

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Backend-Разработка

Отчет

Лабораторная работа 2

Выполнил:
Бункута Натан

Группа:
К33412

Проверил:
Добрияков Д.И

Санкт-Петербург

2023 г.

Задание:

В рамках данной лабораторной работы Вам предложено выбрать один из нескольких вариантов. Выбранный вариант остается единым на весь курс и будет использоваться в последующих лабораторных работах.

По выбранному варианту необходимо будет реализовать RESTful API средствами express + typescript. Вариант: Сайт криптобиржи

- Вход
- Регистрация
- Портфель пользователя с указанием различных криптовалют и их количеством
- Графики роста криптовалют
- Поиск по криптовалютам с возможностью фильтрации по дате добавления на биржу

Выполнение работы

Controllers/users/user.ts

```
src > controllers > users > TS User.ts > UserController > get
1  import User from '../../../models/users/User'
2  import UserService from '../../../services/users/User'
3  import APIError from '../../../errors/APIError'
4  import jwt from 'jsonwebtoken'
5  import {jwtOptions} from '../../../middlewares/passport'
6  import RefreshTokenService from '../../../services/auth/RefreshToken'
7
8  class UserController {
9    private userService: UserService
10
11    constructor() {
12      this.userService = new UserService()
13    }
14
15    // get = async (request: any, response: any) => {
16    //   try {
17    //     const user: User | APIError = await this.userService.getById(
18    //       Number(request.params.id)
19    //     )
20    //     response.send(user)
21    //   } catch (error: any) {
22    //     response.status(404).send({'detail': error.message})
23    //   }
24    // }
25
26    post = async (request: any, response: any) => {
27      const {body} = request
28      try {
29        const user: User | APIError = await this.userService.create(body)
30        response.status(201).send(user)
31      } catch (error: any) {
32        response.status(400).send({'detail': error.message})
33      }
34    }
35
36    // Get only allowed for myself
37    get = async (request: any, response: any) => {
38      const {user} = request
39      if (user) {
```

Models/user.ts

```
18  ✓ @DefaultScope(() => ({
19    |   attributes: ['id', 'firstName', 'lastName', 'email']
20    | })
21  ✓ @Scopes(() => ({
22    |   withPassword: {
23    |     |   attributes: ['id', 'firstName', 'lastName', 'email', 'password', 'salt']
24    |     | }
25    | })
26  @Table
27  ✓ class User extends Model {
28    |   @AllowNull(false)
29    |   @Column
30    |   firstName: string
31    |
32    |   @AllowNull(false)
33    |   @Column
34    |   lastName: string
35    |
36    |   @AllowNull(false)
37    |   @Unique
38    |   @Column
39    |   email: string
40    |
41    |   @HasMany(() => Wallet)
42    |   wallets: [Wallet]
43    |
44    |   @AllowNull(false)
45    |   @Column
46    |   password: string
47    |
48    |   @Column
49    |   salt: string
50  }
```

Routes/api/index.ts

```
src > routes > api > TS index.ts > ...
1  import express from 'express'
2  import userRoutes from './users/User'
3  import authRoutes from './auth/Auth'
4  import walletRoutes from './wallet/Wallet'
5  import coinRoutes from './wallet/Coin'
6
7  const router: express.Router = express.Router()
8
9  router.use('/users', userRoutes)
10 router.use('/auth', authRoutes)
11 router.use('/wallets', walletRoutes)
12 router.use('/coins', coinRoutes)
13
14 export default router
15
```

