje musí být uchovávány ve formě, která umožňuje identifikaci subjektů údajů, ne déle, než je nezbytné pro účely, pro které jsou osobní údaje zpracovávány.

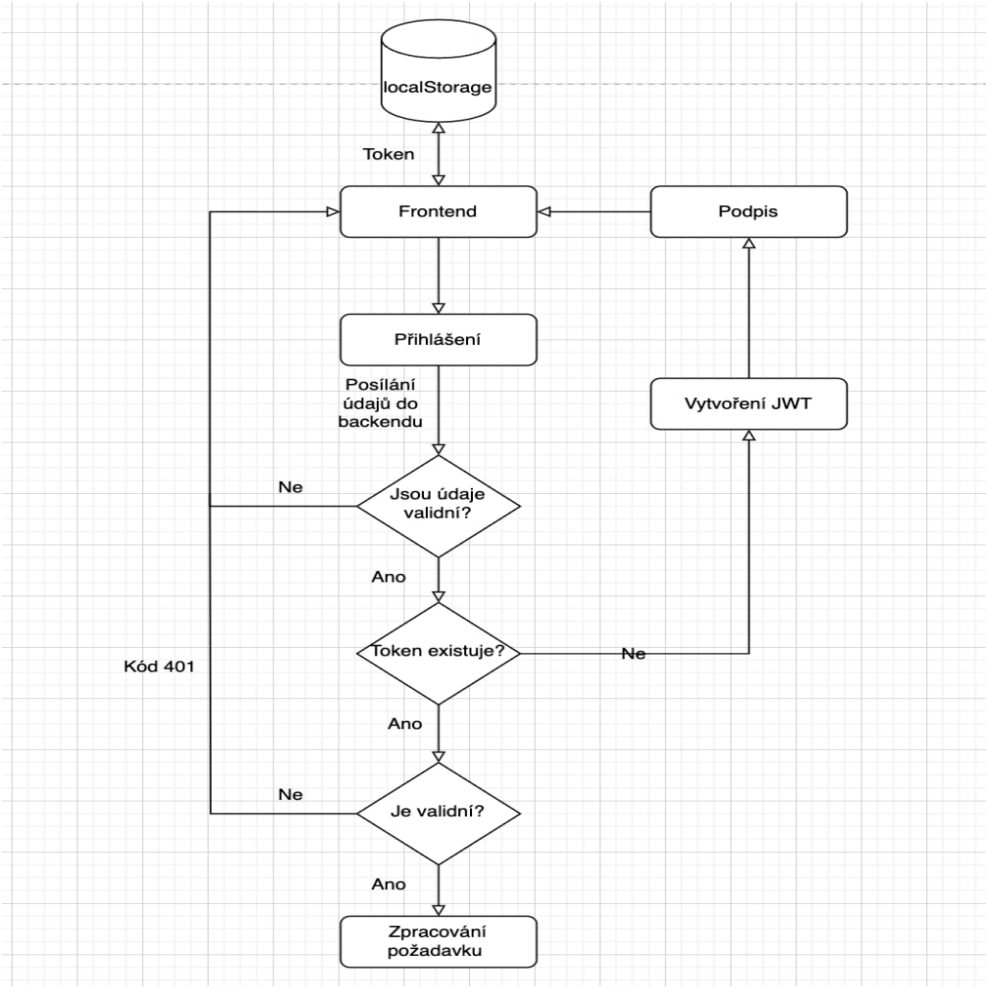
Integrita a důvěrnost: Článek 5(1)(f) GDPR stanoví, že osobní údaje musí být zpracovávány způsobem, který zajišťuje adekvátní bezpečnost osobních údajů, včetně ochrany před neoprávněným nebo nezákonným zpracováním a proti náhodné ztrátě, zničení nebo poškození, s použitím vhodných technických nebo organizačních opatření.

V rámci aplikace implementace dodržování GDPR, kromě bezpečného ukládání dat a získání souhlasu s zpracováním osobních údajů, zahrnuje možnost, že uživatel může kdykoliv napsat e-mailu zástupci této aplikace a požádat o smazání dat nebo využít práva být zapomenut.

