Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Portál kolektívnej inteligencie

Bakalárska práca

2022
Róbert Vangor

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Portál kolektívnej inteligencie

Bakalárska práca

Študijný program:	Aplikovaná Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	RNDr. Peter Borovanský, PhD.

Bratislava, 2022
Róbert Vangor

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Róbert Vangor
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: Portál kolektívnej inteligencie
Collective Intelligence System

Anotácia: Práca navrhne a implementuje klient-server aplikáciu otázok a odpovedí študentov. Backend bude postavený na technológiach Spring, Hibernate, Postgre SQL, frontend bude Angular. Užívateľské rozhranie umožní zadávanie matematického textu v LaTeX.

Cieľ: Cieľom práce je vytvorenie portálu UNIBASK pre zvýšenie kolektívnej inteligencie na základe vzájomného predávania si informácií medzi študentmi. Systém integruje do kategórií otázky a odpovede študentov riešiacich rôzne študijné problémy. Všetky otázky a odpovede sú študentami hodnotené, pričom ich hodnotenie sa odzrkadľuje na reputácii študentov v systéme. Pre zvýšenie zapojiteľnosti je študentom umožnené anonymne klásť otázky a odpovedať. Komunita je uzavretá pre študentov Univerzity Komenského. Text otázok a odpovedí dovolí vyjadrovať matematické rovnice alebo kód v rôznych programovacích jazykoch. Systém by mal byť členený do rozumných a pre užívateľov ľahko pochopiteľných a navigovateľných kategórií. Mal by taktiež obsahovať nejakú formu notifikácií na základe užívateľovho výberu. Systém musí tiež riešiť spôsob, akým v komunite zachovať základy slušného správania. Prvé experimentálne nasadenie systému pre zozbieranie feedbacku by malo prebehnúť v letnom semestri 2021/2022.

Vedúci: RNDr. Peter Borovanský, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 14.09.2021

Dátum schválenia: 30.09.2021

doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Touto cestou by som sa chcel poďakovať svojmu školiteľovi RNDr. Petrovi Borovanskému, PhD. za jeho ochotu a cenné rady. Taktiež by som sa chcel poďakovať svojím blízkym, ktorí mi pomáhali pri testovaní aplikácie.

Abstrakt

Portál kolektívnej inteligencie Unibask je webová aplikácia, ktorá umožňuje študentom pýtať sa a odpovedať na otázky. Používatelia vedia otázky a odpovede hodnotiť a pomôcť tak rozlíšiť dobré príspevky od tých zlých. Hodnotenie príspevkov používateľov sa odzrkadľuje na ich celkovej reputácií v používateľskom rebríčku. Otázky sú triedené podľa kategórií. Používatelia vedia kategórie sledovať a dostať tak rýchly prístup ku otázkam ich záujmu. Môžu sa rozhodnúť pre zapnutie upozornení na pribudnutie novej otázky v nejakej kategórií alebo na pribudnutie nového príspevku v nejakej otázke. Upozornenia si môže nechať zasielať aj prostredníctvom emailovej adresy. Aplikácia využíva moderné technológie, poskytuje intuitívny dizajn a je optimalizovaná pre všetky veľkosti obrazoviek. Portál bol ostro nasadený vo výučbe predmetu 'Programovanie 4' katedry aplikovanej informatiky v letnom semestri školského roku 2021/2022.

Kľúčové slová: otázky a odpovede, fórum, stackoverflow, webová aplikácia, kolektívna inteligencia

Abstract

Unibask — portal of collective intelligence — is a web application, which allows students to ask and answer questions. The quality of both questions and answers is decided by users, who can rate them positively or negatively. The reputation of a user in the user's leader board is calculated from a rating of his entries. Questions are sorted by categories. Users can follow a category, which allows them to have quick access to questions that are of his interest. Users can decide to turn on notifications for when a new question is created in some category or for when a new entry is created in some question. They can also opt-in for the option of having notifications delivered to them through email. Application is using modern technologies, intuitive design and is optimized for all screen sizes. Portal was deployed in a course 'Programming (4)' of the Department of Applied Informatics during the summer semester of the school year 2021/2022.

Keywords: questions and answers, forum, stackoverflow, web application, collective intelligence

Obsah

Zoznam obrázkov

Zoznam výpisov

Úvod

V dnešnom svete hrá medziľudské zdieľanie informácií obrovskú rolu. Príchod internetu nám poskytol nové možnosti odovzdávania svojich vedomostí ďalším ľuďom. Ľudia už nie sú limitovaní na svoje okolie a spojiť sa môžu s hocikým na svete. Ak hľadáme riešenie nejakého problému, existuje veľká šanca, že niekto sa s daným problémom už stretol a vyriešil ho. Po zadaní problematiky do internetového vyhľadávača sa nám pravdepodobne zobrazí niekoľko relevantných výsledkov z rôznych fór alebo stránok. Čo však vtedy, ak sa tak nestane?

Na svete existuje mnoho portálov určených na vyhľadávanie otázok a odpovedí. Či už je človek programátor, grafik, matematik alebo filozof – existuje priestor, kde môže svoju otázku položiť. Je veľká pravdepodobnosť, že na danú otázku dostane aj odpoveď. Ku februáru roku 2015 bolo na stránke Stack Overflow položených 8.88 miliónov otázok. Približne 92% otázok bolo aj zodpovedaných. [1] V priemere bola otázka zodpovedaná do 11 minút. [2] Odpoveď akceptovaná pýtajúcim sa používateľom bola publikovaná v priemere do 24 dní. [3]

No stále môže nastať problém, keď sa chceme opýtať otázku špecifickú pre neverejnú skupinu ľudí. Takýto prípad môže nastať vtedy, ak nás zaujímajú napríklad administračné veci danej skupiny, alebo máme iné otázky, ktoré sú súčasťou iba našej komunity, a nie verejnej domény. V tomto prípade by nám pomohol systém prispôsobený špeciálne pre potreby danej komunity, kde by sa sústreďovali jej členovia. Takýto systém umožňuje väčší priestor zdieľania informácií v rámci komunitného kontextu.

Ak by nasadenie do prevádzky a vytvorenie dostatočne veľkej komunity bolo úspešné, boli by sme schopní zlepšiť kolaboráciu medzi študentmi a vyučujúcimi, a taktiež medzi študentmi navzájom. Poskytli by sme ďalšiu možnosť čerpania informácií – zo samotnej komunity študentov. Študenti, ktorí nejaký predmet už absolvovali, z neho väčšinou majú dostatočné vedomosti na to, aby pomohli svojim spolužiakom v nižších ročníkoch. Odbremenení by takto boli aj vyučujúci, keďže mnoho otázok je zodpovedateľných aj vedomostne zdatnými študentami. Niektoré otázky sa taktiež každoročne opakujú a študent si vie dohľadať svoju otázku v histórií - nie je teda potrebné znovu položiť

rovnakú otázku.

Náplňou tejto bakalárskej práce je vytvorenie takéhoto systému na úrovni univerzity a jeho zasadenie do prevádzky. Jeho úspešnosť bude závislá od vôle študentov adoptovať nové veci. Systém môže byť užitočný iba v prípade, ak bude využívaný, a na otázky bude mať kto odpovedať.

Podobný systém s názvom Askalot bol vytvorený na Fakulte informatiky a informačných technológií Slovenskej technickej univerzity v Bratislave pod vedením Ing. Ivan Srbu, PhD. (kapitola 1.1.6). Tento systém mimo zodpovedania početného množstva otázok študentov pomohol vyprodukovať niekoľko vedeckých výstupov.

Jeden z nich sa zaoberal perspektívou 182 študentov informatiky z troch univerzít v troch rôznych krajinách na nasadený systém Askalot. [4] V piatich akademických rokoch (počnúc rokom 2013/2014) bolo počas využívania Askalotu na FIIT STU zaregistrovaných 1960 užívateľov. Systém sa aktívne využíval v 51 kurzoch a ponúkal možnosť klásť aj otázky v kategóriách ako je vedenie fakulty. Položených bolo 1176 otázok, zodpovedaných 1553-krát, okomentovaných 1608-krát, uložených bolo 9042 hodnotení a počet kumulatívnych pozretí otázok sa rovnal číslu 255 351. Systém bol nasadený taktiež na Fakulte vied Univerzity v Novom Sade (Srbsko) a Fakulte informatiky University v Lugane (Švajčiarsko). V Novom Sade sa jednalo o jeden kurz a participanti boli ocenení bonusovými bodmi k ich finálnej známke. 89 užívateľov položilo 7 otázok, odpovedalo 23 odpoveďami a 9 komentármi, hodnotilo 52 hlasmi a dokopy si prezreli 1357 otázok. V Lugane sa jednalo o 3 kurzy. 77 užívateľov položilo 71 otázok, odpovedalo 70 odpoveďami a 16 komentármi, hodnotilo 35 hlasmi a dokopy si prezreli 1675 otázok.

V dotazníku si študenti mali vybrať preferujúci spôsob online komunikácie. 62.31% si zvolilo komunikáciu v reálnom čase prostredníctvom chatovacích aplikácií, 30% si zvolilo Askalot a 7.69% študentov si zvolilo už existujúce štandardné nástroje v rámci univerzity, akým je napríklad Moodle (vzorka 130 študentov).

Na základe týchto zistení by sme mohli predpokladať, že existuje priestor pre zlepšenie štandardných spôsobov on-line komunikácie, ktoré sú ponúkané univerzitami. Tým sa častokrát nevenuje dostatok času a mnoho študentov ani nevie o ich existencii. V čase písania tejto práce vo svete pretrváva pandémia vírusu COVID-19 a výučba prebieha už druhý rok on-line formou. To nám dodalo impulz na pokus o zlepšenie sféry on-line komunikácie a kolaborácie medzi študentmi a ich vyučujúcimi. Zakomponovať plnú funkcionalitu Askalotu sa s najväčšou pravdepodobnosťou nepodarí. Vývoj nášho portálu prebieha v jednočlennom tíme počas jedného semestra – na rozdiel od

portálu Askalot, ktorý bol vytváraný niekoľkočlenným vývojovým tímom. Pokúsime sa však zakomponovať esenciálnu funkcionalitu a spraviť portál čo najviac priateľský pre používateľa. To, či sa portál uchytí alebo nie, už bude závisieť od rôznych, nami nie veľmi ovplyvniteľných faktorov. Tými sú napríklad aktivita študentov, zavádzanie portálu vyučujúcimi a propagácia portálu fakultou.

Kapitola 1

Analýza

1.1 Existujúce riešenia

V tejto kapitole si zhrnieme už existujúce riešenia komunikácie medzi študentami. Riešenia zoradíme v poradí od najmenej po najviac podobné tomu nášmu a uvedieme si ich výhody a nevýhody.

1.1.1 E-mailová komunikácia

Najštandardnejší spôsob on-line komunikácie na univerzitách je prostredníctvom e-mailov. Je to najčastejšie preferovaný spôsob komunikácie študenta s vyučujúcim. Ide však viac o súkromnú komunikáciu ako o tú verejnú. E-mailová komunikácia ponúka možnosť hromadnej správy alebo zasielania kópií, no nie je veľmi vhodná na vedenie kolektívneho dialógu. Zatiaľ čo e-mail ponúka rýchly a efektívny spôsob komunikácie, kolaboráciu umožňuje iba na úrovni niekoľkých osôb.

1.1.2 Chatovacie platformy

Takmer každý študent využíva istú formu instantnej komunikácie. Medzi najznámejšie riešenia v tejto kategórií patrí Facebook, WhatsApp, Instagram alebo Apple iMessage. Ich najväčšou výhodou je vysoká adaptabilita medzi študentmi. Ich používanie je intuitívne a jednoduché.

Študenti môžu medzi sebou zefektívniť komunikáciu vytvorením skupiny. Tu však nastáva problém, ktorým je fragmentácia. Tieto skupiny sú častokrát veľmi malé. Väčšinou sú rozdelené podľa kamarátstiev, predmetov alebo študijných skupín. Častokrát neexistuje jedna skupina, v ktorej sa nachádzajú všetci študenti, ktorí tam patria. Introvertnejšie založení študenti môžu mať problém dostať sa do akejkoľvek skupiny. Ďalším problémom je taktiež tendencia študentov riešiť v týchto skupinách veci, ktoré

sa netýkajú ich štúdia, a mnohokrát vytvárajú zbytočný spam. To môže viesť ostatných k odídeniu zo skupiny alebo vypnutiu upozornení pre danú skupinu.

