Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшегообразования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

Лабораторной работе № 1

Автор: Коровин А.И

Факультет: ИКТ

Группа: К33402

Преподаватель: Добряков Д.И.

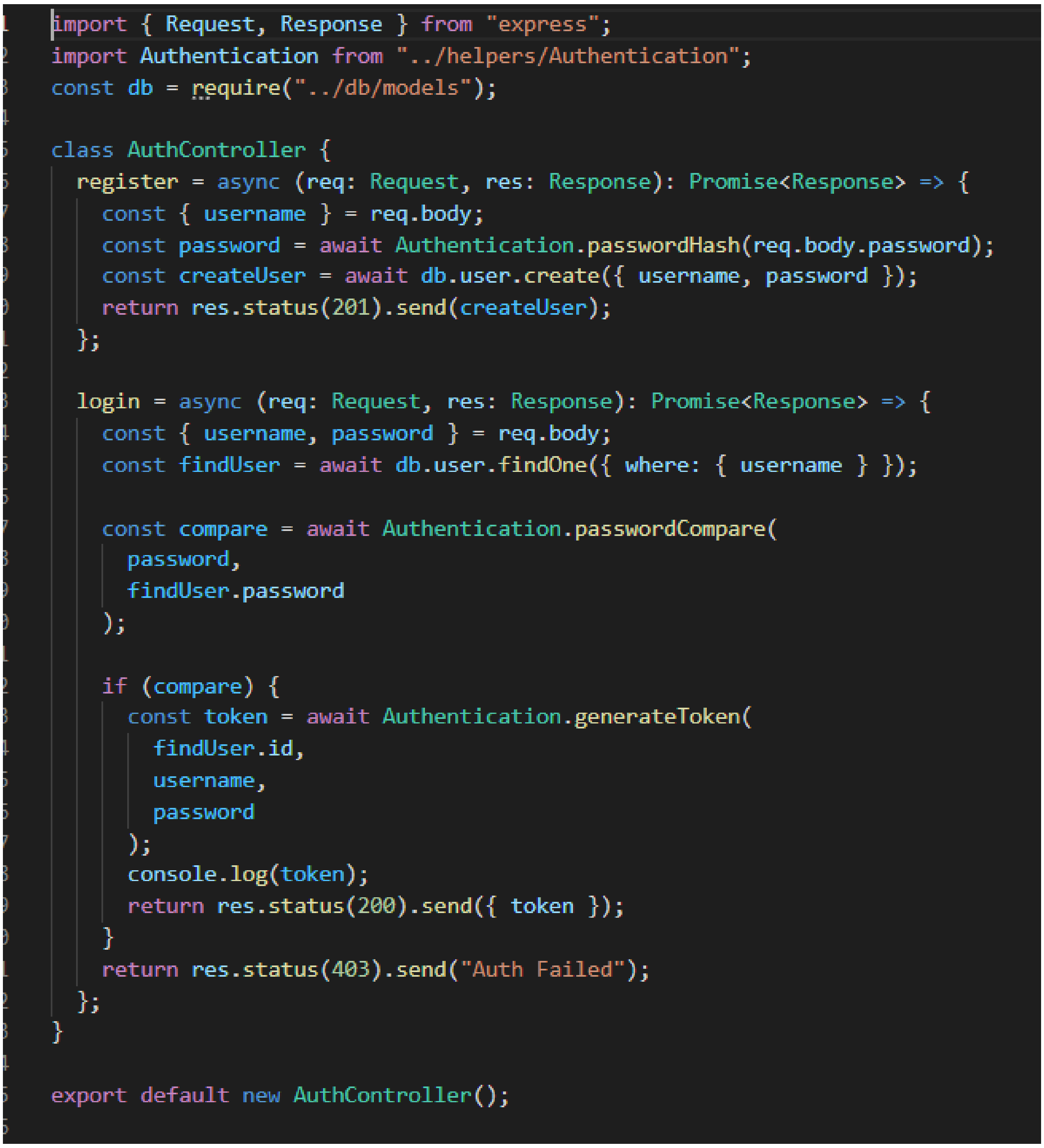
Дата: 17.04.22

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2022

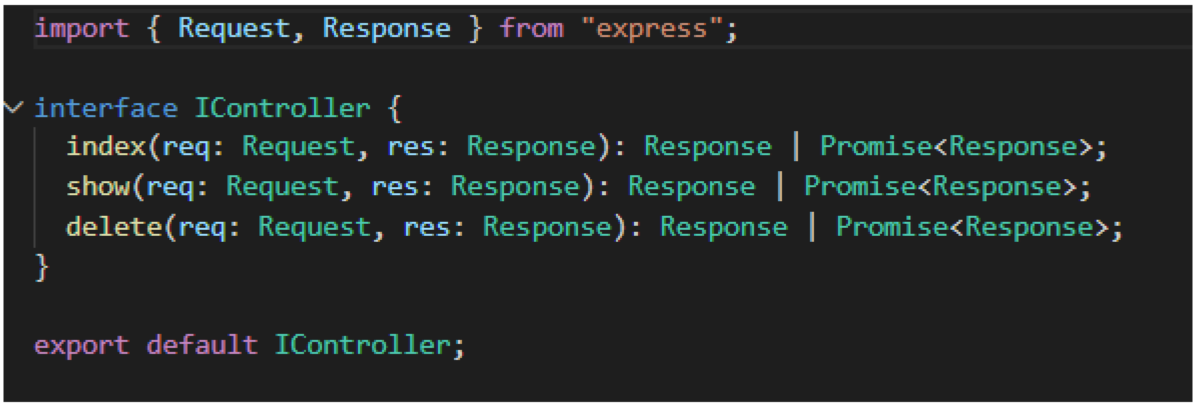
Цель: создать boilerplate на Express с разделением на:

