**STŘEDNÍ PRŮMYSLOVÁ ŠKOLA BRNO, PURKYŇOVA,
PŘÍSPĚVKOVÁ ORGANIZACE**

SPRÁVA TÝMOVÝCH PRACÍ

Lukáš Matuška
V4C

Profilová část maturitní zkoušky
MATURITNÍ PRÁCE

Brno 2021

Střední průmyslová škola Brno, Purkyňova, příspěvková organizace

ZADÁNÍ MATURITNÍ PRÁCE

obhajované před zkušební komisí

Školní rok: 2020/2021

Předmět: Soubor odborných předmětů (zaměření na informační technologie)

Studijní obor: Informační technologie (18-20-M/01)

ŠVP: Informační technologie

Žák: **Lukáš Matuška**

Třída: **V4C**

Vedoucí práce: RNDr. Lenka Hrušková

Termín zadání MP: 1. prosince 2020

Termín odevzdání MP: 22. dubna 2021

Číslo zadání, verze zadání a název práce:

PROG1e: Správa týmových prací

Zadání

Vytvořte aplikaci pro usnadnění administrace a organizace týmových prací 3. ročníků oboru Informační technologiie. Finální výrobek bude OpenSource repozitář na GitLabu s návodem na instalaci a spuštění, případná pomoc škole se spuštěním ve školní infrastruktuře. Bude obsahovat ovládání ročníků, zaměření, uživatelů (student, garant, administrátor), obsahu týmových prací.

Výstupem práce bude

- Funkční webová aplikace
- Postup instalace SW
- Uživatelská příručka
- Dokumentace a zdrojové kódy v elektronické podobě
- Dokumentace bude obsahovat záznam z testování aplikace (jednotkové testy)
- Dokumentace bude obsahovat odkazy a přístupové údaje k aplikaci, DB, Gitu, apod.
- Vývoj aplikace bude možné sledovat pomocí verzovacího systému

Doporučené prostředky pro řešení, technické požadavky

- NodeJS, MongoDB, RedisDB

Součástí zadání je Příloha zadání maturitní práce.

V Brně dne 1. 12. 2020

Matuška

Podpis žáka	Podpis vedoucího práce	Podpis předsedy PK

Prohlášení

Prohlašuji, že jsem maturitní práci na téma *Správa týmových prací* vypracoval pod vedením vedoucího maturitní práce samostatně za použití v práci uvedených pramenů a literatury.

Beru na vědomí, že zpráva o řešení maturitní práce a základní dokumentace k aplikaci bude uložena v elektronické podobě na intranetu SPŠ Brno, Purkyňova, příspěvková organizace.

Beru na vědomí, že bude má maturitní práce včetně zdrojových kódů uložena v knihovně SPŠ Brno, Purkyňova, příspěvková organizace, dostupná k prezenčnímu nahlédnutí. Škola zajistí, že nebude pro nikoho možné pořizovat kopie jakékoliv části práce.

Beru na vědomí, že SPŠ Brno, Purkyňova, příspěvková organizace, má právo celou moji práci použít k výukovým účelům a po mém souhlasu nevýdělečně moji práci užít ke své vnitřní potřebě.

Beru na vědomí, že pokud je součástí mojí práce jakýkoliv softwarový produkt, považují se za součást práce i zdrojové kódy, které jsou předmětem maturitní práce, případně soubory, ze kterých se práce skládá. Součástí práce není cizí ani vlastní software, který je pouze využíván za přesně definovaných podmínek, a není podstatou maturitní práce.

Lukáš Matuška

V Brně 7. května 2021 Podpis

Poděkování

Tímto bych rád poděkoval své vedoucí maturitní práce RNDr. Lence Hruškové za veškerý čas, trpělivost, ochotu a veškeré úsilí věnované mému dílu.

Anotace

Maturitní práce *Správa týmových prací* popisuje postup při vývoji webové aplikace pro zjednodušení správy týmových prací ve třetím ročníku. Webová aplikace je určena pro správce týmových prací.

Klíčová slova

API, Node.js, MongoDB, MVC

Vedoucí práce: *RNDr. Lenka Hrušková*

Obsah

Seznam obrázků

1 Teoretický úvod

1.1 Uvedení do problematiky

Na Střední průmyslové škole Brno se ve druhém pololetí třetího ročníku oboru Informační technologie každoročně konají tzv. týmové práce. Jejich účelem je naučit studenty pracovat v týmu, což je pro absolventa důležitá schopnost a zkušenost.

