САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №2

Выполнил:
Кулёмин С. А.
Группа K33401

Проверил:
Добряков Д. И.

Санкт-Петербург
2022 г.

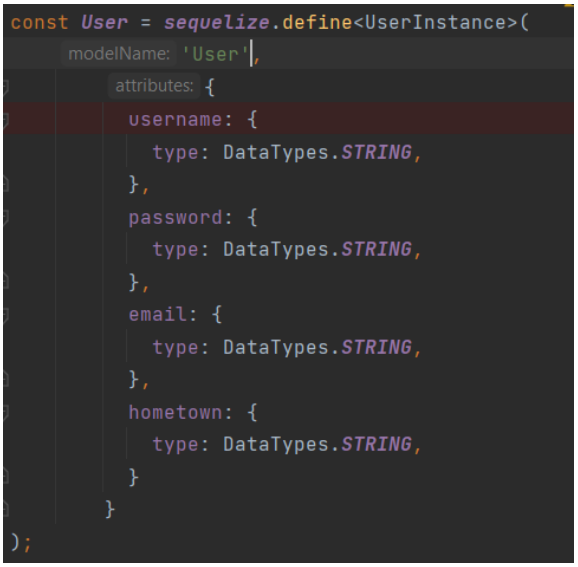
Задание

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 (используя ранее написанный boilerplate).

Выполнение

Модели



```
const User = sequelize.define<UserInstance>(
    modelName: 'User',
      attributes: {
        username: {
          type: DataTypes.STRING,
        },
        password: {
          type: DataTypes.STRING,
        },
        email: {
          type: DataTypes.STRING,
        },
        hometown: {
          type: DataTypes.STRING,
        }
      }
);
```

```
const Restaurant = sequelize.define<RestaurantInstance>(
    modelName: 'Restaurant',
    attributes: {
      name: {
        type: DataTypes.STRING,
      },
      address: {
        type: DataTypes.STRING,
      },
      cuisine: {
        type: DataTypes.STRING,
      },
      average_bill: {
        type: DataTypes.NUMBER,
      },
      rating: {
        type: DataTypes.NUMBER,
      },
      info: {
        type: DataTypes.STRING,
      }
    }
)
```

```
const Booking = sequelize.define<BookingInstance>(
    modelName: 'Booking',
    attributes: {
        username: {
            type: DataTypes.STRING,
        },
        restaurant_name: {
            type: DataTypes.STRING,
        },
        time: {
            type: DataTypes.DATE,
        },
        amount_of_people: {
            type: DataTypes.NUMBER,
        },
        info: {
            type: DataTypes.STRING,
        }
    }
)
```

Routes

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

router.use('/users', userRoutes)
router.use('/bookings', bookingRoutes)
router.use('/restaurant', restaurantRoutes)
```

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

const controller: UserController = new UserController()

//получается регистрация
router.route( prefix: '/add')
    .post(controller.post)

//похоже на вход
router.route( prefix: '/login')
    .post(controller.login)

//хто я?
router.route( prefix: '/me')
    .get(passport.authenticate( strategy: 'jwt',  options: { session: false }), controller.me)

router.route( prefix: '/profiles')
    .get(passport.authenticate( strategy: 'jwt',  options: { session: false }), controller.getAll)

router.route( prefix: '/id/:id')
    .get(passport.authenticate( strategy: 'jwt',  options: { session: false }), controller.getById)

router.route( prefix: '/username/:username')
    .get(passport.authenticate( strategy: 'jwt',  options: { session: false }), controller.getByUsername)
```

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

const controller: RestaurantController = new RestaurantController()

router.route( prefix: '/add')
    .post(passport.authenticate( strategy: 'jwt',  options: { session: false }), controller.post)

router.route( prefix: '/restaurants')
    .get(passport.authenticate( strategy: 'jwt',  options: { session: false }), controller.getAll)

router.route( prefix: '/id/:id')
    .get(passport.authenticate( strategy: 'jwt',  options: { session: false }), controller.getById)

router.route( prefix: '/name/:name')
    .get(passport.authenticate( strategy: 'jwt',  options: { session: false }), controller.getByName)

router.route( prefix: '/ratingHigher/:rating')
    .get(passport.authenticate( strategy: 'jwt',  options: { session: false }), controller.ratingHigher)
```

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

const controller: BookingController = new BookingController()

router.route( prefix: '/add')
    .post(passport.authenticate( strategy: 'jwt',  options: { session: false }), controller.post)

router.route( prefix: '/bookings')
    .get(passport.authenticate( strategy: 'jwt',  options: { session: false }), controller.getAll)

router.route( prefix: '/username/:username')
    .get(passport.authenticate( strategy: 'jwt',  options: { session: false }), controller.getByUsername)

router.route( prefix: '/restaurant/:restaurant_name')
    .get(passport.authenticate( strategy: 'jwt',  options: { session: false }), controller.getByRestaurant)
```

Вывод

В ходе выполнения лабораторной работы был создат RESTful API при помощи express и typescript. Созданы модели Ресторан и Бронь. Для них прописаны контроллеры, сервисы и рауты.