# Baza podataka i web-aplikacija za praćenje treninga u teretani

**Pandža, Petar**

**Undergraduate thesis / Završni rad**

**2024**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:168:763317

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2025-03-22**

*Repository / Repozitorij:*

FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repozitory


SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 1464

# BAZA PODATAKA I WEB-APLIKACIJA ZA PRAĆENJE TRENINGA U TERETANI

Petar Pandža

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 1464

# BAZA PODATAKA I WEB-APLIKACIJA ZA PRAĆENJE TRENINGA U TERETANI

Petar Pandža

Zagreb, lipanj 2024.

**SVEUČILIŠTE U ZAGREBU**
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Zagreb, 4. ožujka 2024.

# ZAVRŠNI ZADATAK br. 1464

| | |
|---|---|
| Pristupnik: | **Petar Pandža (0036543663)** |
| Studij: | Elektrotehnika i informacijska tehnologija i Računarstvo |
| Modul: | Računarstvo |
| Mentor: | prof. dr. sc. Boris Vrdoljak |

Zadatak: **Baza podataka i web-aplikacija za praćenje treninga u teretani**

Opis zadatka:

U sklopu ovog završnog rada potrebno je izraditi bazu podataka u kojoj će se pohranjivati podaci o korisnicima, treninzima i vježbama u teretani. Nakon oblikovanja modela entiteti-veze i odgovarajućeg relacijskog modela baze podataka, treba implementirati bazu podataka koristeći sustav za upravljanje bazom podataka SQLite3. Potrebno je zatim korištenjem radnog okvira Spring Boot napraviti web-aplikaciju koja omogućava pregled, unos, brisanje i izmjenu podataka. Web-aplikacija treba prijavljenom korisniku omogućiti unos novih treninga te pregled odrađenih treninga.

Rok za predaju rada: 14. lipnja 2024.

# Sadržaj

# Uvod

U današnjem digitalnom dobu sportaši sve češće koriste pametne uređaje za čuvanje podataka o treniranju. U tom kontekstu, ovaj završni rad bavi se razvojem aplikacije za praćenje korisničkih treninga u teretani koristeći suvremene tehnologije i pristupe u razvoju softvera. Detaljno praćenje treninga osigurava optimalno treniranje i brži napredak.

Aplikacija za praćenje treninga omogućava jednostavno pregledavanje, dodavanje, uređivanje i brisanje informacija, omogućava pregled bilo kojeg treninga u kratkom vremenu i osigurava trajnost podataka.

U ovom radu za čuvanje podataka koristiti će se SQLite lokalna baza podataka te Hibernate za lakši pristup i manipulaciju podataka. Za izradu web aplikacije koristiti će se programski jezik Java i bogato radno okruženje Spring Boot. Dodatne korištene tehnologije su: Jakarta EE, JSP, JSTL, CSS, JavaScript

Prva dva poglavlja rada opisuju modeliranje baze podataka, pretvaranje u relacijski model, opis podataka te implementacija u kodu. Treće poglavlje opisuje implementaciju web aplikacije, opis tehnologija i postupke zaštite podataka. Četvrto poglavlje prikazuje izgled aplikacije u upotrebi.

# 1. Modeliranje baze podataka

## 1.1. Odabir modela

Za model baze podataka odabran je ER (Entity-Relationship) model zbog svoje jednostavnosti i učinkovitosti u opisu entiteta, veza između njih te njihovih atributa. ER model omogućuje jasno definiranje strukture podataka, što olakšava razumijevanje i planiranje baze podataka. Kroz ovaj model, mogu se detaljno predstaviti entiteti poput korisnika, treninga i vježbi te njihove međusobne veze, kao što su pripadanje korisniku ili uključenost u određeni trening.

Prednosti ER modela uključuju njegovu fleksibilnost u opisu kompleksnih podataka, što omogućuje lakšu prilagodbu promjenama zahtjeva. Također, ER model olakšava komunikaciju između timova u razvoju softvera jer pruža jasan i općeprihvaćen način opisivanja strukture podataka.

ER model također olakšava implementaciju baze podataka kroz konceptualizaciju prije implementacije, što pomaže u izbjegavanju nepotrebnih problema u daljnjem razvoju. Sve te činjenice čine ER model izvrsnim odabirom za dizajniranje baze podataka za teretanu.

ER model se temelji na konceptu entiteta, veza i atributa.

1. Entiteti predstavljaju objekte ili pojmove o kojima želimo pohraniti podatke. Svaki entitet u bazi podataka opisan je skupom atributa koji definiraju karakteristike tog entiteta. Primjeri entiteta u bazi podataka teretane mogu biti korisnici, treninzi, vježbe i serije.

2. Veze definiraju kako su entiteti povezani i kako međusobno djeluju. Veze mogu imati različite spojenosti. Primjerice, veza između korisnika i treninga može biti 1:N, jer svaki korisnik može imati više odrađenih treninga, dok se svaki trening odnosi na samo jednog korisnika.

3. Atributi su svojstva entiteta koja opisuju osobine tog entiteta. Svaki entitet ima skup atributa koji definiraju podatke koje pohranjujemo o tom entitetu. Na primjer, entitet "korisnik" može imati atribute kao što su ime, e-mail adresa, spol itd.

## 1.2. Opis prirode podataka

Glavna svrha aplikacije je praćenje treninga za korisnika.

Za svakog korisnika prati se korisničko ime, e-mail adresa, lozinka u raspršenom obliku, salt za raspršivanje, spol, jedinstveni identifikator te je li korisnik administrator. Svaki korisnik za sebe ima zabilježen proizvoljan broj treninga, dok svaki trening pripada samo jednom korisniku. Svaki korisnik za svaku pojedinu vježbu može vidjeti svoje najbolje rezultate na njoj. Svaki korisnik može imati proizvoljan broj stvorenih predložaka, dok se svaki predložak odnosi na samo jednog korisnika.

Za svaki trening prati se vrijeme početka, trajanje u minutama, naziv koji je dao korisnik, jedinstveni identifikator te odrađene vježbe i serije tijekom treninga. Svaki trening se mora sastojati od barem jedne serije jedne vježbe.

Za svaku vježbu prati se ime, opis, jedinstveni identifikator te vrsta. Korisnik za svaku vježbu i broj ponavljanja vježbe ima zabilježenu najveću dignutu kilažu te kad je postignuta. Vrste vježbi mogu biti: vježba sa tjelesnom težinom, vježba sa slobodnim utezima, vježba sa šipkom ili vježba na spravi.

Za svaku seriju prati se vježba koja se izvodila, broj ponavljanja, broj kilograma s kojim se vježba izvodila te vrsta serije. Vrste serije mogu biti: obična serija, serija smanjenja kilaže(drop set), serija do neuspjeha ili super serija. Svaka serija odnosi se na jednu vježbu odrađenu u jednom treningu.

Svaki predložak treninga sastoji se od imena, jedinstvenog identifikatora te barem jedne vježbe.

## 1.3. ER model

Slika 1.1 grafički prikazuje ER model.

### 1.3.1. Entiteti

Popis entiteta s pripadajućim atributima (primarni ključevi podvučeni):

**Korisnik** – idKorisnik, korisničkoIme, hashLozinka, saltLozinka, email, spol, jeAdmin

**Trening** – idTrening, imeTrening, vrijemePočetka, trajanje

**Vježba** – idVježba, imeVježba, opisVježba

**VrstaVježbe** – idVrstaVježbe, imeVrstaVježbe

**Serija** – redniBrojSerije, idTrening, idVježba, brPonSerija, brKgSerija

**VrstaSerije** – idVrstaSerije, imeVrstaSerije

**PredložakTreninga** – idPredložak, imePredložak

### 1.3.2. Veze

Popis veza s pripadajućima tributima (primarni ključevi podvučeni):

**odradioTrening** – idTrening, idKorisnik

**najboljiRezultat** – idKorisnik, idVježba, najboljiBrojPon, najboljiBrKg, datumPostignuća

**imaPredložak** – idPredložak, idKorisnik

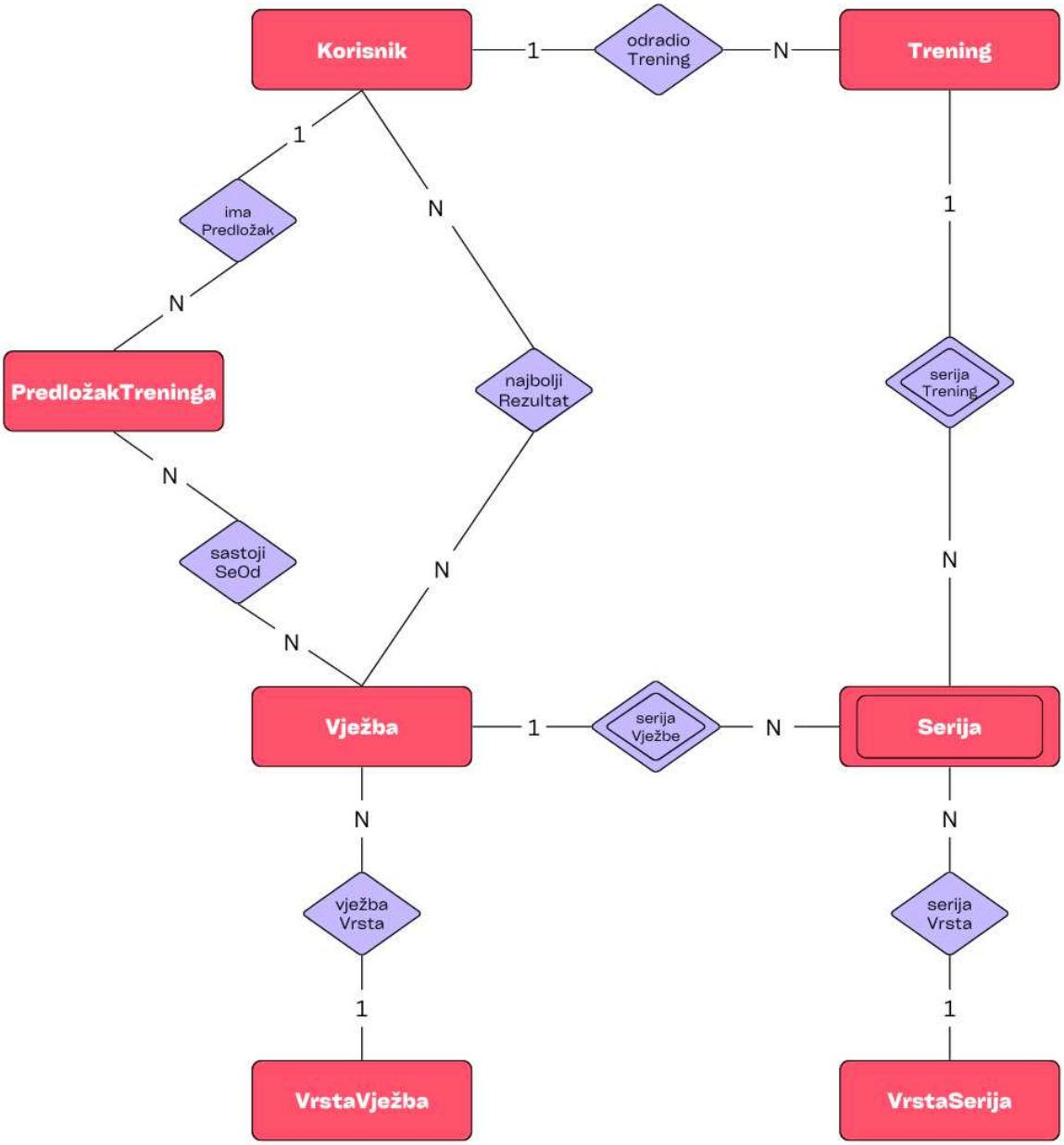
**sastojiSeOd** – idPredložak, idVježba

**serijaTreninga** – idSerija, idVježba, idTrening

**serijaVježbe** – idSerija, idTrening, idVježba

**vježbaVrsta** – idVježba, idVježbaVrsta

**serijaVrsta** – idSerija, idSerijaVrsta




Slika 1.1 ER Model

## 1.4. Relacijski model

U izradi baze podataka, ER model služi kao početna faza dizajna koja se koristi za konceptualno modeliranje podataka. Međutim, da bi baza podataka bila funkcionalna u praksi, ER model potrebno je pretvoriti u relacijski model.

