САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №2

Выполнила:
Барышева З. А.
Группа:
K33412

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Задача:

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

Вариант 3: Сайт криптобиржи

1. Вход
2. Регистрация
3. Портфель пользователя с указанием различных криптовалют и их количеством
4. Графики роста криптовалют
5. Поиск по криптовалютам с возможностью фильтрации по дате добавления на биржу

Ход работы

Были реализованы три модели, помимо уже существующих в boilerplate:

1. Currency — название валюты и цена

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

    @Column
    value: number

}
```

2. Briefcase — данные о портфелях пользователей(id пользователя, id валюты и количество)

```
@Table
class Briefcase extends Model {
    @AllowNull(false)
    @Column
    count: number

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

    @ForeignKey(() => Currency)
    @Column
    currencyId: number
}
```

3. History — данные о изменении цены валют

```
@Table
class History extends Model {
    @AllowNull(false)
    @Column
    value: number

    @ForeignKey(() => Currency)
    @Column
    currencyId: number
}
```

Реализованы три сервиса для работы с моделями:

1. Currency
    a. создание валюты,
    b. получение списка всех валют,
    c. получение валюты по id,
    d. получение списка валют в отсортированном порядке (возрастающем и убывающем) по дате.

Пример некоторых функций:

```
async getAll() {
    const currency = await Currency.findAll()

    if (currency) return currency

    throw new CurrencyError('currency not found')
}

async filter(filter: string) {
    try {
        console.log(filter)
        const currency = await Currency.findAll({order : [["createdAt", filter]]})

        return currency
    }
    catch(e: any){

        throw new CurrencyError('smth wrong')
    }
}
```

2. Briefcase
    a. добавление в портфель пользователя валюты,
    b. получение списка всех валют пользователя,
    c. получение валюты пользователя по id записи,
    d. изменение количества валюты пользователя,
    e. удаление валюты из портфеля пользователя.

Пример некоторых функций:

```
    async update(id: number, newCount: object) {
        try {
            const briefcase = await Briefcase.update(newCount, {where: {id: id}})

            return this.getById(id)
        }
        catch (e: any){
            throw new Error(e)
        }
    }

    async getAllByUser(id: number) {
        console.log(id)

        const briefcase = await Briefcase.findAll({where: {userId: id}})

        if (briefcase) return briefcase

        throw new BriefcaseError('briefcase for user not found')
    }
}
```

3. History
    a. добавление записи о состоянии валюта-цена,
    b. получение всех записей о валюте в отсортированном по возрастанию порядке по дате(от ранней к текущей).

```
class HistoryService {
    async create(historyData: object) : Promise<History> {
        try {
            const history = await History.create(historyData)

            return history.toJSON()
        }
        catch (e: any) {
            throw new HistoryError(e)
        }
    }

    async getByCurrency(currencyId: number) {
        const history = await History.findAll({
            where : {currencyId: currencyId},
            order : [[ "createdAt", "ASC"]]})

        if (history) return history

        throw new HistoryError('currency not found')
    }
}
```

Реализованы три контроллера:

1. Currency

Пример функции:

```
updateValue = async (request: any, response: any) => {
    const body = request.body
    const currencyId = request.params.id

    console.log(body)

    try {

        const historyService = new HistoryService()

        const currency: Currency = await this.currencyService.updateValue(
            currencyId, body
        )
        const history : History = await historyService.create({
            "currencyId": currencyId,
            "value": body.value,
            "createdAt": currency.updatedAt
        })


        response.status(200).send(currency)
    }
    catch(e: any){
        response.status(404).send({ "error": e.message })
    }
}
```

2. Briefcase

Пример некоторых функций:

