САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа № 2

Выполнил:

Гуторова Инна

Группа К3341

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

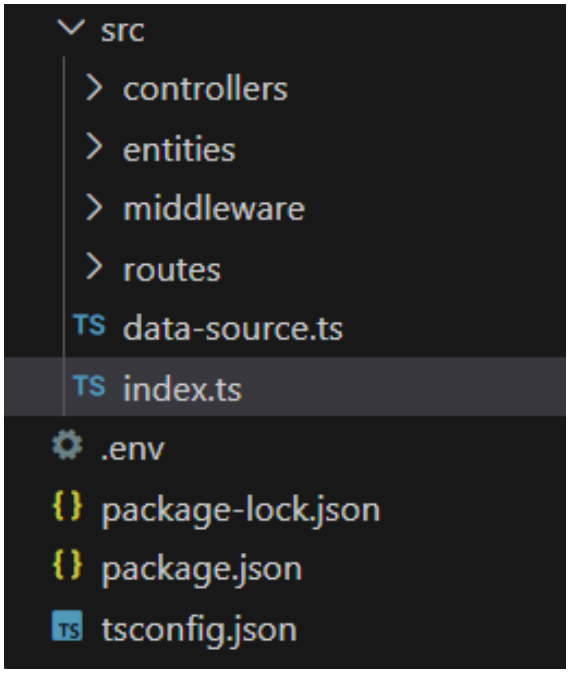
Задача

По выбранному варианту необходимо будет реализовать RESTful API средствами express + typescript (используя ранее написанный boilerplate).

Ход работы

Так как в предыдущих заданиях были реализованы boilerplate с авторизацией и основные crud-ы по варианту, основная задача этой работы – объединить их и немного расширить.

Структура проекта такая же, как и в boilerplate:



Были перенесены все сущности, контроллеры и роуты по варианту.

Необходимо было адаптировать юзера, чтобы он был совместим с boilerplate, но и хранил всю информацию по варианту.

Итоговый юзер (теперь хранит hashed_password):

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

    @Column({ unique: true })
    username!: string;

    @Column({ unique: true })
    email!: string;

    @Column()
    hashed_password!: string;

    @Column({ nullable: true })
    first_name?: string;

    @Column({ nullable: true })
    last_name?: string;

    @Column({ type: 'timestamp', default: () => 'CURRENT_TIMESTAMP' })
    registration_date!: Date;

    @Column({ default: false })
    isAdmin!: boolean;

    @OneToMany(() => Booking, booking => booking.user)
    bookings!: Booking[];

    @OneToMany(() => Favorite, favorite => favorite.user)
    favorites!: Favorite[];

    @OneToMany(() => Comment, comment => comment.user)
    comments!: Comment[];

    @OneToMany(() => Review, review => review.user)
    reviews!: Review[];
```

Также обновлен контроллер и роуты, добавлены методы редактирования и удаления.

Реализовано разделение прав – есть пользователи-админы, у них есть доступ ко всем функциям.

Middleware/authAdm.ts:

```
export const authorizeAdmin = async (req: Request, res: Response, next: NextFunction) => {
  const userId = (req as any).userId;
  if (!userId) {
    return res.status(401).json({ message: "Unauthorized" });
  }

  try {
    const userRepo = AppDataSource.getRepository(User);
    const user = await userRepo.findOneBy({ id: userId });

    if (!user || !user.isAdmin) {
      return res.status(403).json({ message: "Access denied: Admins only" });
    }

    next();
  } catch (error) {
    return res.status(500).json({ message: "Server error" });
  }
};
```

Обновлены роуты всех сущностей, добавлены ограничения прав. Пример:

```
import { authenticate } from "../middleware/auth";
import { authorizeAdmin } from "../middleware/authAdm";

const attractionRouter = Router();

attractionRouter.post('/', authenticate, authorizeAdmin, createAttraction);
attractionRouter.get('/', getAttractions);
attractionRouter.get('/:id', getAttractionById);
attractionRouter.put('/:id', authenticate, authorizeAdmin, updateAttraction);
attractionRouter.delete('/:id', authenticate, authorizeAdmin, deleteAttraction);

export default attractionRouter;
```

Проверены сценарии:

- Регистрация/авторизация пользователя.
- Доступ к защищенным endpoints (например, удаление пользователя — только для админов).
- Обработка ошибок (неверный токен, недостаток прав).

Вывод

Реализовано рабочее REST API с 10+ сущностями. Интегрирована система аутентификации и авторизации. Настроены роли пользователей (администратор/пользователь). API соответствует требованиям REST, поддерживает все запланированные функции и готово к дальнейшему расширению.