Картина, която съдържа текст, графична колекция

Описанието е генерирано автоматичноТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА

Факултет по изчислителна техника и автоматизация

Катедра „Софтуерни и Интернет технологии“

Проект по дисциплината   
“Проектиране на бази данни”

**Тема**

Фитнес социална мрежа за менажиране на програми

|  |
| --- |
| Изготвил: Петър Илков |
| Специалност: СИТ |
| Група: 2Б |
| Факултетни номера: 22621624 |
|  |

2025

Съдържание

[I. Увод 4](#_Toc198286627)

[Задание 4](#_Toc198286628)

[Описание и идея на проекта 4](#_Toc198286629)

[Цел на проекта 4](#_Toc198286630)

[II. Модели 5](#_Toc198286631)

[Таблици 5](#_Toc198286632)

[Релационен и логически модел 9](#_Toc198286633)

[Модели реализирани чрез ORM – Hibernate 11](#_Toc198286634)

[Модели реализирани чрез MySQL заявки 18](#_Toc198286635)

[Описание на полетата 21](#_Toc198286636)

[Таблица: users (Потребители) 21](#_Toc198286637)

[Таблица: programs (Програми) 22](#_Toc198286638)

[Таблица: exercises (Упражнения) 22](#_Toc198286639)

[Таблица: exercise\_muscle\_groups (Групи мускули към упражнения) 23](#_Toc198286640)

[Таблица: workouts (Тренировки) 23](#_Toc198286641)

[Таблица: logs (Логове за изпълнение на тренировки) 23](#_Toc198286642)

[Таблица: logs\_exercises (Упражнения, изпълнени в логове) 23](#_Toc198286643)

[Таблица: workout\_exercises (Упражнения в тренировки) 24](#_Toc198286644)

[Таблица: workout\_schedules (График на програма) 24](#_Toc198286645)

[Таблица: likes (Харесвания на програми) 25](#_Toc198286646)

[Таблица: notifications (Известия) 25](#_Toc198286647)

[Примери с реални данни 25](#_Toc198286648)

[III. Избрани технологии за реализация 26](#_Toc198286649)

[Spring Boot – Java 26](#_Toc198286650)

[Spring Data JPA (Hibernate) 26](#_Toc198286651)

[Spring MVC – Thymeleaf 26](#_Toc198286652)

[Spring Security 26](#_Toc198286653)

[HTML, CSS и JavaScript 26](#_Toc198286654)

[MySQL 27](#_Toc198286655)

[DataGrip 27](#_Toc198286656)

[IV. Описание на отделните части на системата 27](#_Toc198286657)

[Микросървисна архитектура 27](#_Toc198286658)

[Модулна структура в отделни пакети 27](#_Toc198286659)

[V. Ръководство за потребителя 28](#_Toc198286660)

[Потребител 28](#_Toc198286661)

[ Регистрация и вход в системата 29](#_Toc198286662)

[ Създаване на упражнения 29](#_Toc198286663)

[ Създаване на тренировки 30](#_Toc198286664)

[ Създаване на програми 30](#_Toc198286665)

[ Преглед на програми от други потребители 31](#_Toc198286666)

[ Харесване на програми 31](#_Toc198286667)

[ Логване на тренировки 32](#_Toc198286668)

[ Преглед на всички одобрени упражнения 33](#_Toc198286669)

[ Преглед на собствения профил 33](#_Toc198286670)

[ Редакция на профил 34](#_Toc198286671)

[ Преглед на профилите на други потребители 34](#_Toc198286672)

[ Преглед на качените упражнения 35](#_Toc198286673)

[ Преглед на създадените програми 35](#_Toc198286674)

[ Преглед на създадените тренировки 36](#_Toc198286675)

[ Преглед на харесаните програми: 36](#_Toc198286676)

[Администратор 36](#_Toc198286677)

[ Одобрение или отхвърляне на упражнения 37](#_Toc198286678)

[ Управление на потребителски профили 39](#_Toc198286679)

[ Редакция на профил 39](#_Toc198286680)

[ Преглед на качените програми и упражнения 39](#_Toc198286681)

[VI. Примерна работа и описание на функционалност 39](#_Toc198286682)

[Начална конфигурация 39](#_Toc198286683)

[Демонстрация на основните функционалности 40](#_Toc198286684)

[VII. Възможности за бъдещо развитие 41](#_Toc198286685)

[Разширени възможности за социална интеракция 41](#_Toc198286686)

[Мобилна версия или мобилно приложение 41](#_Toc198286687)

[Подобрена статистика и анализи 41](#_Toc198286688)

[Поддръжка на повече типове съдържание 42](#_Toc198286689)

[Персонализирани препоръки 42](#_Toc198286690)

[Интеграция с външни устройства 42](#_Toc198286691)

[Модул за администраторска статистика 42](#_Toc198286692)

[VIII. Заключение и обобщение 42](#_Toc198286693)

# **I. Увод**

## Задание

Проектът е разработен по дисциплината *„Проектиране на бази данни“* и представлява самостоятелно избрана и реализирана тема. Той цели да демонстрира цялостен процес по моделиране, проектиране и реализация на база данни чрез съвременни технологии. Темата е избрана с оглед на практическото ѝ приложение и актуалността ѝ в контекста на нарастващия интерес към персонализираните дигитални фитнес решения и социалните мрежи.

## Описание и идея на проекта

Проектът представлява уеб-базирана платформа, съчетаваща функционалностите на фитнес приложение с тези на социална мрежа. Основната идея е да се създаде система, чрез която потребителите могат да организират и проследяват своите фитнес тренировки и програми, както и да взаимодействат с други потребители чрез споделяне, харесване и преглед на програми.

Приложението осигурява възможност за:

* създаване и персонализиране на упражнения и тренировки;
* изграждане на седмични програми;
* споделяне на програми с общността;
* администриране и модериране на съдържание;
* събиране и визуализиране на информация чрез база данни;
* използване на микросървис за харесване на програми и изпращане на нотификации.

Идеята на проекта е да покаже как базата данни и архитектурата на системата могат да бъдат използвани ефективно за решаване на реален казус – поддържане на дигитална тренировъчна история и изграждане на ангажирана потребителска общност.

