Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Webová aplikace pro asistenci při hodnocení odevzdaných úloh

Bc. Ondřej Chrbolka

Diplomová práce
2023

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Ondřej Chrbolka**
Osobní číslo:	**I20205**
Studijní program:	**N0613A140007 Informační technologie**
Téma práce:	**Webová aplikace pro asistenci při hodnocení odevzdaných úloh**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování

Cílem diplomové práce je navrhnout a implementovat webovou aplikaci pro usnadnění hodnocení odevzdaných prací studentů (např. zápočtů, zkoušek apod).
Předpokladem je navržení systému, který bude disponovat šablonou pro hodnocení s předdefinovanými texty a bude vyučujícímu asistovat při hodnocení nejrůznějších úloh (např. napovídat texty pro hodnocení jednotlivých částí úloh). Šablony mohou být definovány uživatelsky nebo mohou být definovány kombinovaně (částečně automaticky na základě definovaného zdroj. kódu, částečně uživatelsky).
Na výstupu systém poskytne celkové slovní hodnocení úlohy, dále systém bude uchovávat hodnocení se základním statistickým hromadným hodnocení prací z minulosti – jak hromadným, tak pro jednotlivé části úloh za účelem poskytnutí zpětné vazby vyučujícímu.
Předpokladem je, že část systému s hodnocením bude budována v podobě přístupného API také pro možnost vytváření alternativních aplikací.

Rozsah pracovní zprávy: **50-60 normostran**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. TURNQUIST L. G.Learning Spring Boot 2.0 – Second Edition: Simplify the development of lightning fast applications based on microservices and reactive programming. Packt Publishing, 2017, 370 pp. ISBN978-1786463784.
2. PORCELLO E, BANKS A. Learning React: Modern Patterns for Developing React Apps . O'Reilly Media; 2nd Edition, 2020, 310 pp. ISBN: 978-1492051725.

Vedoucí diplomové práce: **doc. Ing. Michael Bažant, Ph.D.**
Katedra softwarových technologií

Datum zadání diplomové práce: **8. listopadu 2021**
Termín odevzdání diplomové práce: **20. května 2022**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

prof. Ing. Antonín Kavička, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 30. listopadu 2021

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

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

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 25. 5. 2023

Bc. Ondřej Chrbolka

PODĚKOVÁNÍ

Tímto bych rád poděkoval doc. Ing. Michaelu Bažantovi, Ph.D. za vedení diplomové práce a za všechny poskytnuté cenné rady.

ANOTACE

Diplomová práce se zaměřuje na návrh a implementaci systému pro podporu výuky formou asistence při hodnocení odevzdaných úloh. Hlavním cílem je usnadnit proces hodnocení odevzdaných řešení a poskytnout lepší zpětnou vazbu studentům.

Teoretická část se věnuje popisu použitých technologií pro vývoj webových aplikací. Také je zde probrána problematika testování a hodnocení zdrojových kódů. Součástí je i průzkum již existujících řešeních na trhu.

Praktická část je orientována na popis jednotlivých modulů a analýzu výsledného řešení. Jednotlivé moduly na straně back-endu jsou popsány z vývojářského hlediska implementace. Front-end je popsán z hlediska uživatelského zážitku.

KLÍČOVÁ SLOVA

Spring Boot, Spring Webflux, React.js, hodnocení zdrojového kódu

TITLE

ANNOTATION

The diploma thesis focuses on the design and implementation of a system for teaching support in the form of assistance in the evaluation of submitted tasks. The main goal is to facilitate the process of evaluating submitted assignments and provide better feedback to students.

The theoretical part is dedicated to the description of the technologies used for the development of web applications. The issue of testing and evaluation of source codes is also discussed here. It also includes a survey of already existing solutions on the market.

The practical part is oriented towards the description of the individual modules and the analysis of the resulting solution. Individual modules on the back-end side are described from the developer's point of view of implementation. The front-end is described in terms of user experience.

KEYWORDS

Spring Boot, Spring Webflux, React.js, source code evaluation

Obsah

SEZNAM OBRÁZKŮ

SEZNAM ZDROJOVÝCH KÓDŮ

SEZNAM VÝPISŮ Z KONZOLE

SEZNAM TABULEK

SEZNAM ZKRATEK

AOP	Aspektově orientované programování
API	Application Programming Interface
AWS	Amazon Web Services
BPALP	Praktikum z programování a algoritmizace
BZAPR	Základy programování
CD	Continuous Delivery/ Continuous Deployment
CGLIB	Code Generation Library
CI	Continuous Integration
CPU	Centrální procesorová jednotka
CSS	Cascading Style Sheets
ČVUT	České vysoké učení technické
DI	Dependency injection
DI	Dependency Injection
EC	Ecmascript
EJB	Enterprise Java Bean
EJB2	Enterprise JavaBeans 2
FEI	Fakulta elektrotechniky a informatiky
GIT	Global Information Tracker
GPT	Generative Pre-trained Transformer
HMR	Hot Module Replacement
IOC	Inversion Of Control
JDBC	Java Database Connectivity
JDK	Java Development Kit
JPA	Java Persistence API
JS	JavaScript
JSON	Java Script Object Notation
JVM	Java Virtual Machine
MVC	Model-View-Controller
NOSQL	Not only SQL
NPM	Node Package Modelu
ODBC	Open Database Connectivity
ORM	Object Relational Mapping
POJO	Plain Old Java Object
QL	Query language
REST	REpresentational State Transfer
SPA	Single Page Application
SSO	Single Sign On
TS	TypeScript
URL	Uniform Resource Locator
UTF	Unicode Transformation Format
UUID	Universally Unique Identifier
VŠB	Vysoká škola báňská
XML	eXtensible Markup Language

ÚVOD

V dnešním moderním světě se oblast vývoje softwaru stala základním nástrojem pro řešení problémů. Nejen v akademickém prostředí se klade čím dál větší důraz na efektivní výuku programování.

Hodnocení zdrojových kódů je časově náročný a nákladný proces. Ve firemním prostředí je provozován na denní bázi pro zkvalitnění doručeného řešení. V případě akademického prostředí se však tento proces může stát o to náročnější vzhledem k faktu, že vyučující musí poskytnout mnohem detailnější zpětnou vazbu. S narůstajícím počtem odevzdaných řešení se z této úlohy stává rutinní činnost a hrozí pokles kvality zpětné vazby. Vytvoření vhodné aplikace s vylepšeným uživatelským rozhraním a automatickou kontrolou pro některé části řešení by tento proces mohlo vylepšit a poskytnou vyučujícím více času na ostatní činnosti.

Cílem diplomové práce je navrhnout a vyvinout software pro asistenci při hodnocení odevzdaných řešení. Vyučujícím by měl poskytnout rozhraní pro efektivnější zadávání hodnocení. Zároveň by měl být schopen poskytnout hodnocení na základě automatické kontroly podle předem definované šablony. Takto sepsané hodnocení by mělo být uchováno pro další statistickou analýzu, která poskytne vyučujícím přehled o aktuálním stavu znalostí studentů. Systém bude plně přístupný jako otevřené API pro tvorbu dalších aplikací.

První část diplomové práce je zaměřena na teorii ohledně testování a hodnocení zdrojových kódů. Poskytuje detailnější přehled o nástrojích, které jsou aktivně využívány v komerčním prostředí. Součástí je i průzkum trhu na řešení, které umožňují uživatelům otestovat své znalosti nejen v oblasti vývoje softwaru a poskytují jim zpětnou vazbu (komunitní i automatickou). Teoretická část je zakončena detailním popisem technologiím, jež budou při vývoji použity.

Druhá část práce se zabývá analýzou navrhovaného softwaru. Jsou zde rozepsány jednotlivé požadavky, které software musí splňovat a pro vizualizaci jsou uvedeny i diagramy. Front-end část je popsána více z pohledu uživatelského rozhraní, zatímco back-end moduly jejich implementací.

1 MOTIVACE K ŘEŠENÍ

V současné době musí být kontrola správnosti řešení provedena vyučujícím analýzou zdrojového kódu nebo regresním otestováním. Vzhledem k opakující se povaze se může tento proces stát pro vyučujícího náročný a hrozí, že některá řešení nebudou plnohodnotně zkontrolována

Tabulka 1 Počet odevzdaných řešení studentů ke kontrole za období 2022/2023.

Zkratka předmětu	Odevzdaná řešení	Zápočty	Opravné zápočty
BPALP	90 / cvičení	-	3 / semestr
BZAPR	-	42 / semestr	15 / semestr

Vyučující může tento proces zautomatizovat napsáním jednotkových, integračních nebo selenium testů. Tento proces ale není vhodný vzhledem ke skutečnosti, že studenti odevzdávají své práce prostřednictvím portálu STAG. Stále je zde nutná interakce v podobě stažení odevzdaných řešení a synchronizace s napsanými testy. Vzhledem k tomu, že neexistuje norma nebo univerzální řešení, může se učitel setkat s nekonzistencí odevzdaných souborů. Jen pro programovací jazyk Java je možné projekty odevzdat v následujících podobách:

- Projekt vytvořený přes nástroj Gradle;
- Projekt vytvořený přes nástroj Maven;
- Projekt vytvořený přes nástroj Ant;
- Zdrojové soubory s příponou java bez projektu.

Student se základními znalosti vývoje softwaru nemá od portálu STAG zpětnou vazbu, že odevzdané řešení je alespoň validní a je možné jej odeslat ke kontrole vyučujícímu.

1.1 Testování

Testování zdrojového kódu je jednou z hlavních součástí systému. V komerčním prostředí se proces testování liší podle použitého procesu řízení softwaru. „Waterfall“ rozdělil proces vývoje a testování softwaru do dvou rozdílných kroků. Vývojové oddělení odešle funkcionalitu po dokončení vývoje na otestování testovacímu oddělení. Tento tým sepíše nové testy a spustí je nad odevzdaným řešením. Také pečlivě zkontroluje, zda nová funkcionalita nezpůsobila nefunkčnost již existujících řešení. Tento krok však není dlouhodobě udržitelný s rostoucí

komplexitou projektu, která může vyústit až k odložení termínu vydání. Nehledě na fakt, že většina členů testovacího týmu je placena od počtu nalezených chyb a tím přivádějí vývojový tým do nepřátelské pozice. [27]

V případě agilního vývoje je hlavním cílem postupně doručit řešení zákazníkovi v odpovídající kvalitě. Tradiční testovací metodiky v tomto případě nezapadají do vývoje a jsou upraveny. Testování probíhá průběžně s vývojem. Cyklus agilního testování obsahuje pět fází:

- **Impact Assessment** – získání podnětů od zúčastněných stran a uživatelů.
- **Agile Testing Planning** – naplánování harmonogramu testování a očekávané výstupy.
- **Release Readiness** – přezkoumávání vyvinutých funkcionalit.
- **Daily Scrums** – každodenní standupy pro posouzení aktuálního stavu vývoje.
- **Test Agility Review** – zahrnuje týdenní schůzky se zúčastněnými stranami. [27]

V případě testovacího systému jsou uplatněny kombinace obou vývojových metodik. Testovací scénáře jsou implementovány před tím, než studenti započnou vývoj. Jejich kontrola však může být uplatněna až po odevzdání finálního řešení. Zde záleží na vyučujícím, zda projekt s připravenými testy poskytne studentům při zadání úlohy. Přezkoumání vyvinutých funkcionalit poté probíhá automaticky na základě testů nebo poskytnutím zpětné vazby vyučujícího. [28]

1.1.1 Blackbox testing

Praktika, při které testovací tým nemá žádný přehled ani znalosti o interním fungování a návrhu testovaného softwaru. Porovnávají se zde pouze výstupy na základě zadaných vstupů. Zaměřuje se na testování více z pohledu uživatelského zážitku po dokončení vývoje aplikace. Mezi použité praktiky patří:

- Průzkum systémových požadavků a specifikací.
- Vytvoření validních testovacích scénářů pro ověření správnosti systému.
- Vytvoření nevalidních testovacích scénářů pro ověření ke zjištění limitů systému a jeho schopnosti reagovat na ně.
- Vytvoření a nasazení testovacích scénářů.
- Porovnání výsledků testů s očekávanými. [29]

1.1.2 Whitebox testing

Praktika, při které testovací tým zkoumá vnitřní fungování a design aplikace. Zkoumá tedy průchod vstupů až k očekávanému výstupu. Zaměřuje se více na hledání chyb kódu a možných optimalizací řešení. Mezi použité praktiky patří:

- Identifikace vlastností pro testování.
- Identifikace všech možných scénářů a jejich zapsání do grafu.
- Získání informací o uživatelském flow, případů použití a technických informací.
- Napsání testů pro každý scénář.
- Testování a kontrola, zda bylo dosaženo požadovaných výsledků.

V případě testovacího systémů mohou být uplatněny oba způsoby testování. [29]

1.1.3 Jednotkové testy

Technika Whitebox testingu považována za první krok procesu testování při vývoji softwaru, kdy dochází k testování jednotlivých částí neboli modulů. Hlavním cílem je zjistit, zda tyto moduly fungují podle očekávání. Včasná identifikace pomáhá odhalit malé chyby hned na začátku vývoje. Primárním cílem je izolovat jednotlivé části kódu a testovat je odděleně. Testy jsou obvykle napsány jako pseudokód, který je následně implantován v příslušných programovacích jazycích. [30]

1.1.4 Integrační testování

Typ testování, při kterém jsou moduly testovány jako skupina. V typickém softwarovém projektu bývá každý modul naprogramován samostatným programátorem. Cílem tohoto testování je přijít na chyby, které by mohly nastat při vzájemné komunikaci těchto modulů před jejich nasazením. Na rozdíl od jednotkových testů se zaměřují na:

- Ověření funkčnosti spolupráce jednotlivých modulů.
- V průběhu vývoje mohou být požadavky klientem změněny, což může mít za cíl ovlivnění komunikace modulů.
- Komunikace s databází nebo externím hardwarovým systémem může způsobovat chybové stavy. [45]

1.1.5 Selenium testování

Open-source framework pro psaní automatických testů pro validaci webových aplikací napříč různými webovými prohlížeči a platformami. Je zcela zdarma ke komerčnímu použití. Dostupný je pro programovací jazyky java, C#, Python. Framework je sdružením několika testovacích nástrojů. Některé byly v rámci verze Selenium 2 sjednoceny do samostatného nástroje:

- **Selenium Integrated Development Environment (IDE)** – zjednodušený framework pro psaní automatických testů bez nutnosti znalosti vývoje softwaru. Je dostupný jako plugin pro webové prohlížeče Chrome a Firefox. Vzhledem k jeho omezenému použití je používán pouze jako prototyp pro tvoření testů.
- **Selenium Remote Control (RC)** – první verze nástroje pro psaní automatických testů webových služeb. K zápisu je využita syntaxe programovacího jazyka, podle výběru vývojáře.
- **WebDriver** – modernější způsob při provádění automatických testů. Provádí testovací scénáře přímou komunikací s webovým prohlížečem. K zápisu je využita syntaxe programovacího jazyka, podle výběru vývojáře.
- **Selenium Grid** – nástroj, který v kombinaci se Selenium RC dokáže provádět paralelní testy na více zařízení. [44]

1.2 CI/CD

Praktika softwarového vývoje, při které vývojáři průběžně commitují svou práci do centrálního repositáře (např. GitHub, GitLab, BitBucket), aby se zajistila maximální kvalita nasazovaného softwaru. Každá změna v kódu je automaticky otestována tak, aby se udržela maximální integrita kódu. Klíčovým cílem CI je najít a včas upozornit vývojáře na chyby, které tak mohou být odhaleny v kratším časovém horizontu. Je prokázáno, že využívání CI/CD setří přibližně 25–30 % času vývoje. [14]

1.2.1 Jak CI/CD funguje?

Vývojář pošle změny v kódu na centrální repositář. Tato změna automaticky spustí sadu úkolů, které jsou definovány v automatickém nástroji. Mezi takové úkoly patří sestavení balíčku a spouštění testů. Pokud některý z úkolů skončí chybou, je vývojář na tuto skutečnost neprodleně upozorněn. Některé úlohy mohou být označeny jako nepovinné a jejich chybou není

proces CI ukončen. V případě úspěšného průchodu všech úkolů je řešení připraveno k nasazení do některého prostředí nebo do jiné vývojové větve. [14]

Systém může být nastaven tak, aby konzistentní kód automaticky nasadil do prostředí pro otestování nebo využívání zákazníky. Jedná se o další úlohu po otestování integrity kódu. Řeší problém rutinních operací, které musely být dosud manuálně prováděny členy týmu a zpomalovaly tím proces nasazení softwaru. [15]

1.2.2 CircleCI

Cloudové řešení pro založení CI/CD procesů s minimální konfigurací. Pro snadnou konfiguraci jsou dostupné jsou programovací jazyky jako C++, NET, PHP, Python, JavaScript, a Ruby. Řešení je možné nasadit ve vlastní infrastruktuře (CircleCI Server) nebo využít jako řešení v cloudu (CircleCI cloud). Kontejnerizace je možná pod operačními systémy Linux, Windows a Mac OS. Nevyžaduje žádnou dodatečnou instalaci a zákazníkům jsou k dispozici tzv. „Orbs“. Jedná se o kusy předem definovaného kódu urychlující proces prvotní konfigurace. [16]

Mezi základní vlastnosti patří:

- Konfigurace procesů pomocí YAML souboru umístěného v kořenovém adresáři projektu.
- Jednoduché nastavení.
- Poskytuje REST API.

1.2.3 TeamCity

Nástroj pro CI/CD a správu sestavených řešení vydaný společností JetBrains 2. listopadu 2006. TeamCity je zdarma pro použití, pokud uživatel nepřekročí 100 sestavených řešení měsíčně a vystačí si se třemi „Build Agent“. Cena jedné serverové komerční licence se odvíjí od požadovaných funkcionalit a počtu „Build Agent“. Mezi další vlastnosti patří: [17]

- Jednoduchá instalace.
- Historie sestavených řešení.
- Tvorba infrastruktury.
- Sledování kvality kódu.
- Žádné dodatečné pluginy.
- Jednoduchá integrace s již existujícími IDE od společnosti JetBrains.

- Multiplatformní řešení.

1.2.4 Jenkins

Open-source řešení napsané v jazyce Java a běžící pod Apache Tomcat jako „servlet“ kontejner. Je populární hlavně díky široké komunitě fanoušků. První verze, tehdy ještě pod názvem „Hudson“, byla napsána pracovníkem společnosti „Sun Mikrosystem“ Kohsuke Kawaguchi v létě 2004. První publikace nástroje proběhla o rok později. Po odkoupení Sun Mikrosystem společností Oracle byl komunitou schválen návrh na vytvoření projektu Jenkins. V únoru 2011 měla společnost Oracle v úmyslu pokračovat na vývoji Hudson, a proto byl vývoj na Jenkins rozvětven. Přestože byly projekty vyvíjeny odděleně, získal Jenkins vetší popularitu a vývoj projektu Hudson byl ukončen. Mezi hlavní výhody patří:

- Jednoduchá instalace a konfigurace.
- Open-source řešení.
- Široká škála dostupných pluginů.
- Jednoduchá distribuce napříč zařízeními. [18]

1.3 Existující řešení

Na trhu se již v současné době nachází řešení pro vyhodnocení zdrojových kódů. Některá řešení jsou specifická pouze pro určitý programovací, skriptovací nebo značkovací jazyk. Některé výukové portály zaměřené na vývoj softwaru doplňují vzdělávací videa a materiály i o mezikroky, ve kterých musí studenti splnit zadání dopsáním zdrojového kódu do editoru nebo přímo nahráním vypracovaného projektu. Příkladem takového portálu je Treehouse.