- Модели
- Контроллеры
- Роуты
- Сервисы для работы с моделями



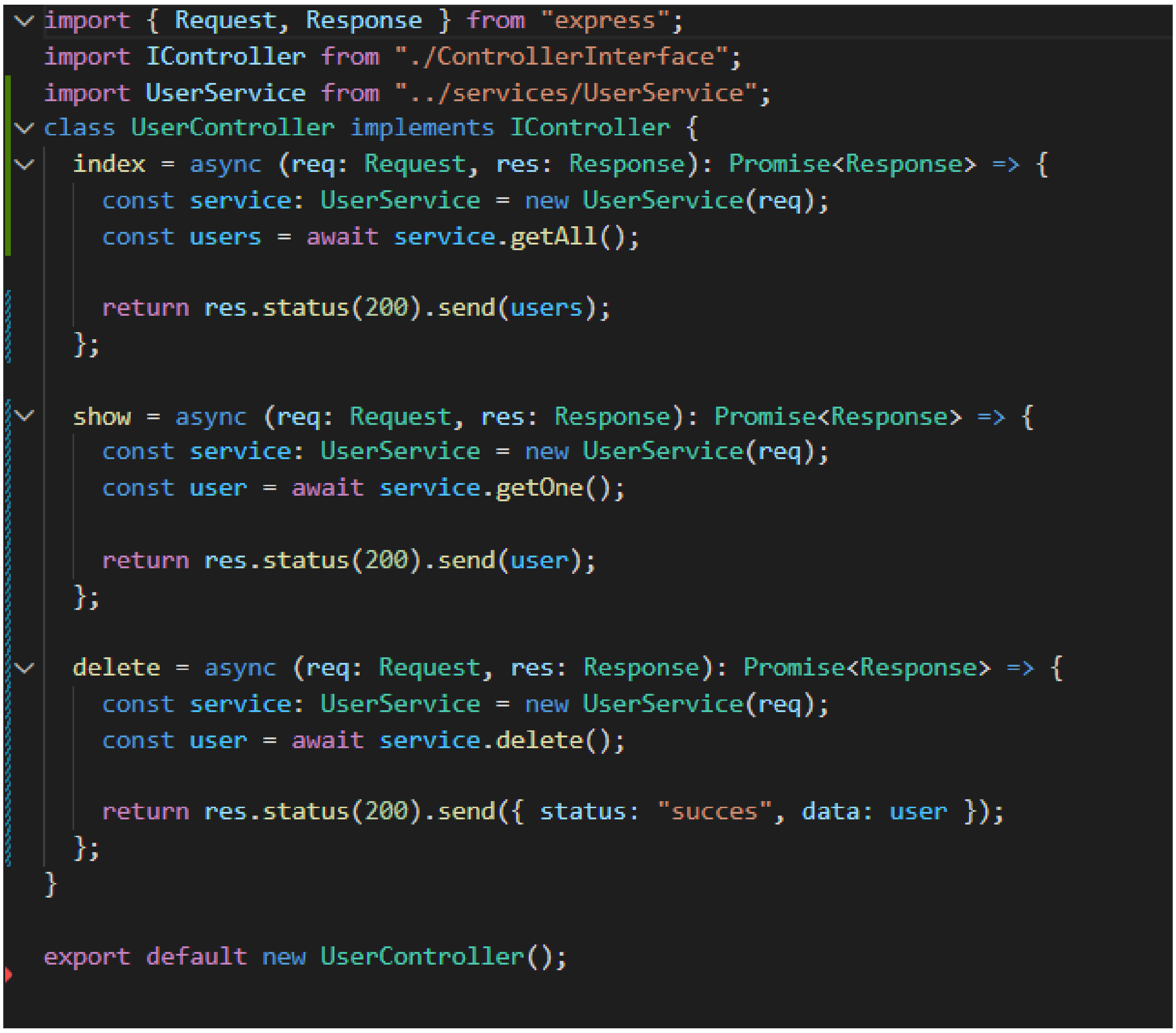
```
import { Request, Response } from "express";
import Authentication from "../helpers/Authentication";
const db = require("../db/models");

class AuthController {
  register = async (req: Request, res: Response): Promise<Response> => {
    const { username } = req.body;
    const password = await Authentication.passwordHash(req.body.password);
    const createUser = await db.user.create({ username, password });
    return res.status(201).send(createUser);
  };

  login = async (req: Request, res: Response): Promise<Response> => {
    const { username, password } = req.body;
    const findUser = await db.user.findOne({ where: { username } });

    const compare = await Authentication.passwordCompare(
      password,
      findUser.password
    );

    if (compare) {
      const token = await Authentication.generateToken(
        findUser.id,
        username,
        password
      );
      console.log(token);
      return res.status(200).send({ token });
    }
    return res.status(403).send("Auth Failed");
  };
}

export default new AuthController();
```