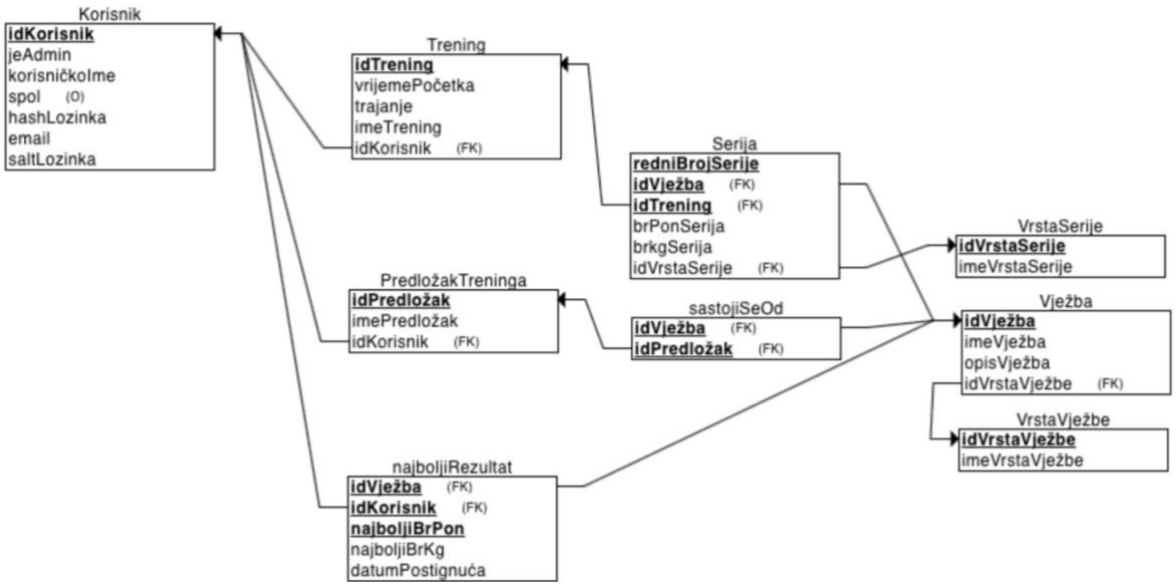
Relacijski model je logički model koji se temelji na tablicama i koristi se u sustavima za upravljanje relacijskim bazama podataka.

### 1.4.1. Pretvaranje iz ER modela u relacijski

Pretvaranje ER modela u relacijski odvija se kroz nekoliko koraka:

1. Svaki entitet u ER modelu postaje tablica u relacijskom modelu.
2. Veze između tablica se pretvaraju u strane ključeve ili nove tablice uzimajući u obzir nekoliko pravila:
    a. 1:1 veza – Jedna od tablica dobije strani ključ na drugu.
    b. 1:N veza - Strani ključ se dodaje tablici koja odgovara entitetu na strani N.
    c. N:N - Stvara se nova tablica koja sadrži strane ključeve oba entiteta.
3. Svaki atribut u entitetima mapira se u stupce tablica, gdje se definira tip podataka za svaki stupac, osiguravajući da domena atributa ostane konzistentna.

Primjenom ovih pravila stvaramo relacijski model, čiji dijagram možemo vidjeti na slici 1.2. Model je napravljen u programu ERDPlus[7].




Slika 1.2 Dijagram relacijskog modela

### 1.4.2. Tablice

Zbog ograničenosti tipova podataka u bazi podataka SQLite (poglavlje 2.1.1.), tipovi koji bi predstavljali binarnu vrijednosti prikazani su INTEGER tipom podataka, dok su tipovi koji predstavljaju vremensku vrijednost prikazani TEXT tipom podataka.

**Korisnik**

| Atribut | Tip Podatka | Može biti NULL? | Primarni Ključ? | Strani Ključ? |
|---|---|---|---|---|
| idKorisnik | INTEGER | | DA | |
| email | TEXT | | ALT. | |
| korisničkoIme | TEXT | | | |
| jeAdmin | INTEGER | | | |
| spol | INTEGER | DA | | |
| hashLozinka | TEXT | | | |
| saltLozinka | TEXT | | | |

**Trening**

| Atribut | Tip Podatka | Može biti NULL? | Primarni Ključ? | Strani Ključ? |
|---|---|---|---|---|
| idTrening | INTEGER | | DA | |
| imeTrening | TEXT | | | |
| vrijemePočetka | TEXT | | | |
| trajanje | INTEGER | | | |
| idKorisnik | INTEGER | | | Korisnik (idKorisnik) |

**VrstaVježbe**

| Atribut | Tip Podatka | Može biti NULL? | Primarni Ključ? | Strani Ključ? |
|---|---|---|---|---|
| idVrstaVježbe | INTEGER | | DA | |
| imeVrstaVježbe | TEXT | | | |

**VrstaSerije**

| Atribut | Tip Podatka | Može biti NULL? | Primarni Ključ? | Strani Ključ? |
|---|---|---|---|---|
| idVrstaSerije | INTEGER | | DA | |
| imeVrstaSerije | TEXT | | | |

**PredložakTreninga**

| Atribut | Tip Podatka | Može biti NULL? | Primarni Ključ? | Strani Ključ? |
|---|---|---|---|---|
| idPredložak | INTEGER | | DA | |

| | | | | |
|---|---|---|---|---|
| imePredložak | TEXT | | | |
| idKorisnik | INTEGER | | | Korisnik (idKorisnik) |

**Vježba**

| Atribut | Tip Podatka | Može biti NULL? | Primarni Ključ? | Strani Ključ? |
|---|---|---|---|---|
| idVježba | INTEGER | | DA | |
| imeVježba | TEXT | | | |
| opisVježba | TEXT | | | |
| idVrstaVježbe | | | | VrstaVježbe (idVrstaVježbe) |

**Serija**

| Atribut | Tip Podatka | Može biti NULL? | Primarni Ključ? | Strani Ključ? |
|---|---|---|---|---|
| redniBrojSerije | INTEGER | | DA | |
| idVježba | INTEGER | | DA | Vježba (idVježba) |
| idTrening | INTEGER | | DA | Trening (idTrening) |
| brPonSerija | INTEGER | | | |
| brKgSerija | REAL | | | |
| idVrstaSerije | INTEGER | | | VrstaSerije (idVrstaSerije) |

**najboljiRezultat**

| Atribut | Tip Podatka | Može biti NULL? | Primarni Ključ? | Strani Ključ? |
|---|---|---|---|---|
| najboljiBrPon | INTEGER | | DA | |
| idVježba | INTEGER | | DA | Vježba (idVježba) |
| idKorisnik | INTEGER | | DA | Korisnik (idKorisnik) |
| najboljiBrKg | REAL | | | |
| datumPostignuća | TEXT | | | |

**sastojiSeOd**

| Atribut | Tip Podatka | Može biti NULL? | Primarni Ključ? | Strani Ključ? |
|---|---|---|---|---|
| idVježba | INTEGER | | DA | Vježba (idVježba) |
| idPredložak | INTEGER | | DA | Predložak Treninga (idPredložak) |

# 2.Implementacija baze podataka

## 2.1. SQLite

Za ovaj projekt korištena je SQLite baza podataka.

SQLite je lagana baza podataka koja se često koristi u aplikacijama gdje je potrebno pohraniti podatke lokalno, bez potrebe za kompleksnim mrežnim konfiguracijama. Za razliku od tradicionalnih sustava za upravljanje bazama podataka (DBMS) poput MySQL-a, PostgreSQL-a ili Oracle-a, SQLite je "embeddable" baza podataka. To znači da se baza podataka integrira izravno u aplikaciju, bez potrebe za zasebnim serverom.

**Prednosti SQLite-a:**

- **Jednostavnost:** Instalacija i korištenje SQLite-a su jednostavni. Dovoljno je uključiti SQLite biblioteku u projekt i baza podataka spremna je za korištenje.
- **Performanse:** SQLite optimiziran je za brzinu i učinkovitost. Zbog svoje male veličine i minimalnih zahtjeva, često je brži od mnogih drugih baza podataka u situacijama gdje se koristi.
- **Portabilnost:** SQLite datoteka baze podataka obična je datoteka koja se može lako premještati između različitih sustava.

**Razlike između SQLite-a i drugih baza podataka:**

- **Arhitektura:** SQLite integriran je u aplikaciju kao biblioteka, dok su druge baze podataka obično bazirane na serveru, što znači da aplikacija komunicira s bazom podataka putem mreže.
- **Korisničke dozvole:** SQLite nema kompleksan sustav korisničkih dozvola kao što je to slučaj kod nekih drugih baza podataka.
- **Veličina baze:** SQLite pogodan je za manje do srednje velike baze podataka. Kod vrlo velikih baza rješenja bazirana na serverima mogu biti učinkovitija.

### 2.1.1. Tipovi podataka u SQLite

SQLite ima specifičan pristup tipovima podataka, koji se razlikuje od tradicionalne relacijske baza podataka poput PostgreSQL-a. Tipovi podataka su manje striktni, što omogućava veću fleksibilnost, ali može dovesti do nekih nesuglasica prilikom migracije podataka između različitih sustava.