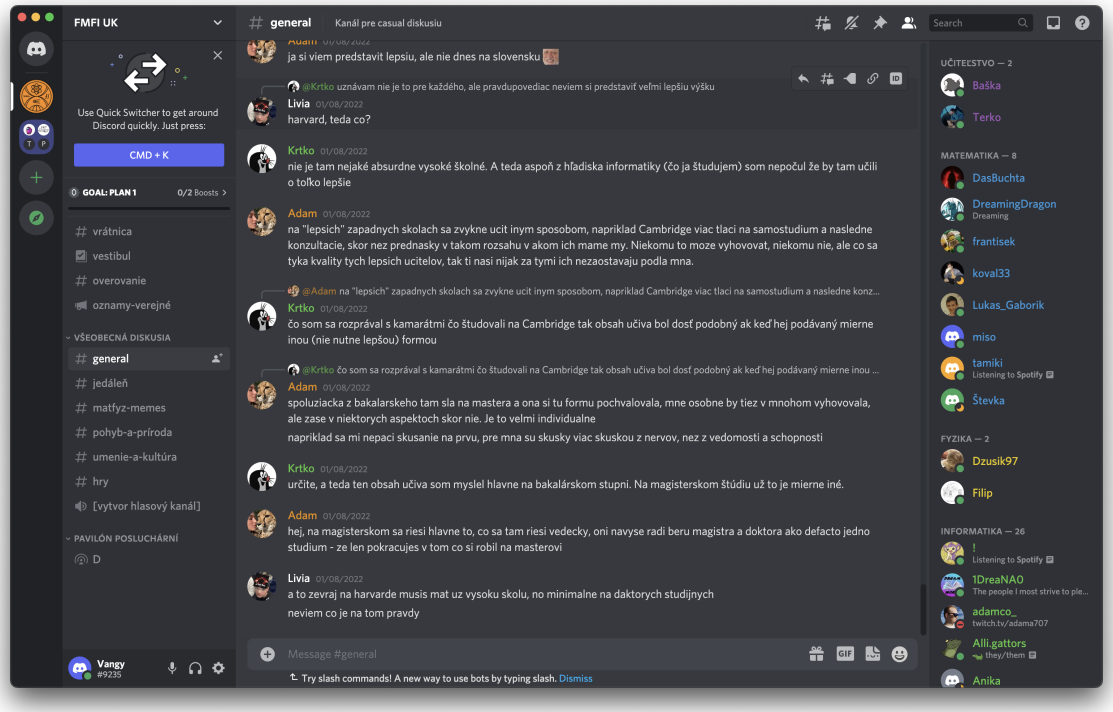
1.1.3 Microsoft Teams

Microsoft Teams [5] je komunikačná platforma vhodná pre väčšie skupiny ľudí. Používatelia môžu vytvárať tímy, ktoré slúžia ako organizačné jednotky. Tím tvorí skupina ľudí, ktorých spája rovnaký záujem. Jeden tím môže obsahovať niekoľko kanálov rozdelených podľa účelu. Každý kanál obsahuje archivovanú textovú konverzáciu a poskytuje možnosť začatia mítingu. V mítingu môžu pripojení používatelia komunikovať prostredníctvom audia, videa alebo zdieľania svojej obrazovky (prezentovania). Taktiež je im umožnené poskytnúť niekomu vzdialený prístup ku svojej ploche. Používatelia môžu komunikovať aj súkromne mimo tímu.

Jedná zo sfér, kde táto platforma nadobudla popularitu počas pandémie Covid-19, je tá akademická (Microsoft Teams for Education). Väčšinou platí, že jeden kurz = jeden tím a jedna prednáška = jeden míting. Tu vidíme výhodu v tom, že študenti sú zakomponovaní do jednej veľkej skupiny v rámci kurzu, a teda majú hromadný prístup k ostatným študentom rovnakého kurzu. Otázky sa v kanáloch môžu pýtať aj v prípade, že neprebieha prednáška. Z našej skúsenosti to však robiť nezvyknú a komunikujú majoritne iba počas prebiehajúcej prednášky. Výhodou je možnosť archivovať už odučené kurzy, čo môže študentom budúcich kurzov pomôcť v hľadaní odpovedí na ich otázky. Ako nevýhodu vidíme nedostatočné oddelenie otázok od ostatnej diskusie a neexistenciu reputácie používateľov.

1.1.4 Discord

Discord [6] je platforma umožňujúca používateľom komunikovať pomocou textových správ a audio/video hovorov. Môžu spolu zdieľať rôzne súbory a je im umožnené zdieľať aj obrazovku svojho zariadenia. Používatelia majú možnosť komunikovať privátne alebo pripojením sa na komunikačný server pomocou pozvánky. Takýto server vie vytvoriť aj sám používateľ. Server sa skladá z viacerých miestností, ktoré sú buď textové alebo hlasové. Miestnosť môže byť aj súkromná, teda prístupná iba zvoleným používateľom. Každý server má administrátora, ktorý ho spravuje. Táto platforma má medzi študentami vysokú adaptabilitu, ktorú si vybudovala najmä počas dištančnej výučby. Svoj vlastný Discord server má aj naša fakulta. O jeho spustenie sa postarali študentskí senátori. Je využívaný skôr na rekreáciu ako na komunikáciu ohľadom štúdia. Ako nevýhodu Discordu vidíme to, že komunikácia prebieha v chatoch a dohľadať niečo konkrétne je problematické. Neponúka ideálnu štruktúru pre pýtanie sa otázok, ktoré by boli archivované a jednoducho prístupné budúcim študentom.



Obr. 1.1: Rozhranie aplikácie Discord

1.1.5 Stack Exchange

Stack Exchange [7] je sieť webstránok zaoberajúcich sa otázkami a odpoveďami na rôzne témy. Skladá sa z niekoľkých komunít a každá je určená inému účelu. Medzi najznámejšiu z nich patrí Stack Overflow, ktorá je venovaná primárne programátorom. K roku 2022 bolo v Stack Overflow komunite položených viac ako 22 miliónov otázok. Ďalšími komunitami sú napríklad Mathematics – určená pre otázky týkajúce sa matematiky, Graphic Design – určená pre otázky týkajúce sa grafického dizajnu alebo Physics – určená pre otázky ohľadom fyziky. Stack Exchange umožňuje používateľom pýtať sa a odpovedať na otázky. Na rozdiel od bežného diskusného fóra tu musí každé vlákno začínať otázkou a každá odpoveď sa musí snažiť reagovať na túto otázku. Jediný priestor pre diskusiu je sekcia komentárov pod otázkou alebo odpoveďou. Otázky aj odpovede sú hodnotené ostatnými používateľmi buď kladne alebo záporne. Toto hodnotenie sa odzrkadľuje na reputácií autora príspevku.

Reputácia sa dá nadobudnúť aj získavaním rôznych odznakov či napísaním odpovede, ktorá je akceptovaná ako riešenie otázky. Vo všeobecnosti platí, že čím má užívateľ vyššiu reputáciu, tým majú jeho príspevky väčšiu kredibilitu. S väčšou reputáciou používateľom pribúdajú aj iné oprávnenia. Pri dosiahnutí určitej reputácie používateľ dostane kompetencie na editáciu alebo dokonca vymazanie príspevkov pridaných ostatnými používateľmi. To umožňuje stránke moderovať sa primárne samostatne bez

Obr. 1.2: Ukážka položenej otázky - portál Stack Overflow

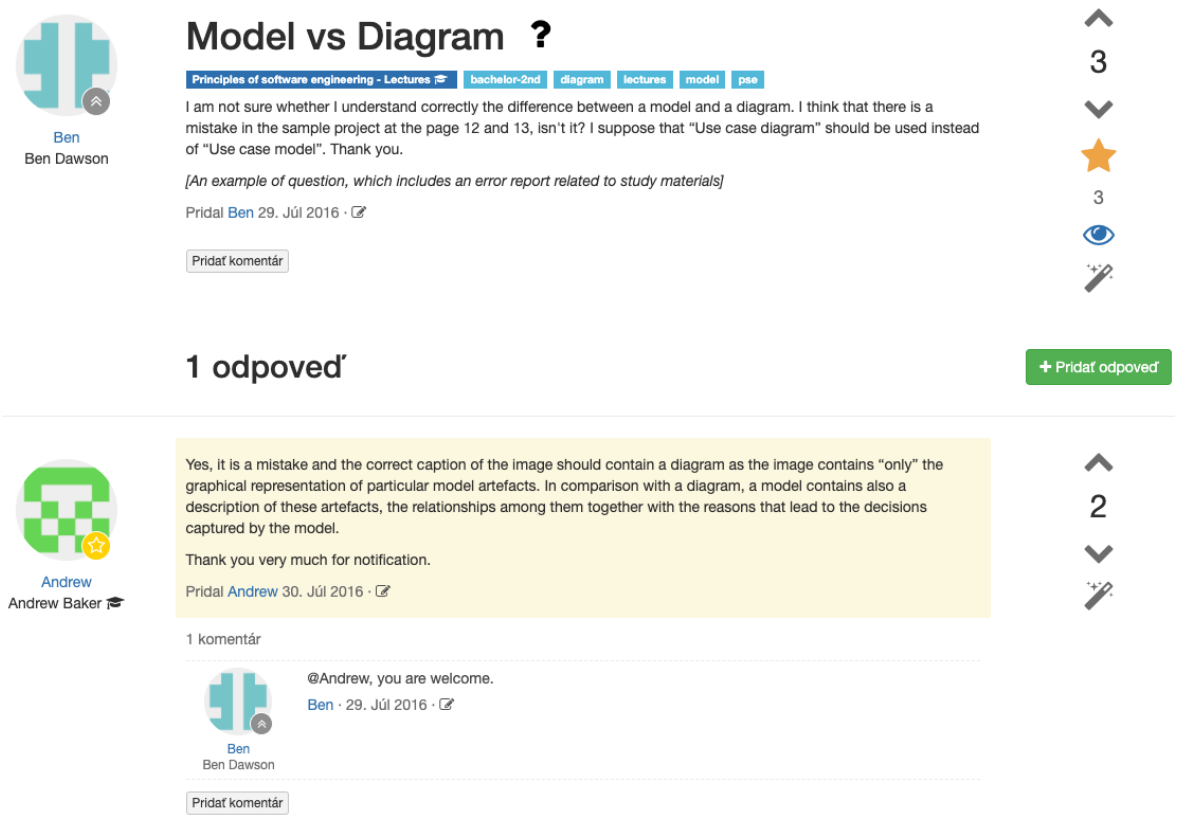
potreby zásahu zamestnancov.

Zatiaľ čo Stack Exchange ponúka množstvo funkcií a má obrovský dosah, študent by sa tam ťažko dopátral odpovediam na otázky, ktoré sa týkajú záležitostí špecifických pre jeho fakultu. Ďalšou nevýhodou je nepodporovanie anonymných otázok a odpovedí.

1.1.6 Askalot

Askalot [8] je portál otázok a odpovedí, ktorý sa zrodil na Fakulte informatiky a informačných technológií Slovenskej technickej univerzity v Bratislave. Vo svojej pôvodnej forme bol aktívne využívaný v rokoch 2014 až 2020. Prístupný bol iba študentom fakulty. Študentom okrem diskusie o jednotlivých kurzoch umožňoval aj vyjadriť sa k iným organizačným veciam a k fungovaniu fakulty.

Portál funguje na podobnom princípe ako Stack Exchange. Člení otázky za pomoci kategórií alebo tagov. Používatelia môžu hlasovať o užitočnosti otázky alebo odpovede.



Obr. 1.3: Ukážka položenej otázky - portál Askalot

Zakladateľ otázky môže jednu z odpovedí označiť ako odpoveď ktorá otázku vyriešila. Používatelia môžu označovať svoje obľúbené otázky. Askalot kategorizuje používateľov na vyučujúcich a študentov. Vyučujúci môže ohodnotiť odpovede a otázky podľa jeho uváženia. Veľkým rozdielom od Stack Exchange je možnosť klásť otázky anonymne. Táto funkcia môže študentov odbremeniť od tlaku, že sa náhodou opýtajú zlú otázku.

Askalot má najväčšiu podobnosť nášmu portálu. Avšak má už svoj vek a niekoľko rokov nie je udržiavaný. Našou snahou bude vytvoriť modernejší a intuitívnejší portál.

1.2 Potrebné technológie

1.2.1 Spring

Spring [9] je aplikačný framework vhodný pre budovanie aplikácií malého aj veľkého rozsahu. Skladá z niekoľkých modulov, ktoré je možné do projektu pripojiť podľa potreby. V našom projekte budeme využívať moduly určené pre tvorbu webových aplikácií, komunikáciu s databázou, zabezpečenie a e-mailovú komunikáciu.

Webový modul ponúka možnosť komunikácie s okolím prostredníctvom HTTP pro-

tokolu. Databázové moduly zjednodušujú komunikáciu medzi aplikáciou a databázou. Zahŕňajú aj objektovo relačné mapovanie a ovládače pre komunikáciu s rôznymi typmi databáz. Zabezpečovací modul ponúka funkcionalitu autentifikácie (overenie používateľa) a autorizácie (overenie právomoci prihláseného používateľa). E-mailový modul zjednodušuje zasielanie e-mailov aplikáciou.

Spring Boot je rozšírenie, ktorý pomáha urýchliť a zjednodušiť vývoj aplikácií v Spring frameworku. Mimo iného nás aj odbremeňuje od zložitej konfigurácie, keďže sa stará o automatické prednastavenie tých najvhodnejších nastavení, ktoré určujú vývojári frameworku. Tieto nastavenia však stále môžeme jednoducho zmeniť v prípade potreby.

Spring využijeme pri programovaní serverovej časti aplikácie. Aplikácia bude poskytovať API, ktoré umožňuje komunikáciu s vonkajším svetom. Jej ďalšou zodpovednosťou je komunikácia s databázou a staranie sa o autentifikáciu používateľov. Framework sme si vybrali kvôli jeho zrelosti, naším dlhoročným skúsenostiam s frameworkom, veľkému množstvu poskytovaných knižníc a množstve voľno prístupnej komunitnej podpory.