Services/User.ts

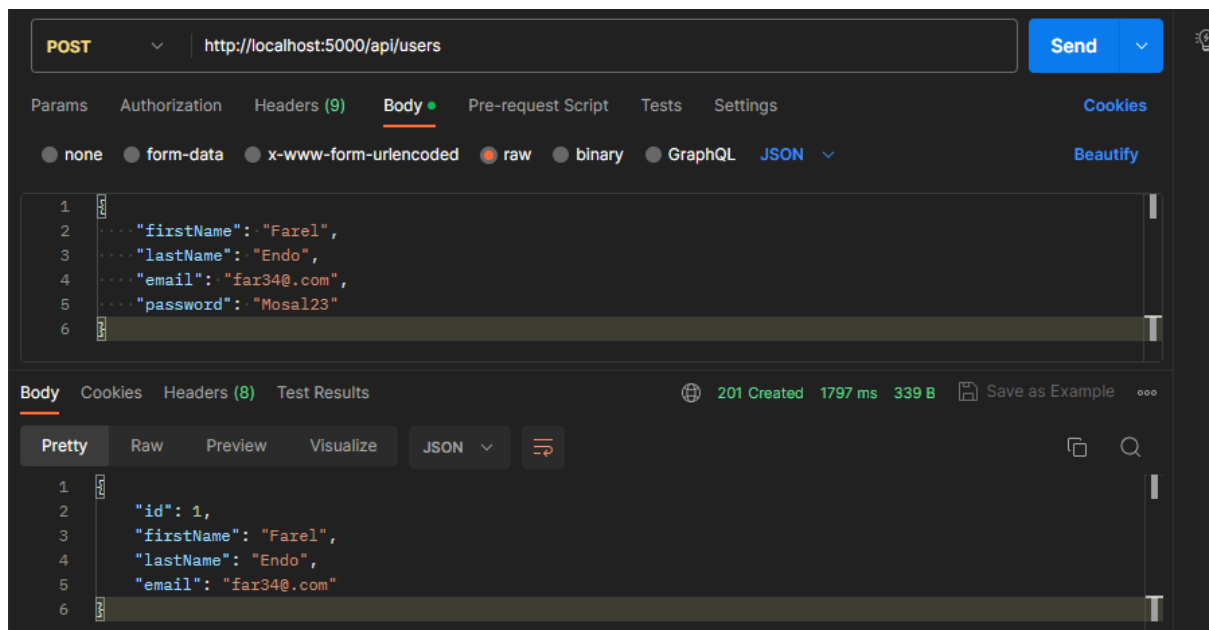
```
src > services > users > TS User.ts
1  import User from './../models/users/User'
2  import APIError from './errors/APIError'
3  import checkPassword from './../utils/checkPassword'
4
5
6  class UserService {
7      async getById(id: number): Promise<User | APIError> {
8          const user = await User.findByPk(id)
9          if (user) {
10             return user.toJSON()
11          }
12          throw new APIError('User not found')
13      }
14
15      async create(userData: any): Promise<User | APIError> {
16          try {
17              const user = await User.create(userData)
18              await user.reload()
19              return user.toJSON()
20          } catch (e: any) {
21              const errors = e.errors.map((error: any) => error.message)
22              throw new APIError(errors)
23          }
24      }
25
26      async update(id: number, userData: any): Promise<User | APIError> {
27          let user = await User.findByPk(id)
28          if (user) {
29              try {
30                  user = await user.update(userData)
31                  await user.reload()
32                  return user.toJSON()
33              } catch (e: any) {
34                  const errors = e.errors.map((error: any) => error.message)