Obor je rozdělen na čtyři zaměření, přičemž týmy sestávají obvykle ze čtyř studentů z různých zaměření. Těmi jsou konkrétně Programování a databáze, Grafika a webdesign, Počítačové sítě, Automatizace.

1.1.1 Srovnání současného a nového řešení

V současnosti se zadávání i výběr týmových prací řeší pomocí školní elektronické pošty. Ruční vyřízení řádově stovek e-mailových zpráv je pomalá a jednotvárná činnost.

Na počátku musí pedagogický pracovník sepsat e-mail se seznamem zadání jednotlivých týmových prací a rozeslat jej studentům. Ti na něj následně odpovídají, přičemž může dojít k výběru stejného tématu několika studenty zároveň. Tato situace se musí řešit operativně a ukrojí oběma stranám mnoho času. Studenti, kteří se k danému tématu úspěšně přihlásili, jsou poté pedagogem zapsáni do lokální tabulky MS Excel, která slouží jako finální přiřazení studentů k pracím.

S lokálně uloženým souborem souvisí také fakt, že ostatní učitelé nemohou získat k aktuální verzi souboru jednoduše přístup. V případě, že je momentální stav zajímá, musejí o tabulku žádat, často i opakovaně.

Poslední často skloňovanou nevýhodou byla nutnost odevzdávat výstup týmu na školní vzdělávací portál Moodle, který je studenty považován za velmi nepřehledný a neintuitivní.

1.1.2 Cíl práce

Správa týmových prací představovala značnou administrativní zátěž. Cílem této maturitní práce je vytvoření systému pro automatizaci jak zadávání týmových prací vyučujícími, tak výběr projektu studenty. To vše pomocí v dnešní době dostupných a moderních nástrojů.

Hlavním přínosem by mělo být snížení časové náročnosti a pracovní vytíženosti pedagogických pracovníků při každoroční organizaci týmových prací.

2 Rozbor řešení

Celá webová aplikace využívá výhradně řešení server-klient. Běží na školním serveru a díky zvolenému řešení je přístupná kdykoliv z celého světa prostřednictvím webového prohlížeče. Jedinou podmínkou je dostupnost internetového připojení.

Výstup je maximálně uzpůsoben potřebám školy. Vychází z postřehů a doporučení paní učitelky Hruškové, která má organizaci týmových prací každý rok na starosti.

2.1 Backend

Serverová část aplikace je napsána v jazyce JavaScript s využitím principů objektově orientovaného programování.

O data se stará NoSQL databáze MongoDB[1].

2.2 Frontend

Jako frontendová šablona je použita Argon Dashboard[1] vydaná pod licencí MIT. Ta klade důraz na jednoduché a uživatelsky přívětivé prostředí a ovládání. Stojí na aktuálních verzích webových technologií HTML5 a CSS3.

Za účelem dosažení vyšší uživatelské přívětivostí aplikace využívá také AJAX a některé frameworky a knihovny třetích stran[2].

2.3 Dostupnost

Zpracování dat na serveru zajišťuje Node.js[2] proces běžící v PM2[3]. PM2 je správce procesů, který pomáhá spravovat a udržovat běžící aplikace dostupné 24 hodin 7 dní v týdnu.

