Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Agilná výuka nových vlastností jazyka C++ v C++14, C++17, C++20

Bakalárska práca

2022
Martin Zavadzan

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Agilná výuka nových vlastností jazyka C++ v C++14, C++17, C++20

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Ing. František Gyárfáš, PhD

Bratislava, 2022
Martin Zavadzan

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Martin Zavadzan
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Agilná výuka nových vlastností jazyka C++ v C++14, C++17, C++20
Agile e-learning web application for learning new versions of C++

Anotácia: Cieľom bakalárskej práce je navrhnúť a vytvoriť interaktívne webové prostredie pre výuku nových vlastností jazyka C++11, C++14, C++17 a C++20 využitím agilných metód programovania. Prostredie umožní študentom riešenie úloh využívajúcich vlastnosti niektorej z verzií jazyka a refaktorizáciu do inej verzie. Úlohy budú definované pomocou testmi riadeného programovania (TDD). Web aplikácia bude obsahovať editor kódu v C++ a na serveri databázu úloh a študentských riešení, kompilátor a virtuálny server pre zbiehanie TDD riešení úloh. Súčasťou práce bude niekoľko ukážkových úloh, demonštrujúcich možnosti systému. Systém bude realizovaný pomocou technológii/nástrojov: PostgreSQL, HTML5, CSS, SASS, JavaScript (jQuery, Bootstrap, Node, React, Redux/Store, Express), TypeScript knižnice pre testovanie, kompilátor a virtuálny server.

Vedúci: Ing. František Gyarfaš, CSc.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 12.10.2021

Dátum schválenia: 13.10.2021 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Chcel by som poďakovať môjmu školiteľovi Ing. František Gyárfáš, PhD za jeho pomoc a ochotu pri tvorbe bakalárskej práce. Ďalej by som chcel poďakovať svojej rodine za podporu a svojím kamarátom, ktorí mi pomohli s testovaním výslednej aplikácie.

Abstrakt

Cieľom bakalárskej práce je navrhnúť a vytvoriť interaktívne webové prostredie pre výuku nových vlastností jazyka C++11, C++14, C++17 a C++20 využitím agilných metód programovania. Prostredie umožní študentom riešenie úloh využívajúcich vlastnosti niektorej z verzií jazyka a refaktorizáciu do inej verzie. Úlohy budú definované pomocou testmi riadeného programovania (TDD). Web aplikácia bude obsahovať editor kódu v C++ a na serveri databázu úloh a študentských riešení, kompilátor a virtuálny server pre zbiehanie TDD riešení úloh. Súčasťou práce bude niekoľko ukážkových úloh, demonštrujúcich možnosti systému.

Kľúčové slová: C++, agilné programovanie, webová aplikácia, testami riadený vývoj, edukačný softvér.

Abstract

The aim of the bachelor thesis is to design and implement interactive web environment for learning new features of C++11, C++14, C++17 and C++20 by agile programming methods. The environment will let students to solve problems by usage of specific language version features and refactorization to another version. Problems will be defined by Test-driven development (TDD). The webpage contain: C++ code editor, database of problems and student's answers, compiler and virtual server. Bachelor thesis will contain samples of problems which will show possibilities of system.

Keywords: C++, agile programming, web application, test-driven development, educational software.

Obsah

Zoznam obrázkov

Úvod

Žijeme vo svete, ktorý je výrazne ovplyvnený rýchlo sa vyvíjajúcimi a meniacimi informačnými technológiami. Vo svete, v ktorom vznikajú nové programovacie jazyky, ktoré sú schopné veľmi rýchlo nabrať na popularite, ale aj rýchlo upadnúť do zabudnutia. Je len pár takých, ktoré sa držia na vrchole dlhodobo a pravdepodobne tam nejakú tú dobu ešte zostanú. Po krátkom nahliadnutí do historických tabuliek obľúbenosti programovacích jazykov si môžme všimnúť, že jedným z týchto jazykov je práve jazyk C++. Toto si všimol vo svojej knihe aj Jeff Langr [13] a opísal to nasledovne. Napriek súčasnému boomu vo vývoji programovacích jazykov, C++ odoláva naďalej a je štvrtým najpopulárnejším programovacím jazykom podľa TIOBE indexu pre rok 2013. Dnes o takmer 10 rokov neskôr môžme v TIOBE indexe [7] vidieť jazyk C++ na nezmenenom 4 mieste, čo len svedčí o pravdivosti predchádzajúceho tvrdenia. Už krátko po svojom vzniku sa mohol jazyk C++ tešiť záujmu odbornej verejnosti v oblasti informačných technológií a aj v súčasnosti zostáva v istých oblastiach informatiky ako možnosť číslo jedna. Toto však nie je zásluha len už dávno získanej popularity, ale hlavne neustálemu vývoju a rozširovaniu vlastností jazyka tak, aby sa minimálne vyrovnal, ak aj nepredbehol moderné programovacie jazyky. Keďže nové verzie jazyka C++ vychádzajú od verzie C++11, vydanej v roku 2011, v 3 ročných intervaloch, počet nových vlastností a úprav v jazyku je pomerne obsiahli a je náročne udržať si prehľad o všetkom čo bolo pridané alebo upravené.

Práve toto ma priviedlo k myšlienke zamerať svoju bakalársku prácu na tému agilnej výuky nových vlastností jazyka C++. Výsledná aplikácia by mala byť využívaná na výuku a oboznamovanie sa s novými vybranými vlastnosťami jazyka C++. Aplikácia vedie tvorcu úlohy k tomu, aby boli úlohy navrhnuté práve tak, aby bol riešiteľ nútený zamyslieť sa nad fungovaním niektorej z nových funkcionalít a pokúsiť sa zreplikovať jej správanie. Toto prostredie bude vo forme webovej aplikácie. Sekcia Návrh je zameraná na aplikáciu vo fáze dizajnu, popisuje, ako by mala aplikácia fungovať a čo by mala spĺňať, zatiaľ čo sekcia Riešenie popisuje samotnú implementáciu finálneho riešenia a technológie, ktoré boli pri tom použité. Sekcia s názvom Výuka je zameraná na ozrejmenie niektorých vybraných vlastností nových verzií jazyka, ich stručný opis a taktiež ponúka sadu vzorových úloh, ktoré slúžia ako ukážka funkcionality aplikácie.

Kapitola 1

Východisková kapitola

1.1 Agilný vývoj softvéru

V nasledujúcej kapitole si povieme niečo o agilnom vývoji softvéru. Vznik samotných metód agilného vývoju softvéra je spájaný s Manifestom agilného vývoja softvéra[10]. Manifest vznikol vo februári 2001, kedy sa skupina 17 ľudí, propagátorov agilných metód ako aj priaznivcov podporujúcich potrebu alternatívnych metód vývoja softvéru, rozhodla napísať vyhlásenie, v ktorom sa prihlásili k určitým hodnotám pri vývoji softvéru[12]. Jednou z týchto hodnôt je napríklad Testami riadené programovanie, o ktorom si povieme viac v nasledujúcej kapitol.

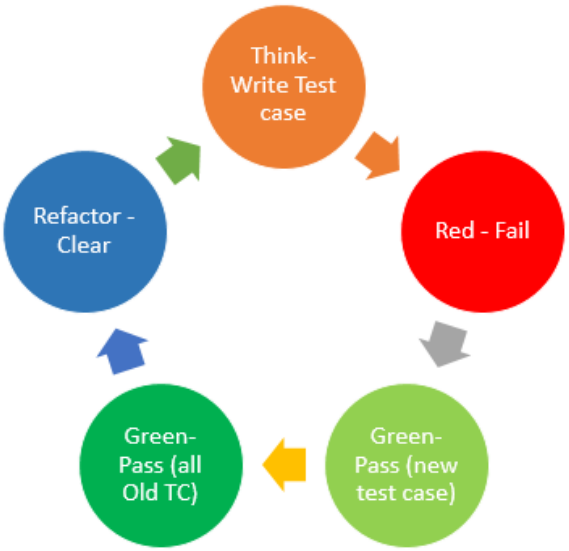
1.1.1 Testami riadené programovanie

Testami riadene programovanie alebo aj Test driven development (TDD) je spôsob vývoja softvéru, pri ktorom najskôr napíšeme test, potom napíšeme kód, ktorý daným testom úspešne prejde. Potom sa snažím nájsť najlepší dizajn (refaktorizácia) pre to čo už máme hotové, spoliahajúc sa na testy, ktoré už máme, tak aby sme nič nepokazili. [11] Tento princíp budem využívať aj vo svojej práci, keďže testy k jednotlivým úlohám budú dané skôr ako užívateľ začne s riešením úlohy. Takýmto spôsobom si presne vymedzíme funkcionalitu a v konečnom dôsledku sa tak vyhneme tvorbe funkcionality, ktorá nebola žiadaná resp. potrebná. Testy delím zvyčajne na 2 časti a to:

- Unit tests - sú to testy, ktoré kontrolujú funkcionalitu veľmi malej časti kódu. Pri objektovo orientovanom hovoríme napríklad o metóde triedy, pri funkcionálnom programovaním môžme hovoriť o atomickej funkcii [14]. Tieto testy by mali byť rýchle a na sebe nezávislé.

- Integration tests - sú to testy, ktoré majú za úlohu overiť, či jednotlivé na sebe nezávislé časti a moduly fungujú správne ako celok.

Nasledujúci obrázok 1.1 popisuje celý postup vývoja aplikácie pomocou metód testom riadeného vývoja.



Obr. 1.1: Cyklus vývoja metódou TDD [9]

1.2 Nové vlastnosti jazyka C++ vo verziách 11, 14, 17 a 20

C++11 je druhá významnejšie verzia jazyka C++ a zároveň najvýznamnejšia verzia od verzie C++98.Priniesla so sebou sebou veľa zásadných zmien, akou je napríklad abstraktnosť.[1]
Jej vydaniu predchádzala verzia C++03 a teda vyšla až po 8 rokoch od predchádzajúcej verzie, čo bol zatiaľ najväčší časový rozostup medzi verziami. Odvtedy nové verzie vychádzajú v 3 ročných rozostupoch, poslednou verziou je verzia C++20. Verzie C++14 až C++20 nepriniesli už také zasadne zmeny, skôr mierne vylepšenia a opravy chýb, ale aj nejakú novú funkcionalitu, ktorej sa budem venovať v tejto práci.