```
class BriefcaseController {
    private briefcaseService: BriefcaseService

    constructor() {
        this.briefcaseService = new BriefcaseService()
    }

    getAll = async (request: any, response: any) => {
        const id = request.params.userId
        console.log(id);

        try {
            const briefcase: Briefcase[] | BriefcaseError = await this.briefcaseService.getAllByUser(
                Number(id)
            )

            response.send(briefcase)
        } catch (error: any) {
            response.status(404).send({ "error": error.message })
        }
    }

    getOne = async (request: any, response: any) => {
        const caseId = request.params.caseId
        console.log(caseId);
```

3. History

```
class HistoryController {
    private historyService: HistoryService

    constructor() {
        this.historyService = new HistoryService()
    }

    get = async (request: any, response: any) => {
        const id = request.params.id
        console.log(id);

        try {
            const history: History[] | HistoryError = await this.historyService.getByCurrency(
                Number(id)
            )

            response.send(history)
        } catch (error: any) {
            response.status(404).send({ "error": error.message })
        }
    }

}
```

Добавлены роуты:

1. Currency

```
const controller: CurrencyController = new CurrencyController()

router.route('/')
    .get(controller.getAll)

router.route('/:id')
    .get(controller.get)

router.route('/:id')
    .put(controller.updateValue)

router.route('/create')
    .post(controller.post)

router.route('/filter/:filter')
    .get(controller.filter)
```

2. Briefcase

```
const controller: BriefcaseController = new BriefcaseController()

router.route('/user/:userId')
    .get(controller.getAll)

router.route('/:caseId')
    .get(controller.getOne)

router.route('/create')
    .post(controller.post)

router.route('/:caseId')
    .delete(controller.delete)

router.route('/:caseId')
    .put(controller.update)
```

3. History

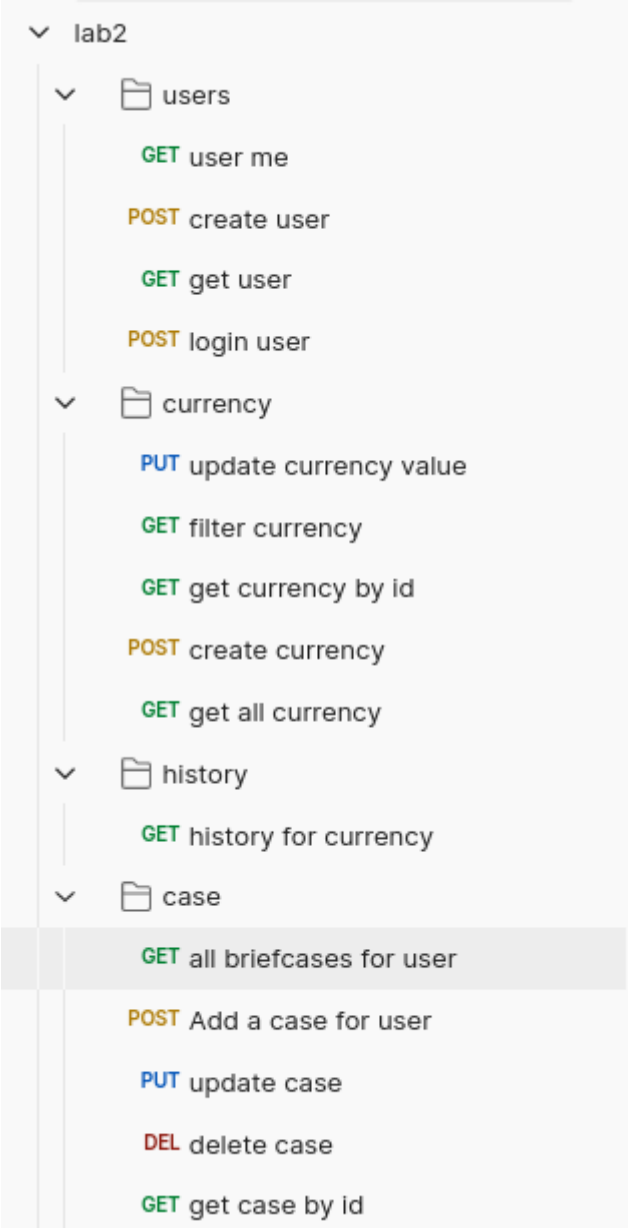
```
const controller: HistoryController = new HistoryController()