1.3.1 CSS Battle

Webová aplikace pro pořádání turnajů mezi uživateli, kteří chtějí otestovat své CSS znalosti plněním úloh. Tyto úlohy jsou nazývány „battles“ a soustředí se hlavně na kreslení různých vzorů nebo tvorbu webového rozvržení pouze za pomocí CSS. Jedna z hlavních výhod, kterou portál nabízí, je možnost sdílet mezi vývojáři již hotová řešení a poskytovat tak novým hráčům přehled, jak jejich řešení obstálo v porovnání s ostatními. Jednotlivá řešení je možné ohodnotit a vést k nim diskusi. Aplikace není nijak omezená zkušenostmi a zahrát si mohou zkušení vývojáři i úplní začátečníci. [1]

Projekt vznikl jako výzva na reakci přátel dvou CSS nadšenců Kushagra Agarwala a Kushagra Goura, že psaní CSS začíná být snadné a nudné. První prototyp hry byl založen na principu,

kde hráči měli replikovat zobrazený obrázek pomocí CSS s napsáním co nejmenšího množství znaků. Sami autoři tomuto konceptu propadli a dali si za cíl vydat hotový produkt během jednoho měsíce. Aplikaci později ocenili známí experti v oblasti CSS jako Lea Verou a v dnešní době má více než 19 tisíc aktivních hráčů. [2]

Jádro aplikace obsahuje React knihovnu poháněnou technologií Vercel. Pro backendovou část byl použit Firestore jako perzistentní vrstva, Cloud functions pro spuštění kódu a Cloud storage pro uchovávání dat uživatelů. Blog je hostován na Evelenty. Celá webová stránka je koncipována jako monorepozitář, který se skládá ze hry a blogu. [2]

1.3.2 HackerRank

Webový portál, na kterém programátoři mohou otestovat své znalosti v oblastech počítačové vědy, jako je algoritmizace, kritické myšlení, umělá inteligence nebo návrhové vzory. [3] Platformu také využívá více než 3 000 společností po celém světě k otestování znalostí nově nabíraných pracovníků. První zmínky o HackerRank pochází z roku 2009, kdy absolventi počítačových věd Vivek a Hari začali pracovat v Amazonu a IBM v Barceloně. Oba měli na starosti nábor nových uchazečů, kdy si začali všímat faktu, že společnosti nabírají nekvalifikované pracovníky, zatímco jejich kvalifikovaní přátelé nejsou přijati z důvodu chybějících praktických zkušeností nebo vysokoškolského diplomu. [4]

V tom viděli zásadní problém, ale i příležitost pro vytvoření nového systému pro hodnocení technických znalostí uchazečů. Oba se rozhodli dát výpověď a vytvořili InterviewStreet, webovou stránku, která spojuje kantory a studenty. Portál zaznamenal obrovský úspěch a během několika týdnů se a zaregistrovalo více než 1 000 aktivních studentů. Skromné začátky jim vynesly jeden šek v hodnotě 5 dolarů a 68 centů. I přes pozitivní reakce uživatelů se nedařilo portál prosadit v oblasti náborových procesů a v září 2010 hrozil oběma vývojářům bankrot. Přihlásili tedy svůj produkt do seed kampaně Y Combinator v Palo Altu. Obdrželi dotaci v hodnotě 200 000 dolarů na zdokonalení jejich startupu. O rok později navýšili příjmy na tři miliony dolarů a jejich produkt se začal uplatňovat při náborových procesech ve společnostech jako Airbnb, Box a Amazon. V září 2012 získal produkt své současné jméno. [4]

1.3.3 Codewars

Platforma, kde mohou uživatelé otestovat své programovací znalosti plněním úloh „kata“ v programovacích jazycích C, C++, C#, Dart, Go, Haskell, Java, JavaScript, Kotlin, Lua, PHP, Python, R, Ruby, Rust, SQL, Scala, Solidity a Swift. Na rozdíl, od již zmíněných systémů se Codewars zaměřují na uživatelský zážitek ve formě herní výzvy. Uživatelé za plnění úloh

získávají ocenění a stoupají v žebříčcích. Stejně, jako při klasické videohře, se jim za úspěšné plnění úkolů zvyšuje herní level. Úkoly jsou zaměřeny na začátečníky i zkušené uživatele a každý může platformu obohatit o nové úkoly. [6]

1.3.4 Progtest

Systém pro odevzdání zdrojových kódu studentů ČVUT pro předměty zaměřené na programování (Programování a algoritmizace 1, Operační systémy 1…). Provádí automatickou kontrolu plagiátorství a testování odevzdaných řešení. První verze byla vyvinuta za pouhý jeden týden Ing. Ladislavem Vágnerem, Ph.D. tehdy ještě na Fakultě elektrotechniky pro usnadnění časově náročné kontroly odevzdaných studentských prací. V současné době je Progtest využíván primárně Fakultou informačních technologií. Je dostupný i pro další univerzity a akademie v České republice, například VŠB-TUO nebo Akademie Silicon Hill. Jeden ročník byl využíván i cvičícími na výměnném pobytu na mexické univerzitě Tecnológico de Monterrey. [5]

2 ANALÝZA

Softwarová analýza má za cíl odpovědět na základní otázky a technologie před započetím vývoje softwaru. Minimalizuje také vznik rizik a včasně identifikuje specifické potřeby, aby tým mohl učinit správná rozhodnutí. Při analýze je nutná komunikace se zadavatelem tak, aby software splňoval jeho představy. [47]

2.1 Funkční a nefunkční požadavky

Definování požadavků je jedna z nejdůležitějších částí při analýze projektu. Studie prokazují, že špatná definice požadavků může zvýšit dobu vývoje softwaru až o 60 %. Funkční požadavky jsou vlastnosti produktu, které musí vývojáři implementovat tak, aby uživatelům byly dostupné hlavní vlastnosti produktu. Definují tedy, co by produkt měl umět, zatímco nefunkční požadavky spíše definují, jak by měl produkt tyto vlastnosti umožňovat. [46]

2.1.1 Funkční požadavky

- Systém musí umožňovat autentifikaci uživatelů.
- Systém musí umožňovat registraci uživatelů.
- Systém musí umožňovat vytvoření nového kurzu.
- Systém musí umožňovat kurz rozdělit do jednotlivých semestrů.
- Systém musí umožňovat se uživatelům zapsat do konkrétního kurzu.
- Systém musí umožňovat vyučujícím správu uživatelů.
- Systém musí umožňovat vyučujícím založení nové úlohy.
- Systém musí umožňovat opatřit každou úlohu sadou automatických testů.
- Systém musí umožňovat omezit vybrané typy úloh pouze pro konkrétní uživatele.
- Systém musí revizorům umožnit zadat hodnocení k dané úloze.

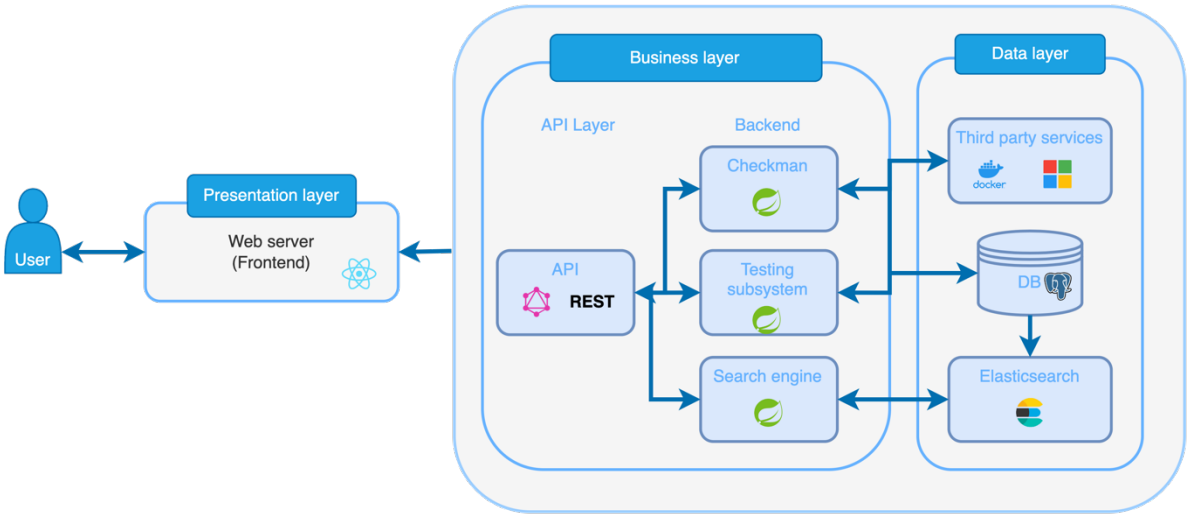
2.1.2 Nefunkční požadavky

- Systém musí umožňovat autentifikaci přes Microsoft SSO.
- Systém musí umožnit registraci uživatelů pouze s platnou univerzitní doménou.
- Systém musí autorizovat uživatele podle typu požadavku.
- Systém musí autorizace rozlišovat podle dostupných rolí.

- Systém musí zamezit spuštění odevzdaného řešení na hostitelském zařízení.
- Systém musí využívat výhod kontejnerizace.
- Systém musí zamezit vyčerpání zdrojů hostitelského zařízení.
- Systém musí umožnit zadat hodnocení formou zpětných vazeb.
- Systém musí poskytovat základní statistiku pro daný semestr.
- Systém musí umožňovat našeptávání zpětných vazeb z minulých hodnocení.
- Systém musí obsahovat přívětivé uživatelské rozhraní.

2.2 Diagram infrastruktury

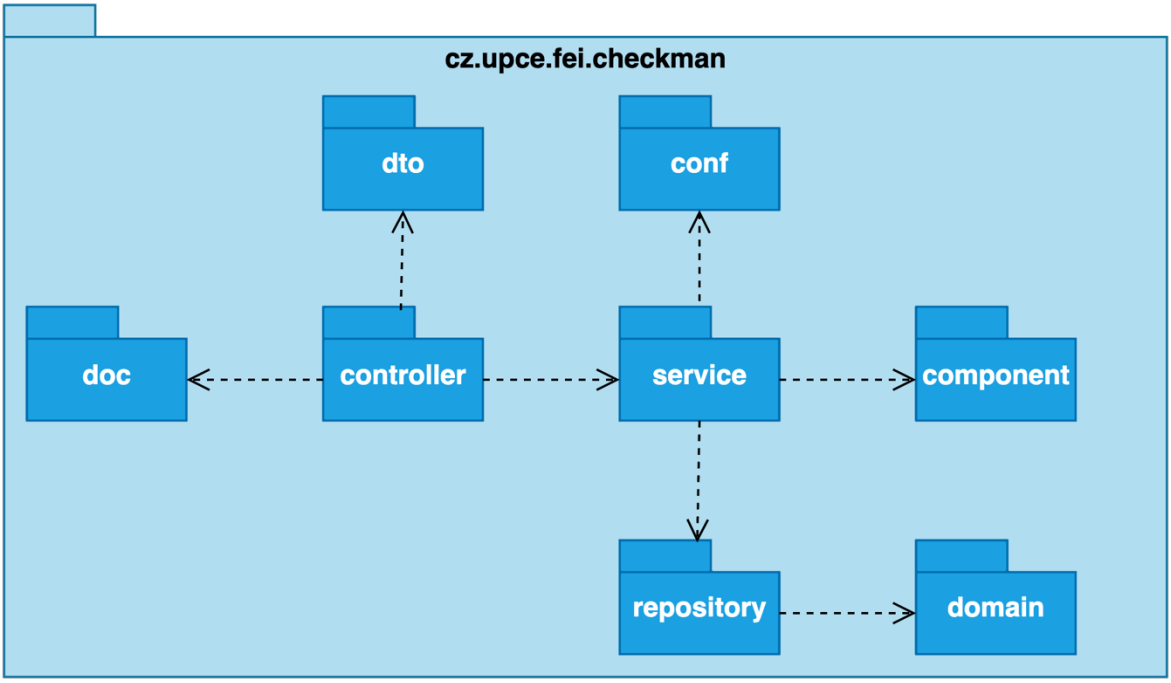
Popisuje uspořádání a fungování jednotlivých modulů v ekosystému výsledné aplikace. Jednotlivé části jsou opatřeny ikonkou technologie nebo služby, které byly použity při jejich vývoji viz obrázek 1.



Obrázek 1 Diagram jednotlivých služeb aplikace

2.3 Package diagram

Popisuje uspořádání balíčků pro backendové moduly.

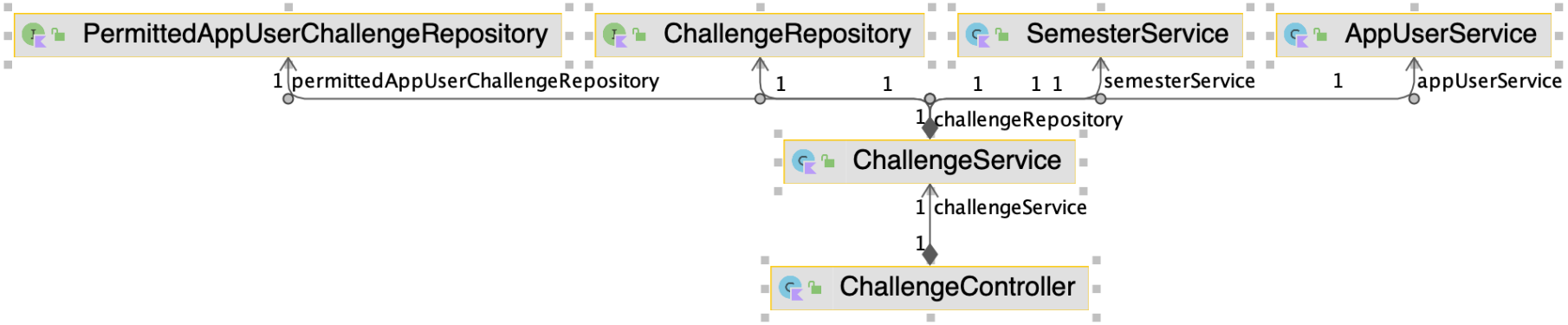


Obrázek 2 Package diagram na straně backendu

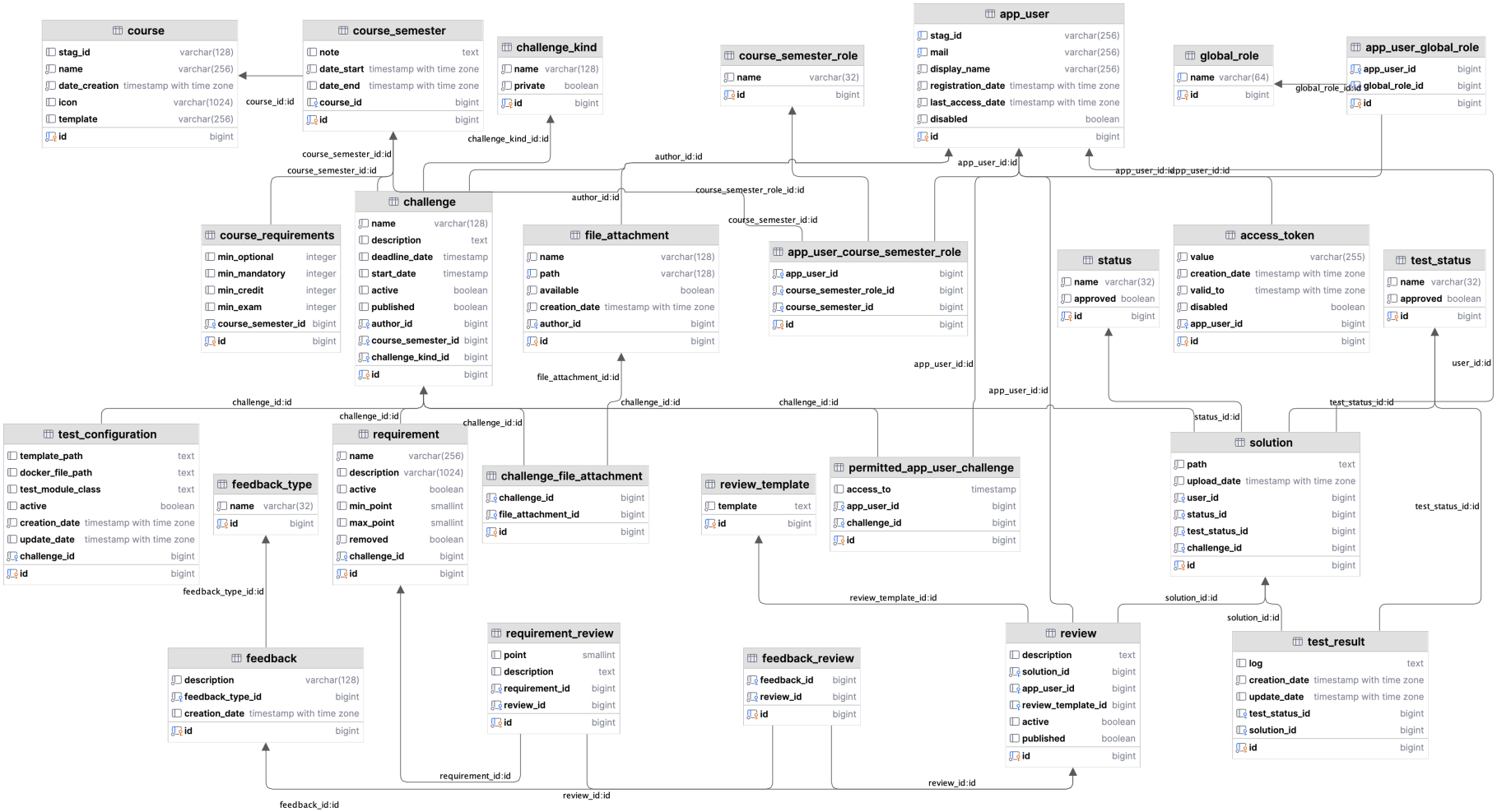
2.4 Class diagram

Někdy také nazýván jako diagram implementace, obsahuje všechny třídy, které budou dostupné ve finální aplikaci. Diagram je platformově závislý a specifický pro určitý programovací jazyk. Není v něm zapsána diakritika a jedná se o návod pro programátora, jak má finální třídy realizovat. [48]

V rámci implementovaného systému je využita běžná struktura Spring Boot aplikací. Diagram tříd (viz obrázek 2) ukazuje závislosti mezi jednotlivými třídami pro manipulaci s jednou entitou. Pro databázový pohled je využit ER diagram viz obrázek 3.



Obrázek 3 Class diagram ukazující závislost od GraphQL vstupního bodu pro operace nad zadáním



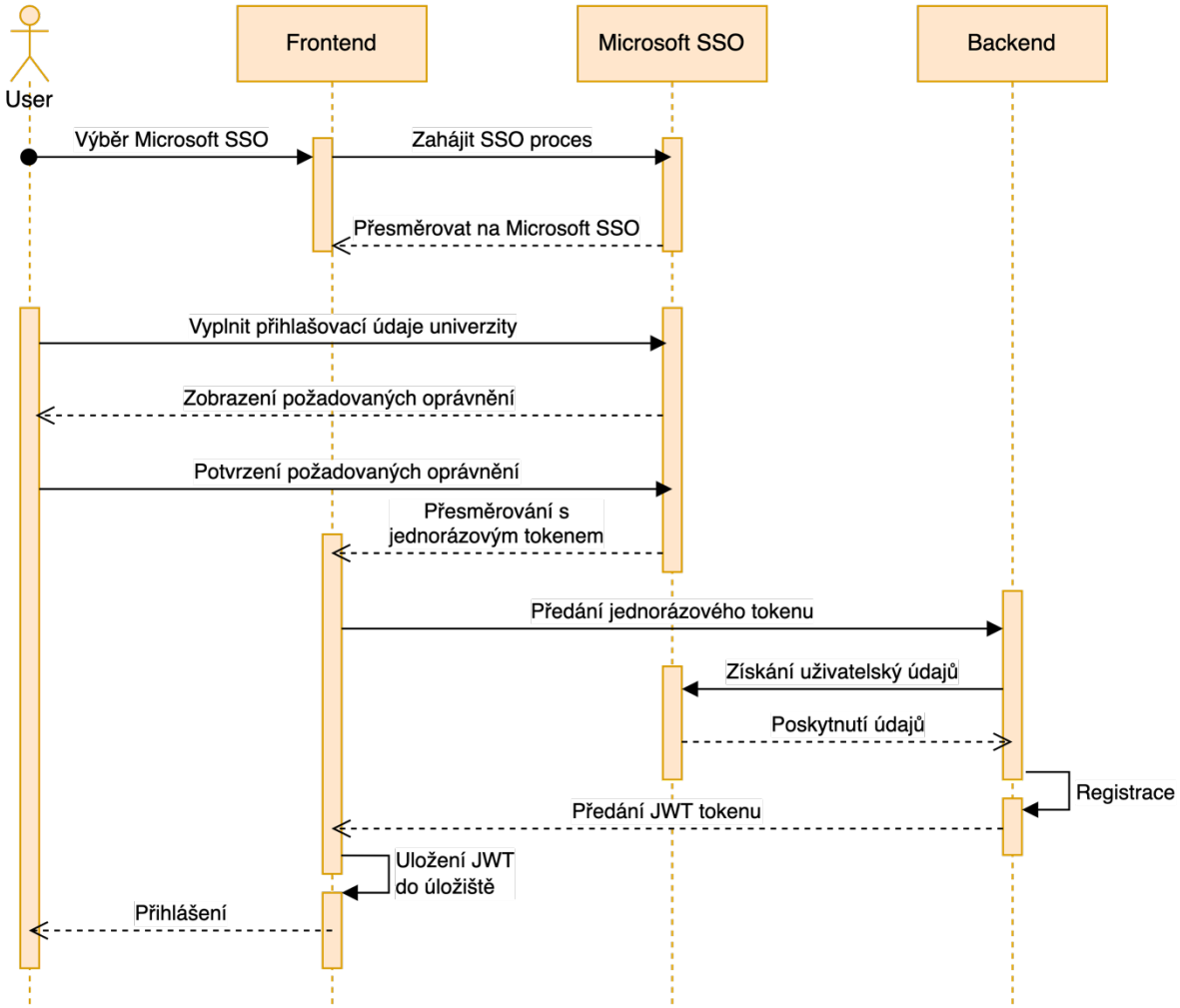
Obrázek 4 ER diagram

2.5 Sekvenční diagramy

Sekvenční diagramy popisují průchod požadavku od uživatelů v rámci jednotlivých systémů. Následující diagramy popisují průchod nejdůležitějšími částmi ekosystému.

