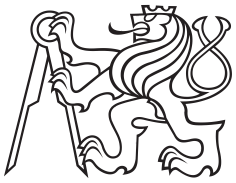
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Software pro vytváření školních testů

Backend

Jan Krčil

Vedoucí: Ing. Petr Aubrecht, Ph.D.
Obor: Softwarové inženýrství
Květen 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Krčil** Jméno: **Jan** Osobní číslo: **491891**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Software pro vytváření školních testů, Backend

Název bakalářské práce anglicky:

Software for School Tests, Backend

Pokyny pro vypracování:

Během lockdownu trpí učitelé nedostatkem softwarové podpory pro vzdálenou výuku, speciálně na základních školách. Zaměřte se na oblast testů, automatizovaného zkoušení, dotazníků ap. Tato práce má za úkol implementovat backend. Úkolem je navrhnout a implementovat vytváření dotazníků, které budou použity pro zkoušení vyučované látky na školách. Analyzujte, jaké typy dotazníků je potřeba vytvořit. Několik nejdůležitějších typů implementujte.
1. Zanalyzujte a popište aktuální situaci v oblasti online dotazníků.
2. Navrhněte řešení, které by učitelům usnadnilo vytvoření dotazníků, snadné zadávání, použití pro žáky, kontrolu správnosti.
3. Řešení implementujte v technologii JakartaEE.
Příklady typů dotazníků, které určitě musí být obsažené, jsou výběr jedné možností, výběr z více možností, plain text, číslo.

Seznam doporučené literatury:

[1] The Jakarta EE 8 Tutorial: https://eclipse-ee4j.github.io/jakartaee-tutorial/toc.html
[2] Jakarta EE Cookbook - Second Edition, https://www.packtpub.com/programming/jakarta-ee-cookbook-second-edition
[3] Patterns of Enterprise Application Architecture — Martin Fowler

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Petr Aubrecht, Ph.D. katedra počítačů FEL

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

Datum zadání bakalářské práce: **08.02.2022** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **30.09.2023**

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

III. PŘEVZETÍ ZADÁNÍ

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

. Datum převzetí zadání

Podpis studenta

Poděkování

Tímto bych rád poděkoval Ing. Petru Aubrechtovi, Ph.D. za pomoc při vypracování této práce a Tomášovi Geržičákovi za spolupráci. Také bych chtěl poděkovat svojí rodině a blízkým za podporu po celou dobu mého studia na ČVUT.

Prohlášení

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

V Praze, 20. května 2022

Abstrakt

Pandemická doba nám ukázala, že v dnešní době chybí uživatelsky přívětivý software pro vytváření školních testů. Také téměř žádný software nepodporuje interaktivnější vyplňování testů, jako například poslechová a doplňovací cvičení, nebo spojování pojmů.

Problém je obzvláště patrný na nižších stupních, kde učitelé často tráví velké množství času vytvářením množství různých testů pro různé třídy. Na vyšších stupních je snaha ze strany učitelů podobné aplikace využívat, často ale naráží na omezení v implementaci nebo opět na uživatelskou nepřívětivost.

Práce pojednává o analýze a tvorbě webové aplikace, která ze šablony generuje testy pro třídu, popřípadě skupinu žáků. Podporuje interaktivní otázky s velkým zaměřením na uživatelskou přívětivost, jak ze strany učitele, tak žáka.

Klíčová slova: online testování, generování testů, Jakarta EE, PostgreSQL, REST, Backend, Webová aplikace

Vedoucí: Ing. Petr Aubrecht, Ph.D.
Katedra počítačů,
Technická 2,
Praha 6

Abstract

The pandemic times have showed us that there is currently a lack of a user friendly quiz creation software. Also, the implementations that are available usually do not support advanced questions like listening exercises, filling in missing words or connecting corresponding terms.

The problem is particularly significant in primary and secondary education, where teachers spend large amounts of time creating tests for different classrooms and with multiple versions for each classroom.

The thesis is about an analysis and implementation of a web application that generates tests for a given classroom, alternatively a different group of students. It supports interactive questions mentioned above with a focus on the user experience of the teacher as well as of the students.

Keywords: online testing, generating tests, Jakarta EE, PostgreSQL, REST, Backend, Web application

Title translation: Online testing software — Backend

Obsah

Obrázky

Kapitola 1

Úvod

Cílem této práce je navrhnout a implementovat software pro snadnou tvorbu a zadávání online školních testů. Prioritou je uživatelská přívětivost. Návrh na spolupráci na tomto projektu jsem dostal od kolegy Tomáše Geržičáka, který jako syn učitelky zaměstnané na základní škole viděl potřebu pro tento software. Čas, který učitelé tráví nad tvorbou testů, obzvláště na nižších stupních a v době pandemie, je opravdu znatelný. Aplikace ale není zamýšlena pouze na online výuku. Výhledově počítáme i s možností generování do pdf a online testy obecně mají využití i mimo tuto specifickou dobu.

Jsme si vědomi toho, že na trhu již pro tvorbu online školních testů existují řešení. Obecně s nimi ale uživatelé nejsou příliš spokojeni. Z toho důvodu bude součástí práce analýza existujících řešení a jejich problémů. Dalším cílem je definovat funkční a nefunkční požadavky a z nich vyplývající případy užití.

Druhá část bude zaměřena na implementaci. Budeme navazovat na již existující řešení OnlineEdu, což je výsledek bakalářské práce Bc. Daniela Kotena, Softwarová podpora pro vzdálenou výuku [1]. Z té budeme používat zejména databázi učitelů, žáků, tříd a s nimi spojené funkcionality. Na aplikaci budeme pracovat ve dvou, s tím, že mojí zodpovědností je backend; část, která u webové aplikace slouží k administraci webu a ke zpracování dat. Kromě základních funkcionalit zajišťujících životní cyklus testů chceme implementovat i méně obvyklé typy otázek, jako například poslechová cvičení, doplňování slov to textu, spojování pojmů nebo otázky založené na obrázcích.

1.1 Rozhovor s učitelem

V úvodní části dokumentu je zmíněno, že existuje potřeba pro tento software. Tento úsudek není založen pouze na zkušenostech kolegy Geržičáka, ale také na řadě rozhovorů s učiteli jak na nižších stupních, tak na ČVUT.

Kolega Geržičák se zaměřil spíše na nižší stupně, informace z této oblasti získané z jeho rozhovorů jsou tedy v jeho práci.

Já jsem vyzpovídal Ing. Pavla Náplavu, Ph.D. Jedná se dle mého názoru, i osobních zkušeností, o jednoho z učitelů nejvíce využívajících možnosti tvorby testů, zejména na Moodle. K tématu online testů měl velké množství relevantních připomínek, především na téma, co by na Moodlu a také na Microsoft Teams změnil, což jsou informace zásadní pro naši práci.

Zaprvé byla zmíněna uživatelská nepřívětivost a kostrbatost Moodle. Dále zde nelze vytvořit otázku, u které by ani jedna odpověď nebyla správná. Generování testů se děje přes tagy, což také není ideální. Co se týká tvorby kvízů na Microsoft Teams, bylo zmíněno málo variant způsobů hodnocení otázek a také absence náhodného generování testů s různými otázkami pro každého studenta.

Ing. Náplava nám také potvrdil, že otázky doplňovacího typu, obrázkové otázky nebo spojování pojmů by využil. Navrhl nám typ příběhové otázky, která vede studenta různě podle toho, jak odpovídá na otázky.

Část I

Analýza

Kapitola 2

Požadavky

Požadavky na systém, které jsou definovány zákazníkem. V našem případě se jedná zejména o učitele, se kterými jsme vedli rozhovory. Ze strany žáků, kteří budou testy vyplňovat, jsme vycházeli z vlastních zkušeností a ze zkušeností našich kolegů.

2.1 Funkční požadavky

Funkční požadavky nebo také uživatelské požadavky jsou abstraktní tvrzení. Specifikují jaké uživatelské funkce by systém měl podporovat[2].

- FP1: Přihlášení
- FP2: CRUD[1] operace s testem a testovou šablonou.
- FP3: Generování testů z testové šablony.
  - Učitel si vytvoří šablonu, kam zadá otázky. Poté bude moc generovat test odlišný pro každého studenta.
- FP4: Zobrazení všech testovacích šablon.
  - Systém umožní učiteli zobrazit seznam všech jím vytvořených testovacích šablon.

[1] Create, Read, Update a Delete

- FP5: Zobrazení testů ze strany studenta
  - Student si může zobrazit testy, které ho teprve čekají, nebo které má už za sebou, kde může provést analýzu výsledků
- FP6: Vyplnění testu
  - Student vyplní a odešle test. V průběhu vyplňování mu bude umožněno se vracet k otázkám a v případě zavření okna nedojde ke ztrátě odpovědí.
- FP7: Odevzdání testu
  - Student odevzdá test. Systém automaticky vyhodnotí otázky, u kterých je to možné. Pokud test obsahuje otázky vyžadující kontrolu učitelem, předá systém tuto informaci žákovi.
- FP8: Učitel provede hodnocení testu
  - Ačkoliv by systém měl podporovat automatické vyhodnocení u většiny typů otázek, učitel musí provést manuální vyhodnocení například u otázek otevřeného typu.
- FP9:Zobrazení výsledků testů
  - Učitel bude mít možnost si zobrazit podrobnou analýzu testu. Bude se tak moc dozvědět, ve kterých otázkách žáci chybovali.
- FP10: CRUD operace s otázkami a odpověďmi na ně

2.2 Nefunkční požadavky

Seznam pozorovatelných vlastností dle kterých lze analyzovat způsob chování a kvalitu fungování systému.[2]

Naše aplikace má nároky zejména na uživatelskou přívětivost a přehlednost která jsou otázkou především frontedu. Backendové požadavky jsou zaměřeny především na stabilitu a bezpečnost.