1.3 Testovanie

Z hľadiska bezpečnosti budem musieť riešiť otázku toho, ako bezpečne zbuildovať, spustiť a teda aj otestovať užívateľsky kód. Keďže o samotnom kóde nič vopred nevieme, môže potencionálne byť bezpečnostným rizikom. Mohol by sa napríklad pokúsiť manipulovať s dátami na serveri a podobne. Kvôli tomu je potrebné užívateľský kód zaobaliť

aby nebol schopný narábať s dátami na servery. Tu sa nám naskytuje hneď niekoľko možnosti, no práve nasledujúce mi prišli ako najvhodnejšie.

1.3.1 Virtualny počítač

Virtuálny počítač (VM) sa nijak nelíši od klasického počítaču, tak ako ho poznáme. Má svoj procesor, pamäť a dokonca aj úložisko. Jediný rozdiel je v tom, že na rozdiel od klasického počítaču nie sú tieto súčasti hardvérové a sú teda len softvérovo vytvorené.

- Výhody VM
  - Možnosť behu viacerých OS na jednom hostiteľskom zariadení súčasne
  - Izolovanosť od hostiteľského zariadenia
- Nevýhody VM
  - Menšia rýchlosť a efektívnosť, keďže práca s hardvérom nie je priama
  - Výrazne väčšia pamäťová náročnosť v porovnaní s kontajnerom

1.3.2 Kontajner

Z hľadiska bezpečnosti budem musieť riešiť otázku, toho ako bezpečne zbuildovať, spustiť a teda aj otestovať užívateľsky kód. Keďže o samotnom kóde nič vopred nevieme, môže potencionálne byť bezpečnostným rizikom. Mohol by sa napríklad pokúsiť manipulovať s dátami na servere a podobne. Kvôli tomu je potrebné užívateľský kód izolovať, aby nebol schopný narábať s dátami na servery. Tu sa nám naskytuje hneď niekoľko možnosti, no práve nasledujúce mi prišli ako najvhodnejšie. V konečnom dôsledku som sa rozhodol práve pre kontajnery, pre ich menšiu pamäťovú náročnosť, vyššiu rýchlosť a taktiež preto, že je možné spúšťať súčasne väčšie množstvo kontajnerov ako virtuálnych mašín.

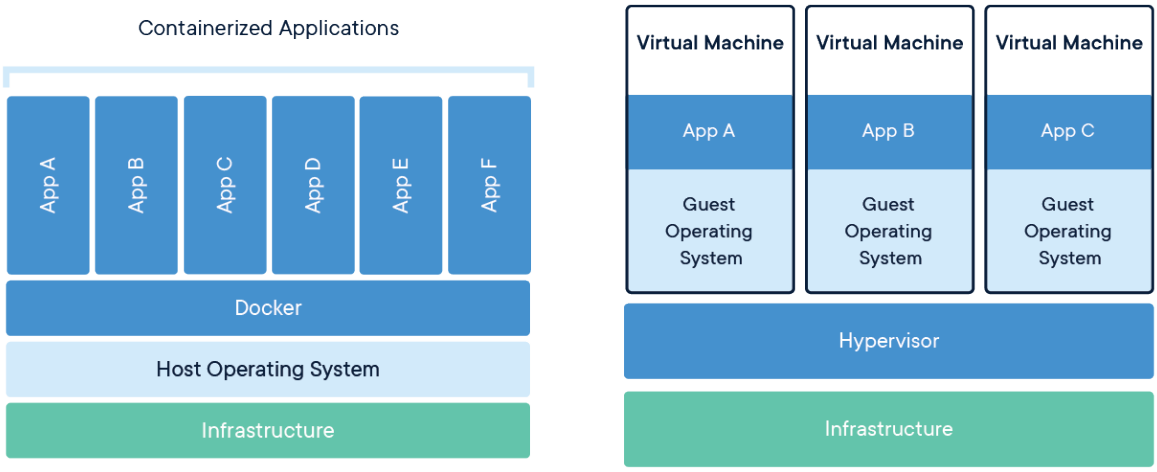
1.4 Existujúce riešenia

V tejto kategórií sa pozrieme na riešenia, ktoré sa istou časťou zaoberajú rovnakou problematikou

1.4.1 Podobné webové aplikácie

LearnCpp

Celá webová stránka je výhradne zameraná na výučbu jazyka C++. Výuka začína od úplných základov a je vhodná pre užívateľov so základnými znalosťami práce s počítačom. Jednotlivé lekcie sú pomerne dlhé, zato však popisujú problematiku veľmi



Obr. 1.2: Porovnanie štruktúry docker kontajneru a VM [8]

podrobne a zrozumiteľne. Hlavnou nevýhodou tohto webu je, že kód sa nedá písať a spúšťať priamo na webe, no užívateľ je nútený kód písať, kompilovať a spúšťať u seba na počítači. To má za dôsledok to, že užívateľ potrebuje inštalovať editor kódu a kompilátor. Forma testov na stránke je riešená pomocou kontrolných otázok so správnymi odpoveďami na konci jednotlivých častí kurzu. V porovnaní s mojou prácou web nie je zameraný na rozdiely v jednotlivých verziách jazyka C++, ale prevažne na jeho základnú funkcionalitu.

Another example

Here's a slightly more complex example. Remember, lifetime is a runtime property, and scope is a compile-time property, so although we are talking about both in the same program, they are enforced at different points.

```
#include <iostream>

int add(int x, int y) // x and y are created and enter scope here
{
    // x and y are visible/usable within this function only
    return x + y;
} // y and x go out of scope and are destroyed here

int main()
{
    int a{ 5 }; // a is created, initialized, and enters scope here
    int b{ 6 }; // b is created, initialized, and enters scope here

    // a and b are usable within this function only
    std::cout << add(a, b) << '\n'; // calls function add() with x=5 and y=6

    return 0;
} // b and a go out of scope and are destroyed here
```

Parameters *x* and *y* are created when the *add* function is called, can only be seen/used within function *add*, and are destroyed at the end of *add*. Variables *a* and *b* are created within function *main*, can only be seen/used within function *main*, and are destroyed at the end of *main*.

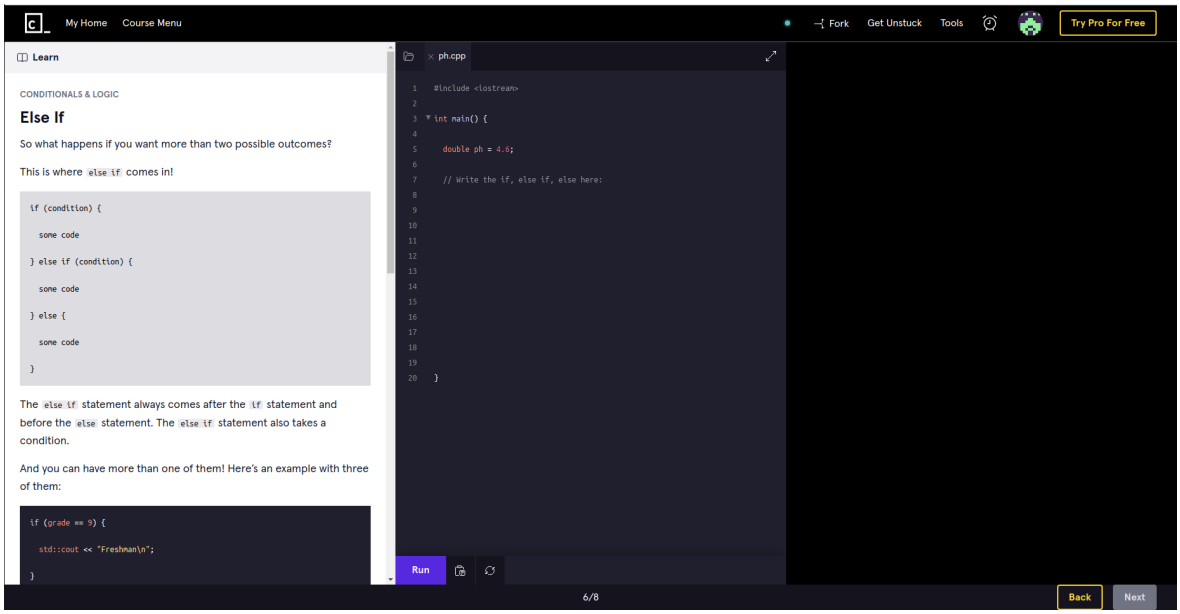
To enhance your understanding of how all this fits together, let's trace through this program in a little more detail. The following happens, in order:

- execution starts at the top of *main*
- *main*'s variable *a* is created and given value *5*
- *main*'s variable *b* is created and given value *6*
- function *add* is called with values *5* and *6* for arguments
- *add*'s variable *x* is created and initialized with value *5*
- *add*'s variable *y* is created and initialized with value *6*
- *operator+* evaluates expression *x + y* to produce the value *11*
- *add* copies the value *11* back to caller *main*
- *add*'s *y* and *x* are destroyed
- *main* prints 11 to the console
- *main* returns *0* to the operating system
- *main*'s *b* and *a* are destroyed

Obr. 1.3: Ukážka stránky LearnCpp

Codecademy

Webová stránka Codecademy ponúka široké spektrum kurzov v rôznych programovacích jazykoch, medzi, ktorými sa nachádza aj jazyk C++. Medzi hlavné výhody v porovnaní so stránkou LearnCpp patrí možnosť písania a spúšťania samotného kódu priamo na stránke a teda nie je potrebná žiadna inštalácia na strane používateľa. čo sa týka obsahu stránka Codecademy neponúka tak podrobné a detailne vysvetlenie problematiky ako stránka LearnCpp. Pre celkové dokončenie kurzu je potrebné zakúpenie PRO verzie, keďže niektoré časti kurzu sú dostupné len v PRO verzií. Podobne ako webová stránka LearnCpp tak aj C++ kurz na webovej stránke Codecademy nie je zameraný na rozdiely v jednotlivých verziách jazyka C++, ale prevažne na jeho základnú funkcionalitu.



Obr. 1.4: Ukážka stránky Codecademy

Web teaching tool for agile learning new features in C++11, C++14 and C++17, Matúš Gál, 2020

Toto riešenie ponúka možnosť riešenia rôznych úloh v jazyku C++ za použitia techník agilného programovania. Riešenie obsahuje 4 druhy úloh a to:

- Theory - má za úlohu užívateľovi priblížiť oblasť, na ktorú je úloha zameraná.
- First contact - ako už môžme z názvu usúdiť, jedna sa o prvú časť, v ktorej sa užívateľ prvýkrát stretáva s danou funkcionalitou. Tieto úlohy sú zamerané hlavne na ľahké úvodné príklady, tvorené len jednou funkciou alebo viacerými jednoduchými funkciami.

