

Srednja škola Krapina  
Šetalište hrvatskog narodnog preporoda 6  
49000 Krapina, Hrvatska

# **Web aplikacija za izradu, rješavanje i statistiku ispita i obrazaca**

## **Elaborat završnog rada**

**Zanimanje:** tehničar za računalstvo

**Predmet:** Napredno i objektno programiranje

**Mentor:** Stjepan Šalković, mag. inf. univ. spec. oec.

**Učenik:** Ivan Dolovčak, 4.AT

2. svibnja 2024.

## Sadržaj

---

<b>1. Uvod</b>	<b>4</b>
1.1. Problematika . . . . .	4
1.2. Tema . . . . .	4
<b>2. Programski alati</b>	<b>5</b>
2.1. Uređivači izvornog koda . . . . .	5
2.2. Kontrola izvornog koda . . . . .	6
2.3. Terminal . . . . .	6
2.4. <i>Web stack</i> . . . . .	6
2.5. <i>Web</i> preglednici . . . . .	7
2.6. Ostalo . . . . .	7
<b>3. Programski i ostali jezici</b>	<b>8</b>
3.1. <i>Front-end</i> . . . . .	8
3.2. <i>Back-end</i> . . . . .	9
3.3. Ostalo . . . . .	9
<b>4. Arhitektura baze podataka</b>	<b>10</b>
4.1. Dizajn baze podataka . . . . .	10
4.1.1. Tipovi podataka . . . . .	10
4.1.2. Entitet <i>User</i> . . . . .	11
4.1.3. Entitet <i>Document</i> . . . . .	11
4.2. <i>SQL</i> skripta baze podataka . . . . .	11
4.2.1. Izrada MariaDB korisnika i baze podataka . . . . .	11
4.2.2. Izrada i povezivanje tablica . . . . .	12
<b>5. Izvedba aplikacije</b>	<b>13</b>
5.1. PHP konfiguracija . . . . .	13
5.2. Recikliranje koda . . . . .	13
5.2.1. <i>Util</i> klasa . . . . .	13
5.2.2. <i>DB</i> klasa . . . . .	14
5.2.3. Jezik aplikacije . . . . .	15
5.2.4. Osnovna struktura stranice . . . . .	16
5.3. Korisničke postavke ( <i>front-end</i> ) . . . . .	17
5.4. Korisničke postavke ( <i>back-end</i> ) . . . . .	18
5.4.1. <i>Preferences</i> klasa . . . . .	18
5.5. Registracija korisnika ( <i>front-end</i> ) . . . . .	19

5.5.1. Prikaz obrasca za registraciju . . . . .	19
5.5.2. HTML/PHP struktura obrasca registracije . . . . .	20
5.5.3. JS kod za validaciju . . . . .	21
5.6. Registracija korisnika ( <i>back-end</i> ) . . . . .	22
5.6.1. Obrada obrasca za registraciju . . . . .	22
5.6.2. Spremanje korisnika u bazu podataka . . . . .	24
5.7. Prijava korisnika . . . . .	25
5.8. Profil korisnika . . . . .	26
5.9. Početna stranica ( <i>front-end</i> ) . . . . .	27
5.9.1. Popis dokumenata . . . . .	27
<b>6. Zaključak</b>	<b>29</b>
<b>Popis slika</b>	<b>30</b>
<b>Popis izvornih kodova</b>	<b>31</b>
<b>Literatura</b>	<b>32</b>

# 1. Uvod

---

## 1.1. Problematika

Ispiti i obrasci su svakodnevica obrazovnog i poslovnog okruženja. Napretkom tehnologije i digitalizacijom pojavila se ideja za automatiziranjem procesa izrade profesionalnih obrazaca i ispita. Tako su nastale prve (web) aplikacije za izradu, ali i obradu takvih dokumenata. Takve aplikacije su bile inspiracija za temu ovog završnog rada.

## 1.2. Tema

Tema završnog rada je web aplikacija za izradu, rješavanje i statistiku ispita i obrazaca. Nadalje u tekstu riječ *dokument* se odnosi na obrase i na ispite. Aplikacijom je moguće izraditi dokumente s određenim postavkama: tip dokumenta (obrazac ili ispit), rok predaje, broj dozvoljenih pokušaja predaje i vidljivost dokumenta (javni, privatni ili skriveni). Nakon izrade dokumenta moguće je i mijenjati te postavke. Također je moguće i obrisati dokumente.

Aplikacija pruža različite prikaze popisa dokumenata. Moguće je vidjeti vlastite dokumente, ali i pretražiti tuđe, te ih i riješiti (predati).

Da bi krajnji korisnik/ca aplikacije mogao/la pristupiti svim mogućnostima aplikacije, potrebno se registrirati i prijaviti u aplikaciju. Korisnik/ca nakon prijave može mijenjati detalje svog profila (ime, prezime, korisničko ime i e-mail adresa) te izbrisati svoj profil.

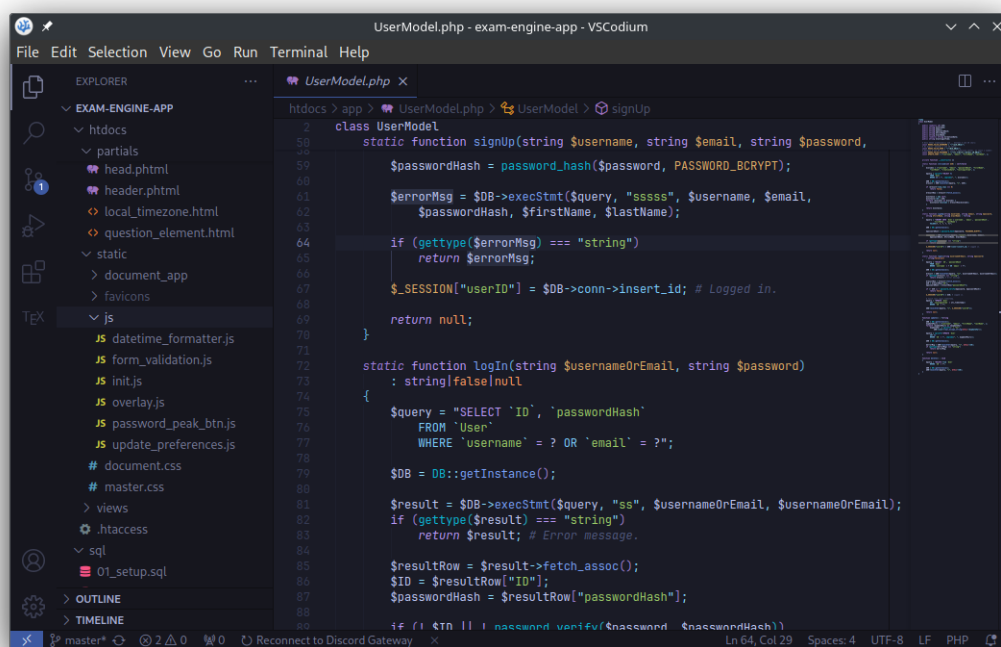
Također, korisnik/ca može mijenjati izgled aplikacije i bez da je prijavljen/a u aplikaciju. Moguće je odabrati jezik aplikacije (hrvatski ili engleski), temu (svijetla ili tamna) i proizvoljnu boju isticanja (engl. *accent color*).

## 2. Programski alati

### 2.1. Uređivači izvornog koda

#### VSCodium

*VSCodium* je program uređivač izvornog koda koji je razvio Microsoft 2015. g. Vrlo je popularan, funkcionalan i prilagodljiv. Podržava otklanjanje grešaka, ugrađenu *git* kontrolu, isticanje sintakse i automatsko dovršavanje koda. *VSCodium* je inačica programa *VSCode* iz koje je uklonjeno telemetrijsko prikupljanje korisničkih podataka. Izvorni kod programa je besplatan i otvoren.



**Slika 1:** VSCodium uređivač koda s otvorenim projektom.

*VSCodium* sam koristio kao glavni uređivač teksta za cijeli projekt. Omogućio mi je brzo, organizirano i funkcionalno digitalno radno okruženje.

