Web aplikacija kao platforma za postavljanje i korištenje kvizova

Šumanovac, Leon

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni diplomski studij Računarstvo

WEB APLIKACIJA KAO PLATFORMA ZA POSTAVLJANJE I KORIŠTENJE KVIZOVA

Diplomski rad

Leon Šumanovac

Osijek, 2025

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju

Ocjena diplomskog rada na sveučilišnom diplomskom studiju

Ime i prezime pristupnika:	Leon Šumanovac
Studij, smjer:	Sveučilišni diplomski studij Računarstvo
Mat. br. pristupnika, god.	D1331R, 07.10.2022.
JMBAG:	0165079459
Mentor:	prof. dr. sc. Dominika Crnjac Milić
Sumentor:	doc. dr. sc. Hrvoje Leventić
Sumentor iz tvrtke:	Tomislav Kaučić, inž.el.
Predsjednik Povjerenstva:	izv. prof. dr. sc. Alfonzo Baumgartner
Član Povjerenstva 1:	prof. dr. sc. Dominika Crnjac Milić
Član Povjerenstva 2:	prof. dr. sc. Krešimir Nenadić
Naslov diplomskog rada:	Web aplikacija kao platforma za postavljanje i korištenje kvizova
Znanstvena grana diplomskog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Izraditi Ruby on Rails web aplikaciju kao platformu za postavljanje kvizova s ciljem poticanja stjecanja novih znanja iz različitih područja. Registriranom korisniku se omogućuje postavljanje kviza, dok registracija nije nužna za pristup kvizovima i njihovo rješavanje. Postavljeni kvizovi moraju biti javno dostupni radi postizanja širokog kruga korisnika. Potrebno je pomoću PostgreSQL osigurati mogućnost izrade baze podataka. Potrebno je istražiti programski okvir Ruby on Rails te istaknuti prednosti njegova korištenja naspram sličnih programskih okvira (primjerice

Datum ocjene pismenog dijela diplomskog rada od strane mentora:	26.01.2025.
Ocjena pismenog dijela diplomskog rada od strane mentora:	Izvrstan (5)
Datum obrane diplomskog rada:	20.02.2025.
Ocjena usmenog dijela diplomskog rada (obrane):	Izvrstan (5)
Ukupna ocjena diplomskog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:	21.02.2025.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 21.02.2025.

Ime i prezime Pristupnika:	Leon Šumanovac
Studij:	Sveučilišni diplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D1331R, 07.10.2022.
Turnitin podudaranje [%]:	8

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija kao platforma za postavljanje i korištenje kvizova**

izrađen pod vodstvom mentora prof. dr. sc. Dominika Crnjac Milić

i sumentora doc. dr. sc. Hrvoje Leventić

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD

Kvizovi u današnjem društvu imaju veliku ulogu kao način zabave, obrazovanja i socijalizacije. Prisutni su u raznim emisijama koje se prikazuju na televiziji, u kvizovima u školama na kraju predavanja kako bi se utvrdila usvojenost prođenog gradiva, pa sve do pub kvizova koji promoviraju druženje i poboljšavaju timski rad. Kvizovi mogu biti raznih tematika, od kvizova općeg znanja u kojem sudionici odgovaraju na pitanja iz raznih područja, do kvizova specifičnog područja u kojima sudionici evaluiraju svoje znanje iz tog konkretnog područja. Osim svoje raznovrsnosti, kvizovi sa sobom unose i elemente natjecanja, što ih čini popularnijima i privlačnijima za različite dobne skupine.

Zadatak ovog rada bio je analizirati različite programske okvire (engl. *framework*) za izradu web aplikacija, te usporediti ih s programskim okvirom *Ruby on Rails*. Također, zadatak rada bio je i izrada web aplikacije za kreiranje i korištenje kvizova s pomoću programskog okvira *Ruby on Rails* (u nastavku teksta se koristi skraćeni naziv *Rails*) u suradnji s tvrtkom *Bamboo Lab*. Zadatak rada je da prijavljeni korisnik ima mogućnost kreiranja novih kvizova i odgovaranja na postojeće kvizove, dok gostujući korisnik, odnosno korisnik koji nije prijavljen, ima samo mogućnost korištenja postujućih kvizova uz upisivanje vlastitog korisničkog imena pod kojim će njegov rezultat biti pamćen.

U drugom poglavlju su analizirana trenutna rješenja aplikacija za kreiranje i rješavanje kvizova koja su dostupna na tržištu. Trećim poglavljem su detaljnije opisani programski okviri s naglaskom na *Ruby on Rails* pomoću kojeg je napravljeno programsko rješenje ovog rada, te je odrađena njegova usporedba s programskim okvirima *Django, ASP.NET, CakePHP* i *Elixir*. U četvrtom poglavlju je opisano programsko rješenje web aplikacije raspodijeljeno u četiri cjeline: planirano rješenje, instalacija programskog kostura i postavljanje okruženja, implementacija i izrada funkcionalnosti te analiza programskog rješenja. U petom poglavlju nalazi se zaključak rada u kojem se navode moguća proširenja programskog rješenja.

2. APLIKACIJE ZA RJEŠAVANJE KVIZOVA

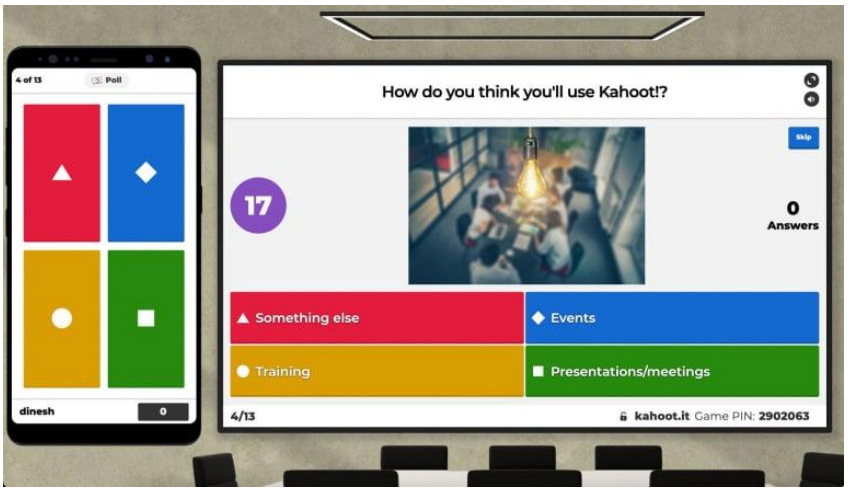
Internet je pun raznih web i mobilnih aplikacija za kreiranje i rješavanje kvizova. Aplikacije se razlikuju po ključnim značajkama poput načina rješavanja kvizova, bodovanja i mogućnosti integracija s drugim aplikacijama. Odabir korištenja pojedinih aplikacija ovisi najviše o njihovim integracijama i funkcionalnostima jer je intuitivnost problem samo u početku korištenja aplikacije.

Ovo poglavlje predstavlja uvid na postojeća rješenja aplikacija na tržištu koja omogućavaju korisnicima kreiranje i rješavanje kvizova.

2.1. Kahoot!

Kahoot! je jedna od najpopularnijih platformi za kvizove koja omogućava korisnicima kreiranje i rješavanje kvizova u stvarnom vremenu. Koristi se u poslovnim, edukacijskim i zabavnim okruženjima. *Kahoot!* nudi pitanja na pisanje kratkih odgovora i pitanja s odabirom jednog ili više točnih odgovora. U pitanja je moguće dodati slike, videozapise i zvukove [1] kako bi bilo moguće lakše postaviti pitanje i samim time pruža korisnicima veći spektar pitanja koja mogu postavljati unutar pojedinih kvizova. Također, *Kahoot!* nudi i element natjecanja u smislu da svako pitanje donosi bodove, a bodovi se računaju na brzini i na točnosti odgovora, što omogućuje interaktivnije i zanimljivije rješavanja kvizova u stvarnom vremenu.

Ima mogućnost rješavanja kvizova s ugrađenim načinom bodovanja ovisno o brzini i točnosti odgovaranja na pitanja. To ga čini odličnim za učitelje, predavače, nastavnike i profesore jer omogućava praćenje napretka ispitanika kroz svoje izvještaje. Platforma također nudi mogućnost postavljanja vremenskog ograničenja na pitanja. *Kahoot!* dolazi u besplatnoj verziji koja omogućuje osnovne funkcionalnosti i ograničen broj sudionika, dok napredna verzija koja se plaća nudi veći broj sudionika, bolje izvještavanje i personalizaciju dizajna. Dostupan je kao web i mobilna aplikacija te samim time omogućava korisnicima pristupanje kvizovima iz bilo kojeg mobilnog uređaja ili računala povezanog na internet. Također ima integraciju s *Microsoft Teams* i *Google Classroom* te samim time omogućava korisnicima rješavanje kvizova u stvarnom vremenu preko mobilne aplikacije unošenjem korisničkog imena i odgovarajućeg *PIN*-a.



Slika 2.1. Prikaz izgleda *Kahoot!* aplikacije [2]

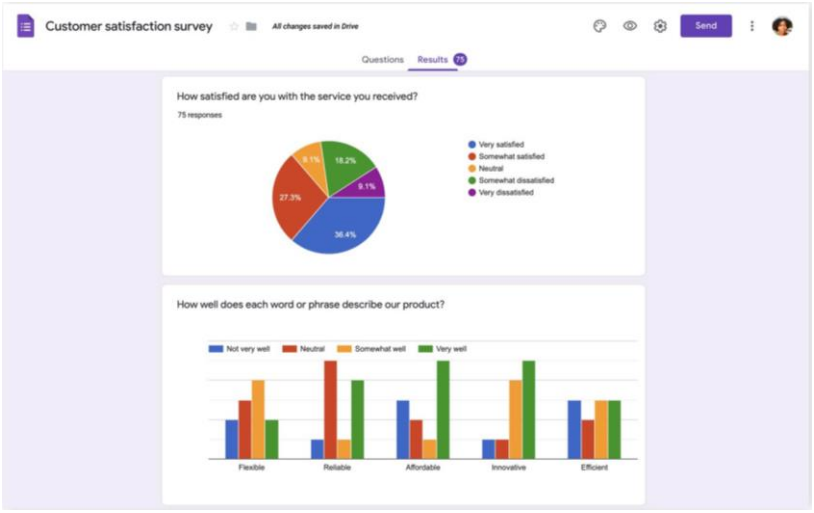
2.2. Socrative

Socrative je platforma koja omogućava učiteljima, trenerima i raznim edukatorima da na intuitivan način kreiraju kvizove, testove i ankete za procjenu znanja pojedinca ili grupe. [3] Također, *Socrative* nudi kvizove i izvještaje u stvarnom vremenu. Postoje različite vrste kvizova u *Socrativeu*, kao što su *Quiz*, *Space Race* i *Exit Tickets*. *Quiz* omogućava korisnicima rješavanje kvizova bez vremenskog ograničenja. *Space Race* omogućava korisnicima bodovno natjecanje na brzinu i točnost odgovora na pitanja. *Exit Tickets* su kratke ankete koje su predviđene za korištenje na kraju predavanja za prikupljanje povratnih informacija u svrhu procjene razumijevanja predavanja. *Socrative* dolazi u besplatnoj i plaćenoj verziji, a razlika između njih je jednaka kao i kod *Kahoot!*-a opisanog u poglavlju 2.1.

Slika 2.2. Prikaz *Space Race* mogućnosti *Socrative*-a [4]

2.3. Google Forms

Google Forms je jednostavna platforma koja je dio *Google Workspacea* koja omogućava kreiranje anketa i kvizova. Ova platforma ne nudi mogućnost rješavanja kviza s uvidom u odgovore u stvarnom vremenu, nego generira izvještaj tek nakon što korisnik preda kviz. Omogućava integraciju s raznim *Google* alatima, od kojih je jedan *Google Classroom*, što omogućava jednostavnost dijeljenja kviza. [5] Pristup kvizu se može dodijeliti putem jedinstvenog URL-a kojeg je moguće poslati na e-mail adrese ispitanika tijekom procesa izrade kviza. Također nudi jednostavnu mogućnost personalizacije izgleda kviza kroz promjenu boja, dodavanje slika i videozapisa u pitanjima. Odgovori mogu biti tekstualnog oblika, označavanje jednog ili više točnih odgovora i skale brojeva. [6] Osim integracija i novih vrsta pitanja, prednost *Google Formsa* je što dolazi samo u besplatnoj verziji. Još jedna prednost *Google Formsa* je što su rezultati kvizova privatni, odnosno, ispitanici nemaju pristup točnim odgovorima na kraju rješavanja kviza što ga čini odličnim izborom za edukacijske svrhe.

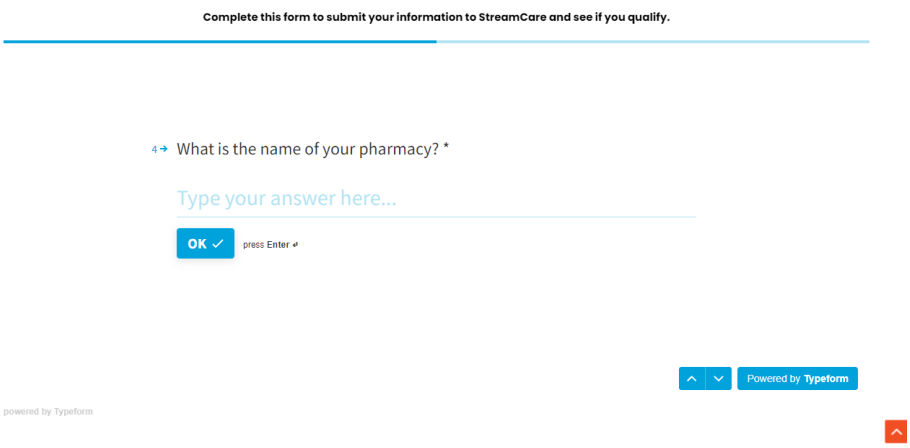


Slika 2.3. Prikaz *Google Forms* rezultata [7]

2.4. Typeform

Typeform je platforma za izradu anketa i kvizova. Omogućava korisnicima kreiranje jednostavnih i intuitivnih obrazaca. Sučelje *Typeforma* je tako da ispitanici u svakom koraku kviza vide samo jedno pitanje. [8] Takav pristup u dizajnu sučelja omogućava korisnicima fokusiranost na pitanje umjesto na cijeli kviz, dajući im čist vizualni prostor da razmisle prije nego što predaju odgovor. Odgovori mogu biti tekstualni, izbor jednog ili više točnih odgovora, skala brojeva i *drag-and-drop* slika. Nudi analiziranje rezultata u stvarnom vremenu uz razne korisničke statistike.

Typeform nudi besplatnu verziju koja je ograničena s brojem odgovora i funkcionalnostima, a puni pristup se može dobiti uz pretplatu. Također, plaćena verzija nudi mogućnosti prilagodbe dizajna, veći broj integracija i bolje analitike.



Slika 2.4. Prikaz *Typeform* pitanja [9]

2.5. Quizizz

Quizizz je platforma za izradu kvizova i anketa. Nudi brzo, lako i jednostavno kreiranje kvizova i pitanja. Također nudi postojeću bazu podataka s unaprijed pripremljenim pitanjima koja se mogu dodati u pojedini kviz. Aplikacija za izradu kvizova nudi pitanja na izbor jednog ili više odgovora, unos teksta i prepoznavanja sa slike uz mogućnost postavljanja određenog vremena za odgovor na pojedina pitanja. Omogućava kreiranje *team mode*-a u kojem se sudionici spajaju u timove i zajedno odgovaraju na postavljena pitanja. [10] Nudi integraciju s raznim drugim alatima poput *Google Classroom*-a, što olakšava organizaciju kvizova u stvarnom vremenu i praćenje napretka pojedinih korisnika.