2.5.1 Autentifikace uživatele

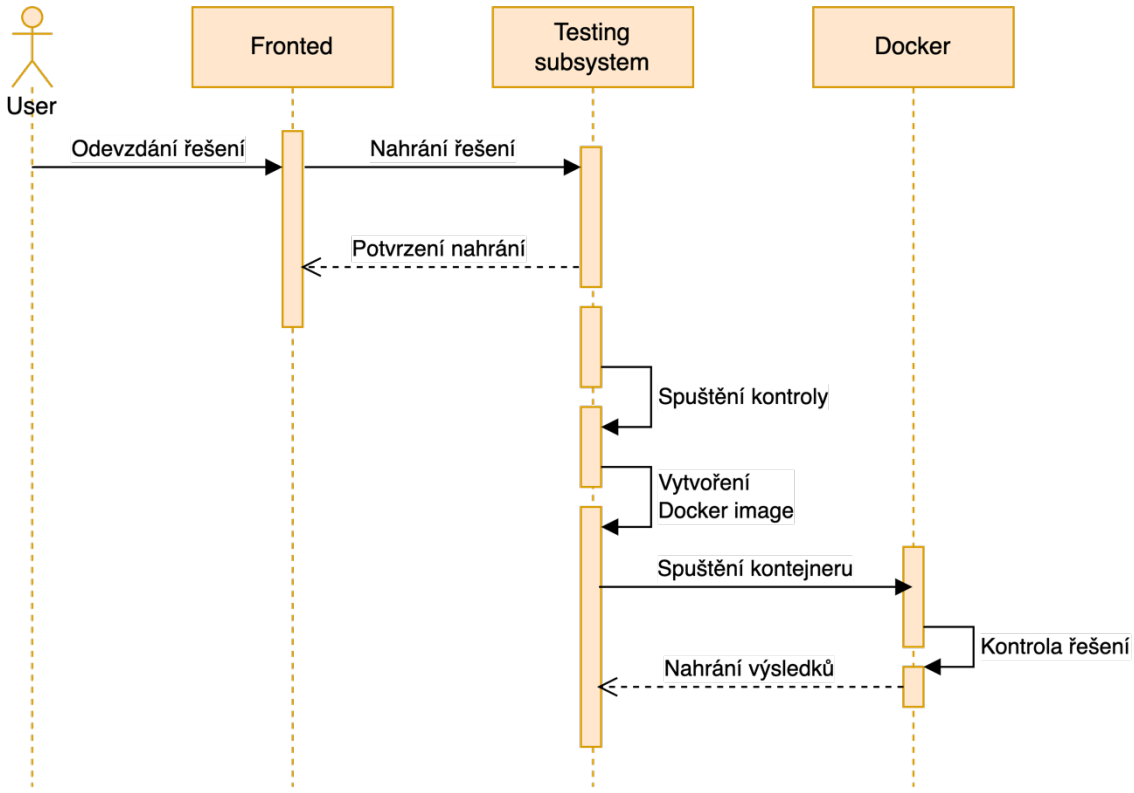
Popisuje registraci formou prvotního přihlášení přes službu Microsoft SSO.



Obrázek 5 Sekvenční diagram autentifikace uživatele

2.5.2 Automatická kontrola odevzdaného řešení

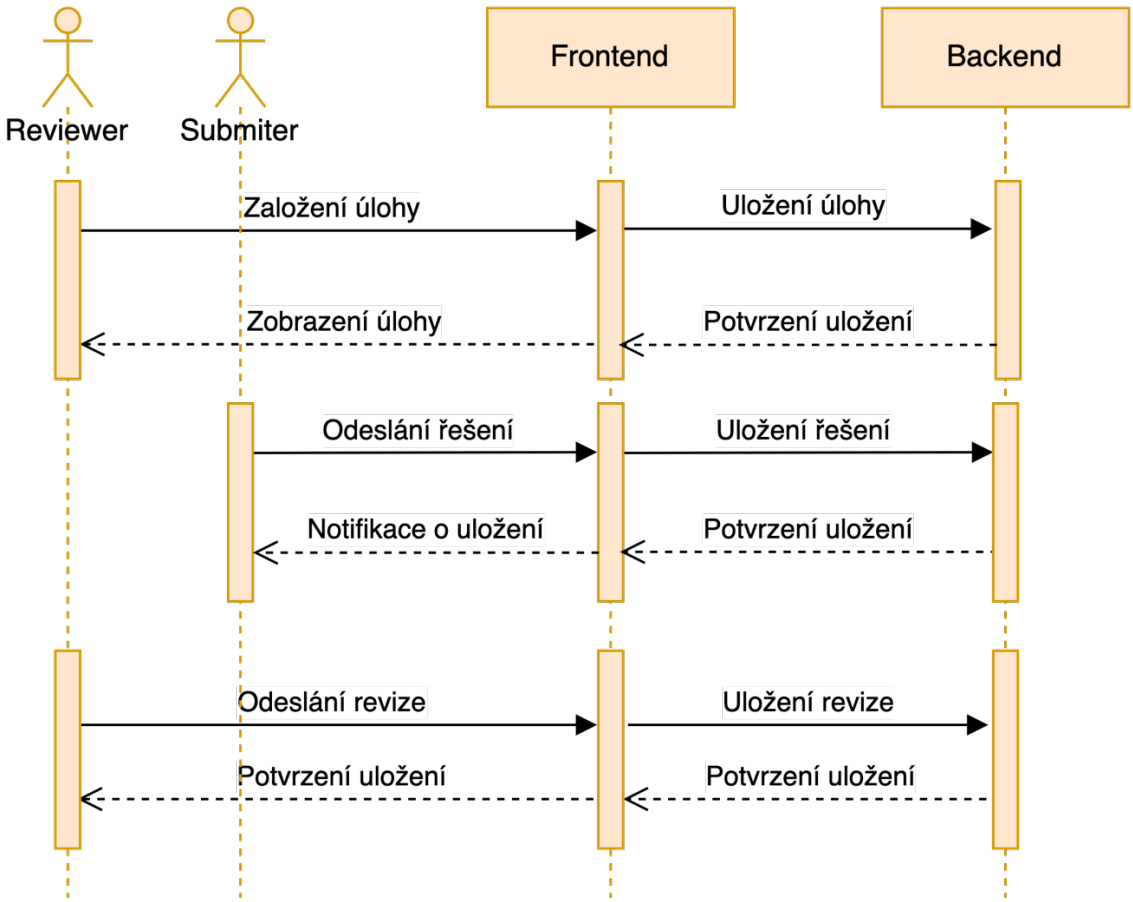
Popisuje průchod otestování řešení studenta.



Obrázek 6 Sekvenční diagram automatické kontroly odevzdaného řešení

2.5.3 Kontrola řešení

Popisuje prvotní založení úlohy vyučujícím a následné odevzdání studentem. Průchod je zakončen revizí odevzdané úlohy.



Obrázek 7 Sekvenční diagram vytvoření, nahrání a kontroly řešení k úloze

3 POUŽITÉ TECHNOLOGIE

Kapitola popisuje jednotlivé technologie a knihovny použité při vývoji všech subsystémů. Při výběru bylo dbáno na následující kritéria:

- Kvalitně zpracovaná dokumentace;
- Početná komunita;
- Všeobecná známost;
- Periodické vydávání nových verzí;
- Aktivita ze strany vývojářů na GitHub.

3.1 Kotlin

Staticky typovaný, objektově orientovaný programovací jazyk, vyvinutý společností JetBrains, který funguje pod JVM. Původně byl vyvinut pro vylepšení programovacího jazyka Java, ale je často využíván společně s ním. Jedná se o primární jazyk pro vývoj mobilních aplikací na operační systém Android, ale je možné jej použít k vývoji i dalších druhů aplikací:

- **Server-side development** – pro vývoj backend aplikací, které původně jako primární jazyk využívaly Javu.
- **Full-stack web development** – kód napsaný v jazyce kotlin může být přeložen do programovacího jazyka javascript pro vývoj webových aplikací. Díky tomu je možné použít stejný kód pro backend i frontend.
- **Multiplatform mobile development** – podpora platforem jako Apple iOS, Apple watchOS nebo Linux.
- **Data science** – často bývá také využíván pro „data science“ při procesech, jako tvorba „data pipelines“ a nasazování neuronových sítí do produkce. [19]

3.1.1 Historie

Jazyk byl vyvinut společností JetBrains hlavním sídlem v České republice. Oficiální uvedení proběhlo v roce 2016. Na konferenci Google I/O v roce 2017 byl Kotlin oznámen jako první oficiální programovací jazyk pro platformu Android. V roce 2019 již byl vývojářům doporučen jako preferovaný pro vývoj. [19]

3.1.2 Vlastnosti

Mezi hlavní výhody programovacího jazyka kotlin patří:

- **Interoperabilita** – stejně, jako Java, je kotlin překládán do byte kódu, proto může být využit při vývoji již existujících projektů. Díky tomu, že sdílí nástroje s Javou, je možné snadno migrovat existující aplikaci do Kotlinu. Překlad je možný také do jazyka JavaScript nebo LLVM.
- **Bezpečnost** – Kotlin byl navržen tak, aby předcházel běžným chybám při vývoji softwaru. Obsahuje „null safety" – vlastnost, která eliminuje chyby při manipulaci s null referencí.
- **Čirost** – Kotlin eliminuje některé redundantní části kódu, které se běžně vyskytují v syntaxi Javy. Díky stručnější syntaxi mohou vývojáři psát programy s menším množstvím šablonovitého kódu, což významně zvyšuje jejich produktivitu.
- **Podpora nástrojů** – pro Kotlin jsou k dispozici nástroje jako Android studio od společnosti JetBrains, Android KTX a Android SDK. [19]

3.1.3 Kotlin vs. Java

Oba programovací jazyky jsou ve svém základu staticky typované. Kotlin je považován za nástupce jazyka Java, i když není kompatibilní s jeho syntaxí, dokáže spolupracovat s jeho kódem a knihovnami. V historii byla většina Android je dokumentace napsána v Javě, proto vývojáři mohli preferovat Javu než migrovat na Kotlin. Mezi minimalizace redundantního kódu patří:

- Zjednodušený funkcionální kód.
- Řádky není nutné zakončovat středníky, ale vývojáři je mohou i tak nadále používat, pokud jsou zvyklí na tuto syntaxi.
- Odstranění některých klíčových slovíček v opakující se syntaxi. Například absence klíčového slovíčka „new" při vytváření nové instance.
- Konstanty již nyní nemusí obsahovat klíčové slovíčko „final" před deklarací, ale je možné je přímo definovat pomocí klíčového slovíčka „val".
- Predikce datového typu na základě inicializační hodnoty a představení klíčového slovíčka „var". Datový typ je stále možné uvést za deklarací proměnné. Tuto vlastnost již obsahuje Java 10.

- Třída může deklarovat primární konstruktor rovnou ve své signatuře. Parametry konstruktoru poté zároveň slouží i jako atributy dané třídy.
- Datové třídy jsou jednoduchým zápisem POJO. Vývojář může všechny atributy deklarovat v hlavičce třídy a osvobozuje vývojáře od implementace getterů, setterů, „equals“ a „hashcode“ metod. [20] [19]

3.2 Spring framework

Framework pro vývoj různých druhů aplikací využívající Java platformu. Snaží se usnadnit složitou situaci ve vývoji podnikových aplikací tím, že nabízí technologie:

- Aspektově orientované programování (AOP)
- Dependency Injection (DI)
- Plain Old Java Object (POJO)
- Hibernate
- WEB MVC [21]

Mezi hlavní výhody patří:

- **Loose Coupling** – závislosti a instance jednotlivých tříd jsou snadno řízeny pomocí DI.
- **Easy and Simple to Test** – celou aplikaci je možné snadno otestovat díky DI, aniž by bylo nutné její nasazení na existující server.
- **Non-invasive** – využíváním techniky POJO je Spring snadný na implementaci, aniž by nutil vývojáře k dědičnosti z určitých tříd nebo implementaci jakéhokoliv rozhraní.
- **Lightweight IoC** – IoC je ve Springu mnohem šetrnější oproti alternativnímu řešení EJB. To umožňuje nasazení aplikace do prostředí s omezenou operační pamětí nebo CPU.
- **Constant Transaction Management** – rozhraní zjednodušující databázové transakce pro aplikace využívající jednu databázi. [21]

3.3 Spring Core

Pro backendová řešení byly využity následující funkcionality ze základního projektu Spring core.

3.3.1 Dependency Injection (DI)

Koncept Springu můžeme definovat jako „Inversion of Control (IoC)“. DI umožňuje vytvářet instance mimo třídu a poskytnout je dalším třídám bez nutnosti volání konstruktoru v kódu. Toto předávání můžeme definovat přes parametry konstruktoru dané třídy nebo pomocí atributů. [21]

Závislost tedy můžeme definovat jako vztah mezi dvěma třídami. V následujícím příkladu třída ChallengeController definuje v konstruktoru závislost na třídě ChallengeService:

Zdrojový kód 1 Ukázka DI z aplikace

```
@Controller
@Validated
class ChallengeController(
    private val challengeService: ChallengeService
) {
    ...
}
```

Spring IoC vyřeší tuto závislost pomocí DI. Instance třídy B bude vložena do konstruktoru třída A při vytváření instance. Takto napsaný kód je jednodušší na testování a znovu používání. Při vytváření složitějších Java aplikací by mělo být zajištěno, aby jednotlivé třídy byly na sobě, pokud možno, co nejméně závislé. DI Tuto skutečnost umožňuje, ale zároveň poskytuje třídám vzájemnou závislost, aniž by došlo k porušení konceptu. [21]

3.3.2 Aspektově orientované programování (AOP)

Je to jedna z klíčových vlastností Spring frameworku, která doplňuje objektově orientovaný přístup o modularitu. AOP Rozděluje logiku programu do samostatných částí zvaných koncerny. Tyto koncerny jsou funkce („Advice“) přidávající určitou logiku do kódu, aniž by došlo k jeho změně nebo porušení obchodní logiky.

Hlavní využití v aplikacích:

- Logování;
- Cashování;
- Řízení databázových transakcí;
- Authentifikace. [22]

Vývojář specifikuje, ke které části kódu chce přidat dodatečné chování, pomocí speciálního výrazu daného programovacího jazyka („Pointcuts“). Spring AOP využívá k aplikaci tohoto chování „JDK dynamic proxies“, která je dostupná v JDK nebo open-source „CGLIB“, jež je dostupná v rámci balíčku „spring-core“. Mezi základní bloky při programování AOP patří:

- **JoinPoint** – část kódu, ke které bude přidáno nové chování.
- **Advice** – nově implementované chování, také zvané koncern. Toto chování bývá většinou reprezentováno metodou, která se aktivuje při dosažení „JointPoint“. Toto chování může být aktivováno před vykonáním, po vykonání nebo v průběhu vykonávání metody podle typu. Ve Spring frameworku jsou k dispozici následující anotace.
    - **@After** – chování je aktivováno po dokončení kódu metody.
    - **@Before** – chování je aktivováno před vykonáním metody.
    - **@Around** – chování je aktivováno v průběhu vykonání metody.
    - **@AfterReturning** – chování je aktivováno pouze při úspěšném dokončení kódu metody.
    - **@AfterThrowing** – chování je aktivováno v případě vyvolání konkrétní výjimky v průběhu vykonání metody.
- **Pointcut** – výraz programovacího jazyka, který definuje, v jakých „JointPoint“ se má „Advice“ aktivovat. Může se jednat například o jakoukoliv anotaci, která je součástí programovacího jazyka Java, Kotlin, nebo která byla vytvořena samotným vývojářem.
- **Aspect** – třída, ve které jsou definovány „PointCut“ a „Advice“. Takovou třídu můžeme označit anotací @Aspect. [22]

3.3.3 Data Access Framework

Uchovávání jakýchkoliv typů dat je stavebním pilířem všech dynamických webových aplikací. Komunikace s databází je však jedna z komplexnějších částí aplikace. Vývojáři by implementace vlastního řešení mohla zabrat delší dobu. Spring usnadňuje komunikaci s databází tím, že poskytuje podporu pro rozšířené frameworky, jako je „JDBC“, Hibernate a „JPA“. [21]

3.3.4 Transaction Management Framework

Transakcemi rozumíme v Javě sérii událostí, které musí být úspěšně dokončeny. Pokud jedna nebo více událostí selže, všechny dosud provedené musí být stornovány a stav aplikace musí

být obnoven do bodu před jejich započetím. Musí zde být zajištěno, že integrita aplikace nikdy nebude kompromitována. Transakce mohou být rozděleny do dvou druhů podle počtu zdrojů, se kterými manipulují:

- **Resource Local Transactions** – transakce manipuluje pouze s 1 zdrojem, například sekvence SQL dotazů vykonávané nad jednou databází.
- **Global Transactions** – transakce manipulující s více než 1 zdrojem, například sekvence SQL dotazů nad více databázemi. Nebo kombinace operací nad databází a frontou zpráv. [23]

3.3.5 JDBC

Java API vydané v rámci JDK 1.1 v roce 1997. Slouží ke správě připojení k databázi, spouštění SQL dotazů a správě příchozích dat. Původně bylo koncipováno pouze pro připojení na straně klienta. Ke změně došlo s vydáním verze JDBC 2.0, která obsahovala volitelný balíček podporující připojení i na straně serveru. Předchůdcem tohoto řešení bylo ODBC, ze kterého se JDBC částečně inspirovalo. Jednoduchou manipulaci s databází můžeme definovat podle následujících kroků:

1. Vložení JDBC knihovny a ovladače do „classpath“.
2. Využití JDBC k navázání spojení k dostupné instanci databáze.
3. Spuštění SQL dotazů a získání odpovědi.
4. Uzavření databázového připojení. [25]

3.3.6 Hibernate

Open source ORM nástroj poskytující framework k mapování objektově orientovaných tříd do tabulek relačních databází. Vývoj začal v roce 2001 Gavienm Kingem a jeho kolegy ze společnosti „Cirrus Technologies“ jako alternativa k „EJB2“. Hlavním cílem vývoje bylo poskytnutí lepší možnosti perzistence. ORM je založeno na principu kontejnerizaci objektu za využití abstrakce. Ta umožňuje adresovat, vytvářet a manipulovat s objekty, aniž by byla nutná znalost, jak souvisí se zdroji dat. [24]

3.3.7 JPA

Jakarta Persistence API (dříve označována jako Java Persistence API) se zabývá specifikací perzistence. Svým způsobem není nástrojem nebo frameworkem, spíše definuje pravidla, kterých by se vývojáři měli držet. Původně byl vytvořen pouze pro použití s relačními

databázemi, některé jeho implementace však byly rozšířeny pro použití s NoSQL databázemi. Mezi populární frameworky, které poskytují podporu JPA s NoSQL databázemi, je například „EclipseLink“ a „Hibernate“. Hlavním cílem JPA je osvobodit vývojáře od nutnosti myslet relačně a definovat jednotlivé objekty a vazby pomocí tříd. [26]