#### vim

*vim* je program uređivač izvornog koda nastao 1991. g. Za razliku od *VSCodiuma* koji ima grafičko korisničko sučelje (GUI), *vim* ima isključivo tekstualno korisničko sučelje (TUI).

*vim* sam koristio isključivo za uređivanje pojedinačnih datoteka izvan projekta i malih konfiguracijskih datoteka (npr. Apache konfiguracija).

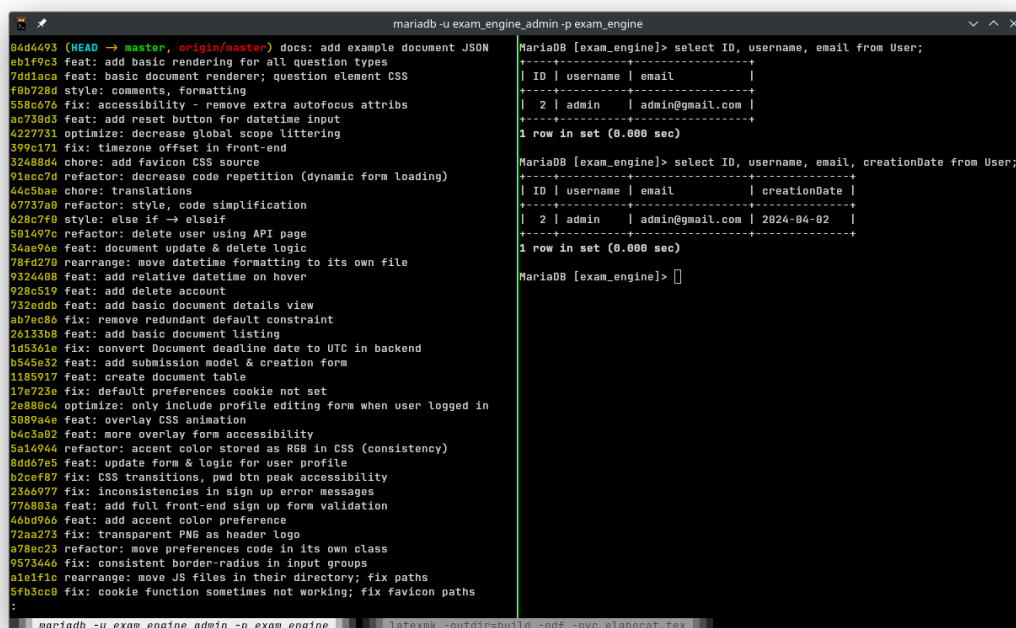
## 2.2. Kontrola izvornog koda

*git* je program za upravljanje izvornim kodom nastao 2005. g. Njegov kreator Linus Torvalds je ujedno i voditelj razvoja Linux operativnog sustava.

Pomoću *gita* sam spremao i bilježio napredak svog projekta. Također sam objavio izvorni kod svoje aplikacije na web sjedištu *GitHubu*, koji nudi uslugu tzv. *hostinga* izvornog koda, što je ujedno služilo za čuvanje sigurnosne kopije.

## 2.3. Terminal

*kitty* je napredni Linux emulator terminala (konzola). Koristio sam ga za uklanjanje grešaka, praćenje izlaznih informacija *Apache* servera, upravljanje bazom podataka slanjem naredba, za *git* naredbe i općenito za upravljanje datotekama u projektu.



```
mariadb -u exam_engine_admin -p exam_engine

84d4493 (HEAD -> master, origin/master) docs: add example document JSON
eb1f9c3 feat: add basic rendering for all question types
7dd1aea feat: basic document renderer; question element CSS
f0b728d style: comments, formatting
558cc76 fix: accessibility - remove extra autofocus attribs
ac730d3 feat: add reset button for datetime input
4227731 optimize: decrease global scope littering
399c171 fix: timezone offset in front-end
32488d4 chore: add favicon CSS source
91ecc7d refactor: decrease code repetition (dynamic form loading)
44c5bae chore: translations
67737a0 refactor: style, code simplification
628c7e0 style: else if -> elseif
501497c refactor: delete user using API page
34ae94e feat: document update & delete logic
78f0270 rearrange: move datetime formatting to its own file
9324488 feat: add relative datetime on hover
928c519 feat: add delete account
732eddb feat: add basic document details view
ab7ec86 fix: remove redundant default constraint
26133b8 feat: add basic document listing
1d5361e fix: convert Document deadline date to UTC in backend
b5d5e32 feat: add submission model & creation form
1185917 feat: create document table
17e723e fix: default preferences cookie not set
2e888c4 optimize: only include profile editing form when user logged in
3089a4e feat: overlay CSS animation
b4c3a02 feat: more overlay form accessibility
5a14944 refactor: accent color stored as RGB in CSS (consistency)
8dd67e5 feat: update form & logic for user profile
b2aeef7 fix: CSS transitions, pnd btn peak accessibility
236e977 fix: inconsistencies in sign up error messages
776803a feat: add full front-end sign up form validation
46bd966 feat: add accent color preference
72ae273 fix: transparent PNG as header logo
a78ec23 refactor: move preferences code in its own class
9573446 fix: consistent border-radius in input groups
a1e1f1c rearrange: move JS files in their directory; fix paths
5fb5cc8 fix: cookie function sometimes not working; fix favicon paths

MariaDB [exam_engine]> select ID, username, email from User;
+-----+-----+-----+
| ID | username | email |
+-----+-----+-----+
| 2 | admin | admin@gmail.com |
+-----+-----+-----+
1 row in set (0.008 sec)

MariaDB [exam_engine]> select ID, username, email, creationDate from User;
+-----+-----+-----+-----+
| ID | username | email | creationDate |
+-----+-----+-----+-----+
| 2 | admin | admin@gmail.com | 2024-04-02 |
+-----+-----+-----+-----+
1 row in set (0.008 sec)

MariaDB [exam_engine]>
```

Slika 2: *kitty* terminal s više otvorenih kartica i panela.

## 2.4. Web stack

Aplikacija je razvijana na klasičnom *LAMP* web stogu programske podrške (engl. *web stack*) - Linux, Apache, MariaDB i PHP. Apache je popularan HTTP web server. MariaDB je popularan DBMS (engl. *Database Management System*).

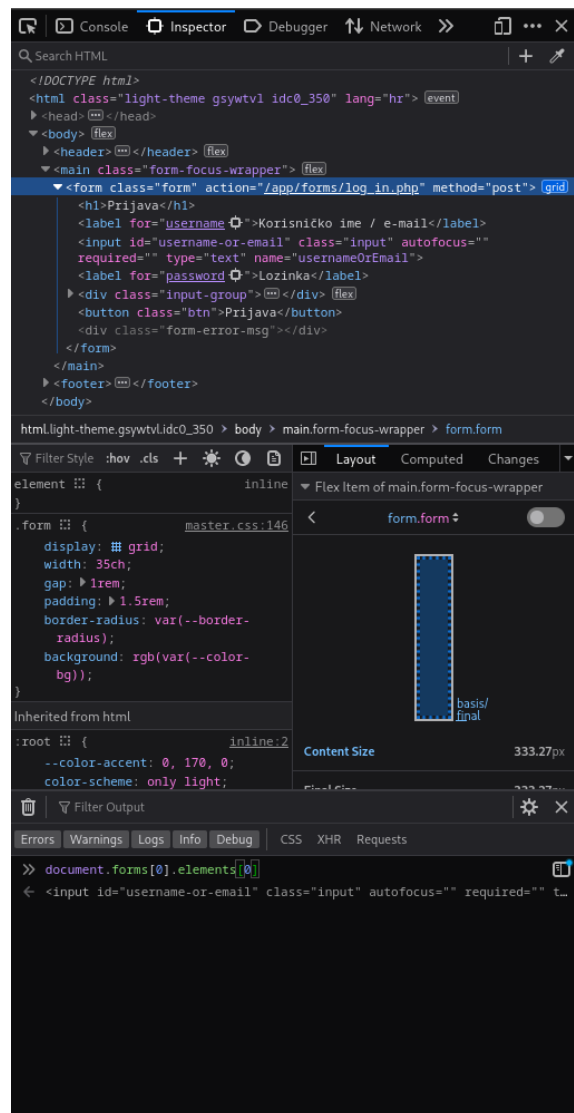
## 2.5. Web preglednici

Aplikacija je testirana na dvoje najpopularnijih web preglednika: Google Chrome i Mozilla Firefox. Potrebno je testirati na više preglednika radi konzistencije izgleda sučelja jer svaki preglednik ima malo drugačiju implementaciju HTML, CSS i JS standarda.