1.2.2 Angular

Angular [10] je aplikačný framework slúžiaci pre vytváranie jednoducho webových aplikácií. Využíva programovací jazyk Typescript, ktorý je nadmnožinou programovacieho jazyka Javascript so statickou typovou kontrolou. Je založený na komponentovej architektúre a ponúka množstvo vstavaných knižníc.

Umožňuje nám vytvárať jednostránkové aplikácie, ktoré sa od bežných webových stránok líšia tým, že prechody medzi stránkami si nevyžadujú kompletné obnovenie stránky. Všetok potrebný kód sa načíta pri prvotnej návšteve stránky a stránka sa obnovuje dynamicky pomocou HTTP požiadaviek. Takáto jednostránková webová aplikácia sa pocitovo podobá používaniu natívnej aplikácie, vytvorenej špeciálne pre operačný systém daného zariadenia. Angular aplikácia bude bežať na strane klienta a so serverom bude komunikovať výhradne prostredníctvom HTTP protokolu.

Angular bude zodpovedný za aplikáciu, ktorá bude bežať vo webovom prehliadači na strane používateľa. Pomocou REST API [11] bude komunikovať so serverovou časťou aplikácie. Framework sme si vybrali z dôvodu, že poskytuje možnosť vytvárať moderné jednostránkové webové aplikácie. Ďalšími faktormi výberu boli naše dlhoročné skúsenosti s frameworkom a množstvo knižníc či podpory zo strany komunity.

1.2.3 PostgreSQL

PostgreSQL [12] je objektovo-relačný databázový systém. Ide o robustnú technológiu s otvoreným zdrojovým kódom a rozsiahlou vývojovou komunitou. Je navrhnutý tak, aby bol využiteľný v systémoch s minimálnym počtom užívateľov, ale aj vo veľkých podnikových aplikáciách. Ponúka veľmi dobrú podporu transakcií, ktoré musia dodržiavať atomicitu, konzistentnosť, izoláciu a trvalosť.

Pre tento databázový systém sme sa rozhodli vďaka skúsenostiam, ktoré sme nadobudli na kurzoch databáz, kde sa využíval práve systém PostgreSQL. V čase písania práce ho tiež považujeme za všeobecne najlepší databázový systém s otvoreným zdrojovým kódom. Poskytuje tiež dobrú kompatibilitu s ostatnými technológiami, ktoré budeme využívať.

1.2.4 Hibernate

Hibernate [13] je framework umožňujúci objektovo-relačné mapovanie pre programovací jazyk Java. Funguje ako vrstva medzi aplikáciou a databázovým systémom a snaží sa programátorovi zjednodušiť manipuláciu s databázou. S dátami pracujeme pomocou Java tried, kde jedna trieda je rovná jednej tabuľke. Programátor teda nie je nútený písať databázové dopyty vlastnoručne pre každú operáciu.

Výhodou je aj jednoduchá zámena databázových systémov. Hibernate má na starosti prekladanie našich operácií do databázových dopytov. Tento preklad vie spraviť pre niekoľko kompatibilných databázových systémov. Musíme sa však zaviazať tým, že budeme dodržiavať pravidlá, ktoré nám Hibernate určí. Ak teda potrebujeme napísať vlastný databázový dopyt, musíme použiť Hibernate Query Language. Hibernate nám poskytuje aj možnosť využívania natívnych dopytov nami používaného databázového systému, avšak za cenu bezproblémovej migrácie databázového systému v budúcnosti.

Hibernate nám bude slúžiť pre zjednodušenie práce s dátami v aplikácií. Pre Hibernate sme sa rozhodli z dôvodu, že je to štandard vo svojom odvetví.

Kapitola 2

Návrh portálu

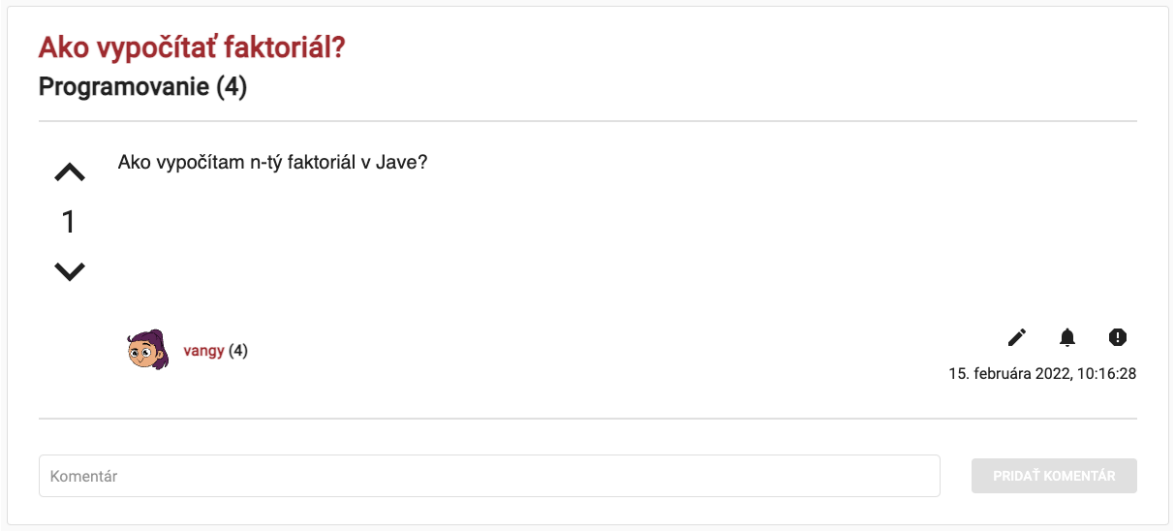
Hlavným cieľom portálu Unibask je zjednodušenie komunikácie v rámci univerzitnej komunity. Každá diskusia začína položením otázky, ktorá je kategorizovaná do jednej z kategórií. Do diskusie sa môžu zapájať všetci používatelia pomocou odpovedí alebo komentárov a zdieľať tak svoje vedomosti.

Vhodnosť otázok a odpovedí je meraná reputáciou. Je ich možné hodnotiť kladným alebo záporným hodnotením. Sumáciou reputácie príspevkov používateľa sa vyráta aj reputácia samotného používateľa. Takto vieme rozlíšiť dôveryhodných používateľov od tých nedôveryhodných. Používateľova reputácia slúži aj ako forma určitej prestíže. Všetci používatelia sú zobrazení v rebríčku, kde sú zoradení podľa ich reputácie.

Portál poskytuje zoznam všetkých otázok a umožňuje ich vyhľadávať na základe ich nadpisu a textu. Zoznam je možné triediť podľa kategórií otázok. Používateľ taktiež môže sledovať kategórie a tým pádom vidieť v zozname sledovaných otázok iba otázky sledovaných kategórií.

Používateľ vo svojom profile vidí všetky svoje príspevky. Dokáže si zmeniť heslo, študijný program alebo profilovú fotku. Môže si zapnúť funkcionalitu doručovania upozornení prostredníctvom emailovej adresy.

Do portálu je možné zaregistrovať sa iba s platnou emailovou adresou Univerzity Komenského. V prípade zabudnutia hesla môže používateľ požiadať o jeho zresetovanie, ktoré musí potvrdiť prostredníctvom emailu priradeného k danému účtu.



Obr. 2.1: Položená otázka

2.1 Diskusia

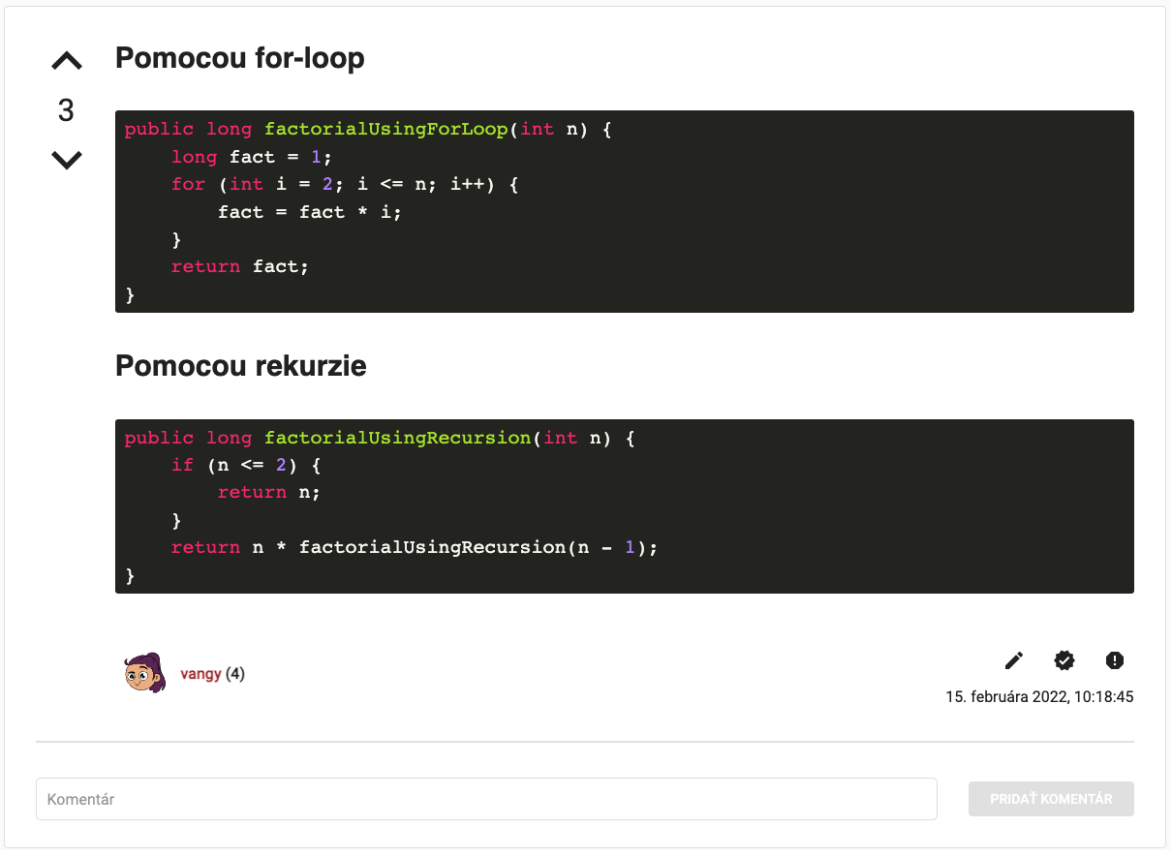
2.1.1 Otázky

Základným prvkom každej diskusie je otázka. Otázky môžu byť pokladané všetkými používateľmi portálu. Je možné ich položiť vo svojom vlastnom mene, alebo anonymne, čím používateľ môže predísť strachu z opýtania sa. Povinnými prvkami pri založení otázky je vyplnenie nadpisu, tela otázky a zvolenie kategórie do ktorej otázka zapadá. Telo otázky je možné formátovať za pomoci špeciálneho editoru. Po vytvorení otázky bude otázke priradený unikátny identifikátor a stáva sa viditeľnou v zozname otázok. Používatelia ku nej môžu následne pridávať odpovede, komentovať ju, alebo ju hodnotiť na základe jej užitočnosti. Ak je otázka nevhodná, je možné ju nahlásiť. V prípade, že si používateľ želá byť upozornený ohľadom novej aktivity vo vlákne nejakej otázky, môže zvoliť možnosť zapnutia upozornení pre konkrétnu otázku.

Zakladateľ má po založení otázky právomoc na jej úpravu. V prípade, že bola jeho otázka zodpovedaná, môže označiť správnu odpoveď a pomôcť tak budúcim čitateľom, ktorí riešia rovnaký problém.

2.1.2 Odpovede

Používatelia môžu na otázky odpovedať. Tak ako pri otázkach, aj tu je možné urobiť to anonymne. Pri odpovedaní je povinným prvkom iba vyplnenie tela odpovede. Telo odpovede je možné formátovať za pomoci špeciálneho editoru. Odpovede je možné hodnotiť na základe ich užitočnosti, prípadne ich nahlásiť ako nevhodné. Zakladateľ odpovede môže svoju odpoveď upravovať.



Obr. 2.2: Odpoveď na otázku

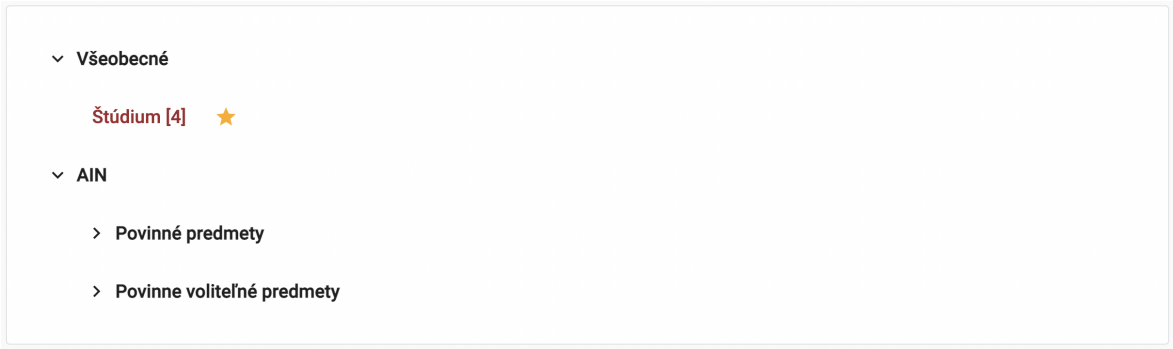
2.1.3 Komentáre

Otázky aj odpovede je možné komentovať. Komentáre slúžia pre získanie viac informácií alebo zanechanie spätnej väzby autorovi ohľadom otázky/odpovede. Komentáre na rozdiel od odpovedí umožňujú hlbšiu diskusiu medzi používateľmi. Komentáre je možné hodnotiť iba kladne – záporné hodnotenie nie je povolené.