- Blind tests - tieto úlohy sú inšpirované princípmi TDD. Užívateľ môže odoslať svoje riešenie na server kde je otestované sadou testov. Keď niektorý z testov zlyhá užívateľ o tom dostane detailnú správu na základe, ktorej je schopný svoj kód upraviť. Celý proces opakuje kým neskončia všetky testy úspešne.
- Legacy code - Kód, ktorý užívateľ obdržal je plne funkčný a jeho úlohou je jeho refaktorizácia.

úlohy spomenuté v bakalárskej práci boli primárne cielené práve na agilne programovacie techniky.

Web teaching tool for learning agile programming, Tamara Savkova, 2019

Práca je zameraná na výučbu rôznych metód extrémneho programovania, ako napríklad práca s legacy kódom a TDD. Webová stránka ponúka niekoľko typov úloh ako napríklad oprava chýb v legacy kóde pomocou testov, pridávanie novej funkcionality a refaktorizácia legacy kódu. Na rozdiel od mojej práce je táto práca primárne zameraná na agilné programovanie, a teda nerieši problematiku jednotlivých verzií jazyka.

1.5 Použité technológie

V nasledujúcej časti si povieme niečo o front-end a back-end technológiách, ktoré som použil pri tvorbe môjho finálneho riešenia.

1.5.1 Frontend technológie

React.js

React je JavaScriptový webový framework určený na tvorbu užívateľského rozhrania. Už ako môžme predpokladať z názvu, je primárne určený na tvorbu reaktívnych užívateľských rozhraní. V súčasnosti sa teší veľkej popularite a patrí medzi najpoužívanejšie JavaScriptové frameworky. React aplikácie sú tvorené pomocou react komponentov, ktoré sú spoločne spájané a skladané, tým vytvoria konečné celkové užívateľský interface. Komponent je tvorený rôznymi znovupoužitelnými HTML komponentami, svojou vlastnou zapúzdrenou funkcionalitou a spravuje si svoj vlastný vnútorný stav resp. state, o ktorom si viac povieme pri frameworku Redux. Takýto spôsob práce s dátami nazývame deklaratívny.

Redux

Redux je jedna z najpoužívanejších state manažment knižníc v kombinacií s frameworkom React. Úlohom tejto knižnice je spravovať centralizované úložisko, ďalej nazývané

store tak, aby bolo prístupné každému komponentu aplikácie. Podľa oficiálnej dokumentácie [5] je Redux založený na týcho štyroch princípoch:

- Predictable - Redux pomáha vytvárať aplikaci, ktorá sa správa konzistentne bez ohľadu na prostredie.
- Centralized - Celý stav storu, v ktorom sa aplikácia aktuálne nachádza, je uložený na jednom mieste a tak vieme jednoducho reagovať na zmeny vo rôznych častiach aplikácie.
- Debuggable - S použitím špeciálnych DevTools pre Redux vieme sledovať kde, kedy a ako sa zmenil stav aplikácie.
- Flexible - Je schopný pracovať s ľubovolnou vrstvou užívateľského rozhrania a môže byť rozšírený o široké spektrum doplnkov.

Material-UI

Material-UI je JavaScriptova knižnica, ktorá ponúka široké spektrum rôznych React komponentov a tým zrýchľuje celý proces tvorby aplikácie, kedže developer nemusí vytvárať jednotlivé komponenty od základu. Veľkou výhodou Material-UI je, že komponenty sa dajú ľahko modifikovať tak aby spĺňali všetky požiadavky či už z pohľadu funkcionality alebo samotného vzhľadu.

Monaco editor

Monaco editor je editor zdrojového kódu, ktorý podporuje široké množstvo programovacích jazykov, podporuje aj zvýraznovanie textu pre jazyk C++ a preto som sa ho rozhodol využiť v tejto bakalárskej práci. Je vyvíjaný spoločnosťou Microsoft rovnako ako jeden z najznámejších desktopových editorov Visual Studio Code. Už na prvý pohľad si môžeme všimnúť, že Monaco editor je vlastne oklieštená verzia tohto editora. To je ďaľší z dôvodov prečo som sa rozhodol práve pre tento editor. Kedže Visual Studio Code je jeden zo súčasne najpoužívanejších editorov, je veľká šanca, že značná časť užívateľov bude už oboznámená s funkcionalitou Monaco editoru. Editor je taktiež vyvinutý vo rôznych formách, jednou z nich je práve reactovský komponent a to mi umožní ho bezproblémov používať s ostatnými knižnicami, ktoré sú vyvíjané práve pre React projekty.

1.5.2 Backend technológie

Node.js

Node.js je JavaScriptový open-source server, navrhnutý na tvorbu škálovateľných sieťových aplikácií.[2] Taktiež beží na jednom vlákne, je asynchrónny a neblokujúci, čo

má za dôsledok to, že nie je tak pamäťovo náročný. Práve pre tieto vlastnosti som si vybral Node.js. Pri nasadení aplikácie bude nutné, aby zvládala viacero užívateľských requestov zároveň a užívatelia sa medzi sebou neblokovali. Node zvláda toto všetko bez toho aby bol programátor nútený použiť viac vlákien a teda aj riešiť možný deadlock.

Express.js

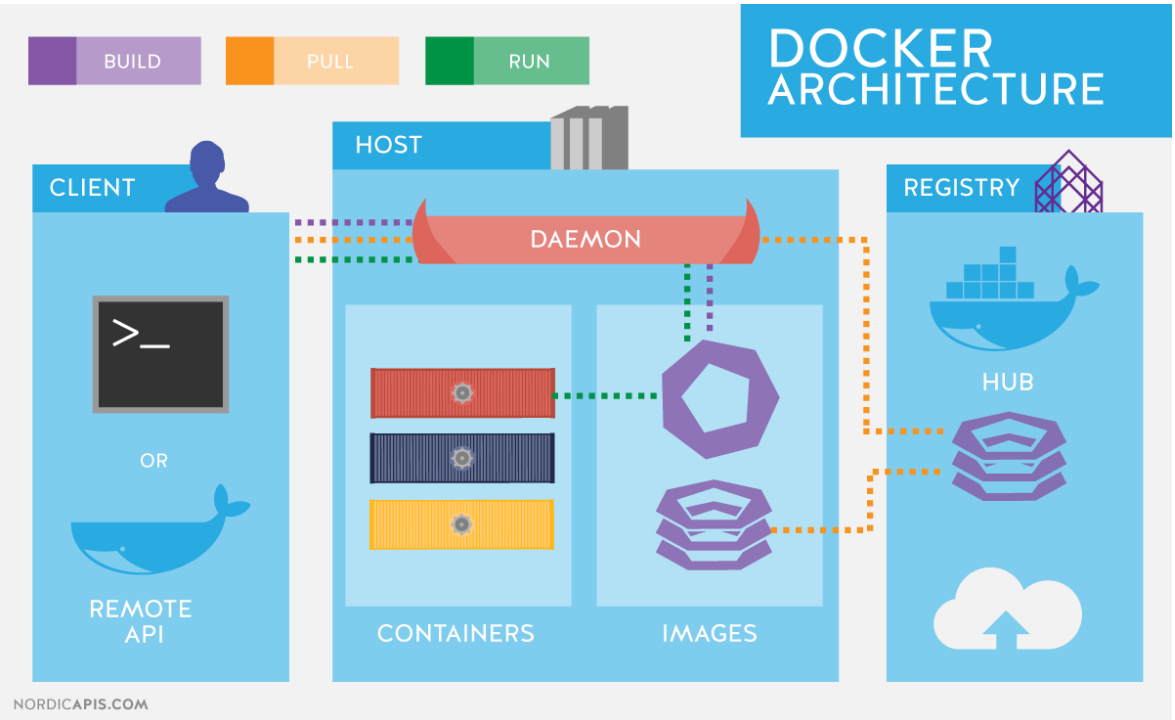
Express je minimalistická a flexibilný webový Node.js framework, ktorý poskytuje robustnú sadu funkcií pre webové a mobilné aplikácie.[4] Vďaka svojej jednoduchosti a robustnosti je v súčasnosti používaný vo veľkom množstve rôznych iných Node.js frameworkov.

Docker

Docker je open-source platforma určená na vývoj a aj samotný beh aplikácie. Pomocou Dockeru sme schopní aplikácie uvádzať do prevádzky jednoducho a rýchlo vďaka vysokej forme automatizácie. Vďaka tomu, že docker využíva takzvanú kontajnerizáciu, môžeme s istotou povedať, že sa aplikácia zaobalená v tomto kontajneri bude správať rovnako na všetkých zariadeniach, kde sme schopní tento docker kontajner spustiť. Ďalšou z výhod je samotná izolovanosť kontajnerou, to znamená, že dokážeme spúšťať kontajner na hostiteľskom zariadení bez toho, aby sme ovplyvnili či už ostatné kontajneri bežiace na tomto zariadení alebo aj iné aplikácie. Na vytvorenie docker aplikácie potrebujeme docker image, čo je vlastne nejaký tamplate, podľa ktorého docker dokáže vytvoriť kontajner. Inštrukcie pre tvorbu tohoto templatu sú zapísané v Dockerfile. To je formát, v ktorom dokážeme dockeru povedať:

- čo má daný kontajner robiť
- ako ma aplikáciu kontajner spúšťať
- aké zdroje a dáta ma kontajneru poskytnúť

Image vieme vytvoriť aj z viacerých imagov, ktoré sú dokonca vo veľkom množstve verejne dostupné, čo prispieva k rýchlemu a flexibilnému vývoju.



Obr. 1.5: Docker architektúra [3]

GoogleTest

V nasledujúcej sekcií vychádzam z oficiálnej Google Test dokumentacie [6]

Google test je testovací framework vytvorený Testovacím tímom podľa požiadaviek Googlu. Vznikol ako nástroj na pomoc pri vývoji a hlavne samotnom testovaní C++ aplikacií. GoogleTest podporuje rôzne typy testov, nielen unit testy, ako je mnohokrát mylne uvádzané.

Aby boli GoogleTest čo najužitočnejšie pri vývoji, boli tvorené na základe nasledujúcich presne zadefinovaných požiadaviek.

1. Mali by byť nezávislé. Aby testy neboli vyhodnotené ako neúspešné resp. úspešné kvôli inému testu, testy sú izolované a spúšťané na rozdielnych objektoch. Testy je možne spúšťať aj jednotlivo a nie len ako celok, čo dokáže výrazne ušetriť čas a tým pádom aj zvýšiť efektivitu pri vývoji aplikácie.

2. Testy by mali byť organizované a teda nejak štrukturované. Toto je dosiahnute tým, že je možne testy členiť na jednotlivé skupiny testov napríklad na základe toho, ktorú časť aplikácie testujú.

3. Testy by nemali byť závisle od platformy. Google Test teda fungujú na rôznych platformách a dokonca aj operačných systémoch.