JPA ve své specifikaci poskytuje „metadata anotace“, které vývojář může použít k mapování objektů a databází. K mapování je možné použít i XML soubory. Každá implementace poskytuje i vlastní řešení pro správu těchto anotací. Specifikace také poskytuje nástroje, jako „PersistanceManager“ nebo „EntityManager“, které komunikují s JPA systémem:

- **@Entity** – perzistentní objekty jsou občas nazývány entity. Třída označená touto anotací informuje JPA, že má být součástí perzistence.
- **@Table** – třída označena touto anotací reprezentuje databázovou tabulku.
- **@PrimaryKey** – atribut označený touto anotací reprezentuje primární klíč záznamu. Měl by být použit pokaždé, když je objekt uložen v tabulce.
- **@OneToMany** – objekt je ve vztahu 1: N s jiným objektem.
- **@ManyToOne** – objekt je ve vztahu N: 1 s jiným objektem.
- **@ManyToMany** – objekt je ve vztahu M: N s jiným objektem.
- **@OneToOne** – objekt je ve vztahu 1: 1 s jiným objektem. [26]

3.4 Spring Boot

Projekt, který staví na Spring frameworku. Poskytuje snadnější a rychlejší prvotní nastavení projektu pomocí RAD jen s minimální konfigurací. Obsahuje i vložený server pro běh Spring aplikací a odstraňuje nutnost konfigurace bean pomocí XML souborů. Naproti tomu zvětšuje velikost aplikace, protože obsahuje závislosti, které vývojář nemusí využít. Mezi hlavní výhody patří:

- Tvorba „stand-alone“ aplikací, které mohou být spuštěny bez nasazení war souborů do Tomcat služby pomocí jednoho ze zabudovaných http serverů (Tomcat, Jetty).
- K dispozici jsou různé pluginy i pro build služby jako Maven nebo Gradle.
- Poskytuje zabudované funkcionality jako statistiky, sledování stavu serveru a externí konfigurace.
- Minimalizuje psaní duplicitního kódu, XML konfigurací a anotací. [39]

3.5 Spring WebFlux

Framework pro vytváření neblokujících aplikací na vrstvě HTTP za pomoci funkcionálního programování. Je poskytován jako alternativa ke Spring MVC od verze Spring 5. Podporován je kontejnery Tomcat, Jetty a Netty. Poslední ze zmiňovaných je nejčastěji používán pro implementaci. Díky tomu je možné WebFlux snadno napojit na již existující řešení. Využívá funkcionality projektu „Project Reactor" a přidává nové jako:

- **Non-blocking threads** – vlákna mohou dokončit hlavní úlohu, aniž by musela čekat na dokončení předchozí úlohy.
- **Reactive Stream API** – asynchronní zpracování streamů s neblokující „backpressure".
- **Asynchronous data processing** – data jsou zpracována na pozadí a uživatel může pokračovat v běžném využívání aplikace bez přerušení. [38]

Na rozdíl od běžné MVC architektury využívá WebFlux jiný model souběžného programování. Spring MVC předpokládá, že při každém požadavku budou vlákna blokována, a proto využívá velké množství předem definovaných vláken. To způsobuje jeho velkou náročnost na zdroje. WebFlux oproti tomu využívá menší, předem dané množství vláken, protože předpokládá, že vlákna budou aktivní, i při vzniku blokující události. Tato vlákna se nazývají „event loop workers" a prochází příchozími požadavky rychleji než MVC. [38]

3.5.1 Reaktivní systém

Systémy navržené podle vzoru, který upřednostňuje použití na sobě nezávislých flexibilních a škálovatelných komponent. Myšleno je i na zachování funkčnosti při výpadku jednoho ze systémů. Reaktivní systémy dokážou obsluhovat více neblokujících požadavků za jeden časový úsek. Požadavky tak na sebe nemusí navzájem čekat. Díky tomu dosahují lepších výsledků ve výkonosti a škálovatelnosti. Reaktivní systémy se tedy zaměřují na:

- **Reactiveness** – reaktivní systémy by měly odpovědět rychleji na jakýkoliv uživatelský požadavek a měly by zlepšit uživatelský zážitek a práci s daty.
- **Resilience** – návrh by měl obsahovat řešení chybových stavů. Při chybě v jedné komponentě by nemělo dojít k nefunkčnosti celého systému.
- **Elasticity** – systém se musí přizpůsobit velikosti pracovní zátěže. Některé implementace také obsahují prediktivní škálování, aby vyhověly těmto potřebám.

- **Message-driven communication** – všechny komponenty reaktivního systému jsou propojeny hranicemi. Systém by měl využívat komunikaci přes zprávy mezi komponentami a informovat je o selhání a následné delegaci na komponenty určené pro tuto situaci. [38]

3.5.2 Project Reactor

Project Reactor je reaktivní knihovna založená na specifikaci Reactive Streams pro vytváření neblokujících aplikací v JVM. Project Reactor je základem reaktivního zásobníku v ekosystému Spring a je vyvíjen v úzké spolupráci se Springem pod názvem WebFlux. [10]

Práce s reaktivními streamy je obdobná jako práce s obecnými streamy dostupnými v Javě 8. Vývojář tak má plnou kontrolu nad tokem dat v reaktivních streamech. Komponenta si může kdykoliv vyzvednout data, pokud jsou k dispozici, nebo je cashovat na své straně. V případě, že některá z komponent nestíhá data v reaktivním streamu dostatečně rychle zpracovávat, může aktivovat tzv. „backpressure“. To umožňuje snížení proudu dat z nadřazené komponenty, aby nedošlo k přetížení. [38]

Řešení nabízí dvě datové struktury pro práci s těmito streamy:

- **Flux** – komponenta pracující s proudem 0–N dat.
- **Mono** – komponenta pracující s proudem 0–1 dat. [38]

API poskytuje i čtyři hlavní rozhraní pro snadnější práci s reaktivními streamy. Jejich implementace mají automaticky zabudovanou podporu back-pressure a asynchronního zpracování:

- **Publisher** – funguje jako centrální bod pro subscribery a poskytuje nové události podle jejich potřeb.
- **Subscriber** – přijímá a zpracovává zprávy zaslané publisherem. Nemusí jít pouze o spojení 1:1, ale několik subscriberů se může navázat na jednoho publishera a reagovat rozdílně na tu samou událost.
- **Subscription** – definuje vztah mezi subscriberem a Publisherem.
- **Processor** – reprezentuje aktuálně zpracovávanou část subscribera. [38]

3.6 PostgreSQL

Open-source relační databáze poskytující funkcionality jazyka SQL, jako jsou cizí klíče, vnořené dotazy, triggery a možnost definovat vlastní datové typy a funkce. Je primárně

využívána pro uchování dat webových, mobilních, geolokačních a analytických aplikací. V současné době se jedná o druhou nejpoužívanější relační databázi hned za MySQL. [31]

3.6.1 Spolehlivost a dodržování norem

Databázový systém plně dodržuje ACID zásady pro provádění transakcí a má plnou podporu cizích klíčů, dotazů napříč více tabulkami, tvorbu pohledů a ukládání procedur a funkcí. Obsahuje nejpoužívanější datové typy jazyka SQL: celočíselné, textové, datové a logický true/false. Dodatečně umožňuje ukládat binární soubory velkého objemu jako obrázky, videa a zvukové nahrávky. [31]

3.6.2 Rozšíření

Databázový systém je možné rozšířit o další dodatečné funkcionality:

- Obnovení v čase.
- Řízení souběžnosti více verzí.
- Souběžný zápis a čtení jednotlivých řádků tabulky.
- Blokace souběžných aktualizací stejného řádku a přecházení kolizím. [31]

3.6.3 Škálovatelnost

Umožňuje souběh více uživatelů a dokáže spravovat velké množství dat. Je navíc nezávislý na operačním systému a je možné jej provozovat na řešení od všech známých výrobců: Microsoft, Mac OS X, Linux, FreeBSD a Solaris. [31]

3.7 Liquibase

Open-source nástroj pro automatizované nasazování databázových změn napsaný v jazyce Java. Zabraňuje riziku nekonzistence databázového schématu napříč různými prostředími. Je možné jej zařadit do CI/CD procesu a minimalizovat tak riziko opomenutí změn při nasazení nové verze softwaru. Databázové změny jsou definovány v „changelog“ souborech. Ty mohou mít následující formáty: SQL, XML, YAML, JSON. Při spuštění jsou převedeny do SQL dotazů. Mezi hlavní výhody patří:

- Automatické nasazení databázových změn.
- Konzistentní nasazení na všechna prostředí.
- Možnost vrácení změn při chybovém stavu během nasazení.
- Detailní informace o každém nasazení.

Pro správné fungování nástroje jsou potřeba dvě dodatečné tabulky v databázi. Liquibase je v případě integrace do Spring Boot aplikace automaticky vytvoří při prvním spuštění:

- **DATABASECHANGELOG** – obsahuje detailní historii provedených změn.
- **DATABASECHANGELOGLOCK** – zabraňuje uživatelům dělat v databázi změny ve stejném čase a udržuje tak konzistenci. [32]

3.8 Redis

Open-source je založená na principu klíč-hodnota. Hlavní využití nalezne při budování vysokovýkonných a škálovatelných webových aplikací. Mezi základní vlastnosti patří:

- Celá databáze je uložena v paměti RAM.
- Oproti alternativním řešením má Redis bohatší množství datových typů. Mezi ně patří základní kolekce, jako je list, set (seřazený i s využitím hashování).
- Data mohou být replikována napříč instancemi.
- Průměrná rychlost pohybuje okolo 110 000 zápisů a 81 000 přístupů za sekundu.
- Všechny operace jsou prováděny atomicky, což zajišťuje, že dva a více klientů si vzájemně nenaruší přístup.

Redis je možné využít na více typů funkcionalit, jako je cashování, messaging queues a pro uchování dat s krátkou dobou uložení. [33]

3.9 GraphQL

Dotazovací jazyk, který se vyznačuje větší efektivitou a flexibilitou při práci s API než REST. Původně vyvíjený jako open-source projekt společností Facebook. Dnes je spíše udržován komunitou. Api je zapsáno jako sada pravidel, které umožňují programům komunikovat mezi sebou. GraphQL je čistě dotazovací jazyk, zatímco REST je spíše architektonický styl pro navrhování webových služeb. [34]

3.9.1 Hlavní výhody oproti REST

Obě řešení mají své výhody, ze kterých mohou profitovat. GraphQL dotazy samy o sobě nejsou rychlejší než REST. Ale díky možnosti si vybrat konkrétní data bude odpověď ze serveru vždy menší velikosti a tedy efektivnější. Oproti tomu REST vrací vždy všechna požadovaná data, tedy i ta, které klient nutně nemusí využít. Navíc není nutné implementovat datovou strukturu pro celou odpověď, ale vždy jen pro data, která jsme si vyžádali. Jazyk také umožňuje zažádat

o data z více vstupních bodů v rámci jednoho požadavku. Při použití REST by klient musel volat více endpointů. [34]

3.9.2 Hlavní nevýhody oproti REST

I přes převažující výhody je lepší v některých situacích využít REST rozhraní. GraphQL nalézá využití hlavně ve větších aplikacích. Stejného efektu můžeme dosáhnout i s REST v případě menších aplikací, neboť data mívají menší množství atributů a není tedy potřeba provádět selekci. [34]

Díky standardizovaným chybovým kódům je analyzování chyb v REST mnohem jednodušší. Chybové stavy jsou v rámci GraphQL definovány ve vlastním atributu v odpovědi. Jejich hodnotu si však vývojář určí sám a může tak dojít k nejednoznačné identifikaci chyby. [34]

3.9.3 Schéma soubor

GraphQL schéma soubor popisuje funkcionalitu dostupnou pro klienta. Díky tomu je možné použít GraphQL v jakémkoliv programovacím jazyce. Stačí použít dostupné knihovny nebo si vytvořit vlastní rozhraní: [36]

3.9.4 Validace

Pro dodržování validnosti dotazu využívá GraphQL systém „skalární“ typové kontroly. Datové typy jsou reprezentovány jako strom, jehož větve jsou zakončeny primitivními datovými typy. Úroveň složitosti tohoto stromu záleží na typu odpovědi. Mezi základní datové typy, které jsou zabudované v GraphQL, patří:

- Int – 32bitová reprezentace celého čísla (záporná i nezáporná).
- Float – reálné číslo obsahující desetinou tečku (záporná i nezáporná).
- String – textový řetězec podle standardizace UTF-8. Může obsahovat i číslo větší, než je limit datového typu Int.
- Boolean – true nebo false.
- ID – unikátní identifikátor. Reprezentováno vždy jako textový řetězec (např. UUID), ale lze jej definovat i jako celočíselný datový typ.

Další dodatečné datové typy, které nejsou součástí GraphQL, je možné definovat jako tzv. „scalars“. Mezi ně patří například Date, Time nebo Url.

K dispozici je také výčtový datový typ neboli „enum“. Hodnota atributu pak musí nabývat jedné z definovaných:

Složitější datové struktury je možné definovat pomocí klíčového slovíčka „type“:

Kolekce jsou reprezentovány jako list. Atribut definujeme jako kolekci zapsáním jeho datového typu do hranatých závorek.

3.9.5 Operace

Mezi základní operace, které je možné volat na serveru, patří:

- **Query** – čtení dat ze serveru. Alternativou v REST rozhraní je operace GET.
- **Mutation** – zápis nebo aktualizace dat na server. Alternativou v REST rozhraní jsou operace POST, PUT a DELETE.
- **Subscription** – asynchronní čtení dat ze serveru pomocí Websocketu ve webové aplikaci. [35]

3.9.6 Spring GraphQL

Podporu GralphQL můžeme přidat do Spring Boot projektu přidáním závislosti „Spring for GraphQL“. Tato závislost využívá funkcionality projektu „GraphQL Java“. Pro navázání funckionalit jsou dostupné následující anotace:

- **@SchemaMapping** – definuje obecnou GraphQL operaci nebo vnořenou podoperaci. Typ operace se určuje dodatečnými atributy. Umisťuje se nad metodu.
- **@QueryMapping** – definuje vstupní bod pro Query operaci. Umisťuje se nad metodu.
- **@MutationMapping** – definuje vstupní bod pro Mutation operaci. Umisťuje se nad metodu.
- **@Argument** – definuje vstupní parametry operace. V metodě se umisťuje k parametrům. Je nutné zajistit, aby název parametru byl shodný s názvem parametru v GraphQL schématu. Jinak je potřeba rozdílný název uvést v dodatečném parametru k anotaci. [37]

3.10 Vite

Javascriptový nástroj pro sestavování, zjednodušení a urychlení vývoje frontendových webových aplikací. Byl vytvořen vývojářem a autorem knihovny Vue.js Evanem Yu v roce 2021. Obsahuje HMR, který umožňuje okamžitě nahrát nově provedené změny

v Javascriptových modulech do webové prohlížeče při úpravě kódu vývojářem. Součástí jsou i skripty a nástroje pro sestavení produkční verze s minimální konfigurací. [7]

Po instalace je možné založit nový projekt z příkazové řádky pomocí nástroje NPM. Podporuje většinu známých frontendových frameworků a knihoven jako React.js, Vue.js, Svetle, Lit a Preact. Při založení React.js projektu si vývojář může vybrat i podporu Typescript. Při vývoji jsou k dispozici v nástrojích pro vývojáře webového prohlížeče originální soubory se zdrojovým kódem (EC moduly), nikoliv pouze zkompilovaný javascriptový kód. [7]

3.11 React.js

Deklarativní javascriptová knihovna pro snadnou tvorbu uživatelských rozhraní. React.js je označován jako „component-based“ knihovna. Komponenta je část zdrojového kódu, která reprezentuje různé části webového rozhraní (např.: tlačítko, video nebo text). Jednotlivé změny komponent jsou řízeny pomocí stavů. Změna stavu způsobí okamžité překreslení dané komponenty. [8]

3.12 TypeScript

Jazyk vyvinutý společností Microsoft v roce 2012 doplňující JavaScript o typovou kontrolu. Při překladu TypeScript zastupuje roli kompilátoru, kde je cílovým jazykem překladu Vanilla JavaScript. Díky tomu je dosažena zpětná kompatibilita se staršími webovými prohlížeči. Proces kompilace zároveň poskytne vývojáři zpětnou vazbu na chyby v kódu, mezi které patří:

- Odkazovaní na proměnné, které neexistují.
- Volání metod a vlastností na objektech, které obsahují jinou referenci.

Naopak při použití JavaScriptu jako dynamického jazyka jsou chyby v typové kontrole zjištěny až při spuštění kódu ve webovém prohlížeči. [9]

3.13 Tailwind

Vysoce škálovatelný low-level CSS framework pro rychlou tvorbu vlastního UI. Zápis se provádí přímo do atributu „class“ v HTML tagu. Vývojář je tedy kompletně osvobozen od psaní CSS stylu. Výběr designu je velmi intuitivní podle názvu třídy, která odpovídá aplikovaným vlastnostem. Tento název je zvolen tak, aby byla aplikována co největší abstrakce a znovu použitelnost a nedošlo tak ke zbytečnému navyšování velikosti projektu. Kromě stylování stránky umožňuje také aplikovat plně responzivní design. Dodatečné styly, jako jsou barvy, velikosti písem, stíny a další, je možné definovat v konfiguračním souboru. Díky tomu, že je

Tailwind napsán v JavaScriptu, je možné využít syntaxe programovacího jazyka při aplikaci tříd. [40]

3.14 Apollo Client

Knihovna, která zjednodušuje zprávu stavů, vzdálených a lokálních dat získaných přes GraphQL. Obsahuje systém cashování a deklarativní zápis který pomáhá vývojáři pracovat s daty ze serveru snadněji při zapsání menšího množství kódu. Deklarativní zápis zajišťuje za vývojáře proces získání dat ze serveru od začátku do konce, včetně načítání a chybových stavů, s napsáním jen minimální konfigurace. [41]

3.15 Axios

Isomorfní http klient pro Node.js server a webový prohlížeč. Na straně serveru využívá klasický nativní node.js http modul, zatímco na straně klienta ve webovém prohlížeči využívá XMLHttpRequests. Podporuje všechny známé metody REST rozhraní. Mezi základní vlastnosti patří:

- Asynchronní volání na straně prohlížeče nebo serveru.
- Podpora Promise API.
- Deserializace dat z odpovědi nebo serializace dat pro požadavek. [42]

3.16 Docker

Open-source platforma pro vývoj, nasazování a správu aplikací v kontejnerizovaném prostředí. Umožňuje oddělit aplikaci od infrastruktury a tím urychlit proces nasazení softwaru. Kontejnerizace začala nabývat na popularitě s přesouváním firemních aplikací do cloud-native a hybrid multicloud prostředí. K červnu roku 2020 činil počet publikovaných repositářů na Docker Hub přes 7 miliónů. [11]

Docker využívá klient-server architekturu, kdy klient komunikuje s Docker deamon, který provádí tvorbu, spouštění a distribuci kontejnerů. Klient a server se nemusí nacházet na jednom zařízení, ale je možné je oddělit a komunikaci provádět vzdáleně pomocí REST API, UNIX socketů nebo síťového rozhraní. [11]

3.16.1 Docker Desktop

Desktopová aplikace volně dostupná pro operační systémy Mac OS, Windows i Linuxové distribuce. Je vhodná pro uživatele, kteří preferují správu kontejnerizace, aplikací a image přes

uživatelské rozhraní místo příkazové řádky. Instalační balíček také obsahuje Docker deamon, klienta, Docker compose a další nástroje. [12]

3.16.2 Kontejner

Docker kontejnery jsou odlehčenější verzí virtualizovaného prostředí pro běh aplikací. Každý kontejner obsahuje softwarové balíčky, systémové nástroje, závislosti a konfigurační soubory pro běh dané aplikace. Kontejner je zcela izolovaný a nezávislý na hostitelském zařízení a ostatních instancí. [13]

Kontejnery jsou na rozdíl od virtuálních strojů virtualizovány na aplikační vrstvě a sdílí jádro operačního systému s hostitelským zařízením. Díky tomu jsou méně náročnější na dostupné prostředky a poskytují rychlejší a snadnější konfiguraci. Vlastnostmi agnostického systému jsou i rychlá a jednoduchá konfigurace, spuštění, ukončení a odstranění kontejnerů. Vývojářům umožňuje pracovat nebo spouštět stejný projekt na různých prostředí, aniž by hrozilo nečekané chování nebo snížení výkonu. [13]

3.17 Elasticsearch

Distribuovaný open-source engine pro vyhledávání a analýzu dat naprogramovaný v jazyce Java a postavený na frameworku Apache Lucene. Umožňuje efektivně uchovávat, vyhledávat a analyzovat v obrovských sadách dat v reálném čase. Odpověď bývá v rámci milisekund díky faktu, že Elasticsearch nevyhledává text přímo, ale pomocí indexu. Pro vývojáře je k dispozici REST API. Primárně je využíván při:

- **Fulltextové vyhledávání** – vyhledávání informací v databázi webových stránek s velkým obsahem dat.
- **Enterprise search** – vyhledávání v rámci složitějších systémů a datových struktur, například soubory, blogové příspěvky, uživatelských účtů a dalších typů dat.
- **Logování** – nejčastější způsob využívání pro vyhledávání a analyzování data logů v reálném čase.
- **Statistiky a analýza** – sběr statistických dat z vnitropodnikových systémů, bezpečnostních incidentů a jakéhokoliv jiného druhu podnikání. [43]

3.17.1 Dokument

Základní jednotka informace, kterou může Elasticsearch indexovat v JSON formátu. Pro zjednodušení je možné si tento koncept představit jako databázovou tabulku pro danou entitu.