2.2 Kategórie

Každá otázka musí byť zaradená do nejakej kategórie. Kategórie slúžia na zoskupovanie otázok, ktoré spolu nejakým spôsobom súvisia. V portáli Unibask sú vhodnými kandidátmi kategórií jednotlivé predmety, ktoré sú vyučované na fakulte. Je potrebné, aby kategórie boli flexibilné a dali sa jednoducho rozširovať.

Kategórie využívajú dátovú štruktúru strom. Koreňom tohoto stromu je neviditeľná kategória, ktorá obsahuje ľubovoľné množstvo podkategórií. Listom stromu je kategória, ktorá neobsahuje žiadne podkategórie. Máme dva typy kategórií – tie ktoré nie sú listami stromu a tie ktoré sú.



Obr. 2.3: Zoznam kategórií

Kategórie, ktoré nie sú listami stromu slúžia na zoskupovanie ostatných kategórií, no nie je možné k ním priradzovať otázky. Priradzovanie otázok týmto kategóriám sme zakázali z dôvodu zvýšenia organizácie a donútenia používateľov voliť čo najšpecifickejšiu kategóriu, aby otázky neboli rozhádzané po viacerých miestach.

Kategórie, ktoré sú listami stromu slúžia na zoskupovanie otázok. Ich zoznam sa používateľovi zobrazuje pri pokladaní novej otázky. Používatelia môžu takéto kategórie označovať ako obľúbené. Po označení kategórie za obľúbenú bude aplikácia vedieť, že otázky položené v danej kategórií majú byť pre používateľa priorizované.

2.3 Zoznam otázok

Otázky musia byť používateľom jednoducho prístupné pomocou zoznamu otázok. Zoznam obsahuje otázky zoradené podľa dátumu poslednej aktivity otázok. Každá otázka má svoj náhľad. Náhľad obsahuje nasledujúce informácie:

- Názov otázky
- Názov kategórie otázky
- Hodnotenie otázky
- Počet odpovedí na otázku
- Počet zobrazení otázky
- Dátum poslednej aktivity v otázke

Po kliknutí na náhľad bude používateľ presmerovaný na konkrétnu otázku so všetkými jej informáciami. Zoznam na začiatok načíta 10 otázok. Ďalšie otázky načítava v prípade, že sa používateľ blíži ku spodku stránky.

Obr. 2.4: Zoznam otázok

Máme 3 typy zoznamov. Prvý typ zoznamu obsahuje úplne všetky otázky a je úvodnou stránkou portálu. Druhý typ zoznamu obsahuje otázky konkrétnej kategórie a nachádzajú sa v ňom všetky otázky používateľom zvolenej kategórie. Tretí typ zoznamu obsahuje otázky všetkých kategórií, ktoré používateľ označil za obľúbené.

V prípade, že používateľ hľadá nejakú konkrétnu otázku, môže využiť vyhľadávanie v zozname otázok. Vyhľadávanie filtruje otázky, ktoré obsahujú v nadpise alebo v tele otázky vyhľadávanú frázu.

2.4 Hodnotenie

Otázky, odpovede a komentáre je možné hodnotiť. Každý používateľ môže hodnotiť otázky a odpovede buď kladne, alebo záporne. Komentáre je možné hodnotiť iba kladne. Za jeden príspevok vie jeden používateľ zahlasovať vždy iba raz. Hodnotenie slúži na určenie kvality príspevku a v širšom zmysle aj na určenie dôveryhodnosti používateľa. Hodnotenie kvality príspevku sa vypočíta súčtom všetkých hodnotení príspevku. Na základe tohoto hodnotenia budú aj vo vlákne otázky uprednostňované odpovede s najlepším hodnotením. Hodnotenie a teda reputácia používateľa sa vypočíta pomocou súčtu hodnotení všetkých jeho príspevkov. Celková reputácia používateľa je zobrazená pri každom príspevku, ktorý uverejní a tým pádom majú ostatní používatelia väčší prehľad o jeho dôveryhodnosti.

Obr. 2.5: Rebríček používateľov

Rebríček používateľov slúži ako zoznam všetkých používateľov portálu zoradených podľa ich výslednej reputácie. Ponúka určitý druh prestíže pre používateľov. Mal by ich motivovať ku prispievaniu hodnotných príspevkov. Prví traja používatelia v rebríčku by mali byť od ostatných nejako odlíšení. V rebríčku sa kliknutím na používateľa zobrazí jeho kompletný profil.

2.5 Používateľský profil

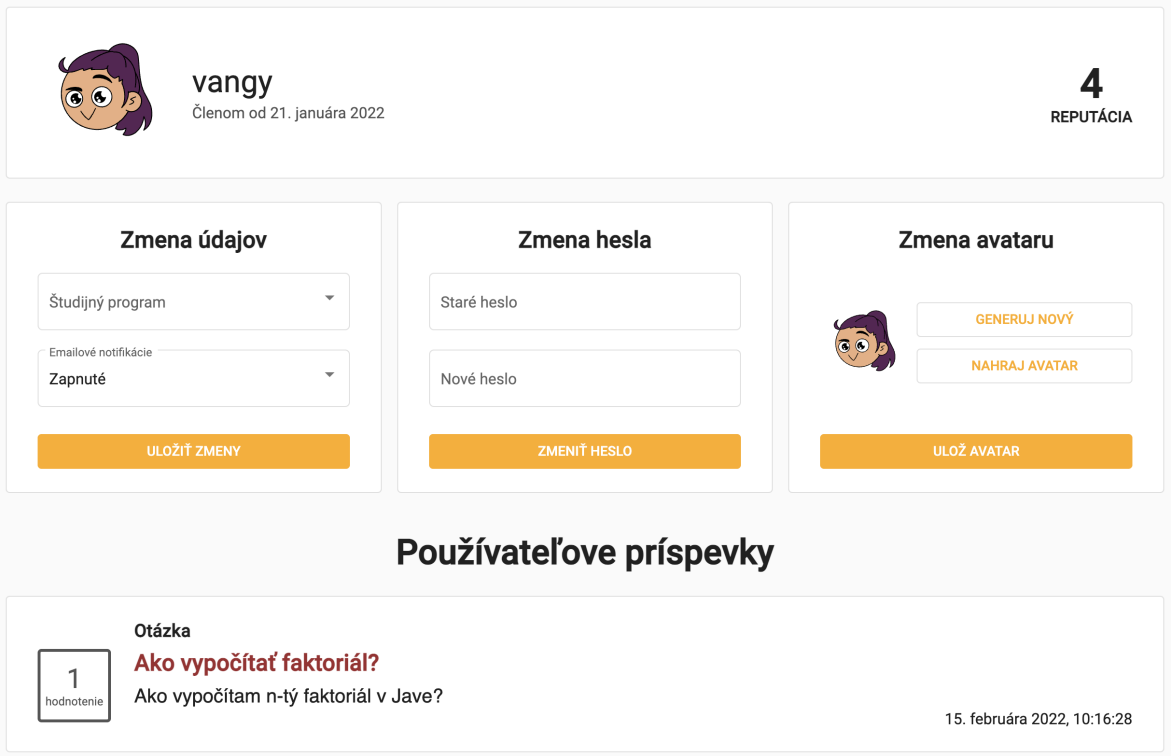
Používateľský profil obsahuje informácie o používateľovi. Konkrétne jeho meno, študijný plán, reputáciu, profilovú fotku, dátum registrácie. Ďalej obsahuje zoznam všetkých jeho otázok, odpovedí a komentárov, ktoré kedy vytvoril.

Používateľ si vo svojom profile dokáže meniť údaje. Vie si zmeniť svoj študijný program, svoje heslo a nastaviť politiku emailových upozornení. Môže si nechať vygenerovať náhodnú profilovú fotku, alebo nahrať svoju vlastnú.

2.6 Upozornenia

Upozornenia slúžia na informovanie používateľa ohľadom novej aktivity. Aplikácia obsahuje dva typy upozornení. Upozornenia na novú aktivitu v otázke a upozornenia na vznik novej otázky v kategórií.

Upozornenia na novú aktivitu v otázke si používateľ môže zapnúť sledovaním konkrét-



Obr. 2.6: Používateľský profil

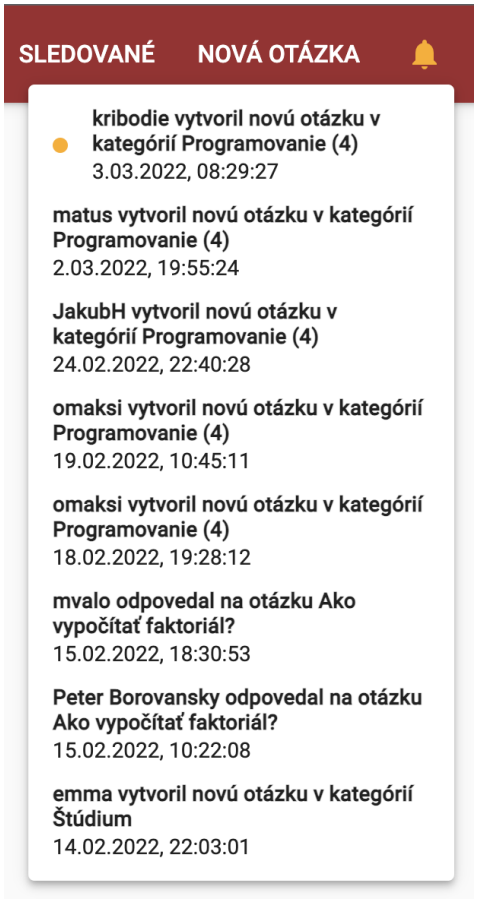
nej otázky. Pre autora otázky budú tieto upozornenia automaticky zapnuté. Upozornenia na vznik novej otázky v kategórii si používateľ môže zapnúť sledovaním kategórie.

Zoznam upozornení sa nachádza v navigačnej lište pod tlačidlom s ikonou zvončeku. Ikona je plná ak existujú nové upozornenia a prázdna ak od posledného prezretia zoznamu nepribudlo žiadne nové upozornenie. Otvorením tohoto zoznamu sa vždy načítajú najnovšie upozornenia. Kliknutím na upozornenie sa otvorí otázka prislúchajúca danému upozornenie.

Používateľ si vo svojom profile môže zapnúť funkciu zasielania upozornení na emailovú adresu. Každé upozornenie mu potom bude zaslané aj s jeho obsahom a odkazom na otázku prostredníctvom emailu.