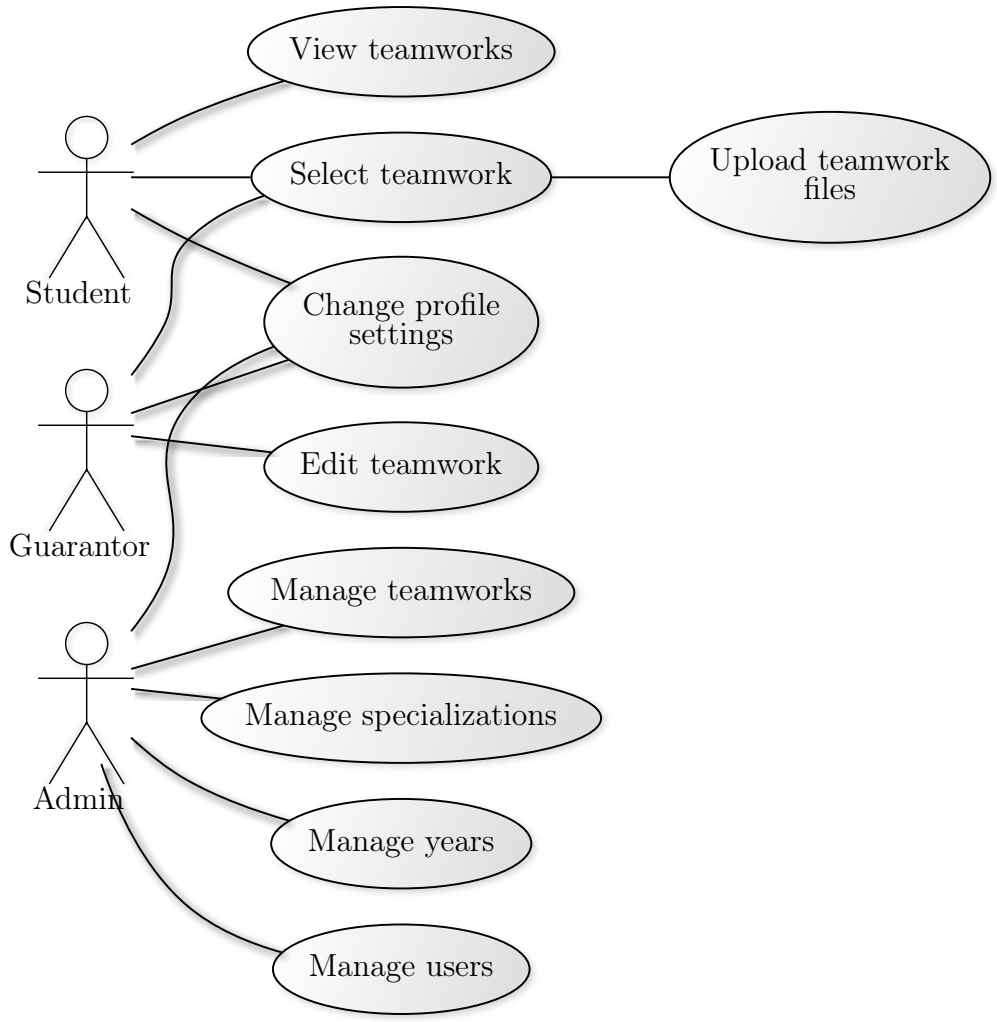
Při restartu serveru nebo pádu aplikace spouští PM2 proces znovu automaticky bez nutnosti zásahu administrátora nebo jiné osoby. Stará se také o ukládání logů. Lze specifikovat lokaci souboru s logy zvlášť pro `stdout` výstup a zvlášť pro `stderr` výstup, což v lozích usnadňuje následnou orientaci.

[1] https://www.creative-tim.com/product/argon-dashboard

[2] Moment.js, jQuery, Select2, SweetAlert2, DataTables aj.

2.4 Případy užití

Následující UML diagram znázorňuje vztah, možnosti a dostupnost funkcí aplikace jednotlivých rolí uživatelů – tzv. případy užití (use cases).



Obrázek 1: Use case UML diagram

2.5 Produkční běh ve škole

2.5.1 VMware vSphere

Projekt má na školním serveru přidělený vlastní virtuální počítač v produkčním virtualizačním prostředí VMware vSphere[4]. Ten jsem pro školu podle stanovených podmínek celý připravil, aby ho stačilo jen importovat a spustit v interní síti.

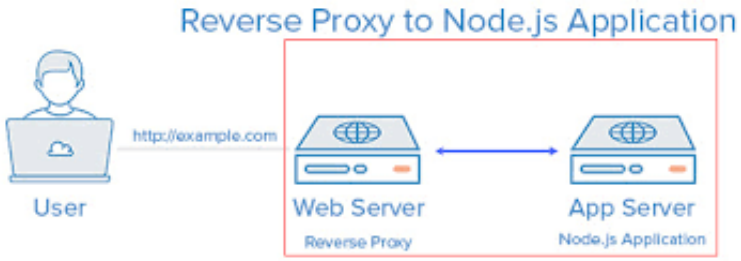
Jako operační sytém jsem zvolil open source distribuci Debian[5] založenou na kernelovém jádru Linux, protože je zdarma. Škola tak nebude muset platit za další licenci operačního systému. Více o konfiguraci webserveru se dočtete níže, v kapitole 2.5.2.