SQLite koristi dinamičko tipiziranje, što znači da tipovi podataka nisu strogo vezani uz stupce, već uz same vrijednosti. Postoji pet osnovnih tipova podataka koje SQLite prepoznaje:

- **INTEGER:** Cijeli brojevi.
- **REAL:** Decimalni brojevi.
- **TEXT:** Tekstualni podaci.
- **NULL:** Vrijednost je NULL.
- **BLOB:** Binary Large Object (binarnim podacima koji su pohranjeni točno onako kako su uneseni).

### 2.1.2. Upiti za stvaranje tablica

SQL upiti za stvaranje tablica u dijalektu SQLite:

```
CREATE TABLE Korisnik
(
    jeAdmin INTEGER NOT NULL,
    korisnickoIme TEXT NOT NULL,
    spol INTEGER,
    idKorisnik INTEGER PRIMARY KEY AUTOINCREMENT,
    hashLozinka TEXT NOT NULL,
    email TEXT NOT NULL UNIQUE,
    saltLozinka TEXT NOT NULL
);

CREATE TABLE Trening
(
    vrijemePocetka TEXT NOT NULL,
    trajanje INTEGER NOT NULL,
    imeTrening TEXT NOT NULL,
    idTrening INTEGER PRIMARY KEY AUTOINCREMENT,
    idKorisnik INTEGER NOT NULL,
    FOREIGN KEY (idKorisnik) REFERENCES Korisnik(idKorisnik)
);
```

```sql
CREATE TABLE VrstaVjezbe
(
    idVrstaVjezbe INTEGER PRIMARY KEY AUTOINCREMENT,
    imeVrstaVjezbe TEXT NOT NULL
);

CREATE TABLE VrstaSerije
(
    idVrstaSerije INTEGER PRIMARY KEY,
    imeVrstaSerije TEXT NOT NULL
);

CREATE TABLE PredlozakTreninga
(
    idPredlozak INTEGER PRIMARY KEY AUTOINCREMENT,
    imePredlozak TEXT NOT NULL,
    idKorisnik INTEGER NOT NULL,
    FOREIGN KEY (idKorisnik) REFERENCES Korisnik(idKorisnik)
);

CREATE TABLE Vjezba
(
    imeVjezba TEXT NOT NULL,
    idVjezba INTEGER PRIMARY KEY AUTOINCREMENT,
    opisVjezba TEXT NOT NULL,
    idVrstaVjezbe INTEGER NOT NULL,
    FOREIGN KEY (idVrstaVjezbe) REFERENCES
        VrstaVjezbe(idVrstaVjezbe)
);

CREATE TABLE Serija
(
    brPonSerija INTEGER NOT NULL,
    brkgSerija REAL NOT NULL,
    redniBrojSerije INTEGER NOT NULL,
    idVjezba INTEGER NOT NULL,
    idTrening INTEGER NOT NULL,
    idVrstaSerije INTEGER NOT NULL,
    PRIMARY KEY (redniBrojSerije, idVjezba, idTrening),
    FOREIGN KEY (idVjezba) REFERENCES Vjezba(idVjezba),
    FOREIGN KEY (idTrening) REFERENCES Trening(idTrening),
    FOREIGN KEY (idVrstaSerije) REFERENCES
        VrstaSerije(idVrstaSerije)
);

CREATE TABLE najboljiRezultat
(
    najboljiBrKg REAL NOT NULL,
    datumPostignuca TEXT NOT NULL,
    najboljiBrPon INTEGER NOT NULL,
    idVjezba INTEGER NOT NULL,
    idKorisnik INTEGER NOT NULL,
    PRIMARY KEY (najboljiBrPon, idVjezba, idKorisnik),
    FOREIGN KEY (idVjezba) REFERENCES Vjezba(idVjezba),
    FOREIGN KEY (idKorisnik) REFERENCES Korisnik(idKorisnik)
);

CREATE TABLE sastojiSeOd
(
    idVjezba INTEGER NOT NULL,
    idPredlozak INTEGER NOT NULL,
    PRIMARY KEY (idVjezba, idPredlozak),
    FOREIGN KEY (idVjezba) REFERENCES Vjezba(idVjezba),
    FOREIGN KEY (idPredlozak) REFERENCES
        PredlozakTreninga(idPredlozak)
);
```

## 2.2. ORM (Object-Relational Mapping)

ORM (Object-Relational Mapping) je tehnika koja omogućuje rad s podacima pohranjenim u relacijskoj bazi podataka koristeći objektno-orijentirani pristup. Umjesto spajanja direktno na bazu podatka, podacima se pristupa preko sučelja. ORM mapira tablice u bazi podataka na razrede u programskom jeziku, a redove iz tih tablica na instance tih razreda.

ORM ima prednost mogućeg lakšeg razvoja aplikacija. Programeri su spašeni pisanja dugih i složenih SQL upita.

Hibernate, koji je korišten za ovaj projekt, je primjer ORM-a napisan za programski jezik Java.

## 2.3. Hibernate

Hibernate je implementacija ORM-a za Javu koji olakšava rad s relacijskim bazama podataka koristeći objektno-orijentirane principe. Hibernate djeluje kao most između aplikacijskog koda i baze podataka.

Glavna svrha Hibernate-a je apstrahirati složene SQL operacije te omogućiti rad s bazom podataka na objektno-orijentirani način.

Podaci se mogu dohvatiti na dva načina:

1. Za jednostavne operacije poput dohvaćanja objekta preko primarnog ključa već postoje ugrađene metode. Primjer dohvaćanja korisnika se može vidjeti na slici 2.1

```
@Override
public User getUserById(int id) {
    return JPAEMProvider.getEntityManager().find(User.class, id);
}
```

Slika 2.1 Kod za dohvat korisnika preko primarnog ključa

2. Upotreba posebnog upitnog jezika HQL (Hibernate Query Language). HQL, za razliku od SQL-a, upite izvršava na entitetima, a ne na tablicama. Uz to, omogućuje entitetu da posjeduje listu drugih objekata u slučaju veze 1:N. Primjer HQL upita koji dohvaća sve korisnike koji su odradili barem 5 treninga je na slici 2.2

```
public List<User> getUsersWith5Sessions() {
    return JPAEMProvider.getEntityManager().createQuery(
        "SELECT u FROM User u WHERE SIZE(u.trainingSessions) >= 5", User.class)
        .getResultList();
}
```

Slika 2.2 Kod za dohvat svih korisnika s barem 5 treninga

Oba ova primjera prikazuju dohvat objekta/objekata tipa User. Da bi Hibernate znao vezu između razreda User i tablice Korisnik u bazi moramo ga konfigurirati. Dva su moguća rješenja. Prvo rješenje je napraviti XML datoteku koja opisuje samu vezu. Drugo rješenje, koje je korišteno u ovom projektu, su anotacije koje postavimo u razredu. Slika 2.3 prikazuje razred User i sve anotacije koje su potrebne.

Dodavanje novih podataka u bazu podataka je izuzetno jednostavno i intuitivno koristeći Hibernate. Hibernate omogućuje instanciranje objekta preko konstruktora, zadavanje varijabli te pozivanje metode save. Slika 2.4 prikazuje kod korišten za dodavanje svih 4 vrsta serija u bazu podataka.

```java
@Entity
@Table(name = "Vjezba")
public class Exercise {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "idVjezba")
    private Integer exerciseId;

    @Column(name = "imeVjezba", nullable = false)
    private String name;

    @Column(name = "opisVjezba", nullable = false)
    private String description;

    @ManyToOne
    @JoinColumn(name = "idVrstaVjezbe", nullable = false)
    private ExerciseType exerciseType;

    public Exercise() {
    }

    public Exercise(String name, String description, ExerciseType exerciseType) {
        this.name = name;
        this.description = description;
        this.exerciseType = exerciseType;
    }

    // getters and setters
}
```

Slika 2.3 Razred User

```java
SetType regular = new SetType(1, "Regular set");
dao.save(regular);

SetType dropSet = new SetType(2, "Drop set");
dao.save(dropSet);

SetType superset = new SetType(3, "Superset");
dao.save(superset);

SetType toFailure = new SetType(4, "To failure");
dao.save(toFailure);
```

Slika 2.4 Kod za dodavanje vrsta serija u bazu podatka.

## 2.4. DAO

Oblikovni obrazac DAO (Data Access Object) koristi se za apstrahiranje i enkapsuliranje svih pristupa podacima. DAO obrazac omogućava odvajanje poslovne logike od sloja pristupa podacima, čime se postiže veća modularnost i lakša promjena implementacije pristupa podacima bez utjecaja na ostatak aplikacije.

DAO obrazac uključuje sljedeće komponente:

- **DAO sučelje:** Deklarira metode za pristup podacima koje će implementirati konkretni razred. Izgled dijela sučelja DAO može se vidjeti na slici 2.5
- **DAO implementacija:** Implementira metode deklarirane u DAO sučelju koristeći specifične tehnologije pristupa podacima. U ovom projektu to je tehnologija JPA (Java Persistance API), i razred JPADAO. Izgled implementiranih metoda iz slike 2.5 vidljiv je na slici 2.6
- **DAO provider:** Ova komponenta upravlja pružanjem konkretnih DAO objekata. Primjer je oblikovnog obrasca singleton. Razred DAOProvider vidljiv je na slici 2.7

```java
/**
 * Interface that defines the methods for data access, modification, and deletion.
 */
public interface DAO {

    /**
     * Saves the object to the database.
     * @param o Object to be saved
     */
    void save(Object o);

    /**
     * Deletes the training session from the database.
     * @param trainingSession Training session to be deleted
     */
    void deleteTrainingSession(TrainingSession trainingSession);

    // and so much more...
}
```

Slika 2.5 Nekoliko metoda iz sučelja DAO

```java
/**
 * Class implements DAO interface and provides methods for saving,
 * deleting and getting data from database.
 * It uses JPA for communication with database.
 */
public class JPADAO implements DAO {

    @Override
    public void save(Object o) {
        JPAEMProvider.getEntityManager().persist(o);
    }

    @Override
    public void deleteTrainingSession(TrainingSession trainingSession) {
        deleteSets(trainingSession.getSets());
        JPAEMProvider.getEntityManager().createQuery(
                "DELETE FROM TrainingSession t WHERE t = :trainingSession")
                .setParameter("trainingSession", trainingSession)
                .executeUpdate();
    }

    // and a few more...

}
```

Slika 2.6 Nekoliko implementiranih metoda iz sučelja DAO

```java
/**
 * Singleton class that provides the DAO object.
 */
public class DAOProvider {
    private static final DAO dao = new JPADAO();

    public static DAO getDAO() {
        return dao;
    }
}
```

Slika 2.7 Razred DAOProvider

# 3.Implementacija web aplikacije

## 3.1. Maven

Maven je alat za upravljanje projektima i automatizaciju gradnje u Java ekosustavu. Omogućava učinkovito upravljanje ovisnostima korištenjem konfiguracijske datoteke pom.xml. Olakšava prevođenje, pakiranje i izvođenje projekta. U svom središnjem repozitoriju sadrži veliku količinu plugina. U ovoj aplikaciji Maven je korišten za upravljanje ovisnostima i automatizaciju gradnje, što je olakšalo razvoj i održavanje koda.