Važno je spomenuti i integrirane alate za web programere koje nude oba web preglednika: inspektor (engl. *inspector*), konzola (engl. *console*), spremište (engl. *storage*) itd.

## 2.6. Ostalo

- JSON linter/validator/minifier:  
<https://jsonlint.com/>
- regular expressions 101:  
<https://regex101.com/>
- phpMyAdmin:  
<https://www.phpmyadmin.net/>
- Fontovi:  
<https://fonts.google.com/>
- Ikonice:  
<https://icons.getbootstrap.com/>



**Slika 3:** Alati za web programere, preglednik Mozilla Firefox.

## 3. Programski i ostali jezici

---

### 3.1. *Front-end*

Pod *front-end* dio aplikacije spadaju sve stranice web aplikacije koje su vidljivi krajnjem korisniku/ci. Preko njih korisnik/ca upravlja i prosljeđuje podatke aplikaciji.

#### **HTML**

HTML (engl. *HyperText Markup Language*) je označni jezik za definiranje osnovne strukture i sadržaja web dokumenata. Koristim aktualnu inačicu i standard *HTML5*.

#### **CSS**

CSS (engl. *Cascading Style Sheets*) je jezik za oblikovanje i stiliziranje HTML dokumenata. Omogućava definiranje položaja elemenata, boje, fonta, efekata i sl. Koristim aktualnu inačicu i standard *CSS3*.

#### **JavaScript**

*JavaScript* (skraćeno *JS*) je programski jezik za interakciju s korisnikom, razmjenu podataka i kreiranje dinamične, funkcionalne web stranice. Jezik je visoke razine, dinamički pisan, OO (objektno orijentiran) i interpretiran. Valja spomenuti i DOM (engl. *Document Object Model*) pomoću kojeg je moguće dinamički manipulirati HTML-om i CSS-om dokumenta.



**Slika 4:** *Front-end jezici.*



### 3.2. Back-end

#### SQL (MariaDB)

SQL (engl. *Structured Query Language*) je deklarativni jezik koji služi za postavljanje upita (engl. *query*) DBMS-u (bazi podataka). Upitima je moguće kreirati (engl. *create*), čitati (engl. *read*), ažurirati (engl. *update*) i brisati (engl. *delete*) podatke - tzv. CRUD operacije.

#### PHP

PHP je skriptni programski jezik koji se izvršava isključivo na poslužitelju. Jezik je visoke razine, dinamički pisan, OO (objektno orijentiran) i interpretiran. Služi za komuniciranje aplikacije s bazom podataka. Nakon izvršavanja na serveru, sav HTML izlaz PHP skripte šalje se klijentskom računalu i korisnik vidi gotovu stranicu.



**Slika 5:** Back-end programski jezici i programska podrška.

### 3.3. Ostalo

#### JSON

JSON (JavaScript Object Notation) je tekstualni format za spremanje podataka. JSON zapisi razumljivi su i čovjeku i računalu (programu).

#### LaTeX

PDF dokument elaborata napravljen je pomoću označnog jezika *LaTeX* i prevoditelja (engl. *compiler*) *latexmk*. Iz izvornog *LaTeX* koda prevoditelj generira profesionalno formatirani i stilizirani PDF dokument.

## 4. Arhitektura baze podataka

### 4.1. Dizajn baze podataka

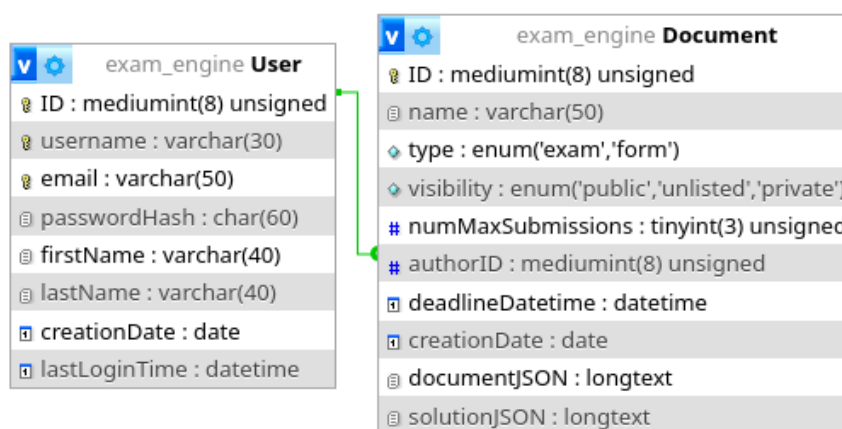
Baza podataka sastoji se od 2 tablice: *User* – korisnici i *Document* – dokumenti. Veza je 1:M (1 naprema više).

Svaki zapis entiteta jedinstven je po svojem primarnom ključu (engl. *primary key*, oznaka *PK*). PK je broj koji se inkrementira za svaki novi zapis tog entiteta.

Tablice se vezuju pomoću tzv. stranih ključeva (engl. *foreign key*, oznaka *FK*). *FK* pokazuje na odgovarajući *PK* u nekoj drugoj tablici, i tipovi podataka im moraju biti identični.

Većina atributa imaju i *not null constraint*, što znači da vrijednosti tog atributa svakog zapisa u tablici treba biti poznat (definiran).

Baza podataka je projektirana u skladu s prve 3 normalne forme (1NF, 2NF, 3NF).



Slika 6: Dijagram baze podataka prikazan u phpMyAdminu.

#### 4.1.1. Tipovi podataka

Važno je kvalitetno odabrati tipove podatka atributa u tablicama tako da se uštedi na memoriji, ali ujedno osigura i buduća prilagodba i proširivost funkcionalnosti baze.

Primjer toga je način korištenja char i varchar tipova podataka. Brojevi u zagradi (npr. *varchar(40)*) označavaju najveći broj znakova u tom atributu. Također su korišteni i unsigned mediumint primarni ključevi da se dodatno uštedi na pohrani.

#### 4.1.2. Entitet *User*

Ova tablica služi za pohranu podataka o svakom registriranom korisniku. Čuvaju se: korisničko ime, e-mail, ime, prezime, datum registracije i datum i vrijeme zadnje prijave. Također postoji i atribut *passwordHash*, koji sprema lozinku korisničkog računa, čime se ostvaruje autentikacija i autorizacija korisničkih podataka.

Valja napomenuti da se prije pohrane u bazu sve lozinke procesiraju kroz jednosmjerni algoritam za tzv. *hashiranje*, tj. kriptira se, što služi kao osnovna mjera zaštite korisničkih lozinka.

#### 4.1.3. Entitet *Document*

Ova tablica služi za pohranu podataka o dokumentima, tj. ispitima i obrascima. Čuvaju se: naziv dokumenta, tip, vidljivost, broj dozvoljenih pokušaja (predaja), autor dokumenta (FK), datum i vrijeme roka predaje i datum izrade.

Atribut *documentJSON* je tipa *JSON* i služi za pohranu pitanja i sadržaja od kojih se sastoji svaki dokument.

Atribut *solutionJSON* pohranjuje samo rješenja dokumenta (ako je taj dokument tipa ispit). Kako bi aplikacija bila povjerljiva, iz sigurnosnih razloga, rješenja dokumenta (odgovori) su odvojena od samih pitanja.