2.5.2 Nastavení webserveru

V případech hostingu podobných aplikací je běžnou praxí použití reverzní proxy. Ta zajišťuje interní nezabezpečený provoz mezi zabezpečeným webovým serverem[3] a samotnou Node.js aplikací. Přenos nemusí být zabezpečený právě z toho důvodu, že se jedná pouze o interní provoz.

Osobně jsem si oblíbil webový server nginx, který funkci reverzní proxy nativně podporuje. Bez dalších doplňků zvládá například také servírování statických souborů či provoz PHP aplikací. Konfigurační soubor pro nginx jsem připravil přímo pro potřeby této aplikace. Na administrátorovi serveru tak bylo už jen doplnění cesty ke školnímu wildcard TLS certifikátu.

Po domluvě s administrátorem byly správně nasměrovány porty 80 a 443 dostupné z internetu a nastaveny DNS záznamy – jak ve škole, tak i na veřejných DNS serverech. Správa týmových prací bude dostupná z jakéhokoliv místa na světě právě díky tomuto.



Obrázek 2: Schéma reverzní proxy a Node.js aplikace

[3] obvykle pomocí protokolu HTTPS a HTTPS/2

2.6 Testování při vývoji

2.6.1 Lokální instance

Při vývoji na lokálním stroji (vlastním notebooku) byl namísto PM2 použit `nodemon`. Ten – narozdíl od PM2, které restartuje Node.js proces při chybě – restartuje program při detekování změny některého ze sledovaných souborů (typicky zdrojových kódů aplikace).

2.6.2 Docker

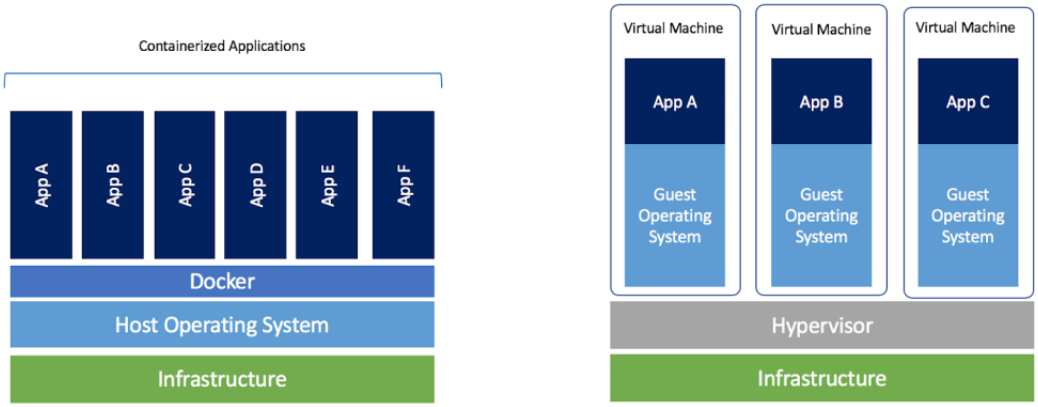
Docker[6] je open source projekt, jehož cílem je poskytnout jednotné rozhraní pro izolaci aplikací do kontejnerů. Jedná se o „odlehčenou virtualizaci“, kde kontejner obsahuje pouze soubory aplikace a pro ně specifické soubory, ale neobsahuje (virtualizovaný) operační systém. Díky tomu se výrazně snižují režijní nároky na rozdíl od standardních virtuálních strojů.

Díky svým dosavadním zkušenostem s Dockerem jsem byl pro projekt schopný vytvořit i vlastní obraz kontejneru. Ten jsem postavil nad obrazem kontejneru Node.js verze 14, kam jsem doimplementoval již zmíněného PM2 démona. Podrobnosti lze nalézt přímo v souboru `Dockerfile`.