4.3.2 Bezpečnost aplikace

Ve světě dnešních informačních technologií musí být každá aplikace na internetu, která si ukládá různé druhy dat, dobře chráněna proti různým zranitelnostem. Tato aplikace není výjimkou. Pro ochranu osobních údajů uživatelů a dodržení GDPR byla vyvinuta pravidla pro bezpečný přenos a ukládání dat.

Pro autorizaci uživatele byl vybrán JWT (JSON Web Token). JWT je otevřený standard (RFC 7519)[19], který definuje kompaktní a samostatný způsob bezpečného přenosu informací mezi stranami ve formě objektu JSON. Tyto informace lze ověřit a důvěřovat jim, protože jsou podepsány digitálním podpisem. Pro implementaci tohoto standardu byla vybrána knihovna Spring Security. Proces autorizace uživatele popsan na vývojovým diagramu:



Obrázek 4.3: Vývojový diagram

Je také třeba poznamenat, že pro maximální možnou bezpečnost aplikace je tajný klíč pro token JWT uložen na bezpečném místě, ke kterému má přístup pouze systémový administrátor.

Kromě tradiční formy autorizace pomocí přihlašovacího jména a hesla byla do aplikace také integrována autorizace prostřednictvím služeb Google, a to konkrétně prostřednictvím protokolu OAuth 2.0 [20]. OAuth 2.0 je otevřený standard autorizace, který umožňuje uživatelům poskytnout bezpečný delegovaný přístup ke svým účtům na různých službách, aniž by přitom odhalili své přihlašovací údaje. Místo toho jsou použity přístupové tokeny, které jsou vydány autorizačním serverem na žádost klienta.

Dalším aspektem bezpečnosti aplikace je ukládání hesel ve šifrované formě. Díky téže knihovně Spring Security byl použit algoritmus Bcrypt [21]. Bcrypt je funkce pro hašování hesel, která byla vyvinuta k ochraně hesel jejich převodem na jednosměrný hash. Tato metoda šifrování je obzvláště ceněna pro svou schopnost odolávat útokům hrubou silou díky možnosti nastavení ”náročnosti” hašování, která určuje složitost výpočtu hash. Jednou z hlavních charakteristik je jednosměrnost hašování, což znamená, že hash vytvořený pomocí Bcrypt nelze převést zpět na původní heslo,

což je bezpečný způsob ukládání hesel.

Další důležitou součástí zabezpečení aplikace je validace dat, která uživatel odesílá na backend. Aplikace používají mnoho validací k snížení možnosti SQL injekcí, jako například: kontrola délky řetězce, kontrola formátu, regulární výrazy.

Pokračujíc v předchozím aspektu validace dat, je třeba zmínit použití vynikajícího nástroje pro práci s databázemi - Hibernate [22]. Pro prevenci SQL injekcí Hibernate využívá mechanismy, jako jsou připravené dotazy, které oddělují SQL příkaz od dat, jež mají být vložena. To znamená, že data odeslaná v dotazu nejsou interpretována jako součást SQL kódu, což činí SQL injekce nesmyslnými. Útočník nemůže změnit strukturu dotazu vložením škodlivého kódu do dat. Také Hibernate nabízí vysoceúrovňová API pro tvorbu dotazů na databázi, která také přispívají k prevenci SQL injekcí. Tato API pracují na úrovni objektů a entit, nikoli přímých SQL dotazů, což minimalizuje riziko náhodného začlenění škodlivého kódu do dotazů. A díky ORM Hibernate řídí transformaci dat mezi databází a objektovým modelem aplikace. Toto abstrahování od přímých SQL dotazů snižuje pravděpodobnost, že vývojáři omylem zahrnou zranitelné konstrukce do svého kódu.

4.4 Notifikace

V rámci informačního systému je klíčovým prvkem efektivní komunikace s členy a účastníky různých akcí. Systém proto zahrnuje rozsáhlé možnosti notifikací, včetně odesílání SMS, e-mailů a generování PDF dokumentů pro tisk. Tyto funkce jsou nezbytné pro udržení plynulé a efektivní výměny informací mezi organizací a jejími členy nebo zájemci o akce.

