САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бек-энд разработка

Отчет

Лабораторная работа 2

Выполнил:

Старовойтова Елизавета

Группа:

К33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2024 г.

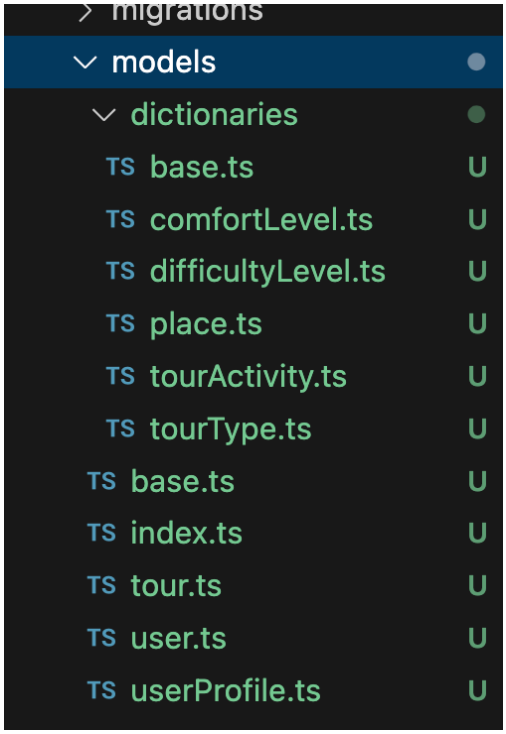
Задача

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

Вариант: Сервис-помощник в планировании путешествий.

Ход работы

Создаем модели.



На примере тура.

```
class Tour extends BaseModel {
    @Column
    name: string
    @Column
    price: number
    @Column
    canGoWithChildren: boolean
    @Column
    maxPeople: number

    @Column
    @ForeignKey(() => ComfortLevel)
    comfortLevelId: number
    @Column
    @ForeignKey(() => DifficultyLevel)
    difficultyLevelId: number
    @Column
    @ForeignKey(() => Place)
    placeId: number

    @BelongsToMany(() => TourActivity, {
        through: 'tour_has_tour_activity
        foreignKey: 'tourId',
        otherKey: 'tourActivityId',
    })
    tourActivities: Array<TourActivity>
    @BelongsToMany(() => TourType, {
        through: 'tour_has_tour_type',
        foreignKey: 'tourId',
        otherKey: 'tourTypeId',
    })
    tourTypes: Array<TourType>

    @BelongsTo(() => ComfortLevel)
    comfortLevel: ComfortLevel
    @BelongsTo(() => DifficultyLevel)
    difficultyLevel: DifficultyLevel
    @BelongsTo(() => Place)
    place: Place
}
```

Создаем контроллеры. Пример с туром.

```
find = async (request: express.Request, response: express.Response, next: NextFunction) => {
    try {
        const tour: Tour = await FindTourUseCase.run(Number(request.params.id))
        ApiResponse.payload(response, transform(tour, new TourTransformer()))
    } catch (e: any) {
        next(e)
    }
}

store = async (request: express.Request, response: express.Response, next: NextFunction) => {
    try {
        await this.validate(request, [
            body('name').notEmpty().isString(),
            body('price').notEmpty().isInt(),
            body('canGoWithChildren').notEmpty().isBoolean(),
            body('maxPeople').notEmpty().isInt(),
            body('comfortLevelId').notEmpty().isInt(),
            body('difficultyLevelId').notEmpty().isInt(),
            body('placeId').notEmpty().isInt(),
            body('tourActivities').notEmpty().isArray(),
            body('tourTypes').notEmpty().isArray(),
        ])
        const tour: Tour = await StoreTourUseCase.run(request.body)
        ApiResponse.payload(response, transform(tour, new TourTransformer()))
    } catch (e: any) {
        next(e)
    }
}
```

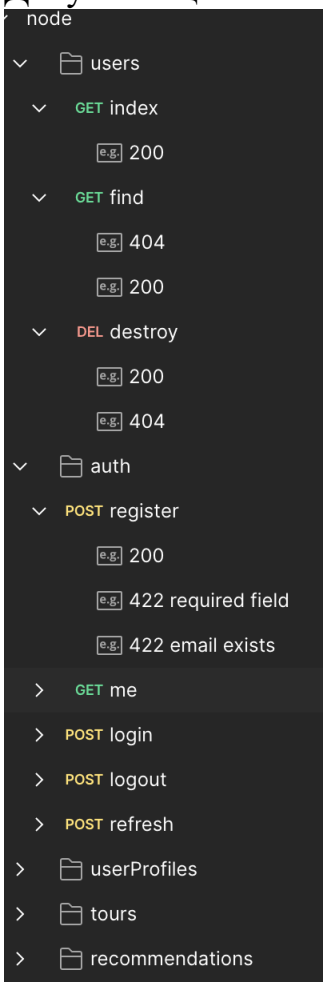
Сервис (useCase) на примере с туром.

