САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа

Выполнил:

Тюмин Никита

Группа К33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2024 г.

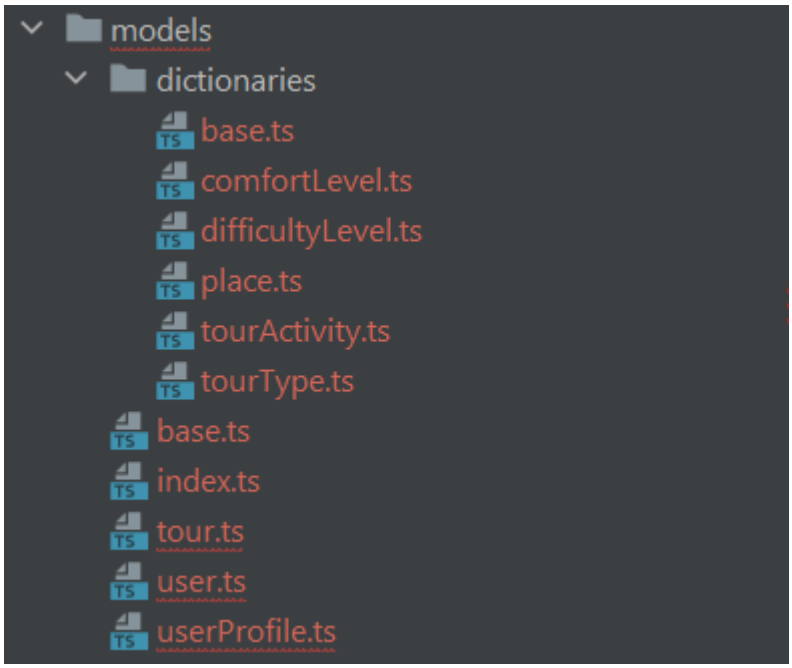
Задача

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:

```
import ...

@Table( options: {
    tableName: 'tours'
})
class Tour extends BaseModel {
    @Column
    name: string
    @Column
    price: number
    @Column
    canGoWithChildren: boolean
    @Column
    maxPeople: number

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

    @BelongsToMany( associatedClassGetter: () => TourActivity, options: {
        through: 'tour_has_tour_activity',
        foreignKey: 'tourId',
        otherKey: 'tourActivityId',
    })
```

Создаем контроллеры:

На примере Тура:

```
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, rules: [
            body( fields: 'name').notEmpty().isString(),
            body( fields: 'price').notEmpty().isInt(),
            body( fields: 'canGoWithChildren').notEmpty().isBoolean(),
            body( fields: 'maxPeople').notEmpty().isInt(),
            body( fields: 'comfortLevelId').notEmpty().isInt(),
            body( fields: 'difficultyLevelId').notEmpty().isInt(),
            body( fields: 'placeId').notEmpty().isInt(),
            body( fields: 'tourActivities').notEmpty().isArray(),
            body( fields: 'tourTypes').notEmpty().isArray(),
        ])
        const tour: Tour = await StoreTourUseCase.run(request.body)
        ApiResponse.payload(response, transform(tour, new TourTransformer()))
    } catch (e: any) {
        next(e)
    }
}
```

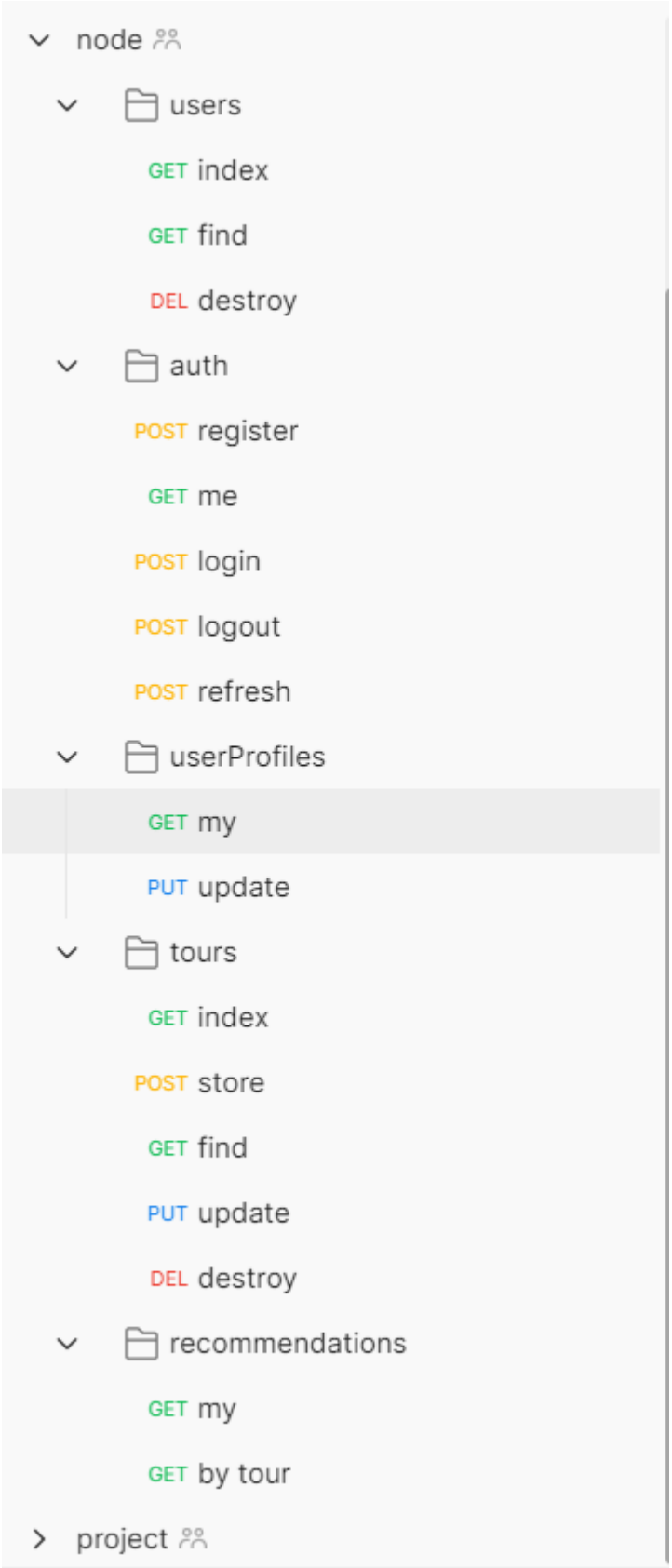
StoreTourUseCase:

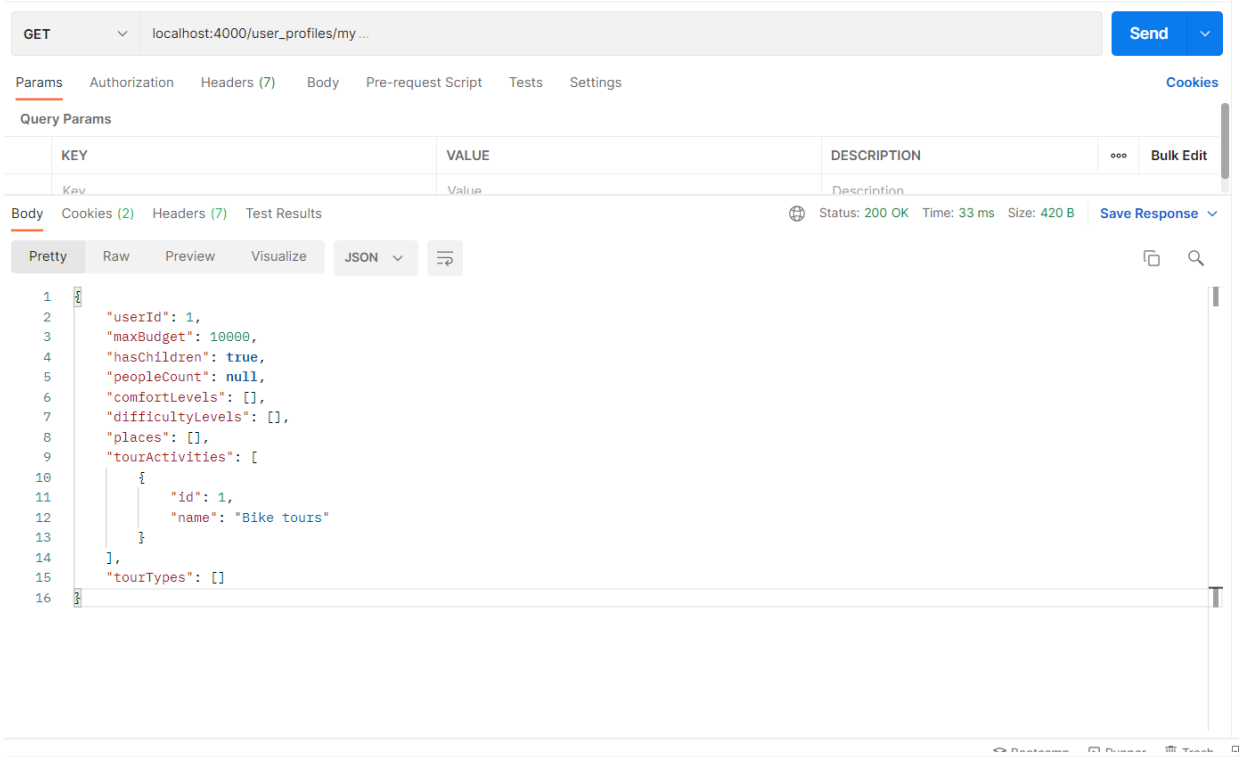
```
class StoreTourUseCase {
    static async run(data: IStoreTour): Promise<Tour> {
        return await sequelize.transaction( autoCallback: async (transaction : Transaction ) => {
            try {
                const tour: Tour = await Tour.create( values: {
                    name: data.name,
                    price: data.price,
                    canGoWithChildren: data.canGoWithChildren,
                    maxPeople: data.maxPeople,
                    comfortLevelId: data.comfortLevelId,
                    difficultyLevelId: data.difficultyLevelId,
                    placeId: data.placeId,
                }, options: {returning: true})

                await tour.$set( propertyKey: 'tourActivities', data.tourActivities)
                await tour.$set( propertyKey: 'tourTypes', data.tourTypes)

                await tour.reload( options: {
                    include: [
                        'comfortLevel',
                        'difficultyLevel',
                        'place',
                        'tourActivities',
                        'tourTypes',
                    ]
                })

                return tour
            } catch (error) {
```