Контроллер авторизации



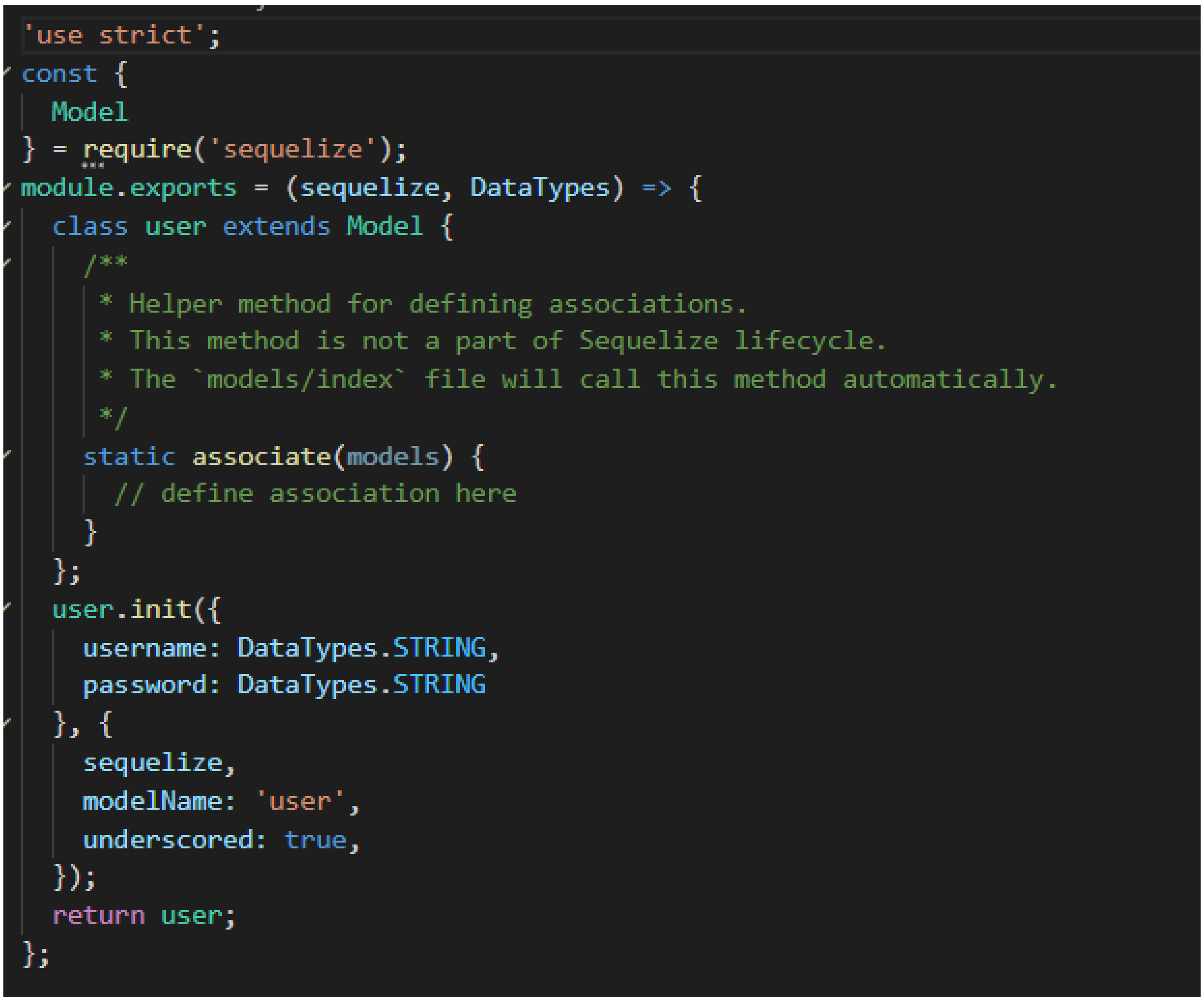
```typescript
import { Request, Response } from "express";

interface IController {
  index(req: Request, res: Response): Response | Promise<Response>;
  show(req: Request, res: Response): Response | Promise<Response>;
  delete(req: Request, res: Response): Response | Promise<Response>;
}

export default IController;
```