### 4.2. SQL skripta baze podataka

#### 4.2.1. Izrada MariaDB korisnika i baze podataka

```
-- Datoteka: sql/01_setup.sql
-- Napomena: korisničko ime, ime baze i lozinka su drugačiji na
--            produkcijskom serveru. Ovo je samo primjer.
create user 'exam_engine_admin'@'localhost' identified by 'admin';
create database 'exam_engine';
-- Davanje "exam_engine_admin" korisniku svih prava (potpuna kontrola
--            nad bazom):
grant all privileges on 'exam_engine'.* to 'exam_engine_admin'@'
localhost';
```

**Izvorni kod 1:** SQL – Izrada MariaDB korisnika i baze podataka.

## 4.2.2. Izrada i povezivanje tablica

```
-- Datoteka: sql/02_schema.sql

create table 'User' (
    'ID' mediumint unsigned not null auto_increment,
    'username' varchar(30) not null,
    'email' varchar(50) not null,
    -- PHP password_hash() koristi algoritam PASSWORD_BCRYPT koji
    -- uvijek vraća hash duljine 60 znakova:
    'passwordHash' char(60) not null,
    'firstName' varchar(40) not null,
    'lastName' varchar(40) not null,
    'creationDate' date not null default utc_date(),
    'lastLoginTime' datetime not null default utc_timestamp(),
    primary key ('ID'),
    constraint 'UK_username' unique key ('username'),
    constraint 'UK_email' unique key ('email')
);

create table 'Document' (
    'ID' mediumint unsigned not null auto_increment,
    'name' varchar(50) not null,
    'type' enum("exam", "form") not null,
    'visibility' enum("public", "unlisted", "private")
    not null default "private",
    'numMaxSubmissions' tinyint unsigned,
    'authorID' mediumint unsigned not null,
    'deadlineDatetime' datetime,
    'creationDate' date not null default utc_date(),
    'documentJSON' json,
    'solutionJSON' json,
    primary key ('ID'),
    constraint 'FK_author'
    foreign key ('authorID') references 'User'('ID')
    on delete cascade
);
```

Izvorni kod 2: SQL – Izrada i povezivanje tablica.

## 5. Izvedba aplikacije

---

### 5.1. PHP konfiguracija

Ova PHP skripta se obavezno *includea* na vrhu svake druge PHP skripte. Sadrži neke osnovne PHP postavke.

```
<?php
# Datoteka: htdocs/app/config.php

declare(strict_types=1);

# Sintaksne greške u kodu se prikazuju samo u razvojnem okruženju, tj.
# ako je definirana Apache varijabla DEVELOPMENT.
if (isset($_SERVER["DEVELOPMENT"])) {
    error_reporting(E_ALL);
    ini_set("display_errors", true);
    ini_set("display_startup_errors", true);
    mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
}

# U back-endu su svi datumi i vremena spremljeni u UTC vremenskoj zoni
date_default_timezone_set("UTC");

# Automatsko includeanje klasa:
spl_autoload_register(fn($className) => require "$className.php");

# Učitavanje korisničkih postavki:
if (! isset($_COOKIE[Preferences::COOKIE_NAME]))
    Preferences::savePreferences(Preferences::DEFAULT_PREFERENCES);

# Korisničke postavke su dostupne u ovoj globalnoj varijabli:
$preferences = Preferences::loadPreferences();

# Učitavanje jezika stranice:
require_once "lang_{$preferences["lang"]}.php";
```

**Izvorni kod 3:** PHP – Osnovna konfiguracija

### 5.2. Recikliranje koda

Jedan od temeljnih principa programiranja je izbjeći lošu redundanciju u kodu (engl. *DRY - Don't Repeat Yourself.*). Iz tog razloga sam kod aplikacije pisao vrlo organizirano i modularno, te pokušavao definirati metode i funkcije za kod koji se ponavlja.

Npr. postoje klase u projektu *UserModel* i *DocumentModel*. *UserModel* sadrži sve metode i attribute koje opisuju korisnika. Slično vrijedi i za *DocumentModel* klasu.

#### 5.2.1. *Util* klasa

*Util* klasa (engl. *Utility*) sadrži statičke metode koje se često koriste na raznim mjestima u projektu, ali ne mogu se kategorizirati u klasu za sebe.

```

<?php
# Datoteka: htdocs/app/DB.php

class Util
{
    # Ova metoda služi za dohvaćanje poruke greške (ako ta greška postoji).
    # U kodu se na raznim mjestima varijabli sesije formErrorMsg dodjeljuje neka poruka greške, koja se kasnije prikazuje korisniku/ci.
    public static function getFormError() : ?string
    {
        $errMsg = $_SESSION["formErrorMsg"] ?? null;
        unset($_SESSION["formErrorMsg"]);
        return $errMsg;
    }

    # Ova metoda služi za sanitizaciju podataka poslanih iz obrasca.
    public static function sanitizeFormData(string $data) : ?string
    {
        $sanitizedData = htmlspecialchars(stripslashes(trim($data)));
        if (empty($sanitizedData))
            return null;
        return $sanitizedData;
    }

    # Ova metoda obavlja redirekciju na dani URL.
    public static function redirect(string $URL) : void
    {
        header("Location: $URL");
        die;
    }
}

```

**Izvorni kod 4:** PHP – Util klasa

### 5.2.2. DB klasa

DB klasa (engl. *DataBase*) sadrži često korištene metode i svojstva vezane za upravljanje bazom podataka, koje su smještene u tzv. *singleton* klasu.

```

<?php
# Datoteka: htdocs/app/DB.php

require "sql_auth.php";

class DB
{
    private static self $obj; # Singleton instance
    public MySQLi $conn;      # MySQLi objekt - veza s bazom podataka

    # Prilikom inicijalizaciju singleton instance, uspostavlja se veza s bazom podataka.
    private function __construct()
    {
        # Konstruktoru MySQLi se prosljeđuju informacije potrebne za uspostavu veze: korisničko ime, host, lozinka i ime baze.
        # Ove konstante su definirane u zasebnoj datoteci sql_auth.php
        $this->conn = new MySQLi(SQL_HOSTNAME, SQL_USERNAME, SQL_PASSWORD, SQL_DATABASE);
    }
}

```

```

# Singleton klasa - moguće je napraviti samo jednu instancu ove
klase, čime se sprečava redundantno uspostavljanje i prekidanje
veze s bazom podataka (efikasnost).
static function getInstance() : self
{
    if (! isset(self::$obj))
        self::$obj = new self;

    return self::$obj;
}

# Ova metoda izvršava dani SQL upit pomoću tzv. prepared
statementa - još jedan sigurnosni sloj.
function execStmt(string $query, ?string $types, mixed ...
$queryArgs) : MySQLi_result|false|string
{
    try {
        $stmt = $this->conn->prepare($query);
        if (isset($types))
            $stmt->bind_param($types, ...$queryArgs);
        $stmt->execute();
        return $stmt->get_result();
    }
    catch (MySQLi_SQL_exception $e) {
        return $e->getMessage();
    }
}

# Ova metoda testira je li vrijednost za dani stupac u tablici
zauzeta.
# Koristi se kasnije za provjeru dostupnosti korisničkog imena i e
-mail adrese.
function isTaken(string $table, string $column, string $value) :
bool|string
{
    $query = "SELECT '$column'
FROM '$table'
WHERE '$column' = ?";

    $DB = self::getInstance();
    $result = $DB->execStmt($query, "s", $value);

    if (gettype($result) === "string")
        return $result;
    elseif ($result->num_rows > 0)
        return true;
    else
        return false;
}

# Prije uništavanja singleton instance, prekida se veza s bazom
podataka.
function __destruct()
{
    $this->conn->close();
}
}

```

**Izvorni kod 5:** PHP – DB klasa

### 5.2.3. Jezik aplikacije

Višejezičnost aplikacije je postignuta tako da su svi znakovni nizovi korišteni u aplikaciji organizirani u dvije slične PHP datoteke - lang\_en.php i

lang\_hr.php. Obje datoteke sadrže asocijativno polje *LANG* – ključevi su isti, a vrijednosti prevedene.