4.4.1 SMS notifikace

Pro odesílání SMS zpráv systém využívá službu Twilio [23], což je cloudová platforma umožňující integraci různých typů komunikace do aplikací. Díky knihovně Twilio, která je integrována do Spring Boot aplikace, je možné automatizovat proces odesílání SMS zpráv členům a účastníkům akcí. Tato funkcionalita je obzvláště užitečná pro rychlé sdílení důležitých informací, jako jsou změny v plánovaných akcích, připomenutí blížících se událostí nebo specifické instrukce týkající se konkrétní akce.

Implementace SMS notifikací přes Twilio vyžaduje, aby aplikace byla konfigurována s platnými přihlašovacími údaji k Twilio API, které zahrnují Account SID a Auth Token. Po konfiguraci může aplikace vytvářet a odesílat SMS zprávy pomocí jednoduchého API volání, kde je specifikován odesílatel (číslo schválené Twilio), příjemce (telefonní číslo člena nebo účastníka) a text zprávy.

4.4.2 E-mailové notifikace

E-mailové notifikace jsou dalším základním pilířem komunikace v systému. Pro odesílání e-mailů aplikace využívá Spring Mail, což je součást Spring Frameworku,

umožňující snadnou integraci e-mailové komunikace do Java aplikací. Tato funkcionalita umožňuje systému odesílat personalizované e-maily jednotlivým členům nebo skupinám členů, ať už se jedná o informace o nadcházejících událostech, potvrzení registrace na akci, nebo jiné důležité oznámení.

4.4.3 Generování PDF

Pro účely tisku nebo archivace informací systém poskytuje možnost generování dokumentů ve formátu PDF. Tato funkce je užitečná pro vytváření reportů, potvrzení účasti, seznamů účastníků akcí a dalších dokumentů, které mohou být potřebné v tištěné formě nebo pro elektronickou distribuci ve formátu, který je snadno přístupný a univerzálně čitelný.

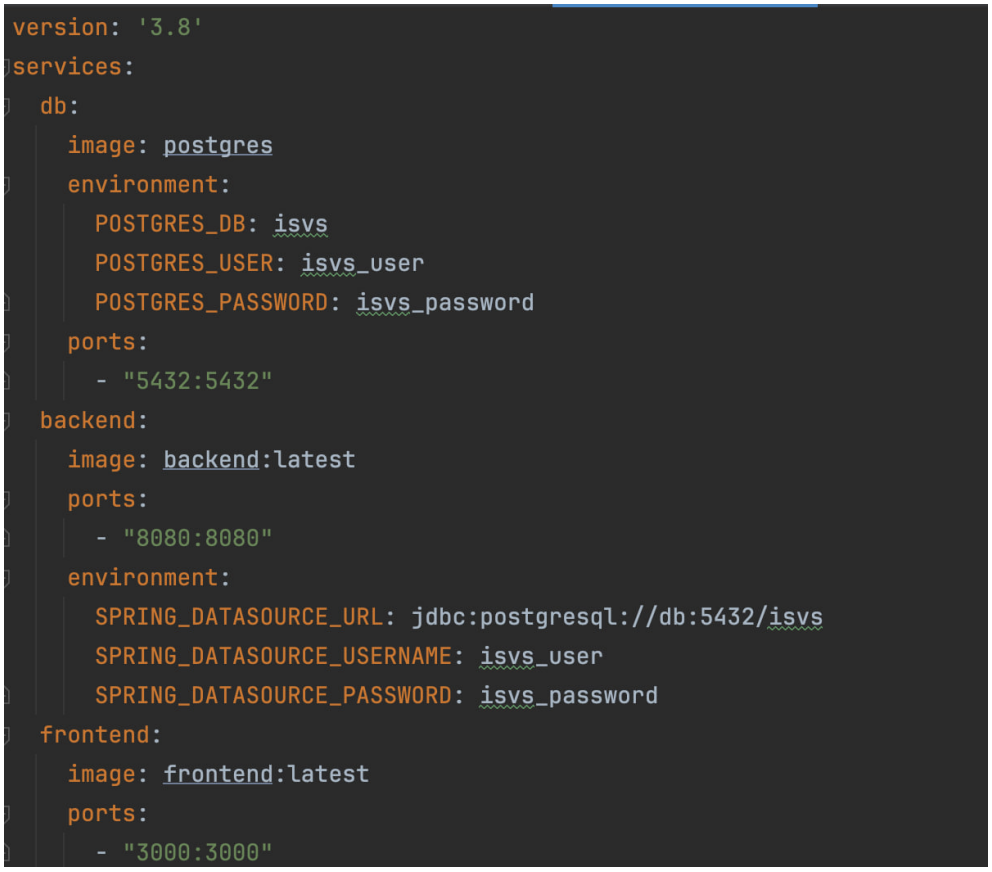
Generování PDF v aplikaci je realizováno s využitím knihovny iText [24], která umožňují vytváření a manipulaci s PDF dokumenty v Java aplikacích. Tato knihovna nabízí širokou škálu funkcí pro práci s textem, grafikou a layoutem dokumentů, což umožňuje vytvářet profesionálně vypadající dokumenty odpovídající potřebám organizace.