## Цел на проекта

Целта на проекта е да се проектира и реализира база данни и цялостна система, която да поддържа основните процеси във фитнес социална мрежа. Проектът има за задача да демонстрира:

* ефективно моделиране на данни чрез релационна база;
* изграждане на връзки между различни обекти (потребители, упражнения, тренировки, програми);
* използване на ORM технологии (Hibernate) за съвременна работа с базата;
* проектиране на гъвкав потребителски интерфейс, който комуникира с базовите структури;
* разработване на функционалности, позволяващи интеракция между потребителите;
* прилагане на добри практики при реализацията на бекенд логика, сигурност и потребителски роли;
* създаване на начален демонстрационен набор от данни, който да позволява бърз преглед и тестване на системата.

Проектът има както техническа, така и практическа цел – да илюстрира как съвременните подходи за проектиране и управление на бази данни се използват за изграждане на реална уеб платформа с ясна полезност и възможност за бъдещо развитие.

# II. Модели

## Таблици

* Users

Картина, която съдържа текст, екранна снимка, софтуер, Мултимедиен софтуер

Генерираното от ИИ съдържание може да е неправилно.

* Programs

Картина, която съдържа текст, екранна снимка, Шрифт, софтуер

Генерираното от ИИ съдържание може да е неправилно.

* Exercises

Картина, която съдържа текст, екранна снимка, софтуер, Шрифт

Генерираното от ИИ съдържание може да е неправилно.

* Workouts

Картина, която съдържа текст, екранна снимка, Шрифт, номер

Генерираното от ИИ съдържание може да е неправилно.

* Workout\_Exercises

Картина, която съдържа текст, екранна снимка, Шрифт, номер

Генерираното от ИИ съдържание може да е неправилно.

* Workout\_Schedules

Картина, която съдържа текст, екранна снимка, Шрифт, номер

Генерираното от ИИ съдържание може да е неправилно.

* Logs

Картина, която съдържа текст, екранна снимка, Шрифт, номер

Генерираното от ИИ съдържание може да е неправилно.

* Logs\_Exercises

Картина, която съдържа текст, екранна снимка, Шрифт, номер

Генерираното от ИИ съдържание може да е неправилно.

* Exercise\_Muscle\_Groups

Картина, която съдържа текст, екранна снимка, Шрифт, номер

Генерираното от ИИ съдържание може да е неправилно.

* Likes

Картина, която съдържа текст, екранна снимка, Шрифт, номер

Генерираното от ИИ съдържание може да е неправилно.

* Notifications

Картина, която съдържа текст, екранна снимка, Шрифт, номер

Генерираното от ИИ съдържание може да е неправилно.

## Релационен и логически модел

Картина, която съдържа текст, екранна снимка, Шрифт, Мултимедиен софтуер

Генерираното от ИИ съдържание може да е неправилно.Картина, която съдържа текст, екранна снимка, диаграма

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, диаграма, Паралелен

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, електроника, екранна снимка, Шрифт

Генерираното от ИИ съдържание може да е неправилно.

## Модели реализирани чрез ORM – Hibernate

* User

@Builder  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
@Table(name = "users")  
public class User {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*UUID*)  
 private UUID id;  
  
 @Column(unique = true, nullable = false)  
 private String email;  
  
 @Column(unique = true, nullable = false)  
 private String username;  
  
 @Column(nullable = false)  
 private String password;  
  
 @Enumerated(EnumType.*STRING*)  
 @Column(nullable = false)  
 private Country country;  
  
 @Enumerated(EnumType.*STRING*)  
 @Column(nullable = false)  
 private UserRole userRole;  
  
 @Enumerated(EnumType.*STRING*)  
 @Column(nullable = false)  
 private Level level;  
  
 @Column(nullable = false)  
 private LocalDateTime createdOn;  
  
 @Column(nullable = false)  
 private LocalDateTime updatedOn;  
  
 private boolean isActive;  
  
 private String firstName;  
  
 private String lastName;  
  
 private String profilePicture;  
  
 private String bio;  
  
 @OneToMany(mappedBy = "user", cascade = CascadeType.*ALL*)  
 private List<Program> programs = new ArrayList<>();  
  
 @OneToMany(mappedBy = "user", cascade = CascadeType.*ALL*)  
 private List<Log> logs = new ArrayList<>();  
  
 @ManyToOne(cascade = CascadeType.*ALL*)  
 private Program activeProgram;  
}

* Program

@Builder  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
@Table(name = "programs")  
public class Program {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*UUID*)  
 private UUID id;  
  
 @Column(nullable = false)  
 private String name;  
  
 @Column(nullable = false)  
 private Boolean sharedWithOthers;  
  
 @Column(nullable = false)  
 private LocalDateTime createdOn;  
  
 private String description;  
  
 @Enumerated(EnumType.*STRING*)  
 @Column(nullable = false)  
 private Difficulty difficulty;  
  
 @ManyToOne  
 @JoinColumn(name = "user\_id", nullable = false)  
 private User user;  
  
 @OneToMany(mappedBy = "program", cascade = CascadeType.*ALL*)  
 private List<WorkoutSchedule> workoutSchedules;  
  
}

* Exercise

@Builder  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
@Table(name = "exercises")  
public class Exercise {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*UUID*)  
 private UUID id;  
  
 @Column(nullable = false, unique = true)  
 private String name;  
  
 @Column(nullable = false)  
 private String description;  
  
 @Enumerated(EnumType.*STRING*)  
 @Column(nullable = false)  
 private Difficulty difficulty;  
  
 @ElementCollection  
 @CollectionTable(name = "exercise\_muscle\_groups", joinColumns = @JoinColumn(name = "exercise\_id"))  
 @Enumerated(EnumType.*STRING*)  
 @Column(name = "muscle\_group")  
 private List<MuscleGroup> muscleGroups;  
  
 private String mediaUrl;  
  
 @ManyToOne  
 @JoinColumn(name = "created\_by")  
 private User createdBy;  
  
 @ManyToOne  
 @JoinColumn(name = "approved\_by")  
 private User approvedBy;  
  
 @Column(nullable = false)  
 private LocalDateTime createdOn;  
  
 @Enumerated(EnumType.*STRING*)  
 private ExerciseStatus status;  
}

* Workout

@Builder  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
@Table(name = "workouts")  
public class Workout {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*UUID*)  
 private UUID id;  
  
 private Integer duration;  
  
 private String additionalInfo;  
  
 @OneToMany(mappedBy = "workout", cascade = CascadeType.*ALL*)  
 private List<WorkoutSchedule> workoutSchedules;  
  