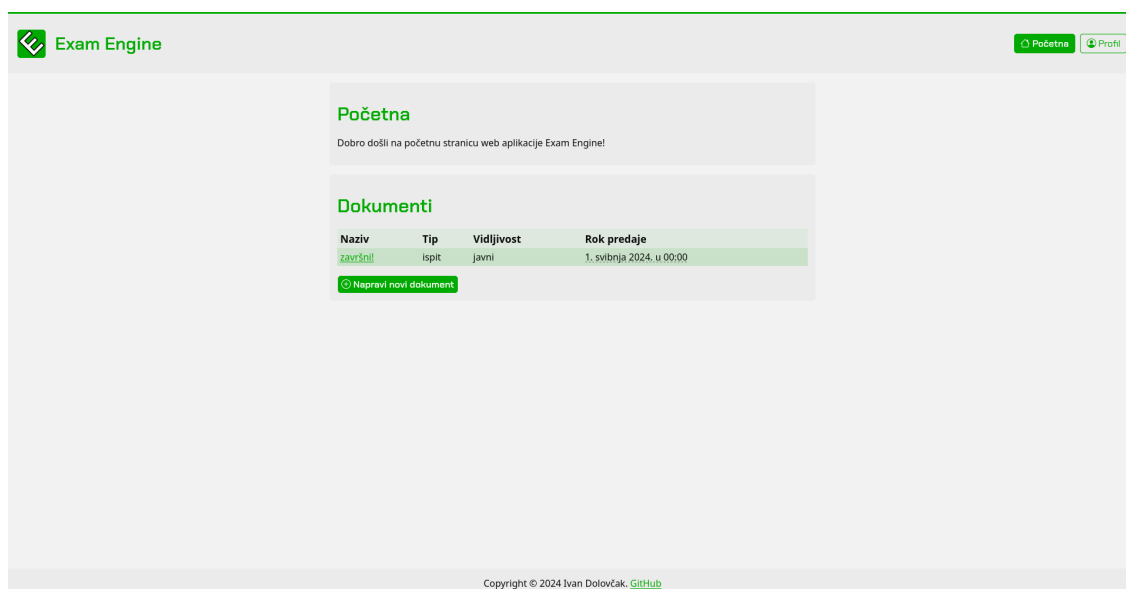
```
<?php
const LANG = [
    # ...
    "invalidPreferences" => "Invalid preferences set.",
    "accentColor" => "Accent color",
    "editProfile" => "Edit profile",
    "save" => "Save",
    "deleteAccount" => "Delete account",
    "deleteAccountConfirmation" => "Are you sure you want to delete
your account and all of its related data? This action is
irreversible!",
    # ...
];
```

**Izvorni kod 6:** PHP – Engleski znakovni nizovi

```
<?php
const LANG = [
    # ...
    "invalidPreferences" => "Nevažeće postavke.",
    "accentColor" => "Istaknuta boja",
    "editProfile" => "Uredi profil",
    "save" => "Spremi",
    "deleteAccount" => "Izbriši račun",
    "deleteAccountConfirmation" => "Jeste li sigurni da želite
izbrisati svoj račun i sve povezane podatke? Ova radnja je
nepovratna!",
    # ...
];
```

**Izvorni kod 7:** PHP – Hrvatski znakovni nizovi

## 5.2.4. Osnovna struktura stranice

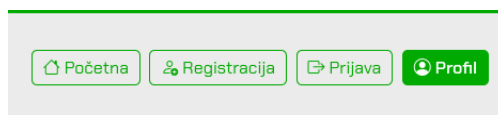


**Slika 7:** Početna stranica s popisom dokumenata.



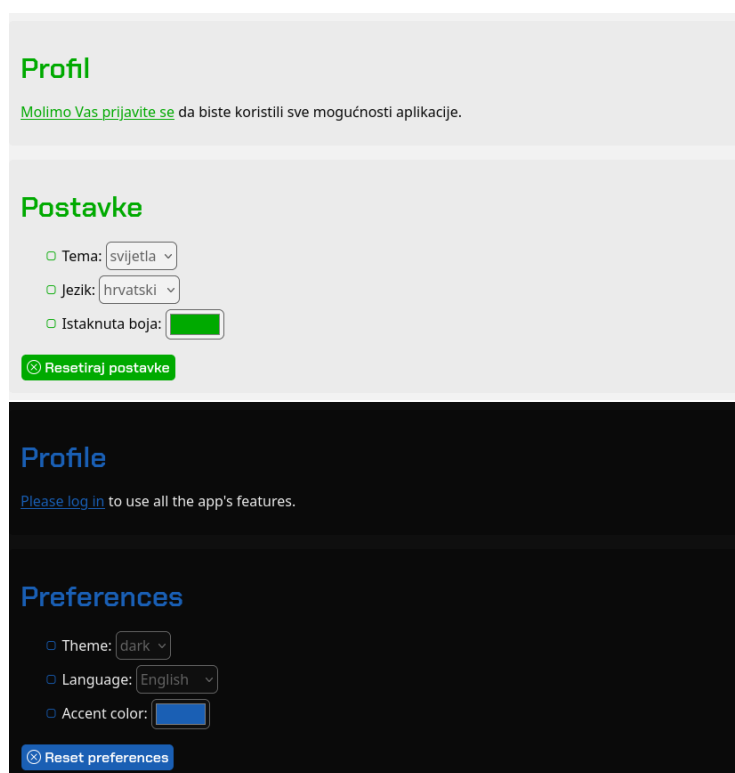
Na svakoj stranici se ponavljaju zaglavlje i podnožje. Zaglavlje i podnožje stranice su stavljani u zasebne datoteke (header.phtml i footer.phtml) te se na svakoj stranici *includeaju*.

Zaglavlje sadržava logotip aplikacije, naslov i navigator s poveznicama koje vode do glavnih stranica (engl. *views*) aplikacije. Navigator mijenja svoj sadržaj ovisno o tome je li korisnik/ca prijavljen/a. Ako nije, prikazuju se poveznice za registraciju i prijavu. Ako je, onda su te poveznice sakrivene.



**Slika 8:** Navigator kad korisnik/ca nije prijavljen/a.

### 5.3. Korisničke postavke (*front-end*)



**Slika 9:** Prilagodljivost sučelja (*profile.phtml*)

Izgled sučelja aplikacije je prilagodljiv korisniku. Korisničke postavke (engl. *Preferences*) sastoje se od:

- teme (svijetla ili tamna),

- jezika (engleski ili hrvatsi)
- i boje isticanja (proizvoljna).

## 5.4. Korisničke postavke (*back-end*)

Da bi mijenjao/la ove postavke prikaza, korisnik/ca ne mora biti prijavljen/a u aplikaciju, jer se ove postavke spremaju u tzv. kolačić (engl. *cookie*). Ako kolačić ne postoji (prva posjeta *web* sjedištu), dodjeljuju se zadane (*defaultne*) postavke (svijetla tema, zelena boja isticanja, engleski jezik).

### 5.4.1. *Preferences* klasa

```
<?php
class Preferences
{
    # Ime kolačića:
    const COOKIE_NAME = "examEnginePreferences";
    # Zadane postavke:
    const DEFAULT_PREFERENCES = [
        "theme" => "light",
        "lang" => "en",
        "accentColor" => "#00AA00",
    ];
    # Za validaciju postavki:
    const VALID_PREFERENCES = [
        "theme" => ["dark", "light", ],
        "lang" => ["en", "hr", ],
        "reset" => null,
        "accentColor" => null,
    ];

    # Privatni konstruktor - nemoguće je instancirati klasu:
    private function __construct() {}

    static function savePreferences($preferences) : void
    {
        # 3. argument - rok trajanja kolačića, 4. argument: putanja u
        # kojoj je kolačić dostupan (/ znači cijelo web sjedište)
        setcookie(self::COOKIE_NAME, json_encode($preferences),
            strtotime("+1 year"), "/");
    }

    # Učitavanje postavki u asocijativno globalno polje:
    static function loadPreferences() : array
    {
        return json_decode($_COOKIE[self::COOKIE_NAME], associative:
            true);
    }
}
```

**Izvorni kod 8:** *PHP – Preferences klasa*

## 5.5. Registracija korisnika (*front-end*)

### 5.5.1. Prikaz obrasca za registraciju

**Registracija**

Korisničko ime  
aaasd

E-mail  
admin@gmail.com

Lozinka  
••••••••

Ime  
Ivan

Prezime  
Dolovčak

Registracija

E-mail je zauzet.

**Registracija**

Korisničko ime  
aaasd

E-mail  
admin1@gmail.com

Lozinka  
•

Ime  
Ivan

Prezime  
Dolovčak

Registracija

Duljine barem 8 znakova, barem 1 veliko slovo i 1 broj.