## 3.2. Model-View-Controller (MVC) arhitektura

MVC je obrazac dizajna koji se koristi za razvoj korisničkih sučelja podjelom aplikacije na tri međusobno povezane komponente: model, pogled i kontroler. Ovaj obrazac pomaže u organizaciji koda, poboljšava održavanje i olakšava razvoj složenih aplikacija.

- Model

Model predstavlja podatke aplikacije i logiku poslovanja. On je odgovoran za dohvat, ažuriranje i pohranu podataka. U ovoj aplikaciji model je implementiran koristeći Hibernate i JPA, omogućujući objektno-relacijsko mapiranje (ORM) između objekata u aplikaciji i zapisa u bazi podataka.

- Pogled

Pogled je odgovoran za prikaz podataka korisnicima. Pogled preuzima podatke iz modela i prikazuje ih korisniku na pregledan i interaktivan način. Kada korisnik interaktira sa pogledom, pogled tu informaciju šalje kontroleru. U ovoj aplikaciji pogled je implementiran koristeći JSP, JSTL, CSS i JavaScript

- Kontroler

Kontroler je posrednik između modela i pogleda. On obrađuje korisničke zahtjeve, manipulira modelom i vraća odgovarajuće poglede. U ovoj aplikaciji kontroler je implementiran koristeći Spring Bootove kontrolere.

## 3.3. Backend

### 3.3.1. Spring Boot

Spring Boot je okvir za brzu izradu samostalnih, proizvodno spremnih aplikacija koje koriste Spring radno okruženje. Pruža unaprijed konfigurirane postavke i olakšava pokretanje i razvoj aplikacija. Glavne prednosti korištenja Spring Boota su:

- Automatska konfiguracija: Spring Boot automatski konfigurira aplikaciju prema ovisnostima koje su uključene u projekt.
- Ugrađeni serveri: Aplikacije se mogu pokrenuti s ugrađenim serverima kao što su Tomcat ili Jetty, što eliminira potrebu za vanjskim serverom.
- Upravljanje ovisnostima: Pruža pojednostavljeno upravljanje ovisnostima koristeći Maven ili Gradle.
- Brzi razvoj: Pruža mnoge gotove komponente koje ubrzavaju razvoj i omogućuju fokusiranje na poslovnu logiku.

Spring Boot čini jezgru backenda ove aplikacije. Pruža komponente kontrolere, koji definiraju ponašanje pri interakciji korisnika s aplikacijom. Kontroleri se vežu za konkretnu adresu, definiraju moguće staze na toj adresi te što uraditi pri njihovom pristupu. To obično uključuje dohvaćanje nekih podataka iz baze podataka te crtanje novog pogleda za korisnika. Slika 3.1 prikazuje TemplateController te putanje definirane unutar njega, neke sa GET zahtjevima, a neke sa POST zahtjevima.

### 3.3.2. Jakarta EE

Jakarta EE (ranije poznata kao Java EE) je skup specifikacija za izgradnju korporativnih aplikacija. Jedna od tih specifikacija je ranije spomenuti JPA koji se koristi za spajanje na bazu podatka.

Bitna funkcionalnost koju pruža u ovoj aplikaciji je pristup sjednici iz koda. Takav pristup možemo vidjeti na slici 3.1. Koristeći HttpSession objekt dohvaćamo trenutno prijavljenog korisnika.

```java
@Controller
public class TemplatesController {

    @GetMapping("/templates")
    public String templates(Model model, HttpSession session) {
        User user = DAOProvider.getDAO().getUserById((int) session.getAttribute("userId"));
        model.addAttribute("user", user);
        return "template/templates";
    }

    @GetMapping("/createTemplate")
    public String createTemplate(Model model) {
        model.addAttribute("exercises", DAOProvider.getDAO().getAllExercises());
        return "template/create_template";
    }

    @PostMapping("/deleteTemplate")
    public String deleteTemplate(@RequestParam("id") int templateId, HttpSession session) {
        if (DAOProvider.getDAO().userHasTemplate((int) session.getAttribute("userId"), templateId)) {
            DAOProvider.getDAO().deleteTrainingTemplate(templateId);
        }
        return "redirect:/templates";
    }

    // and some more...

}
```

Slika 3.1 Razred TemplatesController sa metodama koje definiraju različite putanje

Još jedna funkcionalnost koju nudi je mogućnost postavljanja filtera. Filter je isječak koda koji se aktivira svaki put kad želimo pristupiti poddomeni. Obično se mogu koristiti za provjeru autorizacije, za tu svrhu se koristi i u ovoj aplikaciji. Još jedna primjena je upravljanje transakcijama u bazi podataka. Kada pristupimo nekoj adresi:

1. Uđemo u filter, stvorimo konekciju s bazom podataka i započnemo transakciju te prosljeđuje kontroleru
2. Kontroler izvršava svoju logiku
3. U komunikaciji s modelom izmijeni podatke u bazi podataka
4. Izlazimo iz filtera, ako je sve prošlo u redu izvršavamo transakciju

### 3.3.3. Zaštita podataka

Zaštita podataka je ključna komponenta svake moderne aplikacije, posebno kada se radi o osjetljivim informacijama kao što su korisničke lozinke. U ovoj aplikaciji, lozinke korisnika nisu pohranjene u običnom tekstu, već su zaštićene korištenjem kriptografskih tehnika koje uključuju raspršivanje (dalje hashiranje) i dodavanje salta.

Hashiranje je proces kojim se unos podataka (u ovom slučaju lozinka) prolazi kroz kriptografski algoritam kako bi se generirao jedinstveni niz znakova fiksne duljine, poznat kao hash. U ovoj aplikaciji korišten je algoritam SHA-256 (Secure Hash Algorithm 256-bit) za generiranje hashova lozinki. Hashiranje lozinki je važno jer omogućuje pohranu lozinki na siguran način. Kada se lozinka hashira originalna lozinka ne može se lako vratiti iz hasha, što znači da čak i ako netko neovlašteno pristupi bazi podataka, neće moći izravno pročitati korisničke lozinke.

Salt je dodatni slučajni podatak koji se dodaje lozinki prije hashiranja. Korištenje salta sprječava napade poznate kao "precomputed hash attacks" (napadi unaprijed izračunatih hashova) kao što su napadi s rainbow tablicama. Rainbow tablice su unaprijed izračunate tablice hash vrijednosti za različite kombinacije znakova koje omogućuju brzo prepoznavanje lozinki. Dodavanjem jedinstvenog salta svakoj lozinki, čak i ako dva korisnika imaju istu lozinku, njihovi hashovi će biti različiti, čime se dodatno povećava sigurnost.

Generiranje hasha i salta je odrađeno sigurnim metodama koje su ugrađene u programski jezik Java u paketu java.security.

## 3.4. Frontend

### 3.4.1. JSP

JSP (Jakarta Server Pages) je tehnologija koja omogućava izradu dinamičkih web stranica koristeći Java servlet tehnologiju. Omogućava umetanje Java koda direktno unutar HTML-a, što olakšava izradu dinamičkih web stranica.

### 3.4.2. JSTL

JSTL (Jakarta standard tag libary) je tehnologija koja se koristi kao dodatak za JSP. Sadrži skupinu korisnih tagova za JSP koje olakšavaju rad s Java kodom unutar JSP stranica. Omogućuje korištenje predefinirani tagova za zajedničke zadatke kao što su iteracije, uvjetno izvođenje, formatiranje podataka itd.

### 3.4.3. CSS

CSS (Cascading style sheets) je stilski jezik korišten za opisivanje izgleda i oblikovanja dokumenta napisanog u markup jeziku poput HTML-a. Omogućava razdvajanje strukture sadržaja od prezentacije, čineći web stranice vizualno atraktivnijima i lakšima za održavanje. Omogućava kontroliranje izgleda elemenata na stranici uključujući boje, fontove, raspored i margine. Podržava responzivni dizajn, omogućavajući web stranicama da se prilagode različitim veličinama ekrana i uređajima.

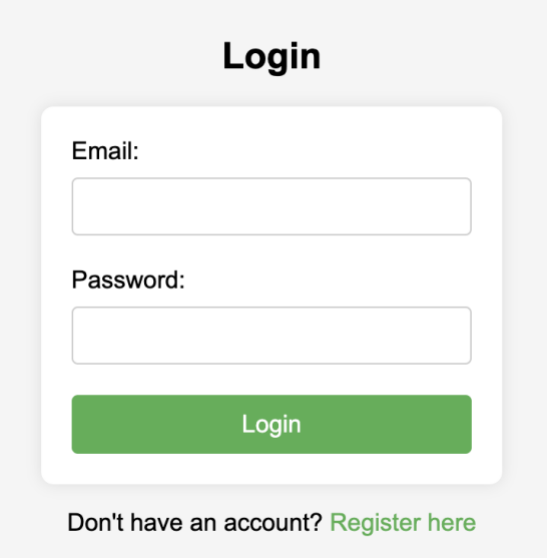
### 3.4.4. JavaScript

JavaScript (JS) je skriptni jezik visoke razine koji je izvorno razvijen za stvaranje dinamičnih i interaktivnih web stranica. Omogućava dodavanje interaktivnih elementa na web stranice poput animacija, formi, validacija i različitih događaja. JavaScript omogućava manipulaciju sadržajem stranice i ponašanjem korisničkog sučelja u stvarnom vremenu. Može raditi na klijentskoj strani, omogućavajući brze i interaktivne odgovore na korisničke akcije bez potrebe za osvježavanjem stranice.

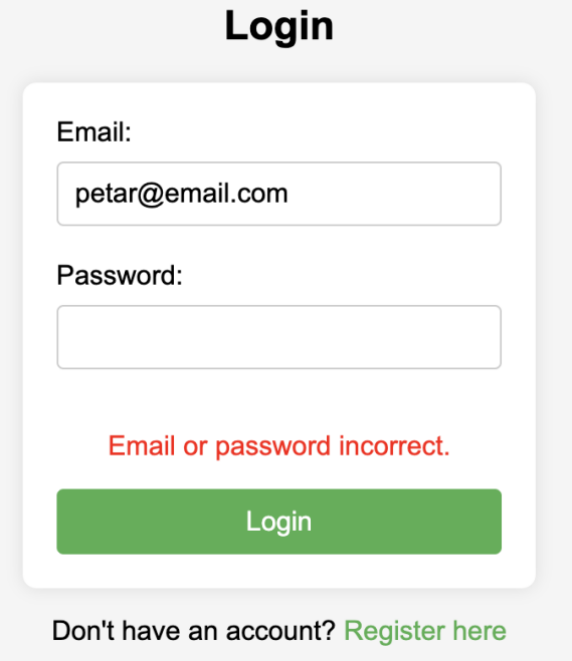
# 4.Korisničke upute

## 4.1. Prijava i registracija korisnika

Stranica za prijavu može se dohvatiti na putanji /login. Sastoji se od forme koja očekuje e-mail adresu i lozinku korisnika i može se vidjeti na slici 4.1. Vrši provjeru formata e-mail adrese lokalno i na serveru. Upozori korisnika ukoliko unese pogrešan par maila i lozinke (slika 4.2). Dodatno, na dnu, nalazi se link na stranicu za registraciju.