TypeScript интерфейс контроллеров



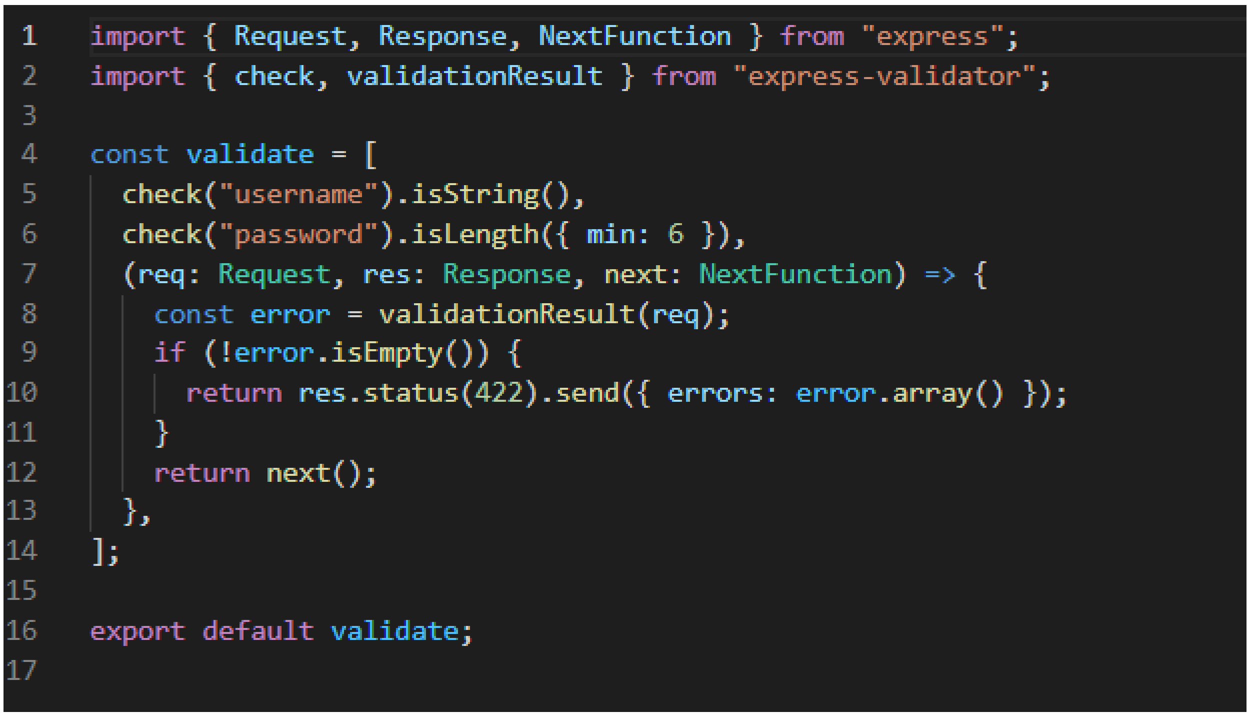
```typescript
import { Request, Response } from "express";
import IController from "./ControllerInterface";
import UserService from "../services/UserService";
class UserController implements IController {
  index = async (req: Request, res: Response): Promise<Response> => {
    const service: UserService = new UserService(req);
    const users = await service.getAll();

    return res.status(200).send(users);
  };

  show = async (req: Request, res: Response): Promise<Response> => {
    const service: UserService = new UserService(req);
    const user = await service.getOne();

    return res.status(200).send(user);
  };

  delete = async (req: Request, res: Response): Promise<Response> => {
    const service: UserService = new UserService(req);
    const user = await service.delete();

    return res.status(200).send({ status: "succes", data: user });
  };
}

export default new UserController();
```

Контроллер user



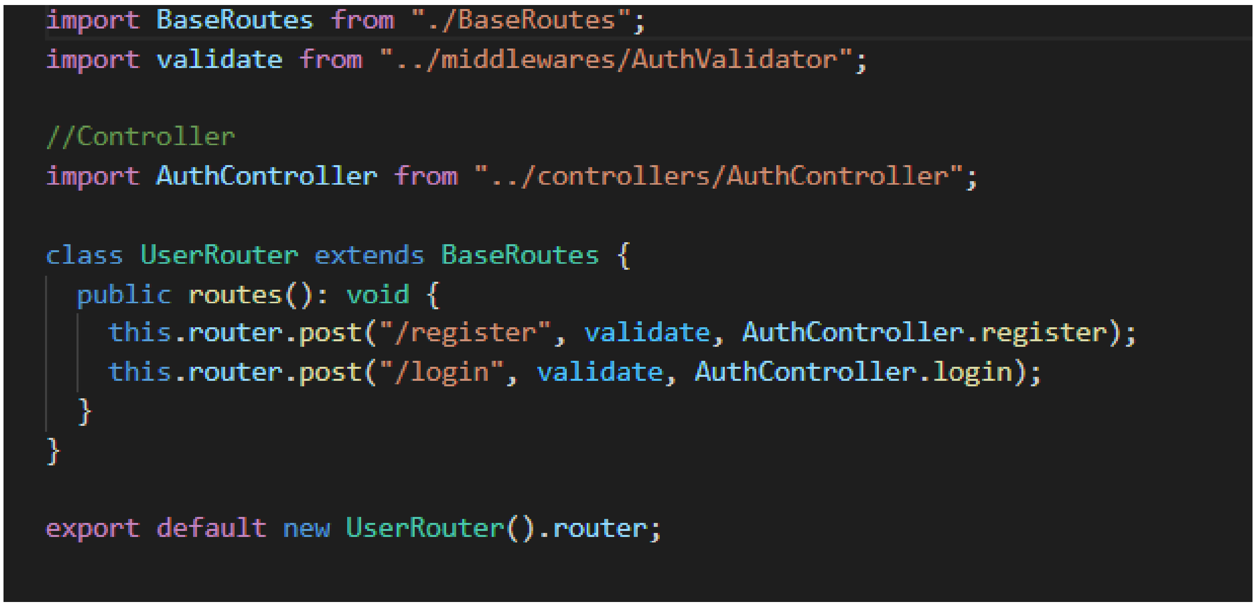
```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class user extends Model {
    /**
     * Helper method for defining associations.
     * This method is not a part of Sequelize lifecycle.
     * The `models/index` file will call this method automatically.
     */
    static associate(models) {
      // define association here
    }
  };
  user.init({
    username: DataTypes.STRING,
    password: DataTypes.STRING
  }, {
    sequelize,
    modelName: 'user',
    underscored: true,
  });
  return user;
};
```