4. Mali by sprostredkovať čo najviac informácii v prípade o testovanej aplikácii. Google testy teda neukončia svoj beh pri prvom neúspešnom teste a pokračujú ďalej, až kým sa ne neuskutočnia všetky testy. Takto teda zabezpečia to, že vývojár dostane maximálny možný objem informácii o správnosti vyvíjanej aplikácie.

5. Jednou z najhlavnejších požiadavok je samotná rýchlosť testov. Vysoká rýchlosť testov je zabezpečená tým, že testy medzi sebou zdiaľajú pamäť beztoho, aby boli testy na sebe závislé.

Kapitola 2

Návrh

V nasledujúcej kapitole sa zameriame na popis celkového návrhu webovej aplikácie. Aplikácia umožní bežnému užívateľovi, ďalej nazývaný študent, riešiť úlohy zamerané na výuku nových vlastností jazyka C++ v jednotlivých verziách. Taktiež umožní administrátorovi, ďalej nazývaný učiteľ, pridávať, editovať a taktiež riešiť úlohy. Po odovzdaní riešenia úlohy bude užívateľ oboznámený s detailnými výsledkami testov správnosti odovzdaného riešenia.

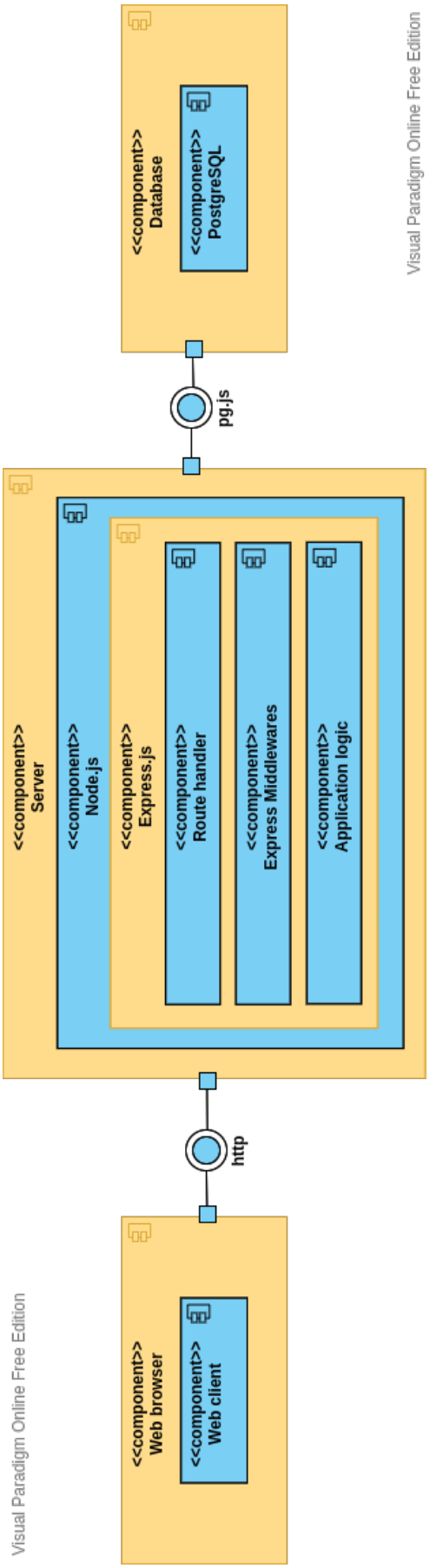
2.1 Architektúra

Aplikácia využíva architektúru klient-server, táto architektúra je zobrazená na komponent diagrame 2.1.

Vrchná vrstva aplikácie, s ktorou narába samotný užívateľ, je sprostredkovaná webovým prehliadačom (frontend). Táto vrstva získava potrebné informácie a funkcionalitu zo serverovej časti aplikácie (backend). Serverová časť teda poskytuje potrebnú funkcionalitu, akou je napríklad spúšťanie testov nad zdrojovým kódom užívateľa, ale aj potrebné dáta z databázy. Serverová časť teda ďalej komunikuje s databázovou časťou, na ktorej beží samotný databázový server.

2.2 Úlohy

Úlohy by mali byť vytvárané hlavne s cieľom priblížiť študentom nové časti jazyka C++ v jednotlivých verziách. Primárne by mal byť študent vedený k tejto činnosti tým, že jeho úlohou bude refaktorovať kód napísaný v niektorej z verzií do inej verzie tak, aby oba kódy fungovali rovnako, teda, aby oba úspešne prešli celou sadou priložených unit testov pre danú úlohu. Aplikácia bude zobrazovať výsledky testov, tu môžu nastať hneď 4 prípady (viď obr. 2.2):



Obr. 2.1: Komponent diagram

```
Running main() from gtest_main.cc
[==========] Running 8 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 8 tests from TestyPrvejUlohy
[ RUN      ] TestyPrvejUlohy.Jedna
[       OK ] TestyPrvejUlohy.Jedna (0 ms)
[ RUN      ] TestyPrvejUlohy.Dva
[       OK ] TestyPrvejUlohy.Dva (0 ms)
[ RUN      ] TestyPrvejUlohy.Dlhy
[       OK ] TestyPrvejUlohy.Dlhy (0 ms)
[ RUN      ] TestyPrvejUlohy.Prazdny
[       OK ] TestyPrvejUlohy.Prazdny (0 ms)
[ RUN      ] TestyPrvejUlohy.JednaNekonci
[       OK ] TestyPrvejUlohy.JednaNekonci (0 ms)
[ RUN      ] TestyPrvejUlohy.DvaKonci
[       OK ] TestyPrvejUlohy.DvaKonci (0 ms)
[ RUN      ] TestyPrvejUlohy.DlhyKonci
[       OK ] TestyPrvejUlohy.DlhyKonci (0 ms)
[ RUN      ] TestyPrvejUlohy.PrazdnyNekonci
[       OK ] TestyPrvejUlohy.PrazdnyNekonci (0 ms)
[----------] 8 tests from TestyPrvejUlohy (1 ms total)

[----------] Global test environment tear-down
[==========] 8 tests from 1 test case ran. (1 ms total)
[  PASSED  ] 8 tests.
```

Všetky testy boli úspešné

```
Running main() from gtest_main.cc
[==========] Running 8 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 8 tests from TestyPrvejUlohy
[ RUN      ] TestyPrvejUlohy.Jedna
/test/tests.cpp:12: Failure
      Expected: startsWith("a", 'b')
      Which is: true
To be equal to: false
[  FAILED  ] TestyPrvejUlohy.Jedna (0 ms)
[ RUN      ] TestyPrvejUlohy.Dva
[       OK ] TestyPrvejUlohy.Dva (0 ms)
[ RUN      ] TestyPrvejUlohy.Dlhy
[       OK ] TestyPrvejUlohy.Dlhy (0 ms)
[ RUN      ] TestyPrvejUlohy.Prazdny
[       OK ] TestyPrvejUlohy.Prazdny (0 ms)
[ RUN      ] TestyPrvejUlohy.JednaNekonci
[       OK ] TestyPrvejUlohy.JednaNekonci (0 ms)
[ RUN      ] TestyPrvejUlohy.DvaKonci
[       OK ] TestyPrvejUlohy.DvaKonci (0 ms)
[ RUN      ] TestyPrvejUlohy.DlhyKonci
[       OK ] TestyPrvejUlohy.DlhyKonci (0 ms)
[ RUN      ] TestyPrvejUlohy.PrazdnyNekonci
[       OK ] TestyPrvejUlohy.PrazdnyNekonci (0 ms)
[----------] 8 tests from TestyPrvejUlohy (1 ms total)

[----------] Global test environment tear-down
[==========] 8 tests from 1 test case ran. (1 ms total)
[  PASSED  ] 7 tests.
[  FAILED  ] 1 test, listed below:
[  FAILED  ] TestyPrvejUlohy.Jedna

 1 FAILED TEST
```

Niektorý z testov nebol úspešný

```
Running main() from gtest_main.cc
[==========] Running 8 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 8 tests from TestyPrvejUlohy
[ RUN      ] TestyPrvejUlohy.Jedna
TESTS RUN OUT OF TIME!
```

Vypršal čas pre vykonenie testov

```
rm: cannot remove 'result.txt': No such file or directory
In file included from /test/tests.cpp:5:
/test/code.cpp: In function 'bool startsWith(std::string, char)':
/test/code.cpp:5:12: error: 'x' was not declared in this scope
    5 |     return x;
      |            ^
gmake[2]: *** [CMakeFiles/solution.dir/build.make:76: CMakeFiles/solution.dir/tests.cpp.o] Error 1
gmake[1]: *** [CMakeFiles/Makefile2:83: CMakeFiles/solution.dir/all] Error 2
gmake: *** [Makefile:91: all] Error 2
```

Nastala kompilačná chyba

Obr. 2.2: Možné výsledky testov

1. úloha úspešne prejde všetkými testami - aplikácia zobrazí výsledky testov, s tým, že všetky budu v stave OK.

2. úloha neprejde úspešne všetkými testami - aplikácia zobrazí výsledky testov, s tým, že niektoré z nich budú označené ako FAILED.

3. užívateľsky kód pre danú úlohu sa nepodarí úspešne zbuildovať - aplikácia zobrazí podrobnú správu o chybe počas buildu.

4. časový limit pre testovanie úlohy bol prekročený - aplikácia zobrazí upozornenie, že časový limit pre testovanie úlohy bol prekročený.

Úlohy budú zámerne navrhnuté tak, aby študent prišiel do kontaktu s niektorou z nových vlastností alebo bude vedený k tomu, aby pri riešení úlohy práve použil niektorú z nových vlastnosti.

2.2.1 Vytváranie úlohy

Vytváranie nových úloh bude možne priamo z grafického rozhrania v prehliadači. Po otvorení stránky pre pridanie novej úlohy môže učiteľ nastaviť meno, popis a dátum odovzdania úlohy. Taktiež je schopný nastaviť testy pre danú úlohu, vybrať verzie jazyka C++, ktoré budú v úlohe použité a nastaviť aj predpripravené časti kódu v ľubovolnom z dvoch editorov, prípadne aj v oboch.

2.2.2 Editovanie úlohy

Táto možnosť bude taktiež výhradne dostupná učiteľovi, stránka vyzerá totožne so stránkou pre vytvorenie novej úlohy a je možné upravovať rovnaké polia, jediný rozdiel je taký, že polia sú už predvyplnené pôvodnými hodnotami.