4.5 Infrastruktura

V této kapitole bude popsáno praktické využití technologií, o kterých bylo hovořeno v teoretické části, konkrétně Docker, Kubernetes, a také bude názorně ukázáno fungování GitHub Actions, který řídí aspekty testování, sestavení a nasazení aplikace v GKE [25].

4.5.1 Použití Dockeru

Jak již bylo řečeno v kapitole o cloudových technologiích, Docker je platforma pro kontejnerizaci aplikací, která přináší mnoho výhod, jako je balení se všemi potřebnými závislostmi, konzistence prostředí, flexibilita a škálovatelnost. Během vývoje a testování této aplikace, ověřující správnost fungování a komunikaci mezi moduly frontendu a backendu, byl použit Docker Compose. Docker Compose [27] je nástroj pro definici a správu vícekontejnerových Docker aplikací. Pomocí jednoduchého YAML souboru pro konfiguraci umožňuje Docker Compose uživatelům instalovat, spouštět a spravovat aplikace, které se skládají z souboru vzájemně propojených kontejnerů.



Obrázek 4.4: Docker Compose

Jak lze vidět na snímku obrazovky, konfigurace Docker Compose je jednoduchá a efektivní. Pokud se podíváme hlouběji do obsahu tohoto souboru, lze vidět, že jsou použity tři služby: db, backend a frontend. Pro správné a spolehlivé fungování každé ze služeb je nutné mít obraz (image) služby. Například obraz databáze je přímo z Docker Hub, který je container registry, jinými slovy je to veřejná služba, kde se nacházejí různé obrazy služeb, jako je například PostgreSQL. Zatímco databázi lze získat z Docker Hub, s backendem a frontendem to není možné. Proto je nutné mít vlastní Dockerfile, ve kterých je potřeba určit sekvenci příkazů pro úspěšnou sestavení obrazu.

4.5.2 Použití Kubernetesu

Pro správnou funkci služby v Kubernetes je nezbytné mít nasazení (deployment) a službu (service). Deployment v Kubernetes je zdroj, který zajišťuje deklarativní aktualizace stavu aplikací. Je zodpovědný za správu verzí a aktualizací, škálování a samovolné obnovení. A služba je abstrakce, která definuje logickou sadu podů a politiku přístupu k nim. Umožňují například vyvažování zátěže a zajišťují abstrakci a izolaci. Při vývoji této aplikace bylo rozhodnuto psát službu nebo deployment v samostatných souborech. Na jednu stranu to umožňovalo rychle se v nich orientovat a mít pouze to, co přímo souvisí s funkcí souboru, ale na druhou stranu kvůli tomu,

že aplikace obsahuje tři důležité komponenty: frontend, backend a databázi, a pro každou z nich je potřebný servis a deployment, vzniklo 6 souborů, což se může zdát zbytečné. V příkladu budou uvedeny pouze služba a deployment pro frontend, jinak by to vyžadovalo několik dalších stránek. Obecně se liší pouze drobnými úpravami v konfiguraci pro backend nebo databázi, ale podstata zůstává stejná.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: isvs-frontend-service
spec:
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 3000
  selector:
    app: isvs-frontend
```

Obrázek 4.5: Frontend Service YAML

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: isvs-frontend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: isvs-frontend
  template:
    metadata:
      labels:
        app: isvs-frontend
    spec:
      containers:
        - name: isvs-frontend
          image: europe-west1-docker.pkg.dev/isvs-408717/isvs-registry/isvs-frontend:latest
          ports:
            - containerPort: 3000
```