Slika 2.5. Prikaz Quizizz pitanja [11]

3. PROGRAMSKI OKVIRI

U današnje doba, s napretkom tehnologije, aplikacije postaju sve veće i zahtjevnije. Kako bi aplikacije mogle obrađivati velike količine podataka u stvarnom vremenu i obavljati zadatke i transakcije koje krajnji korisnik odabere, potrebno je imati dobro optimiziran, brz i točan programski kod. Najčešće na funkcionalnostima aplikacije ne radi samo jedna osoba, nego više njih. Zbog toga postoji mogućnost dolaska do dupliciranja koda zbog neusklađenosti programera, pisanja koda koji nije dovoljno brz ili koda, koji u određenim uvjetima, daje krivi rezultat. Programski okviri se koriste kako bi se riješili navedeni problemi.

Programski okviri su skupina programskih alata za razvoj aplikacija [12] koji pružaju mogućnost dijeljenja zajedničkih logičkih i funkcionalnih mogućnosti svim aplikacijama koje koriste isti programski okvir. Dijeljenje logike i funkcionalnosti među aplikacijama ima razne prednosti:

- smanjivanje vremena izrade aplikacije ili dodatnih funkcionalnosti u aplikaciji,
- smanjenje troškova izrade aplikacije,
- povećanje kvalitete koda,
- povećanje brzine izvođenja koda,
- povećanje točnosti koda.

Točnost koda se povećava na dijelovima koda koji koriste programski okvir, jer je taj dio koda dijeljen kroz velik broj aplikacija i samim time dobro testiran.

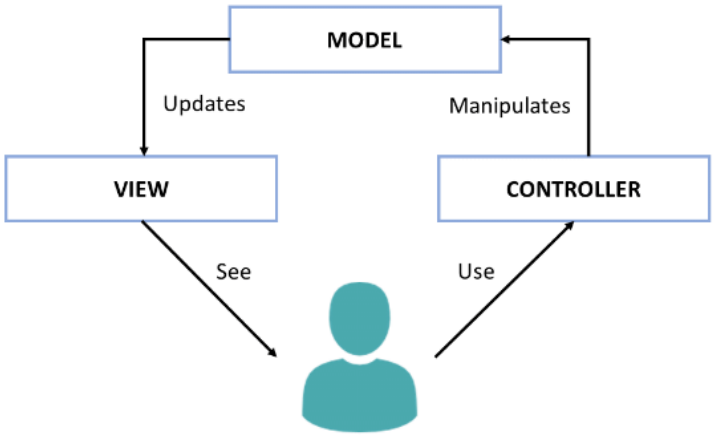
Programski okviri se dijele na 4 vrste: web, mobilni, *backend* i *frontend*. Temeljeni su na programskim jezicima, jer su oni sami po sebi dijelovi programskih jezika, stoga nije moguće koristiti isti programski okvir u različitim programskim jezicima. U nastavku rada će biti opisani web programski okviri *Ruby on Rails* koji je temeljen na programskom jeziku *Ruby*, *Django* koji je temeljen na programskom jeziku *Python*, *ASP.NET* koji je temeljen na *C#* i *CakePHP* koji je temeljen na *PHP*-u.

3.1. Ruby on Rails

Za izradu programskog rješenja ovog diplomskog rada se koristio programski okvir *Ruby on Rails* baziran na programskom jeziku *Ruby*. Napisao ga je David Heinemeier Hansson kao javno dostupan izvorni kod 2004. godine, te je prvi veći uspjeh postigao 2007. godine kada je *Apple* počeo prodavati *Mac OS X v10.5 „Leopard“* koji su imali instaliran *Ruby on Rails*. Kroz godine

su izlazile nove verzije, te je najnovija verzija *Rails* 8. Programsko rješenje ovog rada je pisano u verziji *Rails* 7 iz razloga što je to bila najnovija verzija u trenutku pisanja koda diplomskog rada.

Rails je *full-stack* web okvir koji koristi Model-Pogled-Kontroler (engl. *Model-View-Controller, MVC*) arhitekturni obrazac. Razlog korištenja *MVC* arhitekture je razdvajanje aplikacije na tri glavna dijela (model, pogled i kontroler), [13] te samim time poboljšavanje strukture koda, omogućava lakše i brže snalaženje u kodu, jednostavnije testiranje, i povećavanje skalabilnosti aplikacije. Glavna ideja *MVC* arhitekture je razdvajanje funkcionalnosti programskog koda od prikaza stranice na web-pregledniku.



Slika 3.1. Općeniti prikaz MVC Arhitekture [14]

Model označava dio koda koji opisuje objekte u stvarnom svijetu, te sadržava čiste podatke ili definira osnovne komponente aplikacije. [15] *Rails* koristi *ORM* (engl. *Object-Relational Mapping*), što znači da je korištenjem *Rails*-a moguće pozivati *SQL* (engl. *Structured query language*) kod na objektima bez pisanja samog *SQL*-a. U ovom diplomskom radu, neki od modela su *quiz, topic* i *choice* koji označavaju kviz, temu i jedan izbor u pitanjima s više ponuđenih odgovora. U *Rails*-u, modeli su izrađeni u *Ruby* datotekama (datoteke označene nastavkom .rb).

Pogled označava dio programskog koda koji korisnik vidi. U *Rails*-u, modeli su rađeni u *ERB* datotekama (engl. *Embedded Ruby*, datoteke označene nastavkom .erb). *ERB* datoteke omogućuju korisniku pisanje i *HTML* i *Ruby* koda unutar jedne datoteke, čime se omogućuje preglednije iščitavanje koda jer se cijeli prikaz nalazi na istom mjestu. *ERB* također omogućuje korisnicima izradu predložaka (engl. *template*), te spremanje koda unutar njih ako se isti kod koristi na više različitih mjesta, te jednostavnim pozivom predložaka, preglednici mogu iscrtati (engl. *render*) navedeni kod.

Kontroler označava dio koda koji je posrednik između modela i pogleda. On prima korisnikove zahtjeve i odabire što će učiniti s njima. U načelu, svaki model ima svoj kontroler, kako bi se moglo upravljati s njime. U ovom diplomskom radu neki od kontrolera su *quizzes_controller*, *topics_controller* i *quiz_sessions_controller*.

Rails olakšava upravljanje bazama podataka, podršku *RESTful* arhitekturi, te jednostavno kreiranje predložaka za sve dijelove *MVC*-a korištenjem ugrađenih *generator*-a, što programerima ubrzava i olakšava rad. Navedene mogućnosti su implementirane koristeći razne biblioteke unutar *Rails*-a. *Rails*, pošto je baziran na *Ruby*-u, također omogućava dodatno preuzimanje i podešavanje raznih *Gem*-ova. *Gem*-ovi su paketi programskog koda pisani u *Ruby*-u koji pružaju mogućnost korištenja dijelova koda koji su pisali drugi programeri. *Gem*-ovi osim što su *open-source*, korišteni su od drugih programera koji koriste *Ruby*, te samim time, kao i programski okviri, su bolje testirani i manje skloni greškama programera. *Gem*-ovi korišteni u radu su objašnjeni u dijelu izrade programskog rješenja.

Rails se koristi i u aplikacijama koje su puno veće od aplikacije koja je izrađena u svrhu ovog diplomskog rada. Neki od većih postojećih web aplikacija koje koriste *Rails* su: GitHub, Airbnb, Shopify i Kickstarter.

3.2. Django

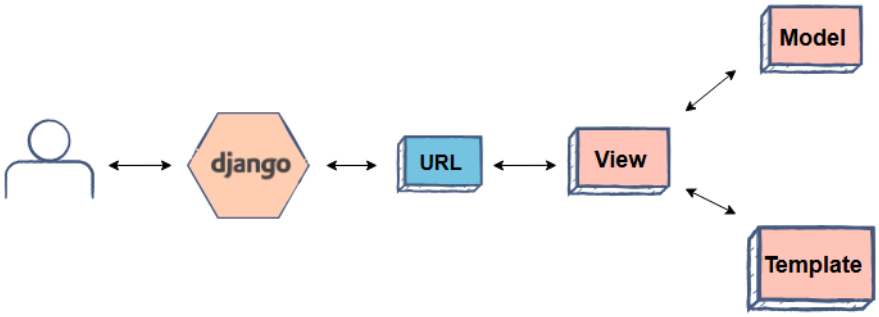
Django je web okvir koji je pisan u programskom jeziku *Python*, te za razliku od *Rails*-a, koristi *MVT* (engl. *Model-View-Template*) arhitekturni obrazac. [16] Dizajniran je u svrhu omogućavanja programerima brzu i efikasnu izradu web aplikacija s naglaskom na brzinu, jednostavnost i omogućavanje ponovnog korištenja zasebnih dijelova aplikacije.

MVT arhitektura je slična uobičajenoj *MVC* arhitekturi, no između njih postoje određene razlike. *Model* dio *MVT*-a označava *shemu* unutar baze podataka ili sami objekt koji definira strukturu podataka unutar *Django* aplikacije. *Django*, kao i *Rails*, koristi *ORM*, te samim time programerima omogućava rad s bazom podataka bez potrebe za pisanjem *SQL* koda. Za razliku od *Rails*-a, *Django* nema mogućnost automatskog kreiranja modela s pomoću korištenja terminala, nego programer treba ručno napisati modele. U slučaju promjene modela, potrebno je izgenerirati *migracijski* file, koji sam prepoznaje promjene između trenutnog modela i modela spremljenog u bazu, te ga je potrebno migrirati.

View dio *MVT*-a je posrednik koji prihvaća *HTTP* zahtjev, obrađuje ga i vraća *HTTP* odziv. U dijelu obrađivanja zahtjeva, *view* komunicira s *model*-om, te izvršava potrebne radnje na podacima, kao što su obrada podataka, sortiranje ili filtriranje, ovisno o uputama dobivenima iz zahtjeva.

Template dio *MVT*-a je tekstualna datoteka koja je zadužena za dio web aplikacije koji je prikazan korisniku. Najčešće je pisana u *HTML* formatu, no *template* datoteke mogu biti i drugih formata, kao što su *JSON* i *XML*. U njih se može pisati *Python* kod koji govori kako bi se dinamični dijelovi aplikacije, kao što su kvizovi, teme i pitanja unutar ovog diplomskog rada, trebali prikazati, što ga čini sličnim *ERB* datotekama kod *Rails*-a.

MVT arhitektura radi tako da korisnik komunicira s *Django* aplikacijom koristeći *URL* (engl. *Uniformed Resource Locator*) koji se prosljeđuje *MVT* arhitekturi, te ga *URL mapper* preusmjerava na odgovarajući *view*. Nakon što zahtjev dođe do *view*-a, *view* komunicira s *model*-om i dobiva odgovarajuće potrebne podatke iz baze podataka kako bi korisniku mogao iscrtati odgovarajući *template* uz podatke koje je primio od *model*-a.



Slika 3.2. Prikaz toka podataka unutar *MVT Django* arhitekture [17]

Django se također koristi u aplikacijama koje su puno veće od aplikacije napravljene za ovaj diplomski rad, a neka od većih imena koja koriste *Django* su *Instagram* i *Pinterest*.

U usporedbi s *Railsom*, *Django* ima prednost kod web aplikacija koje koriste strojno učenje jer je pisan u programskom jeziku *Python*, koji je u vremenu pisanja ovog diplomskog rada, vodeći programski jezik za pisanje koda za strojno učenje [19], te samim time ima prednost pri korištenju velike količine podataka. Međutim, sintaksa *Rails*-a i *Django*-a je vrlo slična, i slična je engleskom jeziku, te je samim time vrlo razumljiva i jednostavna za učenje. *Rails* koristi *MVC* arhitekturalnu strukturu, dok je *Django* jedini programski okvir koji koristi *MVT* arhitekturalnu strukturu, čime *Rails* ima prednost zato što je *MVC* struktura daleko popularnija. [20]

3.3. ASP.NET

ASP.NET je programski okvir za pisanje web aplikacija koji koristi programski jezik C#. [21] ASP.NET koristi *MVC* arhitekturalnu strukturu kao i *Rails*. Dio je .NET platforme koja sadrži razne alate i biblioteke koje pomažu u izgradnji raznih vrsta aplikacija. Osnovna platforma sadrži razne komponente koje se mogu koristiti u svim vrstama aplikacija, dok ostali programski okviri, kao što je ASP.NET, nasljeđuju i proširuju osnovnu platformu s komponentama koje se koriste za specifične vrste aplikacija.

ASP.NET u svom proširivanju osnovne .NET platforme omogućuje mogućnosti kao što su:

- Osnovni web okvir obrađuje web zahtjeve pomoću programskih jezika C#,
- Razor: pruža mogućnost modela pisanja koda po zasebnim web stranicama, što pojednostavljuje izradu korisničkog sučelja (time smanjuje potrebu za korištenjem kontrolera, čime čini kreiranje i upravljanje manjim web aplikacijama puno jednostavnije),
- Biblioteke za zajedničke web obrasce, kao što je *MVC*,
- Sustav autorizacije koji sadrži predloške stranica za prijavu i registraciju, razne biblioteke i bazu podataka. Omogućuje *2FA* (engl. *Two Factor Authorization*), te prijave pomoću vanjskih stranica, kao što su *Google* i *X*.

Prilikom kreiranja izgleda web stranica izgrađenih pomoću ASP.NET-a, koristi se *Blazor* – suvremeni *frontend* web programski okvir koji koristi HTML, CSS i C# [22] i omogućuje programeru brzu izgradnju skalabilnih web aplikacija. *Blazor* pruža mogućnost korištenja pojedinih komponenti na više različitih mjesta.

Programi koji koriste *Django* i *Rails* su većinski pisani u tekstualnom *editoru VS Code,* dok je tekstualni *editor* za pisanje ASP.NET web aplikacija *Visual Studio*. *Visual Studio* je manje prilagodljiv od *VS Code*-a, jer je predviđen za izgradnju .NET aplikacija. Prilikom pokretanja, odabire se koje će se vrste aplikacija kreirati, te se instaliraju za nju potrebni paketi usluga. ASP.NET web aplikacije se i dalje mogu kreirati koristeći *VS Code*, no zbog boljeg iskustva je preporučeno korištenje *Visual Studio*-a koji smanjuje vrijeme potrebno za početno postavljanje web aplikacije građene korištenjem ASP.NET-a.

Rails isključivo prati *MVC* arhitekturalni obrazac, dok je ASP.NET *modularni* programski okvir koji može koristiti razne pristupa kao što su *Web forms, MVC, Razor Pages* i *Blazor*. [23] To čini *Rails* pristupačnijim za učenje, pošto kada programer jednom shvati *MVC* obrazac, lako ga može

primijeniti na svaku aplikaciju, dok kod ASP.NET-a programeri imaju mogućnost izbora i boljih rješenja ovisno o web aplikaciji koju rade. Osim toga, *Rails* se fokusira na *DRY* (engl. *Don't Repeat Yourself*) načelo, čime smanjuje dupliciranje koda, te je čitljiviji jer je pisan u *Ruby*-u koji je temeljen na engleskom jeziku, dok je ASP.NET pisan u C#-u.

3.4. CakePHP

CakePHP je web programski okvir koji je pisan u programskom jeziku *PHP*, te koristi *MVC* arhitekturalni obrazac [24] isto kao i *Rails*. Programski okvir sadrži razne generatore za kreiranje *CRUD* (engl. *Create, Read, Update, Delete*) operacije [25], što olakšava programerima kreiranje novih aplikacija i samih mogućnosti unutar njih jer smanjuje potrebu za pisanjem dijelova koda koji su slični unutar web aplikacija. *CakePHP* je jednostavan za početno postavljanje tako da je sve što je potrebno je kreirati i konfigurirati bazu podataka koju će web aplikacija koristiti.

