САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бек-энд разработка

Отчет

Лабораторная работа “REST, RESTful, SOAP, GraphQL”

Выполнили:
Соколовская Арина
Пронина Мария

К333392

Проверил:
Добряков Д. И.

Санкт-Петербург

2024 г.

Задание

В рамках данной лабораторной работы Вам предложено выбрать один из нескольких вариантов. Выбранный вариант останется единым на весь курс и будет использоваться в последующих лабораторных работах.

По выбранному варианту необходимо будет реализовать RESTful API средствами express + typescript

1. Платформа для проведения онлайн-хакатонов (пример: https://devpost.com)

Есть несколько сущностей: жюри хакатона, участники, главный администратор, кураторы задач. У участников есть возможность выбрать одну из задач (регистрируется и имеет доступ к системе только капитан команды), после выбора задачи капитан может предложить решение, скачать какие-то файлы, которые ему предложены, посмотреть на ссылки, которые есть в задаче.

Ссылки и файлы к задачам добавляют кураторы задач через отдельный админский интерфейс, кроме того у них есть доступ к решениям, как и у членов жюри. Куратор может назначаться только на одну задачу и проводить консультации (например, в Zoom, ссылку на консультацию он крепит к самой задаче и это выводится у команды в ЛК). Жюри может оценивать решения участников, с комментариями, сортировать решения по дате публикации.

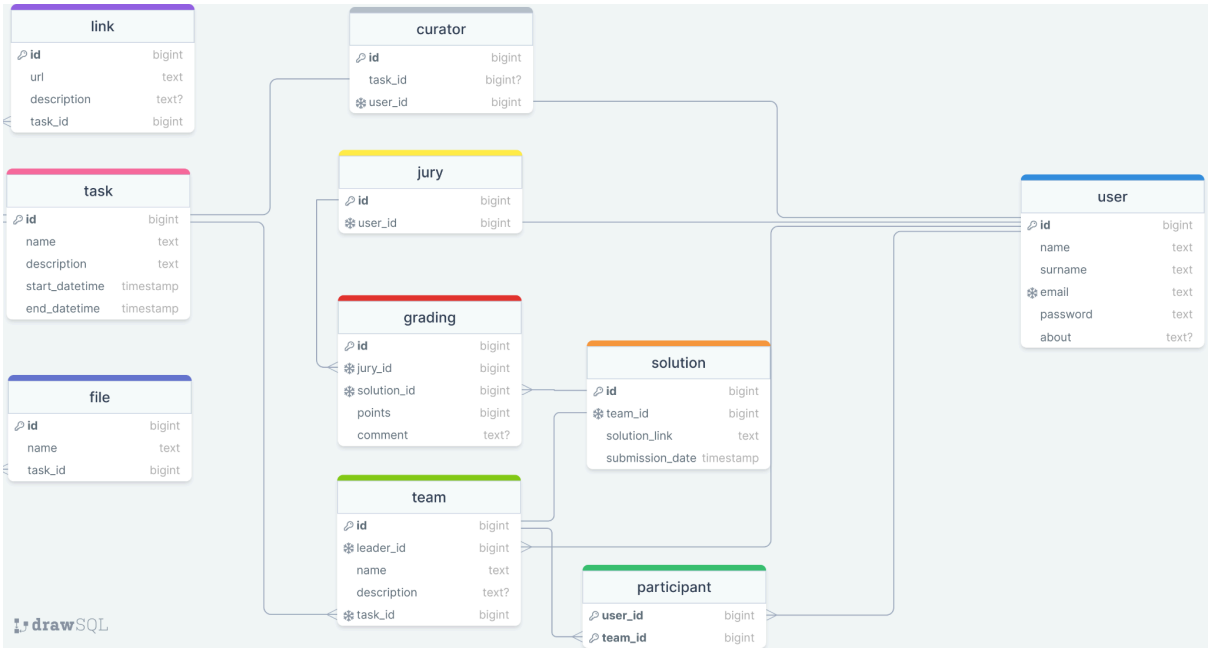
Капитан при регистрации заполняет только свои учётные данные, после в кабинете команды — он может заполнить данные по каждому участнику, название команды и какой-нибудь условный девиз/описание.