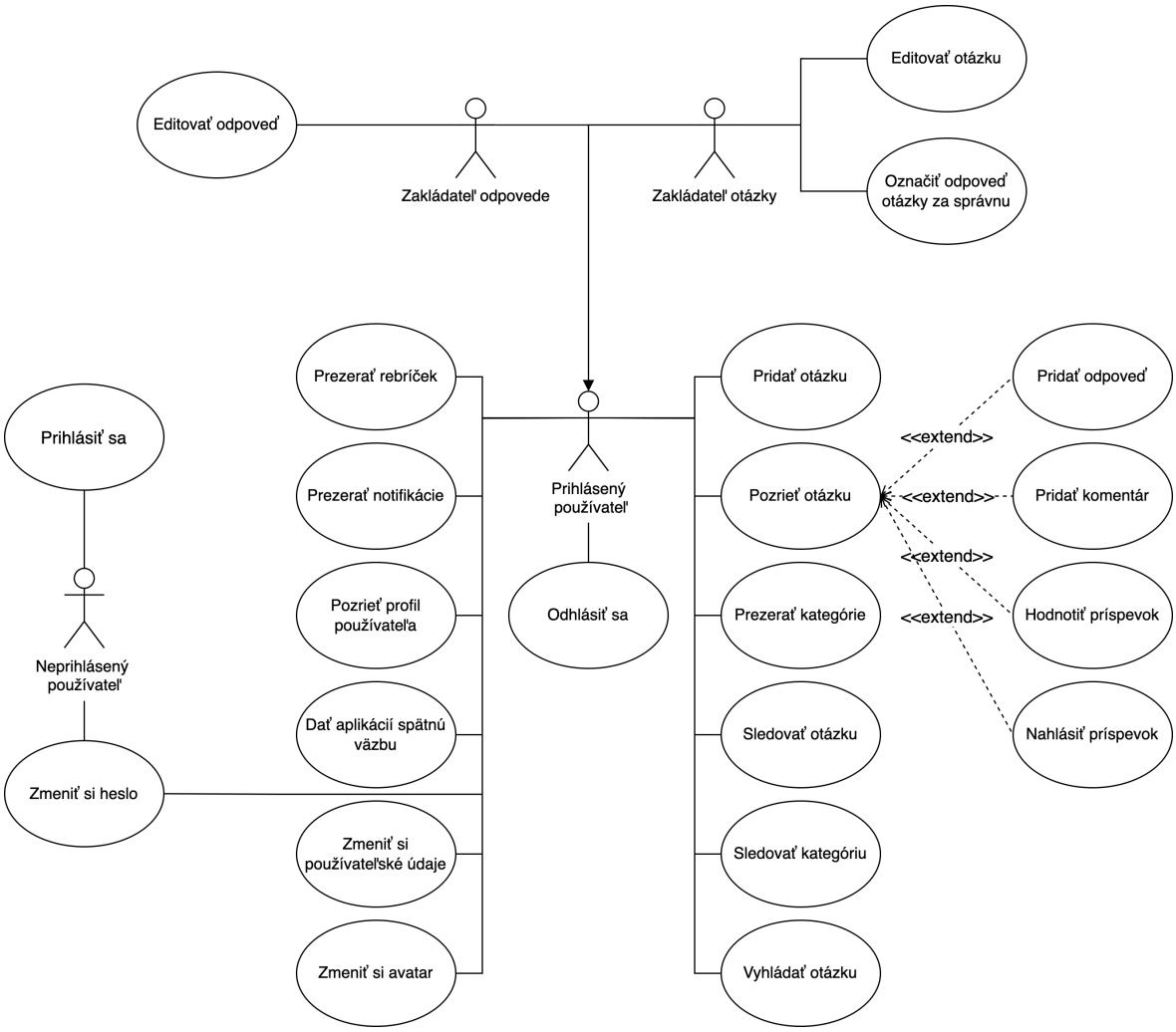
2.7 Logo

Pri návrhu loga sme sa inšpirovali logom Univerzity Komenského – konkrétne dvojitým okrajom a minimalistickým čiernobielym prevedením [14]. Logo vyjadruje hlavnú funkcionalitu portálu. Okrúhla bublinka reprezentuje otázku a obdĺžniková bublinka reprezentuje odpoveď.



Obr. 2.7: Panel upozornení

Obr. 2.8: Logo portálu



Obr. 2.9: Diagram prípadov použitia

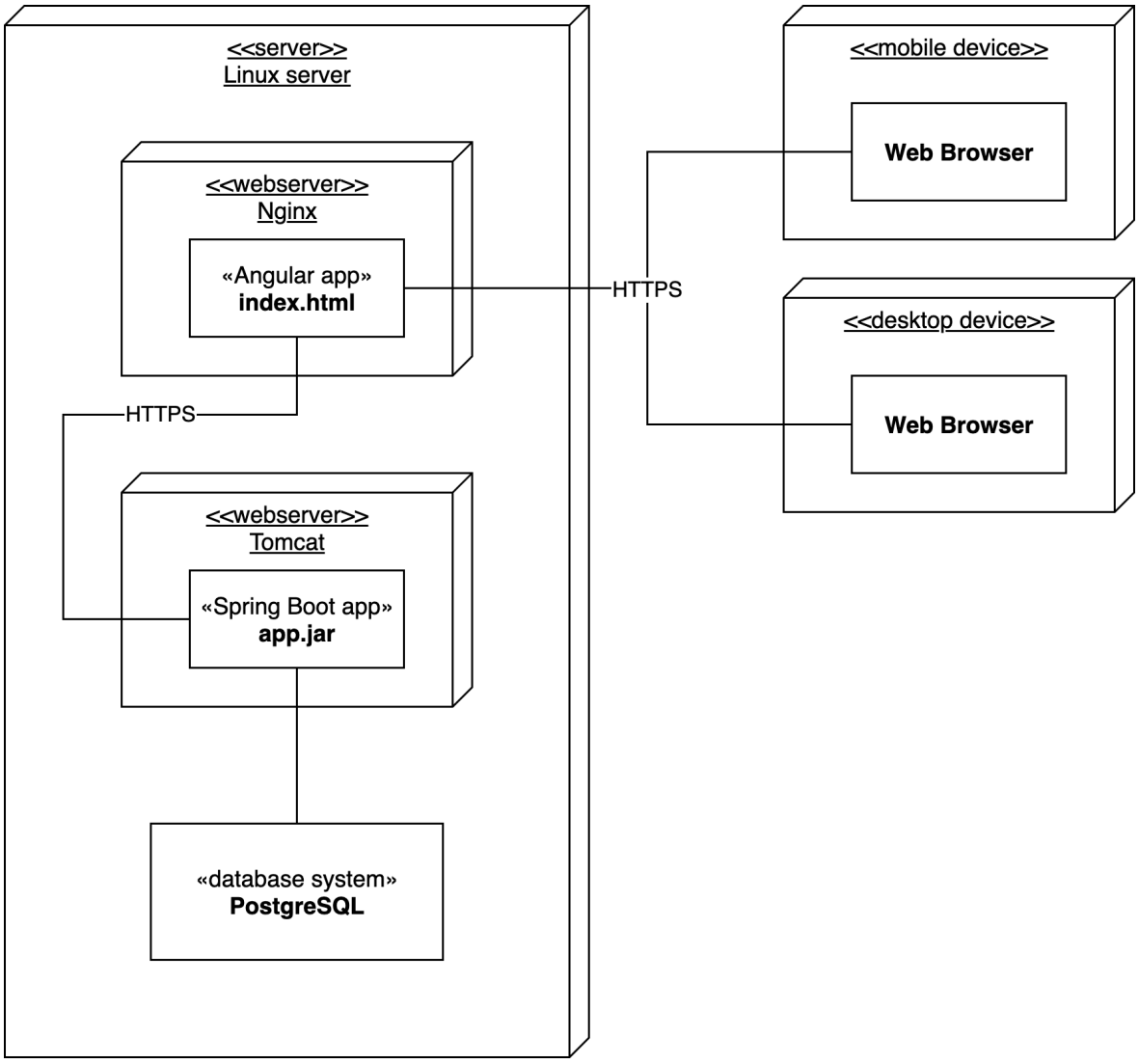
2.8 Diagramy

2.8.1 Diagram prípadov použitia

Diagram prípadov použitia slúži na grafickú reprezentáciu možných interakcií používateľa s portálom. V diagrame sme popísali interakcie 4 rôznych používateľov našej aplikácie. Neprihlásený používateľ sa môže prihlásiť, alebo si zmeniť heslo v prípade zabudnutia. Prihlásený používateľ má veľa možností interakcie. Zvyšné 2 typy používateľov dedia všetku funkcionalitu prihláseného používateľa, no pribúdajú im ďalšie možnosti interakcie so systémom.

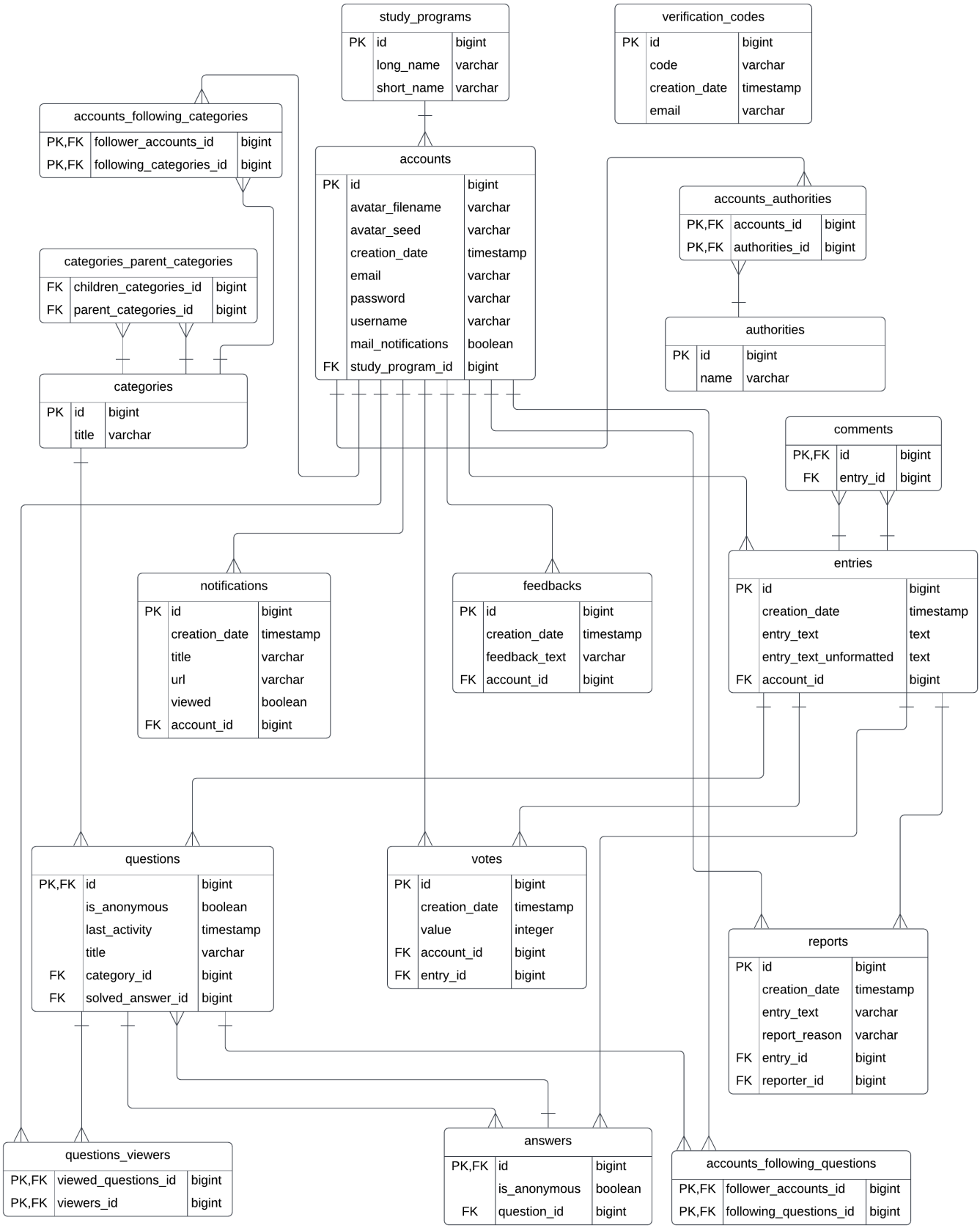
2.8.2 Diagram nasadenia

Diagram ponúka vizualizáciu toho ako má byť systém nasadený. Základnými časťami sú uzly, ktoré sú reprezentované kvádrami a artefakty, ktoré sú reprezentované obdĺžnikmi. Asociácie medzi prvkami sú reprezentované spojovacími čiarami.



Obr. 2.10: Diagram nasadenia

Aplikácia bude bežať na fyzickom Ubuntu [15] serveri. Na serveri bude bežať databázový systém PostgreSQL, webový server Tomcat [16] a webový server Nginx [17]. Používateľ môže s klientskou časťou aplikácie komunikovať prostredníctvom webového prehliadača na mobilnom zariadení alebo počítači. Táto klientská časť je poháňaná webovým serverom Nginx. Webový server Nginx poskytuje súbory statickej stránky, ktorá je vybudovaná pomocou technológie Angular. Požiadavky, ktoré sú určené serverovej časti aplikácie zasiela pomocou reverzného proxy na webový server Tomcat, kde beží serverová časť aplikácie. Serverová časť aplikácie ako jediná komunikuje s databázovým systémom.



Obr. 2.11: Model dátovej vrstvy

2.8.3 Model dátovej vrstvy

Diagram znázorňuje štruktúru databázy, ktorá je využívaná aplikáciou. Jeden obdĺžnik znázorňuje jednu tabuľku databázy aj s dátami, ktoré sú v nej udržiavané. Čiary medzi nimi znázorňujú vzťahy medzi tabuľkami. Zakončenie čiary kolmicou znázorňuje vzťah 'one'. Rozvetvenie čiary na tri znázorňuje vzťah 'many'. Rozhodli sme sa do diagramu pridať aj informácie o dátových typoch tabuliek a primárnych a cudzích kľúčoch.

Tabuľka **accounts** udržiava účty používateľov, vrátane prihlasovacích údajov a informácií o používateľoch. Na túto tabuľku sa odkazuje mnoho ostatných tabuliek. Priradzujú tak rôzne dáta ku konkrétnemu účtu. Účet môže mať priradený študijný program, preto sa tabuľka odkazuje na tabuľku **study_programs**.

Tabuľka **categories** slúži na udržiavanie kategórií otázok. Chceme, aby bolo kategórie možné jednoducho modifikovať, preto ich držíme v databáze.

Tabuľka **categories_parent_categories** zabezpečuje to, aby kategórie mohli byť štruktúrované ako strom. Umožňuje kategóriám mať svoje podkategórie.

Tabuľka **entries** slúži na udržiavanie všeobecného základu všetkých príspevkov, ktorými sú otázky, odpovede a komentáre. Môžeme povedať, že od tejto tabuľky dedia. A teda tabuľky **questions**, **answers**, **comments** sa na túto tabuľku odkazujú cudzím kľúčom. Účelom tejto tabuľky je zovšeobecnenie príspevku. Vďaka tomu sa môžeme napríklad tabuľkou **votes** odkazovať na tabuľku **entries** miesto odkazovania sa na tri rôzne tabuľky, v ktorých držíme otázky, odpovede a komentáre.