CakePHP je proširiv, u smislu da omogućava instalaciju dodatnih nadogradnji u vidu *plugin*-a, koji su jednaki *Rails gem*-ovima i *Django package*-ovima. Također, kao i svi ostali navedeni programski okviri, ima ugrađeni *ORM* koji olakšava kreiranje i upravljanje podacima iz baze podataka korištenjem objekata unutar *PHP*-a.

3.5. Phoenix

Phoenix je web programski okvir koji koristi *MVC* arhitekturalni obrazac, te je pisan u programskom jeziku *Elixir* [26], poznatom po svojim visokim performansama za programe u stvarnom vremenu. Za iscrtavanje podataka koristi *HEEx* (engl. *HTML + Embedded Elixir*) datoteke, na način sličan *ERB* datotekama kod *Rails*-a: omogućava izdvajanje komponenti koje se koriste na više mjesta i poziv njihovih *partial*-sa na mjestima gdje ih je potrebno iscrtati. Također, za razliku od ranije opisanih programskih okvira, *Phoenix* dolazi s ugrađenim uređivačem koda. Uređivač koda omogućava cijelom kodu pisanom u programskom okviru, bio on *HTML* ili *Elixir*, automatsko formatiranje na spremanju pomoću nadogradnje *ElixirLS*. [27]

Osim ugrađenog formatiranja, *Phoenix* sadrži i *compiler check*-ove. To znači da ako programer pozove određenu varijablu koja nije definirana ili ako napravi neku drugu sintaksnu pogrešku prilikom pisanja programa, tekstualni *editor* će ju označiti i izbaciti odgovarajuću pogrešku.

3.6. Zaključak izbora programskog okvira

Svi programski okviri navedeni u poglavljima 3.1. do 3.5. su vrlo slični. *Django* odstupa od široko popularnog *MVC* arhitekturalnog obrasca, dok ASP.NET programeru nudi više mogućih pristupa za način pisanja koda.

Rails nudi vrlo brz razvoj web aplikacije pomoću ponuđenih generatora, te omogućava ponovno korištenje velikog dijela koda koristeći *partial*-se i ORM. *Django* nudi odličnu mogućnost implementacije vlastitog koda sa strojnim učenjem zbog toga što je baziran na *Python*-u. ASP.NET nudi visoku skalabilnost s naglaskom na sigurnost i performanse, te zbog svoje povezanosti s *Microsoft*-ovim sustavom nudi široki spektar alata za razvoj i izradu kompleksnih web aplikacija. *CakePHP* nudi balans između fleksibilnosti i brzine razvoja na način da omogućava programerima razne generatore kojih ima manje nego u *Rails*-u, te drži naglasak na stabilnost i sigurnost. *Phoenix* je pisan u *Elixir*-u, koji je za razliku od ostalih programskih jezika opisanih u ovom poglavlju funkcijski programski jezik, što ga čini drugačijim i neuobičajenim za programere koji su navikli na objektno orijentirane programske jezike.

Programski okviri nude slične pristupe razvoju web aplikacija, a odabir odgovarajućeg okvira najčešće se svodi na zahtjeve specifične aplikacije i iskustvo tima.

4. PROGRAMSKO RJEŠENJE

Web aplikacija za postavljanje i korištenje kvizova nastaje radi širenja popularnosti kvizova u svrhu educiranja ispitanika i u svrhu socijaliziranja među sudionicima. Većina ostalih postojećih rješenja u sebi sadrže pitanja koja više nisu relevantna u današnje doba, imaju preuzak izbor pitanja čime se nakon nekoliko odigranih kvizova pitanja počnu ponavljati ili nemaju mogućnost biranja tema u okviru kojih su postavljena.

Cilj web aplikacije izrađene u ovom diplomskom radu je otklanjanje nekih od postojećih problema s kojima se ostale slične web aplikacije za upravljanje i korištenje kvizova susreću, a to su:

- Jednostavno i moderno korisničko sučelje: pruža gostima (korisnici koji nisu prijavljeni u aplikaciju), registriranim korisnicima i administratorima jednostavan i intuitivan uvid na teme, kvizove, pitanja i uređivanje istih,
- Upravljanje temama: administratori će imati mogućnost kreiranja, mijenjanja i brisanja tema. Administrator također ima mogućnost dodavanja fotografije temi koja će biti prikazana prilikom izbora tema,
- Upravljanje kvizovima: korisnici koji su se registrirali i prijavili u aplikaciju će moći praviti vlastite kvizove, koje oni ili administrator aplikacije mogu u bilo kojem trenutku obrisati,
- Odgovaranje kao gost: korisnici koji se ne žele registrirati na web aplikaciju i dalje mogu odgovarati na postojeće kvizove, no nemaju mogućnost kreiranja novih kvizova,
- Besmislena pitanja: postojat će administrator koji ima pravo brisanja kvizova i pitanja ako su u krivoj temi. Administrator ima mogućnost uklanjanja pitanja unutar kviza koja nisu pitanja, što sprječava kreiranje besmislenih kvizova i poboljšava korisničko iskustvo,
- Tablica postignuća: svaki kviz ima tablicu postignuća, odnosno tablicu koja prikazuje zadnjih 5 korisnika koji su najbolje riješili kviz. Ukoliko više od 5 korisnika ima jednak broj bodova, prikazuje se 5 najnovijih korisnika koji su riješili kviz.

U odnosu na rješenja koja su dostupna na tržištu i opisana u drugom poglavlju, web-aplikacija ne donosi inovativno programsko rješenje, ali pruža korisnicima mogućnost kreiranja vlastitih kvizova i drugim korisnicima pristupanje kreiranim kvizovima, te osim toga pruža mogućnost rješavanja kvizova i gostima, odnosno, korisnicima koji se ne žele registrirati u aplikaciju.

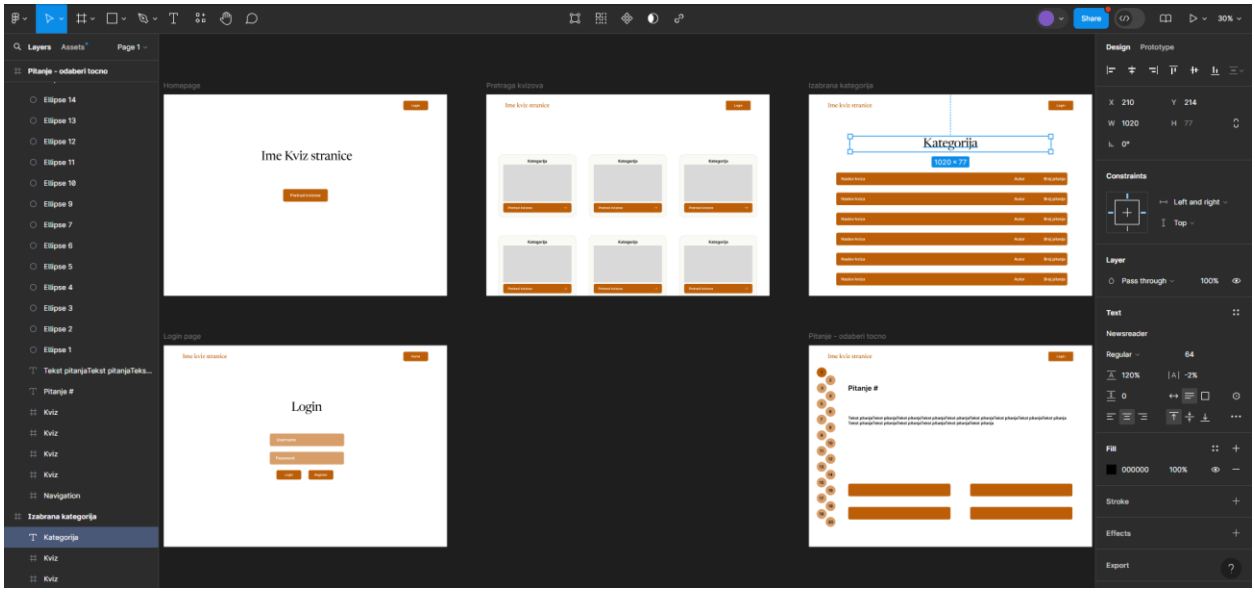
U nastavku ovog poglavlja opisano je planirano rješenje, predstavljeni su rani prikazi koncepta izgleda korisničkog sučelja web-aplikacije, te su opisani dijagrami klasa i baze podataka po kojima je ovaj rad pisan, aplikacija izrađena i zaključena analiza izrađenog rješenja.

4.1. Planirano rješenje

Planiranje web-aplikacije prije početka izrade je potrebno kako bi se postavio jasan cilj kod kreiranja aplikacije i kako bi svi potrebni zahtjevi bili ispunjeni. U planiranju je kreiran okvirni nacrt web aplikacije, te su kreirani dijagrami baze podataka i dijagrami klasa s odgovarajućim metodama i parametrima koji će se koristit u aplikaciji.

4.1.1. Izrada izgleda web aplikacije

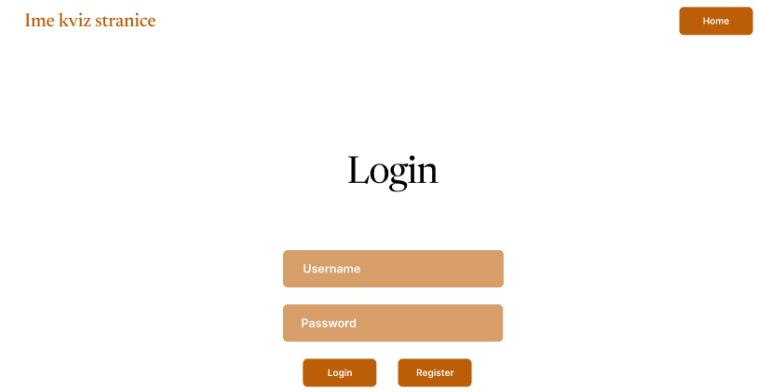
Kako bi se postavila jasna ideja izgleda aplikacije, koristila se web aplikacija *Figma* – alat namijenjen web-dizajnerima kako bi napravili izgled aplikacije [28], bila to web aplikacija, mobilna aplikacija ili bilo koji drugi digitalni proizvod. *Figmu* mogu koristiti više dizajnera u isto vrijeme, radeći zajedno na istom projektu, dizajniranjem iste ili različitih stranica ili elemenata. *Figmu* osim web-dizajnera koriste i *frontend* programeri kako bi implementirali zadano dizajnersko rješenje.



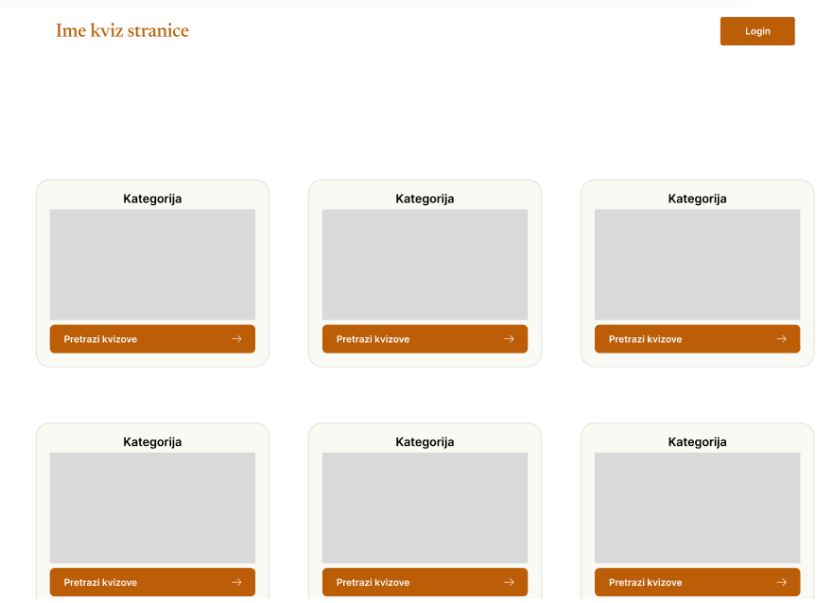
Slika 4.1. Izgled projekta diplomskog rada u alatu *Figma*

Sučelje za izradu nacrta web-aplikacije u *Figmi* se sastoji od tri glavna dijela: padajućeg izbornika svih elemenata na lijevoj strani, glavnog prozora u kojem se vidi trenutni dizajn, kreiraju novi elementi i mijenjaju postojeći jednostavnim načinom *drag-and-drop* i detaljnog opisa označenog elementa na desnoj strani (slika 4.1.).

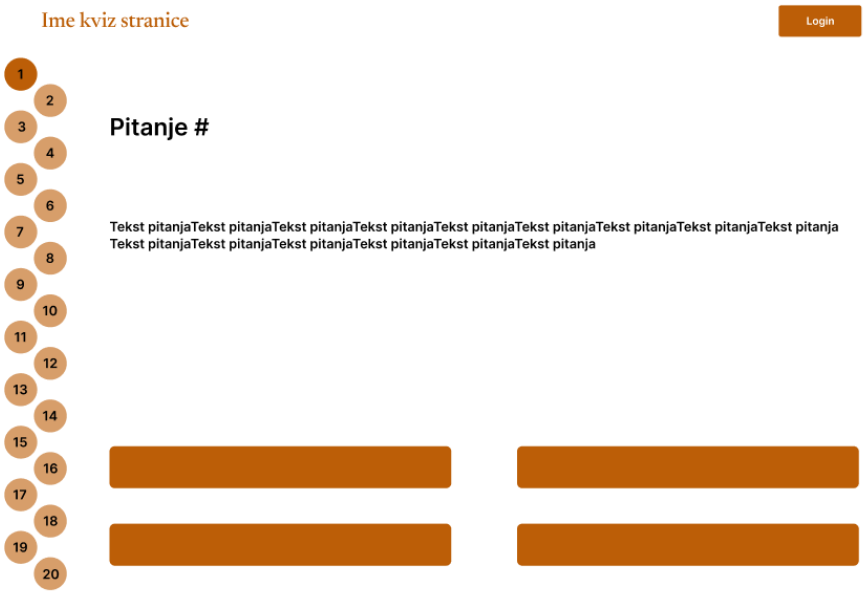
U nastavku su prikazani uvećani prikazi dizajna *Login* stranice, stranice za izbor kategorija kvizova i stranice s pitanjima.



Slika 4.2. Prikaz prototipa dizajna login stranice



Slika 4.3. Prikaz prototipa dizajna pretrage kategorija kvizova



Slika 4.4. Prikaz prototipa dizajna pitanja

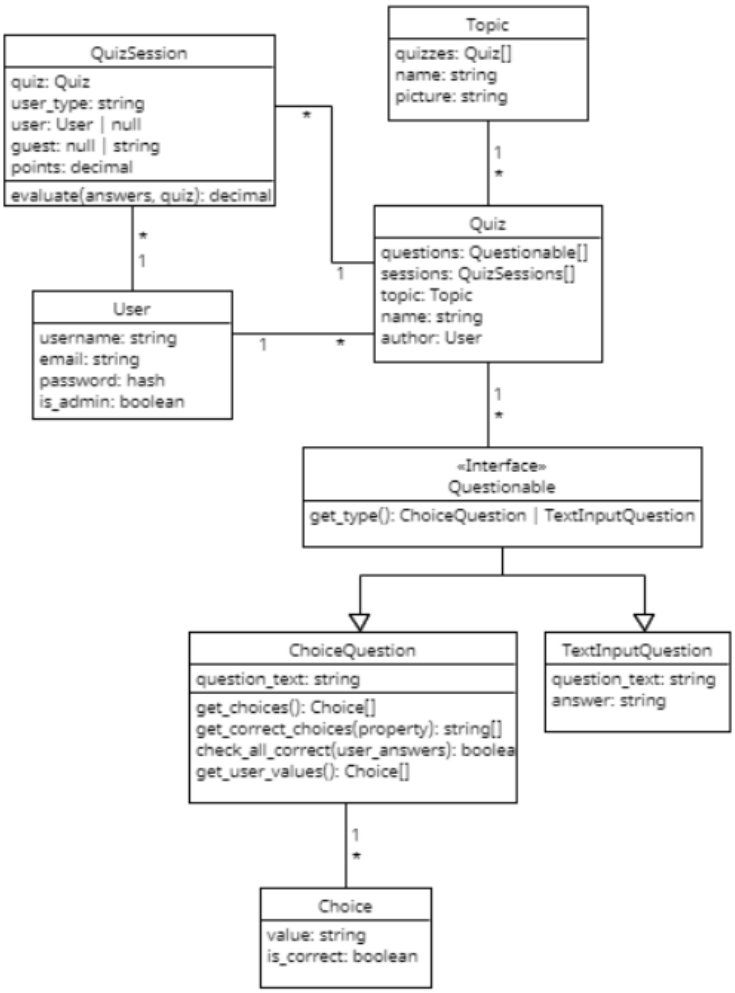
4.1.2. Izrada dijagrama klasa

Za nastavak planiranja, bilo je potrebno odabrati klase, odnosno modele, koji će se koristiti u radu. Modeli se prikazuju s pomoću dijagrama, a za kreiranje dijagrama se koristio program *Visual Studio Code* (u nastavku *VS Code*). *VS Code* je *IDE* (engl. *Integrated development environment*), odnosno tekstualni *editor* koji omogućava programerima olakšano pisanje, pokretanje, testiranje koda i popravljanje grešaka u kodu. Može ga se koristiti za sve programske jezike uz pomoć ekstenzija (engl. *extensions*) koje se preuzimaju u samom programu. Ovaj diplomski rad je pisan koristeći program *VS Code*, no više o tome će biti spomenuto kasnije.