- NP1: Uživatelská přívětivost
  - Software je zaměřen na učitele na nižších stupních, kde je zásadní, aby prostředí bylo uživatelsky přívětivé.
- NP2: Bezpečnost

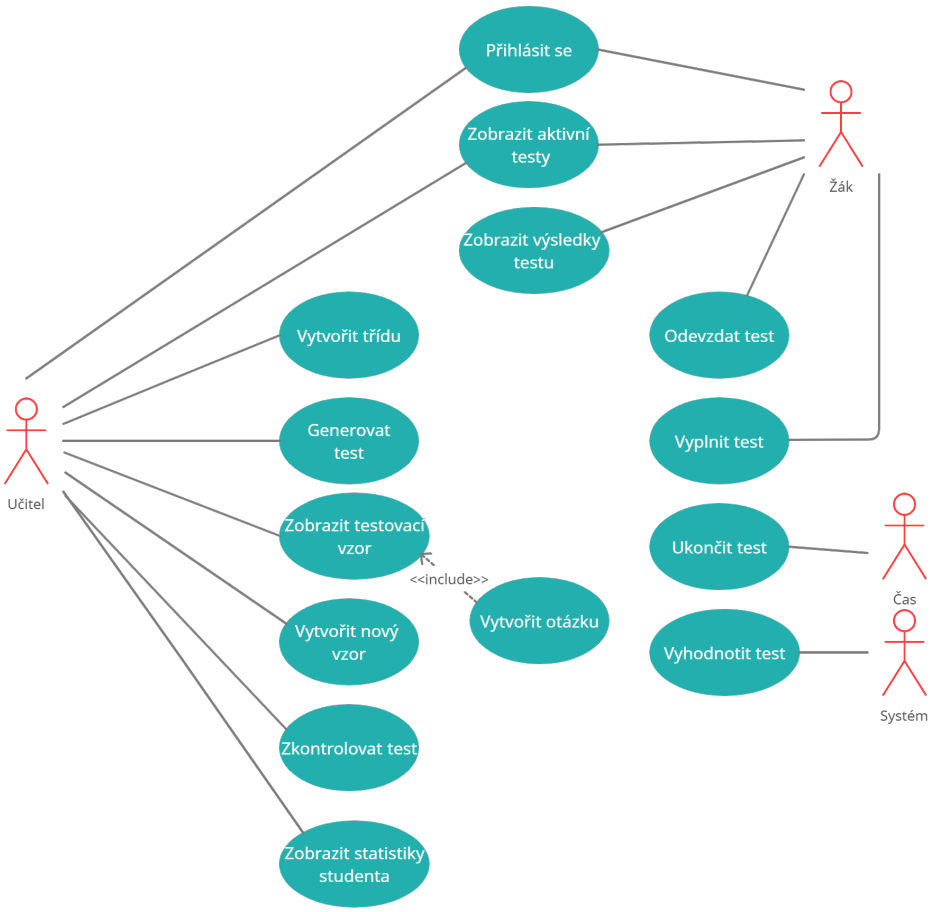
- Aplikace by měla být odolná proti SQL injection, Cross site scripting a Cross site request forgery. Více o těchto útocích a o implementaci tohoto požadavku v kapitole Bezpečnost.

- NP3: Stabilita
  - Jeden z hlavních problémů softwaru jako je Moodle je jeho nestabilita při větším zatížení. Je potřeba software navrhnout tak, aby zvládl, když test bude najednou vyplňovat 60 a více žáků.

Kapitola 3

Případy užití

Seznam případů užití definuje interakce uživatelů se systémem. Pomáhají konkrétněji organizovat a objasňovat požadavky na systém. V případě složitějších případů je lze doplnit scénářem, podle kterého mají probíhat. Případy užití, role a jejich vzájemnou interakci v této aplikaci můžeme vidět na obrázku 3.1



Obrázek 3.1: Případy užití

Kapitola 4

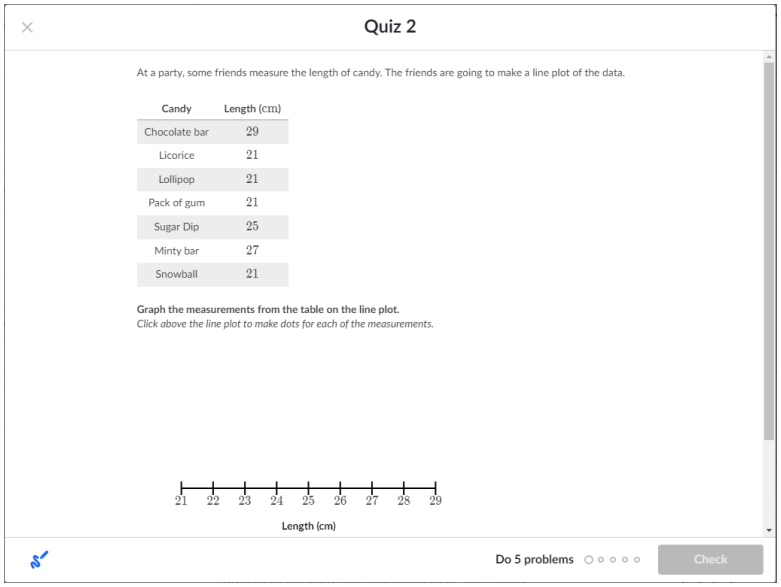
Existující řešení

4.1 Khan Academy

Khan Academy je aplikace primárně určená na vzdělávání, podporuje však i testování formou kvízů. Ačkoliv aplikace neumožňuje samostatnou tvorbu testů, lze zde vytvářet třídy a zadávat studentům kvízy. Učitel pak může vidět, kdo měl jaký výsledek v daném kvízu.

Aplikace je uživatelsky velmi přívětivá, umožňuje importovat google třídy a zadat test imaginární třídě bylo otázkou několika minut.

Software má také velmi pokročilé otázky, kde studentům například umožňuje nakreslit graf myší. Ukázky rozhraní 4.1.



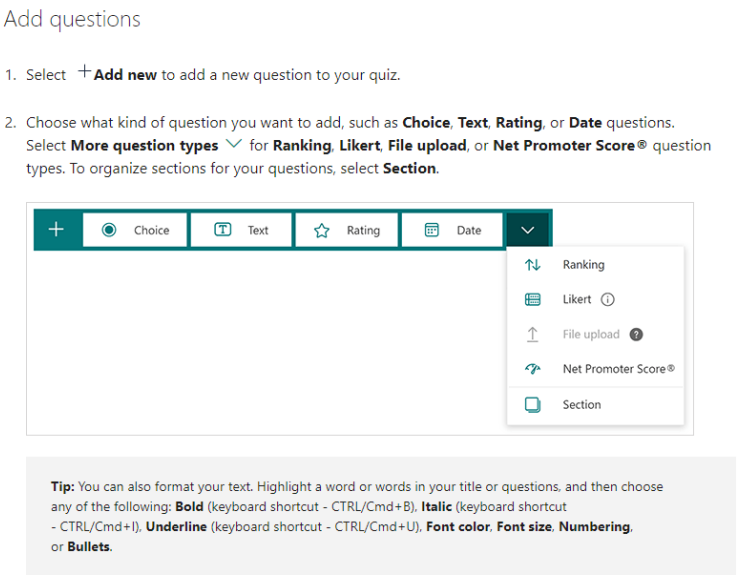
Obrázek 4.1: [3]Příklad testu z webu Khan Academy, kreslení grafu.

4.2 Microsoft teams

Microsoft teams se v průběhu distanční výuky dostal na výsluní zejména jako výuková platforma, která dobře zvládá organizaci na předměty a třídy.

Aplikace však podporuje i tvorbu testů. Testy lze komponovat pomocí otázek, velmi podobně jako se to děje v naší aplikaci. Na ukázce rozhraní 4.2 je vidět okno pro tvorbu otázky. Tato část aplikace je přehledná a dobře zpracovaná.

Nevýhodou Microsoft Teams je, že nemají generování testů sestávajících se z různých otázek ze šablon. Lze zde vytvořit pouze jeden test, který se pak zadá třídě. Je možnost míchat odpovědi, to však případným pokusům o podvádění nezabrání. Dalším problémem je málo způsobů hodnocení kvůli relativně rigidní struktuře nastavování tohoto aspektu.