35                  throw new APIError(errors)
36              }
37          }
38          throw new APIError('User not found')
39      }
40  }
```

Providers/db.ts

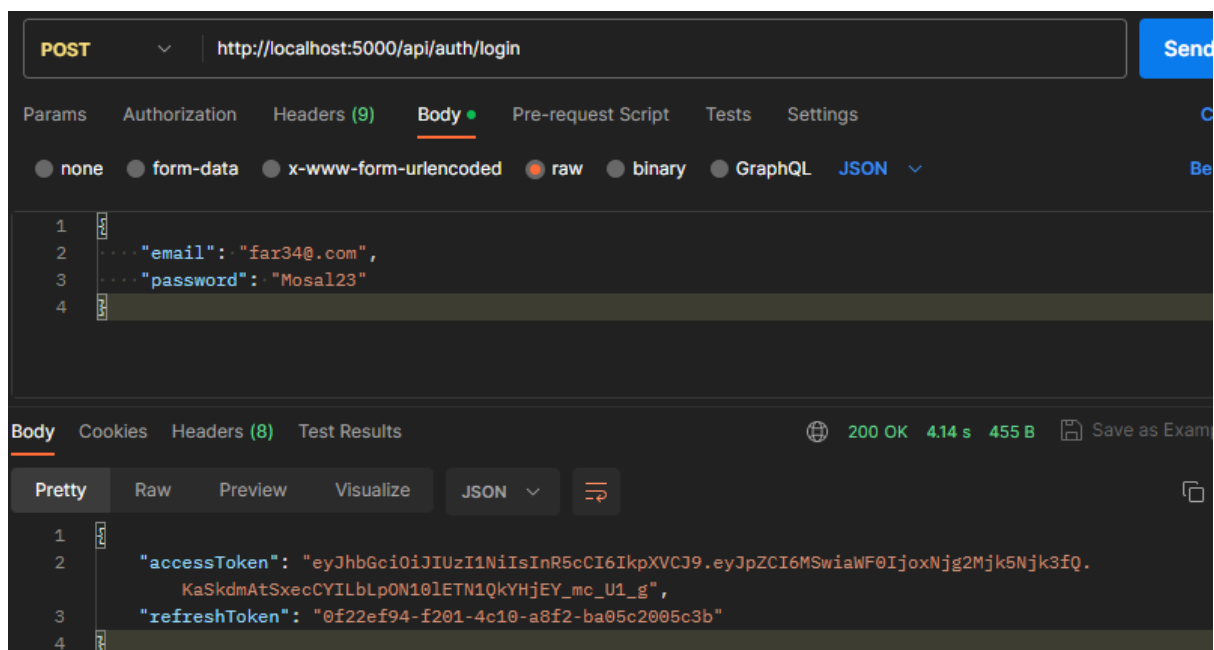
```
src > providers > TS db.ts > ...
1  import {Sequelize} from 'sequelize-typescript'
2  import RefreshToken from '../models/auth/RefreshToken'
3  import User from 'module "c:/Users/natha/OneDrive/Documents/Иформатика/LW2/src/models/wallet/Wallet"'
4  import Wallet from '../models/wallet/Wallet'
5  import Coin from '../models/wallet/Coin'
6  import CoinWallet from '../models/wallet/CoinWallet'
7  import {dbConfig} from '../configs'
8
9
10 const sequelize = new Sequelize({
11   ...dbConfig,
12   logging: console.log
13 })
14
15 const models = [User, RefreshToken, Wallet, Coin, CoinWallet]
16
17 sequelize.addModels(models)
18 console.log('Models added')
19
20 sequelize
21   .sync()
22   .then(() => {
23     console.log('Models synchronized')
24   })
25   .catch((e) => console.log(e))
26
27 async function testConnection() {
28   try {
29     await sequelize.authenticate()
30     console.log('Connection established successfully.')
31   } catch (error) {
32     console.error('Connection to the database failed:', error)
33   }
34 }
35
36 testConnection()
37
38 export default sequelize
39
```