Kako bi mogli kreirati dijagram u programu koji se većinski koristi kao *editor* teksta, bilo je potrebno preuzeti i instalirati nadogradnju *UMLet*. *UMLet* je besplatna ekstenziju koja se koristi za kreiranje *UML* (engl. *Unified Modeling Language*) dijagrama. Za kreiranje novog *UML* dijagrama koristeći *UMLet* potrebno je kreirati novu datoteku s *ekstenzijom* .uxf i otvoriti ju u programu *VS Code.*

UML je standardizirani naziv za jezik modeliranja dijagrama [29] koji predstavlja bitan dio planiranja novih aplikacija i sposobnosti aplikacija (engl. *feature*). Programerima smanjuje potencijalne probleme koji bi nastali ako se odmah počne pisati programski kod, te se u sredini

kreiranja novih sposobnosti aplikacije uvidi problem zbog kojeg je potrebno ostatak programa prilagoditi na nove dodatke.



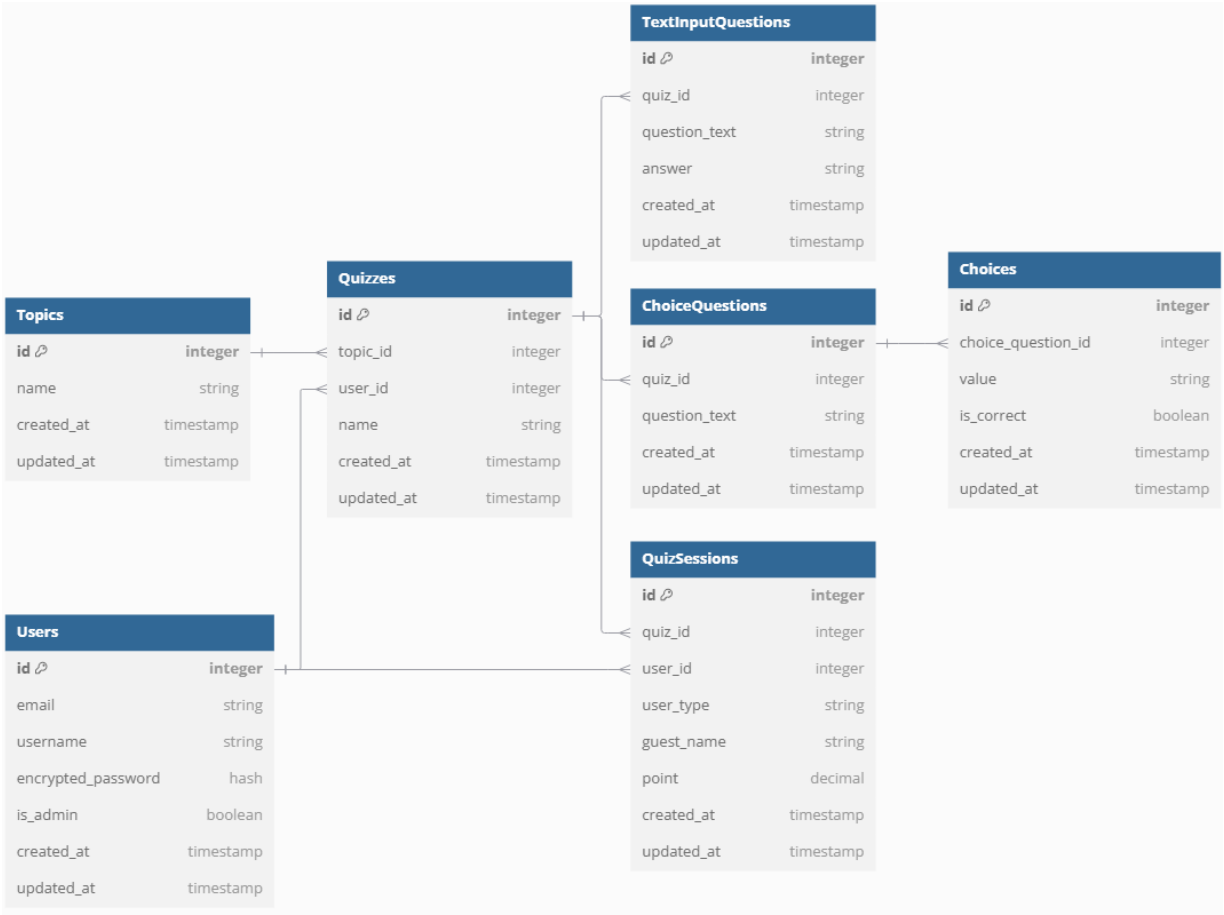
Slika 4.5. Prikaz dijagrama klasa

4.1.3. Izrada dijagrama baze podataka

Za bazu podataka u ovom diplomskom radu bit će korištena *PostgreSQL* baza podataka. *PostgreSQL* je sustav upravljanja relacijskom bazom podataka otvorenog izvornog koda koji je usklađen i proširuje popularan programski jezik za relacijske baze podataka *SQL*. [30] *PostgreSQL* je u razvoju preko 35 godina te je kroz to vrijeme dobio jaki ugled za svoju pouzdanost, performanse i robusnost značajki.

Za izradu dijagrama baze podataka koristi se web aplikacija *Dbdiagram*. Ona je besplatni alat za kreiranje relacijskih dijagrama baze podataka pomoću pisanja koda.

Sučelje za izradu dijagrama web aplikacije *Dbdiagram* sastoji se od 2 glavna dijela: tekstualnog *editora* s lijeve strane u kojem se kreiraju tablice, relacije između njih i podatci koji se spremaju u tablice te glavnog prozora u kojem je grafički prikaz napisanih tablica i njihovih relacija.



Slika 4.6. Prikaz tablica, podataka i relacija korištenih u diplomskom radu

Na slici 4.6. su prikazane tablice, podatci i relacije koje su korištene u ovom diplomskom radu. Svaka tablica ima *created_at* i *updated_at* podatke koji su automatski dodani od strane *Rails*-a pri kreiranju novih tablica, te administratoru prikazuju podatke o vremenu kreiranja i zadnjem vremenu promjene pojedinih unosa u bazi podataka.

Osim navedenih tablica na slici 4.6., u diplomskom radu su korištene i 4 dodatne tablice: *active_storage_attachemnts*, *active_storage_blobs*, *active_storage_variant_records* i *ar_internal_metadata*. One su kreirane i korištene od strane *Ruby Gem*-ova koji su korišteni u izradi ovoga rada.

4.2. Instalacija programskog kostura i postavljanje okruženja

U ovom poglavlju je dan uvid u razvoj same web aplikacije za kreiranje, upravljanje i rješavanje kvizova. U poglavlju je opisana okolina korištena za izradu web aplikacije, *gem*-ovi i sama implementacija i izrada funkcionalnosti rada.

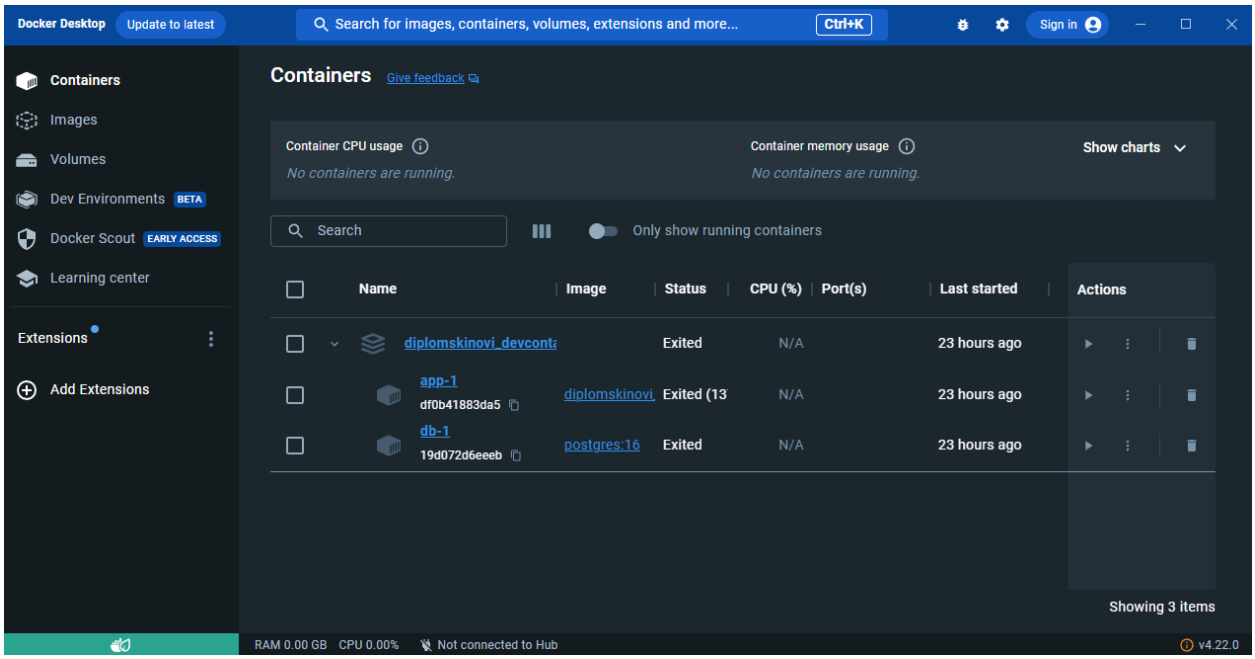
4.2.1. Programsko okruženje

Kako bi se započeo rad na samom programskom dijelu diplomskog rada bilo je potrebno odabrati koje će se okružje koristit. Za ovaj rad je odabran *Development container* pokrenut koristeći *docker container*-e. Prednost *Development* container-a nad radom na vlastitom računalu je izolirano i konstantno okruženje i velika portabilnost. [31] Ukoliko koristimo *VS Code* kao program za uređivanje koda, također je moguće preuzeti nadogradnje koje je moguće koristiti samo u trenutnom *container*-u. Preuzimanje nadogradnji po pojedinom kontejneru omogućava programerima rad s više različitih verzija programskih jezika i smanjuje opterećenje i zatrpanost računala s dodatcima koji nisu potrebni u trenutnom okruženju.

Za konfiguraciju *Development container*-a potrebno je kreirati novi direktorij u kojem će se diplomski rad raditi i u njemu kreirati direktorij pod nazivom „.devcontainer“. [32] Pošto je izabrano korištenje *Docker*-a, nakon toga je potrebno kreirati tri datoteke: *devcontainer.json*, *docker-compose.yml* i *Dockerfile*.

Datoteka *devcontainer.json* je datoteka koja opisuje kako bi *development container* trebao biti pokrenut, odnosno kojim *port*-ovima treba omogućiti vanjsku upotrebu i koje nadogradnje je potrebno instalirati u trenutnom okruženju. U datoteci *docker-compose.yml* su definirani sami *development container*-i koji se koriste u ovom diplomskom radu. Koriste se dva container-a: jedan *container* za samu aplikaciju i jedan poseban *container* na kojem je pokrenuta baza podataka. U datoteci *Dockerfile* se nalaze naredbe i instrukcije za kreiranje samog *image*-a.

Nakon postavljanja datoteka s instrukcijama *Docker*-u je potrebno instalirati *Docker Desktop* aplikaciju koja omogućuje upravljanje *Docker container*-ima na operacijskom sustavu *Windows* i instalirati *VS Code* nadogradnju *Dev Containers*.



Slika 4.7. Prikaz *Docker Desktop* aplikacije

Nakon instalacija je potrebno otvoriti datoteku u kojoj će se diplomski rad raditi u *development containeru* i pokrenuti *Rails* komandu uz dodatne izborne parametre za inicijalizaciju novog *Rails* projekta koja generira samu strukturu datoteka za *Rails* okruženje. Izborni parametri koji su korišteni za kreiranje ovog projekta su:

- css=sass parametar kojim se odabire *CSS* procesor koji se koristi u projektu,
- javascript=esbuild parametar kojim se odabire *Javascript boundler*,
- database=postgresql parametar kojim se odabire baza podataka koja se koristi u projektu.

Nakon generiranja početne strukture datoteka projekta, pošto se koristi *development container* za bazu podataka, potrebno je u datoteku *config/database.yml* dodati red koji govori u kojem se *development containeru* nalazi baza podataka.

4.2.2. Ruby gem-ovi

U ovom poglavlju su detaljnije opisani neki od bitnijih *gem*-ova koji su korišteni prilikom izrade ovog diplomskog rada kao što su *Puma*, *simple_form*, *Turbo-rails* i *Devise*. *Gem*-ove je moguće instalirati u postojeći projekt izravno iz terminala ili dodavanje njihovog naziva u *Gemfile*, te pokretanjem naredbe za instalaciju svih *gem*-ova. Preporučeni način instaliranja je dodavanje naziva u *Gemfile*, kako bi i svi ostali programeri koji rade na projektu koristili iste *gem*-ove s istim verzijama.

Puma

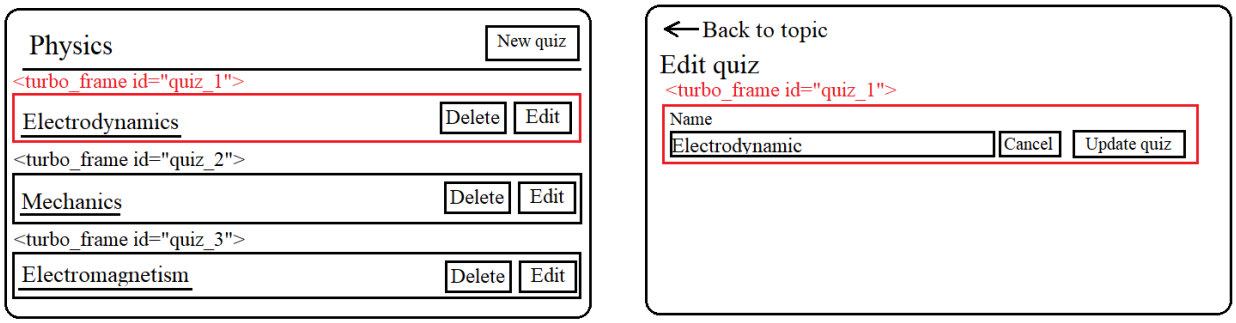
Puma je *gem* koji pruža uslugu brzog i jednostavnog web servera za aplikacije pisane u *Rails*-u. [33] Pruža visoke performanse i optimizaciju za istovremeno procesiranje zahtjeva pomoću paralelizma na razini niti i procesa. *Puma* također pruža podršku za *stream* podataka, odnosno omogućava slanje podataka korisnicima u realnom vremenu.