Tento obraz jsem využil zejména při testování na budoucím produkčním serveru. Proces automatického nasazení vypadal přibližně následovně:

1. po vytvoření nového commitu v repozitáři GitLab runner běžící na virtuálním serveru sestavil nový Docker image z aktuálního zdrojového kódu
2. na serveru byl zastaven kontejner s předešlou verzí aplikace
3. byl spuštěn nový kontejner z aktuálního Docker image, a to včetně přednastavených environmentálních proměnných

Kdyby se škola v budoucnu rozhodla přejít na Kubernetes nebo Docker, může tak učinit bez nutnosti dalšího vývoje nebo speciální konfigurace.



Obrázek 3: Kontejnery vs. virtuální počítače

3 Návrh databáze

3.1 Ročníky

Rozdělení na ročníky slouží pro filtrování týmových prací a přidělování práv uživatelům. Definují také časový úsek, kdy si studenti mohou své týmové práce vybírat (a které z nich).

Důležitým parametrem je parametr `status`. Ten určuje stav konkrétního ročníku. Může nabývat tří hodnot, jimiž jsou následující:

- **active** – označuje ročník jako aktivní (právě probíhající), tuto hodnotu může vlastnit pouze jeden záznam (ročník) v databázi,
- **archived** – identifikuje archivovaný ročník (to je takový, který již proběhl a je ukončen),
- **prepared** – informuje o tom, že ročník ještě neproběhl, ale je připravený na zahájení.

3.2 Specializace studentů

Specializace studentů slouží k rozdělení studentů do jednotlivých pozic v týmových pracích podle oborového zaměření, jež si během studia zvolili. Student se zaměřením na programování a databáze si tak nemůže vybrat například pozici pro grafiku a webdesign.

Specializace, s nimiž aplikace pracuje, nejsou závislé na ročnících. Díky tomu se mohou v různých ročnících opakovat.

3.3 Uživatelé

Datový model pro uživatele pracuje s oběma předchozími modely. Se specializacemi však pouze v případě, že je uživatel studentem[4]. V takovém případě je k účtu studenta uloženo ID jeho specializace, a to do parametru `specialization`. U nestudentských účtu nabývá parametr hodnoty `null`.

3.4 Šablony týmových prací

Šablony šetří čas při vytváření nových týmových prací v aplikaci. Stejně jako zaměření nejsou závislé na ročnících, aby se mohly opakovat.

[4]uživatel je studentem, pokud je hodnota parametru `type` rovna `student`

3.5 Týmové práce

Model samotných týmových prací je závislý na modelech ročníků, specializací a především uživatelů. Každá práce má svůj název, číslo, popis a očekávaný výstup.

Aplikace také komunikuje prostřednictvím API se školním cloudovým úložištěm OwnCloud. Pro každou týmovou práci zde vytváří vlastní adresář pro následné nahrávání souborů ze strany studentů nebo vyučujících. Data týkající se cesty do daného adresáře jsou uložena v parametru `owncloud`. Nutno podotknout, že po termínu odevzdání týmových prací již studenti budou mít sdílenou složku jejich týmové práce pouze pro čtení.

Myslel jsem i na komunikaci v týmech. Konzultanti a garanti mohou ke každé práci uložit odkaz na jimi preferovanou komunikační platformu, prostřednictvím níž pak mohou se studenty řešit dále vývoj projektu.

Součástí aplikace je mj. i hodnocení týmů a jednotlivých studentů. Hodnocení má na starost garant každé práce. Hodnocení konkrétního studenta uvidí pouze tento student, garanti a konzultanti.