router.route('/:id')
    .get(controller.get)
```

4. Файл index.ts

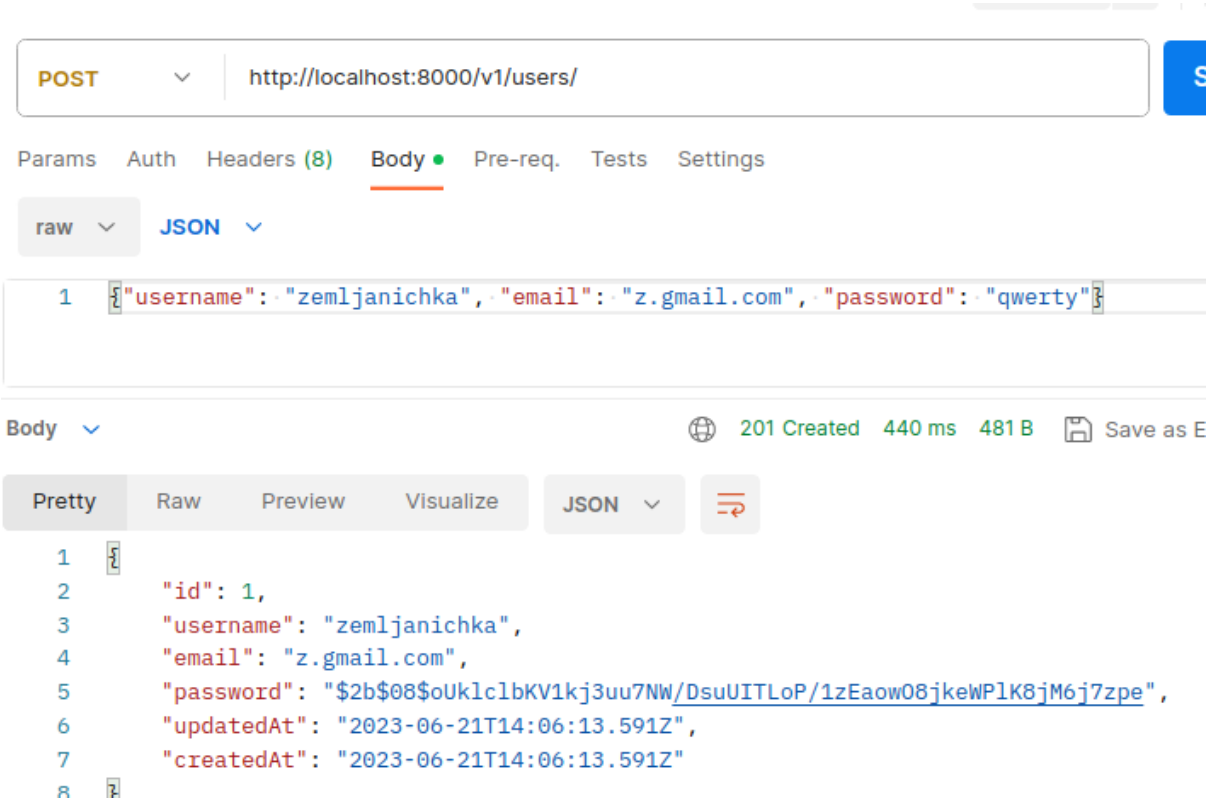
```
const router: express.Router = express.Router()