Simple_form

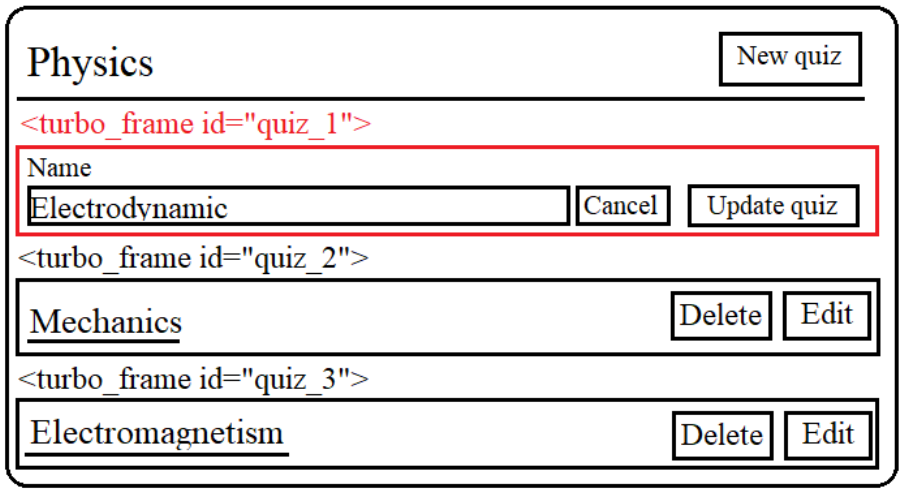
Simple_form je *gem* koji pruža uslugu jednostavnijeg kreiranja i uređivanja *form*-i. [34] *Form*-e su dijelovi web aplikacije koji se koriste kada korisnik dodaje novi ili uređuje postojeći podatak kao što su u ovoj web aplikaciji sami korisnik, tema, pitanje, odgovor i izbor.

Turbo-rails

Turbo-rails (u nastavku teksta će biti korišten skraćeni naziv *Turbo*) je *gem* koji pruža mogućnost kreiranja *single-page* web aplikacija bez ili s jako malo pisanja *JavaScript* koda. [35] S pomoću *turbo-stream*-ova korištenjem web *socket*-a iscrtava novi izgled na web aplikaciji. U ovom diplomskom radu se koristio za izradu svih *form*-i. Većina *form*-i u ovom radu nema puno potrebnih podataka te bi samim time bilo potrebno *render*-ati stranicu u kojoj najčešće postoji samo jedan *text_field* za unos podatka. Osim toga se koristi i za prikazivanju svih podataka iz razloga da prilikom dodavanja, mijenjanja ili brisanja podataka nema potrebe za *reload*-om web stranice. Pomoću *turbo*-a podatci se dodaju, promijene ili uklone iz popisa podataka bez vidljivog prijelaza stanja s korisničke strane. S obzirom na to da stariji web preglednici ne podržavaju *turbo*, prvo je potrebno napraviti da web aplikacija radi i bez *turbo*-a te zatim dodati i omogućiti *turbo*. Na slici 4.8. je prikazan primjer promjene naziva kviza „Electrodynamics“ unutar web aplikacije na web preglednicima koji ne podržavaju *turbo*, dok je na slici 4.9 je prikazan rad web aplikacije na web preglednicima koji podržavaju *turbo*.



Slika 4.8. Prikaz rada web aplikacije na preglednicima bez *turbo*-a



Slika 4.9. Prikaz rada web aplikacije na preglednicima s podržanim *turbo*-om

Devise

Devise je *gem* koji se koristi za autorizaciju na web aplikaciji. Pruža gotova rješenja koja su prijava korisnika, registracija korisnika, promjena zaporke, potvrda *e-mail* adrese i druga. [36] U ovom radu su korištene mogućnosti prijave i registracije korisnika i administratora.

4.3. Implementacija i izrada funkcionalnosti

U ovom poglavlju je dan uvid na način na koji su izrađeni najvažniji dijelovi aplikacije, od korisničkog sučelja, do funkcionalnosti aplikacije. Potpuni izvorni kod i pristup aplikaciji omogućeni su pomoću poveznice u prilogu diplomskog rada.

4.3.1. Izrada i implementacija elemenata korisničkog sučelja

Prototipni grafički dizajn kreiran pomoću web aplikacije *Figma* je opisan u poglavlju 4.1.1. Zbog nemogućnosti izravnog kopiranja elemenata iz *Figme* unutar *Rails*-a, potrebno je bilo svaki element rekonstruirati kroz kod. Rekonstrukcija elemenata je omogućila kreiranje parcijalnih elemenata (engl. *partials*), koje je moguće koristiti na više mjesta unutar web stranice jednostavnim pozivom *Rails* metode „render <partial_name>“. Kao atribute metode *render* je moguće poslati varijable potrebne u parcijalnom elementu pomoću kojih će *Rails* generirati odgovarajući parcijalni element čiji naziv počinje s „_“ i u nastavku je ime tipa varijable.

_topic.html.erb

app > views > topics > _topic.html.erb

```
<%= turbo_frame_tag topic, class:"topic__turbo_frame" do %>
  <div class="topic">
    <%= link_to topic.name,
                topic_path(topic),
                data: { turbo_frame: "_top" },
                class: "topic__center" %>

    <%= link_to image_tag(topic.picture, class: "topic__image"), topic_path(topic), data: { turbo_frame: "_top" } %>

    <% if current_user != nil %>
      <% if current_user.is_admin == 1 %>
        <div class="topic__actions">
          <%= button_to "Delete",
                        topic_path(topic),
                        method: :delete,
                        class: "btn btn--light" %>
          <%= link_to "Edit",
                      edit_topic_path(topic),
                      class: "btn btn--light" %>
        </div>
      <% end %>
    <% end %>
  </div>
<% end %>
```

Slika 4.10. Prikaz koda koji će se iscrtati prilikom poziva metode *render* s tipom varijable *Topic*

4.3.2. Stvaranje i prikaz tema, kvizova

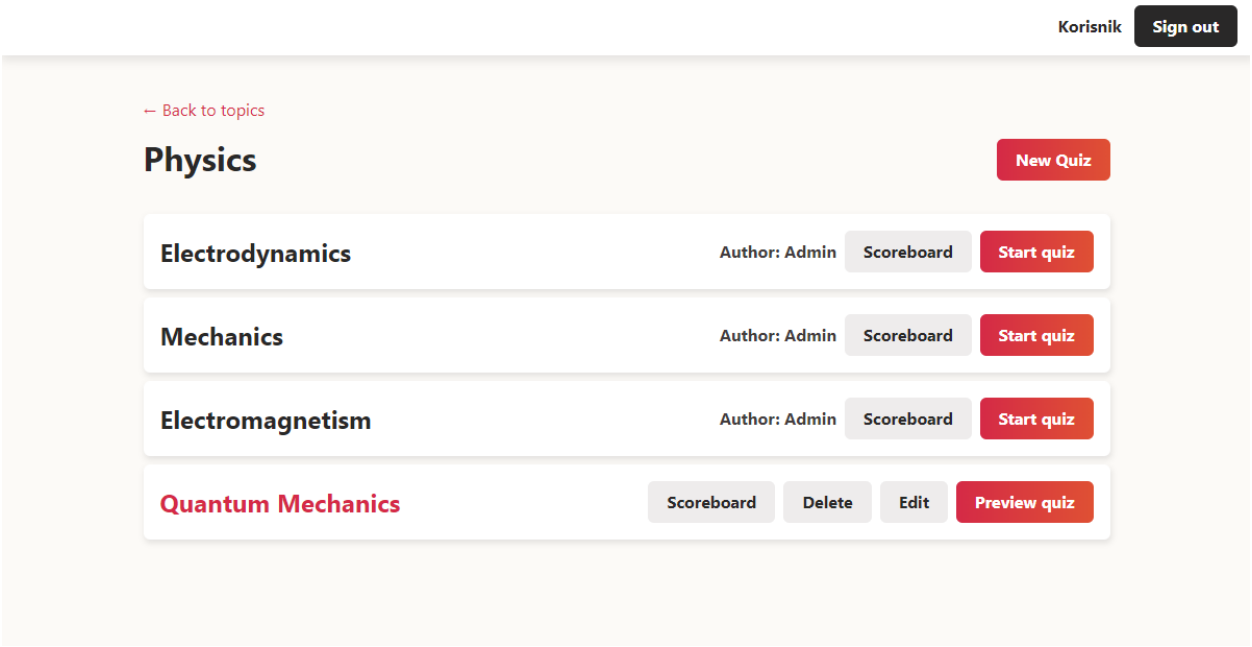
Rails koristi kontrolere za izvođenje logike koja se odvija iza korisničke interakcije s web aplikacijom kao i svi ostali programski okviri koji koriste *MVC* arhitekturu opisanu u poglavlju 4.1. Unutar kontrolera su definirane metode koje se pozivaju interakcijom s određenim gumbovima. Prilikom kreiranja novih dijelova kviza, bila to tema, kviz, pitanje ili izbor na pitanje, poziva se metoda *new* unutar odgovarajućeg kontrolera. Metoda *new* kreira novi podatak sa odgovarajućim parametrima koje primi iz web preglednika te ga pokušava spremiti u bazu podataka. Ako ga uspješno spremi u bazu podataka, provjerava tip odgovora te ako je tip odgovora „HTML“ (u slučaju da web preglednik ne podržava *turbo stream*), preusmjeravaju korisnika na putanju zadanu unutar metode. Ako je tip odgovora „turbo-stream“, metode dodaju, uređuju ili uklanjaju podatak iz web preglednika sukladno metodi. Osim metoda za brisanje, uređivanje i dodavanje novih podataka, kontroleri imaju i metodu za prikaz podataka. U njoj se prikupljaju podaci iz baze podataka koji su potrebni za prikaz na web pregledniku. Kontroleri korišteni u ovom diplomskom radu se nalaze na zadanoj lokaciji svih *Rails* kontrolera: app/controllers.

4.3.3. Provjera korisničkog identiteta

Provjera korisničkog identiteta se odvija korištenjem *gem*-a *Devise*. *Devise* pruža uslugu generiranja parcijalne forme za registraciju i prijavljivanje korisnika i generiranje samog modela korisnika. Model generiran od strane *Devise*-a je nadograđen u ovom diplomskom radu u svrhu dodavanja korisničkog imena svakom korisniku. *Devise* pruža nekoliko *helper* metoda koje se koriste u ovom radu, kao što su: „user_signed_in?“ koja provjerava je li korisnik prijavljen u aplikaciju i vraća vrijednost *true* ili *false*, te „current_user“ koja vraća objekt trenutno prijavljenog korisnika. Navedene *helper* metode se koriste kako bi se korisniku iscrtali odgovarajući dijelovi web aplikacije ovisno o njegovom statusu: gost, registrirani korisnik, autor kviza i administrator.

4.3.4. Implementacija uloga

Gost ima mogućnost odgovaranja na kvizove. Registrirani korisnik ima mogućnost odgovaranja na kvizove i kreiranja novih kvizova. Autor kviza ima mogućnost odgovaranja na kvizove, pregled, uređivanje i brisanje svojih kvizova i kreiranje novih kvizova. Administrator ima mogućnost pregleda, uređivanja i brisanja svih kvizova i mogućnost kreiranja novih tema. Na slici 4.11. je prikazan pregled teme Physics od strane korisnika koji je autora kviza „Quantum Mechanics“.



Slika 4.11. Pregled teme „Physics“ od strane prijavljenog korisnika „Korisnik“

4.3.5. Stvaranje i prikaz pitanja

Na web aplikaciji postoje dvije vrste pitanja: pitanja na koja korisnik unosi točan odgovor i pitanja na kojima korisnik bira točne odgovore. Pitanja korisnik kreira nakon što je odabrao kviz za koji želi kreirati pitanja. Kvizovi nemaju gornju granicu broja pitanja, dok se kvizovi koji nemaju postavljena pitanja ne mogu pokrenuti.

4.3.6. Evaluacija kviza

Korisnik može odgovarati na kviz nakon što je kviz kreiran i ima barem jedno pitanje u sebi. Ukoliko je korisnik autor kviza ili administrator, on nema opciju odgovaranja na sami kviz iz razloga što zna odgovore na zadana pitanja ili ima uvid u njih. Administrator i autor kviza imaju opciju „Preview quiz“ koja je jednaka kao i samo odgovaranje na kviz, osim što se postignuti rezultat ne sprema i ne utječe na stanje tablice postignuća. Ostali registrirani korisnici i gosti imaju opciju „Start quiz“ koja ih vodi na novi prozor u kojem odgovaraju na pitanja iz kviza. Svako pitanje je postavljeno tako da nosi jedan bod. Ukoliko u kvizu postoje pitanja u kojima se treba odabrati točan odgovor, a nije postavljen niti jedan izbor, pitanje nije iscrtano korisniku niti se računa za maksimalan broj bodova. Pitanja na izbor točnog odgovora ne moraju imati ponuđen točan odgovor, te samim time korisnik dobiva jedan bod ako ostavi to pitanje prazno. Jedan bod se dijeli na broj točnih odgovora ukoliko pitanja na izbor točnog odgovora imaju više točnih odgovora. Korisnik dobiva nula bodova ukoliko na takvom pitanju označi jednak broj točnih i netočnih odgovora ili ako označi više netočnih odgovora. Korisnik ne mora paziti na velika i mala slova kod pitanja u kojima unosi točan odgovor jer se prilikom bodovanja ona ne gledaju. Ako korisnik nije prijavljen, odnosno ako je gost, prije predaje kviza je potrebno i unijeti ime s kojim će biti prikazan u tablici postignuća ukoliko bude imao dovoljno dobar rezultat.

4.3.7. Tablica postignuća

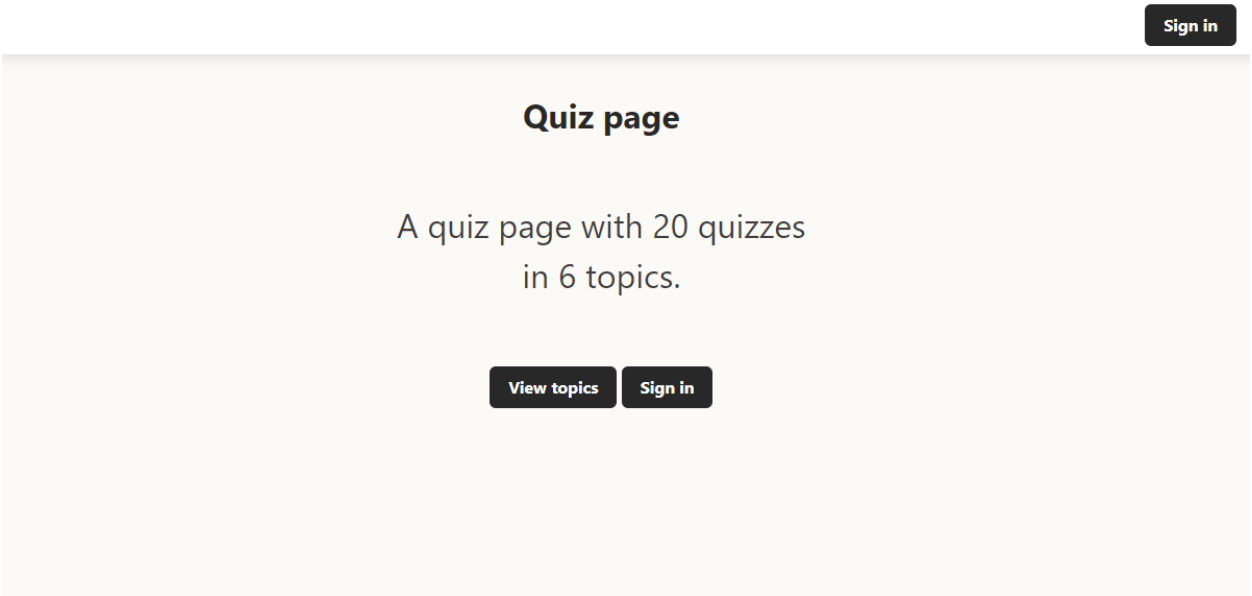
Do tablice postignuća je moguće doći pritiskom na gumb „Scoreboard“ koji je vidljiv na stranici za odabir kvizova (slika 4.1.). Tablica postignuća prikazuje pet najboljih korisnika, odnosno gostiju koji su riješili kviz i njihove bodove. Prednost na tablici postignuća imaju korisnici koji su kasnije rješavali kviz ukoliko više korisnika, odnosno gostiju ima jednak broj bodova. Na tablici postignuća je također moguće razaznati je li prikazano ime pripada korisniku ili gostu na način da svaki gost koji je prikazan ima prefiks „Guest“.