Přikládám pro zajímavost i ukázku kódu samotného modelu.

```
1  /**
2   * Team work database model
3   * @author Lukas Matuska (lukynmatuska@gmail.com)
4   * @version 1.2
5   */
6
7  /**
8   * Libs
9   */
10 // library for easy database manipulations
11 const mongoose = require('../libs/db')
12
13 // the schema itself
14 var teamWorkSchema = new mongoose.Schema({
15   name: {
16     type: String,
17     required: true
18   },
19   description: {
20     type: String,
21     required: true
22   },
23   number: {
```

```
24      type: Number,
25      default: 1
26    },
27    result: {
28      type: String,
29      required: true,
30    },
31    students: [{
32      user: {
33        type: mongoose.Schema.Types.ObjectId,
34        ref: 'User'
35      },
36      position: {
37        type: mongoose.Schema.Types.ObjectId,
38        ref: 'Specialization',
39        required: true
40      },
41      task: {
42        type: String,
43        required: true
44      }
45    }],
46    guarantors: [{
47      user: {
48        type: mongoose.Schema.Types.ObjectId,
49        ref: 'User',
50        required: true
51      },
52      task: {
53        type: String,
54        required: true
55      }
56    }],
57    consultants: [{
58      user: {
59        type: mongoose.Schema.Types.ObjectId,
60        ref: 'User'
61      },
62      task: String
63    }],
64    year: {
65      type: mongoose.Schema.Types.ObjectId,
66      ref: 'Year',
67      required: true
```

```
68    },
69    author: {
70      type: mongoose.Schema.Types.ObjectId,
71      ref: 'User',
72      required: true
73    },
74    media: {
75      type: Object,
76      default: {},
77      kanban: String,
78      meeting: String,
79      repositories: [String],
80    },
81    owncloud: {
82      link: String,
83      shares: {
84        students: [String],
85        consultantsAndGuarants: [String],
86      }
87    },
88    finalFeedback: {
89      type: String,
90    },
91    feedbacks: [{
92      author: {
93        type: mongoose.Schema.Types.ObjectId,
94        ref: 'User',
95        required: true
96      },
97      date: {
98        type: Date,
99        required: true
100     },
101     student: {
102       type: mongoose.Schema.Types.ObjectId,
103       ref: 'User',
104     },
105     text: {
106       type: String,
107       required: true
108     },
109   }],
110 })
111
```

```
112 // Duplicate the ID field.
113 teamWorkSchema.virtual('id').get(function () {
114   return this._id.toHexString()
115 })
116
117 teamWorkSchema.virtual('fullname').get(function () {
118   if (this.number == undefined || this.number == null || this.
        number <= 1) {
119     return this.name
120   } else {
121     return String(`${this.name} ${this.number}`)
122   }
123 })
124
125 // Ensure virtual fields are serialised.
126 teamWorkSchema.set('toJSON', {
127   virtuals: true
128 })
129
130 // export
131 module.exports = mongoose.model('TeamWork', teamWorkSchema, '
      teamwork')
```

Listing 1: MongoDB model pro týmové práce

4 Uživatelé

Existuje několik druhů uživatelů aplikace. Každá role disponuje jinými oprávněními a možnostmi. Všechny role, kromě nepřihlášeného uživatele, se přihlašují svým e-mailem a heslem.

4.1 Výchozí návštěvník

Výchozím návštěvníkem je každý nepřihlášený uživatel. Ten má zpřístupněný pouze seznam týmových prací aktuálního ročníku s jejich názvem, popisem a očekávaným výstupem.

4.2 Student

Role přihlášeného studenta, který má možnost se zapsat do týmové práce (pokud už není zapsán v jiné). Může tak učinit jen v rámci časového úseku nastaveného u daného ročníku. Časový úsek je stanovován administrátorem.

4.3 Konzultant

Role pro učitele, kteří jsou u dané týmové práce nastaveni jako konzultanti. Mohou si prohlížet informace o týmových pracích (včetně obsazených pozic), přidávat hodnocení a upravovat zadání práce či pozice studentů. Dále mohou organizovat práci svých týmů, např. svolávat schůzky.

4.4 Garant

Role pro garanta nebo garanty týmové práce. V praxi takový člověk zodpovídá za samotnou práci, v rámci aplikace však má stejná oprávnění jako konzultanti. V případě, že by se aplikace rozšiřovala o nové funkce, může role fungovat jako nadřazená konzultantům.

4.5 Administrátor

Uživatel s nejvyššími právy. Jako jediný disponuje přístupem do administrace. V té může – pomocí webového rozhraní – měnit hodnoty většiny parametrů všech databázových modelů, spravovat jednotlivé ročníky a práce aj.

5 Spuštění systému

Aplikaci lze spustit přímo v operačním systému nebo v Docker kontejneru, jak bylo popsáno v předchozích kapitolách. Níže jsou podrobněji rozebrány obě možnosti.

5.1 Operační systém

Pro samotný chod přímo v operačním systému je potřeba ho mít na serveru již nainstalovaný. Zde bych rád upozornil, že maturitní práce byla vyvinuta na GNU/Linux distribuci Debian, tudíž na jiných distribucích nebo operačních systémech nemusí fungovat korektně.