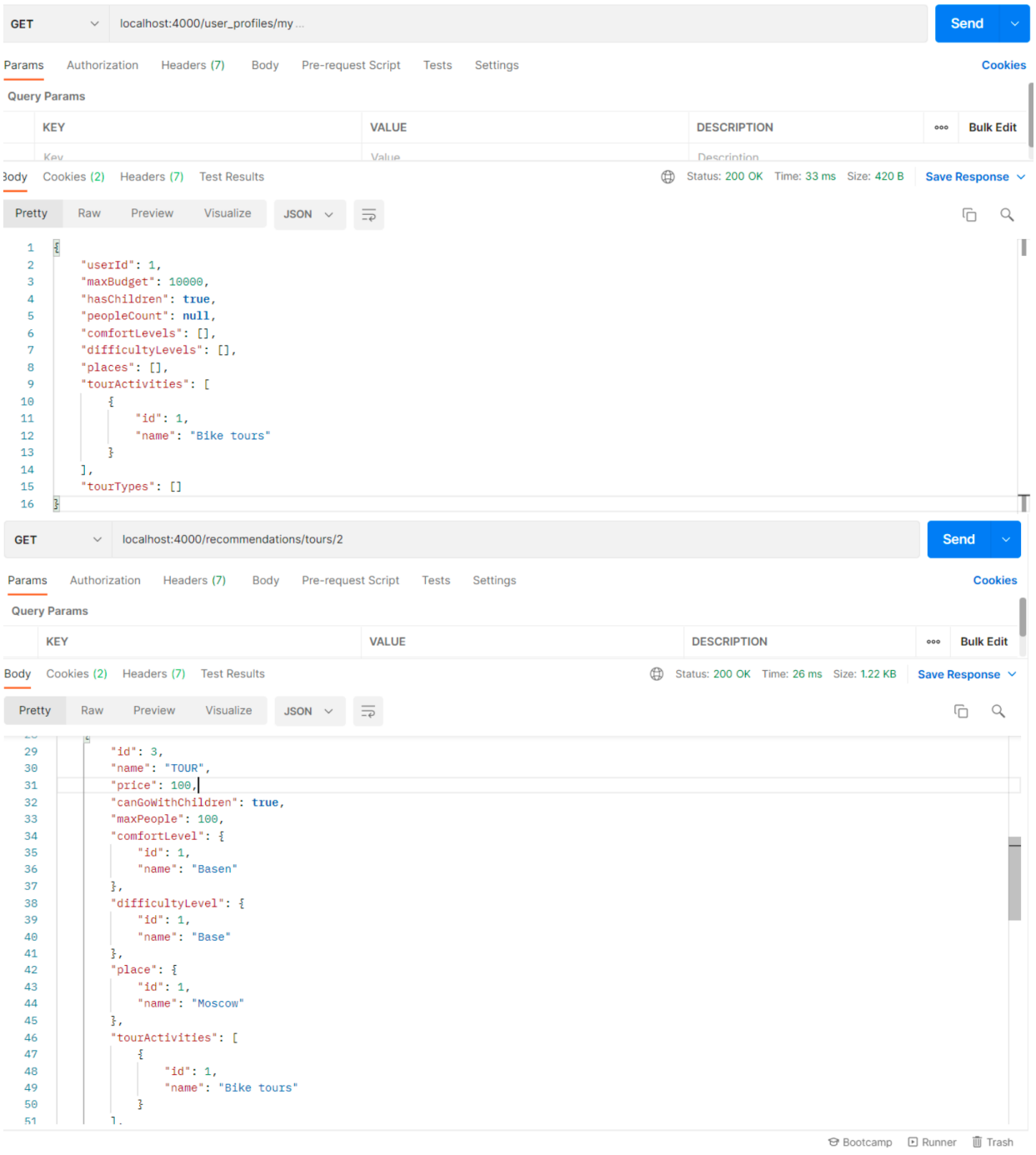
```
class FindTourUseCase {
    static async run(id: number): Promise<Tour> {
        // @ts-ignore
        const tour = await Tour.findByPk(id, {
            include: [
                { model: ComfortLevel },
                { model: DifficultyLevel },
                { model: Place },
                { model: TourActivity },
                { model: TourType },
            ]
        });
        if (!tour) {
            throw new NotFoundError('Tour does not exists')
        }
        return tour
    }
}

export default FindTourUseCase
```

Маршруты приложения.

```
import usersRouter from './users'
import authRouter from './auth'
import userProfilesRouter from './userProfiles'
import toursRouter from './tours'
import recommendationsRouter from './recommendations'

export const routers = [
    { prefix: '/users', router: usersRouter },
    { prefix: '/auth', router: authRouter },
    { prefix: '/user_profiles', router: userProfilesRouter },
    { prefix: '/tours', router: toursRouter },
    { prefix: '/recommendations', router: recommendationsRouter },
]
```

Документация в Postman.





Вывод:

В ходе данной работы был написан свой RESTful API средствами express + typescript (используя ранее написанный boilerplate).