2.2.3 Riešenie úlohy

Stránka riešenia úlohy je dostupná hlavne študentom, študentovi umožňuje riešiť zadanú úlohu priamo na stránke pomocou písania kódu v jazyk C++, študent si vie overiť správnosť svojho riešenia odoslaním vyhotovenej úlohy na server. Na stránke sa mu následne zobrazia výsledky jednotlivých testov. Študent taktiež vidí všetky testy, zadanie a názov úlohy priamo na stránke.

2.3 User interface

Aplikácia bude mať moderné reaktívne užívateľské prostredie s dôrazom na intuitívnosť. Tento cieľ docielime pomocou nasledujúcich prostriedkov.

2.3.1 Webový kód editor

Aplikácia bude obsahovať editor zdrojového kódu, ktorý bude poskytovať základnú funkcionalitu bežných desktopových editorov zdrojového kódu, akými sú napríklad zvýrazňovanie kľúčových slov, odsadzovanie textu alebo zvýrazňovanie zátvoriek. Cieľom je, aby aplikácia užívateľovi ponúkla čo najlepší user experience a teda aby sa čo najviac priblížila bežným desktopovým editorom, na ktoré sú už užívatelia zvyknutí.

2.3.2 Veľkostne nastaviteľný editor

Keďže bude obrazovka pre prácu s úlohou tvorená hneď 5 oknami s editorom zdrojového kódu, je potrebné nejakým spôsobom umožniť si ich veľkosť upravovať podľa potreby, toto bude zabezpečené tým, že editory bude možné rozťahovať, prípadne zmenšovať rovnako ako okná v operačných systémoch. Tým sa zabezpečí potrebný komfort pri práci v niektorom z okien a opäť sa priblížime k štandardným desktopovým editorom, ktoré túto možnosť obsahujú

2.3.3 Stránka pre prihlásenie

Keďže všetky ostatné časti aplikácie sú dostupné iba prihláseným užívateľom, je nutné, aby bol užívateľ autorizovaný skôr, ako sa dostane do zvyšných častí aplikácie. Táto stránka bude obsahovať formulár pre prihlásenie, ktorý bude tvorený dvoma poliami a to emailom a heslom, tlačidlom na potvrdenie prihlásenia, odkazom pre presmerovanie na registračný formulár nového užívateľa a odkaz pre prípad zabudnutého hesla . Pre aktiváciu tlačidla na odoslanie je potrebné, aby boli všetky polia validné a teda aby mal vstup pre pole s emailom skutočne formát emailu. Ak email nebude validný, zobrazí sa upozornenie. Po úspešnom vyplnení formuláru a stlačení tlačidla pre prihlásenie môžu nastať dve udalosti a to:

- zamietnuté prihlásenie - užívateľovi sa zobrazí upozornenie oznamujúce mu, že zadaný email alebo heslo nie sú správne
- úspešne prihlásenie - užívateľovi sa podarilo úspešne prihlásiť a je presmerovaný na úvodnú stránku aplikácie

2.3.4 Stránka pre registráciu nového užívateľa

Stránka obsahuje formulár pre email, heslo, overenie hesla a tlačidlo pre odoslanie dát. Tlačidlo je aktívne, len keď sú všetky polia formuláru validné, čiže v poli pre email sa nachádza validná emailová adresa a hodnoty v poli pre heslo a overenie hesla sú totožné. Ak prebehne celý proces registrácie úspešne, užívateľovi sa zobrazí notifikácia o úspešnom registrovaní a môže pokračovať prihlásením do aplikácie.

2.3.5 Stránka pre obnovu zabudnutého hesla

Obsahuje formulár s polom pre emailovú adresu, ktorej heslo chceme zmeniť a tlačidlo pre odoslanie požiadavky. Tlačidlo je aktívne, len ak je zadaná emailová adresa validnou emailovou adresou. Po úspešnom vykonaní požiadavky obdrží užívateľ email na danú adresu s novým náhodne vygenerovaným heslom. Heslo si môže po prvom úspešnom prihlásení zmeniť na stránke pre správu účtu na ľubovolné heslo.

2.3.6 Stránka pre správu užívateľského účtu

Obsahuje dva formuláre, jeden slúži na zmenu užívateľovho mena alebo priezviska. Druhý formulár slúži na zmenu aktuálneho hesla k účtu, sú v ňom dve polia a to pole pre nové heslo a pre potvrdenie nového hesla. Formulár je možne odoslať len vtedy, keď sú údaje validné a teda sa nové heslo a potvrdenie hesla zhodujú.

2.4 Dátový model

Táto sekcia popisuje dátový model pre finálnu webovú aplikáciu. Uchováva údaje o cvičeniach, užívateľoch, rozpracovaných cvičeniach, zdrojových kódoch užívateľov, odovzdaných riešeniach a ich výsledkoch. Celá dátová štruktúra aplikácie je bližšie popísaná na diagrame.

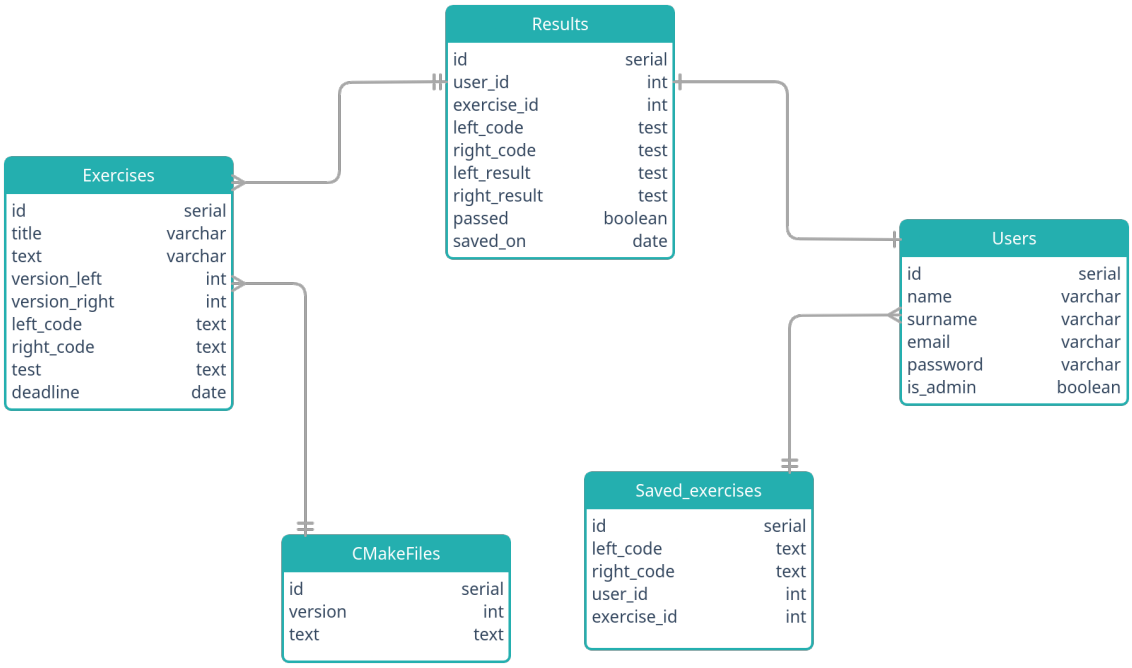
Diagram popisuje relácie medzi jednotlivými tabuľkami databázy. Tabuľky si osobitne popíšeme v nasledujúcich podsekciach.

2.4.1 Tabuľka Užívatelia (users)

Ako je už zo samotného názvu zrejmé, v tejto tabuľke sú vedené údaje o užívateľoch systému. Do tabuľky ukladáme meno užívateľa a priezvisko užívateľa. Ďalej si ukladáme email a heslo pre daného užívateľa, ktoré sú používané na prihlásenie do systému. Tieto dáta si užívateľ určí sám pri registrácií do systému. Posledný stĺpec tabuľky **is_admin** určuje typ užívateľa a teda či je užívateľ študent alebo učiteľ.

2.4.2 Tabuľka Cvičenia (exercises)

V tejto tabuľke sú uchované informácie o jednotlivých cvičeniach, ktoré sú v aplikácií dostupné užívateľom. Pre každé cvičenie si uchovávame jeho, názov, popis respektíve zadanie, dva inicializačné zdrojové kódy a to teda zdrojový kód, ktorý sa zobrazí v ľavom alebo pravom editore. Pre oba tieto kódy si ďalej evidujeme verziu jazyka C++, v ktorej bude kód kompilovaný pri testovaní. Ako posledné sú v tabuľke uložené unit testy pre príslušnú úlohu.



Obr. 2.3: Diagram popisujúci dátabázu

2.4.3 Tabuľka CMakeFiles

Tabuľka obsahuje predpripravené CMake súbory, ktoré sú potrebné na kompiláciu a testovanie zdrojového kódu od užívateľa. Pre každú verziu jazyka C++ je v tabuľke pripravený osobitný súbor.

2.4.4 Tabuľka Výsledkov (results)

Tabuľka obsahuje výsledky testov zdrojového kódu užívateľov pre jednotlivé úlohy. V tabuľke oba zdrojové kódy od užívateľa a taktiež aj výsledky testov pre dané zdrojové kódy. Ďalej každý riadok tabuľky obsahuje stĺpec **passed**, ktorý je nastavený na hodnotu **true** v prípade, že oba zdrojové kódy úspešne prešli všetkými testami. V opačnom prípade je hodnota **false**. V tabuľke sa môže nachádzať viac výsledkov pre jedno cvičenie od jedného užívateľa.

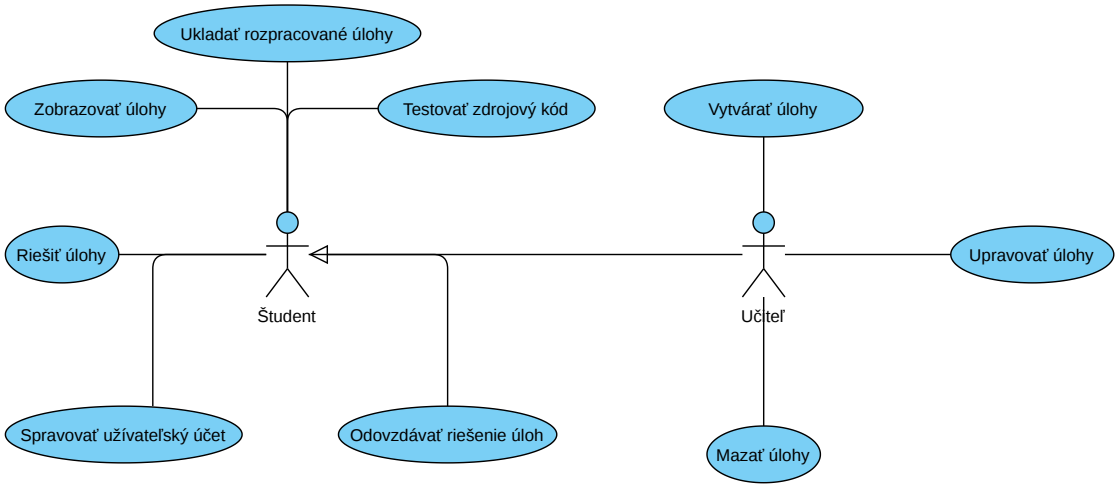
2.4.5 Tabuľka Uložených cvičení (saved_exercises)

V tabuľke sú uložené zdrojové kódy užívateľov, ktoré sú v stave rozpracované. Keď sa užívateľ rozhodne si uložiť svoj progress v aplikácií, oba zdrojové kódy sa uložia do databázy a pri opätovnom otvorení cvičenia sa tento uložený kód zobrazí v editor.