Tabuľka **questions** slúži na udržiavanie otázok. Otázka obsahuje odpovede a jedna z nich môže byť vybratá za správnu odpoveď. Preto sa tabuľka odkazuje na tabuľku **answers**. Otázka musí prislúchať ku kategórií, preto sa odkazuje aj na tabuľku **categories**.

Tabuľka **answers** slúži na udržiavanie odpovedí. Odpoveď musí byť priradená nejakej otázke a preto sa tabuľka odkazuje na tabuľku **questions**.

Tabuľka **comments** slúži na udržiavanie komentárov. Komentár musí byť priradený buď k nejakej otázke alebo odpovedi. Mohli by sme sa odkazovať na dve tabuľky **questions** a **answers**, no lepšiu alternatívou je pre nás odkazovať sa na tabuľku **entries**.

Tabuľka **votes** slúži na udržiavanie hlasov. Hlasovať je možné za ľubovoľný príspe-

vok, preto sa odkazuje na tabuľku **entries**. Každý hlas musí byť priradený nejakému používateľovi, hlavne z dôvodu zamedzenia dvojitého hlasovania – preto sa odkazuje na tabuľku **accounts**.

Tabuľka **notifications** slúži na udržiavanie upozornení. Každý účet má množinu svojich upozornení a preto sa tabuľka odkazuje na tabuľku **accounts**.

Tabuľka **study_programs** slúži na udržiavanie študijných programov.

Tabuľka **feedbacks** slúži na udržiavanie spätnej väzby používateľov. Tabuľka sa odkazuje na používateľa pomocou tabuľky **accounts**, aby sme ho vedeli v prípade potreby kontaktovať.

Tabuľka **authorities** slúži na udržiavanie autorít. Autorita slúži pre určenie právomoci používateľa.

Tabuľka **reports** slúži na udržiavanie nahlasovaní príspevkov. Odkazuje sa na tabuľku **accounts** kvôli udržaniu identity nahlásovateľa a na tabuľku **entries**, pretože nahlásenie musí byť priradené ku nejakému príspevku.

Tabuľka **verification_codes** slúži na udržiavanie kódov, ktoré využívame pre potvrdenie identity používateľského emailu – konkrétne v prípade registrácie a požiadavky na zmenu zabudnutého hesla.

Tabuľka **accounts_following_categories** slúži na udržiavanie sledovaní kategórií používateľmi.

Tabuľka **accounts_following_questions** slúži na udržiavanie sledovaní otázok používateľmi.

Tabuľka **accounts_authorities** slúži na prepojenie autorít s používateľskými účtami. Jeden používateľ môže mať viac autorít.

Tabuľka **questions_viewers** slúži na udržiavanie sledovanosti otázok ľuďmi. Na základe tejto tabuľky vieme vyrátať koľko unikátnych ľudí si zobrazilo nejakú otázku. Jedno pozorovanie je priradené jednej otázke pomocou odkazu na tabuľku **questions** a jednému používateľovi pomocou odkazu na tabuľku **accounts**.

Kapitola 3

Implementácia portálu

3.1 Aplikačné vrstvy

Portál vieme rozdeliť na tri vrstvy. Webovú aplikáciu, serverovú aplikáciu a databázu. Webová aplikácia je vytvorená pomocou technológie Angular. Serverová aplikácia je vytvorená pomocou technológie Spring Boot. Databáza je typu PostgreSQL. Webová a serverová aplikácia spolu komunikujú pomocou REST API [11] – teda GET/POST/PUT/DELETE HTTP požiadaviek [18]. Databáza komunikuje iba so serverovou aplikáciou. Webová aplikácia na strane klienta komunikuje tiež iba so serverovou aplikáciou – nemá priamy prístup ku databáze.

3.2 Zhrnutie frameworku Spring Boot

Cieľom tejto sekcie je oboznámiť čitateľa so základnými štruktúrami Spring frameworku pre lepšie pochopenie implementácie portálu. Predpokladáme, že čitateľ je familiárny s programovacím jazykom Java alebo s iným objektovo-orientovaným programovacím jazykom.

3.2.1 Anotácie

Anotácie [19] sú súčasťou programovacieho jazyka Java a framework Spring ich často využíva. Anotáciam je možné priradiť aj parametre. V aplikácií najčastejšie využívame nasledujúce anotácie:

@Service

Anotácia slúži na označenie komponentu, ktorý v sebe obsahuje výhradne funkcionalitu vrstvy biznis logiky.

@RestController

Anotácia slúži na označenie komponentu, ktorý ponúka REST funkcionalitu. Metódy triedy s touto anotáciou vracajú objekty, ktoré sú automaticky konvertované do JSON formátu [20].

@RequestMapping

Anotáciu využijeme v prípade, že chceme namapovať nejakú HTTP požiadavku k metóde. Medzi jej vstupné parametre patrí cesta a HTTP metóda. Parameter cesty určí cestu, na ktorú má byť požiadavka zaslaná, aby bola spracovaná anotovanou metódou. Parameter HTTP metódy určí konkrétnu HTTP metódu na ktorú chceme reagovať. Konkrétne sa jedná o GET, PUT, POST, DELETE. Spring ponúka aj alternatívu k použitiu tohoto parametra. Môžeme použiť anotácie @GetMapping, @PutMapping, @PostMapping, @DeleteMapping. Týmto anotáciám bude stačiť zadať už iba parameter cesty.

@Autowired

Táto anotácia zjednodušuje vkladanie objektov do nejakej konkrétnej triedy. Objekty nemusíme vytvárať pomocou operátoru 'new'. Spring je inteligentný framework a o spravovanie objektov sa stará sám v pozadí. Ak mu chceme povedať, aby spravoval objekty konkrétnej triedy, musíme danú triedu označiť anotáciou @Service, @RestController alebo inou obdobnou anotáciou. Prípadne môžeme využiť anotáciu @Bean.

@Bean

Ak chceme byť špecifickejší v tom, ako sa ma anotácia @Autowired o nejakú triedu starať, môžeme vytvoriť metódu, ktorá vracia objekt danej triedy a označiť túto metódu anotáciou @Bean. V tele metódy máme slobodu robiť čo chceme, no na konci musíme vrátiť objekt. Pri použití anotácie @Autowired bude vložený objekt v takom stave, v akom bol objekt vrátený v metóde.

@Entity

Anotácia špecifikuje, že daná trieda má byť mapovaná k databázovej tabuľke a že s ňou budeme pracovať ako s databázovou entitou. Konkrétne sa jedná o funkcionalitu objektovo-relačného mapovania.

@Transactional

Anotácia definuje rozsah a kontext jednej databázovej transakcie. Predvolené správanie je také, že všetok databázový kód vykonaný v metóde označenej touto anotáciou sa

vykoná v jednej transakcií.

3.3 Zhrnutie frameworku Angular

Cieľom tejto sekcie je oboznámiť čitateľa so základnými štruktúrami Angular frameworku pre lepšie pochopenie implementácie portálu. Predpokladáme, že čitateľ je familiárny s programovacím jazykom Typescript alebo s iným obdobným programovacím jazykom.

3.3.1 Komponent

Angular komponent [21] predstavuje prvok používateľského rozhrania. Skladá sa z:

- Typescript triedy, ktorá určuje jeho správanie
- HTML šablóny, ktorá určuje ako sa komponent vykreslí
- CSS štýlov, ktoré určujú vizuálne formátovanie komponentu
- Selektoru, ktorý určuje spôsob referencie na komponent

Komponenty je vhodné používať na zapúzdrenie nejakej funkcionality a správania, ktoré sa opakuje. Chceme minimalizovať opakujúci sa kód, aby sme nemuseli rovnakú funkcionalitu meniť na viacerých miestach a zvýšili tak kvalitu nášho kódu. Komponent môže akceptovať vstupné dáta a poskytnúť výstupné dáta. Má svoj životný cyklus a vieme teda zavolať kód pri jeho počiatočnej inicializácií alebo pri jeho konečnom zničení. Komponent by nemal obsahovať biznis logiku. Mal by slúžiť iba ako rozhranie medzi používateľom a aplikáciou. Biznis logiku (napríklad zasielanie HTTP požiadaviek) by mal delegovať servisu.

3.3.2 Servis

Angular servis [22] slúži na zapúzdrenie biznis logiky, ktorú vieme využívať vo viacerých častiach aplikácie. Ide napríklad aj o zasielanie HTTP požiadaviek, validáciu používateľského vstupu, alebo vypisovanie údajov do konzoly. Každý servis by sa mal sústrediť na nejakú konkrétnu doménu a zoskupovať funkcionalitu iba danej domény. To znamená, že funkcionalitu týkajúcu sa autentifikácie a vytvárania otázok nechceme mať v jednom rovnakom servise.

Servis vytvoríme pridaním @Injectable ku triede. Môžeme ho vložiť do ostatných komponentov alebo servisov. Rozsah servisu môže byť na úrovni celej aplikácie, modulu,

alebo komponentu. Ak je na úrovni celej aplikácie, tak celá aplikácia zdieľa jednu inštanciu daného servisu. Ak je na úrovni modulu, tak existuje jedna inštancia servisu pre každý modul. Ak je na úrovni komponentu, tak každý komponent má vlastnú inštanciu servisu.

3.3.3 Modul

Angular umožňuje budovať modulárne aplikácie pomocou modulov [23]. Modul zoskupuje funkcionalitu nejakej konkrétnej domény. Môže obsahovať komponenty alebo servisy, ktorých rozsah je na úrovni daného modulu. Moduly vedia exportovať vybranú funkcionalitu, ktorá môže byť importovaná ostatnými modulmi. Každá aplikácia musí mať minimálne jeden modul. Pomocou modulárneho dizajnu vieme zabezpečiť, aby sa niektoré časti aplikácie používateľovi načítali až vtedy, keď ich bude potrebovať. Profilový modul sa napríklad načíta až vtedy, keď používateľ navštívi cestu '/profile'. Ušetríme tak počet dát, ktoré sa musia prenášať po sieti a zrýchlime celkové načítanie stránky.

3.4 Autentifikácia

Musíme zabezpečiť aby nikto nemohol spochybniť identitu používateľa aplikácie a aby každý používateľ bol prihlásený pod profilom ku ktorému má oprávnenie. Používateľ preberá zodpovednosť za svoj profil a za jeho zabezpečenie. Do profilu sa vie prihlásiť pomocou používateľského mena a hesla, ktoré si nastaví pri vytvorení profilu. Používateľovi je umožnené si dané heslo zmeniť – aj v prípade zabudnutia pomocou jeho emailovej adresy.

Autentifikovaní používatelia sú udržiavaní reláciami a identifikačné údaje týchto relácií sú udržiavané na strane používateľa pomocou HTTP Cookies [24]. Implementáciu autentifikácie riešime pomocou prostriedkov, ktoré ponúka modul zabezpečenia frameworku Spring. Ponúkanú funkcionalitu musíme prispôsobiť naším potrebám, no naším cieľom je zostať kompatibilný s ponúkaným systémom autentifikácie.

Zadefinujeme triedu *UnibaskUserDetailsService*, ktorá implementuje rozhranie *UserDetailsService*. Toto rozhranie je poskytnuté Spring frameworkom a slúži na špecifikáciu toho, ako má byť používateľ načítaný na základe jeho mena. Musíme implementovať metódu *loadUserByUsername*.

```
@Service
public class UnibaskUserDetailsService implements UserDetailsService {
    private final AccountRepository accountRepository;
```

```
    @Autowired
    public UnibaskUserDetailsService(AccountRepository accountRepository) {
        this.accountRepository = accountRepository;
    }

    @Override
    @Transactional
    public UserDetails loadUserByUsername(String email)  {
        Account account = accountRepository.findByEmail(email).get();
        return new UnibaskUserDetails(
                account.getEmail(),
                account.getPassword(),
                account.getAuthorities().stream().map(authority ->
                                new SimpleGrantedAuthority(authority.getName()))
                .collect(Collectors.toList()));
    }
}
```

Kód 3.1: Implementácia rozhrania UserDetailsService

V kóde vidíme, že používateľa načítavame z databázy. Metóda *loadUserByUsername* vracia *UserDetails*, čo je tiež rozhranie poskytnuté Spring frameworkom a slúži na špecifikáciu dát, ktoré reprezentujú prihláseného používateľa. Toto rozhranie je potrebné implementovať a preto vytvoríme ďalšiu triedu *UnibaskUserDetails*.

```
public class UnibaskUserDetails implements UserDetails {
    private final String email;
    private final String password;
    private final List<GrantedAuthority> authorities;

    public UnibaskUserDetails(String email, String password,
                              List<GrantedAuthority> authorities) {
        ... priradenie parametrov k premenným triedy
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities()
        {return authorities;}
    @Override
    public String getPassword() {return password;}
    @Override
    public String getUsername() {return email;}
    @Override
    public boolean isAccountNonExpired() {return true;}
    @Override
```

```
    public boolean isAccountNonLocked() {return true;}
    @Override
    public boolean isCredentialsNonExpired() {return true;}
    @Override
    public boolean isEnabled() {return true;}
}
```

Kód 3.2: Implementácia rozhrania UserDetails

Po zadefinovaní triedy *UnibaskUserDetailsService* ju musíme pridať do bezpečnostnej konfigurácie Spring frameworku. Zadefinovať tiež musíme enkodér hesiel. Zvolíme *BCryptPasswordEncoder*, ktorý je súčasťou Spring frameworku.

```
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {
    private final UserDetailsService userDetailsService;

    @Autowired
    public SecurityConfiguration(UserDetailsService userDetailsService) {
        this.userDetailsService = userDetailsService;
    }

    @Bean
    public PasswordEncoder passwordEncoder() {return new BCryptPasswordEncoder();}

    @Bean
    public AuthenticationProvider daoAuthenticationProvider() {
        var provider = new DaoAuthenticationProvider();
        provider.setPasswordEncoder(passwordEncoder());
        provider.setUserDetailsService(userDetailsService);
        return provider;
    }
    ...
}
```

Kód 3.3: Nastavenie bezpečnostnej konfigurácie

3.5 Komunikácia klient-server

Klient (Angular aplikácaia) so serverom (Spring aplikácia) komunikuje prostredníctvom HTTP požiadaviek v súlade s architektonickým štýlom REST. Klient zasiela HTTP požiadavky, server ich spracováva a odpovedá na nich. Komunikáciu ukážeme na časti našej aplikácie, ktorá je zodpovedná za informácie o používateľských profiloch.