Nemusí nutně uchovávat pouze texty, ale i složitější datové struktury. Každý dokument má své unikátní ID a zvolený datový typ, který popisuje, o jaký druh entity se jedná. [43]

3.17.2 Index

Kolekce dokumentů, které mají podobnou charakteristiku. Jedná se o nejvyšší level, v rámci, kterého může vývojář provádět dotazy. Každý dokument v indexu bývá logicky uspořádán. Každý index je identifikován jménem, přes který je odkazován dokument během indexování, vyhledávání, aktualizace nebo smazání. [43]

3.17.3 Obrácený index

Obráceným indexem bývá obvykle označován Elasticsearch index. Jedná se o datovou strukturu na principu hash mapy, která uchovává mapování z obsahu, např.: slova nebo číslice, které dále odkazují do dokumentů. Obrácený index neukládá textové řetězce, ale rozděluje je do nezávislých částí v rámci dokumentu. Mapa poté vyhledává dokumenty podle vzoru těchto částí. Takto je možné efektivně vyhledávat libovolný text i při velkém množství dat. [43]

3.17.4 Uzly

Uzel je samostatný server, který může být částí clusteru. Uzel uchovává data a rozděluje je do indexování v rámci clusteru. Uzel je možné nakonfigurovat třemi způsoby:

1. **Master Node** – je zodpovědný za veškeré cluster operace v rámci celého ekosystému. Mezi tyto operace patří vytvoření nebo vymazání indexu a dalších uzlů.
2. **Data Node** – uchovává data a provádí nad nimi operace.
3. **Client Node** – předává požadavky pro manipulaci s clustery do master node a požadavky pro operace nad daty do data node. [43]

3.17.5 Logstash

Open-source nástroj pro agregaci, zpracování a přenos dat do Elasticsearch. Zprostředkovává univerzální kanály pro přenos a transformaci dat z různých systémů. Kromě přenosu je zodpovědný i za identifikaci jednotlivých atributů dat a jejich přepis do běžného formátu. Mezi tyto systémy mohou patřit webové servery, relační databáze a různé cloudové služby. [43]

4 BACKEND

Backend je složen ze třech samostatně oddělených modulů:

- **Hlavní modul** – obsahuje GraphQL API, které slouží k provádění operací nad entitami v databázi.
- **Testovací subsystém** – provádí kontrolu zdrojových kódů a automatizuje tak proces hodnocení úlohy.
- **Search engine** – pomocí nástroje Elasticsearch poskytuje data pro fulltextové vyhledávání.[1]
- **Checkman domain** – knihovní modul obsahující sdílené prostředky pro práci s autorizací na úrovni kurzu.

V rámci této kapitoly je podrobněji popsán hlavní modul. Práce a struktura ostatních modulů je na stejném principu.

4.1 Servisní vrstva

Implementuje business logiku aplikace. Třídy mají vypovídající název složený z názvu služby podle jejího určení a postfixu „Service“. Signatura třídy je opatřena anotací @Service tak, aby bylo možné její zapojení do DI.

- **AppUserService** – CRUD operace nad entitou AppUser.
- **MeService** – operace nad aktuálně přihlášeným uživatelem.
- **MicrosoftAuthenticationService** – komunikace se službou Microsoft SSO. Ověřuje univerzitní účet a registruje nové studenty nebo pracovníky univerzity při prvním přihlášení.
- **AuthenticationServiceImpl** – autentifikace uživatelů systému.
- **ChallengeFileAttachmentService** – CRUD operace nad přílohami úloh.
- **RequirementService** – CRUD operace nad požadavkem úloh.
- **FeedbackService** – CRUD operace nad zpětnou vazbou revizorů pro odevzdaná řešení v rámci úloh.

[1] Pro fungování je nutné, aby služba ElasticSearch měla dostupná data z relační databáze pomocí nástroje Logstash. Při použití služby Docker je nutné, aby daný image obsahoval i ovladače pro komunikaci s databází.

- **ReviewService** – CRUD operace nad ohodnocením vyučujících pro odevzdaná řešení.
- **SolutionService** – CRUD operace nad odevzdanými řešeními v rámci úloh.
- **TestConfigurationService** – operace pro čtení záznamů konfigurace automatických testů.
- **TestResultService** – operace pro čtení výsledků automatických testů.
- **ChallengeAuthorizationService** – kontrola oprávnění pro přístup ke chráněným úlohám.
- **CourseAuthorizationService** – kontrola a přidělení oprávnění nebo přístupu pro kurz v rámci semestru.
- **ChallengeService** – CRUD operace nad úlohami.
- **CourseService** – CRUD operace nad kurzy.
- **GlobalRoleService** – kontrola a přidělení globálního oprávnění.

4.2 GraphQL vstupní body

Většina přístupného API je řešena jako GraphQL rozhraní ve třídách pojmenované postfixem „Controller“ v balíčku „controller“:

- **AppUserController** – operace nad entitou AppUser.
- **MeController** – operace nad aktuálně přihlášeným uživatelem.
- **FeedbackController** – operace nad entitou Feedback.
- **RequirementReviewController** – operace nad entitou RequirementReview.
- **ReviewController** – operace nad entitou Review.
- **SolutionController** – operace nad entitou Solution.
- **TestConfigurationController** – operace nad entitou TestConfiguration.
- **TestResultController** – operace nad entitou TestResult.
- **ChallengeController** – operace nad entitou Challenge.
- **PermitChallengeController** – operace pro povolení přístupu ke chráněné úloze.
- **RequirementController** – operace nad entitou Requirement.

- **CourseController** – operace nad entitou Course.
- **SemesterController** – operace nad entitou Semester.

4.3 Vývojové prostředí

Pro spuštění vývojového prostředí je nutné, aby na zařízení bylo nainstalováno JDK minimálně verze 17 a nástroj Gradle. Pokud se vývojář nechce odkazovat plnou cestou při spuštění příkazů, je potřeba se ujistit, aby cesty k binárním souborům byly obsaženy v systémové proměnné PATH. Vývojové prostředí se spouští z kořene projektu pomocí Gradle skriptů. Pro využití maximálního potenciálu projektu je doporučeno použít IDE IntelliJ IDEA.

4.3.1 Sestavení společných závislostí

Některé moduly mají společnou závislost na knihovním modulu „check-man domain“. Ten je nutné před používáním sestavit a nahrát do lokálního Maven úložiště. K tomu stačí z rootu projektu „check-man domain“ spusti následující příkaz:

Výpis konzole 1 Sestavení a nahrání závislosti do lokálního úložiště

```
./gradlew publishToMavenLocal
```

4.3.2 Stažení závislosti

Před spuštěním vývojového prostředí je nutné doinstalovat veškeré závislosti definované v souboru build.gradle.kst pomocí nástroje Gradle. To je možné udělat integrovaným nástrojem v InteliJ IDEA. Nebo otevřít kořen umístění projektu v příkazové řádce a spustit příkaz pro instalaci závislostí:

Výpis konzole 2 Stažení závislostí Gradle projektu

```
./gradlew dependencies
```

Indikace úspěšného stažení všech závislostí je absence jakýchkoliv chyb v logu:

Výpis konzole 3 Ukázka úspěšného stažení závislostí projektu

```
16:15:47: Executing 'dependencies'...

Starting Gradle Daemon...
Gradle Daemon started in 421 ms

> Task :dependencies

------------------------------------------------------------
Root project 'check-man'
------------------------------------------------------------

...

A web-based, searchable dependency report is available by adding the --scan
option.

BUILD SUCCESSFUL in 4s
1 actionable task: 1 executed
16:15:51: Execution finished 'dependencies'.
```

4.3.3 Povinné služby

Pro úspěšné spuštění projektu je nutné, aby aplikace měla přístup k databázím PostreSQL, Redis a Microsoft. Je doporučeno spíše vytvořit nový spring profil než měnit hodnoty existujícího profilu. Tyto hodnoty je nutné definovat v application.yaml souboru:

spring.r2dbc.url: URL, na kterém je dostupná služba PostgreSQL. Obsahuje použitý databázový ovladač, typ databáze, adresu serveru, port a schéma.

spring.r2dbc.username: Uživatelské jméno pro přístup k databázi.

spring.r2dbc.password: Heslo pro přístup k databázi.

login.provider.oauth2.microsoft.client_id: ID Microsoft aplikace. [2]

login.provider.oauth2.microsoft.client_secret: Secret Microsoft aplikace.[3]

login.provider.oauth2.microsoft.redirect_uri: Přesměrování po úspěšném přihlášení.

[2] Identifikátor aplikace vytvořený přes službu Microsoft Azure. Hodnota uložená v docker profilu má omezenou platnost.
[3] Secret aplikace vytvořený přes službu Microsoft Azure. Hodnota uložená v docker profilu má omezenou platnost.

Zdrojový kód 2 Konfigurace databáze a Microsoft služeb v application.yaml ve Spring Boot projektu

```
spring:
  r2dbc:
    url: r2dbc:postgresql://localhost:5432/checkman
    username: checkman
    password: checkman
login:
  provider:
    oauth2:
      microsoft:
        client_id: f57c8ea6-
        client_secret: xkp8Q~
        redirect_uri: http://localhost:${server.port}/v1/authentication
```

Pokud je instance databáze Redis dostupná na stejném zařízení jako Spring Boot aplikace a neobsahuje žádnou formu autentifikace, není potřeba definovat další dodatečné parametry.

4.3.4 Profily

Spuštění vývojového prostředí je možné založením „run konfigurace“ v IntelliJ IDEA nebo spuštěním následujícího příkazu:

Výpis konzole 4 Spuštění Spring Boot projektu pomocí příkazu bootRun

```
./gradlew bootRun
```

Pokud vývojář nevlastní nebo nechce instalovat další dodatečný software na své zařízení, může využít výhod docker kontejnerizace. Instance všech služeb je možné vytvořit pomocí souboru **docker-compose.yaml** umístěného v kořenu projektu. Stačí spustit následující příkaz v příkazové řádce kořene projektu nebo spustit soubor z nástroje IntelliJ IDEA:

Výpis konzole 5 Vytvoření a spuštění kontejnerů pomocí Docker compose

```
docker-compose -f ./docker-compose.yaml up -d
```

Indikace úspěšného vytvoření všech služeb je změna jejich stavu na „RUNNING“ a absence jakýchkoliv chyb ve výpisu. Vytvořené a spuštěné kontejnery je možné vidět i v aplikaci Docker Desktop:

Výpis konzole 6 Úspěšné spuštění docker kontejnerů

```
[+] Running 4/0
 ⠿ Container check-man-be-db-1 Running 0.0s
 ⠿ Container check-man-be-redis-1 Running 0.0s
 ⠿ Container check-man-be-elasticsearch-1 Running 0.0s
 ⠿ Container logstash Running
```

Pro využití docker kontejnerů je nutné spustit aplikaci pod profilem „docker“. Atributy profilu je poté možné měnit v **application-docker.yaml** souboru, který je součástí projektu:

Výpis konzole 7 Spuštění Spring Boot projektu pomocí příkazu bootRun pod vývojářským profilem

```
./gradlew bootRun --args='--spring.profiles.active=docker'
```

Příkazy spustí integrovaný Tomcat kontejner se Spring kontextem. REST rozhraní je dostupné na adrese localhost pod portem uvedeným v **application.yaml.** Každé spuštění zkontroluje konzistenci liquibase souborů a cílové databáze. Při prvním spuštění tedy dojde k vytvoření všech tabulek, pohledu a nahrání inicializačních dat. Při použití docker profilu se navíc nahrají vzorová data pro účely vývoje nových funkcionalit. Soubory se vzorovými daty jsou umístěny ve složce **/src/main/resources/db/changelog/examples**.

Výpis konzole 8 Výpis logů úspěšně spuštěného Spring Boot projektu

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.10)

2023-04-18 16:45:09.465  INFO 45541 --- [  restartedMain]
.b.a.g.r.GraphQlWebFluxAutoConfiguration : GraphQL endpoint HTTP POST /graphql
2023-04-18 16:45:10.111  INFO 45541 --- [  restartedMain] liquibase.database
: Set default schema name to public
2023-04-18 16:45:13.466  INFO 45541 --- [  restartedMain] liquibase.lockservice
: Successfully released change log lock
2023-04-18 16:45:13.547  INFO 45541 --- [  restartedMain]
o.s.b.web.embedded.netty.NettyWebServer  : Netty started on port 9001
2023-04-18 16:45:13.553  INFO 45541 --- [  restartedMain]
c.f.upce.checkman.CheckManApplicationKt  : Started CheckManApplicationKt in
6.824 seconds (JVM running for 7.046)
```

4.4 Autentifikace uživatelů

Pro maximální pohodlí studentů a učitelů je přihlašování vůči systému řešeno přes Microsoft SSO. Proces registrace je úplně vynechán a pokud uživatel vlastní aktivní účet s doménou Univerzity Pardubice, je automaticky zřízen účet. V rámci mezifakultní vazby není přístup omezen pouze pro studenty FEI, ale je plně přístupný i pro studenty a pracovníky ostatních fakult.

4.5 Autorizace na úrovni kurzu

Pro kontrolu řízení přístupu k jednotlivým kurzům jsou zavedena lokální oprávnění (tabulka 2):

Tabulka 2 Seznam dostupných oprávnění na úrovni kurzu.

ID	Název oprávnění / Hodnota výčtu	Popis
0	ACCESS	Základní oprávnění, které musí vlastnit každý uživatel pro přístup do kurzu. Poskytuje jen minimální možnosti, jako je zobrazení úloh
1	CREATE_CHALLENGE	Opravňuje uživatele k vytváření úloh a jejich editaci
2	EDIT_CHALLENGE	Opravňuje uživatele k editaci jakékoliv úlohy
3	SUBMIT_CHALLENGE_SOLUTION	Opravňuje uživatele k odevzdání řešení
4	DELETE_CHALLENGE	Opravňuje uživatele ke smazání jakékoliv úlohy
5	REVIEW_CHALLENGE	Opravňuje uživatele k revizi odevzdaných úloh
6	VIEW_SOLUTIONS	Opravňuje uživatele ke zobrazení odevzdaných řešení

7	MANAGE_USERS	Opravňuje uživatele ke správně uživatelů zapsaných v kurzu, včetně schvalování žádostí pro přidání do kurzu a přidělování oprávnění
8	VIEW_USERS	Opravňuje uživatele ke zobrazení uživatelů zapsaný v kurzu
9	EDIT_COURSE	Opravňuje uživatele k editaci podmínek pro splnění kurzu
10	VIEW_TEST_RESULT	Opravňuje uživatele ke zobrazení výsledků automatických testů
11	VIEW_STATISTICS	Opravňuje uživatele ke zobrazení statistik ohodnocených úloh
12	VIEW_REVIEW	Opravňuje uživatele ke zobrazení již udělených revizí
13	PERMIT_CHALLENGE_CREDIT	Opravňuje uživatele ke zobrazení a udělování přístupu do zápočtových úloh
14	PERMIT_CHALLENGE_EXAM	Opravňuje uživatele ke zobrazení a udělování přístupu do zkouškových úloh

4.5.1 Deklarativní způsob

Nutná oprávnění pro přístup k metodě je možné definovat pomocí anotace @PreCourseSemesterAuthorize:

Zdrojový kód 3 Deklarace anotace PreCourseSemesterAuthorize

```
@Target(AnnotationTarget.FUNCTION)
@Inherited
annotation class PreCourseSemesterAuthorize(
    val value: Array<CourseSemesterRole.Value> =
[CourseSemesterRole.Value.ACCESS],
)
```

Tato anotace se umístí nad metodu, u které vývojář chce kontrolovat přístup. Pro fungování je nutné, aby metoda měla definované následující dva parametry:

- Libovolně pojmenovaný parametr typu Auhtorization. Tuto hodnotu je možné získat z ApplicqtionContext pomocí DI. Pokud je metoda definována na úrovni REST API nebo GraphQL, je automaticky dosazena.
- Libovolně pojmenovaný parametr typu Long, který bude obsahovat primární identifikátor zdroje, ke kterému je vůči kurzu přistupováno. Tento parametr musí být opatřen jednou z následujících anotací, aby aplikace detekovala, přes který zdroj získá informaci o kurzu, viz tabulka 3:

Tabulka 3 Anotace pro identifikaci ID pro přidělení oprávnění na základě kurzu.

Anotace	Související entita
@SemesterId	Semester
@ChallengeId	Challenge
@RequirementId	Requirement
@SolutionId	Solution
@ReviewId	Review
@TestResultId	TestResult

Kód ověřující oprávnění je definován ve třídě **CourseAccessProfilingAspect.kt**. Zde je za pomoci aspektově orientovaného přístupu odchyceno volání anotované metody. Přes reflexi je získána hodnota obou povinných parametrů, instance aktuálně přihlášeného uživatele a výčet jeho rolí vůči dotazovanému kurzu.

Množinu oprávnění, jež uživatel musí splňovat, definuje vývojář výchozím parametrem v anotaci @PreCourseSemesterAuthorize. Jako hodnotu přijímá množinu výčtového typu. Přihlášený uživatel poté musí mít všechna oprávnění, jinak je mu přístup k metodě nebo funkcionalitě zamítnut. Pokud vývojář neuvede žádné oprávnění, je ve výchozím stavu požadováno pouze ACCESS pro přístup ke kurzu.

Pokud přihlášený uživatel disponuje všemi potřebnými oprávněními, je pokračováno ve standardním volání metody. V opačném případě je do reaktivního proudu zanesena chyba, což zabrání dalšímu vykonání.

5 WEBOVÉ ROZHRANÍ

Webové rozhraní pro uživatele je řešeno SPA. Kód aplikace je oddělen v samostatném modulu a není součástí hlavního projektu, jako je tomu běžné u klasické MVC architektury. Kromě přihlášení jsou veškeré funkcionality podmíněny autentifikací a autorizací uživatele.

Jako základ aplikace byla použita knihovna React.js. Pro urychlení a minimalizaci chyb při vývoji je programování veškerého kódu prováděno v jazyce TypeScript. Pro komunikaci s backendem je využita knihovna Apollo. S ostatními subsystémy nebo jinými REST rozhraními probíhá komunikace přes knihovnu Axios.

5.1 Vývojové prostředí

Pro správné fungování je nutné, aby na zařízení byl nainstalován NodeJS a NPM. Pokud se vývojář nechce odkazovat plnou cestou při spuštění příkazů, je potřeba se ujistit, aby cesty k binárním souborům byly obsaženy v systémové proměnné PATH. Vývojové prostředí se spouští z kořene projektu pomocí nástroje NPM. Zdrojový kód je možné zkopírovat z přílohy diplomové práce nebo naklonováním GitHub repositáře autora na libovolné umístění v zařízení vývojáře. Pro využití maximálního potenciálu projektu je doporučeno použít IDE IntelliJ IDEA nebo WebStorm od společnosti JetBrains.

Před spuštěním vývojového prostředí je nutné doinstalovat veškeré závislosti definované v souboru package.json pomocí nástroje NPM. To je možné udělat integrovaným nástrojem v InteliJ IDEA. Případně otevřít kořen umístění projektu v příkazové řádce a spustit příkaz pro instalaci závislostí:

Výpis konzole 9 Instalace závislostí nodejS projektu

```
npm install
```

Indikace úspěšného stažení všech závislostí je absence jakýchkoliv chyb v logu:

Výpis konzole 10 Detekce úspěšné instalace závislostí nodeJS projektu

```
up to date, audited 425 packages in 1s

76 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Spuštění vývojového prostředí je možné založením „run konfigurace“ v IntelliJ IDEA, nebo spuštěním následujícího příkazu z příkazové řádky z kořenu projektu:

Výpis konzole 11 Spuštění vývojového prostředí nodeJS projektu

```
npm run dev
```

Příkaz spustí webový server, ze kterého bude dostupné frontendové prostředí:

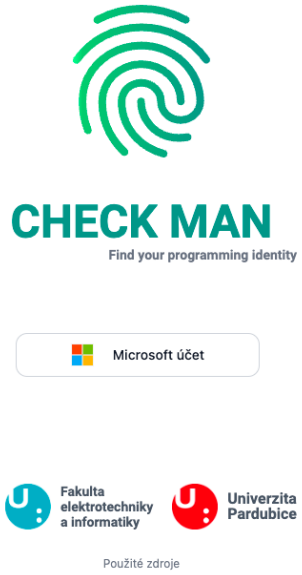
Výpis konzole 12 Ukázka úspěšného spuštění vývojového prostředí projektu s portem

```
vite v2.9.15 dev server running at:

> Local: http://localhost:3001/
> Network: use `--host` to expose

ready in 163ms.
```

Pro fungování webového serveru a HMR je nutné proces udržet naživu a nezavírat okno příkazové řádky. Při použití run konfigurace stačí nechat otevřené vývojové prostředí a neukončovat proces pomocí červeného tlačítka v dolní liště. Otevřením URL ve webovém prohlížeči se zobrazí aplikace, kde se může uživatel přihlásit, viz obrázek 8.