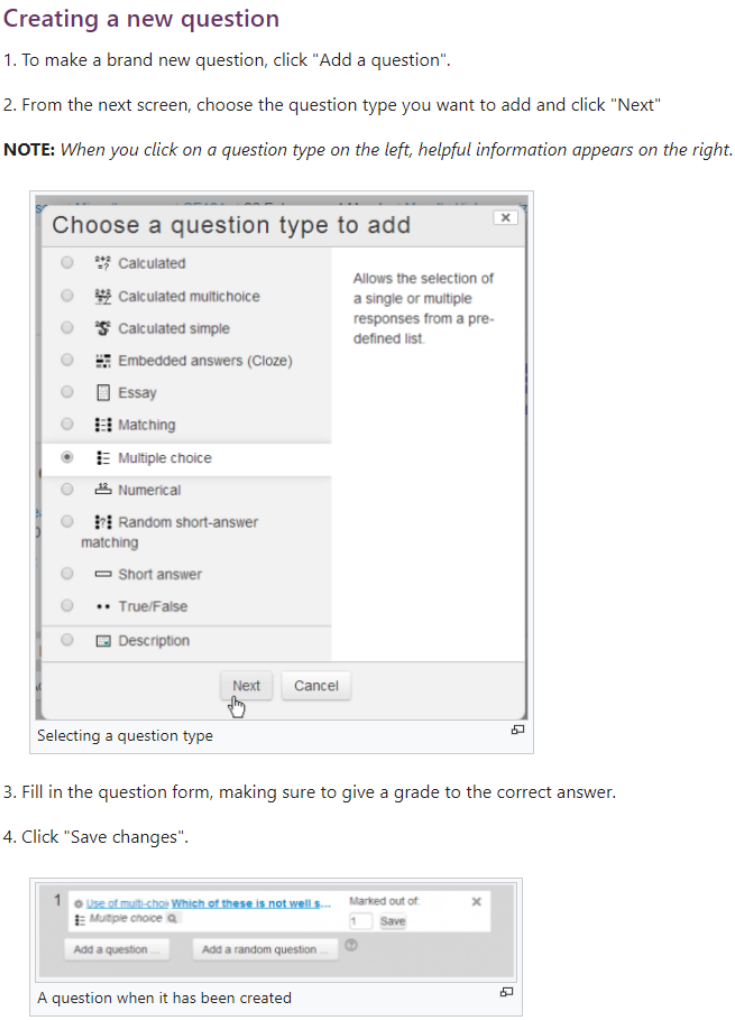
Obrázek 4.2: [4]Příklad tvorby otázky v Microsoft teams

4.3 Moodle

Moodle je software používaný na oboru Softwarové inženýrství. Kromě organizace zdrojů, odevzdávání úkolů a prací podporuje i tvorbu testů.

Ze strany studentů jsou tyto testy relativně uživatelsky přívětivé. Velkým problémem je ale nestabilita, kdy aplikace nezvládá mít spuštěný test pro 30 a více studentů. Žáci pak nestíhají test dokončit v daném čase. Jedná se o nepříjemnou situaci jak pro učitele, tak žáka.

Ze strany učitele vypadá rozhraní méně přívětivě než například Microsoft Teams. Nabízí ale více možností hodnocení, které je však také omezené tím, že například nelze udělat otázku, která by neměla správnou odpověď. Moodle hodnotí testy podle vah, kterou tvůrce nastaví při tvorbě testu a ta se musí vždycky rovnat 100 procentům, ať už je to jedna otázka nebo více otázek. Tato implementace mi také přijde poměrně komplikovaná. Moodle však podporuje generování testů s různými otázkami pro každého studenta. Ukázka rozhraní z návodu na tvorbu testu je na obrázku 4.3.



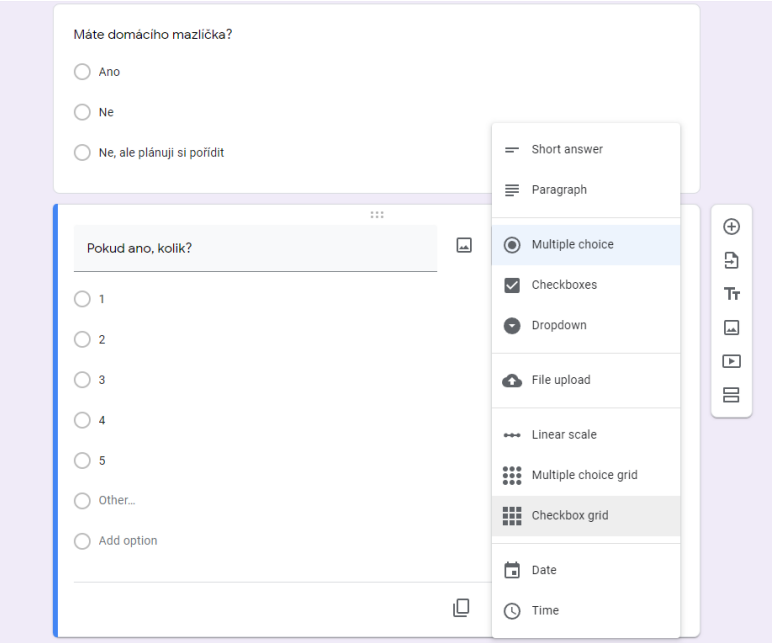
Obrázek 4.3: [5]Příklad tvorby otázky v Moodle

4.4 Google forms

Google forms je software pro tvorbu dotazníků. Používá se však také pro tvorbu testů. Obsahuje pouze tři typy otázek, a to výběr jedné nebo více odpovědí a otázku s otevřenou odpovědí.

Hodnocení je také komplikované, protože dotazníkový software není připraven na to, že někdo bude „dotazník“ hodnotit, proto je tento systém nevhodný k tvorbě testů.

Pozitivní je určitě rychlost a uživatelská přívětivost rozhraní. Také lze aplikaci snadno kombinovat s ostatními google nástroji. Ukázka viz 4.4



Obrázek 4.4: Ukázka uživatelského rozhraní Google Forms, převzato z [6]

Část II

Návrh

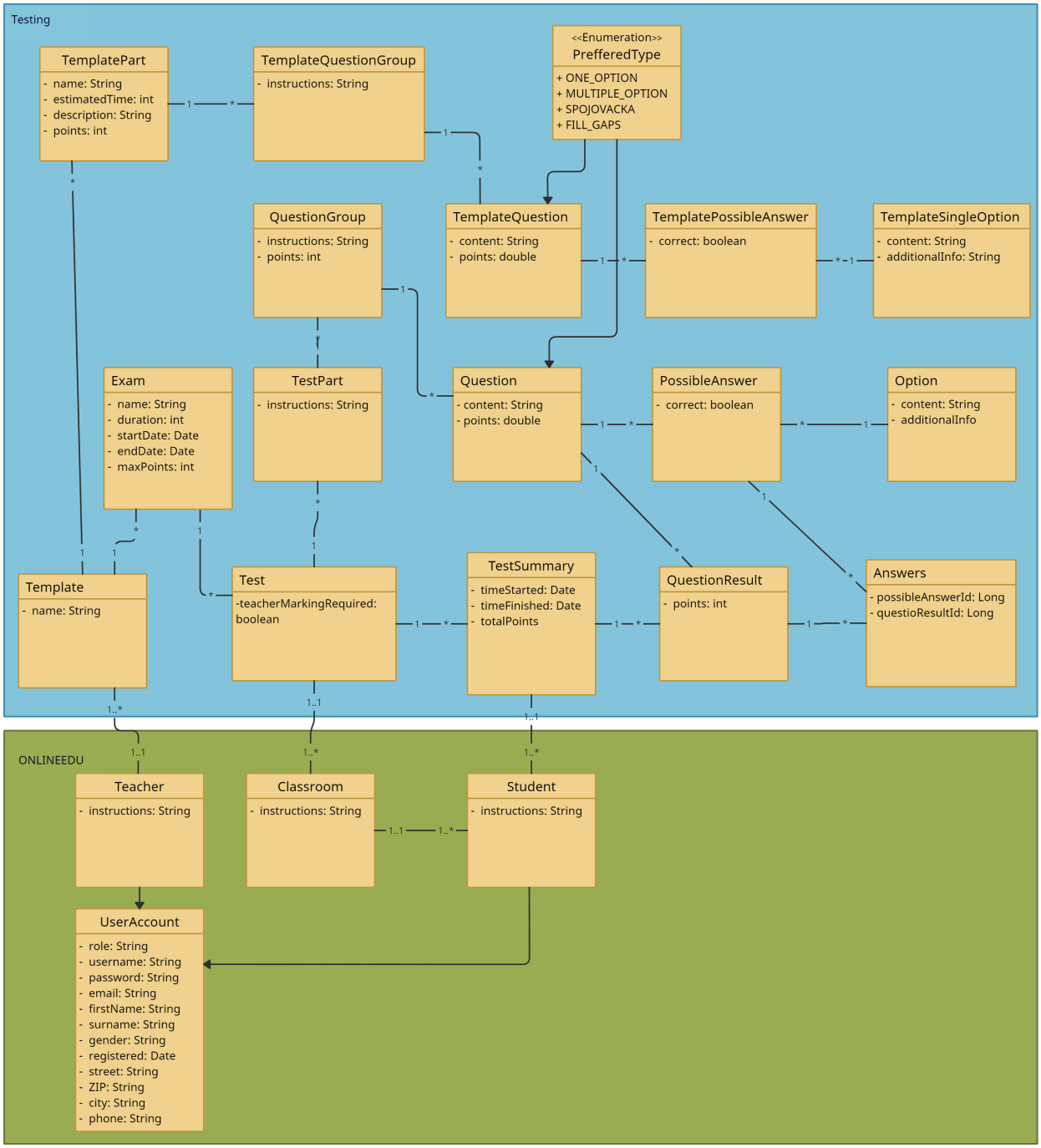
Kapitola 5

Diagram tříd

Diagram tříd je základ backendu aplikace. Obecně se jedná o entity v systému a vztahy mezi nimi. Diagram byl vytvořen v jazyce UML.

Zásadní je rozdělení na šablonové a testové entity. Šablonové entity, tedy s předponou Template, jsou entity, jejichž instance vzniknou při tvorbě šablony testu. Instance testových entit vznikají při generování testu. Ze šablonových entit si vezmou data, jinak ale nejsou nijak propojeny. Tento způsob vytváření testů/dotazníků vznikl po několika různých iteracích, více na toto téma v kapitole 10.