2.5 Use case

Aplikácia bude rozpoznávať dva typy používateľov, jeden z nich bude študent a ten druhý učiteľ (administrátor). Možnosti, ktoré budú mať v aplikácií sú bližšie popísane v use case diagrame.

Visual Paradigm Online Free Edition



Visual Paradigm Online Free Edition

Obr. 2.4: Use case diagram

Kapitola 3

Riešenie

3.1 Frontend

Nasledujúca kategória hovorí o časti aplikácie, ktorá beží u klienta a je pomocou nej schopný pracovať s aplikáciou.

3.1.1 TypeScript

TypeScript som sa rozhodol použiť ako hlavný jazyk pre frontendovú časť aplikácie. Rozhodol som sa tak hneď z niekoľkých dôvodov. Hlavným dôvodom bola snaha o skvalitnenie výsledného kódu a o zlepšenie jeho čitateľnosti. Ďalším dôvodom bolo priniesť statickú kontrolu už pri samotnom písaní zdrojového kódu a teda sa vyhnúť zbytočnému debugovaniu, čo v konečnom dôsledku urýchľuje a uľahčuje celkový vývoj aplikácie. Vyhneme sa tak hľadaniu chýb v javascripte, na ktoré nás je schopný upozorniť TypeScript už pred samotným spustením aplikácie.

3.1.2 React

Pre frontendovú časť aplikácie som sa rozhodol použiť javascriptovú knižnicu React. Ako si môžeme všimnúť už z názvu táto knižnica umožňuje tvorbu reaktívnej webovej aplikácie. Jeden z hlavných dôvodov prečo som sa rozhodol pre túto knižnicu je to, že je plne kompatibilná s jazykom TypeScript, ktorý som sa rozhodol používať ako primárny jazyk pre frontendovú časť aplikácie.

Material UI

Material UI je javascriptová knižnica, ktorá obsahuje široké spektrum reactových komponentov. Túto knižnicu som sa rozhodol použiť pre tvorbu výsledného grafického rozhrania pre jej plnú podporu TypeScriptu a taktiež pre jednoduchú editovateľnosť

funkcionality komponentov a už pripraveného vzhľadu komponentov tak aby spĺňali moju predstavu výsledného vzhľadu aplikácie

Redux Toolkit Query

Redux Toolkit Query (RTK Query) je javascriptová knižnica, ktorá je určená pre získavanie dát na frontendovej časti aplikácie ale aj pre cachovanie dát tak aby sa zamedzilo potrebe riešiť tento problém manuálne v priebehu tvorby webovej aplikácie. Táto knižnica je postavená na knižnici Redux Toolkit. Rozhodol som sa ju použiť z nasledovných dôvodov:

- kompatibilita s TypeScriptom
- schopnosť sledovať stav požiadavky a teda možnosť zobrazovať rôzne stavy načítanie (napríklad spinner pri tlačidle) a tak dať užívateľovi najavo, že sa niečo skutočne deje a tým v konečnom dôsledku navodiť pocit svižného chodu aplikácie aj pri dlhšom čase trvanie získania dát, napríklad pri testovaní zdrojového kódu.
- zamedzenie duplicitným požiadavkám z viacerých častí aplikácie naraz, knižnica dokáže takéto správanie odchytiť. Danú požiadavku na server vykoná len raz a distribuuje ju do všetkých častí aplikácie, ktoré o ňu žiadali.

Resizable

Resizable je javascriptová knižnica, ktorá obsahuje reactovský element pomocou, ktorého vieme vytvárať elementy tak, aby ich mohol ľubovolne veľkostne upravovať potiahnutím kurzoru myši na okraji vybraného elementu. Tento komponent je možne nastaviť ako upravovateľný vo vertikálnej respektíve horizontálnej dimenzií. Avšak tieto elementy vieme do seba ľubovolne vnárať a tak vieme dostať funkcionalitu podobnú veľkostne upravovateľnému gridu.

3.1.3 Podstránky

V tejto sekcií si povieme viac o základnom rozdelení aplikácie na jednotlivé podstránky. Povieme si, čo jednotlivé stránky obsahujú a na čo primárne slúžia

Login stránka

Login stránka obsahuje formulár pre prihlásenie do aplikácie, ako aj možnosť obnovy hesla pri zabudnutí.

Domovská stránka

Domovská stránka je prvá stránka, ktorú užívateľ po úspešnom prihlásení uvidí, obsahuje uvítací text a základné informácie o aplikácií.

Úlohy

Táto stránka zobrazuje zoznam všetkých úloh, ktoré sú užívateľovi v systéme dostupné v podobe ich zoznamu. Jednotlivé prvky zoznamu zobrazujú základne údaje o úlohe, akými sú názov, deadline, skrátený popis, stav úlohy (to, či už užívateľ úlohu úspešne odovzdal alebo nie).

Správa účtu

Stránka pre správu účtu obsahuje možnosť zmeny všetkých informácií o užívateľskom konte okrem e-mailovej adresy. Teda je možné zmeniť svoje meno, priezvisko alebo heslo.

3.1.4 Komponenty

Komponenty sú základnou stavebnou jednotkou každej aplikácie vytvorenej pomocou frameworku React.js. V nasledujúcej sekcií si priblížime práve tie najdôležitejšie pre fungovanie výslednej webovej aplikácie.

LoginPage

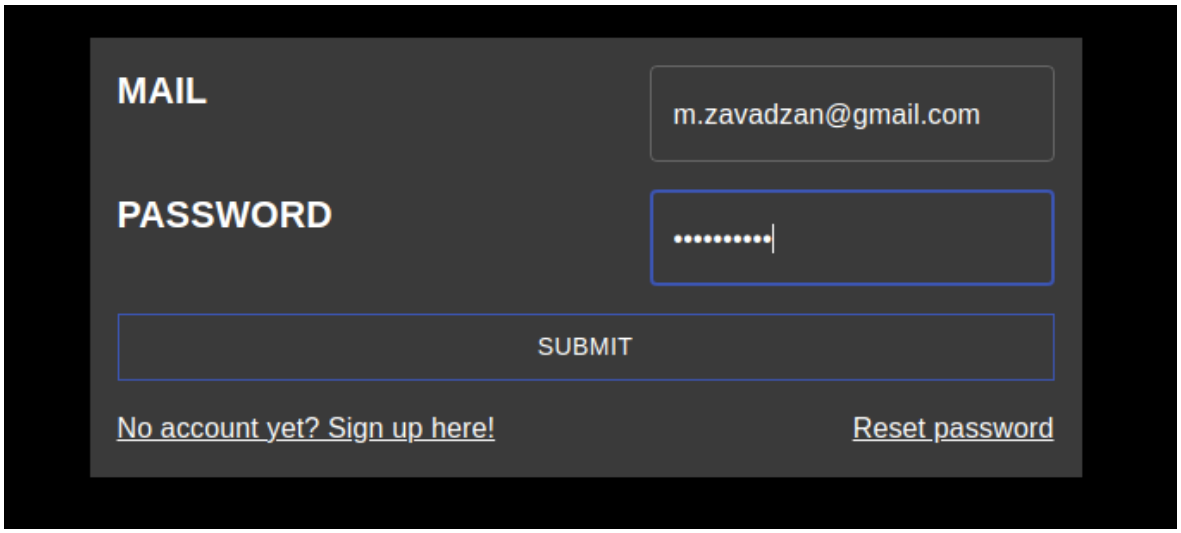
Tvorí celú kostru zobrazenú na stránke s loginom. Je poskladaný z viacerých menších komponentov, akými sú napríklad textové inputy alebo tlačidlá (vid obr.3.1). Po stlačení tlačidla na prihlásenie vykoná požiadavku na server s údajmi pre prihlásenie. Na základe úspešností prihlásenie buď zobrazí chybovú správu, alebo v prípade úspešného prihlásenia presmeruje užívateľa ďalej na úvodnú stránku webovej aplikácie.

ExercisesPage

Komponent, ktorý zobrazuje celý obsah podstránky s úlohou. Komponent spraví požiadavku na server, v ktorom si vyžiada všetky úlohy, ktoré sú pre daného užívateľa prístupné. Keď má tieto potrebné dáta, vygeneruje komponent ExerciseList, kde tieto dáta posiela ako parameter. (vid obr. 3.2)

ExerciseList

Ako argument očakáva zoznam typu Exercise. Na základe tohoto zoznamu v cykle generuje komponenty ExerciseListItem, ktoré ako parameter očakávajú práve objekt



Obr. 3.1: Login formulár

typu Exercise. Po kliknutí na ExerciseListItem komponent je užívateľ presmerovaný na stránku s konkrétnou úlohou.

ExercisePage

Zaobaľuje do seba všetky komponenty, ktoré sú renderované pre stránku s konkrétnou úlohou. Pri svojom renderovaní získa parameter **exerciseId** z URL adresy, ktorá je v tvare /**exercises**/**:exerciseId**. Na základe tohoto id získa potrebné dáta z Redux store, dáta sú tu už predfetchované, a dáta ďalej rozdelí svojim child komponentom, akými sú napríklad Monaco komponent, ktorý zobrazuje zdrojové kódy a testy alebo rôzne dropdown selecty pre zobrazenie príslušných verzií kódu. (vid obr. 3.3)

3.2 Backend

V nasledujúcej sekcií si povieme o samotnom fungovaní serverovej časti aplikácie, ktorá je tvorená Node.js serverom a PostgreSQL databázou. Node.js server je schopný reagovať na viac typov požiadaviek na rôzných endpointoch, o ich úlohe a samotnom fungovaní si viac povieme v nasledujúcej sekcií.

3.2.1 Spustenie serveru

V tejto sekcií si poviem, ako prebieha samotné prvotné spustenie serveru a aké úkony sú pri ňom vykonané.