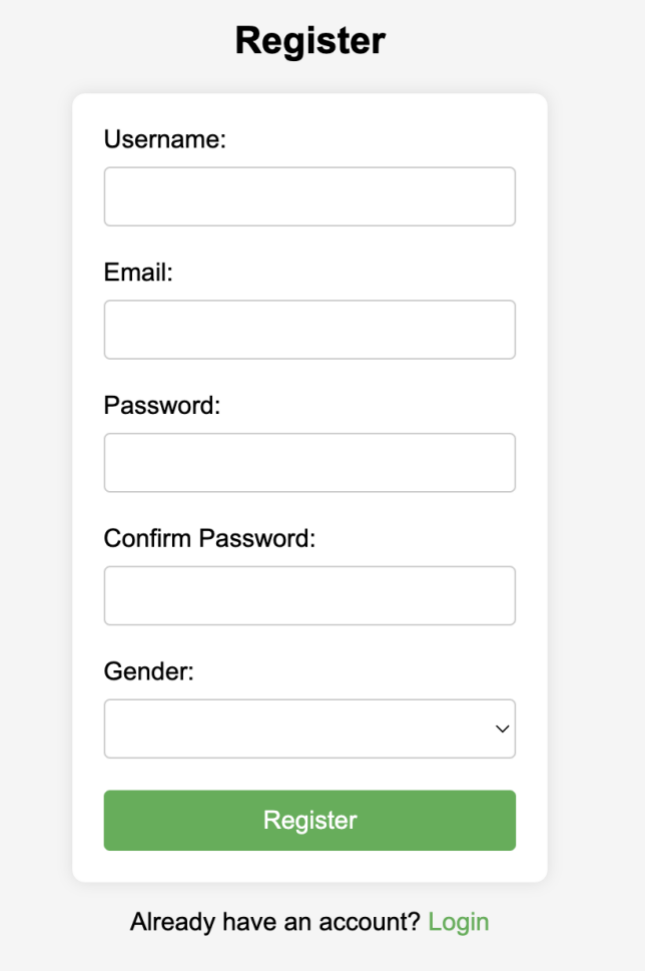



Slika 4.1 Stranica za prijavu




Slika 4.2 Stranica za prijavu s upozorenjem

Stranica za registraciju može se dohvatiti na putanji /register. Sastoji se od forme koja očekuje korisničko ime, e-mail adresu, lozinku, potvrdu lozinke te rod. Može se vidjeti na slici 4.3. Validaciju podataka vrši na serverskoj strani. Kao i stranica za prijavu, ima mogućnost upozoravanja korisnika za pogrešno unesene informacije. Dodatno, na dnu, nalazi se link na stranicu za prijavu.




Slika 4.3 Stranica za registraciju

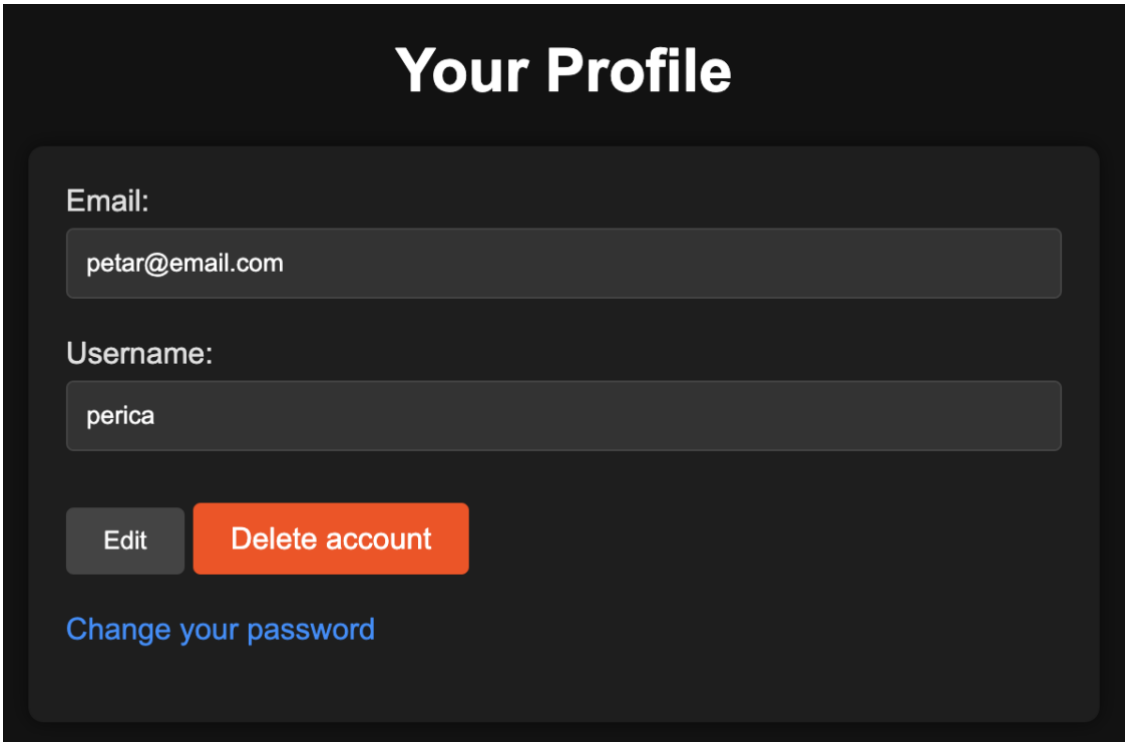
## 4.2. Glavna stranica i stranice za uređivanje korisnika

Nakon prijave ili nakon što korisnik pritisne logo aplikacije, odveden je na početnu stranicu. Ona nam nudi opcije: početak novog praznog treninga, početak treninga od već napravljenog predloška, vidjeti sve vježbe koje aplikacija nudi, vidjeti korisnikove prošle treninge, vidjeti profil korisnika i odjaviti se. Vidljiva je na slici 4.4

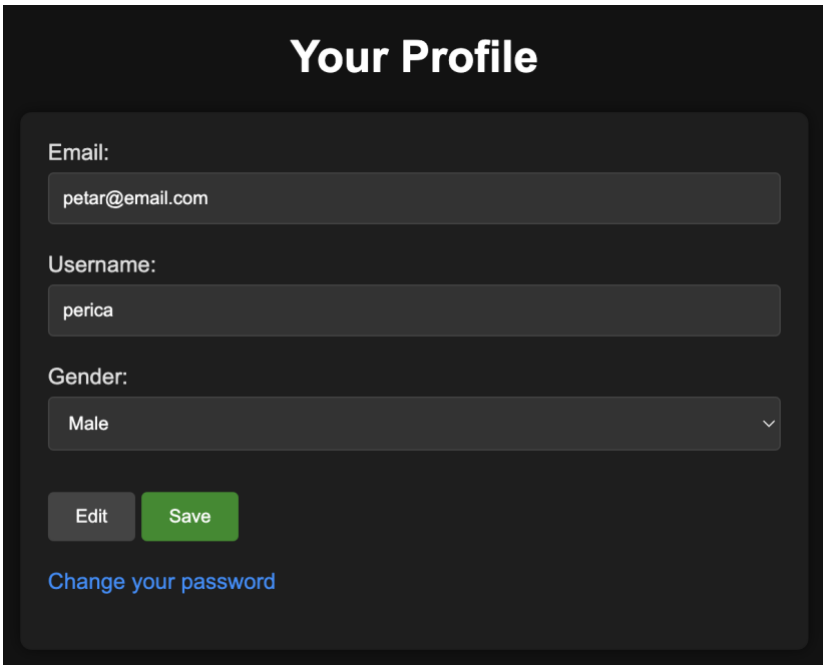

Slika 4.4 Početna stranica

Stranica za profil korisnika početno se nalazi u stanju gledanja i vidljiva je na slici 4.5. Prikazuje e-mail adresu i korisničko ime. Sadrži gumb za brisanje računa koje će nas odvesti na stranicu za potvrdu brisanja. Sadrži link za stranicu za promjenu lozinke.

Ako pritisnemo gumb „Edit“ prebacujemo se u način uređivanja vidljiv na slici 4.6. Stranica se pretvara u formu sa unosom za e-mail adresu, korisničko ime te rod. Nestaje gumb za brisanje računa i zamjenjuje ga gumb za spremanje promjena. Pri unosu nepravilnih informacija korisnik će biti upozoren pri pokušaju spremanja.

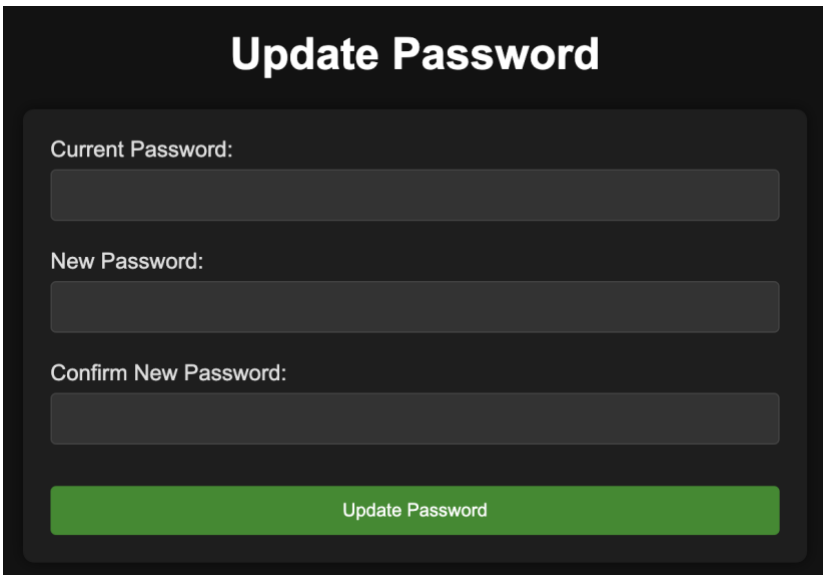



Slika 4.5 Stranica profila u načinu gledanja



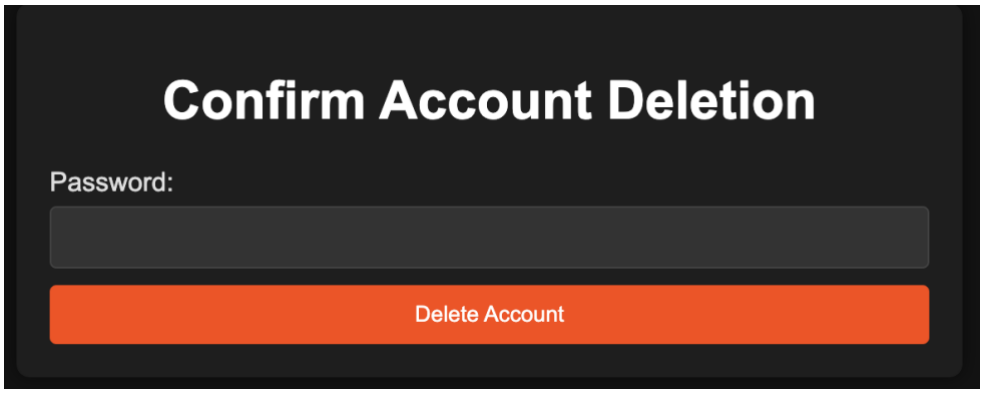

Slika 4.6 Stranica profila u načinu uređivanja

Stranica za ažuriranje lozinke sastoji se od tri polja. Prvo polje očekuje trenutnu korisnikovu lozinku. Drugo i treće polje služe za novu lozinku i potvrdu. Vidljiva je na slici 4.7