Další důležitou částí je struktura QuestionGroup, Question, PossibleAnswer a Option. Tato struktura byla vytvořena proto, aby bylo možné ji použít na co nejvíce typů otázek. Pokud chceme vytvořit jednoduchou otázku s jednou správnou odpovědí, je na QuestionGroup navázána jenom jedna Question, která obsahuje už samotnou otázku. V entitě Option se pak nachází odpovědi a entita PossibleAnswer obsahuje pouze informaci, jestli je daná odpověď správně nebo špatně. Struktura PossibleAnswer – Option byla vytvořena z toho důvodu, aby na frontend nebyla posílána možnost s vlastností, která by určovala, jestli je dobře nebo špatně, protože tato informace by mohla být žáky zneužita. Díky entitě PossibleAnswer se také lépe tvoří otázky, kde je více správných odpovědí. Bylo ale cílem podporovat i složitější otázky, například doplňování pojmů. V tomto případě je na QuestionGroup navázáno vetší množství podotázek, entit Question. Všechny mají stejnou množinu možností, ale každá má v PossibleAnswer uloženo, jaká odpověď je správně.



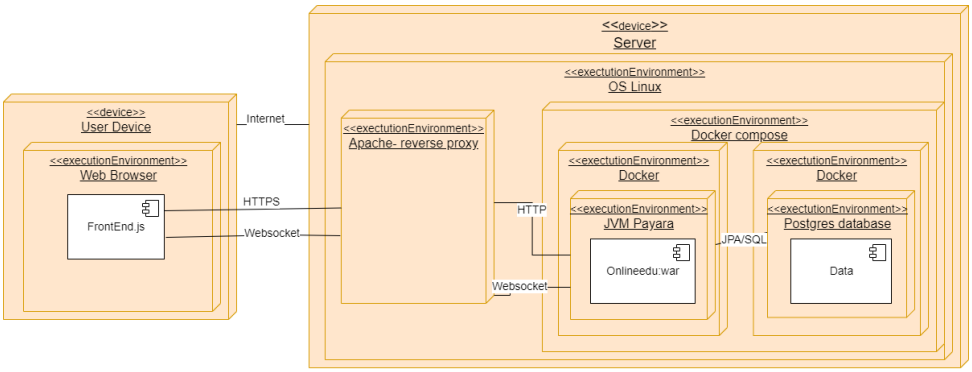
Obrázek 5.1: Diagram tříd

Kapitola 6

Diagram nasazení

Diagram zobrazuje komponenty v nasazeném systému a protokoly pomocí kterých spolu jednotlivé části komunikují.

Serverová část je založena na Docker Compose[1], který obaluje Java Virtual Machine od Payary a Posgres databázi, kde jsou uložena data. Komunikuje přes HTTP protokol s Apache – reverse proxy, která zpracovává požadavky klientů. Požadavek sama zpracuje, nebo ho předá dál. Proxy je už přes protokol HTTPS propojena s klienty komunikující přes webové rozhraní.



Obrázek 6.1: Diagram nasazení

[1] https://docs.docker.com/compose/

6.1 Nástroje

Pro tvorbu diagramu tříd byl využit online nástroj Creately[2]. Diagram nasazení byl vytvořen opět v nástroji Draw.io[3] který kromě podpory tvorby datového modelu nabízí i integraci s aplikací Google Drive[4], což je velmi užitečné pro organizaci a sdílení diagramů vytvořených v tomto nástroji.

[2] https://creately.com/
[3] https://drawio-app.com/
[4] https://drive.google.com/

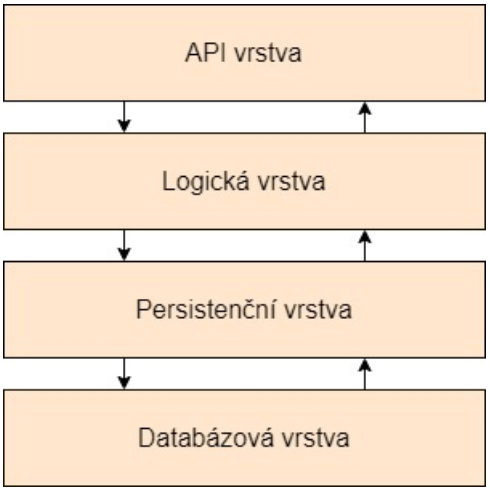
Část III

Implementace

Kapitola 7

Architektura

Aplikace používá rozdělení na čtyři horizontální vrstvy. Je to API[1] vrstva, logická vrstva, persistenční vrstva a databáze. API vrstva zpracovává HTTP požadavky a odpovědi na ně. Logická vrstva řeší businessovou logiku aplikace. Persistenční vrstva získává a zapisuje data do databáze, ve které jsou data uložena.



Obrázek 7.1: Architektura využívající 4 vrstev

[1] Application Programming Interface

7.1 Použité technologie

Technologie použité pro vývoj aplikace byly z velké většiny dány povahou zadání. Bylo definováno, že backend musí být vyvíjen v Javě, konkrétně v Jakartě EE. Vzhledem k tomu, že bylo potřeba integrovat aplikaci s existujícím řešením, viz 7.1.1, byla zde i nutnost použití PostgreSQL, Payara serveru a Dockeru.

7.1.1 Integrace s existujícím řešením

Práci jsme nevypracovávali celou od začátku, ale po konzultaci s vedoucím jsme se rozhodli využít zejména databázi z práce kolegy Daniela Kotena, Softwarová podpora pro vzdálenou výuku. [1]

K tomuto rozhodnutí jsme přistoupili, abychom nemuseli implementovat celý backend, zejména strukturu učitelů, žáků a tříd, a mohli se věnovat funkcím, které naší aplikaci pro tvorbu testů posunou dále jak z hlediska samotných funkcionalit, tak z hlediska uživatelské přívětivosti.

V úvodu kapitoly jsem zmínil, že tato aplikace má 4 vrstvy. API vrstva v našem případě nahrazuje vrstvu, kterou má Bc. Koten zastoupenou technologií JSF[2], která komunikuje přímo s logickou vrstvou, bez potřeby RESTful API. V našem případě je frontend aplikace samostatně a potřebuje tedy toto API, aby mohl přes požadavky komunikovat s logickou vrstvou.

7.1.2 Jakarta EE

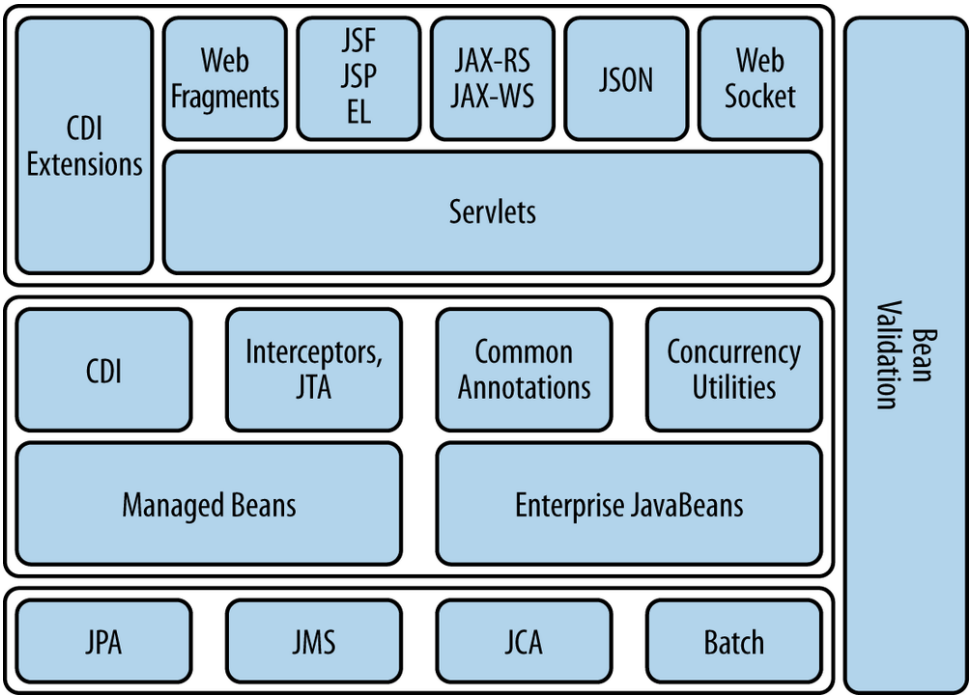
Jakarta EE je komerční platforma, která poskytuje komponenty a API pro tvorbu Java business aplikací [7]. Obecně se jedná o platformu, která umožňuje za pomoci rozdělení na vrstvy a specifikací vytvářet velké enterprise aplikace.

Tato platforma má různé specifikace pro různé účely. Zde jsou ty, které používám v práci.

[2] Java Server Faces

- Web service specifikace
  - obsahuje Jakarta RESTful Web Services, které poskytují podporu pro tvorbu webových servis s paternem REST
- Enterprise specifikace
  - Jakarta Context and Dependency injection poskytuje dependency injection kontejner
  - Jakarta Enterprise Beans
  - Jakarta Persistence, která poskytuje specifikace o objektově-relačním mapování Java tříd na databázové tabulky
  - Jakarta Transactions

Všechny specifikace a jejich rozdělení je vidět na obrázku 7.1.2.



Obrázek 7.2: Jakarta EE. Převzato z [8]

7.1.3 PostgreSQL

PostgreSQL je výkonný objektově relační databázový systém s otevřeným zdrojovým kódem, který využívá a rozšiřuje jazyk SQL v kombinaci s mnoha funkcemi, které bezpečně ukládají a škálují data. [9]

PostgreSQL poskytuje mnoho prvků, které vývojářům pomáhají vytvářet aplikace, například primitivní datové typy (integer, boolean) ale i strukturované datové typy jako Range/Multirange a samozřejmě datum a čas. Také umožňuje práci s geometrickými daty.