Модель user



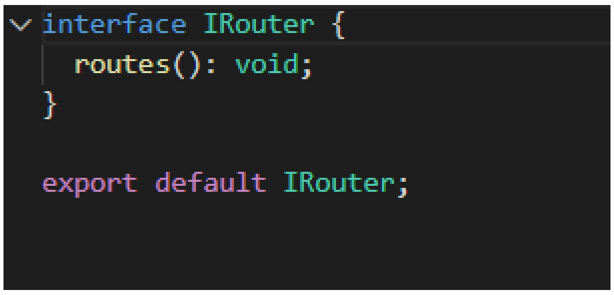
```
import { Request, Response, NextFunction } from "express";
import { check, validationResult } from "express-validator";

const validate = [
  check("username").isString(),
  check("password").isLength({ min: 6 }),
  (req: Request, res: Response, next: NextFunction) => {
    const error = validationResult(req);
    if (!error.isEmpty()) {
      return res.status(422).send({ errors: error.array() });
    }
    return next();
  },
];

export default validate;
```

Валидация регистрации



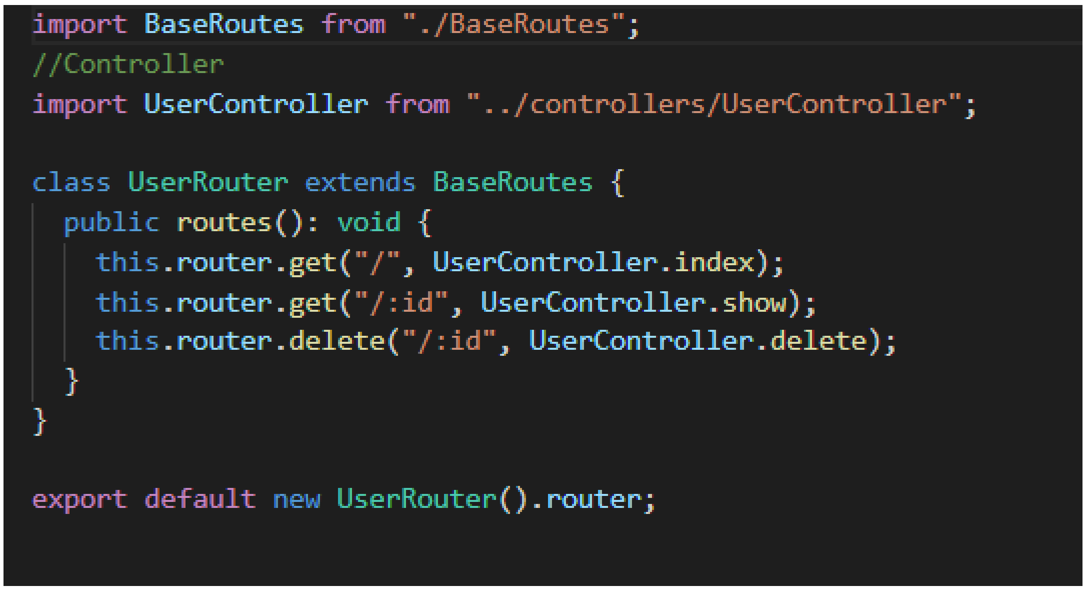
```
import BaseRoutes from "./BaseRoutes";
import validate from "../middlewares/AuthValidator";

//Controller
import AuthController from "../controllers/AuthController";

class UserRouter extends BaseRoutes {
  public routes(): void {
    this.router.post("/register", validate, AuthController.register);
    this.router.post("/login", validate, AuthController.login);
  }
}

export default new UserRouter().router;
```