4.4. Analiza programskog rješenja

U ovom poglavlju je dan uvid na korisnički doživljaj koji predstavlja krajnje vizualno rješenje aplikacije kroz scenarij općenitog korisnika. Nakon opisanog korisničkog doživljaja su dana dva dijagrama tijeka za korištenje web aplikacije od strane gosta i prijavljenog korisnika.

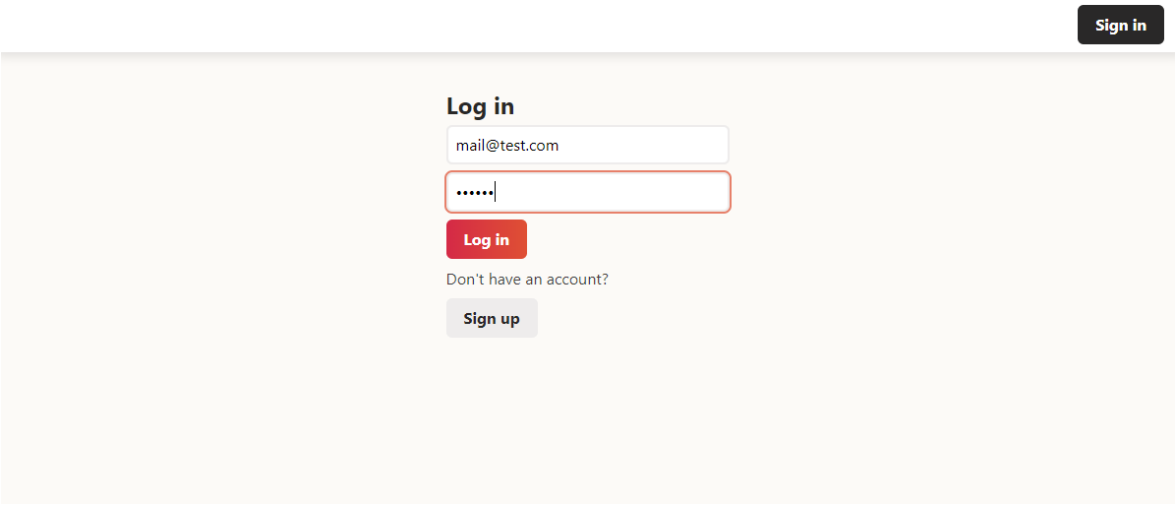
4.4.1. Korisnički doživljaj

Korisnički doživljaj predstavlja krajnji grafički dizajn web aplikacije u koju krajnji korisnik ima uvid. Na slikama u nastavku (Slike 4.12. – 4.28.) je dan uvid u sve zaslone koje web aplikacija za izradu i rješavanje kvizova posjeduje iz perspektive logiranog korisnika. Ukoliko se zasloni mijenjaju ako je korisnik gost ili administrator, dodan je kratki opis na pojedinosti koje su dodane, odnosno uklonjene. Zasloni nisu identični prototipnim zaslonima iz poglavlja 4.1.1. iz razloga mijenjanja aplikacije kroz fazu izrade u kojoj su uočeni vizualni i tehnički nedostatci prototipnih zaslona. U slikama u nastavku je prikazan imaginarni scenario prijavljivanja korisnika i kreiranja kviza unutar teme „Physics“ pod nazivom „Thermodynamics“ s dva pitanja, jednim pitanjem za unos teksta i jednim pitanjem na izbor točnog odgovora te prikazom probnog rješavanja kviza.



Slika 4.12. Prikaz početne stranice web aplikacije

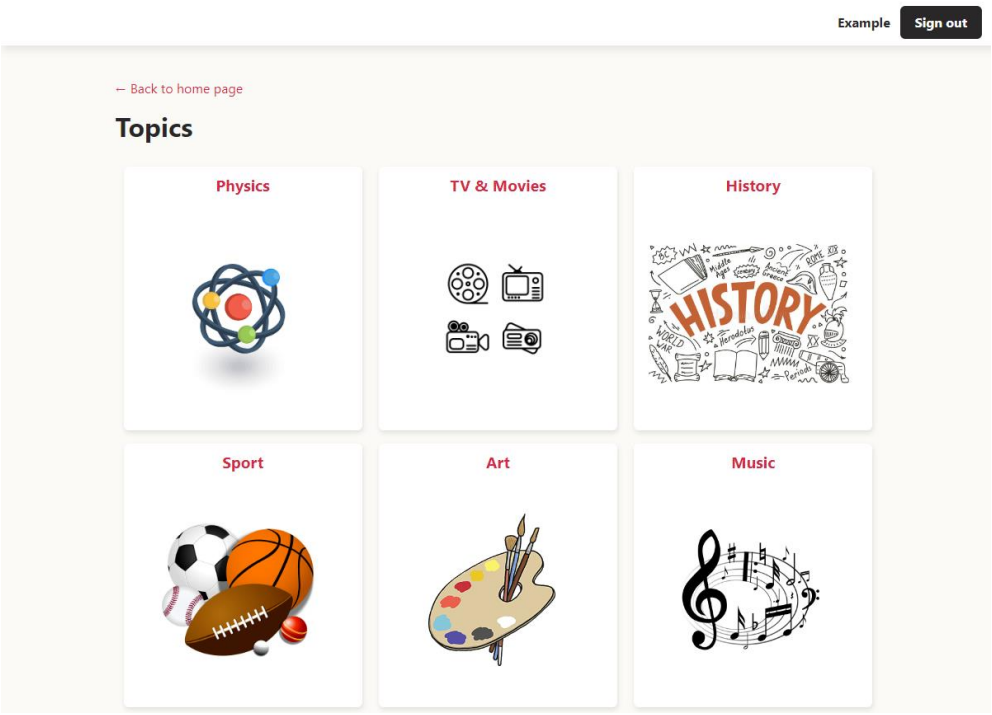
Na slici 4.12. gostujući korisnik može odmah otići na popis tema ove web aplikacije, dok se administrator i korisnik trebaju prijaviti, odnosno registrirati.



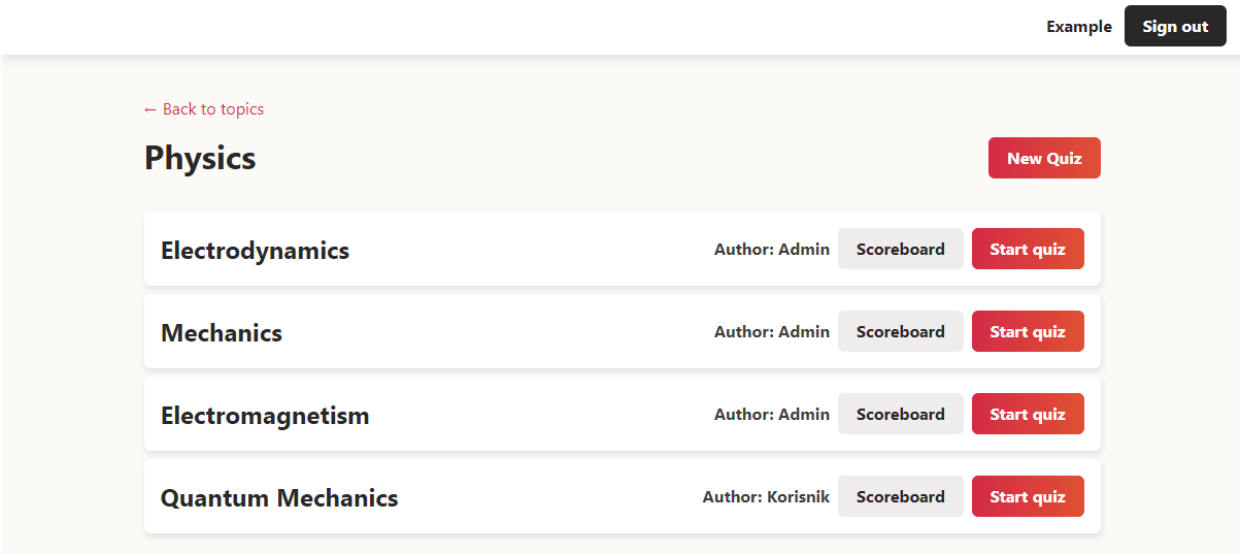
Slika 4.13. Prikaz login stranice web aplikacije

Ukoliko korisnik ne posjeduje račun u ovoj web aplikaciji, odabire gumb „Sign up“, koji ga vodi na stranicu sličnu slici 4.13., s razlikom da je potrebno upisati korisničko ime koje će pisati kod njegovih ostvarenih rezultata i pored kvizova koje je korisnik kreirao.

Ukoliko je korisnik administrator, na slici 4.14. su mu vidljivi dodatni gumbovi koji omogućuju kreiranja novih tema, brisanja i mijenjanja postojećih. Gumb za kreiranje nove teme se nalazi na vrhu stranice, dok svaka tema sadrži gumbove za brisanje i mijenjanje ispod slike teme.

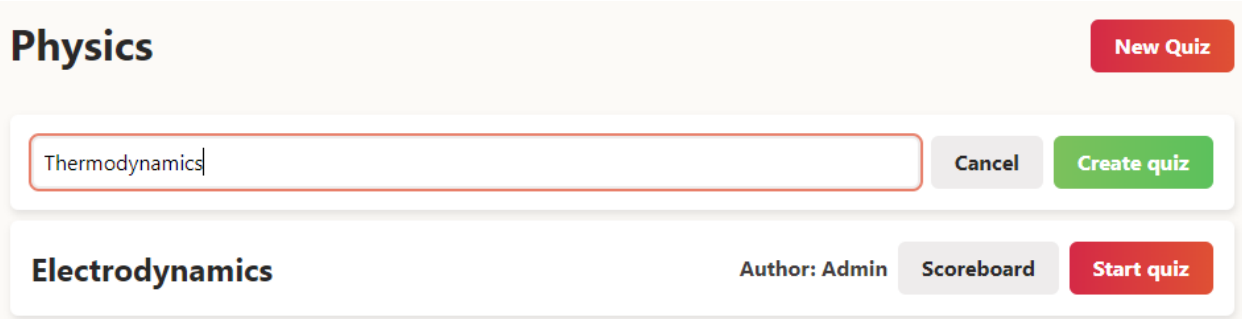


Slika 4.14. Prikaz stranice sa izborom tema

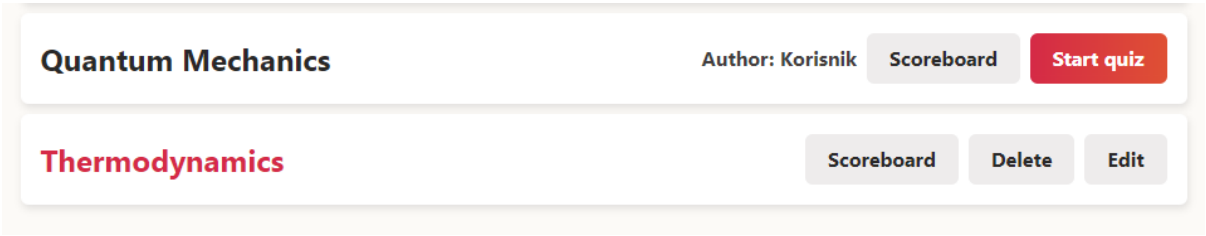


Slika 4.15. Prikaz stranice pod temom Physics

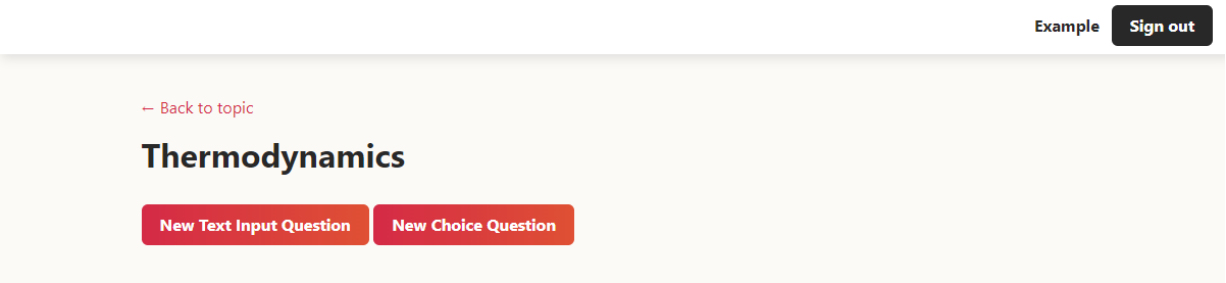
Ukoliko korisnik nije prijavljen, na slici 4.15. neće vidjeti gumb „New Quiz“, te je time onemogućeno kreiranje novih kvizova od strane gostiju. Pritiskom na gumb „New Quiz“ kreira se *turbo frame* koji omogućuje korisniku upisivanje teme novog kviza. Mogućnost promjene naziva kviza izgleda jednako kao i mogućnost kreiranja novog kviza, samo se ne kreira novi *turbo frame*, nego se promjeni *turbo frame* na kojega je korisnik označio (slike 4.8. i 4.9.). Administrator ima mogućnost upravljanja svim kvizovima kao što je prikazano na slici 4.17. za kviz Thermodynamics.