V této práci navazujeme na databázi bc. Daniela Kotena, který zvolil pro svojí práci PostgreSQL zejména z důvodu podpory ze strany Jakarty EE a možnosti ORM, kdy si uživatel vytvoří databázi v SQL a následně se mu vygenerují Java třídy.

Já jsem pro třídy v naší části práce volil metodu přes JPA, kdy jsem si nejdříve napsal třídy v Javě a poté z nich generoval tabulku do databáze.

7.1.4 Payara Server

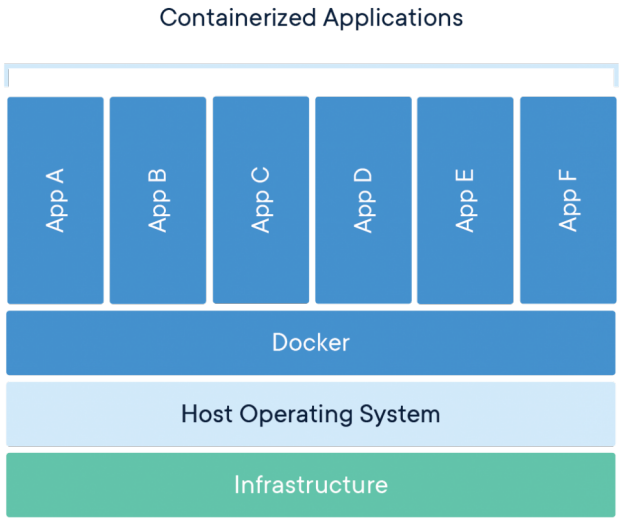
Payara Server Community [10] je nativní cloudová platforma middlewaru pro vývojové projekty, která podporuje aplikace Jakarta EE v jakémkoli prostředí: on–premise, v cloudu nebo hybridní. Vychází z GlassFish Server Open Source Edition. Jedná se taktéž o open source software, který je zdarma, s dobrou podporou a dodržovanými standardy.

7.1.5 Docker

Docker je typ kontejneru, který izoluje procesy. Využívá kernel[3] hostujícího operačního systému. Výhodou je, že potřebuje paměť a výpočetní výkon pouze pro aplikace, které na něm běží. Struktura takového kontejneru je dobře vidět na obrázku 5.2.

Další výhodou je bezpečnost, protože aplikace je uzavřená v kontejneru a neběží lokálně na počítači. V naší implementaci v Dockeru běží dvě aplikace a to Payara server a databáze PostgreSQL.

[3] Jádro operačního systému



Obrázek 7.3: Struktura Dockeru

7.2 Nasazení

Aplikace je nasazena na webové adrese https://online-edu.aubrecht.net/ a restové rozhraní se nachází na adrese https://online-edu.aubrecht.net/rest/ping.

7.3 Komunikace s uživatelským rozhraním

Uživatelské rozhraní je v případě této aplikace oddělené. Pracuje na něm kolega Geržičák a je psané v ReactJs [11]. Toto rozhraní komunikuje s backendem aplikace přes RESTful api, což je architektonický styl pro aplikaci s API, které využívá HTTP požadavky pro přístup a získání dat. Na úpravu nebo čtení těchto dat lze použít requesty GET, PUT, POST a DELETE, které se týkají operací čtení, aktualizace, vytváření a mazání dat. Komunikace v praxi probíhá tak, že uživatel pošle z rozhraní aplikace například požadavek na vytvoření nového testu, v tomto případě POST request, který zavolá příslušnou funkci na backendu a ta zavolá další vrstvu aplikace, která už test vytvoří a poté zapíše do databáze.

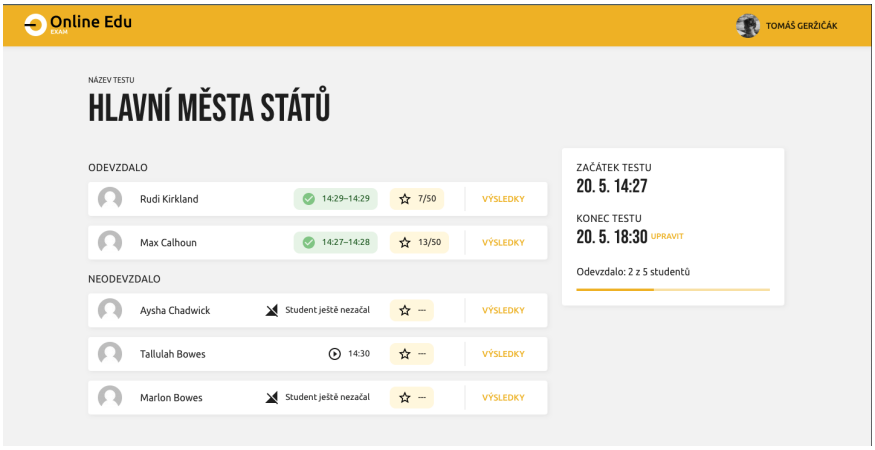
Samotná data jsou v podobně JSON objektů. Pro předávání dat využíváme Data Transfer Objects. Obecně se jedná o objekty, které v sobě většinou nemají žádnou logiku, pouze se do nich uloží informace, kterou je potřeba předat. Díky tomu je možné vytvářet různé reprezentace stejných dat optimalizovaných pro potřeby klienta. V praxi aplikace informace z databáze uloží do objektu ReadDTO a ten pošle na frontend. V případě složitějších objektů (například testová šablona) se jedná o několik objektů ReadDTO. Ukázka takového objektu na obrázku 7.4. Je vidět, jak na sebe jednotlivé objekty ReadDto navazují a tvoří velký JSON objekt který je pak zpracován a zobrazen na frontendu aplikace. Při požadavku, který posílá data do aplikace, se informace propíší do objektu typu SaveDTO. S těmito daty už pak dále pracuje logická část aplikace.


```
{
    "authorId": 1,
    "id": 1,
    "name": "Zemepis",
    "parts": [
        {
            "description": "Zeměpis",
            "estimatedTime": 5,
            "id": 1,
            "name": "Řeky",
            "points": 10,
            "preferredType": "ONE_OPTION",
            "questionGroups": [
                {
                    "groupType": "SIMPLE",
                    "id": 3,
                    "instructions": "Nejdelší řeka v ČR?",
                    "numberOfGeneratedQuestions": 0,
                    "partId": 1,
                    "questions": [
                        {
                            "answerType": "ONE_OPTION",
                            "id": 4,
                            "points": 5.0,
                            "possibleAnswers": [
                                {
                                    "id": 10,
                                    "singleOption": {
                                        "additionalInfo": "",
                                        "id": 54,
                                        "text": "Vltava"
                                    }
                                },
                                {
                                    "id": 12,
                                    "singleOption": {
                                        "additionalInfo": "",
                                        "id": 56,
                                        "text": "Morava"
                                    }
                                },
```


Obrázek 7.4: Ukázka testové šablony a jejích částí

7.3.1 Websocket

V průběhu implementace vznikla potřeba posílat ze serveru na klienta informaci o tom, v jakém stavu jsou testy jednotlivých žáků. Server posílá informaci, jestli už žák test začal vypracovávat a ve chvíli, kdy test ukončí, se asynchronně změní stav testu na straně klienta. Pro lepší představu přikládám ukázku uživatelského rozhraní na obrázku 7.5.



Obrázek 7.5: Ukázka uživatelského rozhraní websocketu na straně klienta

Pro implementaci websocketu bylo zvoleno JSR 356, Java API pro websocket. Poskytuje bi-directional, full-duplex, real-time client/server komunikaci [12].

7.4 Struktura složek projektu

Třídy jsou rozděleny do složek podle zavedených konvencí. Projekt je primárně rozdělen do čtyřech složek a to entity, service, dao a controller. Vzhledem k tomu, že jsme využívali velké množství DTO tříd, jak na ukládání, tak na čtení dat, obsahuje repozitář obsáhlou složku právě pro DTO. Struktura repozitáře je vidět na následujícím digramu 7.4.

```
src
└── main
    └── java
        └── cz.cvut
            └── fel.tests
                ├── controller ..................................API
                ├── dao ..........................Persistenční vrstva
                ├── dto .........................Data Transfer Objects
                ├── entity
                ├── exceptions.....................Vlastní výjimky
                ├── fasade
                ├── filters.........Filtry používané pro autorizaci
                ├── security ........Třídy zajišťující autentizaci a
                │   autorizaci
                └── service............................Logická vrstva
└── test
```

7.5 Generování testů

Samotná tvorba/generování specifického testu probíhá pro každého žáka podle třídy, kterou učitel při generování vybere. Randomizace variant probíhá tak, že se do každé části testu náhodně z každé části šablony vybere otázka nebo skupina otázek. Učitel tak může otázky dobře rozdělit do skupin podobného stylu a náročnosti, aby každý student měl typově stejnou otázku, pouze s jiným zadáním a odpověďmi. Tato funkce ušetří učitelům spoustu času oproti manuální tvorbě testů. Celý případ užití je také velmi intuitivní, učitel pouze vybere třídu, šablonu a klikne na „zadat test“.

7.6 Automatické hodnocení testů

Pro typy otázek, kde je to možné, dojde při ukončení testu k automatickému vyhodnocení. V tomto případě bylo potřeba počítat se strukturou otázek a podotázek. V případě, že je v jedné skupině více otázek, rozdělí se počet bodů rovnoměrně mezi otázky. Analogicky se tak děje s konkrétními možnostmi u typu otázek, kde je více správných odpovědí. Výsledky testu se ukládají a je zde možnost opakování testu a zpětné analýzy jak ze strany studenta, tak učitele. V současné chvíli je implementována pouze jedna možnost hodnocení otázek, aplikace nepodporuje například možnost odečítání bodů. Počítáme s touto možností při případném rozšíření aplikace.