Obrázek 4.6: Frontend Deployment YAML

Jak je možné vidět na snímcích obrazovky výše, konfigurace není příliš velká a je poměrně jednoduchá. Zajímavé body, které lze zdůraznit:

- Service: Typ služby byl vybrán LoadBalancer, protože zajišťuje integraci s cloudovou infrastrukturou pro automatické nastavení vnějšího vyvažovače

zatížení, což usnadňuje přístup ke službám z vnější sítě bez nutnosti ručního nastavování směrování nebo otevírání portů na každém uzlu.

- Deployment: Ve souboru pro nasazení, v sekci kontejnerů, je obrázek odkazován na externí úložiště pro obrázky aplikací. Při nasazování této aplikace byly využity zdroje od Google, který disponuje velmi pohodlným Artifact Registry [29], a proto byl tento registr využit.

4.5.3 Použití GitHub Actions

Jak již bylo řečeno v sekci o CI/CD, pro tuto implementaci byl použit GitHub Actions. Konfigurační soubor je rozdělen do dvou částí. První část zahrnuje přípravu prostředí, spuštění testů a sestavení aplikace. Druhá část přímo závisí na první, což znamená, že pokud první část skončí selháním, tedy testy neprojdou nebo dojde k chybám při sestavení aplikace, druhá část se ani nespustí. Toto nastavení umožňuje vyhnout se nežádoucím komplikacím nebo problémům. Druhá část je zodpovědná za sestavení aplikace do Docker obrazů a jejich nasazení v GKE. V rámci CI/CD je také používán GitHub Secrets, díky kterému je možné skrývat důležité informace používané v projektu, například autorizační údaje v GKE [2].

```yaml
deploy:
  needs: build-and-test
  name: Deploy
  runs-on: ubuntu-latest
  environment: production

  steps:
    - name: Checkout
      uses: actions/checkout@v4

    # Setup gcloud CLI
    - uses: google-github-actions/setup-gcloud@1bee7de035d65ec5da40a31f8589e240eba8fde5
      with:
        service_account_key: ${{ secrets.GKE_SA_KEY }}
        project_id: ${{ secrets.GKE_PROJECT }}
        export_default_credentials: true

    - id: 'auth'
      uses: 'google-github-actions/auth@v0'
      with:
        credentials_json: ${{ secrets.GKE_SA_KEY }}

    - uses: google-github-actions/get-gke-credentials@db150f2cc60d1716e61922b832eae71d2a45938f
      with:
        cluster_name: ${{ env.GKE_CLUSTER }}
        location: ${{ env.GKE_ZONE }}
        credentials: ${{ secrets.GKE_SA_KEY }}
```

Obrázek 4.7: CI/CD Druhá část (1)

```yaml
- name: Download Artifacts
  uses: actions/download-artifact@v2
  with:
    name: isvs-backend-0.0.1-SNAPSHOT.jar
    path: isvs-backend/target/

- name: Build and Push Backend Docker Image
  run: |
    cd isvs-backend/
    gcloud auth configure-docker europe-west1-docker.pkg.dev
    docker build -t europe-west1-docker.pkg.dev/isvs-408717/isvs-registry/isvs-backend:latest .
    docker push europe-west1-docker.pkg.dev/isvs-408717/isvs-registry/isvs-backend:latest
- name: Build and Push Docker images
  run: |
    gcloud auth configure-docker europe-west1-docker.pkg.dev
    cd isvs-frontend/
    docker build -t europe-west1-docker.pkg.dev/isvs-408717/isvs-registry/isvs-frontend:latest .
    docker push europe-west1-docker.pkg.dev/isvs-408717/isvs-registry/isvs-frontend:latest

- name: Deploy to GKE
  run: |
    gcloud components install gke-gcloud-auth-plugin
    gcloud container clusters get-credentials isvs --zone europe-west1
    sed -i "s/\$PROJECT_ID/${{ secrets.GKE_PROJECT }}/g" k8s/*
    kubectl apply -f k8s --validate=false
```

Obrázek 4.8: CI/CD Druhá část (2)