 @OneToMany(mappedBy = "workout", cascade = CascadeType.*ALL*)  
 private List<Log> logs = new ArrayList<>();  
  
 @ManyToOne  
 private User addedBy;  
  
}

* WorkoutExercises

@Builder  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
@Table(name = "workout\_exercises")  
public class WorkoutExercise {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*UUID*)  
 private UUID id;  
  
 private Integer reps;  
  
 private Integer sets;  
  
 private Double addedWeight;  
  
 @ManyToOne  
 @JoinColumn(name = "workout\_id", nullable = false)  
 private Workout workout;  
  
 @ManyToOne  
 @JoinColumn(name = "exercise\_id", nullable = false)  
 private Exercise exercise;  
}

* WorkoutSchedule

@Builder  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
@Table(  
 name = "workout\_schedules",  
 uniqueConstraints = {  
 @UniqueConstraint(columnNames = {"program\_id", "dayOfWeek"})  
 }  
)  
public class WorkoutSchedule {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*UUID*)  
 private UUID id;  
  
 @Enumerated(EnumType.*STRING*)  
 @Column(nullable = false)  
 private DayOfWeek dayOfWeek;  
  
 @ManyToOne  
 @JoinColumn(name = "program\_id", nullable = false)  
 private Program program;  
  
 @ManyToOne  
 @JoinColumn(name = "workout\_id", nullable = false)  
 private Workout workout;  
}

* Log

@Builder  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
@Table(name = "logs")  
public class Log {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*UUID*)  
 private UUID id;  
  
 private LocalDateTime completionDate;  
  
 @ManyToOne  
 @JoinColumn(name = "user\_id", nullable = false)  
 private User user;  
  
 @ManyToOne  
 @JoinColumn(name = "workout\_id", nullable = false)  
 private Workout workout;  
  
 @OneToMany(mappedBy = "log", cascade = CascadeType.*ALL*)  
 private List<LogExercise> logExercises = new ArrayList<>();  
}

* LogsExercise

@Builder  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
@Table(name = "logs\_exercises")  
public class LogExercise {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*UUID*)  
 private UUID id;  
  
 @Column(nullable = false)  
 private Integer completedReps;  
  
 @Column(nullable = false)  
 private Integer completedSets;  
  
 private Double addedWeight;  
  
 @ManyToOne  
 @JoinColumn(name = "exercise\_id", nullable = false)  
 private Exercise exercise;  
  
 @ManyToOne  
 @JoinColumn(name = "log\_id", nullable = false)  
 private Log log;  
}

* ExerciseMuscleGroup

@ElementCollection  
@CollectionTable(name = "exercise\_muscle\_groups", joinColumns = @JoinColumn(name = "exercise\_id"))  
@Enumerated(EnumType.*STRING*)  
@Column(name = "muscle\_group")  
private List<MuscleGroup> muscleGroups;

* Like

@Entity  
@Builder  
@AllArgsConstructor  
@NoArgsConstructor  
@Getter  
@Setter  
@Table(name = "likes")  
public class Like {  
 @Id  
 @GeneratedValue(strategy = GenerationType.UUID)  
 private UUID id;  
  
 @Column(nullable = false)  
 private UUID programId;  
  
 @Column(nullable = false)  
 private UUID likedByUserId;  
  
 @Column(nullable = false)  
 private UUID programOwnerId;  
  
 @Column(nullable = false)  
 private LocalDateTime likedAt;  
}

* Notification

@Entity  
@Builder  
@AllArgsConstructor  
@NoArgsConstructor  
@Getter  
@Setter  
@Table(name = "notifications")  
public class Notification {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*UUID*)  
 private UUID id;  
  
 @Column(nullable = false)  
 private UUID receiverId;  
  
 @Column(nullable = false)  
 private UUID senderId;  
  
 @Column(nullable = false)  
 private String message;  
  
 @Column(nullable = false)  
 private LocalDateTime createdAt;  
}

## Модели реализирани чрез MySQL заявки

create table programs  
(  
 id binary(16) not null  
 primary key,  
 created\_on datetime(6) not null,  
 description varchar(255) null,  
 difficulty enum ('BEGINNER', 'EASY', 'EXPERT', 'HARD', 'MEDIUM') not null,  
 name varchar(255) not null,  
 shared\_with\_others bit not null,  
 user\_id binary(16) not null  
);  
  
create table users  
(  
 id binary(16) not null  
 primary key,  
 bio varchar(255) null,  
 country enum ('BELGIUM', 'BRAZIL', 'BULGARIA', 'CANADA', 'CHINA', 'FRANCE', 'GERMANY', 'INDIA', 'ITALY', 'JAPAN', 'RUSSIA', 'USA') not null,  
 created\_on datetime(6) not null,  
 email varchar(255) not null,  
 first\_name varchar(255) null,  
 is\_active bit not null,  
 last\_name varchar(255) null,  
 level enum ('BEGINNER', 'COMPETENT', 'EXPERT', 'NOVICE', 'PREFER\_NOT\_TO\_SAY', 'PROFICIENT') not null,  
 password varchar(255) not null,  
 profile\_picture varchar(255) null,  
 updated\_on datetime(6) not null,  
 user\_role enum ('ADMIN', 'USER') not null,  
 username varchar(255) not null,  
 active\_program\_id binary(16) null,  
 constraint UK6dotkott2kjsp8vw4d0m25fb7  
 unique (email),  
 constraint UKr43af9ap4edm43mmtq01oddj6  
 unique (username),  
 constraint FKm5c3dhr0ei8ih8yo2m42wqsyg  
 foreign key (active\_program\_id) references programs (id)  
);  
  
create table exercises  
(  
 id binary(16) not null  
 primary key,  
 created\_on datetime(6) not null,  
 description varchar(255) not null,  
 difficulty enum ('EASY', 'HARD', 'MEDIUM') not null,  
 media\_url varchar(255) null,  
 name varchar(255) not null,  
 status enum ('APPROVED', 'PENDING', 'REJECTED') null,  
 approved\_by binary(16) null,  
 created\_by binary(16) null,  
 constraint UKc514au7a6fgk8ahyfws1y84jm  
 unique (name),  
 constraint FK431yca9kw6du2bn960q134hcl  
 foreign key (approved\_by) references users (id),  
 constraint FKp2kcs08pa87cbt6duijofnivm  
 foreign key (created\_by) references users (id)  
);  
  