Vytváranie docker kontajnerov

Pri spustení serveru je potrebné vytvoriť predom stanovený počet docker kontajnerov tzv. workerov. Títo workeri budú zaobstarávať samotné kompilovanie zdrojového kódu a jeho následné testovanie pomocou sady unit testov. Workeri sú pri spustení pridaní do worker poolu, kde čakajú na pridelenie úlohy.

3.2.2 Testovanie užívateľského zdrojového kódu

Keďže o zdrojovom kóde od užívateľa pred jeho spustením nič nevieme, je potrebné maximánle zamedziť možnosti potencionálneho bezpečnostného rizika. Kód by mohol v sebe obsahovať škodlivé časti, ktoré by mohli narušiť alebo až znemožniť beh nášho serveru. Z tohoto dôvodu je potrebné nejakým spôsobom odčleniť prostredie, v ktorom beží server a prostredia, ktoré budú použité na testovanie. Na tento účel som sa rozhodol zvoliť docker kontajnery a to hneď z niekoľkých dôvodov:

1. v porovnaní s virtuálnou mašinou sú menej náročne na zdroje
2. poskytujú dostatočnú izolovanosť
3. bežia priamo nad OS hostiteľského zariadenia
4. sú pomerne jednoduché na spustenie vďaka docker imagom
5. predchádzajúce skúsenosti

Testovanie je zabezpečené funkciou **passTaskToWorker**, ktorá berie ako parameter požiadavku obsahujúcu zdrojové kódy a informácie o príslušnej úlohe a objekt, ktorý uchováva potrebne údaje pre zaslanie odpovede. Funkcia sa pokúsi vziať prvého voľného workera z worker poolu, ak je niektorý z workerov práve v poole a je teda voľný posiela ho ďalej ako parameter funkcii **runTests** spolu s oboma objektami, ktoré funkcia dostala ako parameter. V situácií kedy nie je žiaden z worekrov voľný vloží požiadavku do queue, kde požiadavka čaká na voľného workera.

Funkcia **runTests** získa potrebné dáta z databázy, unit testy uložené v databáze, a vytvorí potrebné súbory pre kompiláciu a testovanie. Spustí príkaz pre vykonanie testov, po ich dokončení vráti ich výsledky ako svoju návratovú hodnotu, tá je nasledovne odoslaná ako odpoveď na frontendovú časť aplikácie, kde je ďalej zobrazená ako výstup testov pre daný zdrojový kód, ktorý bol zaslaný v požiadavke. Testovanie môže skončiť v 4 stavoch a to v takom, že testy boli vykonané, či už úspešne alebo nie. V takomto prípade sú výsledky testov zaslané ako odpoveď. Ďalšou z možností je, že sa zdrojový kód nepodarilo skompilovať, a teda nastala kompilačná chyba. V takomto prípade je presné znenie chyby zaslané ako odpoveď. V poslednom prípade, je možne, že sa testy nepodarilo vykonať v čase stanovenom na ich vykonanie, v našom

prípade to je 30 sekúnd, vtedy je testovanie ukončené a ako odpoveď je odoslaná správa, ktorá oboznamuje užívateľa o tomto probléme. Tento problém môže nastať napríklad pri nekonečnom cykle v užívateľskom kóde, alebo pri riešení, ktoré nie je dostatočne efektívne.

3.2.3 Serverové endpointy

- /login - už ako je zo samotného názvu jasné, tento endpoint m za úlohu autorizovať prihlásenie užívateľa do aplikácie. Endpoint overí správnost prihlasenia (e-mail, heslo) s údajmi v databáze a pridelí užívateľovi access token, na základe, ktorého sa overuje nasledovný prístup k údajom v ostatných požiadavkách od užívateľa. Taktiež udáva privilégia užívateľovi (rola admin, resp. užívateľ)

- /changePassword - slúži na zmenu hesla autorizovanému užívateľovi

- /exercise - slúži na pridanie resp. insertnutie nového cvičenia do systému ale aj na získanie údajov o konkrétnom cvičení na základe jeho id.

- /exercise - získanie všetkých cvičení zo systému, ktoré sú dostupné pre daného užívateľa

- /saveExercise - slúži na uloženie aktuálneho stavu rozpracovanej úlohy do databázy

- /savedExercise - slúži na získanie stavu rozpracovanej úlohy a teda umožňuje aplikacií nahrať stav úlohy tak, ako ju užívateľ zanechal pri poslednom uložení

- /results - vráti doteraz odovzdané úlohy aj s ich kódom v jazyku C++ a aj s výsledkami testov. Taktiež ich prípadný úspešný respektíve neúspešný prechod testami

- /test - slúži na prijímanie požiadaviek na otestovanie zdrojového kódu na základe id cvičenia, ktoré musí byť obsiahnuté v samotnom tele požiadavky získa testy pre danú úlohu. Tieto testy spolu so zdrojovým kódom, ktorý je tiež obsiahnutý v tele požiadavky, poskytne funkcií passTaskToWorker, ktorú sme si už popísali. Výsledok z tejto funkcie vráti užívateľovi v podobe odpovede.

- /submitTest - tento endpoint sa od /test líši len v tom, že výsledok uchováva v databáze

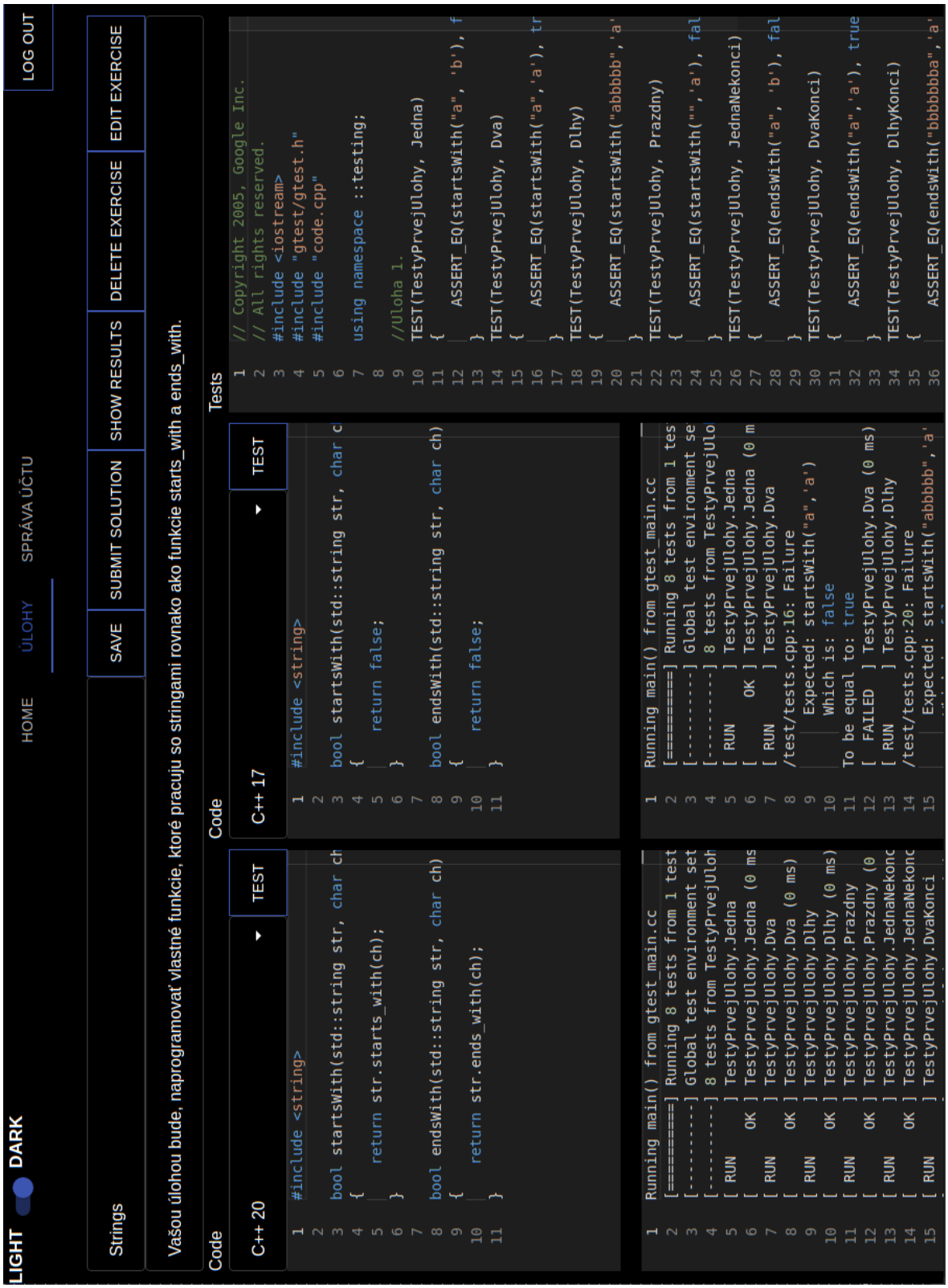
3.3 Bezpečnosť

V tejto sekcií si povieme viac o riešení zabezpečenia samotnej aplikácie. Keďže cela frontendová časť aplikácie beží na počítači užívateľa, je potrebné nejakým spôsobom zabezpečiť kontrolu požiadaviek, ktoré prichádzajú z frontendu na serverovú časť, teda backend. To sa deje hneď niekoľkými spôsobmi:

3.3.1 Autorizácia užívateľa

Pri každej požiadavke na server je nutné v hlavičke požiadavky poskytnúť access-token, ktorý slúži na autorizáciu užívateľa. Nesie v sebe informáciu o tom, o ktorého z registrovaných užívateľov v systéme ide. Na základe tohoto je taktiež možné rozoznať, o aký typ užívateľa ide a to teda, či užívateľ je respektíve nie je adminom a teda či má právo využívať niektoré nadštandardné služby, akou je napríklad pridávanie nových úloh do aplikácie. Tento token užívateľ získa po prihlásení do systému zadaním správnej e-mailovej adresy a správneho hesla. Access-token sa vygeneruje na serverovej časti pomocou javascriptovej knižnice JWT (JSON Web Tokens) a je zaslaný späť užívateľovi, ktorý ho následne využíva na autorizáciou svojich požiadaviek. Tato činnosť sa deje automaticky na frontendovej časti aplikácie.

Obr. 3.2: Stránka so zoznamom úloh



Obr. 3.3: Stránka pre konkrétnu úlohu

Kapitola 4

Výuka

V nasledujúcej kapitole sa zameriame na samotný spôsob výučby pomocou našej webovej aplikácie. Detailne si povieme o nových vlastnostiach v jednotlivých verziách jazyka, ktoré som vybral ako vhodné pre tvorbu úloh na demonštráciu funkcionality.