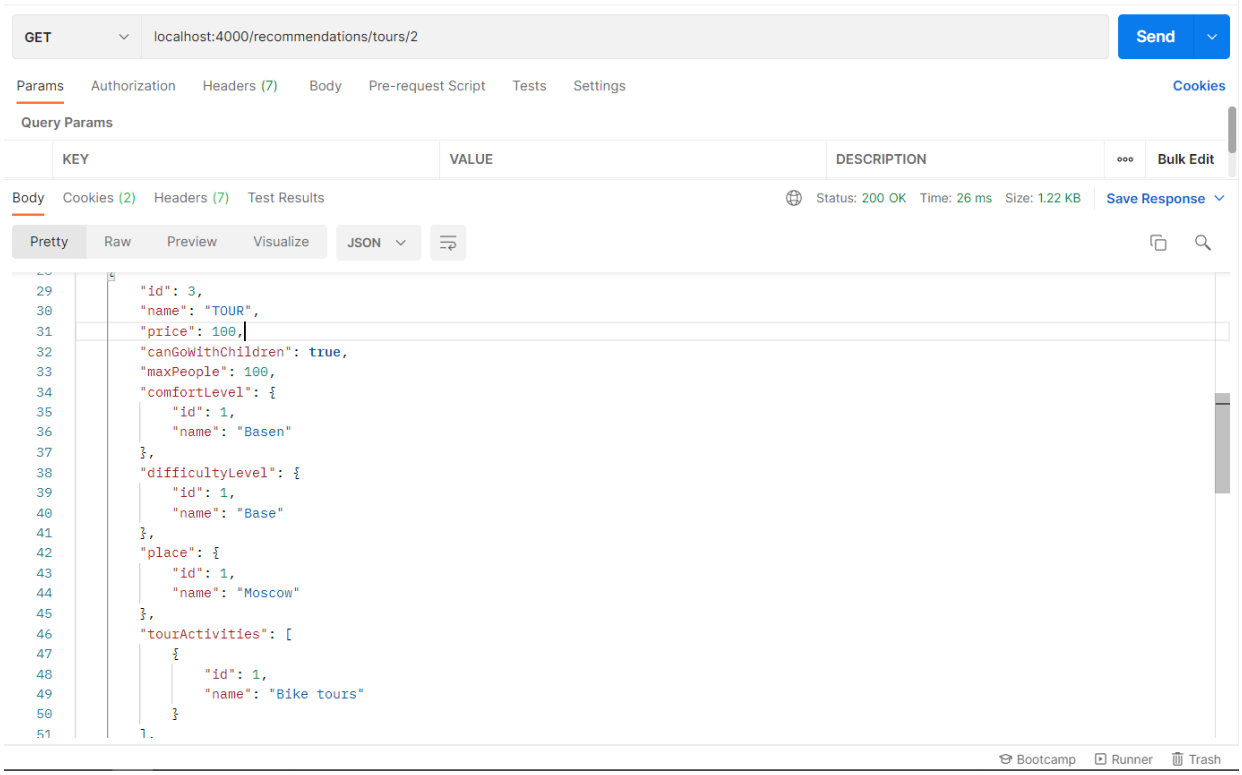
Маршруты приложения:

```
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 постмане:




GET
localhost:4000/user_profiles/my ...
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Body
Cookies (2)
Headers (7)
Test Results
Status: 200 OK
Time: 33 ms
Size: 420 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"userId": 1,
"maxBudget": 10000,
"hasChildren": true,
"peopleCount": null,
"comfortLevels": [],
"difficultyLevels": [],
"places": [],
"tourActivities": [
{
"id": 1,
"name": "Bike tours"
}
],
"tourTypes": []
}


GET
localhost:4000/recommendations/tours/2
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Body
Cookies (2)
Headers (7)
Test Results
Status: 200 OK
Time: 26 ms
Size: 1.22 KB
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": 3,
"name": "TOUR",
"price": 100,
"canGoWithChildren": true,
"maxPeople": 100,
"comfortLevel": {
"id": 1,
"name": "Basen"
},
"difficultyLevel": {
"id": 1,
"name": "Base"
},
"place": {
"id": 1,
"name": "Moscow"
},
"tourActivities": [
{
"id": 1,
"name": "Bike tours"
}
]
Bootcamp
Runner
Trash

Вывод

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