create table exercise\_muscle\_groups  
(  
 exercise\_id binary(16) not null,  
 muscle\_group enum ('ABS', 'BACK', 'BICEPS', 'CALVES', 'CARDIO', 'CHEST', 'FOREARMS', 'FULL\_BODY', 'GLUTES', 'HAMSTRINGS', 'LOWER\_BACK', 'OBLIQUES', 'QUADS', 'SHOULDERS', 'TRICEPS') null,  
 constraint FK1m7k8dr25xe4dhaq0xlbjmvll  
 foreign key (exercise\_id) references exercises (id)  
);  
  
alter table programs  
 add constraint FK9k50s05eujcmatxpbo7vfwby0  
 foreign key (user\_id) references users (id);  
  
create table workouts  
(  
 id binary(16) not null  
 primary key,  
 additional\_info varchar(255) null,  
 duration int null,  
 added\_by\_id binary(16) null,  
 constraint FKon8sakxuhbbgilfp46qqqjsd8  
 foreign key (added\_by\_id) references users (id)  
);  
  
create table logs  
(  
 id binary(16) not null  
 primary key,  
 completion\_date datetime(6) null,  
 user\_id binary(16) not null,  
 workout\_id binary(16) not null,  
 constraint FKe3g8e0odt5e4h204epwchcal5  
 foreign key (workout\_id) references workouts (id),  
 constraint FKgqy8beil5y4almtq1tiyofije  
 foreign key (user\_id) references users (id)  
);  
  
create table logs\_exercises  
(  
 id binary(16) not null  
 primary key,  
 added\_weight double null,  
 completed\_reps int not null,  
 completed\_sets int not null,  
 exercise\_id binary(16) not null,  
 log\_id binary(16) not null,  
 constraint FKb3jp1ukfma6y0hg7auh0ph40m  
 foreign key (log\_id) references logs (id),  
 constraint FKbm6e94ehks1sbmouoswwyxcla  
 foreign key (exercise\_id) references exercises (id)  
);  
  
create table workout\_exercises  
(  
 id binary(16) not null  
 primary key,  
 added\_weight double null,  
 reps int null,  
 sets int null,  
 exercise\_id binary(16) not null,  
 workout\_id binary(16) not null,  
 constraint FK3bn8puhdk29j682es5643p60c  
 foreign key (exercise\_id) references exercises (id),  
 constraint FKd2ychryarm8gp13672lojwr78  
 foreign key (workout\_id) references workouts (id)  
);  
  
create table workout\_schedules  
(  
 id binary(16) not null  
 primary key,  
 day\_of\_week enum ('FRIDAY', 'MONDAY', 'SATURDAY', 'SUNDAY', 'THURSDAY', 'TUESDAY', 'WEDNESDAY') not null,  
 program\_id binary(16) not null,  
 workout\_id binary(16) not null,  
 constraint UKd8h8prfupul21alt4did8aben  
 unique (program\_id, day\_of\_week),  
 constraint FK3yaaul439mvn0toqkcsxshihp  
 foreign key (program\_id) references programs (id),  
 constraint FKq14r95j2a7ilvo9r1kcc246dr  
 foreign key (workout\_id) references workouts (id)  
);

create table likes  
(  
 id binary(16) not null  
 primary key,  
 liked\_at datetime(6) not null,  
 liked\_by\_user\_id binary(16) not null,  
 program\_id binary(16) not null,  
 program\_owner\_id binary(16) not null  
);  
  
create table notifications  
(  
 id binary(16) not null  
 primary key,  
 created\_at datetime(6) not null,  
 message varchar(255) not null,  
 receiver\_id binary(16) not null,  
 sender\_id binary(16) not null  
);

## Описание на полетата

### Таблица: users (Потребители)

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| id | binary(16) | Уникален идентификатор на потребителя (UUID). |
| bio | varchar | Кратка биография на потребителя. |
| country | enum | Държава на потребителя (ограничена до предварително зададени стойности). |
| created\_on | datetime | Дата на регистрация. |
| email | varchar | Имейл на потребителя – уникален |
| first\_name | varchar | Име на потребителя. |
| is\_active | bit | Флаг дали акаунтът е активен (1 = активен, 0 = неактивен). |
| last\_name | varchar | Фамилия на потребителя. |
| level | enum | Ниво на потребителя (например BEGINNER, EXPERT и др.). |
| password | varchar | Хеширана парола. |
| profile\_picture | varchar | URL до снимка на профила. |
| updated\_on | datetime | Последна дата на редакция на профила. |
| user\_role | enum | Роля на потребителя – USER или ADMIN. |
| username | varchar | Потребителско име – уникално. |
| active\_program\_id | binary(16) | Препратка към текущо активната програма. Външен ключ към programs(id). |

### Таблица: programs (Програми)

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| id | binary(16) | Уникален идентификатор на програмата. |
| created\_on | datetime | Дата на създаване. |
| description | varchar | Описание на програмата. |
| difficulty | enum | Трудност – BEGINNER, EASY, MEDIUM, HARD, EXPERT. |
| name | varchar | Име на програмата. |
| shared\_with\_others | bit | Флаг дали е споделена с други (1 = да, 0 = не). |
| user\_id | binary(16) | Потребител, който е създал програмата (външен ключ към users(id)). |

### Таблица: exercises (Упражнения)

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| id | binary(16) | Уникален идентификатор на упражнението. |
| created\_on | datetime | Дата на създаване. |
| description | varchar | Описание на упражнението. |
| difficulty | enum | Трудност на упражнението. |
| media\_url | varchar | Линк към изображение. |
| name | varchar | Име на упражнението. Уникално. |
| status | enum | Статус – PENDING, APPROVED или REJECTED. |
| approved\_by | binary(16) | Потребител, одобрил упражнението (външен ключ към users(id)). |
| created\_by | binary(16) | Създател на упражнението (външен ключ към users(id)). |

### Таблица: exercise\_muscle\_groups (Групи мускули към упражнения)

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| exercise\_id | binary(16) | Препратка към упражнение. |
| muscle\_group | enum | Мускулна група – ABS, CHEST, BACK и т.н. |

### Таблица: workouts (Тренировки)

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| id | binary(16) | Уникален идентификатор. |
| additional\_info | varchar | Допълнителна информация (например съвети, бележки). |
| duration | int | Продължителност в минути. |
| added\_by\_id | binary(16) | Потребител, който е добавил тренировката. |