Postman

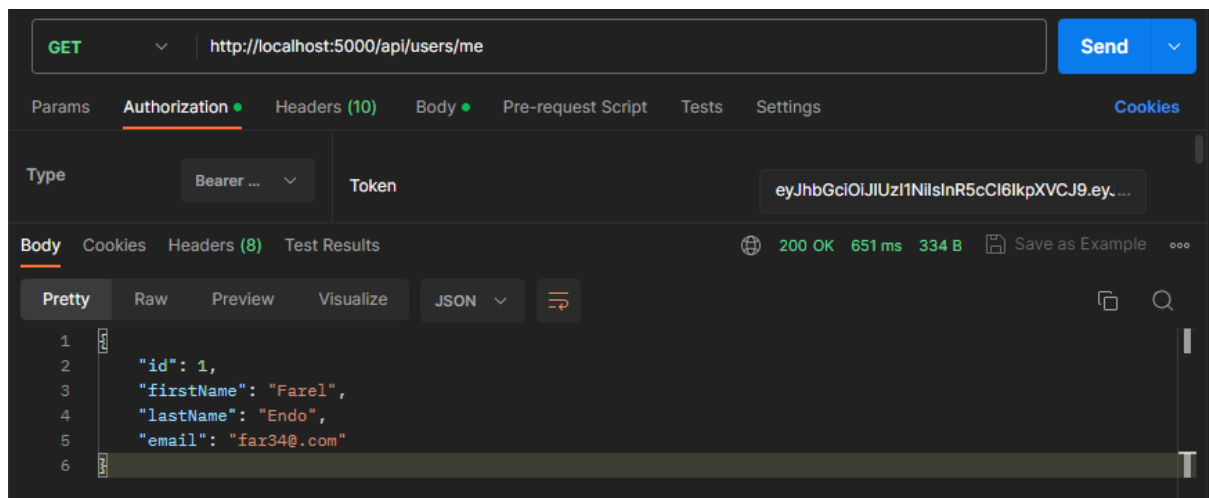
Add a new user



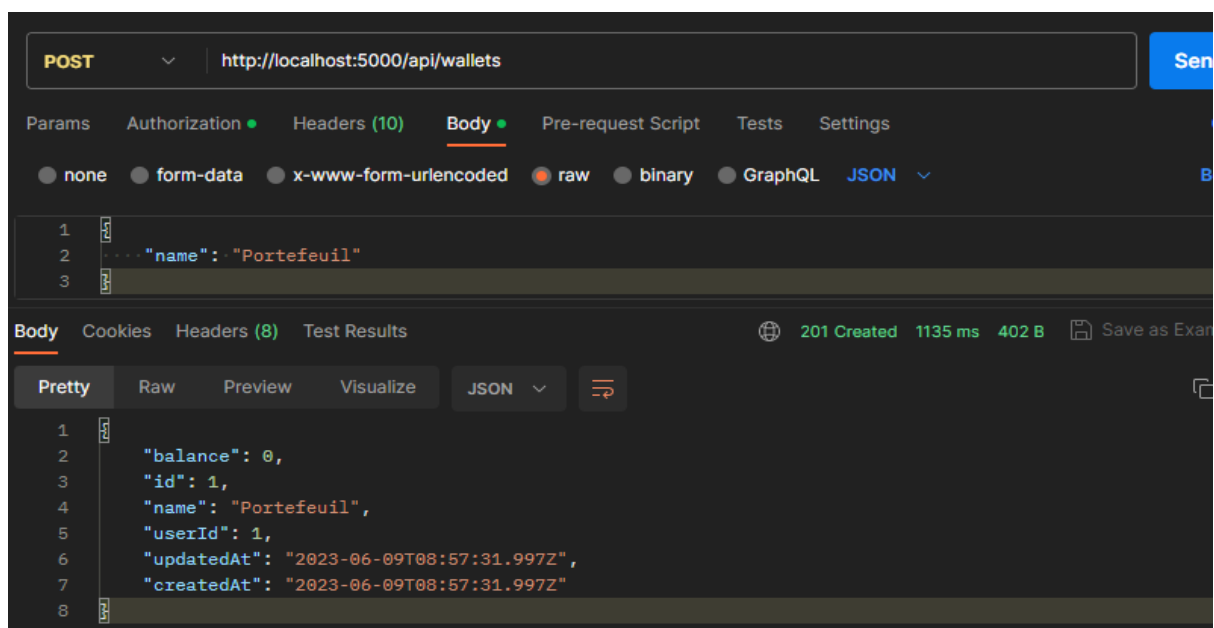
Login



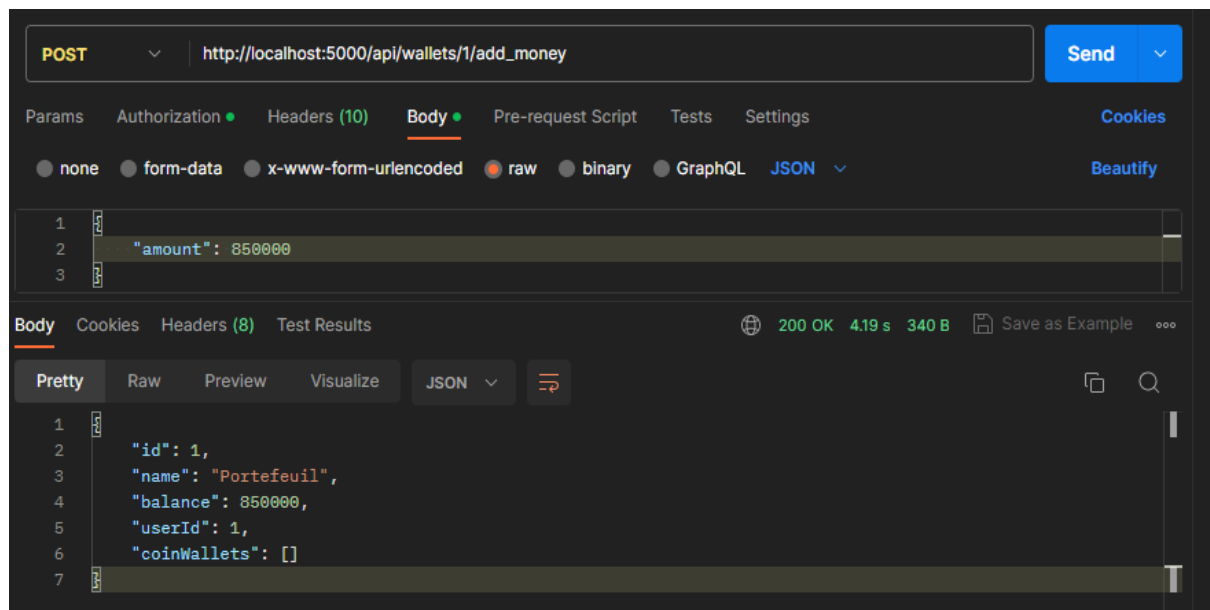
Get user profile (using the token)



Create a Wallet



Add money



Вывод:

При выполнении данной лабораторной работы, я научил реализовать RESTful API с использованием средств `express + typescript` и потестировал в Postman.