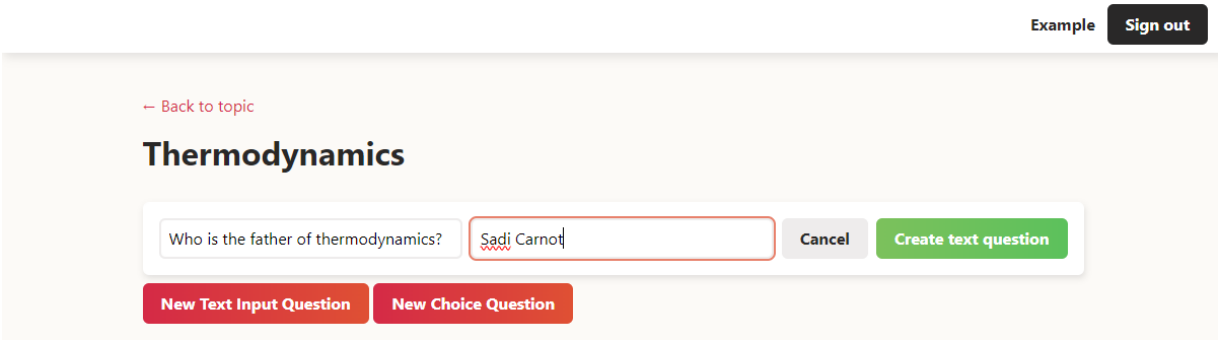
Slika 4.16. Kreiranje novog kviza pod nazivom Thermodynamics



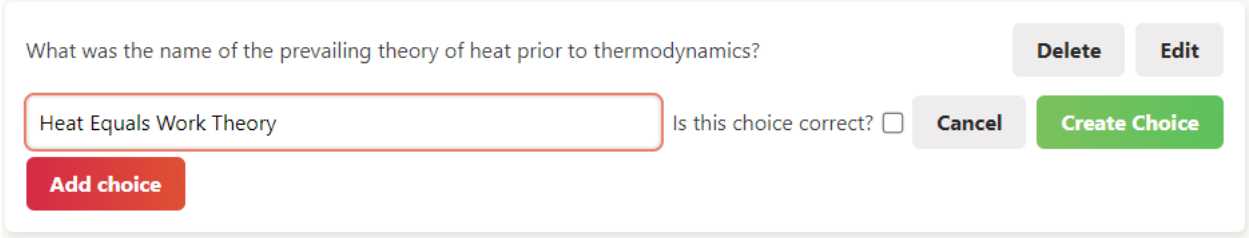
Slika 4.17. Kviz pod nazivom Thermodynamics je kreiran i dodan na kraj popisa kvizova



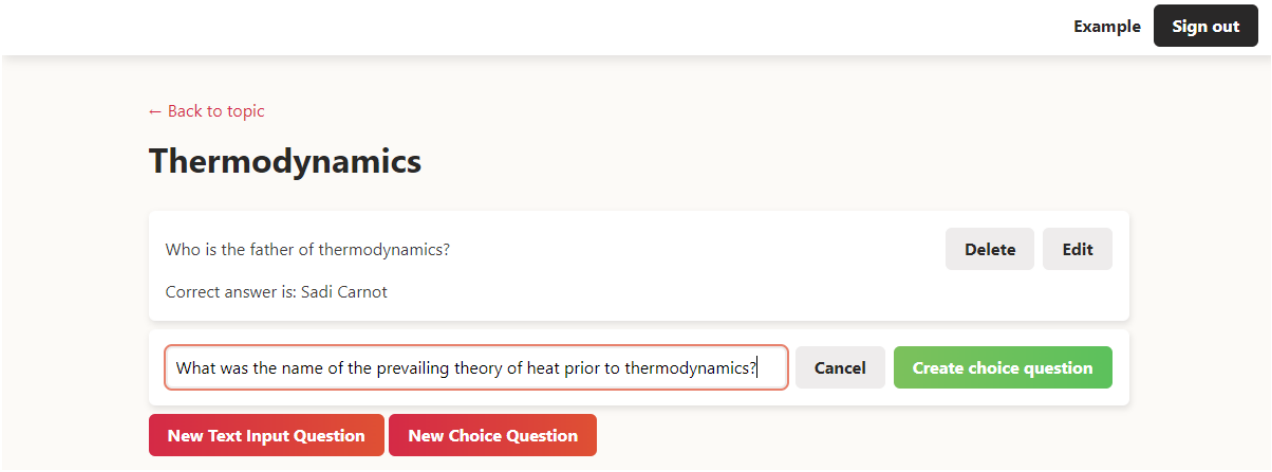
Slika 4.18. Prikaz prazne kviz stranice nakon ulaska u kviz



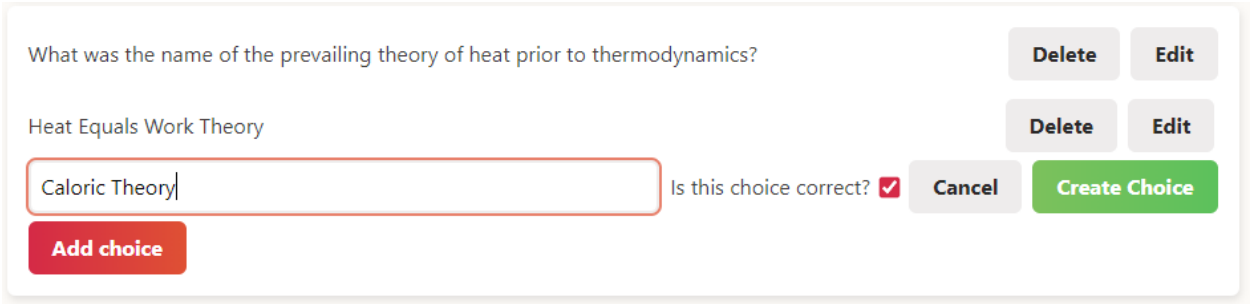
Slika 4.19. Kreiranje novog pitanja za unos teksta



Slika 4.20. Kreiranje netočnog odgovora



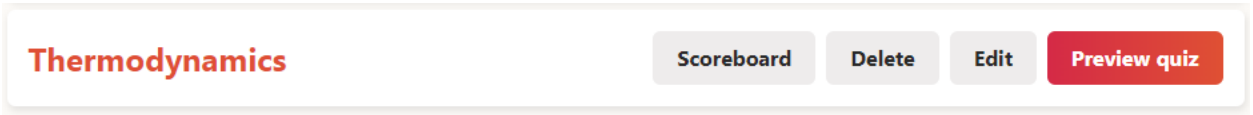
Slika 4.21. Kreiranje novog pitanja s točnim odgovorima



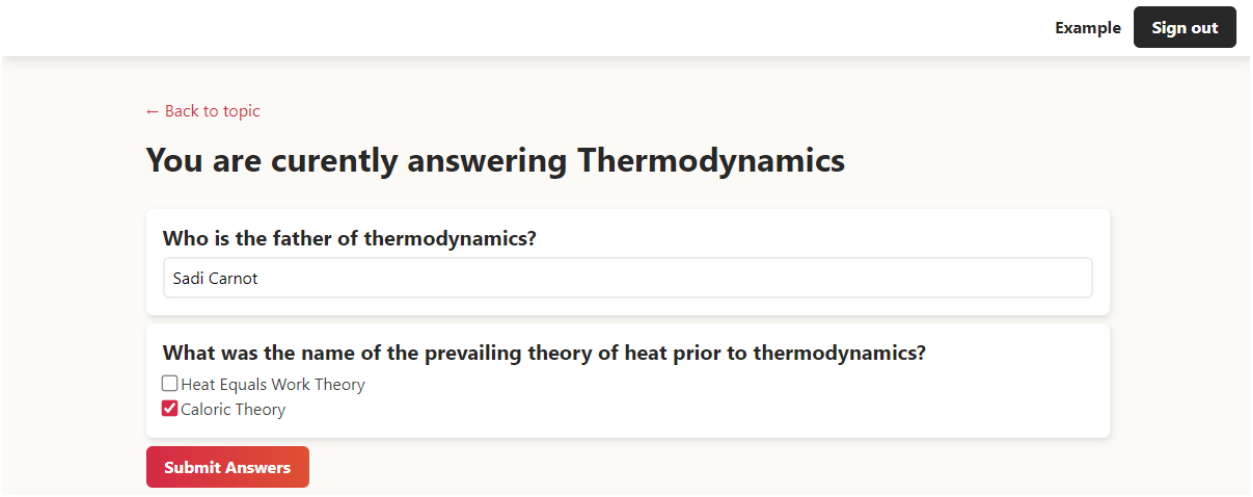
Slika 4.22. Kreiranje točnog odgovora

Promjene pojedinih pitanja, odnosno ponuđenih odgovora izgledaju jednako kao i kreiranje novih pitanja, odnosno odgovora, uz razliku da se u gumbovima ne nalazi ključna riječ „Create“ nego „Update“.

Nakon dovršenog kreiranja pitanja, korisnik pritišće gumb „Back to topic“ koji se nalazi iznad naslova kviza, vidljiv na slici 4.19., te odabire novi gumb Preview Quiz.



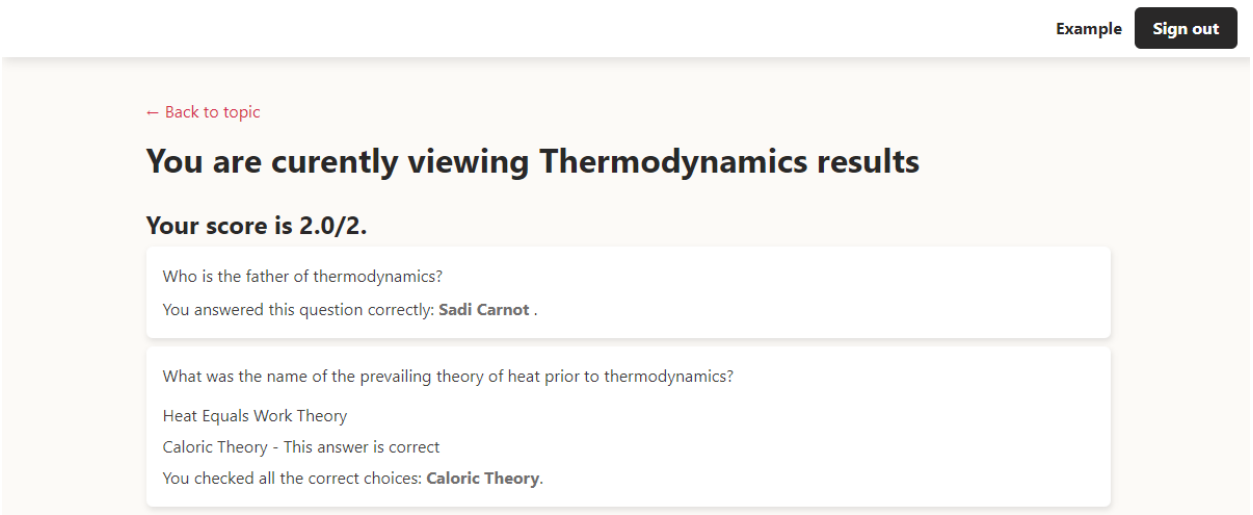
Slika 4.23. Prikazuje se gumb Preview Quiz



Slika 4.24. Prikaz rješavanja kviza

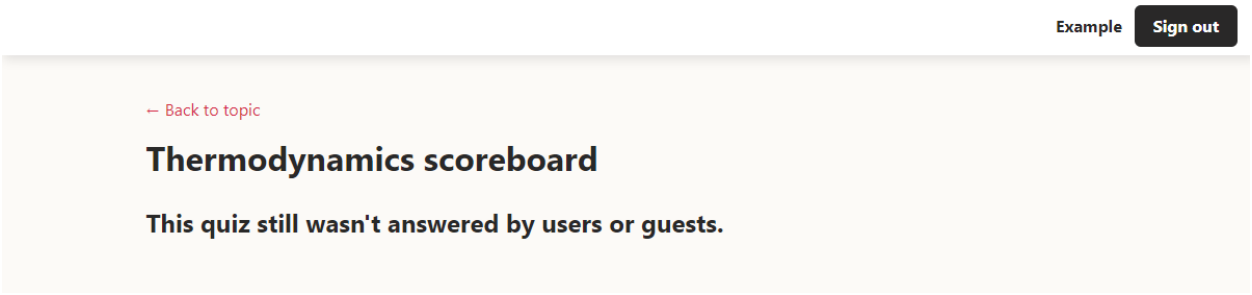
Nakon što je korisnik odabire gumb „Submit Answers“ nakon što je odgovorio na pitanja koja je znao, a on ga vodi na stranicu koja prikazuje prikupljene bodove i točna rješenja. Pošto je trenutni korisnik vlasnik kviza, njegovo rješavanje se ne sprema u bazu podataka i ne utječe na tablicu postignuća.

Ukoliko gost, odnosno korisnik koji nije prijavljen rješava kviz, njemu se kreira još jedno dodatno polje za unos imena koje će se nalaziti na tablici postignuća ako je dovoljno dobro riješio kviz. Kviz nije moguće predati dokle god se polje s nazivom gosta ne popuni.



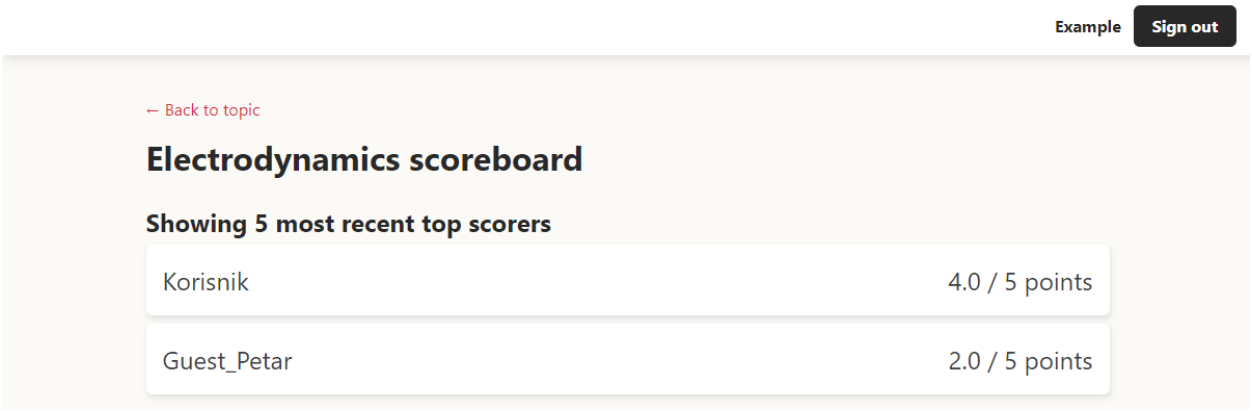
Slika 4.25. Prikaz evaluacije i točnih rješenja

Nakon što je korisnik provjerio odgovore na svoja pitanja i vidio koliko je bodova uspio prikupiti na kvizu, odabire gumb „Back to topic“ koji se nalazi na vrhu stranice. Pošto ovaj kviz trenutno nema korisnika koji su ga rješavali, prikaz tablice postignuća će biti prazan.



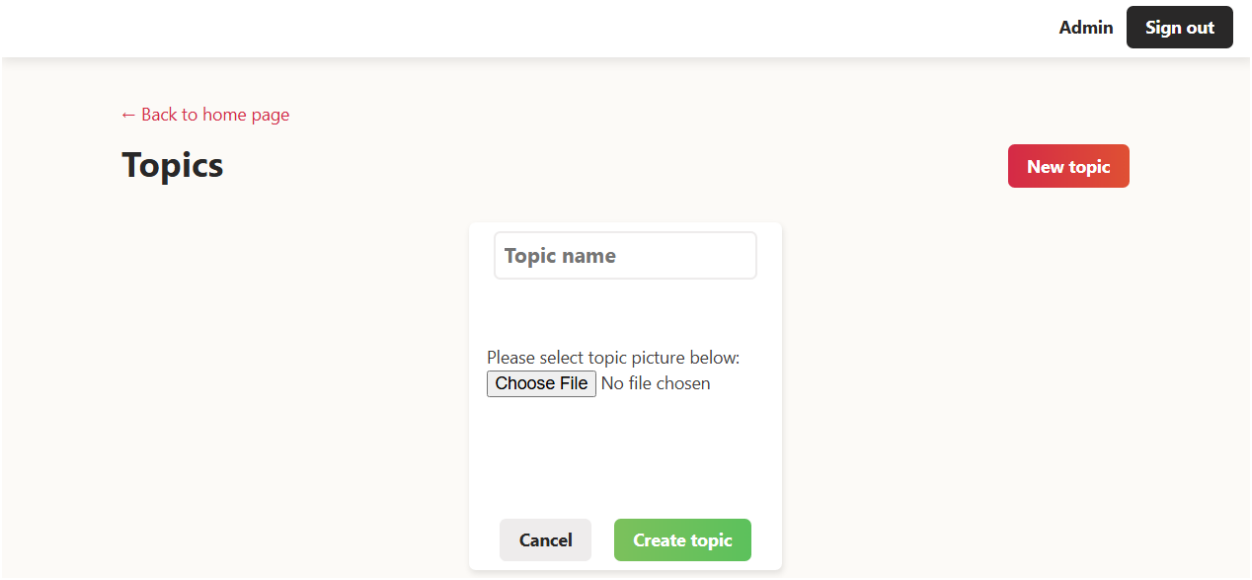
Slika 4.26. Prikaz prazne tablice postignuća

Nakon pregleda tablice postignuća, korisnik može otići nazad na popis kvizova unutar teme pritiskom na gumb „Back to topic“ na vrhu stranice. Na slici 4.27. je prikazana tablica postignuća kviza Electrodynamics, kojega je riješio jedan korisnik s nazivom „Korisnik“ i jedan gost s nazivom „Petar“, čiji je naziv prikazan kao „Guest_Petar“.