router.use('/users', userRoutes)
router.use('/currency', currencyRoutes)
router.use('/briefcases', briefcaseRoutes)
router.use('/history', historyRoutes)
```

Проверка работы через postman

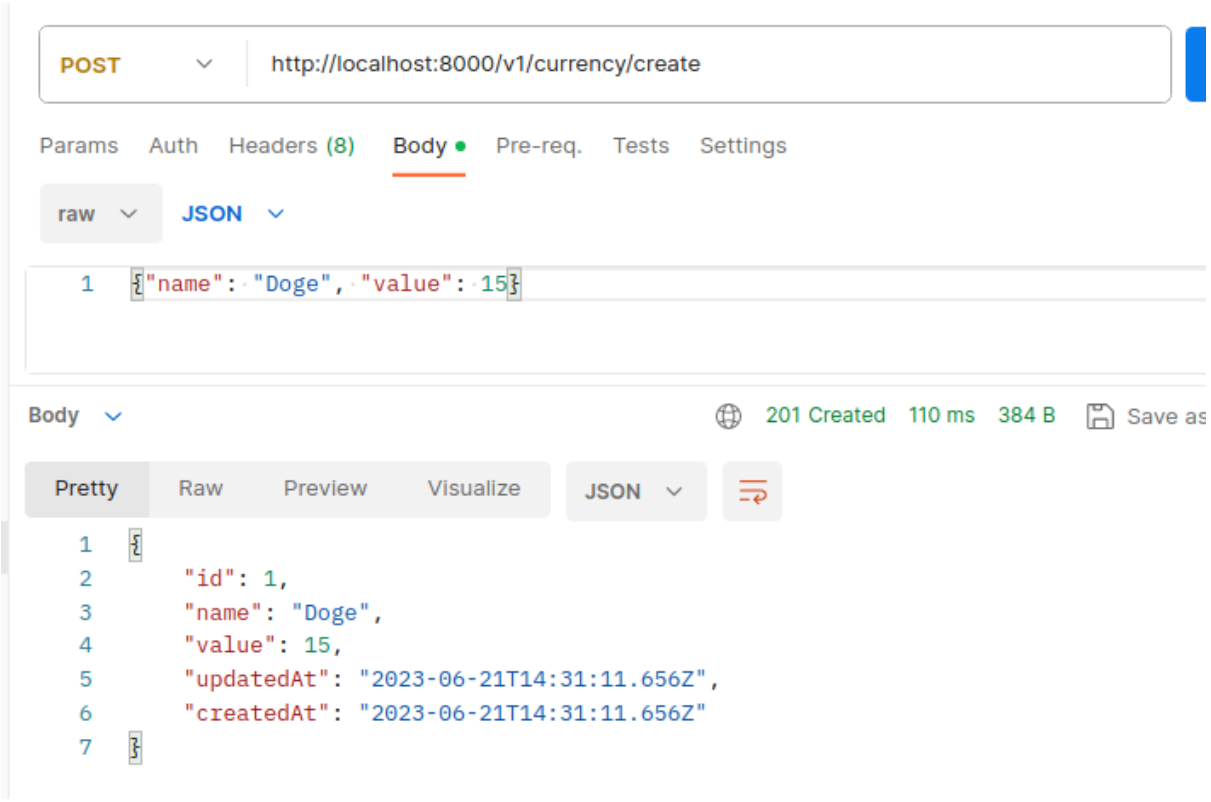