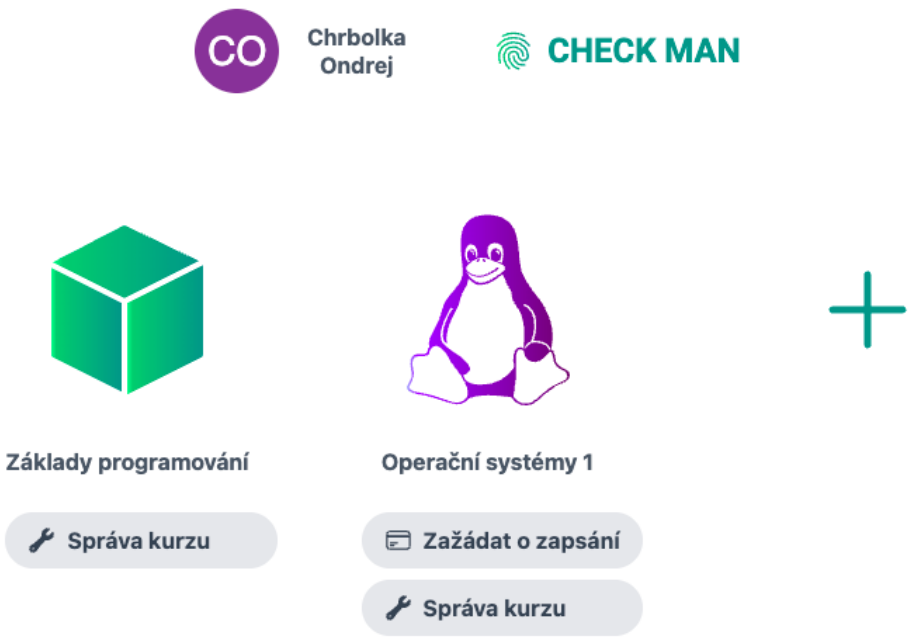


Obrázek 8 Přihlašovací stránka aplikace

5.2 Úvodní menu

Obsahuje ikonky jednotlivých kurzů, do kterých se může student zapsat. Zobrazují se pouze ty kurzy, které je možné absolvovat v daném časovém období. Pokud student není součástí kurzu,

zobrazí se mu pod ikonkou tlačítko pro zapsání. Po kliknutí se student přesměruje na detail kurzu v daném semestru, viz obrázek 9.



Obrázek 9 Hlavní menu aplikace s aktivními i dostupnými kurzy

5.3 Semestr

Obsahuje základní informace o kurzu vyučovaném v daném semestru. Rozvržení stránky je rozděleno na dvě části. Po levé straně se nachází dynamické menu obsahující jednotlivé úlohy dostupné v daném kurzu. Toto menu je možné pro pohodlí uživatelů skrýt tak, aby byla vidět hlavní část stránky. Zbylou část stránky vyplňuje hlavní část, která se mění podle interakcí uživatele v menu.

5.3.1 Podmínky pro splnění kurzu

V horní části detailu jsou uživateli zobrazeny podmínky pro splnění semestru. Podmínky je možné nastavit jako minimální počet splněných úloh v každé kategorii. V případě, že u některé z kategorií není nastaven minimální počet, není uživateli zobrazena. Uživatelé s oprávněním mohou podmínky pro splnění semestru libovolně upravovat pomocí tlačítek pod indikátorem. Provedené změny se automaticky odesílají na backend.

5.3.2 Statistické údaje

Pokud uživatel vlastní oprávnění, může v dolní části stránky vidět i základní statistiky o úspěšnosti v rámci jednotlivých úloh. Statistiky obsahují sloupcové grafy rozdělené do následujících kategorií:

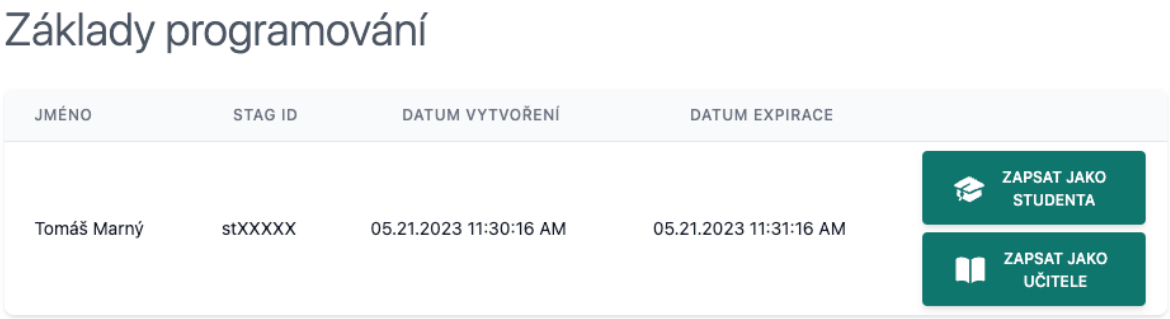
- Nejčastěji udělovaná pozitivní zpětná vazba.
- Nejčastěji udělovaná negativní zpětná vazba.

V dolní části stránky je zobrazena tabulka nejčastěji udělovaných hodnocení. U každého záznamu je zobrazen název hodnocení, typ hodnocení a počet jeho udělení v rámci všech uživatelů a úloh. Mezi jednotlivými statistikami je možné vyhledávat pomocí vyhledávacího pole umístěného nad tabulkou.

5.3.3 Zápis

Uživatelé se do kurzů přihlašují sami odesláním požadavku na hlavní stránce aplikace. Tento požadavek má platnost 60 vteřin a poté je automaticky zamítnut. Během schvalovací doby je uživateli stav indikován pomocí načítací ikonky umístěné pod ikonkou a názvem kurzu.

Uživatel s oprávněním MANAGE_USERS se může pomocí tlačítka v administrační liště přesunout do editoru požadavků. Požadavek je možné schválit, nebo zamítnout přes tlačítka umístěné po pravé straně záznamu. Uživatele lze zapsat jako studenta, nebo vyučujícího. Podle zvolené možnosti jsou automaticky přiřazeny i příslušná oprávnění k daném kurzu. Po schválení je kurz uživateli plně přístupný.



Obrázek 10 Ukázka zápisového formuláře odeslaných žádostí

5.4 Úloha

V rámci semestru může uživatel plnit úlohy. Ty jsou rozděleny do čtyř kategorií (tabulka 4):

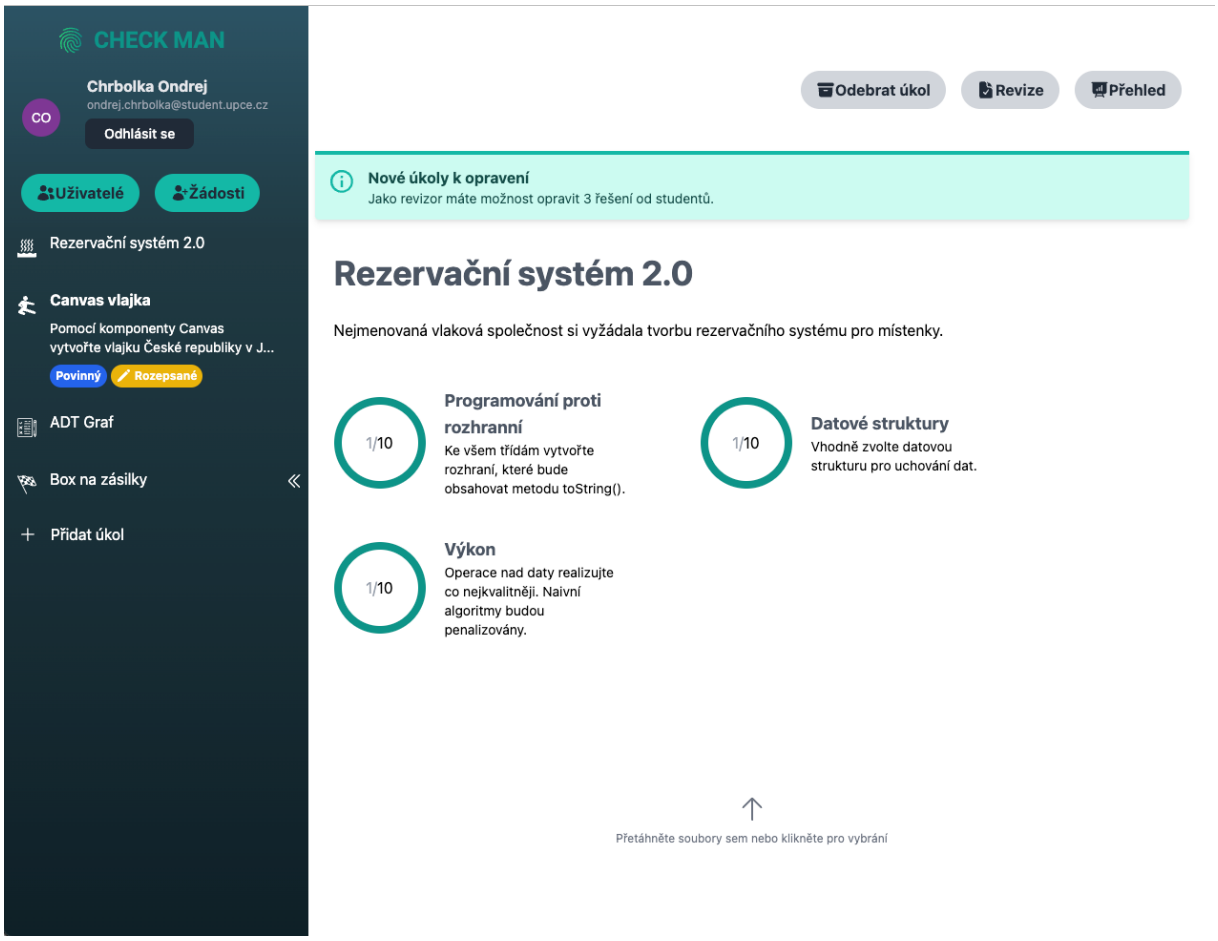
Tabulka 4 Typy úloh

Ikona	Typ	Hodnota výčtu	Popisek
	Dobrovolná úloha	OPTIONAL	Jednoduchá úloha sloužící k procvičení problematiky
	Povinná úloha	MANDATORY	Povinná úloha sloužící k procvičení a zároveň ověření znalostí studenta. Může být i semestrální prací
	Zápočtová úloha	CREDIT	Úloha, jejíž splnění je nutné k udělení zápočtu. Není ve výchozím stavu viditelná pro všechny studenty
	Zkoušková úloha	EXAM	Závěrečná úloha zadávána na termínech zkoušky. Není ve výchozím stavu viditelná pro všechny studenty

Úlohy jsou v menu roztříděny do jednotlivých kategorií. Uživatel může indikovat typ úlohy podle ikonky nebo se po najetí na nadpis úlohy zobrazí čip s názvem typu. Čip je pro snadnější orientaci i barevně odlišený. Po najetí myší na úlohu se zobrazí i část popisku úlohy jako náhled. Ikonky byly navrženy pomocí obrázků z Flaticon.com.

5.4.1 Detail úlohy

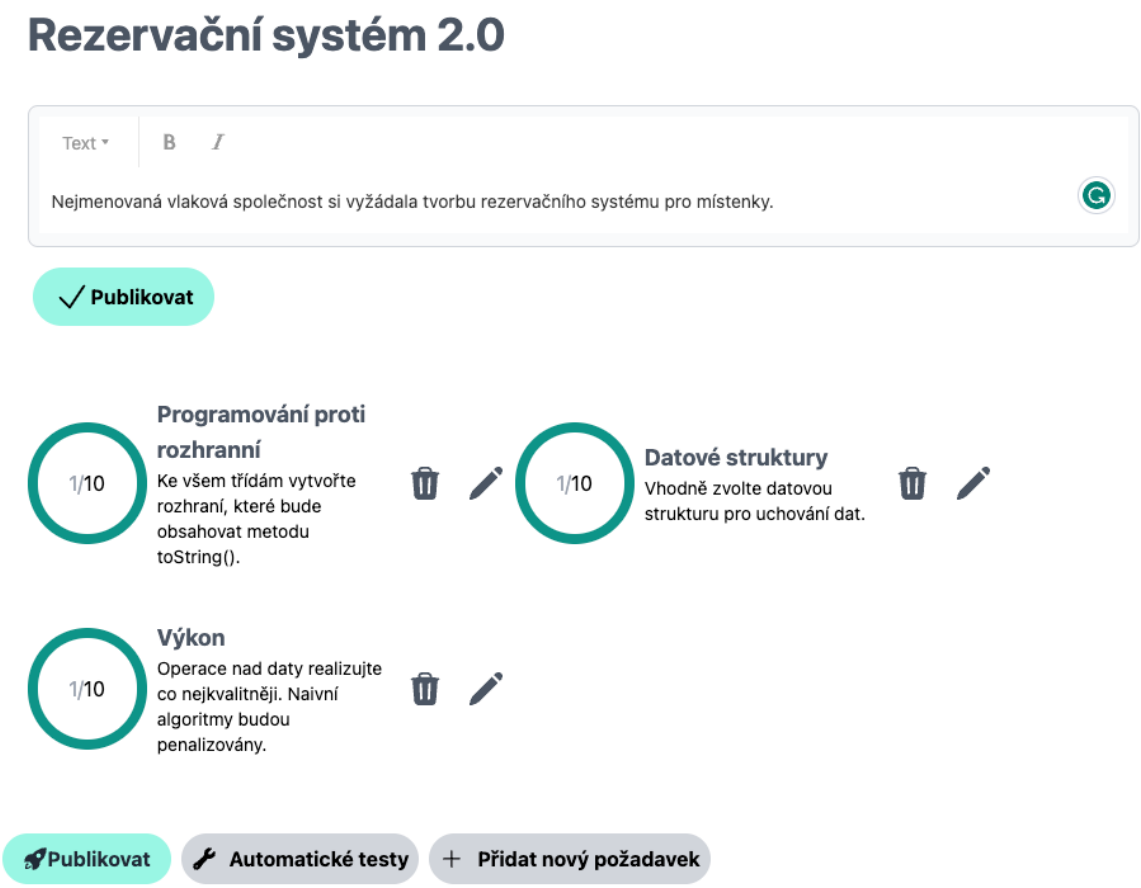
Pro zobrazení detailů stačí kliknout na úlohu v menu. Detail úlohy obsahuje její nadpis a zadání. Pokud úloha není publikovaná a uživatel vlastní oprávnění pro editaci úlohy, je možné zadání dodatečně měnit. Kliknutím na text zadání se stránka transformuje do editoru, ve kterém je možné provádět libovolné úpravy, viz obrázek 10.



Obrázek 11 Detail úlohy s požadavky pro splnění, formulářem pro nahrání a menu pro výběr dostupných úloh

5.4.2 Požadavky úlohy

Za textem následují graficky vyobrazené požadavky úlohy. Jednotlivé požadavky jsou zobrazeny ve formátu karet. V levé části karty je zobrazeno bodové ohodnocení požadavku. Bodové ohodnocení obsahuje minimální počet bodů pro úspěšné splnění požadavku a zároveň i maximální bodové ohodnocení, které student může získat. V pravé části uživatel nalezne nadpis a další dodatečný popis požadavku. Pokud úloha není publikována a uživatel vlastní oprávnění, může požadavky libovolně editovat, přidávat nebo mazat. K těmto operacím lze využít tlačítka umístěná po pravé straně karty, viz obrázek 11.



Obrázek 12 Úloha v editačním módu

5.4.3 Nahrání řešení

Spodní část obsahuje formulář pro nahrání řešení. Formulář je realizován jako drop down oblast pro jednodušší interakci uživatele. Soubor je možné vložit přesunutím ikonky z uživatelova zařízení nebo kliknutím do oblasti. Jako validní vstup se považuje soubor, který je:

- Archiv ve formátu zip.
- Nahrání více souborů není dovoleno.

Další dodatečná kontrola probíhá na serveru. Tyto hodnoty je možné změnit application yaml:

- Maximální velikost souboru je 5 MB.

5.4.4 Odevzdaná řešení

Pod formulářem se nachází seznam již odevzdaných řešení. Odevzdaná řešení jsou seřazena podle data nahrání. Jednotlivé položky seznamu jsou reprezentovány jako karty. Každá karta obsahuje datum nahrání, indikátor stavu revize a jeho název, viz tabulka 5:

Tabulka 5 Stavy revize úloh

Ikona	Typ	Hodnota výčtu	Popisek
	Čeká na revizi	Status.WAITING_TO_REVIEW	Úloha byla odeslána do testovacího subsystému a čeká na revizi od vyučujícího
	Vráceno k opravě	Status.RETURN_TO_EDIT	Úloha byla vyučujícím vrácena k opravě. Student může nahrát nové řešení
	Akceptováno	Status.APPROVED	Úloha byla vyučujícím akceptována. Již není možné nahrát nové řešení
	Zamítnuto	Status.DENIED	Úloha byla vyučujícím zamítnuta. Již není možné nahrát nové řešení

Pokud jsou úloze přiřazeny i automatické testy, zobrazí se indikátor stavu testů v patičce karty. Uživatel si může po kliknutí na tlačítko zobrazit stav testu.

5.4.5 Nastavení automatických testů

Pokud úloha ještě nebyla publikována, může vyučující pomocí tlačítka v patičce stránky nastavit automatické testy. Po stisknutí tlačítka se zobrazí modální okno, ve kterém může uživatel zvolit typ testu podle dostupných modulů na testovacím systému. Modely jsou uživateli zobrazeny ve formě tlačítek v horní části okna. Podle zadání úlohy uživatel vybere příslušný

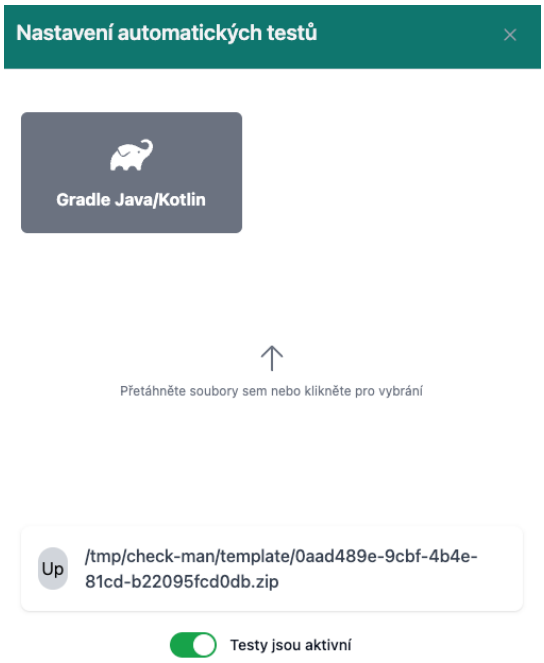
modul. Po vybrání modulu se zobrazí formulář pro nahrání šablony úlohy. Soubor musí splňovat následující požadavky:

- Archiv ve formátu zip.
- Nahrání více souborů není dovoleno.