Kapitola 8

Nástroje

8.1 Intellij IDEA

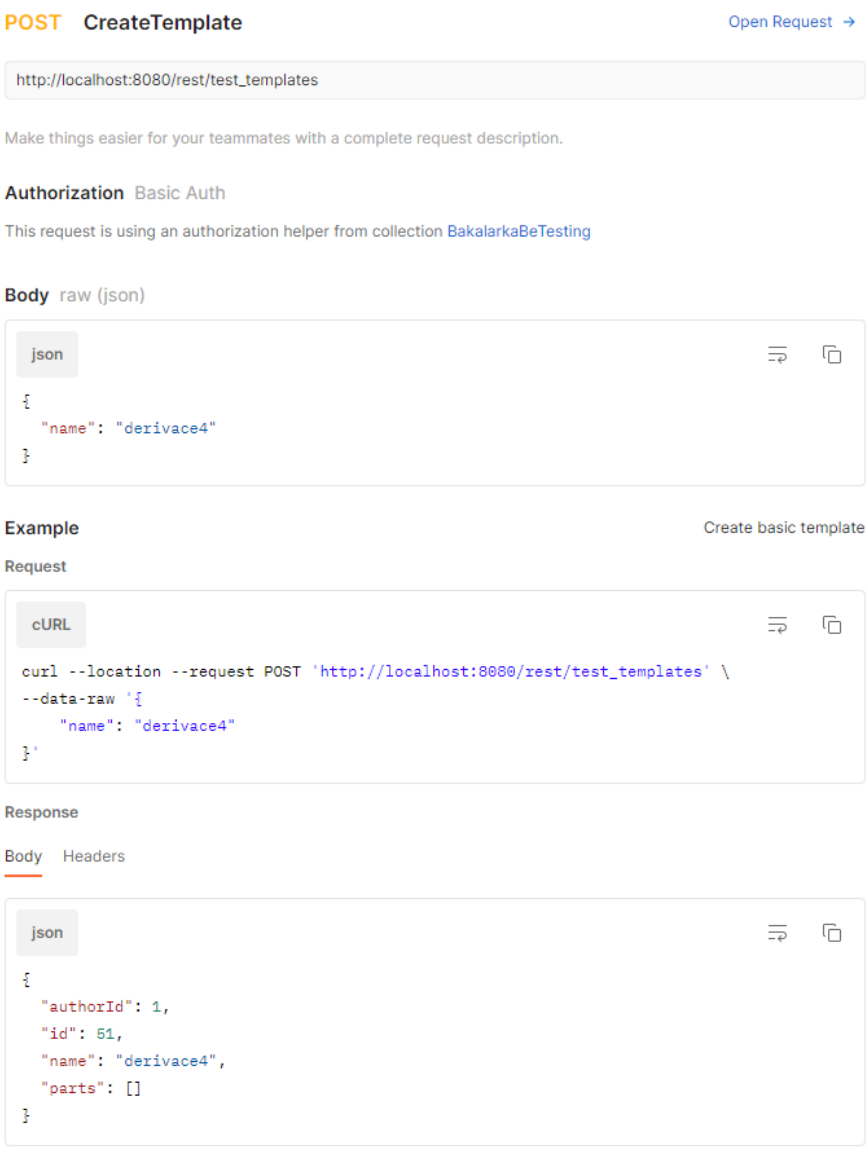
Vývojové prostředí s licencí poskytnutou ČVUT. Jedná se produkt od české firmy JetBrains[13]. Toto prostředí poskytuje vše potřebné k této práci včetně podpory Jakarty EE a také se v něm dobře pracuje s databází.

8.2 Git

Aplikace byla vytvářena v týmu, a tím pádem byla nutnost verzovacího systému. Vzhledem k předchozím pozitivním zkušenostem byl zvolen Git. Používáme Gitlab poskytovaný FEL ČVUT. Backend i frontend mají svůj vlastní, oddělený repozitář. Git z hlediska backendu slouží jako úložiště, záloha a prostředek pro sdílení aktuální verze kódu s frontendovým vývojářem.

8.3 Postman

Postman je API (Application Programming Interface) platforma pro vytváření a používání těchto rozhraní [14]. Lze v něm vytvářet HTTP dotazy pro testování RESTful rozhraní. Ty pak lze dále přehledně organizovat a především sdílet. V neposlední řadě Postman také umožňuje generování dokumentace. Viz obrázek 8.1.



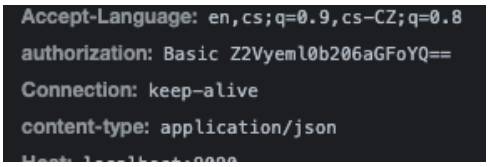
Obrázek 8.1: Ukázka dokumentace vygenerované aplikací Postman

Kapitola 9

Bezpečnost

9.1 Přihlášení

Pro autentizaci používáme v tuto chvíli basic authentication. Server vyzve klienta aby v požadavku poslal i autentizační informaci. Většinou se jedná o uživatelské jméno a heslo v autorizační hlavičce v base-64 kódování. [15] Ukázka na obrázku 9.1. V kolonce authorization je zahashované jméno a heslo oddělené dvojtečkou.



Obrázek 9.1: Basic auth v hlavičce

9.2 SQL Injection

SQL injection je typ útoku, kdy je útočníkovi umožněno přímo ovlivnit SQL dotazy, které aplikace posílá do databáze [16].

Aplikace není na tento typ útoku náchylná, protože je designovaná takovým způsobem, že nikdy nepřejímá argument, který by rovnou poslala do databáze, tím pádem uživatel nemůže například použít SQL příkaz DROP TABLE ke smazání dat. V aplikaci jsou v případě, že uživatel zadává textový input použity named queries.

9.3 Cross-site scripting

Cross-site scripting(XSS) je typ útoku, kdy uživatel vytvoří vstup do aplikace tak, aby mohl číst nebo modifikovat data ostatních uživatelů [17]. Konkrétněji do textového pole vloží script, tedy v javascriptu napsaný příkaz na straně klienta. Tento kód je pak vložen do webové stránky běžného uživatele, vypadá ale důvěryhodně. Pomocí XSS lze například měnit vzhled stránky, a to využít k získání údajů o uživateli. Tento útok probíhá na klientské části aplikace a tam je i ošetřen. Server pak už žádnou kontrolu neprovádí.

9.4 Cross-site request forgery

Cross-site request forgery, dale jen CSRF je typ útoku, kdy útočník oklame již přihlášeného uživatele k tomu, aby udělal to, co po něm útočník chce.[18] Útočník vytvoří falešný odkaz a pak už stačí, aby na něj uživatel klikl. Většinou pošle oběti například odkaz na obrázek a ve chvíli, kdy na něj uživatel klikne, vykoná se upravený příkaz.

Tento typ útoku se soustředí na provádění změn v aplikaci, protože útočník nedostává odpovědi ze serveru, obět ano. Může například změnit email oběti útoku.

V tuto chvíli aplikace nevyužívá autorizační token, tedy není schopna rozlišit mezi CSRF požadavkem a požadavkem pocházejícím přímo z vlastní stránky.