Slika 4.27. Prikaz tablice postignuća za kviz Electrodynamics

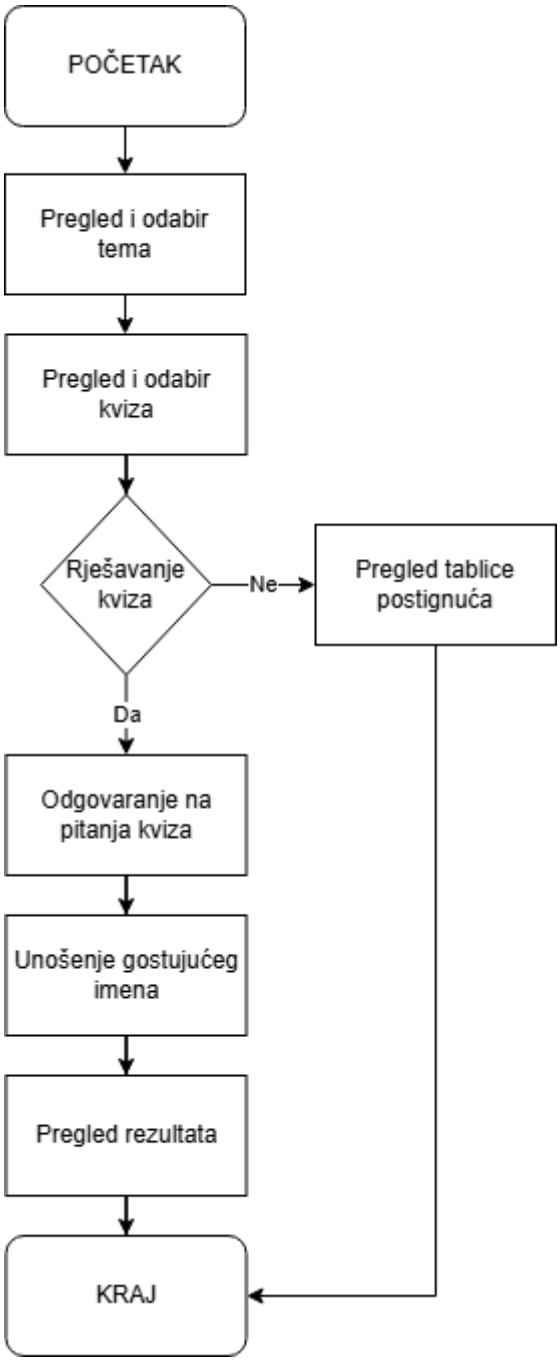
Za kraj poglavlja je prikazan izgled dodavanja nove teme od strane administratora.



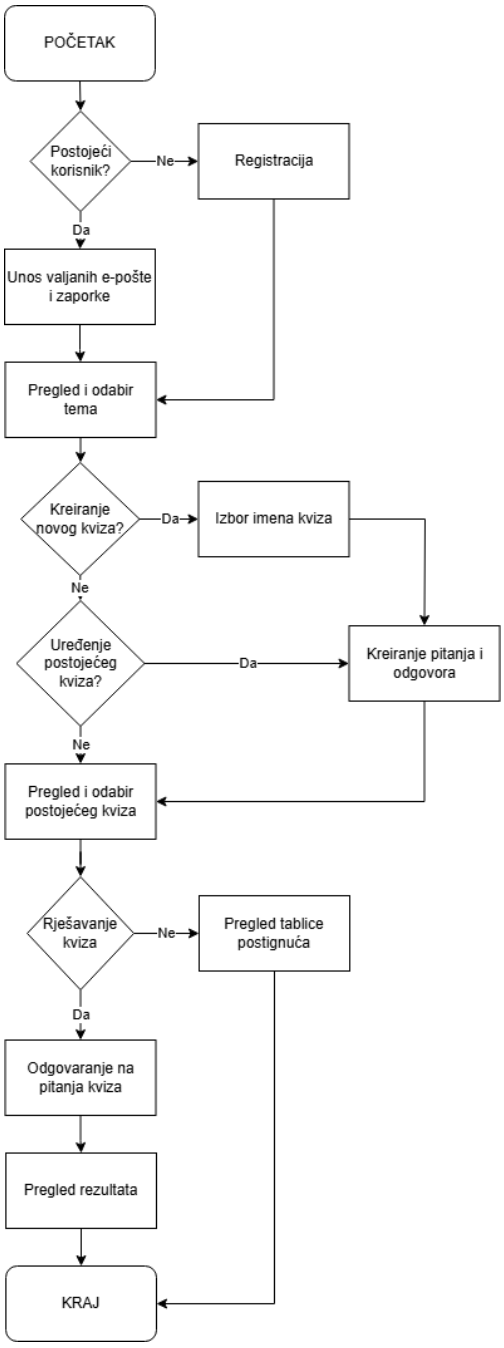
Slika 4.28. Prikaz forme za dodavanje nove teme od strane administratora

4.4.2. Dijagrami tijeka korištenja aplikacije od strana raznih korisnika

U nastavku poglavlja bit će prikazani dijagrami tijeka korištenja aplikacije od strane gostujućeg korisnika i prijavljenog korisnika.



Slika 4.29. Prikaz gostujućeg dijagrama tijeka



Slika 4.3. Prikaz korisničkog dijagrama tijeka

5. ZAKLJUČAK

Današnje društvo teži prema informacijama, te samim time i postavljanju mnogih pitanja sa željom proširivanja vlastitog znanja pojedinca. Jedna od vrsta širenja znanja kroz postavljanje, odgovaranje ili slušanje su kvizovi. Oni postaju sve češći, bili oni kvizovi koji se emitiraju na televizijskim programima, kvizovi koji se rješavaju u edukacijskim ustanovama ili pub kvizovi. Sami kvizaši, slušatelji i gledatelji sudjeluju u kvizovima, neki aktivnije, neki pasivnije, te samim time proširuju vlastito znanje.

Web aplikacija za postavljanje, nadgledanje i rješavanje kvizova razvijena za potrebe ovog diplomskog rada ne daje inovativno rješenje, već pruža mogućnost da svaki korisnik koji je voljan registrirati se na web aplikaciju može postati kreator kviza. Osim toga, korisnici imaju i mogućnost rješavanja kvizova ostalih korisnika, te sami uvid na svoj kviz. Korisnici koji ne žele unijeti svoje osobne podatke poput e-mail adrese u web aplikaciju nisu nužni, no skraćena im je mogućnost kreiranja kvizova. Web aplikacija pruža i blago natjecateljsko okruženje u smislu tablice postignuća za svaki kviz, koja u sebi prikazuje zadnjih pet korisnika s najboljim rezultatima. Osim svega toga, aplikacija pruža mogućnost nadgledanja administratorskim korisnicima, te im daje punu ovlast nad svim temama, kvizovima i samim pitanjima, što im omogućava brisanje i mijenjanja svih komponenti web aplikacije.

Ova web aplikacija popunjava sve zahtjeve zadane u samom opisu diplomskog rada, te pruža i neke dodatne funkcionalnosti, kao što su tablica postignuća i korisnik administrator. Web aplikaciju je moguće proširiti s daljnjim razvojem na razne načine kao što su: dodavanjem novih vrsta pitanja, mogućnost mijenjanja jezika na kojem se stranica prikazuje, mogućnost filtriranja kvizova po jeziku u kojem su napisani i dodavanje statistike svakog korisnika.

Osim same izrade web aplikacije, u radu je uspoređeno nekoliko programskih okruženja s programskim okruženjem *Ruby on Rails,* te su istaknute neke prednosti i mane pojedinih programskih okruženja naspram *Ruby on Rails* programskog okruženja.

LITERATURA

[1] Kahoot! learning experience: supporting students' entire learning cycle from motivation to mastery, [online], dostupno na https://kahoot.com/blog/2024/06/21/the-kahoot-learning-experience-supporting-students-entire-learning-cycle-from-motivation-to-mastery/ [15.11.2024]

[2] Kahoot! Review: Make and play quiz-style games for work, school, and fun, [online], dostupno na https://www.pcmag.com/reviews/kahoot [15.11.2024]

[3] Socrative: Higher Ed, [online], dostupno na https://www.socrative.com/higher-ed/ [15.11.2024]

[4] Socrative: Real-Time Assessment, Instant Insights, [online], dostupno na https://www.socrative.com/ [15.11.2024]

[5] Google Forms: Online Form Creator, [online], dostupno na https://www.google.com/forms/about/#features [16.11.2024]

[6] How to use linear scale questions in google forms, [online], dostupno na https://www.jotform.com/google-forms/google-forms-linear-scale/ [16.11.2024]

[7] Google Forms Reviews, Pricing and Features 2024, [online], dostupno na https://www3.technologyevaluation.com/solutions/54164/google-forms [16.11.2024]

[8] Typeform: People-Friendly Forms and Surveys, [online], dostupno na https://www.typeform.com/product-overview/ [16.11.2024]

[9] Unveiling 17 Typeform Examples: A Deep Dive into Interactive Surveys and Forms, [online], dostupno na https://clickydrip.com/typeform-examples/ [16.11.2024]

[10] Quizizz, [online], dostupno na https://quizizz.com/home/en/product?lng=en#Instructions [16.11.2024]

[11] Quizizz: Question Types Explained, [online], dostupno na https://support.quizizz.com/hc/en-us/articles/4409852287513-Question-Types-Explained [16.11.2024]

[12] R.A.Santelices, M. Nussbaum:A framework for the development of videogames, Software – Pracitce and Experience, vol. 31, No. 11, str. (1091-1107)

[13] Model-View-Controller MVC, [online], dostupno na https://developer.mozilla.org/en-US/docs/Glossary/MVC [15.6.2024]

[14] M. J. M. Razi, Interaction within MVC pattern, [online], dostupno na https://www.researchgate.net/figure/Interaction-within-MVC-pattern-The-Model-component-correlates-with-all-the-data-related_fig2_328716094 [15.6.2024]

[15] MVC, [online], dostupno na https://www.codecademy.com/article/mvc [27.6.2024]

[16] Django, [online], dostupno na https://www.djangoproject.com/ [19.9.2024]

[17] MCT Structure in Django, [online], dostupno na https://www.educative.io/answers/what-is-mvt-structure-in-django [19.9.2024]

[18] S. T. Arasteh, T. Han, M. Lotfinia: Large language models streamline automated machine learning for clinical studies. Nat Commun 15, 1603 (2024)

[19] Whats the best programming language for machine learning, [online], dostupno na https://www.techtarget.com/searchenterpriseai/tip/Whats-the-best-programming-language-for-machine-learning [12.1.2025]

[20] J. S. Linowes,"Evaluating web development frameworks: Rails and django." Parkerhill Technology Group LLC. February (2007).

[21] ASP.NET Web applications, [online], dostupno na https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps [19.9.2024]

[22] ASP.NET Blazor, [online], dostupno na https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor [21.9.2024]

[23] A. Lock: ASP. NET core in Action. Simon and Schuster, 2023. str. (466 – 485)

[24] CakePHP, [online], dostupno na https://cakephp.org/ [19.9.2024]

[25] D. Golding: Beginning CakePHP: from novice to professional. Apress, str. 2-5 (2008)

[26] Phoenix framework, [online], dostupno na https://www.phoenixframework.org/ [21.9.2024]

[27] ElixirLS, [online], dostupno na https://github.com/elixir-lsp/elixir-ls [12.1.2025]

[28] What is Figma, [online], dostupno na https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma [27.6.2024]

[29] What is UML?, [online], dostupno na https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/ [27.6.2024]

[30] PostgreSQL, [online], dostupno na https://www.postgresql.org/ [19.9.2024]

[31] Advantages of DevContainers for Modern App Development, [online], https://www.linkedin.com/pulse/advantages-devcontainers-modern-app-development-juan-pablo-botero-vfjde [27.6.2024]

[32] Introduction to dev containers, [online], https://docs.github.com/en/codespaces/setting-up-your-project-for-codespaces/adding-a-dev-container-configuration/introduction-to-dev-containers [27.6.2024]

[33] Puma gem, [online], dostupno na https://github.com/puma/puma [19.9.2024]

[34] Simple_form gem, [online], dostupno na https://github.com/heartcombo/simple_form [19.9.2024]

[35] Turbo-rails gem, [online], dostupno na https://github.com/hotwired/turbo-rails [19.9.2024]

[36] Devise gem, [online], dostupno na https://github.com/heartcombo/devise [19.9.2024]

SAŽETAK

Kvizovi su u svakidašnjem životu sve popularniji, te doprinose velikom udjelu svakidašnje zabave pojedinaca, stjecanja novih znanja pojedinca i ispitivanju postojećih znanja. Kao rezultat rada opisani su i uspoređeni su *Ruby on Rails, Django, ASP.NET, CakePHP* i *Elixir* programski okviri za kreiranje web aplikacija. Također je izrađena web aplikacija koja omogućuje upravljanje, vođenje i rješavanje kvizova. Gostujućim korisnicima je samo omogućeno rješavanje kvizova i uvid na tablicu postignuća, registriranim korisnicima je omogućeno sve što i gostima uz mogućnost kreiranja novih kvizova i pitanja unutar njih koji mogu biti dva oblika: pitanje s tekstualnim odgovorom i pitanje na izbor točnih vrijednosti. Administrator ima uvid na sve kvizove i njihova pitanja, brisanja i mijenjanja ukoliko je to potrebno, te ima mogućnost kreiranja novih tema. Aplikacija je izrađena u programskom okviru *Ruby on Rails*, a baza podataka koja je korištena za ovu aplikaciju je *PostgreSQL*. Provjera identiteta korisnika, prijava korisnika i administratora u sustav je omogućena pomoću *Ruby gem*-a *Devise.*

Ključne riječi: informacijska tehnologija, kviz, programski okvir, *Ruby on Rails*, web aplikacija

ABSTRACT

Web application for setting and using quizzes

Quizzes are becoming increasingly popular in everyday life, contributing significantly to daily entertainment, the acquisition of new knowledge, and the assessment of existing knowledge. As part of this thesis, the Ruby on Rails, Django, ASP.NET, CakePHP and Elixir web application frameworks were described and compared. A web application that allows management, creation, and solving of quizzes was also developed. Guest users are only allowed to solve quizzes and view the scoreboard, while registered users have access to all the features available to guests, along with the ability to create new quizzes and questions. These questions can take two forms: questions with text based answers and questions with choice answers. The administrator has access to all quizzes and their questions, with the ability to delete or modify any of them if necessary, as well as create new topics. The application was developed using the Ruby on Rails web framework with PostgreSQL database. User identity verification, as well as user and administrator login, is enabled through the Devise Ruby gem.

Key words: information technology, framework, quiz, ruby on rails, web application

ŽIVOTOPIS

Leon Šumanovac rođen je 12.1.2000. u Osijeku. Osnovno obrazovanje započinje 2006. godine u Osnovnoj školi „August Harambašić“ Donji Miholjac. Po završetku osnovne škole 2014. godine obrazovanje nastavlja u Srednjoj Školi Valpovo gdje upisuje smjer elektrotehničar. Srednjoškolko obrazovanje završava 2018. godine te upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, preddiplomski smjer Računarstvo. Nakon završenog preddiplomskog studija 2022. godine upisuje diplomski studij Informacijske i podatkovne znanosti na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

PRILOZI

Prilog 1: Pristup izvornom kodu na GitHubu: https://github.com/Malileon/novi_diplomski