Další dodatečná kontrola probíhá na serveru. Tyto hodnoty je možné změnit application yaml:

- Maximální velikost souboru je 5 MB.

Soubor je poté automaticky nahrán na server. Pro aktivaci automatických testů je nutné ještě zaškrtnout příslušný checkbox umístěný v patičce okna. Testy je možné libovolně zapínat nebo vypínat do aktivace úlohy, viz obrázek 13.



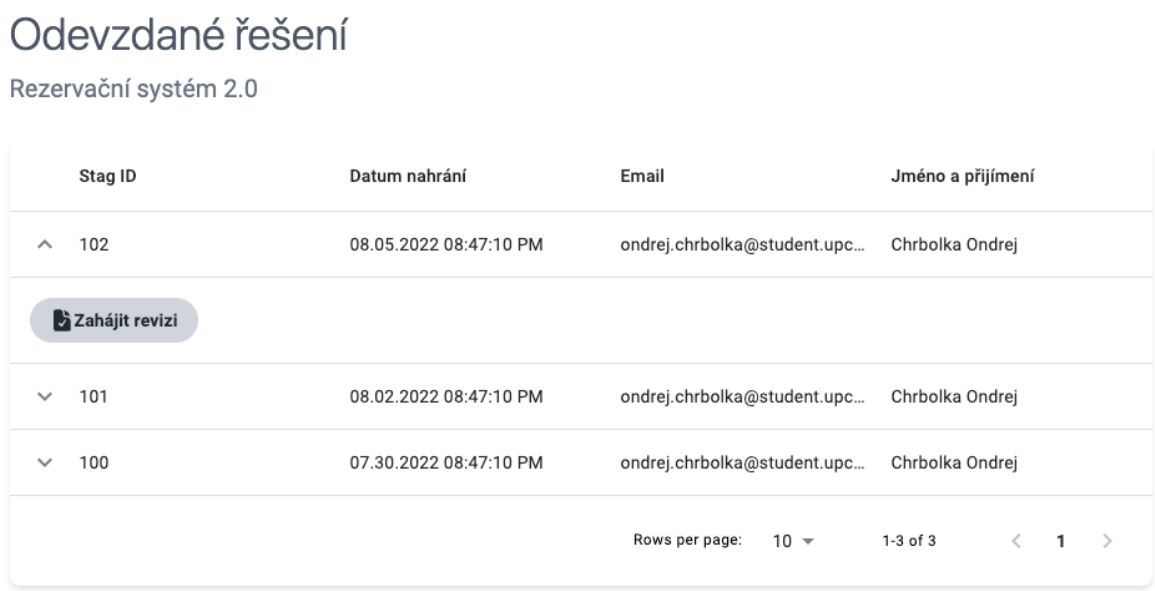
Obrázek 13 Nastavení automatických testů

5.4.6 Založení nové úlohy

Vytvořit novou úlohu může uživatel s oprávněním CREATE_CHALLENGE po kliknutí na tlačítko v levém menu. Následně je uživatel přesměrován na stránku pro zadání názvu úlohy a typu úlohy. Tyto atributy již není možné dodatečně editovat, pouze celou úlohu smazat. Po kliknutí na tlačítko vytvořit je uživatel přesměrován na detail úlohy, která zatím není publikována. Zde je možné dopsat zadání úlohy, vytvořit nebo editovat požadavky a případně nastavit automatickou kontrolu.

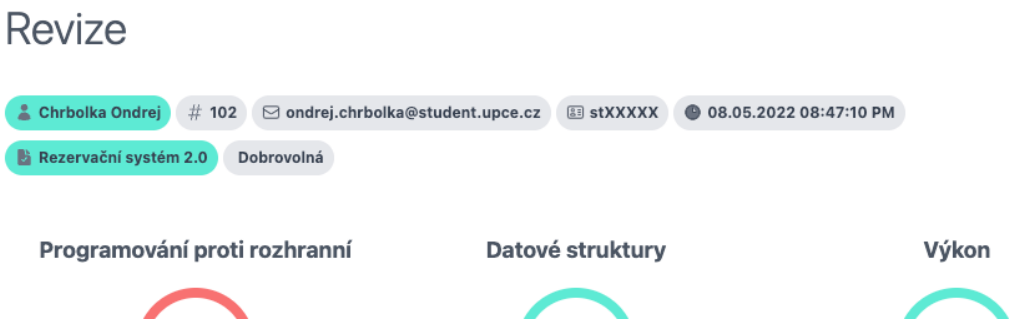
5.5 Revize úlohy

Pokud uživatel vlastní oprávnění REVIEW_CHALLENGE, může pomocí tlačítka v hlavičce stránky provádět revize jednotlivých úloh. Zároveň se v hlavičce detailu úlohy zobrazí notifikace indikující počet úloh k možnému opravení. Po kliknutí na tlačítko je přesměrován na stránku se seznamem možných úloh k revizi.



Obrázek 14 Odevzdaná řešení dostupná k revizi

Každý záznam tabulky obsahuje STAG id, datum nahrání, jméno a příjmení, e-mail uživatele a tlačítko pro přesměrování do editoru k revizi úlohy. Editor pro revizi je rozdělen na čtyři části. Na začátku jsou informace o uživateli a odevzdané úloze, viz obrázek 15.



Obrázek 15 Informace o odevzdaném řešení

5.5.1 Revize požadavků

Pod tabulkami je zobrazen seznam požadavků pro splnění úlohy. Každá položka seznamu je reprezentována jako karta obsahující indikátor splnění nebo nesplnění požadavku. V indikátoru se nachází formulář pro zadání dosaženého počtu bodů a maximální možné bodové ohodnocení, viz obrázek 15.

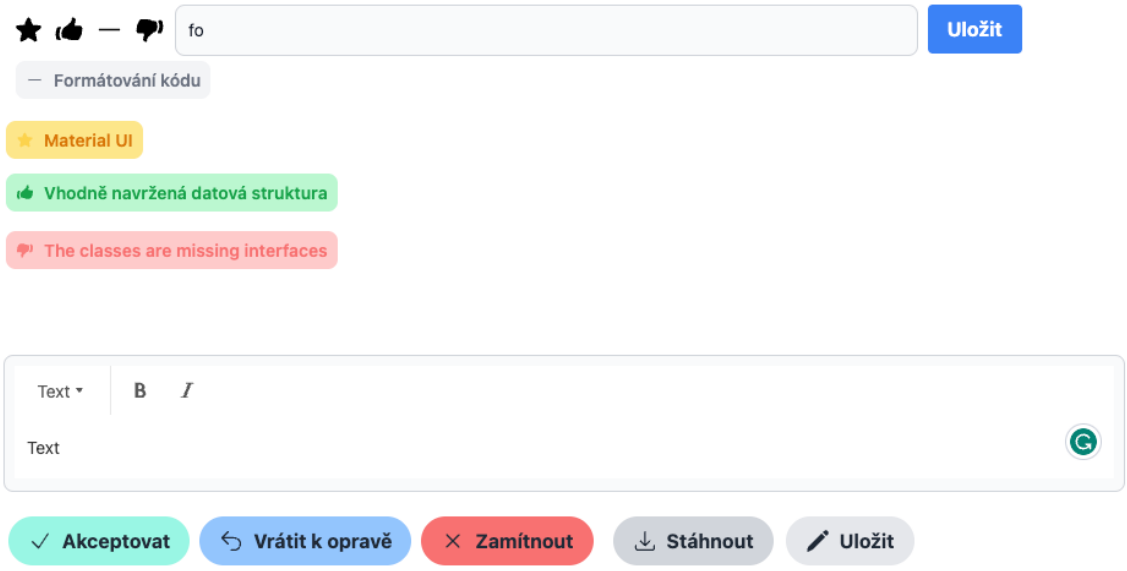
5.5.2 Zpětná vazba

V dolní části stránky se nachází formulář pro zadávání zpětné vazby. Formulář je rozdělen na tři části. První část umožňuje revizorovi zadat dat typ zpětné vazby stisknutím příslušného tlačítko s ikonkou, viz tabulka 6:

Tabulka 6 Typy zpětných vazeb

Ikona	Typ	Hodnota výčtu
☆	Velmi pozitivní	EXTREMELY_POSITIVE
👍	Pozitivní	POSITIVE
👎	Negativní	NEGATIVE
—	Neutrální	NEUTRAL

Slovní ohodnocení může uživatel zadat do vstupního pole ve středu formuláře. Pokud je k dispozici i vyhledávací engine, budou se průběžně pod formulářem zobrazovat již dříve vložená hodnocení. Kliknutím na navrhované hodnocení dojde k okamžitému přidělení. Formulář je zakončen tlačítkem pro uložení zpětné vazby. Editor je zakončen tlačítky pro publikaci a uložení rozpracovaného ohodnocení. Nachází se zde i dodatečné textové pole, do kterého lze dopsat delší text viz obrázek 16.



Obrázek 16 Formulář pro udělení zpětné vazby včetně našeptávače a dodatečnou zpětnou vazbu.

6 TESTOVACÍ SUBSYSTÉM

Subsystém pro testování odevzdaných řešení je samostatná mikroservice, oddělená ve vlastním modulu. Primárním cílem je poskytnout API pro nahrání a vyhodnocení odevzdaných řešení. Vyhodnocení probíhá spuštěním sady jednotkových nebo integračních testů. Výsledky jednotlivých testů jsou ze systému exportovány a uloženy do databáze k danému řešení. Současně jsou výsledky převedeny do formy feedbacku tak, aby bylo možné jejich další zpracování v rámci analýzy a statistiky.

6.1 Servisní vrstva

Následující seznam obsahuje společné třídy pro všechny testovací moduly:

- **AppUserAuthenticationService** – autentifikace uživatelů subsystému.
- **SemesterService** – CRUD operace nad semestry v rámci kurzu.
- **CourseSemesterRoleService** – kontrola a přidělení oprávnění pro kurz v rámci semestru.
- **ReviewService** – CRUD operace nad ohodnocením vyučujících pro odevzdaná řešení.
- **SolutionService** – CRUD operace nad odevzdanými řešeními v rámci úloh.
- **TestConfigurationService** – operace pro čtení záznamů konfigurace automatických testů.

6.2 Zabezpečení

Jakýkoliv zdrojový kód, odevzdaný třetí stranou, představuje potenciální riziko pro zařízení, na kterém je zkompilován a následně spuštěn. I když je aplikace určená pro výuku absolutních základů a předpokládá se, že ji budou obsluhovat nezkušení uživatelé, existuje zde riziko nahrání škodlivého malwaru, který by následně mohl ohrozit funkcionalitu celého zařízení i infrastruktury. Systém využívá výhod kontejnerizace a veškerý kód bude zpracován a následně testován v izolovaném kontejneru. Díky tomu lze minimalizovat riziko ohrožení samotného zařízení.

6.3 Testovací modul

Testovací modul je instance třídy, která provede testování odevzdaného řešení. Testovací subsystém může obsahovat libovolné množství testovacích modelů. Vývojář při implementaci nového modulu musí dodržet:

- Třída testovacího modul musí implementovat rozhraní TestModule.
- Musí být opatřena anotací TestingModule.

6.3.1 TestModule rozhraní

Rozhraní obsahuje dvě metody, které musí vývojář implementovat pro správné fungování testovacího modulu:

Zdrojový kód 4 Rozhraní pro testovací modul

```
interface TestModule {
    fun test(dockerFile: Path, resultPath: Path, testResult: TestResult?,
    log: StringBuffer): Path {
        ...
    }

    fun resultToFeedbacks(resultPath: Path, solution: Solution):
Collection<Feedback>

    ...
}
```

- Metoda **test**, které vývojář implementuje chování pro testování odevzdaného řešení uvnitř docker kontejneru.
    - **dockerFile:** umístění DockerFile souboru obsahujícího konfiguraci pro sestavení image pro testování řešení.
    - **resultPath:** adresář pro nahrání výsledků testování. Toto umístění většinou bývá použito jako volume pro docker image. Jedná se o jediné spojení mezi docker kontejner a hostitelským zařízení.
    - **testResult:** instance entity TestResult. Vývojář ji může využít pro průběžnou aktualizaci logů, případně pro získání dodatečných informací.
    - **log:** instance třídy StringBuffer, do které může vývojář průběžně zapisovat logy uvnitř metody.

- Metoda **resultToFeedbacks**, která převede výsledky z testování uvnitř kontejneru do instancí třídy Feedback tak, aby mohly být uloženy do databáze a přiřazeny k řešení.
    - **resultPath:** adresář po nahrání výsledků testování. Obsahuje soubory získané z docker kontejneru.

6.3.2 Anotace TestingModule

Označuje třídu jako testovací modul pro následnou detekci v rámci testovacího subsystému. Takto označené třídy jsou zároveň označeny jako Spring beany a mohou být využity kdekoliv v projektu pomocí DI. Beana je označena jako prototype. Při každém vyzvednutí z DI je vytvořena nová instance. Do atributů anotace poté vývojář napíše hodnoty určené pro spuštění testovacího modulu:

- **Key:** unikátní identifikátor testovacího modulu v rámci testovacího systému.
- **DockerFilePath:** umístění DockerFile souboru, podle kterého bude vytvořen image pro spuštění kontejneru.
- **Name:** lidsky čitelný název modulu. Tento atribut je poskytován v REST API pro detekci modulu v uživatelském rozhraní.
- **Description:** lidsky čitelný popisek modulu. Tento atribut je poskytován v REST API pro dodatečný popis modulu v uživatelském rozhraní.

Zdrojový kód 5 Deklarace anotace pro identifikaci testovacího modulu

```
@Target(AnnotationTarget.CLASS)
@Retention(AnnotationRetention.RUNTIME)
@Inherited
@Component
@Scope("prototype")
annotation class TestingModule(
    val key: String = "none",
    val name: String = "",
    val description: String = "",
    val dockerFilePath: String = "",
)
```

6.3.3 GradleModule

Testovací subsystém již obsahuje jeden modul určený pro testování zdrojových kódů napsaných v jazyce java nebo kotlin. Podmínkou pro správnou funkčnost je, aby projekt využíval jako

sestavovací nástroj Gradle. Následující příklad ukazuje třídu opatřenou anotací TestingModule a signatury metod.

Zdrojový kód 6 Ukázka implementace testovacího modulu pro Gradle projekty

```
@TestingModule(
    key = "gradle-java-kotlin",
    dockerFilePath = "dockerfile/DockerFile-Gradle-Module",
    name = "Gradle Java/Kotlin",
    description = "Gradle Java/Kotlin testing module"
)
class GradleModule(private val dockerService: DockerService) : TestModule {
    ...

    override fun test(dockerFile: Path, resultPath: Path, testResult:
    TestResult?, log: StringBuffer): Path {
        ...
    }

    override fun resultToFeedbacks(resultPath: Path): Collection<Feedback> {
        ...
    }

    ...
}
```

6.4 Testovací šablona

Testovací šablona je archiv, který obsahuje soubory, pod kterými bude uživatelské řešení otestováno. V případě testovacího modulu GradleModule se jedná o projekt vytvořený pomocí nástroje gradle. Modul je určen primárně pro testování zdrojových kódů odevzdaných studenty v prvním ročníku. Tato skupina studentů má jen minimální znalosti z verzovacích systémů, a proto není vhodné použít jakékoliv jiné dostupné nástroje v kombinaci se CI/CD. V aktuální době probíhá výuka v programovacím jazyku Java za využití moderních IDE (NetBeans, IntelliJ IDEA…).

6.4.1 Tvorba testovací šablony

Zadavatel úlohy musí vytvořit Gradle projekt pomocí některého z nástrojů, například „Gradle Buld Tool“. Tento nástroj je dostupný zdarma pro všechny platformy, včetně Microsoft Windows, Apple Mac OS a Linux. Stažení je možné přes oficiální webové stránky. Případně je možné využít některý z balíčkovacích systémů na unixových a unix-like systémech. Pro

uživatele Mac OS je dostupný v rámci balíčkovacího systému Homebrew. Pro spuštění služby přes příkazovou řádku stačí zadat následující příkaz:

Výpis konzole 13 Vytvoření nového Gradle projektu

```
gradle init
```

Následně je uživateli zobrazeno interaktivní menu a spuštěn průvodce pro vytvoření nového Gradle projektu. V něm má uživatel možnost vybrat ze čtyř druhů projektů. Pro účely výuky 1. ročníku je vhodné zvolit druhou „application“. Při zvolení je vytvořena kostra projektu pro jednoduchou konzolovou aplikaci:

Výpis konzole 14 Výběr typu Gradle projektu

```
Welcome to Gradle 8.0.2!

Starting a Gradle Daemon (subsequent builds will be faster)

Select type of project to generate:
  1: basic
  2: application
  3: library
  4: Gradle plugin
Enter selection (default: basic) [1..4]
```

Ve druhém kroku je zvolen primární jazyk, ve kterém bude uživatel zdrojový kód psát. Zvolíme třetí možnost „Java“:

Výpis konzole 15 Výběr programovacího jazyka Gradle projektu

```
Select implementation language:
  1: C++
  2: Groovy
  3: Java
  4: Kotlin
  5: Scala
  6: Swift
```

Třetí krok umožňuje uživateli rozdělit projekt na více částí, tedy na primární modul s aplikací a další dodatečné knihovny, které budou sdíleny mezi sebou.

Protože jsou úlohy pro 1. ročník jednoduchého charakteru, zvolíme první možnost, tedy projekt s jedinou aplikací bez knihoven:

Výpis konzole 16 Výběr rozdělení Gradle projektu

```
Split functionality across multiple subprojects?:
  1: no – only one application project
  2: yes – application and library projects
```

Ve čtvrtém kroku je zvolen jazyk konfiguračního souboru „build.gradle“. Pro testování nemá tato volba vliv a je plně na uživateli, jakou možnost zvolí.

Výpis konzole 17 Výběr jazyka build.gradle souboru

```
Select build script DSL:
  1: Groovy
  2: Kotlin
Enter selection (default: Groovy) [1..2]
```

Před posledním krokem může být uživatel dotázán na povolení experimentálních funkcí. Pro zajištění maximální funkčnosti v testovacím subsystému je doporučeno tyto experimentální funkce zamítnout. Jako testovací framework je doporučeno zvolit nejmodernější řešení a tím je „JUnit Jupiter“.

Výpis konzole 18 Výběr testovacího frameworku

```
Generate build using new APIs and behavior (some features may change in the next
minor release)? (default: no) [yes, no] no
Select test framework:
  1: JUnit 4
  2: TestNG
  3: Spock
  4: JUnit Jupiter
```

V posledním kroku uživatel vyplní pouze metadata projektu, jako je název a hlavní balíček. Pro fungování testovacího subsystému nemají tyto názvy vliv. Důležitá je hlavně struktura, kterou nástroj vygeneroval.

Výpis konzole 19 Výpis logů po vytvoření projektu

```
Project name (default: Projects): test=module=gradle
Source package (default: test.module.gradle):

> Task :init
Get more help with your project:
https://docs.gradle.org/8.0.2/samples/sample_building_java_applications.html

BUILD SUCCESSFUL in 8m 33s
2 actionable tasks: 2 executed
```

6.4.2 Úprava testovací šablony

Zadavatel úlohy může libovolně přidávat nebo mazat jednotlivé testy. Musí však před nahráním vzorového projektu do testovacího subsystému vyzkoušet jejich funkčnost. Jakkoliv nefunkční nebo nekompilovatelný projekt může ovlivnit funkčnost celého systému. Současně musí dodržet vygenerovanou strukturu složek a souborů. Pro minimalizaci komplikací je vhodné upravovat pouze soubory obsahující zdrojový kód.