**Slika 10:** Obrazac za registraciju – validacija u realnom vremenu

Kako bi koristio/la sve mogućnosti aplikacije, korisnik/ca mora biti registriran/a. To obavlja preko ovog obrasca, koji je intuitivan jer ima ugrađenu tzv. *front-end* validaciju.

Također, desno od polja za unos lozinke nalazi se gumb za prikazivanje lozinke. Lozinka se prikazuje samo kad korisnik/ca drži gumb pritisnutim, inače je lozinka skrivena. To je postignuto trivijalnim JS kodom.

Sva polja moraju biti popunjena i ispravna. Ukoliko neko polje nije ispravno, na dnu obrasca se u realnom vremenu (prije podnašanja obrasca) prikazuje povratna informacija korisniku/ci te mu/joj nije dozvoljeno podnašanje obrasca.

Ukoliko je ispravno, polje se blago osjenča zelenom bojom. Ovo su uvjeti za ispravnost polja:

- Korisničko ime: Koristite velika i mala slova bez dijakritičkih znakova, brojeve i donje crte, duljine najmanje 4 znakova.

- Lozinka: Duljine barem 8 znakova, barem 1 veliko slovo i 1 broj.
- Ime i prezime: Duljine najmanje 3 znakova, bez brojeva.
- Korisničko ime i e-mail adresa ne smiju biti zauzeti

Valja napomenuti da su tijekom dizajniranja *front-enda* uzete u obzir osnovne prakse pristupačnosti (engl. *accessibility*): autofocus atributi, korištenje label elemenata koji su povezani sa svojim poljima za unos, te je za svako polje sačuvan tabindex atribut.

### 5.5.2. HTML/PHP struktura obrasca registracije

```
<!-- Datoteka: htdocs/partials/form_sign_up.phtml -->
<?php
    # Prije includeanja obrasca, postavlja se varijabla $formType (hrv
    . tip obrasca) prema kojoj se određuje naslov obrasca i action URL.
    # Ovo je potrebno zato jer se ovaj identični kod reciklira na
    drugoj stranici (izmjena detalja profila). Na toj stranici je
    obrazac u skočnoj modalnoj formi.

    $formType ??= "create";
    if ($formType === "update") {
        $formTitle = LANG["editProfile"];
        $formAction = "/app/forms/sign_up.php?update";
    }
    else {
        $formTitle = LANG["signUp"];
        $formAction = "/app/forms/sign_up.php";
    }
?>
<form action="<?=$formAction?>" method="post" class="form form-
validate">

<?php if ($formType === "update"): ?>
    <!-- Gumb za zatvaranje modalnog prozora. -->
    <button type="button" data-overlay="edit-profile" class="btn btn-
close-overlay">
        <i class="bi bi-x-lg"></i></button>
<?php endif; ?>

    <h1><?=$formTitle?></h1>

    <!-- Labela i polje za upis korisničkog imena. -->
    <label for="username"><?=$LANG["username"]?></label>
    <!-- Prvo polje se automatski fokusira ako obrazac nije u skočnom
    modalnom prozoru. -->
    <input <?=$formType === "create" ? "autofocus" : null?>
        required type="text" name="username" id="username"
        pattern="<?=$trim(UserModel::REGEX_VALID_USERNAME, "/")?>"
        title="<?=$LANG["invalidUsername"]?>" class="input">
    <!-- pattern atribut sadržava regular expression za validaciju
    korisničkog imena. -->

    <label for="email"><?=$LANG["email"]?></label>
    <input required type="email" name="email" id="email" class="input"
        title="<?=$LANG["invalidEmail"]?>">

<?php if ($formType !== "update"): # Polje za lozinku se prikazuje
samo tijekom registracije ?>
```

```

<label for="password"><?=LANG["password"]?></label>
<div class="input-group">
  <input required type="password" name="password" id="password"
    pattern="<?=trim(UserModel::REGEX_VALID_PASSWORD, "/")?>"
    title="<?=LANG["invalidPassword"]?>" class="input">
  <button type="button" id="btn-password-peak" class="btn">
    <i class="bi bi-eye-fill"></i>
  </button>
</div>
<?php endif; ?>

<label for="first-name"><?=LANG["firstName"]?></label>
<input required type="text" name="firstName" id="first-name"
  pattern="<?=trim(UserModel::REGEX_VALID_NAME, "/")?>"
  title="<?=LANG["invalidName"]?>" class="input">

<label for="last-name"><?=LANG["lastName"]?></label>
<input required type="text" name="lastName" id="last-name"
  pattern="<?=trim(UserModel::REGEX_VALID_NAME, "/")?>"
  title="<?=LANG["invalidName"]?>" class="input">

<button class="btn"><?=$formTitle?></button>

<!-- Ovaj element sadržava povratnu informaciju korisniku (ako je
došlo do neke greške). -->
<div class="form-error-msg"><?=Util::getFormError();?></div>
</form>
<script>
<?php
# Na obrascu za izmjena detalja profila se popunjuju već postojeće
vrijednosti iz baze podataka.
if ($formType === "update") {
  foreach (UserModel::UPDATE_VARS as $inputName) {
    $var = $user->$inputName;
    echo "document.forms[0].elements['$inputName'].
defaultValue = '$var';\n";
  }
}
?>
</script>

```

**Izvorni kod 9:** HTML/PHP – Obrazac za registraciju.

### 5.5.3. JS kod za validaciju

```

// Datoteka: htdocs/static/js/form_validation.js

const mainForm = document.forms[0];

// Funkcija za slanje zahtjeva API-ju aplikacije pomoću JS fetch() API
-ja.
async function apiGet(requestType, value)
{
  value = encodeURIComponent(value);

  const baseURL = "/app/api/form_validation.php";
  const requestURL = `${baseURL}?request=${requestType}&value=${
value}`;

  const request = await fetch(requestURL);
  return request.text();
}

async function validateInput(input)

```

```

{
    const errorMsgElement = mainForm.getElementsByClassName("form-
error-msg")[0];

    // Specijalna provjera dostupnosti e-mail adrese i korisničkog
    imena korisnika.
    if (input.name === "email" && input.value !== input.defaultValue)
    {
        const validationMessage = await apiGet("isUserEmailTaken",
input.value);
        input.setCustomValidity(validationMessage);
    }
    else if (input.name === "username" && input.value !== input.
defaultValue) {
        const validationMessage = await apiGet("isUsernameTaken",
input.value);
        input.setCustomValidity(validationMessage);
    }

    // Prikaz greške/traženog formata na dnu obrasca:
    if (! input.checkValidity() && input.value) {
        if (input.validity.customError)
            errorMsgElement.innerText = input.validationMessage;
        else
            errorMsgElement.innerText = input.title;
    }
    else {
        input.setCustomValidity("");
        errorMsgElement.innerText = "";
    }
}

// Svakom polju na formi se dojdjeljuje gornja funkcija za validaciju
koja se pokreće 200 ms nakon zadnjeg pritiska tipke
for (const input of mainForm.elements) {
    if (input.type === "hidden")
        continue;
    input.addEventListener("keyup", () => {
        setTimeout(async() => validateInput(input), 200);
    });
    input.addEventListener("focus", async() =>
        validateInput(input)
    );
    input.addEventListener("blur", () => {
        mainForm.getElementsByClassName("form-error-msg")[0].innerText
        = "";
    });
}

```

**Izvorni kod 10:** JS – Live validacija forme.

## 5.6. Registracija korisnika (back-end)

Front-end validaciju vrlo je lako zaobići, stoga je iz sigurnosnih razloga potrebno obaviti validaciju i u back-end dijelu aplikacije.