### Таблица: logs (Логове за изпълнение на тренировки)

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| id | binary(16) | Уникален идентификатор. |
| completion\_date | datetime | Дата на приключване. |
| user\_id | binary(16) | Потребител, който е извършил тренировката. |
| workout\_id | binary(16) | Тренировката, която е била направена. |

### Таблица: logs\_exercises (Упражнения, изпълнени в логове)

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| id | binary(16) | Уникален идентификатор. |
| added\_weight | double | Допълнителна тежест (ако е използвана). |
| completed\_reps | int | Изпълнени повторения. |
| completed\_sets | int | Изпълнени серии. |
| exercise\_id | binary(16) | Упражнение, което е било направено. |
| log\_id | binary(16) | С коя лог-запис е свързано. |

### Таблица: workout\_exercises (Упражнения в тренировки)

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| id | binary(16) | Уникален идентификатор. |
| added\_weight | double | Тежест за това упражнение в тренировката. |
| reps | int | Брой повторения. |
| sets | int | Брой серии. |
| exercise\_id | binary(16) | Упражнение, което участва в тренировката. |
| workout\_id | binary(16) | Към коя тренировка принадлежи упражнението. |

### Таблица: workout\_schedules (График на програма)

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| id | binary(16) | Уникален идентификатор. |
| day\_of\_week | enum | Ден от седмицата – Понеделник, Вторник и т.н. |
| program\_id | binary(16) | Програма, за която важи графикът. |
| workout\_id | binary(16) | Тренировка, която е зададена за този ден. |

### Таблица: likes (Харесвания на програми)

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| id | binary(16) | Уникален идентификатор. |
| liked\_at | datetime | Дата и час на харесване. |
| liked\_by\_user\_id | binary(16) | Потребител, който е харесал програмата. |
| program\_id | binary(16) | Програмата, която е харесана. |
| program\_owner\_id | binary(16) | Собственикът на програмата. |

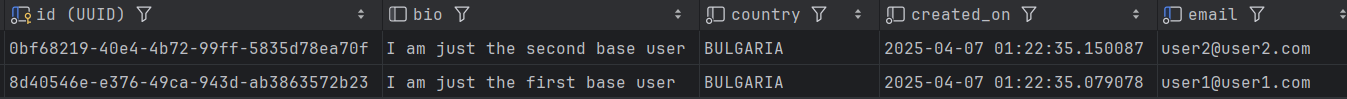
### Таблица: notifications (Известия)

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| id | binary(16) | Уникален идентификатор. |
| created\_at | datetime | Дата и час на изпращане. |
| message | varchar | Съдържанието на известието. |
| receiver\_id | binary(16) | Получател на известието. |
| sender\_id | binary(16) | Изпращач на известието. |

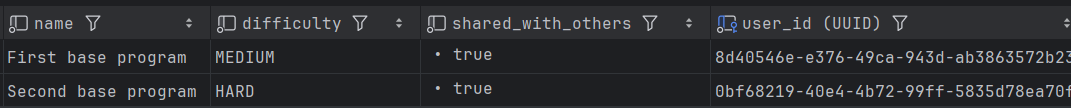
## Примери с реални данни

Приложени са примери за съхранената информация в базата от данни за някои от таблиците:

Users



Programs



Exercises

Картина, която съдържа текст, екранна снимка, Шрифт, номер

Генерираното от ИИ съдържание може да е неправилно.

# III. Избрани технологии за реализация

Реализацията на проекта се базира на съвременни и широко използвани технологии и инструменти за разработка на уеб приложения.

## Spring Boot – Java

Spring Boot e основната платформа за разработка на бекенд частта на приложението. Тя осигурява бърз старт и конфигурация на Spring приложения, като включва вграден уеб сървър, поддръжка на REST API и интеграция с бази данни.

## Spring Data JPA (Hibernate)

Spring Data JPA предоставя абстракция за работа с базата данни чрез обектно-релационно моделиране. Hibernate се използва като JPA провайдър, като позволява лесно създаване и управление на entities, релации между тях и автоматична синхронизация с базата.

## Spring MVC – Thymeleaf

Spring MVC е използван за управление на заявките и изграждане на контролери, докато Thymeleaf служи за сървърно рендериране на HTML шаблони. Това позволява създаване на динамични и интерактивни уеб страници, които отразяват състоянието на данните в реално време.

## Spring Security

Spring Security е използван за осигуряване на сигурност на приложението чрез автентикация и авторизация на потребителите. Технологията позволява контрол на достъпа до различни части на системата, както и управление на роли и права на потребителите. Използвана е и за защита от CSRF и други типични уязвимости.

## HTML, CSS и JavaScript

Фронтенд частта на приложението е изградена с помощта на HTML за структуриране на съдържанието, CSS за стилизиране и JavaScript за добавяне на интерактивни елементи и динамично поведение. В комбинация с Thymeleaf, тези технологии осигуряват плавно потребителско изживяване и визуална привлекателност.

## MySQL

Използвана е релационна база данни MySQL за съхранение на всички структурирани данни в приложението – потребители, упражнения, програми, лайкове и др. Базата е проектирана с ясни релации и нормализация с цел избягване на дублиране и поддържане на целостта на данните.

## DataGrip

DataGrip е използван като среда за визуализация и управление на базата данни по време на разработката. Той е особено полезен при създаване на заявки, тестване на данни, преглед на таблици и дебъгване на структурата на базата.

# IV. Описание на отделните части на системата

## Микросървисна архитектура

Системата е изградена с подход, наподобяващ микросървисна архитектура, при който отделни функционалности са разделени в независими модули с ясно дефинирани отговорности. Основният пример в проекта е **лайк микросървисът**, отговорен за управлението на харесванията върху програми и изпращане на нотификации към потребителите.