Роуты авторизации



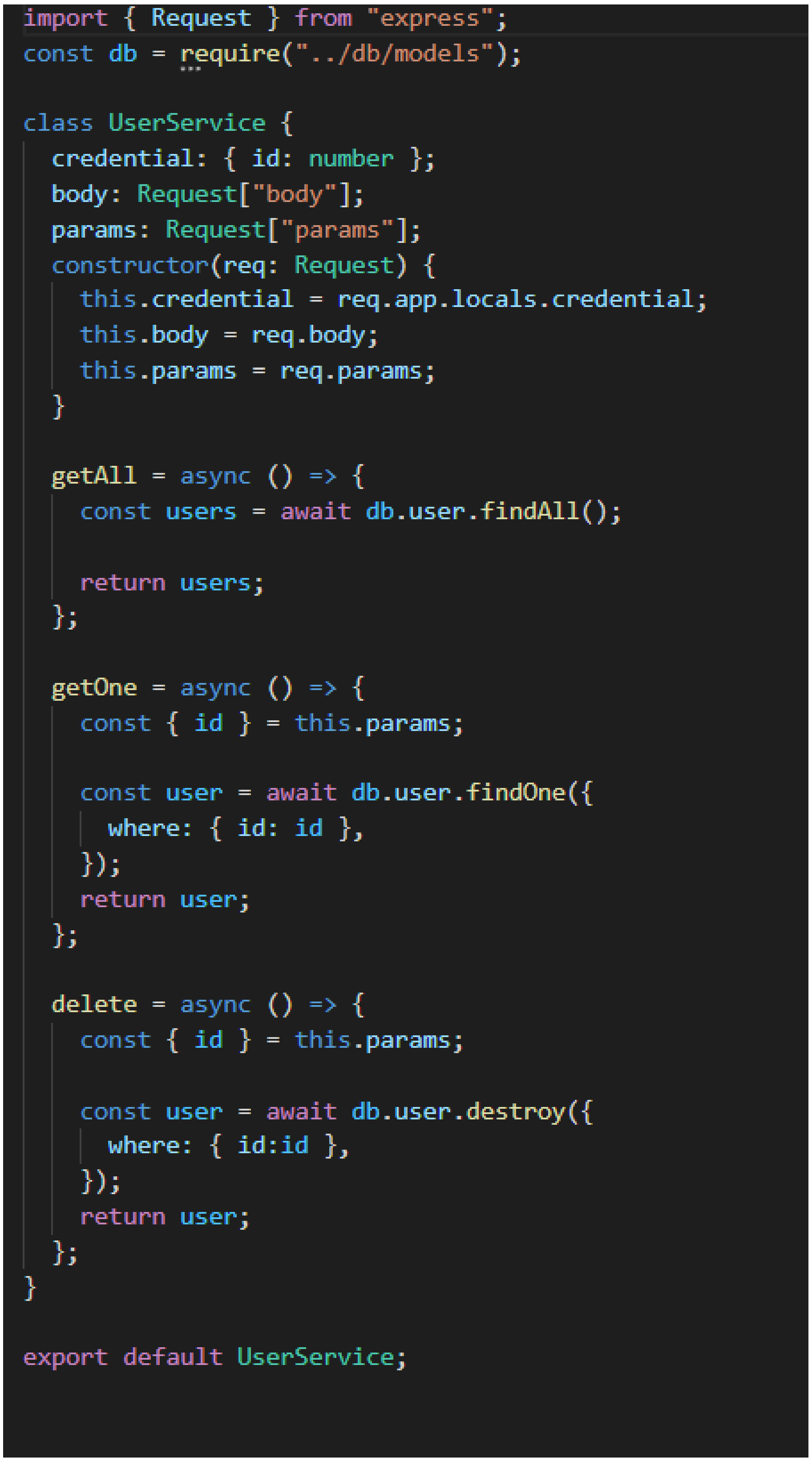
```
interface IRouter {
  routes(): void;
}

export default IRouter;
```

Интерфейс роутов



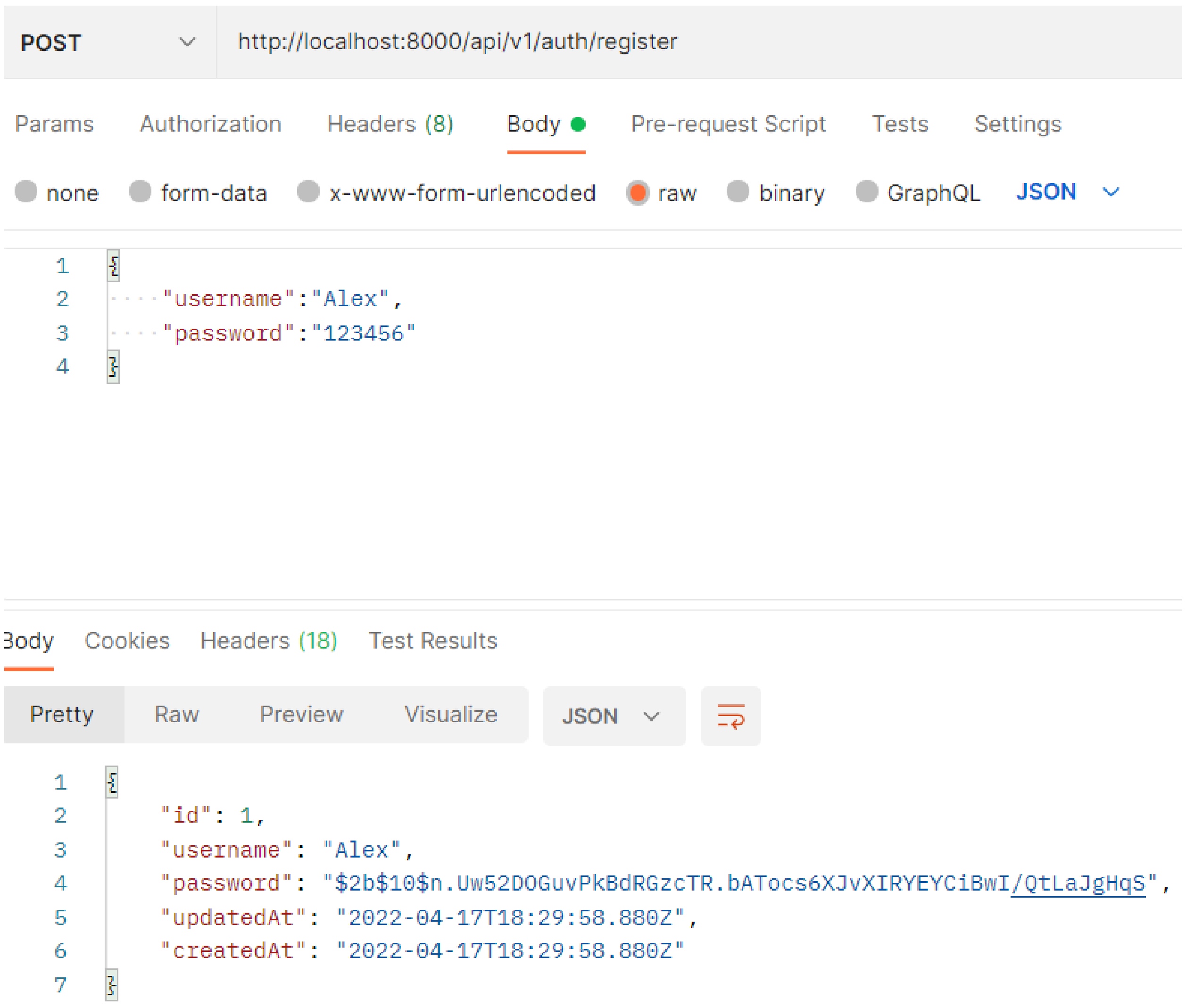
```
import BaseRoutes from "./BaseRoutes";
//Controller
import UserController from "../controllers/UserController";

class UserRouter extends BaseRoutes {
  public routes(): void {
    this.router.get("/", UserController.index);
    this.router.get("/:id", UserController.show);
    this.router.delete("/:id", UserController.delete);
  }
}

export default new UserRouter().router;
```

