САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №2: Написание API

Выполнил:

Кондратьев Алексей

Группа К33412

Проверил:
Добряков Д. И.

Санкт-Петербург

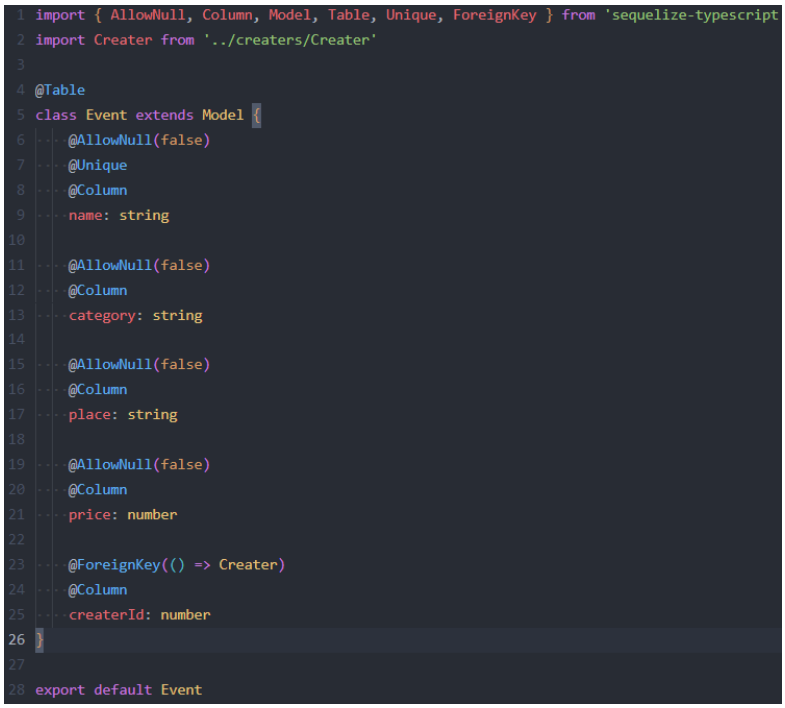
2023 г.

Задача

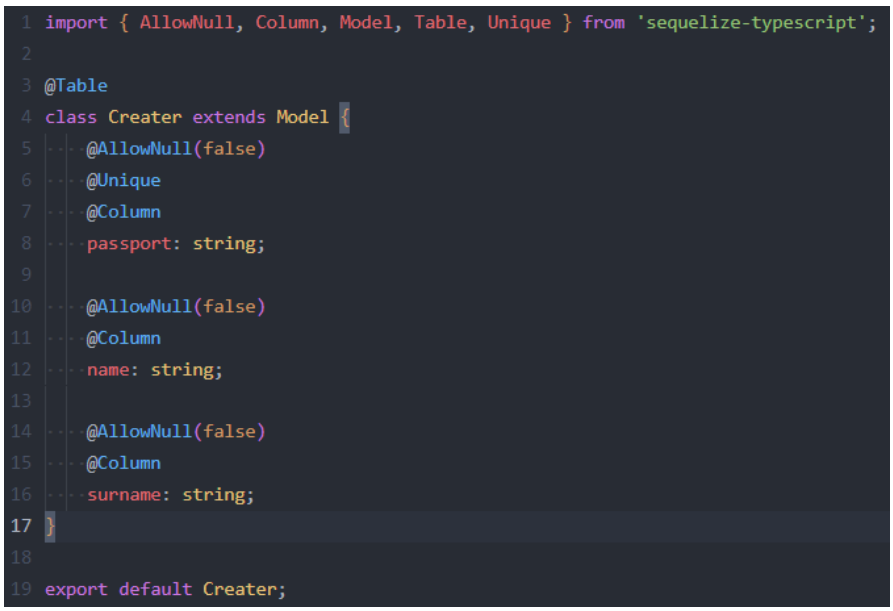
По выбранному варианту необходимо будет реализовать RESTful API средствами express + typescript (используя ранее написанный boilerplate)

Ход работы

Модели для реализации Api:



```
import { AllowNull, Column, Model, Table, Unique, ForeignKey } from 'sequelize-typescript
import Creater from '../creaters/Creater'

@Table
class Event extends Model {
    @AllowNull(false)
    @Unique
    @Column
    name: string

    @AllowNull(false)
    @Column
    category: string

    @AllowNull(false)
    @Column
    place: string

    @AllowNull(false)
    @Column
    price: number

    @ForeignKey(() => Creater)
    @Column
    createrId: number
}

export default Event
```



```
import { AllowNull, Column, Model, Table, Unique } from 'sequelize-typescript';

@Table
class Creater extends Model {
    @AllowNull(false)
    @Unique
    @Column
    passport: string;

    @AllowNull(false)
    @Column
    name: string;

    @AllowNull(false)
    @Column
    surname: string;
}

export default Creater;
```

```ts
import { Table, Column, Model, Unique, AllowNull, ForeignKey } from 'sequelize-typescript'
import User from '../users/User'

@Table
class RefreshToken extends Model {
    @Unique
    @AllowNull(false)
    @Column
    token: string

    @ForeignKey(() => User)
    @Column
    userId: number
}

export default RefreshToken
```

```ts
import { AllowNull, BeforeCreate, BeforeUpdate, Column, Model, Table, Unique } from 'sequelize-typescript'
import hashPassword from '../../utils/hashPassword'

@Table
class User extends Model {
    @AllowNull(false)
    @Column
    firstName: string

    @AllowNull(false)
    @Column
    lastName: string

    @Unique
    @Column
    email: string

    @AllowNull(false)
    @Column
    password: string

    @BeforeCreate
    @BeforeUpdate
    static generatePasswordHash(instance: User) {
        const { password } = instance

        if (instance.changed('password')) {
            instance.password = hashPassword(password)
        }
    }
}

export default User
```

Роуты для наших запросов:

```ts
import express from "express"
import CreaterController from "../../../controllers/creaters/Creater"

const router: express.Router = express.Router()

const controller: CreaterController = new CreaterController()

router.get('/:id', controller.get)
router.post('/', controller.create)
router.put('/:id', controller.update)
router.delete('/:id', controller.delete)
router.post('/filter', controller.getByParams)

export default router
```

```ts
import express from "express"
import EventController from "../../../controllers/events/Event"

const router: express.Router = express.Router()

const controller: EventController = new EventController()

router.get('/:id', controller.get)
router.post('/', controller.create)
router.put('/:id', controller.update)
router.delete('/:id', controller.delete)
router.post('/filter', controller.getByParams)

export default router
```

```ts
import express from "express"
import UserController from "../../../controllers/users/User"
import passport from "../../../middlewares/passport"

const router: express.Router = express.Router()

const controller: UserController = new UserController()

router.post('/', controller.create)
router.get('/profile', passport.authenticate('jwt', { session: false }), controller.me)
router.get('/:id', controller.get)
router.post('/login', controller.auth)
router.post('/refresh', controller.refreshToken)

export default router
```

```ts
import express from "express"
import userRoutes from "./users/User"
import createrRoutes from "./creaters/Creater"
import eventRoutes from "./events/Event"

const router: express.Router = express.Router()

router.use('/users', userRoutes)
router.use('/creater', createrRoutes)
router.use('/event', eventRoutes)

export default router
```

Вывод

В ходе работы мы создали API нашего приложения. Мы создали регистрацию и логин пользователя, взаимодействие с jwt токеном и фильтрацию и взаимодействие с другими таблицами базы данных