Все созданные запросы:



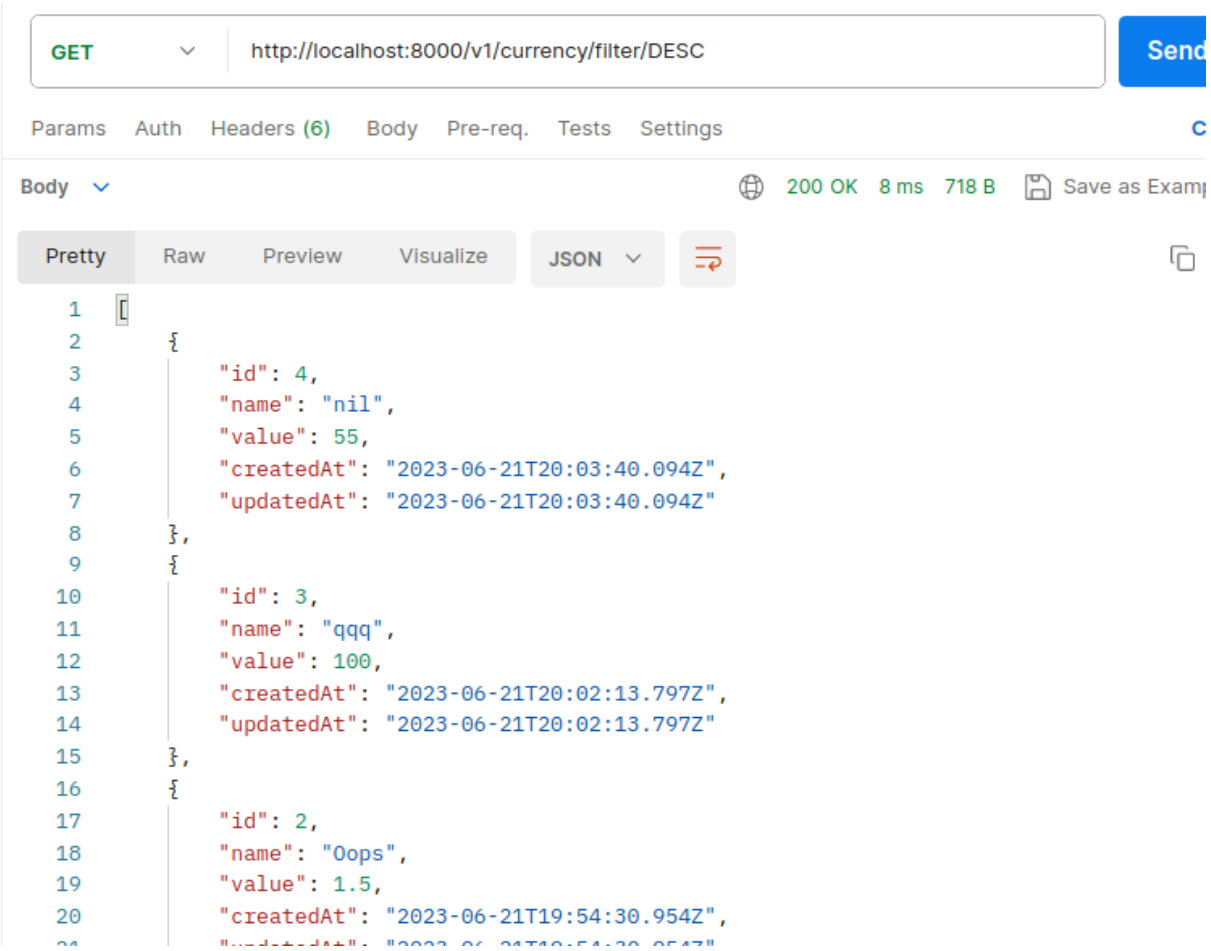
Некоторые из запросов:
Создание пользователя



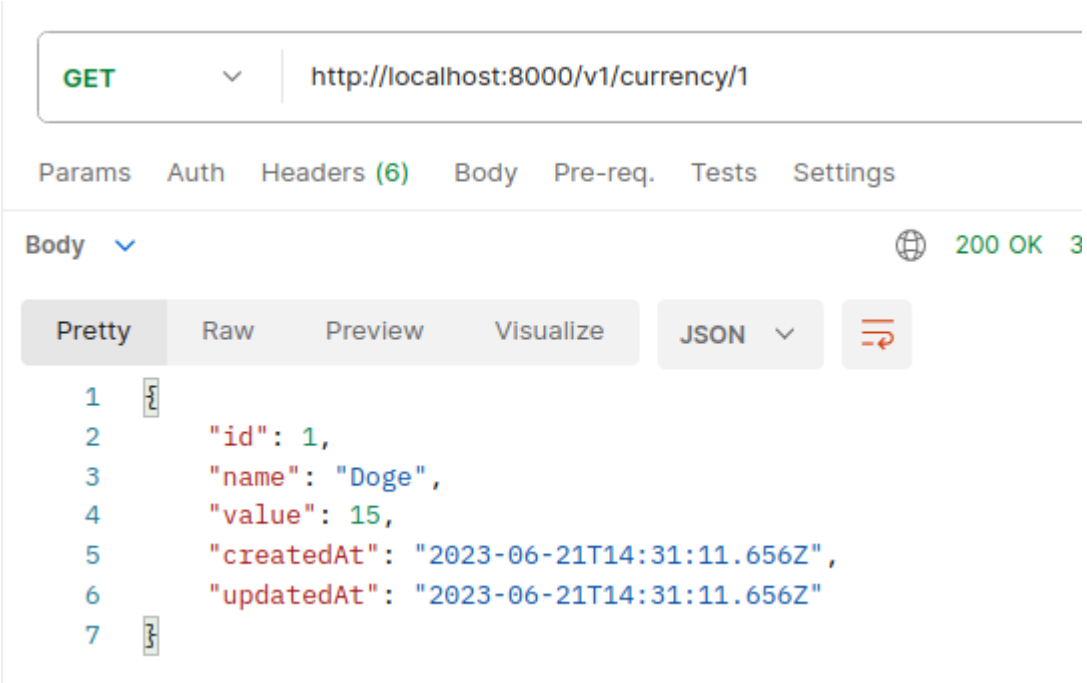
Создание валюты



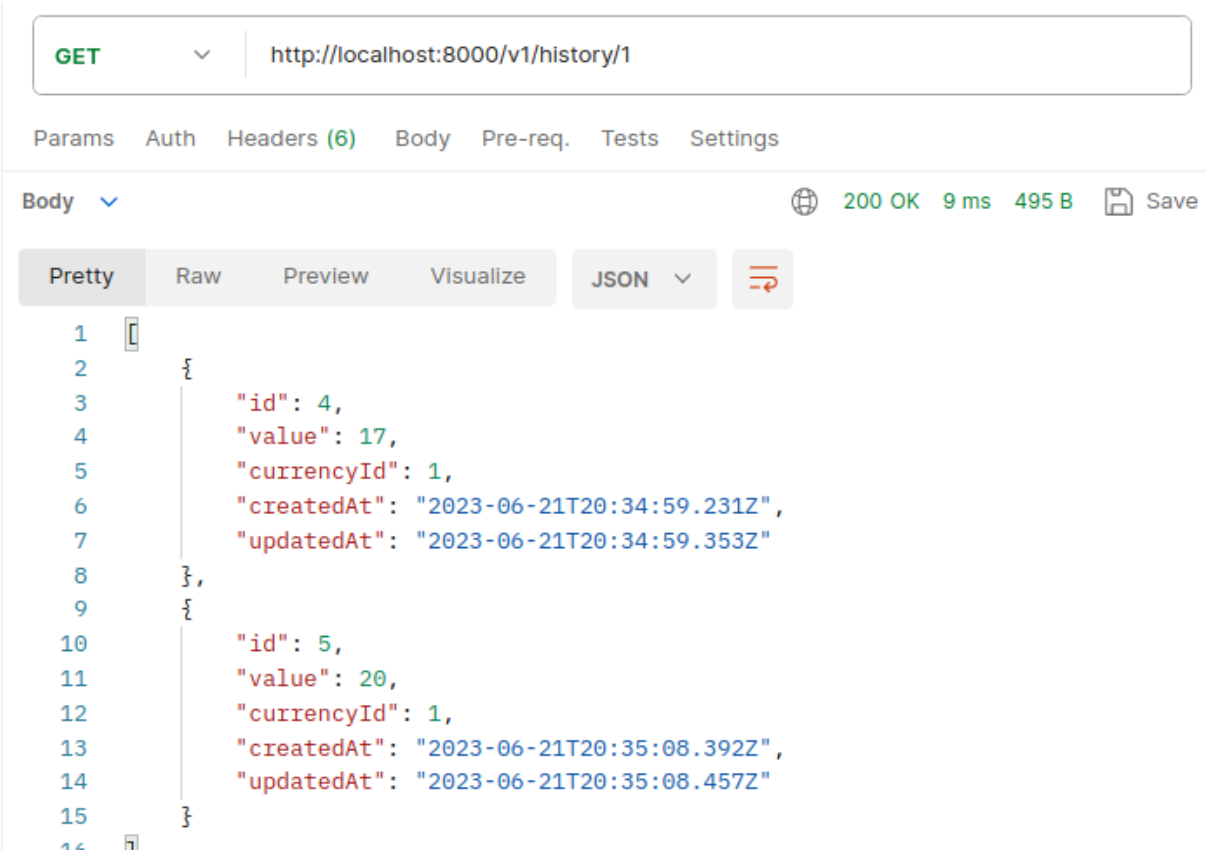
Получение отсортированного списка валют