Роуты user



```
import { Request } from "express";
const db = require("../db/models");

class UserService {
  credential: { id: number };
  body: Request["body"];
  params: Request["params"];
  constructor(req: Request) {
    this.credential = req.app.locals.credential;
    this.body = req.body;
    this.params = req.params;
  }

  getAll = async () => {
    const users = await db.user.findAll();

    return users;
  };

  getOne = async () => {
    const { id } = this.params;

    const user = await db.user.findOne({
      where: { id: id },
    });
    return user;
  };

  delete = async () => {
    const { id } = this.params;

    const user = await db.user.destroy({
      where: { id:id },
    });
    return user;
  };
}

export default UserService;
```

Сервис user с возможностью получения по id, всех и удаление по id

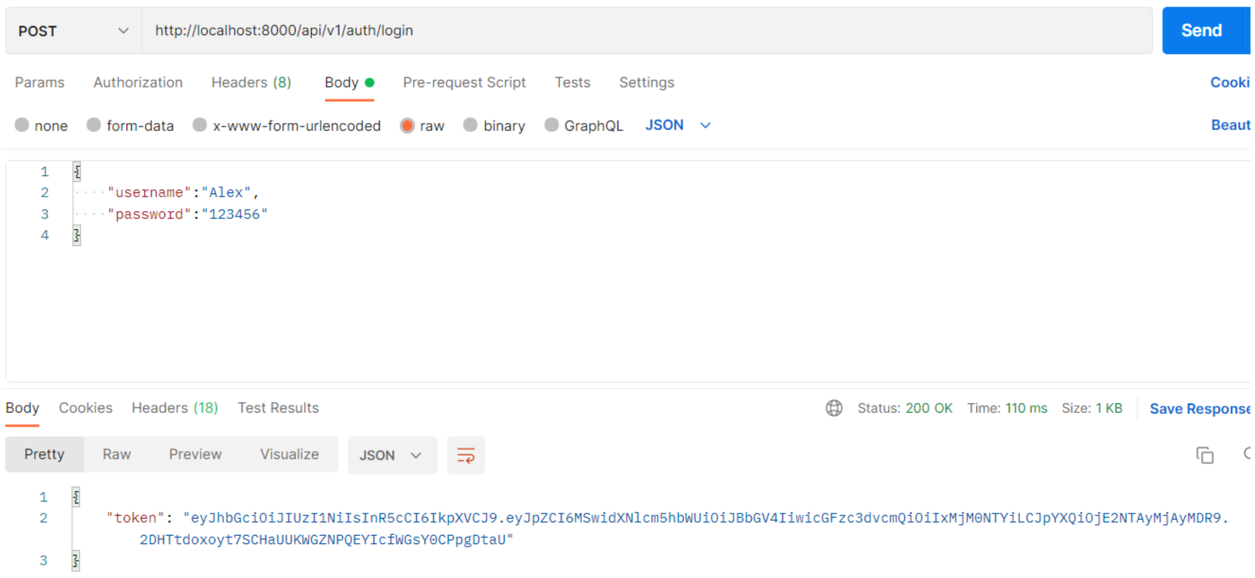


```
http://localhost:8000/api/v1/auth/register
```

```
{
    "username":"Alex",
    "password":"123456"
}
```

```
{
    "id": 1,
    "username": "Alex",
    "password": "$2b$10$n.Uw52DOGuvPkBdRGzcTR.bATocs6XJvXIRYEYCiBwI/QtLaJgHqS",
    "updatedAt": "2022-04-17T18:29:58.880Z",
    "createdAt": "2022-04-17T18:29:58.880Z"
}
```