### 5.6.1. Obrada obrasca za registraciju

```

<?php
# Datoteka: htdocs/app/forms/sign_up.php

```

```

require_once "config.php";
session_start();

# Definiranje stranica na koje će se preusmjeriti korisnika - prva ako
# je zahtjev uspješan, druga ako je došlo do neke greške.
if (isset($_GET["update"])) {
    $successPage = "/views/profile.phtml";
    $failurePage = "/views/profile.phtml";
} else {
    $successPage = "/views/index.phtml";
    $failurePage = "/views/sign_up.phtml";
}

$requiredPostVars = ["username", "email", "firstName", "lastName", ];
if (! isset($_GET["update"]))
    # Polje za lozinku je obavezno samo tokom registracije:
    $requiredPostVars[] = "password";

# Testiranje jesu li sva obavezna polja prisutna u POST zahtjevu. Ako
# nisu, preusmjerava se korisnika i prikazuje se greška.
if ($_SERVER["REQUEST_METHOD"] !== "POST"
    || array_diff($requiredPostVars, array_keys($_POST)))
{
    $_SESSION["formErrorMsg"] = LANG["invalidPost"];
    Util::redirect("$failurePage");
}

# Deklariranje sanitiziranih varijabli kraćeg naziva preko pomoćne
# funkcije (npr. umjesto $_POST["email"], pišem samo $email)
foreach ($requiredPostVars as $postVar)
    $$postVar = Util::sanitizeFormData($_POST[$postVar]);

# Validacija redom svih polja pomoću regex funkcije preg_match() i
# regex patterna definiranih kao javne konstante klase UserModel.
# Ako bilo koje polje nije validno, obavlja se redirekcija i prikazuje
# se greška.
if (! preg_match(UserModel::REGEX_VALID_USERNAME, $username)) {
    $_SESSION["formErrorMsg"] = LANG["invalidUsername"];
}
elseif (! preg_match(UserModel::REGEX_VALID_NAME, $firstName)
    || ! preg_match(UserModel::REGEX_VALID_NAME, $lastName))
{
    $_SESSION["formErrorMsg"] = LANG["invalidName"];
}
elseif (! isset($_GET["update"])
    && ! preg_match(UserModel::REGEX_VALID_PASSWORD, $password))
{
    $_SESSION["formErrorMsg"] = LANG["invalidPassword"];
}
elseif (! filter_var($email, FILTER_VALIDATE_EMAIL)
    || ! checkdnsrr(substr($email, strpos($email, "@")+1)))
{
    $_SESSION["formErrorMsg"] = LANG["invalidEmail"];
}
elseif (! isset($_GET["update"])) {
    # Testiranje jesu li korisničko ime ili e-mail adresa već zauzeti.
    $DB = DB::getInstance();
    if ($DB->isTaken("User", "username", $username))
        $_SESSION["formErrorMsg"] = LANG["usernameTakenError"];
    elseif ($DB->isTaken("User", "email", $email))
        $_SESSION["formErrorMsg"] = LANG["emailTakenError"];
}

if (isset($_SESSION["formErrorMsg"])) {
    Util::redirect("$failurePage");
}

```

```

if (isset($_GET["update"])) {
    # Ažuriranje korisničkih podataka.
    $user = UserModel::ctorLoad($_SESSION["userID"]);

    foreach (UserModel::UPDATE_VARS as $updateVar) {
        $user->$updateVar = $$updateVar;
    }

    $errorMsg = $user->update();
}
else {
    # Zapisivanje korisnika u bazu podataka.
    $errorMsg = UserModel::signUp($username, $email, $password,
        $firstName, $lastName);
}

if (isset($errorMsg))
    $_SESSION["formErrorMsg"] = LANG["dbError"] . ": $errorMsg";

if (isset($_SESSION["formErrorMsg"]))
    Util::redirect("$failurePage");
else
    Util::redirect("$successPage");

```

**Izvorni kod 11:** PHP – Obrada obrasca za registraciju.

## 5.6.2. Spremanje korisnika u bazu podataka

```

<?php
# Datoteka: htdocs/app/UserModel.php

# Klasa UserModel sadrži sve atribute i metode vezane za korisnika.
class UserModel
{
    # Korisničko ime: Samo ASCII znakovi, duljine između 4 i 30
    znakova:
    const REGEX_VALID_USERNAME = "/^\\w{4,30}$/";
    # Ime i prezime: bez brojeva, između 3 i 40 znakova:
    const REGEX_VALID_NAME = "/^\\D{3,40}$/";
    # Između 8 i 50 znakova, barem 1 veliko slovo i barem 1 broj:
    const REGEX_VALID_PASSWORD = "/^(?=.*\\d)(?=.*[A-Z]).{8,50}$/";
    const UPDATE_VARS = ["username", "email", "firstName", "lastName",
    ];

    static function signUp(string $username, string $email, string
    $password, string $firstName, string $lastName) : ?string
    {
        $query = "INSERT INTO 'User'('username', 'email', '
        passwordHash', 'firstName', 'lastName')
        VALUES(?, ?, ?, ?, ?)";

        # Uspostavljanje veze s bazom podataka.
        $DB = DB::getInstance();

        # Hashiranje lozinke.
        $passwordHash = password_hash($password, PASSWORD_BCRYPT);

        # Izvršavanje insert upita.
        $errorMsg = $DB->execStmt($query, "sssss", $username, $email,
            $passwordHash, $firstName, $lastName);

        if (gettype($errorMsg) === "string")
            return $errorMsg;
    }
}

```



```

        # Ako je definirana varijabla sesije userID, to znači da je
        korisnik prijavljen u aplikaciju.
        $_SESSION["userID"] = $DB->conn->insert_id;

        return null;
    }
}

```

**Izvorni kod 12:** PHP/SQL – Spremanje korisnika u bazu podataka.

## 5.7. Prijava korisnika

**Slika 11:** Obrazac za prijavu

Korisnik/ca se može prijaviti ili korisničkim imenom ili e-mail adresom. HTML kod i PHP kod za validaciju je nepotrebno pokazivati jer su vrlo slični kodu svakog drugog obrasca u projektu.

Ako je lozinka pogrešna, prikazuje se greška korisniku na dnu obrasca.

```

<?php
class UserModel
{
    # Vraća null ako je prijava uspješna, u suprotnom vraća false.
    static function logIn(string $usernameOrEmail, string $password) :
    string|false|null
    {
        $query = "SELECT 'ID', 'passwordHash'
        FROM 'User'
        WHERE 'username' = ? OR 'email' = ?";

        $DB = DB::getInstance();

        $result = $DB->execStmt($query, "ss", $usernameOrEmail,
        $usernameOrEmail);
        if (gettype($result) === "string")
            return $result; # Greška baze podataka.

        $resultRow = $result->fetch_assoc();
    }
}

```

```

        $ID = $resultRow["ID"];
        $passwordHash = $resultRow["passwordHash"];

        # Autentikacija - uspoređuje se hashirana lozinka iz baze s
        hashiranom lozinkom podnesenom u obrascu za prijavu.
        if (! $ID || ! password_verify($password, $passwordHash))
            return false;

        # Spremanjem ID-a korisnika u varijablu sesije, aplikacija zna
        da je korisnik/ca s tim ID-jem prijavljen/a.
        $_SESSION["userID"] = $ID;

        # Ažuriranje stupca datuma i vremena zadnje prijave.
        $query = "UPDATE 'User'
                SET 'lastLoginTime' = utc_timestamp()
                WHERE 'ID' = ?";

        $DB->execStmt($query, "i", $_SESSION["userID"]);

        return null;
    }
}

```

**Izvorni kod 13:** PHP – Prijava korisnika

## 5.8. Profil korisnika



**Slika 12:** Stranica s profilom korisnika.

Na ovoj stranici nalaze se detalji korisničkog računa. Također, ako je korisnik/ca prijavljen/a, prikazuju se gumbi za odjavu, uređivanje detalja profila i brisanje samog profila.

Klikom na gumb za odjavu uništava se sesija i preusmjerava se na obrazac za prijavu.

```

<?php
// Datoteka: htdocs/app/forms/log_out.php

require_once "config.php";
session_start();
session_destroy();
Util::redirect("/views/log_in.phtml");

```

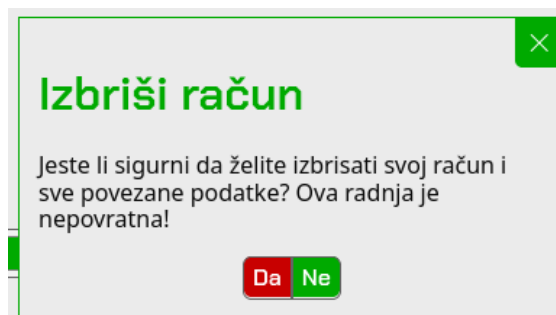
**Izvorni kod 14:** PHP – Odjava korisnika.