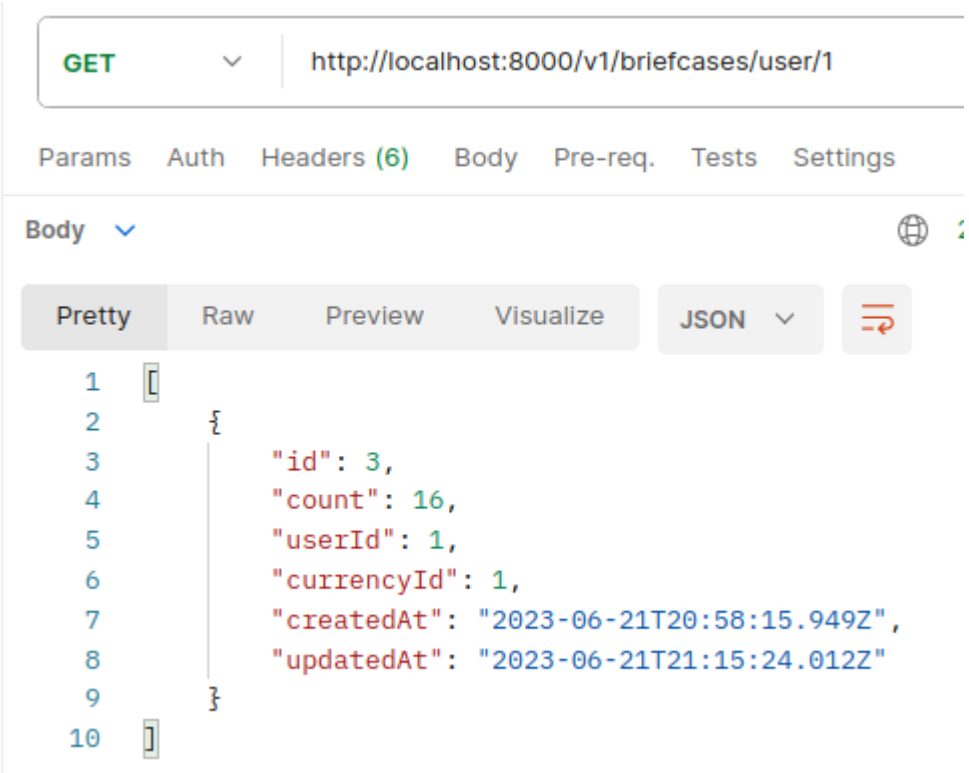
Получение валюты по id



Получение истории валюты



Получение портфеля пользователя



Остальные также были проверены на работоспособность, но не включены в отчет.

Вывод

В ходе работы был реализован RESTful API на основе ранее написанного boilerplate. Реализованы три новые модели, три сервиса и три контроллера. Добавлены все необходимые эндпойнты. Проверка работы эндпойнтов осуществлялась с помощью postman.