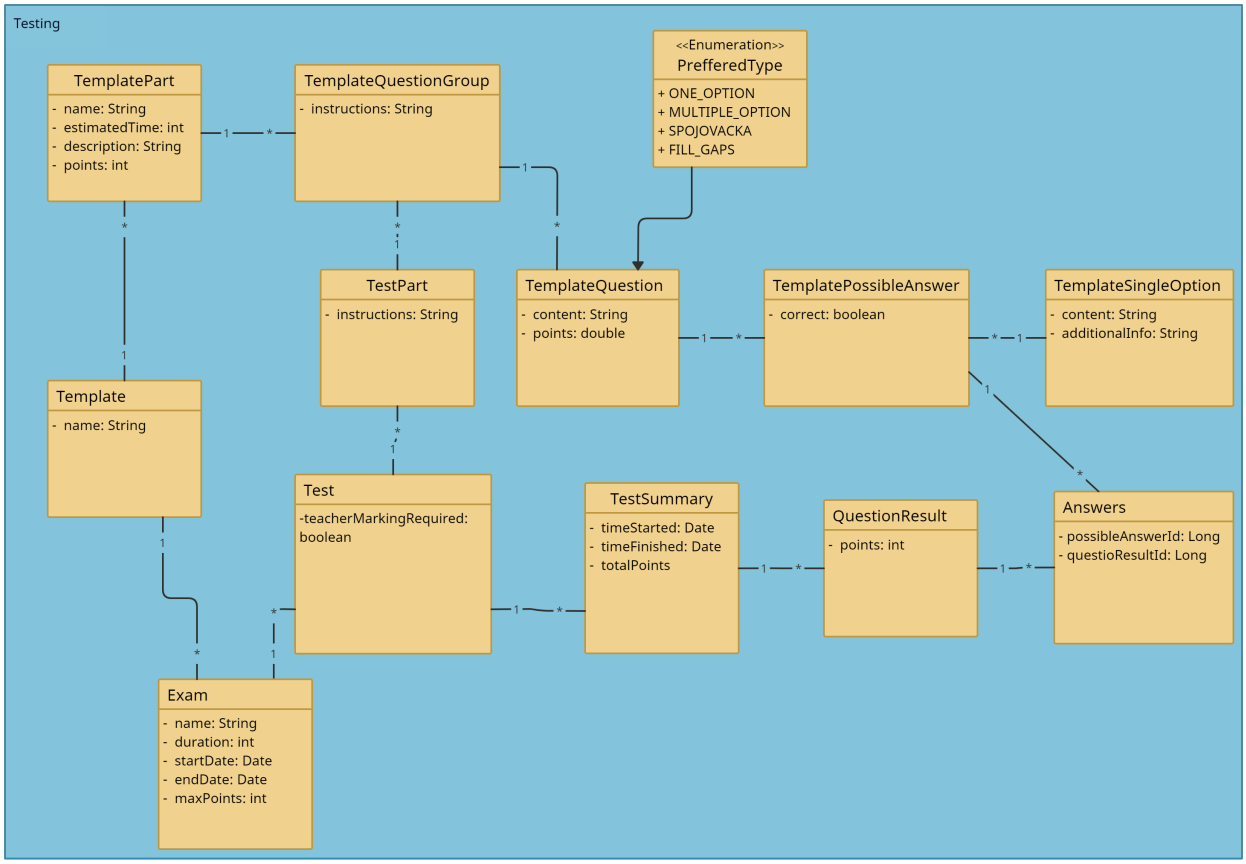
Kapitola 10

Model

Model, graficky zobrazený na obrázku 5.1, byl vytvořen pomocí JPA. Tabulky byly podle potřeby buď vytvářeny pomocí anotace create, popřípadě pokud byla třeba tabulky upravit, byla používána anotace drop-and-create, která smaže všechny data v databázi a vytvoří tabulky znovu.

10.1 Problémy – revize

V průběhu implementace se objevil problém, který ale vznikl už v návrhu. Na obrázku 10.2 je pomocí UML notace znázorněno, jak model vypadal původně. Při generování testu se místo tvorby nových dat, tedy částí testu, otázek, odpovědí atd. pouze vytvořili nové části testu. Zbytek entit zůstal původní, data byla ze šablony a byly pouze vytvořeny nové vazby. Při implementaci update a delete jsme si uvědomili, že pokud bude chtít učitel po vygenerování jedné sady testů měnit šablonu, změní i už vygenerované testy. Vzhledem k tomu, že chceme, aby žáci měli zpětně k testům přístup, došlo k přepracování modelu tak, jak je vidět na obrázku 5.1. Při generování testů pro jednotlivé žáky se vytvoří pro každý test nová sada otázek a odpovědí založených na testových entitách, které pouze převezmou své vlastnosti od šablonových entit.



Obrázek 10.2: Původní diagram tříd

Kapitola 11

Plány na rozšíření

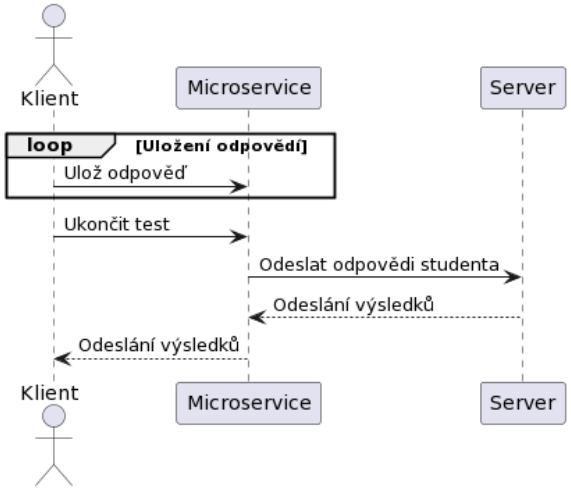
11.1 Microservices

Dlouhodobě by bylo výhodné rozdělit aplikaci na mikroservisy. Rozdělení aplikace by umožnilo podporu velkého množství uživatelů v případě, že by aplikaci používalo velké množství škol. Také by zjednodušilo pozdější vývoj a rozšiřování aplikace. Původní server by tím pádem měl na starosti pouze správu uživatelů, tříd a předmětů. Na mikroservisy bych použil řešení od Payary, Payara Micro[1].

11.1.1 Služba pro ukládání otázek

Služba pro ukládání otázek by běžela na jiném serveru než zbytek aplikace, tím pádem by byl výpočetní výkon serveru využíván pouze na ukládaní odpovědí. Zabránilo by se tak přetížení a následnému pádu serveru v případě, že testy v jednu chvíli vyplňuje velké množství studentů, jak se tomu děje například u aplikace Moodle. Jak by služba spolupracovala s klientem a se serverem je vidět na sekvenčním digramu 11.1.

[1] https://www.payara.fish/products/payara-micro/



Obrázek 11.1: Sekvenční diagram služby pro ukládání otázek.

11.2 Další typy otázek

Jak bylo zmíněno v úvodu, je mezi učiteli zájem o složitější otázky, které budou interaktivní pro studenty a rozšíří jim možnosti vytváření testů. Povedlo se nám implementovat otázku typu „spojovačka“, kde se spojují pojmy k sobě a poté otázku, kde student doplňuje nabízená slova do mezer v textu. Typů otázek, které vidíme dlouhodobě jako zajímavé je však více.

11.2.1 Příběhové otázky

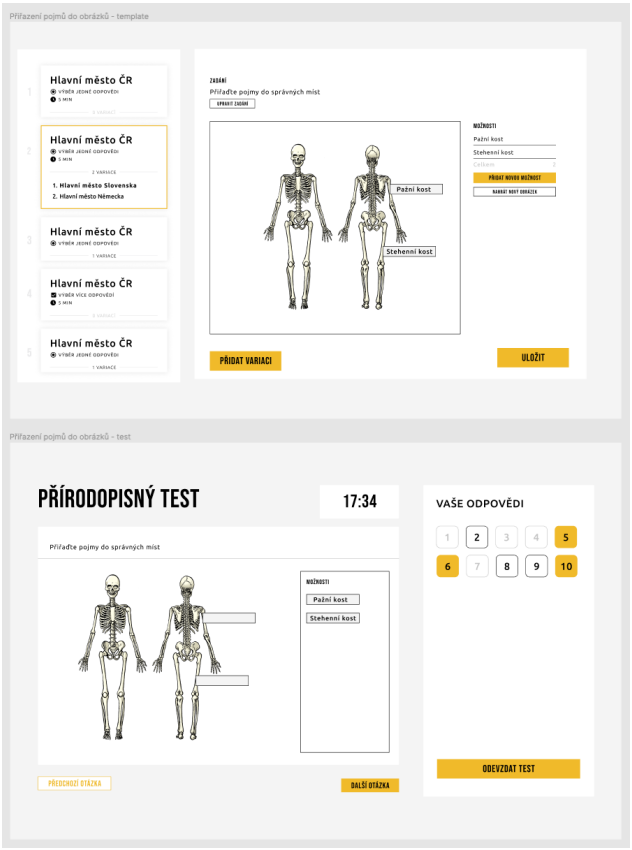
Otázka příběhového typu by měnila navazující otázku, podle toho, jak by student odpověděl na otázku původní. Z hlediska implementace by se jednalo především o návrh grafického rozhraní tak, aby bylo pro učitele možné a stále v rámci možností i uživatelsky přívětivé takovou otázku vytvořit. Z hlediska backendu by pak šlo o vyřešení logiky vybírání další otázky podle výběru s tím, že různých možností by pravděpodobně bylo velké množství. Bylo by tedy potřeba ošetřit okrajové případy a implementovat funkcionalitu tak, aby fungovala při všech kombinacích. Pravděpodobně by byla řešena pomocí informace Následující otázka, který by se ukládala do konkrétní odpovědi. Bohužel v současnou chvíli na to není backend ani frontend připraven.

11.2.2 Poslechová cvičení

Poslechová cvičení jsou velmi užitečná například při testování jazykových schopností studenta. Nabízejí také větší míru interaktivity oproti klasickým otázkám. Z hlediska backendu v současné chvíli chybí podpora ukládání audio dat.

11.2.3 Obrázkové otázky

Otázky, které využívají obrázky například na doplnění pojmů k patřičným částem obrázku. Zde by v oblasti backendu stačilo pouze ukládat obrázky a načítat je z databáze. Co se týká polohy „správné odpovědi“, databáze je na to připravena. Pouze by se do entity Option uložily souřadnice, kde se nachází správná pozice objektu doplňovaného do obrázku. Správných souřadnic by mohlo být i více, podobně jako u multiple choice otázek. Pro lepší představu přikládám návrh uživatelské rozhraní 11.2 vypracovaný kolegou Geržičákem.



Obrázek 11.2: Obrázková otázka, příklad UI

11.2.4 Korektura textu

Dlouhodobě bychom chtěli, aby aplikace podporovala i tento typ otázky. Na backendu by se pravděpodobně pro tento typ otázky musela udělat zvlášť struktura. Odpověď na otázku by měla samostatnou entitu, ve které by byl uložen text odpovědi. Funkce na automatickou opravu této otázky by pak prošla text odpovědi a strhávala by body v případě, že by našla rozdíl oproti správnému textu. Potenciální problém vidím v případě, že by student někde nechal dvojitou mezeru což by funkce spočítala jako chybu.

11.2.5 Vylepšení zabezpečení aplikace

V tuto chvíli používá aplikace basic authentication. Dlouhodobě by bylo ideální přejít na OAuth 2.0[2]. OAuth 2 umožňuje autentizaci uživatele, aniž by kdykoliv sdílel jeho údaje. Využívá k tomu takzvaný acess token, který reprezentuje autorizaci uživatele přistupovat k jednotlivým zdrojům. OAuth také není sám o sobě odolný proti CSRF, lze ho však upravit tak, aby byl díky parametru state. Tento parametr je nastavován na straně serveru a uložen na straně verifikovaného klienta. Při posílání požadavku dojde ke kontrole a pokud nejsou parametry stejné, lze předpokládat, že se jedná o útok.