Slika 4.7 Stranica za ažuriranje lozinke

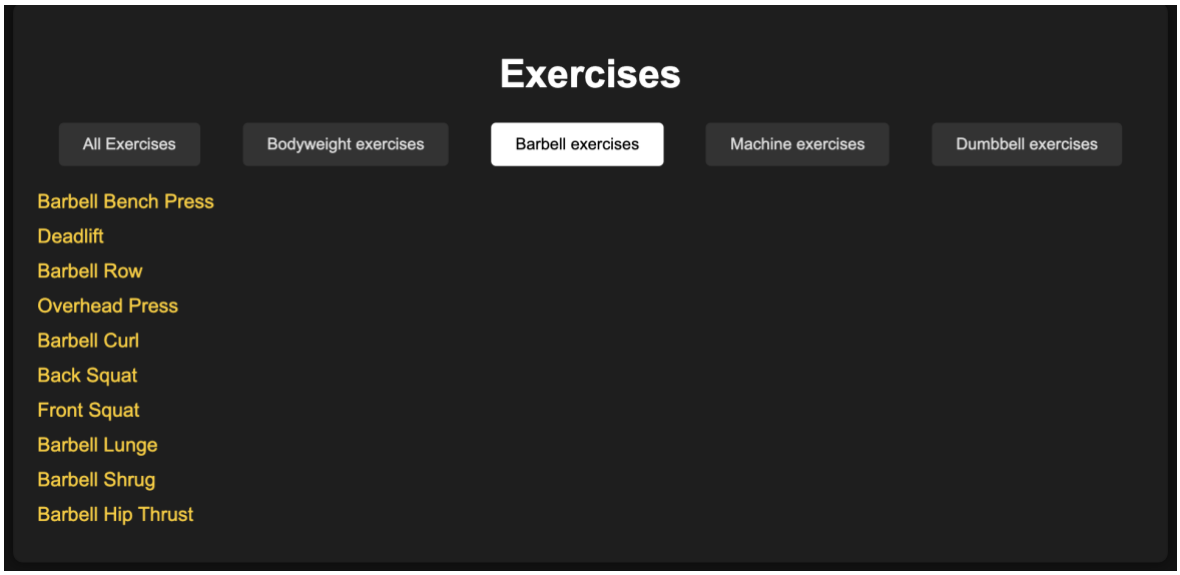
Stranica za brisanje korisničkog računa se sastoji od polja za upis trenutne šifre kao zaštita od mogućih napada. Vidljiva je na slici 4.8




Slika 4.8 Stranica za brisanje korisničkog računa

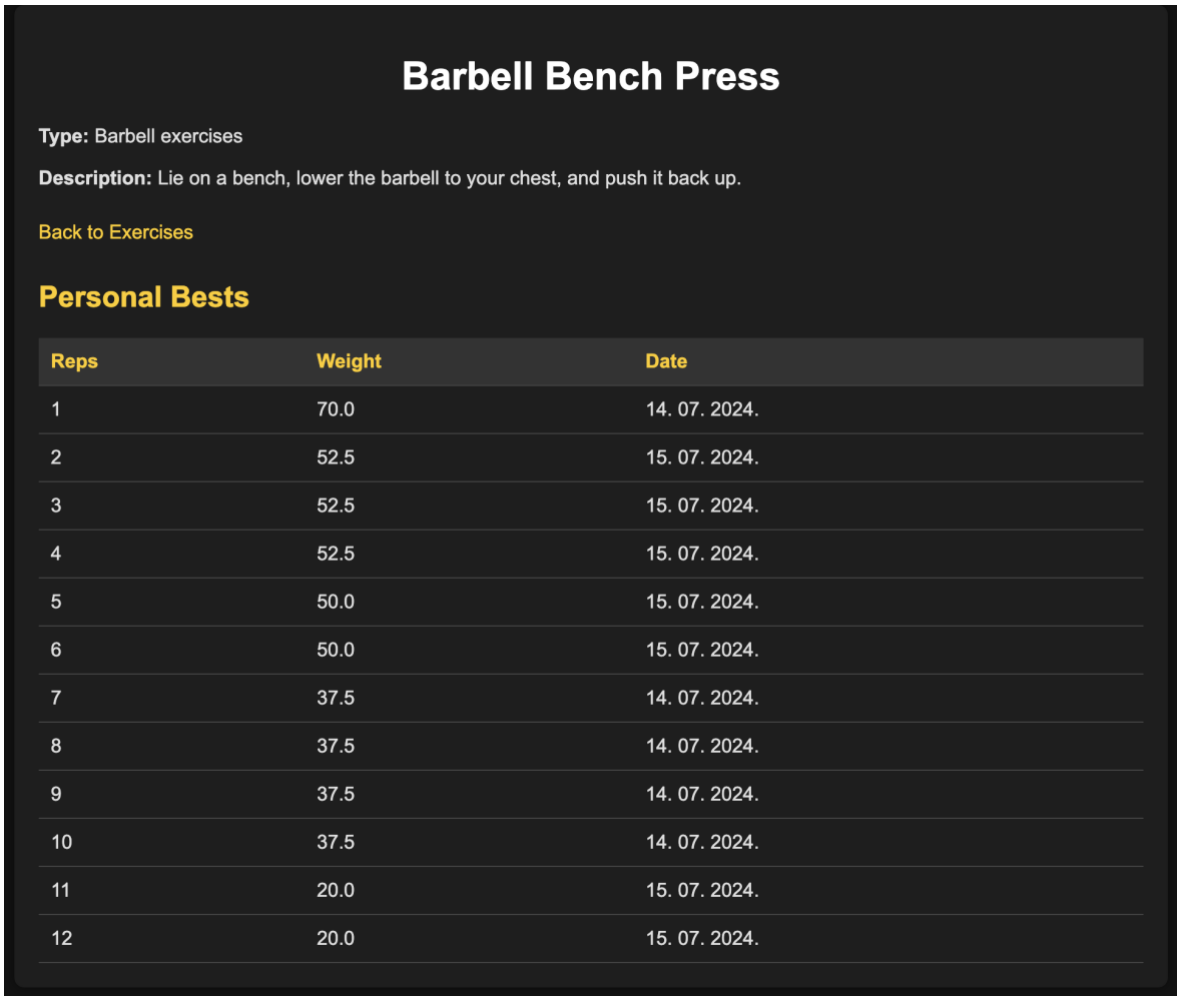
## 4.3. Informacije o vježbama

Stranica koja prikazuje sve dostupne vježbe se nalazi na putanji /exercises i vidljiva je na slici 4.9. Kako su vježbe podijeljene na 4 kategorije, tako i stranica ima 4 kartice te dodatnu karticu za prikazivanje svih vježbi. Klikom na neku vježbu vodi korisnika na detaljne informacije o toj vježbi.



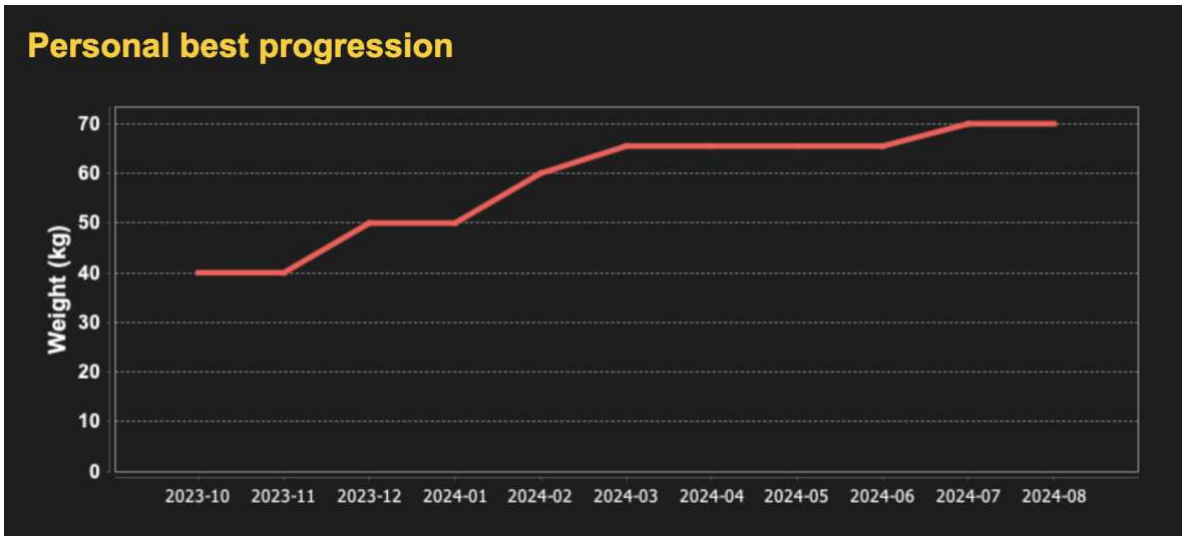

Slika 4.9 Stranica prikazuje sve vježbe koje se izvode sa šipkom

Stranica za detalje vježbe sadrži ime vježbe, kategoriju vježbe te kratki opis izvođenja vježbe. Dodatno, ako je korisnik nekad izvodio tu vježbu, prikazuje informacije o najboljim rezultatima. Opširnije, za svaki broj ponavljanja manji ili jednak od 12 prikazuje najveću dignutu kilažu te kada je postignuta prvi put. Slika 4.10 prikazuje stranicu za vježbu Barbell Bench Press (Potisak šipkom na klupi) za jednog korisnika.