Този микросървис е самостоятелен модул, който комуникира с основното приложение чрез HTTP заявки – Feign Client. Това осигурява:

* По-добра мащабируемост и модулност
* Лесна поддръжка и възможност за бъдещо разширяване
* Разделяне на отговорностите между различни компоненти на системата

## Модулна структура в отделни пакети

Основната част на приложението е организирана в добре структурирани пакети, които следват принципите на **чистата архитектура** и **разделение на отговорности (Separation of Concerns)**. Всеки пакет обслужва специфична част от логиката на системата:

* controller – обработка на HTTP заявки и управление на пренасочвания и изгледи
* service – бизнес логика и обработка на заявки към базата данни
* repository– интерфейси за достъп до базата чрез Spring Data JPA
* entity – класове, отговарящи на таблиците в базата и свързани чрез JPA анотации
* dto (Data Transfer Objects) – модели за трансфер на данни между слоевете
* config – конфигурационни класове за сигурност и други настройки
* utils – помощни класове и методи, използвани на различни места в приложението

Тази модулна структура прави приложението:

* Лесно за четене и разбиране
* Гъвкаво при разработка и разширение
* Тестируемо и подредено според добрите практики

Всеки модул отговаря за конкретна функционалност и е разделен в отделни пакети:

Картина, която съдържа текст, екранна снимка, Шрифт

Генерираното от ИИ съдържание може да е неправилно. Картина, която съдържа текст, екранна снимка, Шрифт

Генерираното от ИИ съдържание може да е неправилно. Картина, която съдържа текст, екранна снимка, Шрифт, дизайн

Генерираното от ИИ съдържание може да е неправилно.

# V. Ръководство за потребителя

## Потребител

Картина, която съдържа текст, екранна снимка, Шрифт

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, Шрифт, лого

Генерираното от ИИ съдържание може да е неправилно. Картина, която съдържа текст, екранна снимка, Шрифт, дизайн

Генерираното от ИИ съдържание може да е неправилно. Картина, която съдържа текст, екранна снимка, Шрифт, номер

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, Шрифт, дизайн

Генерираното от ИИ съдържание може да е неправилно. Картина, която съдържа Графика, Шрифт, лого, графичен дизайн

Генерираното от ИИ съдържание може да е неправилно. Картина, която съдържа Графика, дизайн, лого

Генерираното от ИИ съдържание може да е неправилно.

Обикновеният потребител има достъп до основните функционалности на системата, които му позволяват да създава и управлява свои фитнес програми. Възможностите включват:

* Регистрация и вход в системата  
  Потребителят създава свой профил чрез регистрационна форма и впоследствие се идентифицира с потребителско име и парола.

Картина, която съдържа текст, екранна снимка, Операционна система, софтуер

Генерираното от ИИ съдържание може да е неправилно. Картина, която съдържа текст, екранна снимка, Шрифт

Генерираното от ИИ съдържание може да е неправилно.

* Създаване на упражнения  
  Потребителят може да създава собствени упражнения, които преминават през процес на одобрение от администратор. След одобрение, те стават достъпни и за други потребители.

Картина, която съдържа текст, екранна снимка, Уеб страница, софтуер

Генерираното от ИИ съдържание може да е неправилно.

* Създаване на тренировки  
  Създаване на тренировъчни сесии, като се комбинират различни упражнения с конкретни параметри (серии, повторения, тежест). Всички тренировки са лични, други потребители не могат да ги достъпват.

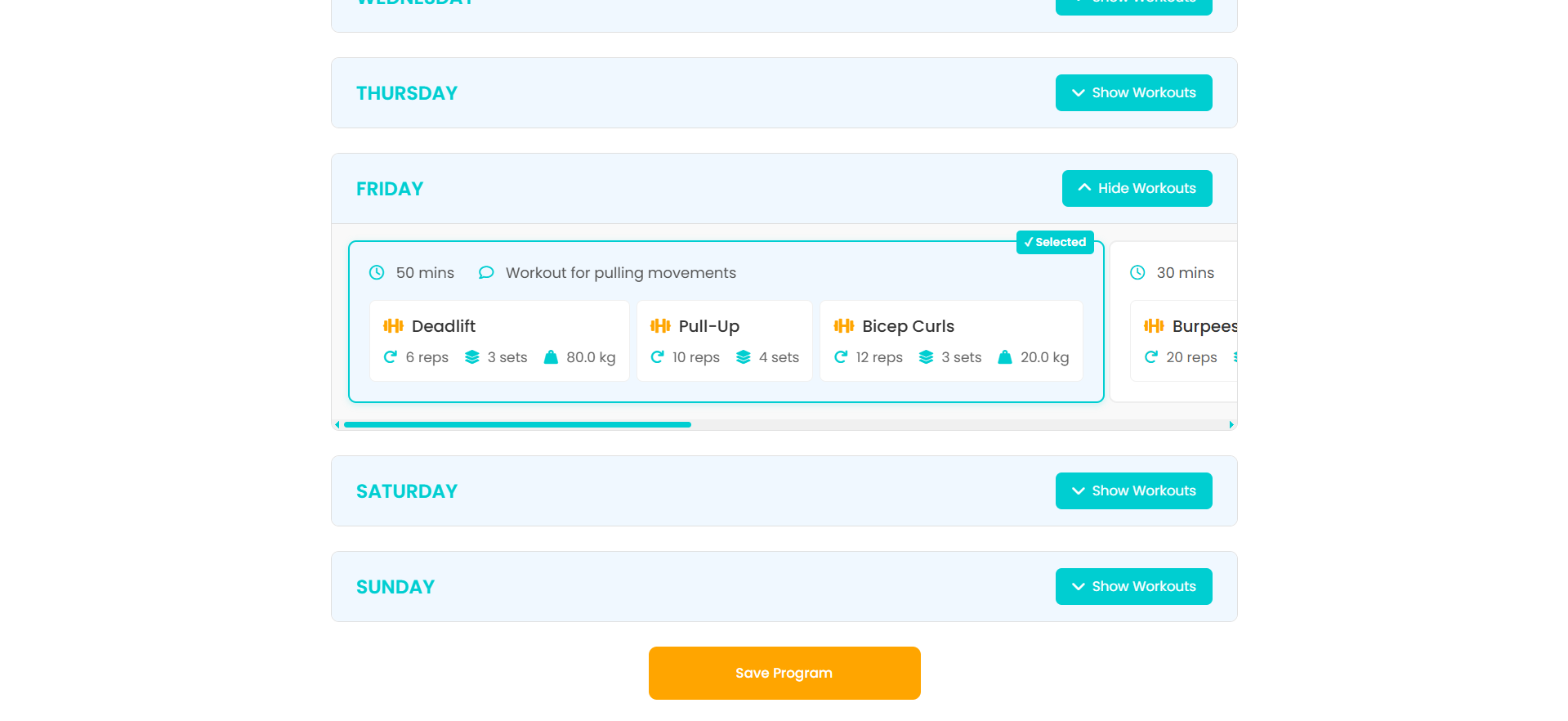
Картина, която съдържа текст, екранна снимка, софтуер, Уеб страница

Генерираното от ИИ съдържание може да е неправилно.