Klikom na gumb za brisanje računa otvara se skočni modalni prozor (engl. *overlay*) za potvrdu brisanja. Ako korisnik potvrdi, briše se račun i svi dokumenti korisnika.

```
<?php
class UserModel
{
    function delete() : void
    {
        $query = "DELETE from 'User'
                WHERE 'ID' = ?";

        $DB = DB::getInstance();
        $DB->execStmt($query, "i", $this->ID);
    }
}
```

**Izvorni kod 15:** PHP/SQL – Brisanje korisnika.



**Slika 13:** Skočni modalni prozor za potvrdu brisanja računa.

Klikom na gumb za uređivanje profila, otvara se isti obrazac kao za registraciju korisnika (osim polja za lozinku) – HTML struktura i kod za validaciju se reciklira. Razlika je u tome što se ovaj obrazac prikazuje u *overlayu*. Polja se popune s vrijednostima u bazi podataka.

## 5.9. Početna stranica (*front-end*)

### 5.9.1. Popis dokumenata

Na početnoj stranici su tabelarno prikazani svi dokumenti korisnika te njihovi detalji. Klikom na poveznicu u prvom stupcu tablice preusmjerava na stranicu detalja dokumenata, gdje su prikazani svi detalji dokumenta. Prelaskom miša (engl. *hover*) preko datuma roka (ako je definiran) prikazuje se u malom prozoru (engl. *tooltip*) *relativan* datum.

×

## Uredi profil

Korisničko ime

ivo

E-mail

admin@gmail.com

Ime

Ivan

Prezime

Dolovčak

Uredi profil

Koristite velika i mala slova bez dijakritičkih znakova, brojeve i donje crte, duljine najmanje 4 znakova.

Slika 14: Skočni modalni prozor za uređivanje profila.

Dokumenti			
Naziv	Tip	Vidljivost	Rok predaje
<a href="#">prvi ispit</a>	ispit	javni	14. svibnja 2024. u 12:00
<a href="#">moj obrazac</a>	obrazac	skriveni	N/A
<a href="#">drugi ispit</a>	ispit	javni	30. travnja 2024. u 11:10
<div>⊕ Napravi novi dokument</div>			

Documents			
Name	Type	Visibility	Deadline
<a href="#">prvi ispit</a>	exam	public	May 14, 2024 at 12:00 PM
<a href="#">moj obrazac</a>	form	unlisted	N/A
<a href="#">drugi ispit</a>	exam	public	April 30, 2024 at 11:10 AM
<div>⊕ Create new document</div>			

Slika 15: Popis dokumenata; višejezičnost.

## 6. Zaključak

---

Elaborat završnog rada dokumentira arhitekturu i programsku izvedbu web aplikacije za kreiranje i rješavanje ispita i obrazaca. Korisnik/ca može nakon prijave kreirati svoje dokumente i rješavati tuđe. Korisnik/ca vidi popise vlastitih i tuđih dokumenata i rješenja tih dokumenata. Moguće je uređivanje detalja i brisanje vlastitog profile i vlastitih dokumenata. Korisnik/ca može prilagoditi sučelje aplikacije svojim potrebama. Sučelje aplikacije je intuitivno i pristupačno.

Prilikom izvedbe web aplikacije korišteni su moderni JavaScript API-ji: Fetch, DOM i Intl. PHP nudi MySQLi API za upravljanje bazom podataka. U PHP-u sam također koristio kolačiće i varijable sesije. I JavaScript i PHP kod pretežito koriste OOP model programiranja.

Zahvaljujući modularnosti koda i kvalitetnim komentarima, aplikaciju je jednostavno proširiti. U budućnosti bih preveo aplikaciju na više jezika, dodao prijavu pomoću Google API-ja, dodao još različitih tipova pitanja te do-tjerao *front-end* da bude još profesionalnijeg izgleda.

Nastojao sam pisati dobro organiziran, siguran, moderan, kvalitetno dokumentiran, efikasan i modularan izvorni kod – koliko je god bilo moguće. Ovo mi je najopširniji programerski projekt do sad. Naučio sam jako puno korisnih stvari vezanih uz struku i predmet te uveliko proširio znanje i vještine stečene u školi.

## Popis slika

---

1.	VSCodium uređivač koda s otvorenim projektom. . . . .	5
2.	kitty terminal s više otvorenih kartica i panela. . . . .	6
3.	Alati za web programere, preglednik Mozilla Firefox. . . . .	7
4.	Front-end jezici. . . . .	8
5.	Back-end programski jezici i programska podrška. . . . .	9
6.	Dijagram baze podataka prikazan u <i>phpMyAdminu</i> . . . . .	10
7.	Početna stranica s popisom dokumenata. . . . .	16
8.	Navigator kad korisnik/ca nije prijavljen/a. . . . .	17
9.	Prilagodljivost sučelja (profile.phtml) . . . . .	17
10.	Obrazac za registraciju – validacija u realnom vremenu . . . . .	19
11.	Obrazac za prijavu . . . . .	25
12.	Stranica s profilom korisnika. . . . .	26
13.	Skočni modalni prozor za potvrdu brisanja računa. . . . .	27
14.	Skočni modalni prozor za uređivanje profila. . . . .	28
15.	Popis dokumenata; višejezičnost. . . . .	28

## Popis izvornih kodova

---

1.	SQL – Izrada MariaDB korisnika i baze podataka. . . . .	11
2.	SQL – Izrada i povezivanje tablica. . . . .	12
3.	PHP – Osnovna konfiguracija . . . . .	13
4.	PHP – Util klasa . . . . .	14
5.	PHP – DB klasa . . . . .	14
6.	PHP – Engleski znakovni nizovi . . . . .	16
7.	PHP – Hrvatski znakovni nizovi . . . . .	16
8.	PHP – Preferences klasa . . . . .	18
9.	HTML/PHP – Obrazac za registraciju. . . . .	20
10.	JS – Live validacija forme. . . . .	21
11.	PHP – Obrada obrasca za registraciju. . . . .	22
12.	PHP/SQL – Spremanje korisnika u bazu podataka. . . . .	24
13.	PHP – Prijava korisnika . . . . .	25
14.	PHP – Odjava korisnika. . . . .	26
15.	PHP/SQL – Brisanje korisnika. . . . .	27

## Literatura

---

- *Stack Overflow* - web sjedište sa pitanjima i odgovorima za profesionalne programere: <https://stackoverflow.com/>
- *PHP* priručnik: <https://www.php.net/manual/en/>
- *MDN Web Docs* - priručnik za *HTML*, *CSS* i *JavaScript*: <https://developer.mozilla.org/en-US/>
- *MariaDB SQL* dokumentacija: <https://mariadb.com/kb/en/documentation/>
- *Apache* dokumentacija: <https://httpd.apache.org/docs/2.4/>



**Konzultacije s mentorom:**

RB	Sadržaj (bilješke o napredovanju)	Potpis mentora	Datum
1.	Mentor prihvatio predloženu temu rada		27.10.2023.
2.	Poslana poveznica za GitHub repozitorij		15.01.2024.
3.	Konfiguracija hostinga		23.02.2024.
4.	Savjeti za pisanje elaborata		25.04.2024.
5.	Slanje prve inačice elaborata mentoru na uvid		01.05.2024.

Datum predaje rada: \_\_\_\_\_

Potpis mentora: \_\_\_\_\_ (mentor je prihvatio izradbu)

Ocjena pisanog rada: \_\_\_\_\_

Datum obrane rada: \_\_\_\_\_

Ocjena obrane rada: \_\_\_\_\_

**Konačna ocjena:** \_\_\_\_\_**Povjerenstvo:**

1. mentor: \_\_\_\_\_

2. \_\_\_\_\_

3. \_\_\_\_\_

4. \_\_\_\_\_

5. \_\_\_\_\_

**Komentar:**