Проверка регистрации через postman



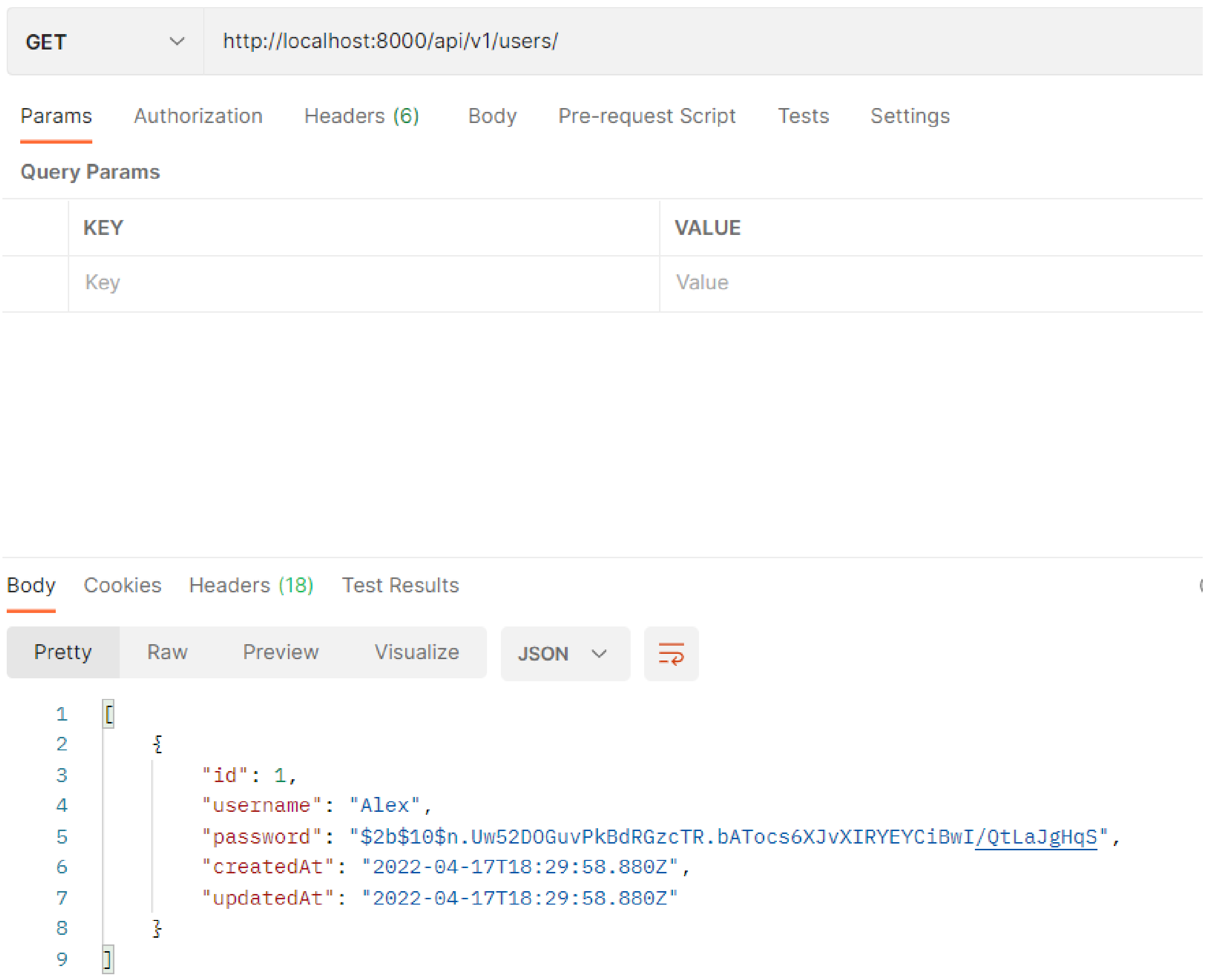
```
http://localhost:8000/api/v1/auth/login
```

```
{
    "username":"Alex",
    "password":"123456"
}
```

Status: 200 OK Time: 110 ms Size: 1 KB

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwidXNlcm5hbWUiOiJBbGV4IiwicGFzc3dvcmQiOiIxMjM0NTYiLCJpYXQiOjE2NTAyMjAyMDR9.
        2DHTtdoxoyt7SCHaUUKWGZNPQEYIcfWGsY0CPpgDtaU"
}
```

Login с получением токена



```
http://localhost:8000/api/v1/users/
```

```
[
    {
        "id": 1,
        "username": "Alex",
        "password": "$2b$10$n.Uw52DOGuvPkBdRGzcTR.bATocs6XJvXIRYEYCiBwI/QtLaJgHqS",
        "createdAt": "2022-04-17T18:29:58.880Z",
        "updatedAt": "2022-04-17T18:29:58.880Z"
    }
]
```

Получение всех юзеров

Вывод: Написан свой boilerplate на express, который включает в себя такие элементы, как модели, контроллеры, роуты и сервисы для работы с моделями.