* Създаване на програми  
  Потребителят съставя програми, които се състоят от тренировки за различните дни от седмицата. Програмите могат да бъдат частни или споделени с други потребители чрез Community секцията. Програмата се състои от 1-7 тренировки за всеки ден от седмицата, като може да се оставят празни дни предназначени за почивка.

Картина, която съдържа текст, екранна снимка, софтуер, Уеб страница

Генерираното от ИИ съдържание може да е неправилно.



* Преглед на програми от други потребители  
  Чрез таба *Community*, потребителят може да разглежда споделени програми и да ги харесва.

Картина, която съдържа текст, Шрифт, екранна снимка

Генерираното от ИИ съдържание може да е неправилно.

Програмата може да се разгъне:

Картина, която съдържа текст, екранна снимка, софтуер, Уеб страница

Генерираното от ИИ съдържание може да е неправилно.

* Картина, която съдържа символ, лого, Шрифт, Графика

  Генерираното от ИИ съдържание може да е неправилно.Харесване на програми  
  Потребителите могат да харесват програми, което задейства микросървиса за нотификации и изпраща известие до автора на програмата. Програма може да се хареса от browse programs страницата.

Картина, която съдържа текст, Шрифт, лого, Графика

Генерираното от ИИ съдържание може да е неправилно.Картина, която съдържа текст, екранна снимка, Шрифт, бял

Генерираното от ИИ съдържание може да е неправилно.

* Логване на тренировки  
  След избор на активна програма, потребителят може да запише резултатите си за даден тренировъчен ден.

Активната програма се показва на главната страница:

Картина, която съдържа текст, екранна снимка, софтуер, Уеб страница

Генерираното от ИИ съдържание може да е неправилно.

След това чрез бутона Log можем да навигираме към страницата за логване:

Картина, която съдържа текст, софтуер, Уеб страница, Уебсайт

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, Шрифт, номер

Генерираното от ИИ съдържание може да е неправилно.След като запишем тренировката си можем да я прегледаме от My Activities > Logs

Картина, която съдържа текст, екранна снимка, Шрифт, дизайн

Генерираното от ИИ съдържание може да е неправилно.

* Преглед на всички одобрени упражнения  
  Всеки потребител може да разглежда списък с всички упражнения, които са одобрени и достъпни за използване в тренировки.

Картина, която съдържа текст, Уеб страница, софтуер, Уебсайт

Генерираното от ИИ съдържание може да е неправилно.

* Преглед на собствения профил  
  Потребителят има достъп до собствената си профилна страница.

Картина, която съдържа текст, екранна снимка, Шрифт, Уеб страница

Генерираното от ИИ съдържание може да е неправилно.

* Редакция на профил  
  Възможност за промяна на лична информация като име и други настройки.
* Преглед на профилите на други потребители  
  Чрез Community секцията потребителят може да отваря профили на други, за да види техните публично споделени програми.

Картина, която съдържа текст, екранна снимка, Уеб страница, Уебсайт

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, софтуер, Компютърна икона, Уеб страница

Генерираното от ИИ съдържание може да е неправилно. Програмата може да се разгъне:

Картина, която съдържа текст, екранна снимка, софтуер, Уеб страница

Генерираното от ИИ съдържание може да е неправилно.

* Преглед на качените упражнения  
  Потребителят може да види кои упражнения е качил и какъв е статусът им (одобрено, чака одобрение, отхвърлено).

Картина, която съдържа текст, Шрифт, Уеб страница, екранна снимка

Генерираното от ИИ съдържание може да е неправилно.

* Преглед на създадените програми  
  Достъп до списък със собствени програми.

Картина, която съдържа текст, екранна снимка, Шрифт, софтуер

Генерираното от ИИ съдържание може да е неправилно.

* Преглед на създадените тренировки

Достъп до списък със собствени тренировки.

Картина, която съдържа текст, софтуер, Уеб страница, Уебсайт

Генерираното от ИИ съдържание може да е неправилно.

### Преглед на харесаните програми:

Картина, която съдържа текст, екранна снимка, софтуер, Уеб страница

Генерираното от ИИ съдържание може да е неправилно.

## Администратор

Картина, която съдържа текст, екранна снимка, Шрифт, дизайн

Генерираното от ИИ съдържание може да е неправилно.

Администраторът има разширени права и достъп до допълнителна функционалност, свързана с управлението на платформата:

* Одобрение или отхвърляне на упражнения  
  Администраторите могат да преглеждат подадените упражнения от потребители и да ги одобряват или отхвърлят, според преценката си. – spodeli za procedurite I scheduler

Картина, която съдържа текст, софтуер, Уеб страница, Компютърна икона

Генерираното от ИИ съдържание може да е неправилно.

Картина, която съдържа текст, екранна снимка, Шрифт, номер

Генерираното от ИИ съдържание може да е неправилно.

В базата данни всяко упражнение има поле със статус (например: PENDING, APPROVED, REJECTED). Това поле се използва за филтриране и визуализиране на упражненията според текущото им състояние.

#### Използване на съхранени процедури за масово одобрение/отхвърляне на упражнения

В приложението са реализирани съхранени процедури в MySQL, които позволяват бързо и ефективно масово одобрение или отхвърляне на всички чакащи упражнения (т.е. със статус PENDING). Това е полезно при администраторска поддръжка, особено при голям обем подадени упражнения.

Реализация в базата данни

В MySQL са създадени следните процедури:

CREATE PROCEDURE *ApproveAllPendingExercises*()  
BEGIN  
 UPDATE exercises SET status = 'APPROVED' WHERE status = 'PENDING';  
END;

CREATE PROCEDURE *RejectAllPendingExercises*()  
BEGIN  
 UPDATE exercises SET status = 'REJECTED' WHERE status = 'PENDING';  
END;

Тези процедури се грижат за това с едно извикване да се промени статусът на всички упражнения, които все още не са били прегледани.

Връзка със Spring Boot - Съответстващите методи в Java кода използват анотацията @Procedure, която позволява извикване на SQL процедури директно от Spring Data JPA repository:

@Procedure(procedureName = "ApproveAllPendingExercises")  
void approveAllPendingExercises();  
  
@Procedure(procedureName = "RejectAllPendingExercises")  
void rejectAllPendingExercises();

Това позволява на администратора с един бутон или команда от бекенда да приложи дадената процедура, без да се пише допълнителна логика на Java ниво.