[2] https://oauth.net/2/

Kapitola 12

Další poznatky z práce

12.1 Bakalářská práce ve dvou lidech

K práci ve dvou se přistoupilo, abychom zvládli zanalyzovat a implementovat software o větším rozsahu. Bohužel komunikace ve většině případů probíhala hlavně online, tak jak jsme byli zvyklí z dob koronavirové pandemie.

Z toho důvodu došlo k několika zásadním nedorozuměním, kdy jsme každý pochopili určitou funkcionalitu úplně jinak. Vzhledem k tomu, že práce byla rozdělena na backend a frontend, které jsou od sebe oddělené, docházelo pak k nekompatibilitě řešení a nutnosti funkcionalitu předělávat nebo implementovat znovu.

Až ke konci práce jsme začali pravidelně pracovat společně, na jednom místě. Kvalitu práce i naši produktivitu to výrazně pozvedlo. Ve chvíli, kdy bylo například třeba změnit strukturu dat posílaných z backendu na frontend, byla to otázka několika minut místo hodin, jako v případě online komunikace.

12.2 Problémy s analýzou

Problémům zmíněným v předchozí části by se také ale dalo zabránit podrobnější analýzou. Při návrhu se nám často nepovedlo dané řešení připravit na všechny případy užití a až při implementaci jsme zjistili, že navržené řešení některé případy nepodporuje.

Také bylo na místě udělat více rozhovorů s učiteli. Opět by nám to lépe pomohlo definovat požadavky a případy užití.

Problémům s modelem, zmíněných v části 10.1, by se také dalo zabránit lepším návrhem diagramu tříd. Při jeho tvorbě jsme se soustředili především na strukturu a s ní spojené generování otázek a odpovědí. Bohužel ostatním požadavkům nebyla věnována taková pozornost.

Kapitola 13

Testování

V průběhu implementace jsem aplikaci testoval především manuálně, pomocí aplikace Postman, viz 8.3. Postupem času začalo ale toto řešení být časově poměrně náročné. Bylo nutné pořád dokola opakovat ty samé požadavky a každý pouštět manuálně, zvlášť.

Proto jsem přistoupil na testování pomocí Collections[1]. Jedná se například o skupinu požadavků, která testuje tvorbu šablony a následné vytvoření testu. Pomocí řetězení požadavků vždy předám informace z JSON objektu do dalšího požadavku, který už s nimi dále pracuje. U každého požadavku také používám jednoduchý test, který mi v případě chyby usnadní její hledání.

```
var jsonData = pm.response.json();

pm.environment.set("templateId", jsonData.id);

pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
```

Listing 13.1: Ukázka získávání proměnné z response body požadavku

[1] https://www.postman.com/collection/

Kapitola 14

Závěr

Zadáním práce bylo vytvořit software pro tvorbu a generování školních testů, který bude alternativou k aplikacím jako Moodle nebo Google Forms. Hlavními požadavky byla podpora základních typů testových otázek, uživatelská přívětivost, stabilita a zaměření na použití zejména na základních a středních školách. Toto zadání se nám z většiny povedlo splnit. Máme první verzi aplikace s kvalitními základy, ze kterých je možné ji dále rozvíjet.

V první fázi práce jsme se soustředili především na analýzu. Věnovali jsme se struktuře otázek a odpovědí, která musela podporovat více podotázek v jedné otázce a více typů otázek, abychom nemuseli mít zvláštní entitu pro každou otázku. Dalším požadavkem bylo zobrazení výsledků žáků pro každou konkrétní podotázku. Bylo také zásadní vymyslet a správně implementovat strukturu pro tvorbu šablon a následně z nich generování samotných školních testů. I tak ale návrh struktury databáze procházel s téměř každou další funkcionalitou aplikace revizí tak, aby byl efektivní a dalo se na něj do budoucna dále navazovat. Nyní máme databázi strukturovanou tak, že zmíněné požadavky splňuje.

Dalším krokem byla integrace s existujícím řešením OnlineEdu, vytvořené Bc. Danielem Kotenem v rámci bakalářské práce Softwarová podpora pro vzdálenou výuku, které nám poskytlo především databázi a základní strukturu. Databáze OnlineEdu je dobře zpracovaná a nebyl problém na řešení navázat. Nakonec používáme pouze čtyři entity z původního návrhu, jak je vidět v diagramu tříd.

Při následné implementaci byly vytvořeny základní typy otázek: výběr z možností (správná jedna nebo více odpovědí) a textové pole. Dále byla realizována otázka typu „spojovačka“, kde musí student právě přiřadit pojmy k sobě a doplnění pojmů do textu. Aplikace podporuje generování testů ze šablon, kde je pro každého studenta vygenerován specifický test. Otázky, u kterých to lze, jsou po ukončení testu automaticky vyhodnoceny.

Do budoucna by šlo značně rozšířit typy otázek například o poslechová cvičení nebo práci s obrázky. Také bychom se chtěli zaměřit na stabilitu systému. Aplikace zatím nebyla testována velkým množstvím uživatelů najednou, pokud by byl se stabilitou problém, je zde možnost vytvoření mikroservisy na ukládání odpovědí. Hlavnímu serveru by se tak značně snížilo vytížení v situacích, kdy test vyplňuje velké množství studentů.

Na této práci jsem si také vyzkoušel nové technologie, především Jakarta EE a Docker. Kromě občasných problémů se získáváním informací k Jakartě EE hodnotím práci s těmito technologiemi pozitivně.

Přílohy

Příloha A

Bibliografie

[1] Daniel Koten. "Softwarová podpora pro vzdálenou výuku". České vysoké učení technické, 2021.

[2] Mohamad H. Kassab. Laplante Phillip A. *Requirements Engineering for Software and Systems*. Auerbach Publishers, Incorporated, 2009. ISBN: 9781000593815.

[3] KhanAcademy. *Khan academy classroom*. URL: `https://www.khanacademy.org/math/get-ready-for-3rd-grade`. (accessed: 12.01.2022).

[4] Microsoft. *Microsft teams question*. URL: `https://support.microsoft.com/en-us/office/create-a-quiz-with-microsoft-forms-a082a018-24a1-48c1-b176-4b3616cdc83d`. (accessed: 12.01.2022).

[5] Moodle. *Moodle teams question*. URL: `https://docs.moodle.org/27/en/Building_Quiz`. (accessed: 12.01.2022).

[6] Google. *Moodle teams question*. URL: `https://docs.google.com/forms`. (accessed: 12.01.2022).

[7] EnterpriseProject. *EnterpriseProject Jakarta*. URL: `https://enterprisersproject.com/article/2021/6/java-ee-and-jakarta-ee-what-it-leaders-should-know`. (accessed: 13.01.2022).

[8] Arun Gupta. *Java EE 7 Essentials*. O'Reilly Media, Inc., 2013-08-01. ISBN: 9781449370176.

[9] PostgreSQL. *PostgreSQL*. URL: `https://www.postgresql.org/about/`. (accessed: 13.01.2022).

[10] payara. *payara*. URL: `https://www.payara.fish/products/`. (accessed: 13.01.2022).

[11] reactjs. *reactjs*. URL: `https://reactjs.org/`. (accessed: 13.01.2022).

[12] Baeldung. *A Guide to the Java API for WebSocket*. URL: https://www.baeldung.com/java-websockets. (accessed: 18.05.2022).

[13] jetbrains. *jetbrains*. URL: https://www.jetbrains.com/idea/. (accessed: 29.04.2022).

[14] postman. *postman*. URL: https://www.postman.com/product/what-is-postman/. (accessed: 29.04.2022).

[15] IBM. *HTTP basic authentication*. URL: https://www.ibm.com/docs/en/cics-ts/5.4?topic=concepts-http-basic-authentication. (accessed: 05.05.2022).

[16] Justin Clarke. *SQL Injection Attacks and Defense*. Elsevier Science & Technology Books, 2012-07-27. ISBN: 9781597499736.

[17] Adam Kieyzun et al. "Automatic creation of SQL injection and cross-site scripting attacks". In: *2009 IEEE 31st international conference on software engineering*. IEEE. 2009, s. 199–209.

[18] KirstenS. *Cross Site Request Forgery (CSRF)*. URL: https://owasp.org/www-community/attacks/csrf. (accessed: 05.05.2022).

Příloha B

Instalace

Zdrojové soubory jsou dostupné na adrese `https://gitlab.com/gerzitom/bakalarka-backend`.

Pro spuštění je také potřeba JDK 11 `https://www.oracle.com/java/technologies/javase-jdk11-downloads.html`.

Také doporučuji nainstalovat Docker (spolu s Docker Compose). Pak v adresáři /docker/ pustit tyto skripty.

- 01-build-src-local-docker.sh
- 02-build-dockers.sh
- 03-run.sh

Aplikace je poté přístupná na adrese `http://localhost:8080/test_templates`.

Běžící aplikace je také nasazena na adrese `https://online-edu.aubrecht.net`.

Adresa na tvorbu testovacích šablon: `https://online-edu.aubrecht.net/rest/test_templates`.

Příloha C

Seznam zkratek

API Application Programming Interface

CRUD Create, Read, Update, Delete

CSRF Cross-site request forgery

DTO Data Transfer Object

HTTP Hypertext Transfer Protoco

HTTPS Hypertext Transfer Protocol Secure

JPA Java Persistence API

JSF Java Server Faces

JSON JavaScript Object Notation

ORM Object/Relational Mapping

SQL Structured Query Language

UML Unified Modeling Language

XSS Cross-site scripting