САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №2

Выполнил:

Акулов Даниил

К3339

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

Задание:

1. Реализовать все модели данных, спроектированные в рамках ДЗ1
2. Реализовать набор из CRUD-методов для работы с моделями данных средствами Express + TypeScript
3. Реализовать API-эндпоинт для получения пользователя по id/email

Ход работы

Выбран вариант №2: Платформа для фитнес-тренировок и здоровья:

- Вход
- Регистрация
- Личный кабинет пользователя (трекинг прогресса, планы тренировок)
- Поиск тренировок с фильтрацией по уровню, типу (кардио, силовые) и продолжительности
- Страница тренировки с видео, описанием и инструкциями
- Блог о здоровье и питании

Схема бд:

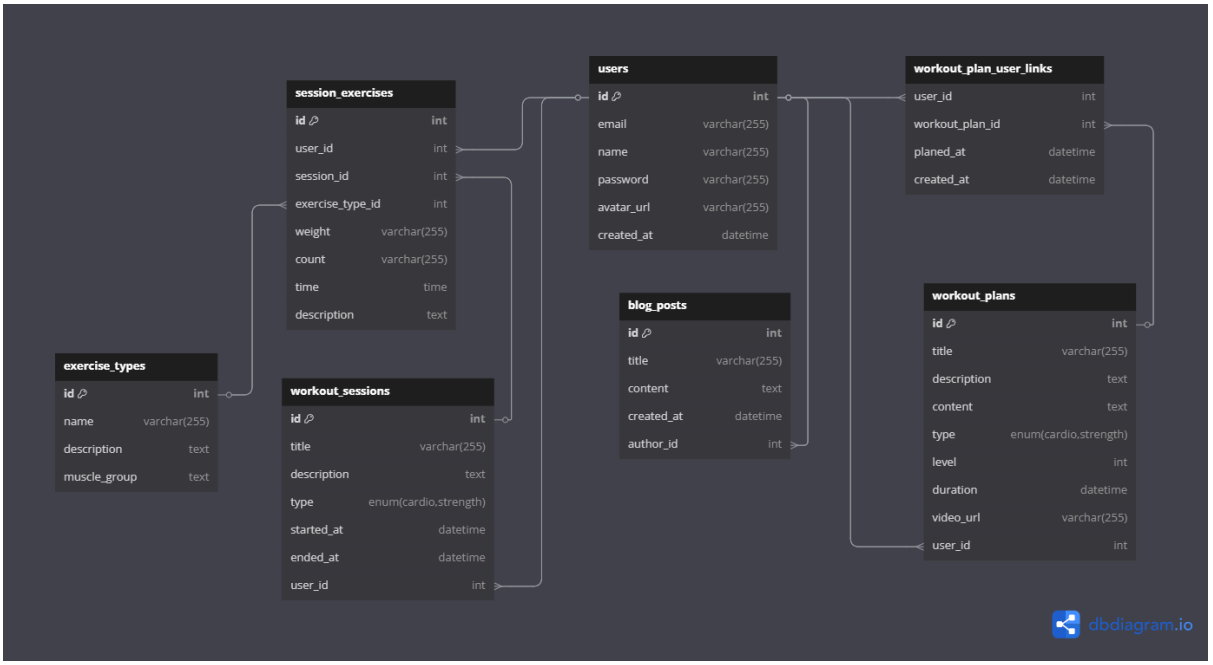

Рисунок 1 - Схема базы данных

На основе схемы базы данных были созданы модели. Пример реализации модели User:

```
@Entity("users")
export class User {
   @PrimaryGeneratedColumn()
   id: number;

   @Column({type: "varchar", length: 256, unique: true})
   email: string;

   @Column({type: "varchar", length: 256})
   password: string;

   @Column({type: "varchar", length: 256})
   avatarUrl: string;

   @Column({type: 'varchar', length: 256})
   name: string;

   @CreateDateColumn()
   createdAt: Date;

        @OneToMany(()   =>   SessionExercise,   sessionExercise   =>
sessionExercise.user)
   sessionExercises: SessionExercise[];

         @OneToMany(()   =>   WorkoutSession,   workoutSession   =>
workoutSession.user)
   workoutSessions: WorkoutSession[];

   @OneToMany(() => BlogPost, blogPost => blogPost.author)
   blogPosts: BlogPost[];
   @OneToMany(() => WorkoutPlan, workoutPlan => workoutPlan.user)
   workoutPlans: WorkoutPlan[];
    @OneToMany(() => WorkoutPlanUserLink, workoutPlanUserLinks =>
workoutPlanUserLinks.user)
   workoutPlanUserLinks: WorkoutPlanUserLink[];
}
```

Реализованы все CRUD-методы. Листинг части контроллера для модели User:

```
const userRepo = dataSource.getRepository(User);
const workoutPlanUserLinkRepo = dataSource.getRepository(WorkoutPlanUserLink);

class UserController {
   getAll: RequestHandler = async (req, res, next) => {
       try {
           const users = await userRepo.find();
           return res.json({users});
       } catch (e) {
           next(ApiError.internal());
       }
   }
```

Реализованы API-эндпоинты для регистрации, авторизации и получения информации о пользователе по id. Листинг роутера для модели User:

```
const router = express.Router();

router.get('/get-all', userController.getAll)
router.get('/get-one/:id', userController.getOne)
router.put('/', checkAuth, userController.update)
router.delete('/', checkAuth, userController.delete)

export default router;
```

Листинг index.ts файла:

```
const app = express();

const PORT = process.env.API_PORT || 8000

app.use(express.json());
app.use('/api', router);
app.use(errorHandler);

dataSource
   .initialize()
   .then(() => {
```

```
        console.log('Data Source has been initialized!');
            app.listen(PORT, () => console.log(`Server started on
http://localhost:${PORT}`));
    })
    .catch((err) => {
          console.error('Error during Data Source initialization:',
err);
    });
```

Вывод

В рамках работы были созданы модели, спроектированные в рамках Д31, для моделей были созданы сервисы, в которых содержатся CRUD-методы, работа которых реализована при помощи TypeORM; контроллеры и роутеры. Вся работа велась при помощи express и TypeScript.