Prezentované snímky obrazovky demonstrují proces nasazování v Google Kubernetes Engine (GKE). Pro jeho úspěšné provedení je nutné vytvořit cluster a nastavit servisní účet, který je nezbytný pro autorizaci. Do GitHub Secrets byla přidána klíčová data pro proces: klíč servisního účtu, identifikátor projektu v GKE, název clusteru a jeho umístění. Nasazení zahrnuje několik kroků: nastavení rozhraní příkazového řádku Google Cloud (gcloud CLI), které se později používá pro vytvoření Docker obrazů a jejich nasazení, a také nastavení autorizace. Pro demonstraci byl název projektu a jeho umístění během procesu vytváření Docker obrazů uvedeny přímo, bez použití proměnných z GitHub Actions. Konkrétně se jedná o název projektu (isvs-408717) a umístění (europe-west1).

4.6 UX Design

Uživatelé hrají klíčovou roli v procesu vývoje každé aplikace, protože nejenže ji používají, ale také přispívají k jejímu růstu a možnosti vydělávat peníze. Atraktivita a pohodlí aplikace jsou důležité, aby uživatelé chtěli aplikaci pravidelně používat. Proto bylo již na začátku návrhu tohoto projektu velký důraz kladen na definování klíčových funkcí a požadavků uživatelů. [1] Tento proces zahrnoval nejen určení, které funkce jsou potřebné, ale také jejich správné začlenění do systému, aby se maximalizovalo pohodlí uživatelů. Jak již bylo dříve řečeno, pro uživatelskou část aplikace byl použit React, který umožňuje připojovat různé knihovny pro pohodlí vytváření uživatelských rozhraní. Existuje mnoho různých knihoven, například Bootstrap (React Bootstrap nebo Reactstrap), Tailwind CSS, Material-UI a Ant Design, který byl použit v této aplikaci. Ant Design je rozsáhlá knihovna komponent s sadou

designových vzorů a standardů pro aplikace. Nabízí detailně vypracované komponenty a nástroje pro vytvoření kvalitního rozhraní. Díky této knihovně se podařilo vytvořit pohodlné, hezké a intuitivně srozumitelné rozhraní. (viz přílohy)

4.7 Testovaní

Testování aplikace je důležitou součástí vývojového procesu. Ujistí nás, že při změnách v kódu bude aplikace fungovat tak, jak bylo zamýšleno. Při provádění změn je důležité brát v úvahu kontext práce; jakékoli změny nějakým způsobem ovlivňují aplikaci a chyby mohou mít vážné důsledky pro její funkčnost. Napsané testy umožňují upozornit vývojáře na problémy a nesrovnalosti v koncepci aplikace. V této aplikaci byly pro testování backendu použity knihovny jako JUnit 5 a Mockito. Také byl pro hodnocení pokrytí použit JaCoCo. A relativně nová technologie v dnešním světě - GitHub Copilot. Jak již bylo řečeno v kapitole o technologiích, GitHub Copilot je nástroj, který představuje revoluční řešení založené na umělé inteligenci, určené k urychlení vývojového procesu prostřednictvím automatického generování kódu na základě komentářů a částečně zadaného kódu vývojářem. Při psaní testů GitHub Copilot pomáhal generovat šablony testů a nabízel varianty kontrol, což výrazně urychlilo proces psaní testů. Copilot se ukázal být obzvláště užitečný při psaní parametrizovaných testů a mockování složitých závislostí. Je nerealistické očekávat testování každého jednotlivého řádku kódu a zároveň by nemělo být cílem testovat absolutně vše bez rozlišení. V testech je důležité rozumět tomu, jak by komponenty měly fungovat a jak by neměly. Také není smysluplné testovat jednoduchý kód, který je zřejmý na první pohled.

Závěr

V rámci této bakalářské práci byl úspěšně vyvinut a implementován systém pro registraci členů veřejného spolku, což demonstruje efektivitu používání moderních technologií a metodik ve vývoji softwaru. Použití cloudových technologií, kontejnerizace a orchestrace prostřednictvím Dockeru a Kubernetes umožnilo zabezpečit vysokou úroveň škálovatelnosti a spolehlivosti systému.