Slika 4.10 Stranica o detaljima vježbe Barbell Bench Press

Na stranici za detalje vježbe, ukoliko korisnik ima zabilježene najbolje rezultate, prikazuje se graf na kojem se vidi najveća dignuta kilaža po mjesecu u prethodnoj godini. Za stvaranje grafa korištena je besplatna biblioteka za Javu JFreeChart[9]. Jedan takav graf se vidi na slici 4.11.



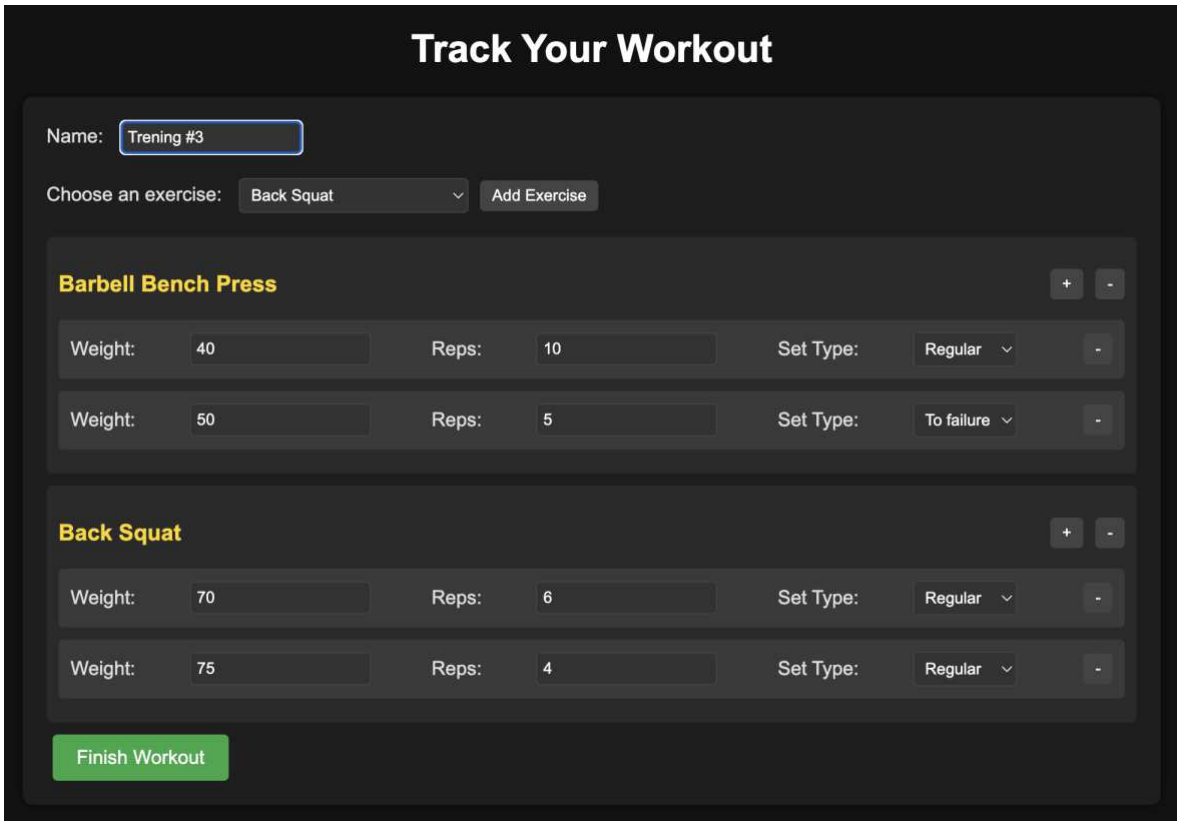

Slika 4.11 Graf koji prikazuje napredak na vježbi

## 4.4. Zapisivanje, uređivanje i pregledavanje treninga

Stranica za zapis treninga služi za početak novog treninga. Može se započeti prazan trening ili trening preko predloška. Prazan trening neće imati ubačene treninge, dok će treningu preko predloška automatski biti ubačene vježbe. Postoji polje za upis imena treninga koje je automatski predloženo. Lista vježbi koristi se za dodavanje novih vježbi. Uz svaku vježbu postoji gumb za dodavanje/brisanje serija. Uz svaku seriju postoje polja za upis kilaže, broja ponavljanja te vrste serije. Pored svake serije ima gumb za brisanje određene serije. Kada korisnik završi sa treningom treba pritisnuti gumb za spremanje.

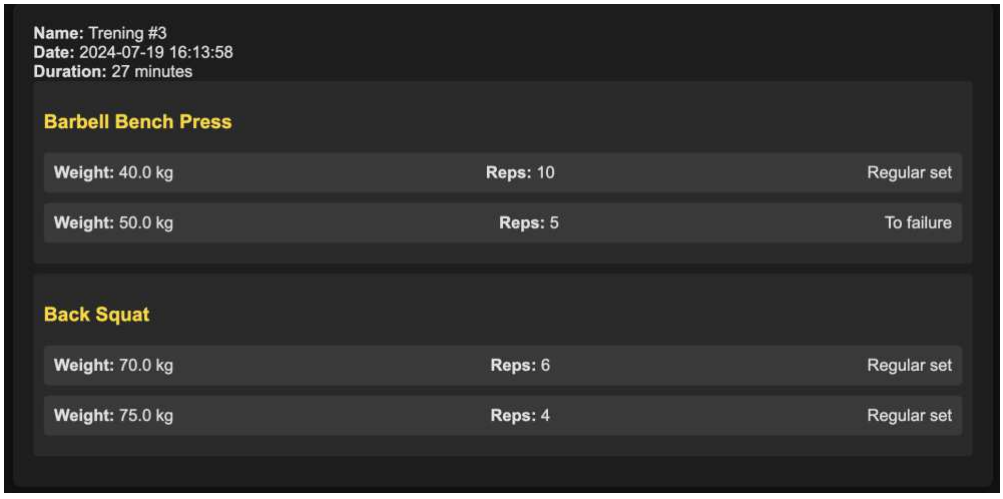
Stranica za uređivanje treninga izgleda identično stranici za zapisivanje treninga. Jedina razlika je u tome da su sve informacije o treningu već unesene te se mogu mijenjati po volji.

Slika 4.12 prikazuje stranicu za zapis i uređivanje treninga.




Slika 4.12 Stranica za zapis i uređivanje treninga

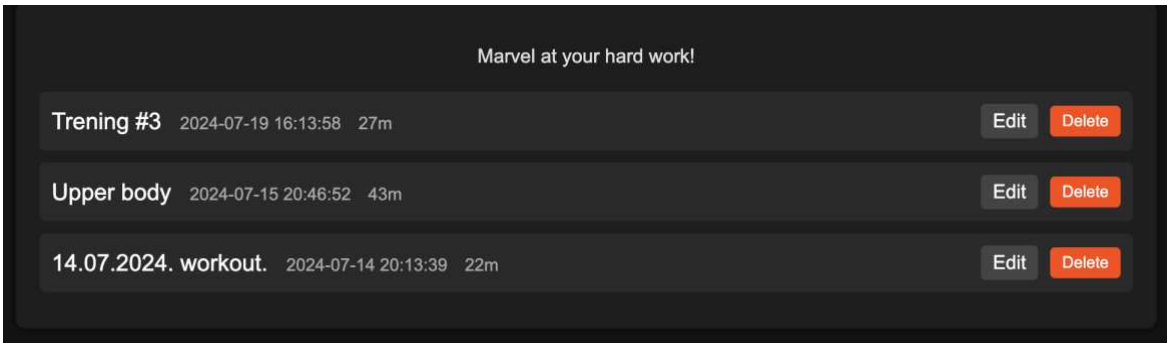
Stranica za pregled sadrži sve informacije o treningu. To su: ime treninga, datum i vrijeme, trajanje, odrađene vježbe i serije. Vidljiva je na slici 4.13




Slika 4.13 Stranica za pregled treninga

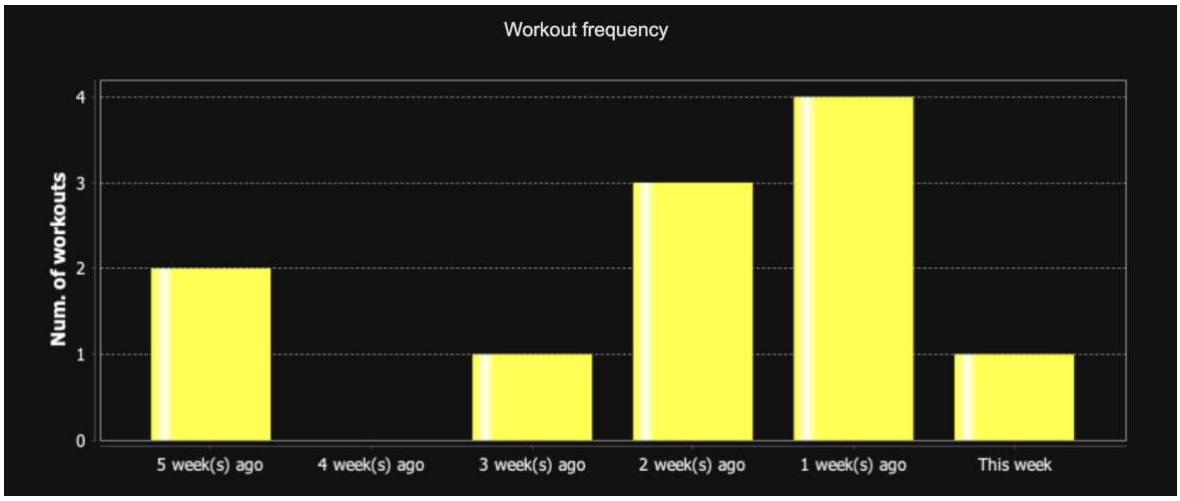
Stranica za pregled svih treninga sastoji se od liste svih odrađenih korisnikovih treninga. Poredani su tako da su najnoviji treninzi  na vrhu. Uz svaku stavku su

informacije o imenu treninga, datumu i vremenu te trajanju. Pritiskom na ime treninga prikazuje detaljne informacije o treningu. Dodatno, uz svaku stavku imamo gumb za uređivanje treninga i gumb za brisanje treninga. Slika 4.14 prikazuje ovu stranicu.




Slika 4.14 Prikaz liste treninga

Na stranici za pregled starih treninga također se nalazi (slika 4.15) graf koji prikazuje broj odrađenih treninga po tjednu u prethodnih 6 tjedana.