У главного админа есть доступ ко всему, но он не может добавлять команды и редактировать их решения. Так же, не имеет права оценивать решения участников. Только просматривать. Ещё он может создавать задачи, которые потом будут дополнять кураторы. Ну и назначать кураторов на задачи, разумеется.

Ход работы

1. Модель базы данных

Первым делом мы спроектировали модель данных, которая бы обладала нужным по варианту функционалом



Далее по этой модели мы описали sequelize модели.

2. Создание необходимых роутов

Были созданы роуты для:

- добавления, изменения хакатона,
- добавления, изменения решения,
- добавления юзера,
- добавления кураторов, жюри,
- выставление оценок.

```
const router = Router();

router.use('', homeRouter);
router.use('/users', userRouter);
router.use('/admin', adminRouter);
router.use('/auth', authRouter);
router.use('/gradings', gradingRouter);
router.use('/hackatons', hackathonRouter);
router.use('/solutions', solutionRouter);
router.use('/teams', teamRouter);

export default router;
```

3. Создание репозиториев и сервисов

Для взаимодействия с БД мы использовали repositoryMode.

```
export class HackathonRepository {
    private repository = sequelize.getRepository(Hackathon);

    async findByPks(ids: number[]): Promise<Hackathon[] | null> {
        const hackathons = await this.repository.findAll({
            where: {
                id: ids,
            },
        })
        return hackathons;
    }

    async findAllExcluding(attributes: string[]): Promise<Hackathon[] | []> {
        const hackathons = await this.repository.findAll({
            attributes: {
                exclude: attributes,
            }
        });
        return hackathons;
    }
```

4. Создание БД

```
export class UserService {
    private participantRepository: ParticipantRepository;
    private teamRepository: TeamRepository;
    private hackathonRepository: HackathonRepository;
    private userRepository: UserRepository;


    constructor() {
        this.participantRepository = new ParticipantRepository();
        this.teamRepository = new TeamRepository();
        this.hackathonRepository = new HackathonRepository();
        this.userRepository = new UserRepository();
    }

    async findUserHackathons(id: number): Promise<Hackathon[]> {
        const participations = await this.participantRepository.findbyUserId(id);
        if (participations == null) return [];
        const teams = await this.teamRepository.findByPks(participations.map(({team_id}) => (team_id)));
        if (teams == null) return [];
        const hackathons = await this.hackathonRepository.findByPks(teams.map(({task_id}) => (task_id)));
        if (hackathons == null) return [];
        return hackathons;
        return [];
    }
```

5. Добавление JWT токенов

```
const generateAcessToken = (id: number, role_name: string) => {
    const payload = { id, role_name };
    return jwt.sign(payload, process.env.secret_key as string, { expiresIn: "24h" });

}
```

6. Создание middleware

```
import { Request, Response } from "express"
import jwt from "jsonwebtoken"

module.exports = function (req: Request, res: Response, next: any) {
    try {
        if (!req.headers.authorization) {
            return res.status(403).json({message: "Unauthorized"})
        }
        const token = req.headers.authorization.split(' ')[1]
        if (!token) {
            return res.status(403).json({message: "Unauthorized"})
        }

        const parsed = jwt.verify(token, process.env.secret_key as string);
        (req as any).user = parsed;
        next()

    } catch(e) {
        if (e instanceof Error) {
            return res.json(e.message)
        }
    }
}       You, 3 days ago • feat: добавлены jwt и authmiddleware
```

```
import { Request, Response } from "express"
import jwt from "jsonwebtoken"
import { User } from "../model/user";

module.exports = function(roles: string[]) {
    return function(req: Request, res: Response, next: any) {
        try {
            if (!req.headers.authorization) {
                return res.status(403).json({message: "Unauthorized"});
            }
            const token = req.headers.authorization.split(' ')[1]
            if (!token) {
                return res.status(403).json({message: "Unauthorized"});
            }

            const parsed = jwt.verify(token, process.env.secret_key as string);
            const user_role = (parsed as User).role_name;
            if (!roles.includes(user_role)){
                return res.status(403).json({message: "Access Denied"});
            }
      You, 2 days ago • feat: добавлены middleware для ролей и капитанов команд…
            next();

        } catch(e) {
            if (e instanceof Error) {
                return res.json(e.message);
            }
        }
    };
}
```

Вывод:

В данной лабораторной работе был реализован сервис для проведения хакатонов с помощью express + sequelize.