Картина, която съдържа текст, Уеб страница, Шрифт, софтуер

Генерираното от ИИ съдържание може да е неправилно.

#### Scheduler (Spring Task Scheduler)

За автоматизация на някои процеси е внедрен Spring Scheduler – това е функционалност, която позволява изпълнението на планирани задачи във фонов режим.

* Автоматично се изтриват упражнения, които са били отхвърлени преди повече от 15 минути.

Използва се анотацията @Scheduled в Spring Boot, с конфигурация за периодичност.

@Component  
@Slf4j  
public class DeleteRejectedExercisesScheduler {  
 private final ExerciseService exerciseService;  
  
 @Autowired  
 public DeleteRejectedExercisesScheduler(ExerciseService exerciseService) {  
 this.exerciseService = exerciseService;  
 }  
  
 *//After every 15 minutes delete all rejected exercises* @Scheduled(fixedRate = 900000)*//15min* public void deleteRejectedExercises() {  
 exerciseService.deleteAllRejectedExercises();  
  
 *log*.info("All rejected exercises have been deleted");  
 }  
}

* Управление на потребителски профили  
  Администраторът има възможност да деактивира профили или да повиши даден потребител до администраторско ниво.

Картина, която съдържа текст, екранна снимка, Шрифт

Генерираното от ИИ съдържание може да е неправилно.





* Редакция на профил  
  Възможност за промяна на лична информация като име и други настройки.

### Преглед на качените програми и упражнения

Тези административни действия гарантират контрол върху съдържанието на платформата и поддържат нейното качество и сигурност.

# VI. Примерна работа и описание на функционалност

В тази част се представя примерна начална конфигурация на системата с предварително добавени потребители, упражнения, тренировки и програми, както и демонстрация на основните функционалности на приложението.

## Начална конфигурация

Системата зарежда примерни данни чрез CommandLineRunner компоненти при стартиране на приложението. Това включва:

* Два обикновени потребителя:
  + user1 / user1
  + user2 / user2
* Един администратор:
  + admin / admin
* Упражнения:
  + Част от упражненията са въведени и одобрени от администратора.
  + Едно упражнение е качено от потребител (user1) и чака одобрение.
* Тренировки:
  + И двамата потребители имат създадени индивидуални тренировки с избрани упражнения и параметри за серии, повторения и тежест.
* Програми:
  + Всеки потребител има по една създадена и споделена програма, достъпна за други потребители чрез секцията Community или от профилната страница.
* Лайкове и нотификации:
  + user2 харесва програмата на user1.
  + В резултат, user1 получава известие чрез микросървис за лайкове.

## Демонстрация на основните функционалности

Регистрация и вход

* Нов потребител може да се регистрира и впоследствие да влезе в системата.

Създаване и преглед на упражнения

* Потребителите могат да предложат нови упражнения.
* Администраторът има възможност да преглежда, одобрява или отхвърля предложенията.
* Одобрените упражнения стават видими за всички.

Създаване на тренировки

* Потребителят може да създаде тренировка като избере налични упражнения и зададе параметри (брой серии, повторения, тежест).
* Тренировките са видими само за създателя им.

Създаване и споделяне на програми

* Потребителите могат да създават седмични програми, като разпределят тренировките по дни.
* Програмите могат да бъдат маркирани като "споделени", за да са видими от други.

Преглед на програми от други потребители

* В секцията Community или в профила на конкретен потребител могат да се разглеждат публично споделени програми.

Харесване и известия

* Всеки потребител може да харесва чужди програми.
* При харесване се изпраща известие до собственика на програмата чрез отделен микросървис.

Логване на тренировъчни резултати

* Потребителят може да избере активна програма и да въвежда резултати от изпълнени тренировки: серии, повторения и тежест.

# VII. Възможности за бъдещо развитие

Системата е проектирана с възможност за лесно разширяване, като в бъдещи версии могат да бъдат реализирани следните подобрения и допълнителни функционалности:

## Разширени възможности за социална интеракция

* Коментиране на програми – потребителите да могат да оставят обратна връзка и коментари.
* Система за съобщения – чат между потребителите за обмяна на идеи, програми и мотивация.
* Следване на потребители – функционалност за "Follow", при която потребителят получава известия, когато следен потребител качи нова програма или тренировка.

## Мобилна версия или мобилно приложение

* Разработка на мобилно приложение (Android/iOS) с основните функционалности.
* Отделен мобилен UI, адаптиран за тъч устройства.

## Подобрена статистика и анализи

* Визуализация на напредъка – графики и таблици със статистика за логнати тренировки.
* Филтри и търсене в Community – по категория, брой харесвания, трудност и други.

## Поддръжка на повече типове съдържание

* Видео демонстрации на упражнения.

## Персонализирани препоръки

* Препоръчване на програми въз основа на историята на потребителя, харесвания и предпочитания.

## Интеграция с външни устройства

* Синхронизация с фитнес тракери и устройства като Garmin, Fitbit, Apple Watch за автоматично логване на тренировки.

## Модул за администраторска статистика

* Панел за администратори с данни за активност на потребители, популярност на упражнения, брой регистрирани и др.

# VIII. Заключение и обобщение

Разработеното уеб-базирано приложение представлява цялостна система за управление на фитнес програми, обогатена с функционалности, типични за социалните мрежи. То дава възможност на потребителите да създават и персонализират упражнения, тренировки и програми, както и да ги споделят с други. Интегрирани са механизми за харесване, известяване и логване на резултати, което стимулира ангажираност и проследимост на напредъка.

Системата демонстрира прилагането на ключови принципи от проектирането на релационни бази данни и използва ORM подход (Hibernate) за ефективно управление на данните. Внедрени са различни нива на достъп и ролеви модели, позволяващи адекватно разграничаване между потребители и администратори.

Чрез микросървисна архитектура и използване на съвременни технологии като Spring Boot, MySQL, Thymeleaf и JavaScript, проектът е структуриран за мащабируемост, гъвкавост и лесна интеграция с външни системи. Приложението е предварително заредено с тестови данни, което улеснява демонстрацията и оценката му.

Като резултат, системата успешно съчетава надеждност, функционалност и възможности за бъдещо развитие. Тя е подходяща не само за лична употреба, но и като основа за по-широкообхватна фитнес платформа, интегрираща социални и тренировъчни функционалности в едно цялостно решение.