3.5.1 Zasielanie HTTP požiadaviek

Metóda *getUser* vytvorí GET požiadavku na cestu '/profile'. V GET požiadavke pribaľuje parameter identifikátora používateľa.

Metóda *getUserInfo* vytvorí GET požiadavku na cestu '/profile/info'.

Metóda *saveUserInfo* vytvorí POST požiadavku na cestu '/profile/info'. V tele požiadavky pribaľuje informácie o identifikátore študijného program a o zapnutí emailových notifikácií.

Všetky metódy vracajú objekt požiadavky. Aby sa požiadavka začala vykonávať je potrebné zavolať na objekt metódu *subscribe*. Požiadavky obsahujú hlavičku a pridaním možnosti 'withCredentials: true' dosiahneme, aby v nich bolo zaslané aj autentifikačné cookie.

```
@Injectable()
export class ProfileService {
  constructor(private http: HttpClient) {}

  getUser(userId: string) {
    return this.http.get<User>(environment.apiUrl + '/profile', {
      headers: new HttpHeaders({'Content-Type': 'application/json'}),
      withCredentials: true,
      params: new HttpParams().append('userId', userId)
    })
  }

  getUserInfo() {
    return this.http.get<UserInfo>(environment.apiUrl + '/profile/info', {
      headers: new HttpHeaders({'Content-Type': 'application/json'}),
      withCredentials: true
    })
  }

  saveUserInfo(userInfo: UserInfo) {
    return this.http.post(environment.apiUrl + '/profile/info', {
      'studyProgramId': userInfo.studyProgramId,
      'mailNotificationsEnabled': userInfo.mailNotificationsEnabled,
    }, {
      headers: new HttpHeaders({'Content-Type': 'application/json'}),
      withCredentials: true
    })
  }
}
```

Kód 3.4: Príklad zasielania HTTP požiadaviek klientskou aplikáciou

3.5.2 Spracovanie HTTP požiadaviek

Uvedieme si príklad spracovania HTTP GET a POST požiadaviek. Triedu v ktorej sa funkcionalita nachádza označíme anotáciou @RestController a anotáciou @RequestMapping, ktorej priradíme cestu ktorú má spracovávať. Táto trieda konkrétne spracováva cestu '/profile'. Aplikačnú logiku chceme mať od tejto triedy oddelenú a preto delegujeme biznis logiku triede *ProfileService*, ktorú do triedy *ProfileController* vkladáme pomocou @Autowired anotácie.

Metóda *getUser* je typu GET a vracia informácie o nejakom používateľovi. Vyžaduje, aby požiadavka obsahovala aj informácie o identifikátore používateľa, ktorého informácie chceme získať.

Metódy *getUserInfo* a *setUserInfo* pridávajú ku ceste triedy dodatok '/info' a teda spracovávajú požiadavky zaslané na cestu '/profile/info'.

Metóda *getUserInfo* je typu GET a vracia informácie o prihlásenom používateľovi. Metóda *setUserInfo* je typu POST a vyžaduje, aby sa v tele požiadavky nachádzali informácie o identifikátore študijného programu a o zapnutí emailových notifikácií. Tieto informácie budú následne nastavené prihlásenému používateľovi.

```
@RestController
@RequestMapping("/api/profile")
public class ProfileController {
    private final ProfileService userService;

    @Autowired
    public ProfileController(ProfileService userService) {
        this.userService = userService;
    }

    @GetMapping
    public UserDto getUser(@RequestParam("userId") String userId) {
        return userService.getUser(Long.valueOf(userId));
    }

    @GetMapping("/info")
    public UserInfoDto getUserInfo() {
        return userService.getUserInfo();
    }
```

```
    @PostMapping("/info")
    public void setUserInfo(@RequestBody Map<String, String> body) {
        userService.setUserInfo(body.get("studyProgramId"),
                Boolean.parseBoolean(body.get("mailNotificationsEnabled")));
    }
}
```

Kód 3.5: Príklad spracovania HTTP požiadaviek serverovou aplikáciou

3.6 Zoznam otázok

Aplikácia musí používateľovi poskytnúť zoznam všetkých položených otázok. Zoznam otázok je dostupný na ceste '/list'. Musíme tiež vyriešiť problém, kedy by portál obsahoval stovky otázok. Bolo by neefektívne načítať ich všetky naraz. To sa dá vyriešiť pomocou paginácie, alebo pomocou podmieneného načítavania otázok v prípade dosiahnutia dna stránky. Zvolili sme druhú možnosť.

Komponent priradený ceste '/list' je *QuestionListComponent*. Typ zoznamu závisí od parametrov, ktoré pridáme k ceste. Ak chceme dostať zoznam všetkých otázok, využijeme cestu '/list' bez parametrov. Ak by sme chceli dostať zoznam otázok v používateľom sledovaných kategóriách, využijeme cestu '/list?followed=true'. Ak by sme chceli dostať zoznam otázok nejakej konkrétnej kategórie, využijeme cestu 'list?category=x', kde x je identifikátor kategórie.

```
export class QuestionListComponent {
  params: Params;

  constructor(private route: ActivatedRoute) {
    this.route.queryParams.subscribe(params => {
      this.params = params;
      this.listQuestionsService.loadInitial(
        params['followed'], params['category'], this.searchPhrase);
    });
  }
  ...
}
```

Kód 3.6: Získanie parametrov z cesty

Komponent *QuestionListComponent* pracuje so servisom *QuestionListService* a s parametrami, ktoré získal z cesty. Pri počiatočnom načítaní stránky alebo pri zmene vyhľadávaného výrazu otázok požiada komponent prostredníctvom servisu o načítanie

prvotných otázok cez jeho metódu *loadInitial*. Dáta o zozname otázok a o stránke načítavaných otázok sú držané v servise. Náhľady načítaných otázok sa následne vizuálne zobrazia v komponente, ktorý z dát každej otázky skonštruuje nový komponent *QuestionPreviewComponent*. V prípade, že sa používateľ dostane na dno zoznamu otázok, zavolá sa metóda servisu *loadMore*, ktorá načíta dodatočné otázky (ak sú dostupné).

Pri metóde *loadInitial* chceme začať načítavať otázky odznovu. Zoznam už načítaných otázok vynulujeme a stránku otázok nastavíme na hodnotu 0. Otázky teda načítavame od úplného začiatku. Odošleme HTTP požiadavku, ktorú nám poskytne metóda getRequest. Odpoveď pridáme do triednej premennej zoznamu otázok a zvýšime stránku otázok o 1.

Pri metóde *loadMore* chceme pokračovať v načítavaní otázok pre momentálne parametre. Odošleme HTTP požiadavku, ktorú nám poskytne metóda getRequest. Odpoveď pridáme do triednej premennej zoznamu otázok a zvýšime stránku otázok o 1. Na rozdiel od metódy loadInitial nenulujeme zoznam a číslo stránky.

```
export class QuestionListService {
  questions: Question[];
  page = 0;
  limit = 10;

  constructor(private questionService: QuestionService) {}

  private getRequest(followed: any, category: any, searchPhrase: string) {
    ... vracia HTTP požiadavku, ktorá je skonštruovaná na základe parametrov
  }

  loadInitial(followed: any, category: any, searchPhrase: string) {
    this.questions = undefined;
    this.page = 0;
    this.getRequest(followed, category, searchPhrase).subscribe(questions => {
      this.questions = []
      this.questions.push(...questions)
    })
    this.page += 1;
  }

  loadMore(followed: any, category: any, searchPhrase: string): void {
    this.getRequest(followed, category, searchPhrase).subscribe(questions => {
      this.questions.push(...questions)
    })
    this.page += 1;
  }
```

```
}
```

Kód 3.7: Definícia servisu QuestionListService

Po odoslaní HTTP požiadavky o zoznam otázok ju server spracuje v triede typu @RestController. Z požiadavky vyberie všetky parametre a posunie ich ďalej na spracovanie metóde *getQuestions* triedy typu @Service. Táto metóda na základe parametrov iba rozhodne, či chceme získavať všetky otázky, otázky sledovaných kategórií alebo otázky nejakej konkrétnej kategórie.

```
@GetMapping("/questions")
public List<QuestionDto> getQuestions(
    @RequestParam(value = "followed") boolean followed,
    @RequestParam(value = "category") Long category,
    @RequestParam(value = "phrase") String phrase,
    @RequestParam(value = "page", defaultValue = "0") int page,
    @RequestParam(value = "limit", defaultValue = "50") int limit) {
    return questionService
            .getQuestions(phrase, followed, category, page, limit);
}
```

Kód 3.8: Spracovanie HTTP požiadavky o získanie zoznamu otázok

Ako príklad si ukážeme získavanie otázok pre konkrétnu kategóriu. Otázky musíme získať z databázy pomocou dopytu v jazyku, ktorý je využívaný frameworkom Hibernate. Z tabuľky otázok získame všetky otázky a následne k ním načítame dodatočné údaje z ostatných tabuliek. Množinu otázok následne vyfiltrujeme tak, aby ostali iba otázky pripadajúce kategórií ktorá nás zaujíma a aby ich nadpis alebo obsah obsahoval text, ktorý používateľ zadal vo vyhľadávaní otázok. Otázky následne zoradíme zostupne podľa ich poslednej aktivity.

```
@Query("SELECT distinct q FROM questions q " +
    "left join fetch q.viewers viewers " +
    "left join fetch q.votes votes " +
    "left join fetch q.answers answers " +
    "where q.category.id = :categoryId " +
    "and (q.title LIKE %:phrase% or q.entryTextUnformatted LIKE %:phrase%) " +
    "order by q.lastActivity desc nulls last")
List<Question> findAllByCategory(@Param("categoryId") Long categoryId,
                                 @Param("phrase") String phrase);
```

Kód 3.9: Získanie zoznamu otázok z databázy podľa kategórie

V zozname otázok, ktorý dostaneme z databázy ešte musíme vyriešiť stránkovanie. V HTTP požiadavke nám bolo zaslané číslo stránky a veľkosť jednej stránky. Využijeme Java Stream API [25]. V zozname pomocou metódy *skip* preskočíme počet otázok vyrátaný vzorcom "stránka * veľkosť jednej stránky". Otázky získavame od toho miesta a ich počet je určený veľkosťou jednej stránky prostredníctvom metódy *limit*. Objekty využívané na prístup ku dátam nie je vhodné vracať ako odpoveď na HTTP požiadavky, preto ich ešte pomocou metódy *map* napamujeme na objekty určené špeciálne pre tento prenos.

```
@Transactional
public List<QuestionDto> getQuestionsByCategory(
        String phrase, long categoryId, int page, int limit) {
    return questionRepository.findAllByCategory(categoryId, phrase)
            .stream()
            .skip((long) page * limit).limit(limit)
            .map(question ->
                    entityToDtoService.questionToQuestionDto(question, null))
            .toList();
}
```

Kód 3.10: Riešenie stránkovania zoznamu otázok

3.7 Vytváranie otázok

Otázky je možné vyvárať na ceste '/ask'. Pre vytvorenie otázky je potrebné zadať nadpis, vyplniť telo otázky, vybrať kategóriu do ktorej otázka patrí a prípadne zvoliť možnosť anonymity.

Pre získanie textu tela otázky nám nestačí bežné textové pole. Chceme aby bol textový vstup formátovateľný a poskytoval ďalšiu funkcionalitu. Využili sme pre to textový editor Quill [26] poskytovaný treťou stranou, ktorý sme prispôsobili naším potrebám. Editor nám povoľuje umožniť používateľom formátovať text, vkladať videá z externých zdrojov, vkladať výpisy kódov, alebo vkladať matematické rovnice. Mimo toho sme editor nakonfigurovali tak, aby umožňoval nahrávanie obrázkov pomocou HTTP požiadaviek priamo na náš server.