Následující zdrojový kód ukazuje jednoduchý jednotkový test testující metodu umístěnou v hlavním modulu zdrojových kódů. Metoda provede součet dvou celých čísel:

Zdrojový kód 7 Ukázka jednoduchého jednotkového testu

```
@Test public void sumPositive() {
     int a = random.nextInt(MIN, MAX);
     int b = random.nextInt(MIN, MAX);

     int expected = a + b;
     int actual = mathOperations.sum(a, b);

     assertEquals(expected, actual);
}
```

Validitu vzorového projektu můžeme otestovat zadáním následujícího gradle příkazu pro spuštění testů. Při prvotním spuštění dojde i ke stažení závislostí.

Výpis konzole 20 První spuštění testů nově vytvořeného Gradle projektu

```
Downloading https://services.gradle.org/distributions/gradle-8.0.2-bin.zip
..........10%.....20%.....30%.....40%.....50%.....
.....60%.....70%.....80%.....90%.....100%
Starting a Gradle Daemon, 1 incompatible Daemon could not be reused, use --
status for details

BUILD SUCCESSFUL in 47s
3 actionable tasks: 3 executed
```

Takto připravený projekt je validní pro umístění do testovacího subsystému.

6.5 DockerFile

Testovací subsystém před každou kontrolou vytvoří nový image, podle kterého bude následně spuštěn kontejner obsahující vzorovou šablonu obohacenou o zdrojové kódy z řešení. Ostatní soubory z odevzdaného řešení jsou z důvodu bezpečnosti ignorovány. Následující příklad obsahuje Dockerfile určený pro spouštění testů v gradle projektu:

Zdrojový kód 8 Deklarace Docker image v Dockerfile

```
FROM gradle:8.0.2-jdk19 AS build
```

Na prvním řádku pomocí klíčového slovíčka FROM definujeme image, který je použit jako základ pro tvorbu nového. Všechny tyto image jsou dostupné na oficiálním Docker hubu. Image již obsahuje nástroj Gradle a je určen pro kompilaci a spuštění Java projektů s verzí JDK 19 a vyšší.

Zdrojový kód 9 Deklarace pracovní složky a COPY příkazů z testovací šablony v Dockerfile

```
WORKDIR /playground
COPY template/settings.gradle template/gradlew template/gradlew.bat /playground/
COPY template/gradle /playground/gradle
COPY template/app/build.gradle /playground/app/build.gradle
COPY template/app/src/test /playground/app/src/test
```

Příkazem WORKDIR definujeme umístění, ve kterém budou následující příkazy probíhat. První sada příkazů COPY zkopíruje nutné soubory projektu Gradle ze šablony nahrané zadavatelem úkolu do pracovního adresáře v docker kontejneru.

Zdrojový kód 10 COPY příkaz z odevzdaného řešení v Dockerfile

```
COPY solution/app/src/main /playground/app/src/main
```

Jediné soubory, které nejsou ze zdrojové šablony zkopírovány, jsou zdrojové kódy obsahující implementaci. Ty jsou překopírovány posledním příkazem z odevzdaného řešení studenta.

Zdrojový kód 11 Vytvoření pracovní složky a nalinkování do hostitelského zařízení v Dockerfile

```
RUN mkdir -p /playground/app/build/test-results
VOLUME /playground/app/build/test-results
RUN chmod -R 777 .
```

Poslední sada příkazů vytvoří v docker kontejneru složku pro uložení výstupu jednotkových a integračních testů. Tato složka je poté pomocí příkazu VOLUME připojena do hostitelského zařízení. Posledním příkazem již jen nastavíme oprávnění pro spuštění gradle scriptu.

6.6 Testovací úloha

Testování je řešeno jako samostatná úloha, která se spouští každých 60 vteřin. Testovací subsystém je zde schopný obsloužit pouze jednu úlohu za 60 vteřin. Asynchronní testování odevzdaných řešení tak zabrání přetížení a možnému nevyzpytatelnému chování testovacího systému. Spouštěcí úloha je definována v souboru TestJob.kt.

Zdrojový kód 12 Zdrojový kód automatické úlohy pro spuštění automatického testu

```
@Component
class TestJob(...) {
    ...
    @Scheduled(fixedRate = 1, timeUnit = TimeUnit.MINUTES)
    fun runTesting() {
        ...
        val solutionToTest = solutionRepository.findFirsToToTest()

        if (solutionToTest != null) {
            val testResult = testingService.initNewTestResult(solutionToTest)
            ...

            try {
                testingService.test(solutionToTest, testResult, log)
                ...
            }
        }
    }
}
```

ZÁVĚR

V rámci diplomové práce byla úspěšně realizována aplikace pro asistenci při hodnocení odevzdaných úloh. Cílem bylo vytvořit efektivní nástroj, který usnadní proces hodnocení odevzdaných řešení a má za cíl zvýšit kvalitu výuky předmětů zaměřených na vývoj softwaru. Jeho využití však nemusí být do budoucna limitováno pouze pro akademické účely.

Práce přispívá k teorii automatizace kontroly zdrojových kódů. Udržuje krok v používání moderních technologií v oblasti vývoje softwaru. Z pohledu uživatele přináší co nejlepší zážitek formou moderního webového rozhraní.

Software je užitečný hlavně pro vyučující, kteří chtějí zautomatizovat některé rutinní činnosti, jako je oprava odevzdaných řešení, a zároveň chtějí studentům poskytnout co nejlepší zpětnou vazbu. Systém zpětné vazby navíc umožňuje studentům pochopit, které dovednosti již ovládají perfektně a které je potřeba dále zlepšovat.

Vzhledem ke stoupajícímu trendu v oblasti umělé inteligence by se mohla diplomová práce v budoucnu rozšířit o podporu nástrojů jako ChatGPT nebo AWS Code Whisperer. Tyto nástroje již obsahují detekci chyb ve zdrojovém kódu nebo mohou vývojáři navrhnout vylepšení pro optimalizaci či zlepšení čitelnosti. Je však nutné podotknout, že tyto nástroje by v žádném případě neměly nahradit práci vyučujícího a měly by pouze asistovat při řešení jednoduchých problémů. Studentům by umělá inteligence mohla asistovat i při zodpovídání jednoduchých dotazů.

Současné řešení poskytuje pouze automatickou kontrolu zdrojových kódů napsaných v programovacím jazyce Java. Je však připraveno i pro implementaci dalších programovacích jazyků nebo úloh. Do budoucna by tedy bylo možné přidat podporu i pro další předměty vyučované na univerzitě a podporu verzovacích systémů, jako je Git.

Diplomová práce představuje významný krok v oblasti automatické kontroly zdrojových kódů oproti současným řešením využívaným na univerzitě. Práce má potenciál zvýšit kvalitu výuky a motivaci nejen z pohledu vyučujících, ale i studentů.

POUŽITÁ LITERATURA

[1] CSS Battle. DiabloDesign: Free Joomla Templates [online]. Kraków: DiabloDesign, © 2005 - 2023 [cit. 2023-04-18]. Dostupné z: https://diablodesign.eu/tools/interactive-learning/css-battle

[2] CSSBattle: Case Study. Vercel [online]. Los Angeles (Kalifornie): Vercel, © 2023 [cit. 2023-04-18]. Dostupné z: https://vercel.com/customers/cssbattle

[3] MCCULLOUGH, Robert. What is HackerRank?. HackerRank: Community [online]. Mountain View (Kalifornie): HackerRank, 2023 [cit. 2023-04-19]. Dostupné z: https://help.hackerrank.com/hc/en-us/articles/115015654028-What-is-HackerRank-

[4] About Us. HackerRank [online]. Mountain View (Kalifornie): HackerRank, 2023 [cit. 2023-04-19]. Dostupné z: https://www.hackerrank.com/about-us/

[5] KLESNILOVÁ, Kristýna. Ladislav Vagner: První verze ProgTestu vznikla za týden. Buď FIT: Časopis FIT ČVUT [online]. Praha: Buď FIT – Časopis Fakulty informačních technologií ČVUT, 2021, 17. 3. 2021 [cit. 2023-04-19]. Dostupné z: https://casopis.fit.cvut.cz/osobnost/ladislav-vagner-prvni-verze-progtestu-vznikla-za-tyden/?ssp=1&darkschemeovr=1&setlang=cs-CZ&safesearch=moderate

[6] Achieve mastery through challenge. Codewars [online]. San Francisco (Kalifornie): Qualified, 2012 [cit. 2023-04-19]. Dostupné z: https://www.codewars.com/

[7] Delaney, Jeff. Vite in 100 Seconds [online video]. YouTube, 2022-02-23. 2 minut 28 sekund. Dostupné z: https://www.youtube.com/watch?v=KCrXgy8qtjM [cit. 2023-04-15].

[8] Cook, Kyle. What Is React And Why You Need To Know It [online video]. YouTube, 2019-08-24. 12 minut 36 sekund. Dostupné z: https://youtu.be/1wZoGFF_oi4 [cit. 2023-04-15].

[9] Delaney, Jeff. TypeScript in 100 Seconds [online video]. YouTube, 2020-11-22. 2 minut 24 sekund. Dostupné z: https://youtu.be/zQnBQ4tB3ZA [cit. 2023-04-15].

[10] JAISWAL, Krishna. Introduction to Project Reactor. Knóldus: konus . commitment . results [online]. Nové Dillí: Knoldus, 2023, September 26, 2022 [cit. 2023-04-16]. Dostupné z: https://blog.knoldus.com/introduction-to-project-reactor/.

[11] Docker overview. Docker docs [online]. Palo Alto (Kalifornie): Docker, © 2013-2023 Docker Inc. [cit. 2023-04-18]. Dostupné z: https://docs.docker.com/get-started/overview/.

[12] Docker Desktop. Docker docs [online]. Palo Alto (Kalifornie): Docker, © 2013-2023 Docker Inc. [cit. 2023-04-18]. Dostupné z: https://docs.docker.com/desktop/.

[13] SIMIC, Sofija. What is Docker?. PhonexiNAP: GLOBAL IT SERVICES [online]. Phoenix (Arizona): PhoenixNAP, © 2022, September 16, 2021 [cit. 2023-04-18]. Dostupné z: https://phoenixnap.com/kb/what-is-docker

[14] What is CI/CD?. GeeksforGeeks [online]. Noida (Uttar Pradesh): GeeksforGeeks, 2023 [cit. 2023-04-21]. Dostupné z: https://www.geeksforgeeks.org/what-is-ci-cd/

[15] What is CI/CD?. Red Hat [online]. Raleigh (Severní Karolína): Red Hat, 2023, May 11, 2022 [cit. 2023-04-21]. Dostupné z: https://www.redhat.com/en/topics/devops/what-is-ci-cd

[16] SHETH, Himanshu. CircleCI vs Travis CI: Comparing The Best CI/CD Tools: A Top-Level View Of CircleCI. LAMBDATEST [online]. San Francisco (Kalifornie): LambdaTest, 2023, October 5, 2020 [cit. 2023-04-21]. Dostupné z: https://www.lambdatest.com/blog/circleci-vs-travis/

[17] KUMAR, Rajesh. What is TeamCity and How it works? An Overview and Its Use Cases. DevOpsSchool: Lets Learn, Share & Practice DevOps [online]. Bengaluru (Karnataka): DevOpsSchool.com, 2023, April 6, 2022 [cit. 2023-04-21]. Dostupné z: https://www.devopsschool.com/blog/what-is-teamcity-and-how-it-works-an-overview-and-its-use-cases/

[18] SHETH, Himanshu. What is Jenkins? Architecture, Installation, and Setup Explained. LAMBDATEST [online]. San Francisco (Kalifornie): LambdaTest, 2023, September 2, 2020 [cit. 2023-04-21]. Dostupné z: https://www.lambdatest.com/blog/what-is-jenkins/

[19] LUTKEVICH, Ben. Kotlin. WhatIs.com [online]. Newton (Massachusetts): TechTarget, 1999–2023 [cit. 2023-04-21]. Dostupné z: https://www.techtarget.com/whatis/definition/Kotlin

[20] KHANI, Behnam. Data Classes in Kotlin. Baeldung: Kotlin [online]. Bucharest (Romania): Baeldung, 2020 [cit. 2023-04-22]. Dostupné z: https://www.baeldung.com/kotlin/data-classes

[21] What Is Spring Framework and Its Advantages. Simplilearn [online]. San Francisco (Kalifornie): Simplilearn Solutions, 2023 [cit. 2023-04-22]. Dostupné z: https://www.simplilearn.com/tutorials/spring-boot-tutorial/what-is-spring-framework-and-its-advantages

[22] CERCENAZI, Abdulcelil. AOP with Spring (Boot). Reflectoring: Where the HOW meets the WHY. [online]. Syndey: Reflectoring, 2023, October 17, 2022 [cit. 2023-04-23]. Dostupné z: https://reflectoring.io/aop-spring/

[23] CHANDRAKANT, Kumar. Introduction to Transactions in Java and Spring. Baeldung [online]. Bucharest (Romania): Baeldung, 2020, November 16, 2022 [cit. 2023-04-23]. Dostupné z: https://www.baeldung.com/

[24] Hibernate. TheServerSide: Your Enterprise Java Community [online]. Newton (Massachusetts): TechTarget, 2000 - 2023 [cit. 2023-04-23]. Dostupné z: https://www.theserverside.com/definition/Hibernate

[25] TYSON, Matthew. What is JDBC? Introduction to Java Database Connectivity. InfoWorld [online]. Needham (Massachusetts): IDG Communications, © 2023, MAY 13, 2022 3:00 AM [cit. 2023-04-23]. Dostupné z:

https://www.infoworld.com/article/3388036/what-is-jdbc-introduction-to-java-database-connectivity.html

[26] TYSON, Matthew. What is JPA? Introduction to Java persistence. InfoWorld [online]. Needham (Massachusetts): IDG Communications, © 2023, MAY 20, 2022 3:00 AM [cit. 2023-04-23]. Dostupné z: https://www.infoworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html

[27] RADIGAN, Dan. Engineering higher quality through agile testing practices: There's still a need for manual testing–but not in the way you might think!. ATLASSIAN [online]. Sydney: Atlassian, © 2023 [cit. 2023-04-27]. Dostupné z: https://www.atlassian.com/agile/software-development/testing

[28] HAMILTON, Thomas. What is Agile Testing? Process & Life Cycle. GURU99 [online]. Ahmedabad: Guru99, 2023 [cit. 2023-04-27]. Dostupné z: https://www.guru99.com/agile-testing-a-beginner-s-guide.html

[29] DUA, Ashwin. Black Box Testing vs White Box Testing: Understanding Key Differences. TURING [online]. Palo Alto (Kalifornie): Turing, © 2023, November 29, 2022 [cit. 2023-04-27]. Dostupné z: https://www.turing.com/blog/black-box-testing-vs-white-box-testing-understanding-key-differences/

[30] Unit Testing Tutorial: A Comprehensive Guide With Examples and Best Practices. LAMBDATEST [online]. San Francisco (Kalifornie): LambdaTest, © 2023 [cit. 2023-04-27]. Dostupné z: https://www.lambdatest.com/learning-hub/unit-testing

[31] What Is PostgreSQL?. Kinsta [online]. Los Angeles (Kalifornie): Kinsta, © 2023 [cit. 2023-04-28]. Dostupné z: https://kinsta.com/knowledgebase/what-is-postgresql/#what-is-postgresql

[32] GRZEGORCZYK, Rafal. What is Liquibase? How to Automate Your Database Script Deployment. DZone [online]. Durham (Severní Karolína): DZone, 2019, Nov. 28, 21 [cit. 2023-04-28]. Dostupné z: https://dzone.com/articles/what-is-liquibase-automate-your-database-script-deployment

[33] Redis - Overview. Tutorialspoint: Simply Easy Learning at your fingertips [online]. Hyderabad City (Telangana): Tutorials Point India Private Limited, © 2023 [cit. 2023-04-28]. Dostupné z: https://www.tutorialspoint.com/redis/redis_overview.htm

[34] MASON, Jeremy. What is GraphQL? (And is it Really Better than REST). Core dna [online]. Melbourne: Core dna, © 2023 [cit. 2023-04-28]. Dostupné z: https://www.coredna.com/blogs/what-is-graphql

[35] RASCIA, Tania. Understanding the GraphQL Type System. DigitalOcean [online]. New York City: DigitalOcean, LLC., © 2023, January 27, 2023 [cit. 2023-04-29]. Dostupné z: https://www.digitalocean.com/community/conceptual-articles/understanding-the-graphql-type-system

[36] GraphQL - Schema. Tutorialspoint: Simply Easy Learning at your fingertips [online]. Hyderabad City (Telangana): Tutorials Point India Private Limited, © 2023 [cit.

2023-04-28]. Dostupné z: https://www.tutorialspoint.com/graphql/graphql_schema.htm

[37] Getting Started with GraphQL and Spring Boot. Baeldung [online]. Bucharest (Romania): Baeldung, 2020, October 7, 2022 [cit. 2023-04-23]. Dostupné z: https://www.baeldung.com/spring-graphql

[38] THELIN, Ryan. Spring WebFlux tutorial: how to build a reactive web app. Educative [online]. Bellevue (Washington): Educative, ©2023, Mar 02, 2021 [cit. 2023-04-30]. Dostupné z: https://www.educative.io/blog/spring-webflux-tutorial#webflux

[39] Spring Boot Tutorial. JavaTpoint [online]. Noida (Indie): www.javatpoint.com, © 2011-2021 [cit. 2023-04-30]. Dostupné z: https://www.javatpoint.com/spring-boot-tutorial

[40] What is Tailwind CSS. Material Tailwind [online]. Bukurešť: Material Tailwind by Creative Tim., © 2023 [cit. 2023-04-30]. Dostupné z: https://www.material-tailwind.com/docs/react/what-is-tailwind-css

[41] Why Apollo Client?: Why choose Apollo Client to manage your data?. APOLLO DOCS [online]. San Francisco (Kalifornie): Apollo Graph, © 2023 [cit. 2023-04-30]. Dostupné z: https://www.apollographql.com/docs/react/why-apollo

[42] Getting Started: Promise based HTTP client for the browser and node.js. AXIOS: Promise based HTTP client for the browser and node.js [online]. John Jakob "Jake" Sarjeant, © 2020 [cit. 2023-04-30]. Dostupné z: https://axios-http.com/docs/intro

[43] ABUEG, Ralf. Elasticsearch: What It Is, How It Works, And What It's Used For. Knowi [online]. Oakland (Kalifornie): Knowi, © 2022, March 7, 2020 [cit. 2023-04-30]. Dostupné z: https://www.knowi.com/blog/what-is-elastic-search/

[44] RUNGTA, Krishna. What is Selenium? Introduction to Selenium Automation Testing. GURU99 [online]. Ahmadábád (Gudžarát): Guru99, © 2023 [cit. 2023-05-01]. Dostupné z: https://www.guru99.com/introduction-to-selenium.html

[45] HAMILTON, Thomas. Integration Testing: What is, Types with Example. GURU99 [online]. Ahmadábád (Gudžarát): Guru99, © 2023 [cit. 2023-05-01]. Dostupné z: https://www.guru99.com/integration-testing.html

[46] A Guide to Functional Requirements (with Examples): Learn how to define requirements and keep all stakeholders aligned. Nuclino [online]. Grünwald: Nuclino, 2015 [cit. 2023-05-07]. Dostupné z: https://www.nuclino.com/articles/functional-requirements

[47] Softwarová analýza a návrh řešení. ARTIO [online]. Ostrava: ARTIO, © 2023 [cit. 2023-05-07]. Dostupné z: https://cms.artio.net/cz/co-delame/sotwarova-analyza-a-navrh#:~:text=K%20tomu%20slou%C5%BE%C3%AD%20softwarov%C3%A1%20anal%C3%BDza.%20Dob%C5%99e%20zpracovan%C3%A1%20anal%C3%BDza,definovat%20po%C5%BEadavky%20na%20software%20a%20popsat%20jeho%20funk%C4%8Dnost.

[48] ČÁPKA, David. Lekce 5 - UML - Class diagram. ITnetwork.cz: Buď žádaným PROFESIONÁLEM! [online]. Praha: itnetwork.cz, © 2023 [cit. 2023-05-07]. Dostupné z: Lekce 5 - UML – Class diagram Zdroj: https://www.itnetwork.cz/navrh/uml/uml-class-diagram-tridni-model

PŘÍLOHY