4.1 Nové vlastnosti jazyka C++

V tejto sekcií si priblížime vývoj jazyka C++ a pozrieme sa na nové vlastnosti jazyk, ktoré som si vybral ako vhodné na tvorbu ukážkových úloh za účelom demonštrácie funkcionality aplikácie.

4.1.1 Verzia 11

Verzia jazyka C++11 je druhá veľká verzia a najdôležitejšie verzia od verzie C++98. Priniesla so sebou veľké množstvo zmien a môžme povedať, že išlo o zlomovú verziu jazyka. Zaviedli sa zmeny s cieľom štandardizovať existujúce postupy a zlepšiť abstrakciu ponúkanú jazykom C++.

4.1.2 Verzia 14

Verzia jazyka C++14 je menšia verzia jazyka, ktorá nasledovala po verzií C++11. V tejto verzií sa kládol dôklad hlavne na malé vylepšenia jazyka, ale aj na opravu niektorých chýb, ktoré sa časom objavili. Jednou z výraznejších zmien bolo zavedenie generického lambda kalkulu.

4.1.3 Verzia 17

Po verzií C++14 prišla o niečo rozsiahlejšie verzia a to verzia C++17. Priniesla so sebou opravu, respektíve vylepšenie niektorých časti jazyka. Priniesla však aj nové prvky, akým je napríklad zavedenie fold expressions.

4.1.4 Verzia 20

Po takmer 10 rokoch od vydania C++11 prišla verzia C++20, ktorá v porovnaní s predchádzajúcimi dvoma verziami (C++14, C++17), ktoré priniesli len drobné úpravy a vylepšenia, priniesla aj viacero signifikantných vylepšení a zmien ako napríklad zavedenie corutín a nové postupy pri importovaní kódu pomocou modulov.

4.2 Cvičenia

Táto kapitola nás prevedie ilustračnými úlohami pre ukážku funkcionality aplikácie. Úlohy sú usporiadané tak, aby ich náročnosť postupne stúpala. Keďže úlohy majú byť cielene na precvičenie a osvojenie nových vlastností jazyka C++, vždy sú zamerané na funkcionalitu, ktorá mi prišla vhodná pre demonštráciu a taktiež je vhodná na testovanie pomocou unit testov, keďže pri niektorých vlastnostiach jazyk je to nemožné.

4.2.1 C++20 strings

Úloha je zameraná na osvojenie nových funkcií pracujúcich so stringami. Funkcie, ktoré som pre túto úlohu vybral, sú nasledovné:

- starts_with - funkcia berie ako argument charakter a v prípade, že daný string začína rovnakým charakterom, vráti hodnotu true, inak vráti hodnotu false.
- ends_with - funkcia berie ako argument charakter a v prípade, že daný string končí rovnakým charakterom vráti hodnotu true, inak vráti hodnotu false.

Úlohou študenta bude naprogramovať funkcie, ktoré pracujú so stringami rovnako ako samotné už vytvorené funkcie vo verzií C++20. Keďže kód napísaný študentom bude skompilovaný v niektorej z nižších verzií jazyka, v tomto prípade verzií C++17, bude musieť dané funkcie naprogramovať sám. Cieľom tohto cvičenia je oboznámiť študentov s tým, že takéto funkcie vôbec vo verzií C++20 už existujú a tak predísť tomu, aby ich v budúcnosti sami pracne programovali. Toto by mohlo viesť k zhoršeniu kvality kódu alebo aj k prípadným chybám pri nesprávnej implementácií danej funkcionality. Študent si taktiež vyskúša naprogramovať túto funkciu a tak ho to prinúti rozmýšľať nad tým, ako samotná funkcia pracuje so stringom na pozadí.

4.2.2 Spaceship operátor

Úloha je zameraná na zoznámenie s novým trojsmerným operátorom <=>, ktorý sa zvykne označovať aj pomenovaním spaceship operátor pre jeho vzhľad. Operátor vracia jednu z nasledujúcich hodnôt ako výsledok porovnania a <=> b:

- less - ak je a menšie ako b
- greater - ak je a väčšie ako b
- equal - ak sa a rovná b
- unordered - ak je jedna z premenných a respektíve b NaN a teda nie je číslo

Pri riešení úlohy študent bude pracovať s kódom, ktorý bude obsahovať funkciu s názvom foo, tento názov je zámerne vybraný tak, aby samotné pomenovanie funkcie neprezradilo, ako funkcia funguje, keďže to bude mať študent za úlohu zistiť. Študent taktiež vidí priložené unit testy, ich vstupy a očakávané výstupy. Pomocou toho by mal byť schopný odhaliť, ako samotná funkcia funguje a teda aj to, ako funguje tento nový operátor, keďže funkcia s ním pracuje. Potom bude mať za úlohu napísať vlastnú verziu funkcie foo, ktorá bude fungovať úplne rovnako a teda prejde všetkými priloženými unit testami. Pri riešení však študent nebude môcť použiť spaceship operátor, keďže ten nebude dostupný vo verzií jazyka c++, v ktorej má výsledný kód napísať. Preto bude musieť samotné správanie operátora naprogramovať sám.

4.2.3 Práca s bitmi

Úloha je navrhnutá tak, aby študentov oboznámila s novými funkciami, ktoré boli pridané na prácu s bitovými číslami. Takých funkcií bolo do jazyka pridaných hneď niekoľko. Pre túto úlohu som sa rozhodol vybrať päť z nich:

- countl_one - spočíta počet za sebou idúcich jednotiek z ľavej strany čísla
- countr_one - spočíta počet za sebou idúcich jednotiek z pravej strany čísla
- countl_zero - spočíta počet za sebou idúcich núl z ľavej strany čísla
- countr_zero - spočíta počet za sebou idúcich núl z pravej strany čísla
- popcount - spočíta počet jednotiek v bitovom zápise čísla

Úlohou študenta bude pomocou priložených testov zistiť, ako tieto funkcie pracujú, prípadne si môže ich funkcionalitu naštudovať z dokumentácie. Potom bude jeho úlohou naprogramovať vlastné funkcie, ktoré pracujú rovnako a tým sa teda zamyslieť nad tým, ako môžu byť tieto funkcie naprogramované v oficiálnej knižnici.

4.2.4 Fold

Nasledujúca úloha je zameraná na prácu s fold expressions. Študent vidí riešenie úlohy pomocou fold expression a jeho úlohou je zreplikovať funkcionalitu tak, aby funkcia

fungovala aj vo verzií C++14. V tej totiž fold expressions ešte nie sú podporované. Ako príklad som vybral funkciu sum, ktorá berie ako parameter ľubovolný počet čísel a ako návratovú hodnotu vráti ich súčet. Toto je ľahko naprogramovateľné práve pomocou foldu. Študent však bude pravdepodobne nútený použiť rekurziu, na to aby takéto správanie docielil.

Záver

Cieľom tejto bakalárskej práce bolo zhotoviť fungujúcu webovú aplikáciu, ktorá umožní učiteľovi pripravovať úlohy, ktoré umožnia študentom s ľahkosťou pochopiť a priblížiť si jednotlivé nové vlastnosti jazyka C++ vo vybraných verziách za pomoci agilných metód programovania. Aplikácia by mala byť intuitívna a teda by sa mala ľahko používať. Samotný webový editor zdrojového kódu by mal byť príjemný na použitie a poskytnúť užívateľom podobný komfort ako desktopové editory zdrojového kódu.

Tieto ciele sa podarilo splniť a výsledná webová aplikácia teda ponúka interaktívne prostredie na tvorbu rozličných úloh zameraných na oboznámenie sa s novými vlastnosťami jazyka C++ pomocou agilných metód programovania. Stránka ďalej ponúka možnosť tieto úlohy riešiť priamo v integrovanom editore zdrojového kód. Kód je možne otestovať a získať výsledky testov.

Ako väčšina technológií tak aj táto ma určitý potenciál na prípadné zlepšenie. Ako jedno z nich vidím rozšírenie aplikácie o iné programovacie jazyky, akým je napríklad Java. Taktiež by bolo možné aplikáciu rozšíriť o ďalšie nove typy úloh alebo rozšíriť aplikáciu aj o lekcie, ktoré by popisovali nové vlastnosti jazyka, tým by aplikácia neslúžila len na samotné riešenie programátorských úloh, ale aj ako miesto, kde by užívateľ bol schopný naštudovať potrebné informácie pre vyriešenie úlohy, a teda by mal všetko potrebne "na jednom mieste". V neposlednom rade by bolo možné optimalizovať samotný beh aplikácie na strane servera a tým zlepšiť rýchlosť kompilácie, behu a testovania užívateľského kódu. Toto by malo za dôsledok zlepšenie efektivity výučby, keďže by študent strávil menej času čakaním na výsledky testov a viac času samotnou prácou na úlohách.

Literatúra

[1] About c++11. https://en.cppreference.com/w/cpp/11. Naposledy zobrazené 2022-06-01.

[2] About node.js. https://nodejs.org/en/about/. Naposledy zobrazené 2022-06-01.

[3] Docker architecture. https://www.aquasec.com/cloud-native-academy/docker-container/docker-architecture/. Naposledy zobrazené 2022-06-01.

[4] Express. https://expressjs.com/. Naposledy zobrazené 2022-06-01.

[5] Getting started. https://redux.js.org/, January. Naposledy zobrazené 2022-06-01.

[6] Googletest guide. https://google.github.io/googletest/primer.html. Naposledy zobrazené 2022-06-01.

[7] Tiobe index. https://www.tiobe.com/tiobe-index/. Naposledy zobrazené 2022-06-01.

[8] What is docker container. https://www.docker.com/resources/what-container/. Naposledy zobrazené 2022-06-01.

[9] What is tdd. https://www.whizlabs.com/blog/what-is-tdd-and-its-phases/. Naposledy zobrazené 2022-06-01.

[10] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, , and Dave Thomas. Manifesto for agile software development. http://agilemanifesto.org/. Naposledy zobrazené 2022-06-01.

[11] Lesse Koskela. *Test driven: practical TDD and acceptance TDD for Java developers*. Manning, 2008.

[12] Tím kyklop. *Tvorba softvéru v treťom tisícročí*. 2002. [Citované 2021-10-01] Dostupné z `http://www2.fiit.stuba.sk/~bielik/courses/msi-slov/kniha/2002/kyklop.pdf`.

[13] Jeff Langr. *Modern C++ programming with test-driven development: code better, sleep better*. Pragmatic Programmers, 2013.

[14] Roy Osherove. *The art of unit testing: with examples in C#*. Manning, 2014.