Kromě instalace operačního systému musí administrátor serveru nastavit další prerekvizity aplikace, jimiž jsou Node.js[5], MongoDB[6] a Redis[7] databáze[7].

Doporučuji také použít správce procesů (jako např. PM2), který se bude starat o bezproblémový chod aplikace. Zajistí například zotavení z pádu aplikace nebo její spuštění po restartu serveru. Volitelně je možné konfigurovat další způsoby chování, jako je restartování procesu při změně některého ze souborů ve stanoveném adresáři (typicky cesta ke zdrojovému kódu projektu).

5.2 V kontejneru

K aplikaci je vytvořený vlastní Docker image. Funkčnost kontejneru je závislá na MongoDB a Redis databázích. Ty mohou běžet ve stejném kontejneru společně se samotnou aplikací, vhodnější variantou je však běh oddělený – v jiných vzájemně propojených kontejnerech nebo na hostujícím operačním systému.

Obraz kontejneru si stačí stáhnout z veřejného Docker Hub registru a následně spustit s danými environmentálními proměnnými, které jsou popsány v souboru `README.md`.

[5] software navržený pro vyvíjení škálovatelných aplikací
[6] multiplatformní NoSQL databáze
[7] úložiště datové struktury v paměti serveru

6 Zápisy a přidělování studentů

U každého ročníku je stanovený časový úsek, v němž si lze týmové práce ze strany studentů vybírat. Předpokladem výběru je skutečnost, že pozice, o niž má student zájem, je v daném projektu stále volná. Jednotlivé pozice definuje administrátor v administraci při vytváření práce (ať už úplně nové, nebo ze šablony).

Systém funguje tím způsobem, že kdo dřív přijde, ten danou pozici získá. Pokud by dva studenti žádali o tutéž pozici, získá ji zájemce, jehož požadavek přišel na server dříve.

7 Přenos a import studentů

Přenos a import studentů jsem měl původně v plánu realizovat pomocí napojení na školní LDAP servery[8], to mi však bylo zamítnuto. Proto jsem musel zvolit alternativní řešení – přenos/import uživatelů pomocí XLSX souboru nahrávaným do aplikace.

Samotnou strukturu XLSX tabulky si zvolila paní Hrušková, která by měla v budoucnu mít samotnou administraci týmových prací na starosti. Vzorový soubor je možné si kdykoliv stáhnout přímo prostřednictvím webového rozhraní. Odpadá tak nutnost řešit přechovávání vzorového souboru (s nadefinovanou strukturou).

[8] autorizační server, který dokáže poskytnout informace o školních účtech

Závěr

Zadání na téma „Správa týmových prací“ jsem splnil v celém původně stanoveném rozsahu. V příštích týdnech budu však funkce dále rozšiřovat nad rámec zadání ve spolupráci s paní Hruškovou.

Snažil jsem se dosáhnout maximální přívětivosti pro uživatele, rychlosti a minimálních systémových požadavků.

Podařilo se mi vytvořit v praxi použitelnou aplikaci pro naši střední školu. Během toho jsem rozšířil i své znalosti z oblasti technologií Node.js a MongoDB, se kterými plánuji nadále pracovat.

Odkazy

[1] MongoDB, Inc. (2021). „MongoDB - NoSQL database,“ WWW: https://www.mongodb.com/ (cit. 10. 04. 2021).

[2] OpenJS Foundation. (2021). „NodeJS - JavaScript runtime,“ WWW: https://nodejs.org/en/ (cit. 10. 04. 2021).

[3] PM2. (2021). „PM2 - Advanced, production precess manager for Node.JS,“ WWW: https://pm2.keymetrics.io/ (cit. 10. 04. 2021).

[4] VMware, Inc. (2021). „VMware vSphere - Server Virtualization Software,“ WWW: https://www.vmware.com/products/vsphere.html (cit. 10. 04. 2021).

[5] (2021). „Debian - The Universal Operating System,“ WWW: https://www.debian.org/ (cit. 10. 04. 2021).

[6] (2021). „Docker - Containers,“ WWW: https://www.docker.com/ (cit. 10. 04. 2021).

[7] Redis Labs. (2021). „Redis - in-memory data structure store,“ WWW: https://redis.io/ (cit. 10. 04. 2021).