Pri voľbe kategórie sme museli riešiť dva problémy. Prvý problém sa týkal toho, ako ušetriť používateľa pred zdĺhavým hľadaním kategórie v prípade, keď ich budeme mať desiatky až stovky. Problém sme vyriešili pridaním vyhľadávania kategórie do ponuky

kategórií. Po každom novom znaku sa vyfiltrujú iba tie kategórie, ktoré vyhovujú vyhľadávanému výrazu. Druhý problém sa týkal prípadu, kedy bude mať viacero kategórií rovnaké meno, no iné rodičovské kategórie. Kategórie majú štruktúru stromu. Problém sme vyriešili rekurzívnym nájdením cesty až ku koreňu pre každú kategóriu. V odpovedi na požiadavku o zoznam kategórií sme ku každej kategórií prebalili aj túto cestu. Cesta je zobrazovaná pri každej kategórii a používateľ vie rozlíšiť medzi dvoma rozličnými kategóriami s rovnakým menom.

Po vyplnení potrebných údajov bude zaslaná HTTP POST požiadavka na server o vytvorenie otázky. V požiadavke sa pošle nadpis, text tela otázky vo formátovanom tvare, text tela otázky v neformátovanom tvare, identifikátor kategórie a voľba zachovania anonymity otázky. Editor Quill formátuje text prostredníctvom HTML. Text vo formátovanom tvare je typu HTML a obsahuje všetky formátovania ako nadpisy, podčiarknutia textu a podobne. Ak zobrazujeme otázku, tak využívame formátovaný text. Text v neformátovanom tvare obsahuje čistý text bez hocakého formátovania. Tento neformátovaný text využívame pri vyhľadávaní otázok.

```
this.http.post<string>(environment.apiUrl + '/question', {
  'title': title,
  'text': text,
  'unformattedText': unformattedText,
  'categoryId': categoryId,
  'isAnonymous': isAnonymous
}, {
  headers: new HttpHeaders({'Content-Type': 'application/json', 'ngsw-bypass': 'true'}),
  withCredentials: true,
})
```

Kód 3.11: Vytvorenie HTTP požiadavky pridania novej otázky

3.8 Profilové fotky

Používateľom je umožnené nastaviť si profilovú fotku dvoma spôsobmi. Buď využitím predvoleného systému avatarov, alebo nahraním vlastného obrázku.

Keďže sme chceli dať portálu nejakú identitu a oživiť ho, rozhodli sme sa, že každý používateľ bude mať na začiatku predvolený avatar náhodne vygenerovanej postavičky. Využili sme webový servis tretej strany DiceBear [27], ktorý ponúka túto funkcionalitu. Zaslaním nejakého textového reťazca pomocou HTTP požiadavky bude z tohoto reťazca vygenerovaný náhodný avatar, ktorý dostaneme prostredníctvom odpovede. Tento avatar je vo vektorovom formáte '.svg' a jeho veľkosť je v jednotkách kilobajtov.

Z rovnakého reťazca dostaneme vždy rovnaký avatar. Používateľ si môže vybrať nový avatar a uložiť ho. V takom prípade mu priradíme reťazec daného avataru ku profilu v databáze.

Nahratie obrázku funguje tak, že používateľ prostredníctvom webovej stránky zašle obrázok HTTP POST požiadavkou na server. Server daný obrázok uloží do priečinku na to určenom a cestu k obrázku priradí k používateľskému profilu v databáze.

3.9 Podpora zariadení

3.9.1 Responzívny dizajn

Webová aplikácia bola navrhnutá tak, aby podporovala všetky relevantné zariadenia s rôznymi veľkosťami obrazovky. Prispôsobili sme ho pre smartfóny, tablety aj monitory rôznej veľkosti. Niekedy bolo potrebné pre niektoré veľkosti zmeniť aj celé rozloženie stránky. Pre zariadenia menšej šírky sme presunuli navigačnú lištu do bočného panelu. Responzívny dizajn sa nám podarilo aplikovať v rozsahu celej aplikácie bez toho, aby sme obetovali dizajnovú identitu, či funkcionalitu pre niektoré veľkosti obrazovky.

3.9.2 Progresívna webová aplikácia

Webová aplikácia spĺňa podmienky progresívnej webovej aplikácie. Webovú aplikáciu je možné nainštalovať a zabezpečiť tak, aby sa správala podobne ako natívna aplikácia. Nainštalovaná progresívna aplikácia podporuje napríklad zasielanie upozornení, prístup k polohe zariadenia, sledovanie menenia orientácie zariadenia a podobne. Používateľovi je prístupná v zozname aplikácií a nemusí sa ku nej dostávať prostredníctvom webového prehliadača. Progresívne webové aplikácie taktiež zefektívňujú načítavanie a rýchlosť stránky. Po prvotnom načítaní stránky sú načítané údaje udržiavané v pamäti. Tým pádom sa budú nasledujúce načítania blížiť rýchlosti natívnej aplikácie.

3.10 Nasadenie aplikácie

Aplikáciu sme nasadili na Ubuntu [15] server. Priradili sme jej doménu unibask.sk a za pomoci projektu Let's Encrypt [28] aj SSL certifikát. Databáza beží na štandardnom PostgreSQL porte 5432 a kvôli zvýšeniu bezpečnosti sme k nej zakázali prístup z vonkajšieho prostredia. Serverová časť aplikácie beží na webovom serveri Tomcat [16] a je jej priradený port 8443. Klientská časť aplikácie beží na webovom serveri Nginx [17] a je jej priradený štandardný HTTPS port 443. Počúva však aj na štandardnom HTTP porte 80 a zabezpečuje presmerovanie http://unibask.sk na https://unibask.sk.

Na webovom serveri Nginx sme nastavili reverzné proxy. Požiadavky smerujúce na https://unibask.sk/api sú vnútorne presmerované na port 8443 a o ich spracovanie sa teda stará serverová časť aplikácie. Dôvod prečo sme sa rozhodli použiť reverzné proxy je ten, že používatelia zvyknú mať firewallom povolené iba štandardné porty. Ak by sme nechali aplikáciu bežať na porte 8443, mohlo by sa stať, že by ľuďom bol zamietnutý prístup. Štandardné porty, ktoré môžeme využiť sú porty 80 a 443. Na porte 443 už beží Nginx server a port 80 neponúka dostatočné zabezpečenie. Použitím reverzného proxy používatelia komunikujú iba s portom 443. V konfigurácií Nginx sme niektorým súborom nastavili čas existencie v pamäti na 30 dní, aby sme zamedzili ich zbytočnému opätovnému prenášaniu po sieti. Pri nasadení novej verzie aplikácie sú súborom vygenerované nové unikátne názvy, takže sa nemusíme báť neaktuálnosti stránky na strane používateľa. Nastavili sme aj kompresiu pre zmenšenie veľkosti prenášaného obsahu.

```
server {
    listen 80;
    server_name unibask.sk www.unibask.sk;
    return 301 https://unibask.sk$request_uri;
}

server {
    listen 443 ssl;

    root /var/www/unibask.sk/html;
    index index.html index.htm index.nginx-debian.html;
    server_name unibask.sk www.unibask.sk;
    ssl_certificate .../fullchain.pem;
    ssl_certificate_key .../privkey.pem;
    gzip on;

    location / {
            try_files $uri $uri/ /index.html;
    }

    location ~* \.(?:css|js|jpg|jpeg|gif|png|ico|cur|gz|svg|svgz|mp4|mp3)$ {
        expires 1M;
        add_header Cache-Control "max-age=2629746, public";
    }

    location /api {
            proxy_pass https://localhost:8443/api;
    }
}
```

Kód 3.12: Konfigurácia webového servera

Záver a vyhodnotenie

Portál bol úspešne navrhnutý a vytvorený v súlade so zadaním projektu. Vo februári sme portál nasadili na predmet 'Programovanie 4' katedry aplikovanej informatiky, ktorý sa konal v období letného semestra 2021/2022. Nasadenie bolo z technického hľadiska bezproblémové – nevyskytli sa žiadne neočakávane ťažkosti. Adoptácia portálu študentami vyzerala spočiatku nádejne, no neskôr sa ich motivácia pridávať nové otázky zmenšovala. Nepodarilo sa nám udržať ich interakciu s portálom.

Počas semestra bolo položených 6 otázok. Otázky boli zodpovedané 11 odpoveďami. K otázkam a odpovediam bolo pridaných 5 komentárov. Do portálu sa zaregistrovalo 60 používateľov. Dokopy si všetci používatelia prezreli otázky 210-krát. Najväčšiu reputáciu (12) získal vyučujúci. Druhú najväčšiu reputáciu (7) získal študent. Možnosť sledovať kategórie využilo 7 používateľov a možnosť sledovať otázky využilo 5 používateľov. Zasielanie mailových notifikácií povolili 4 používatelia. Posledná otázka bola položená na konci tretieho týždňa semestra.

Myslíme si, že po technickej stránke bol portál dostačujúci. Podarilo sa nám zakomponovať všetku funkcionalitu, ktorú sme plánovali a s konečným výsledkom aplikácie sme spokojný. Čo však podľa nás chýbalo k jeho úspechu bola motivácia študentov využívať ho. Aby bol portál úspešný, bolo by potrebné propagovať ho na fakultnej úrovni, aby sa zväčšilo jeho publikum a aby bola udržaná jeho neustála aktivita. Úspech portálu je priamo úmerný s aktívnou veľkosťou komunity. Túto komunitu sa nám však s naším dosahom nevydarilo vybudovať.

Máme však aj nápady na zlepšenie technickej stránky portálu. Prioritná funkcionalita, ktorú portál potrebuje je pridanie spôsobu motivácie používateľov. To sa dá spraviť rôznymi oceneniami za míľniky, akými sú napríklad vytvorenie alebo zodpovedanie niekoľkých otázok. Ešte atraktívnejšie médium motivácie je poskytnutie moderátorských práv používateľom, ktorý presiahnu určitú hranicu reputácie. Takýmto spôsobom by sa portál mohol moderovať aj sám. Používatelia s vysokou reputáciou sú dôveryhodný a bolo by im umožnené vykonávať rôzne moderátorské úkony. To nás dostáva ku ďalšiemu bodu rozšírenia aplikácie, ktorým je pridanie moderátorského rozhrania. V mo-

mentálnej verzii aplikácie neexistuje moderátorské rozhranie a moderácia je umožnená výhradne prostredníctvom databázy. Používatelia si pomocou spätnej väzby vyžiadali funkcionalitu označenia otázok za prečítané. Takto sa dá docieliť, aby používatelia videli v zozname otázok iba novú aktivitu.

Literatúra

[1] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. How do programmers ask and answer questions on the web? (nier track). In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, page 804–807, New York, NY, USA, 2011. Association for Computing Machinery.

[2] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. Design lessons from the fastest q&a site in the west. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, page 2857–2866, New York, NY, USA, 2011. Association for Computing Machinery.

[3] Annie T. T. Ying. Mining challenge 2015: Comparing and combining different information sources on the stack overflow data set. In *The 12th Working Conference on Mining Software Repositories*, page to appear, 2015.

[4] Ivan Srba, Milos Savic, Maria Bielikova, Mirjana Ivanovic, and Cesare Pautasso. Employing community question answering for online discussions in university courses: Students' perspective. *Computers & Education*, 135:75–90, 2019.

[5] Microsoft Teams. `https://www.microsoft.com/microsoft-teams`. 2022-05-01.

[6] Discord. `https://discord.com/`. 2022-05-01.

[7] Stack Exchange. `https://stackexchange.com/`. 2022-05-01.

[8] Askalot. `https://askalot.fiit.stuba.sk/demo/`. 2022-05-01.

[9] Spring Framework. `https://spring.io/`. 2022-05-01.

[10] Angular. `https://angular.io/`. 2022-05-01.

[11] What is a REST API? `https://www.ibm.com/cloud/learn/rest-apis`. 2022-05-01.

[12] PostgreSQL. `https://www.postgresql.org/`. 2022-05-01.

[13] Hibernate. `https://hibernate.org/orm/`. 2022-05-01.

[14] Logá a symboly Univerzity Komenského. https://uniba.sk/o-univerzite/loga-a-symboly-uk/. 2022-05-01.

[15] Ubuntu. https://ubuntu.com/. 2022-05-01.

[16] Tomcat. https://tomcat.apache.org/. 2022-05-01.

[17] Nginx. https://www.nginx.com/. 2022-05-01.

[18] The HTTP protocol. https://www.ibm.com/docs/en/cics-ts/5.3?topic=concepts-http-protocol. 2022-05-01.

[19] Java Annotations. https://docs.oracle.com/javase/tutorial/java/annotations/. 2022-05-01.

[20] JSON. https://www.json.org/json-en.html. 2022-05-01.

[21] Angular Component. https://angular.io/api/core/Component. 2022-05-01.

[22] Angular Service. https://angular.io/guide/architecture-services. 2022-05-01.

[23] Angular Module. https://angular.io/guide/architecture-modules. 2022-05-01.

[24] Using HTTP cookies. https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies. 2022-05-01.

[25] Java Stream API. https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html. 2022-05-01.

[26] Quill. https://quilljs.com/. 2022-05-01.

[27] DiceBear Avatars. https://avatars.dicebear.com/. 2022-05-01.

[28] Let's Encrypt. https://letsencrypt.org/. 2022-05-01.