Použití metodiky Scrum přispělo k optimalizaci vývojového procesu a zvýšilo jeho adaptabilitu na měnící se požadavky projektu. Zvláštní pozornost byla v práci věnována otázkám bezpečnosti: byly integrovány moderní mechanismy autentizace a autorizace, jako jsou OAuth 2.0 a JWT, což výrazně zvýšilo úroveň ochrany osobních údajů uživatelů.

Zadavatel projektu vyjádřil vysokou míru spokojenosti s výsledným systémem, zejména ocenil jeho robustnost a schopnost efektivně reagovat na dynamické požadavky uživatelů. Kladně hodnotil také implementaci bezpečnostních prvků, které značně přispěly k ochraně dat a soukromí členů spolku. Toto ocenění a spokojenost zadavatele potvrzují, že cíle práce byly úspěšně splněny a představují solidní základ pro další rozvoj a údržbu systému.

Bibliografie

[1] KOUCKÁ, Michaela. UX jako pokročilá strategie při tvorbě webu: UX as an advanced strategy in web design. Diplomové práce, vedoucí Jan Skrbek. Liberec: Technická univerzita v Liberci.

[2] DOMINGUS, Justin a ARUNDEL, John. Cloud native devOps with Kubernetes: building, deploying, and scaling modern applications in the cloud. Second edition. Beijing: O'Reilly, 2022. ISBN 978-1-0981-1682-8.

[3] Beck, K., et al. (2001) The Agile Manifesto. Agile Alliance. http://agilemanifesto.org/

[4] W. W. Royce. 1987. Managing the development of large software systems: concepts and techniques. In Proceedings of the 9th international conference on Software Engineering (ICSE '87). IEEE Computer Society Press, Washington, DC, USA, 328–338.

[5] KENNETH, Rubin. Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison-Wesley Professional, 2012. ISBN 978-0137043293.

[6] ANDERSON, David J. Kanban: Successful Evolutionary Change for Your Technology Business. Blue book ed. Blue Hole Press, 2010. ISBN 0984521402.

[7] ERL, Thomas, Ricardo PUTTINI a Zaigham MAHMOOD. Cloud Computing: Concepts, Technology & Architecture. Pearson, 2013. ISBN 9780133387520.

[8] Docker [online], 2024. [cit. 2024-03-03]. Dostupné z: https://docs.docker.com/

[9] Kubernetes, 2024. Kubernetes Docs [online]. [cit. 2024-03-03]. Dostupné z: https://kubernetes.io/docs/home/

[10] Agile software development, 2024. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-2024 [cit. 2024-03-03]. Dostupné z: https://en.wikipedia.org/wiki/Agile_software_development

[11] The Java Language Environment, 1995. Https://www.oracle.com/ [online]. [cit. 2024-04-01]. Dostupné z: https://www.oracle.com/java/technologies/introduction-to-java.html#334

[12] Spring Boot, 2015. Https://www.spring.io/ [online]. [cit. 2024-04-01]. Dostupné z: https://spring.io/projects/spring-boot

[13] JIBU, James, 2023. Java EE vs Spring Boot: What are the Differences. Https://www.sayonetech.com/ [online]. [cit. 2024-04-01]. Dostupné z: https://www.sayonetech.com/blog/java-ee-vs-spring-boot-what-are-differences/

[14] OZER, Cahit, 2023. Which of the Java Frameworks should you use? Https://medium.com/ [online]. [cit. 2024-04-01]. Dostupné z: https://medium.com/codex/which-of-the-java-frameworks-should-you-use--683c7e710f9

[15] PostgreSQL Documentation, 1996. Https://www.postgresql.org/ [online]. [cit. 2024-04-01]. Dostupné z: https://www.postgresql.org/docs/16/intro-whatis.html

[16] React Framework Documentation, 2024. Https://react.dev/ [online]. [cit. 2024--04-01]. Dostupné z: https://react.dev/reference/react

[17] GitHub Copilot, 2024. Https://github.com/ [online]. [cit. 2024-04-01]. Dostupné z: https://github.com/features/copilot

[18] General Data Protection Regulation [online], 2016. [cit. 2024-04-01]. Dostupné z: https://gdpr-info.eu/

[19] JONES, Michael B., John BRADLEY a Nat SAKIMURA, 2015. JSON Web Token (JWT). RFC Editor [online]. [cit. 2024-04-01]. Dostupné z: https://www.rfc--editor.org/info/rfc7519