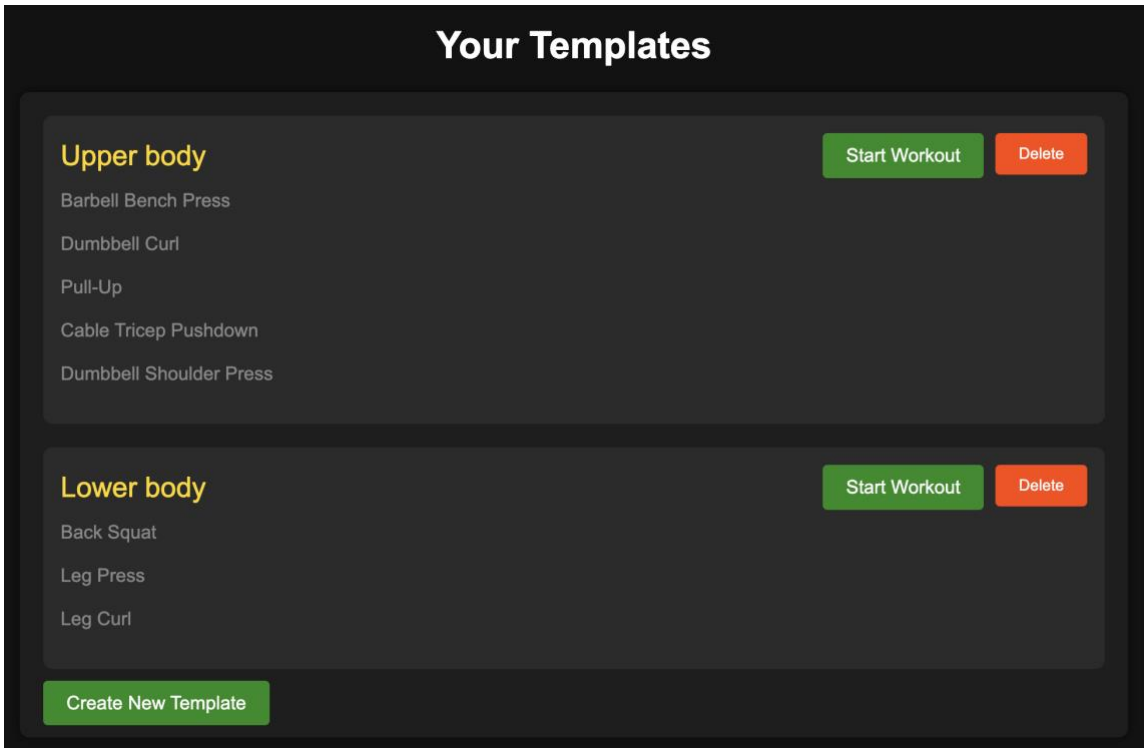

Slika 4.15 Graf koji prikazuje učestalost treninga

## 4.5. Predlošci

Stranica za stvaranje predloška (slika 4.16) sastoji se od polja za upis imena predloška i liste vježbi za dodavanje vježbe u predložak. Kada dodamo vježbu, pojavi se ispod. Svaka vježba ima na sebi gumb za brisanje te vježbe. Kada je korisnik zadovoljan, pritisne gumb za spremanje predloška.


Slika 4.16 Stranica za stvaranje predloška

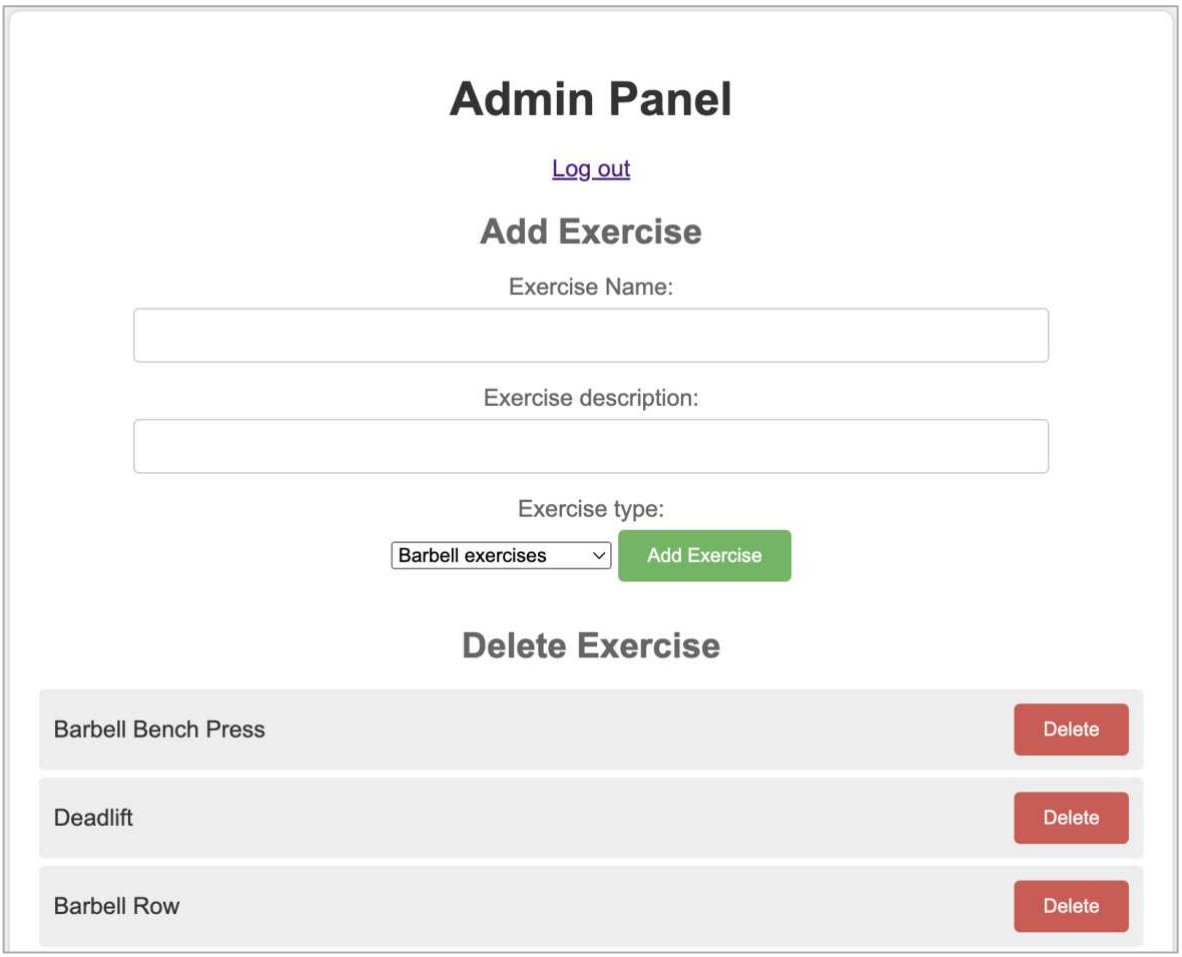
Stranica za pregled predložaka (slika 4.17) prikazuje sve korisnikove predloške. Uz svaku stavku prikazano je ime predloška, vježbe od kojih se sastoji, gumb za početak treninga s tim predloškom te gumb za brisanje predloška. Na dnu je gumb za stvaranje novog predloška.




Slika 4.17 Stranica za pregled predložaka

## 4.6. Administrator

Administrator se pri prijavi odvede na upravljačku ploču. To je jedina stranica kojoj administrator ima pristup. Služi za dodavanje novih vježbi i brisanje starih. Sadrži formu za stvaranje nove vježbe te listu starih vježbi s gumbom za brisanje. Vidljiva je na slici 4.18




Slika 4.18 Upravljačka ploča za administratora

# Zaključak

Cilj zadatka bio je napraviti aplikaciju za praćenje treninga s istim temeljnim funkcijama kao već postojeće i često korištene aplikacije. U aplikaciji je omogućeno dodavanje i uređivanje treninga, definirane su posebne vrste serija i vježbi, razvijena je podrška za predloške, razvijen je prikaz najboljih korisnikovih rezultata te prikaz i uređivanje korisnikovog računa.

Zadatak je počeo modeliranjem baze podataka. Na to je obraćena posebna pažnja zbog utjecaja na cijeli projekt. Nakon definiranja ER modela, pretvoren je u relacijski model te implementiran uz pomoć alata Hibernate.

Moguće ideje za daljnji razvoj projekta su bolja podrška za ručne uređaje i mogućnost povratka zaboravljene lozinke.

# Literatura

[1] Nastavni materijali iz kolegija Baze podataka, Fakultet elektrotehnike i računarstva, srpanj 2024.

[2] Nastavni materijali iz kolegija Razvoj programske potpore za web, Fakultet elektrotehnike i računarstva, srpanj 2024.

[3] Nastavni materijali iz kolegija Odabrana poglavlja razvoja programske potpore 2, Fakultet elektrotehnike i računarstva, srpanj 2024.

[4] Nastavni materijali iz kolegija Sigurnost računalnih sustava, Fakultet elektrotehnike i računarstva, srpanj 2024.

[5] Spring Boot, https://spring.io/projects/spring-boot, 19.7.2024.

[6] Tutorial za Spring Boot, https://www.geeksforgeeks.org/spring-boot/, 19.7.2024.

[7] ERDPlus, https://erdplus.com/, 20.3.2024.

[8] SQLite, https://www.sqlite.org/docs.html, 19.7.2024.

[9] JFreeChart, https://www.jfree.org/jfreechart/, 21.8.2024.

# Sažetak

Tema ovog završnog rada je izrada aplikacije za praćenje treninga u teretani. Prvo opisuje modeliranje baze podataka, pa nakon toga opisuje stvaranje web aplikacije. Na kraju prikazuje izgled i funkcije web aplikacije. Aplikacija omogućuje prijavu i stvaranje novog računa, zapisivanje informacija o treninzima u teretani, stvaranje predložaka za lakše zapisivanje i praćenje najboljih rezultata.

Za implementaciju baze podataka korišteni su SQLite i Hibernate. Za logiku web aplikacije korišteni su Spring Boot i Jakarta EE. Izgled aplikacije je moderan i interaktivan te je postignut uz pomoć tehnologija JSP, JSTL, CSS i JavaScript.

***Ključne riječi***

java, sql, sqlite, spring boot, hibernate, baza podataka, web aplikacija

# Abstract

The topic of this final thesis is the creation of an application for monitoring workouts in the gym. It first describes database modeling, then creating the web application. Finally, it shows the appearance and functions of the web application. The application allows logging in and creating a new account, recording information about workouts in the gym, creating templates for easier recording, and tracking of personal bests.

SQLite and Hibernate were used to implement the database. For the web application logic, Spring Boot was used with Jakarta EE. The appearance of the application is modern and interactive and was achieved with the help of JSP, JSTL, CSS, and JavaScript technologies.

***Keywords***

java, sql, sqlite, spring boot, hibernate, database, web application