[20] OAuth 2.0 [online]. [cit. 2024-04-01]. Dostupné z: https://oauth.net/2/

[21] Password Storage, 2015. Https://docs.spring.io [online]. [cit. 2024-04-01]. Dostupné z: https://docs.spring.io/spring-security/reference/features/authentication/password-storage.html

[22] Hibernate Documentation, 2024. Https://hibernate.org/ [online]. [cit. 2024-04--01]. Dostupné z: https://hibernate.org/orm/documentation/6.4/

[23] Twillio Documentation, 2024. Https://www.twilio.com/ [online]. [cit. 2024-04--01]. Dostupné z: https://www.twilio.com/docs/all

[24] IText Documentation, 2024. Https://itextpdf.com/ [online]. [cit. 2024-04-01]. Dostupné z: https://itextpdf.com/resources/api-documentation

[25] GKE overview, 2024. Https://cloud.google.com/ [online]. [cit. 2024-04-01]. Dostupné z: https://cloud.google.com/kubernetes-engine/docs/concepts/kubernetes-engine-overview

[26] Docker Compose overview, 2024. Https://docs.docker.com/ [online]. [cit. 2024--04-01]. Dostupné z: https://docs.docker.com/compose/

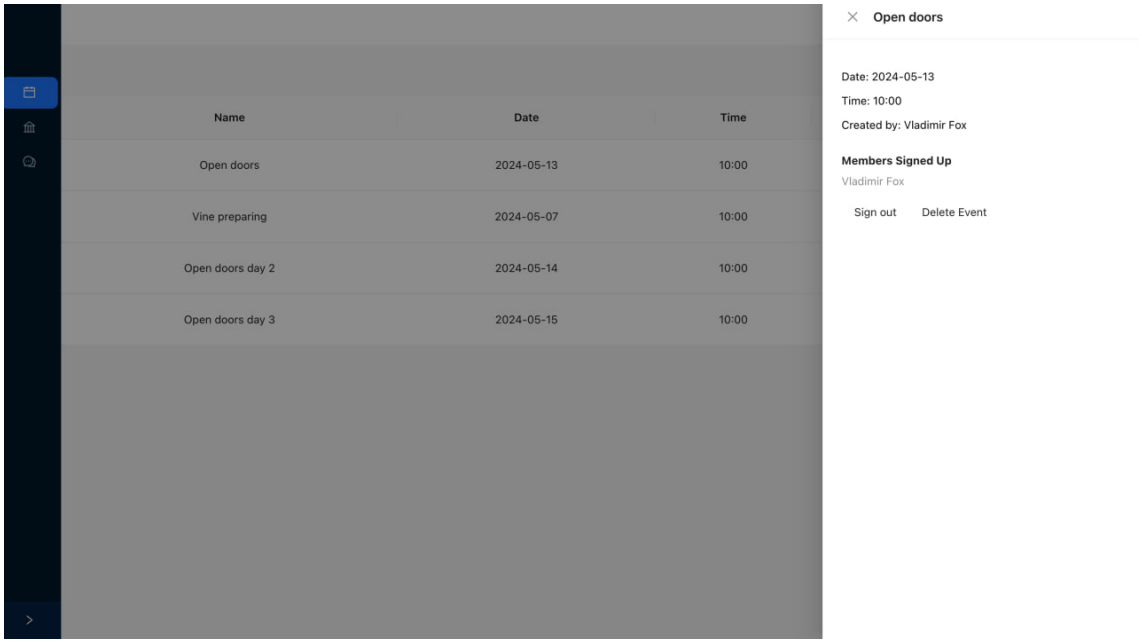
[27] 16th Annual State of Agile Report, 2024. Https://info.digital.ai [2022]. [cit. 2024-04-01]. Dostupné z: https://info.digital.ai/rs/981-LQX-968/images/AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf

[28] Artifact Registry, 2024. Https://cloud.google.com/ [online]. [cit. 2024-04-01]. Dostupné z: https://cloud.google.com/artifact-registry

A Přílohy

A.1 Ukázka uživatelského rozhraní

Obrázek A.1: Hlavni okno



Obrázek A.2: Vytvareni akce