САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №2

Выполнил:

Егоров Мичил

Группа

К33401

Проверил:
Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

Необходимо реализовать RESTful API средствами express + typescript.

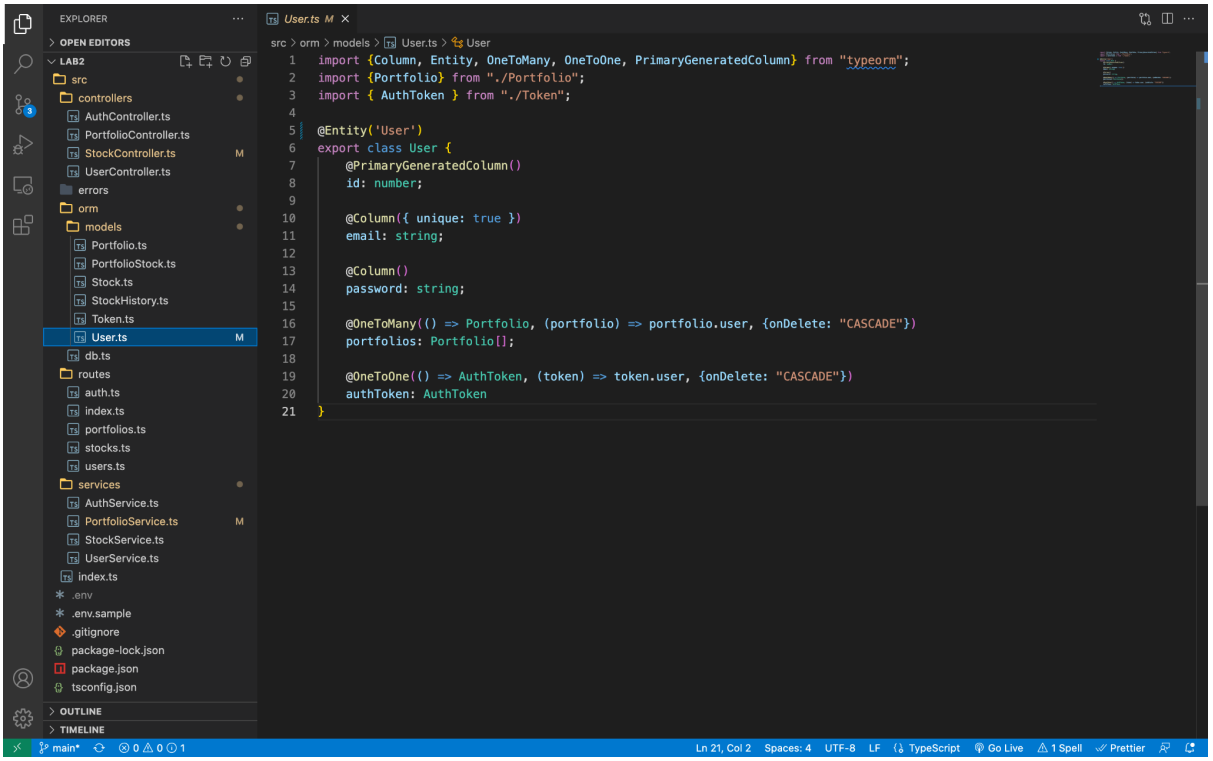
Сайт криптобиржи:

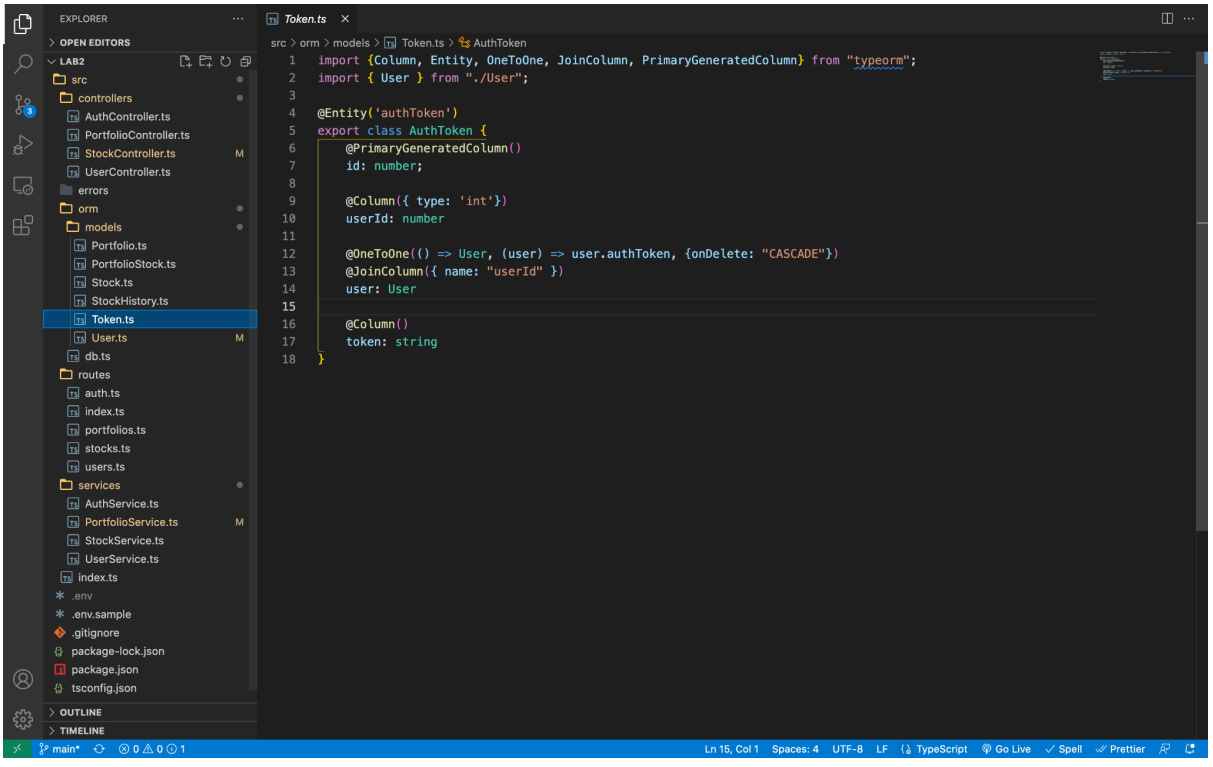
- Вход
- Регистрация
- Портфель пользователя с указанием различных криптовалют и их количеством
- Графики роста криптовалют
- Поиск по криптовалютам с возможностью фильтрации по дате добавления на биржу

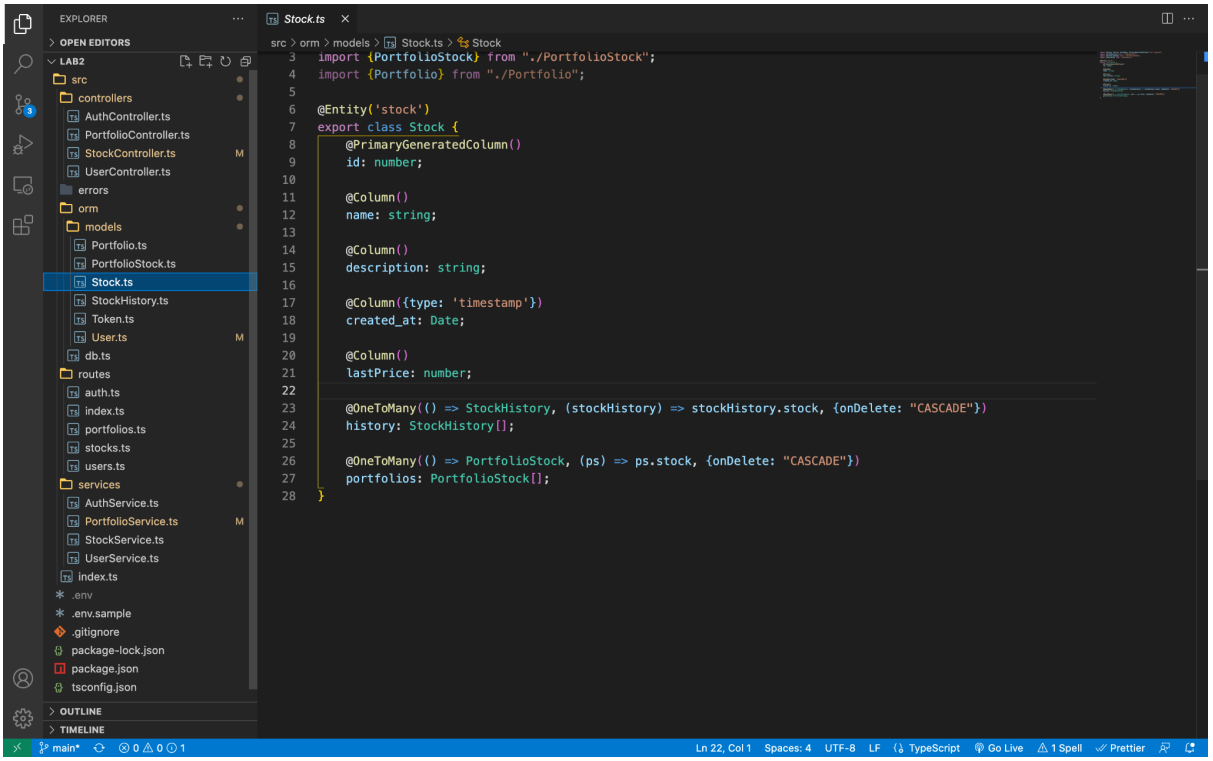
Ход работы

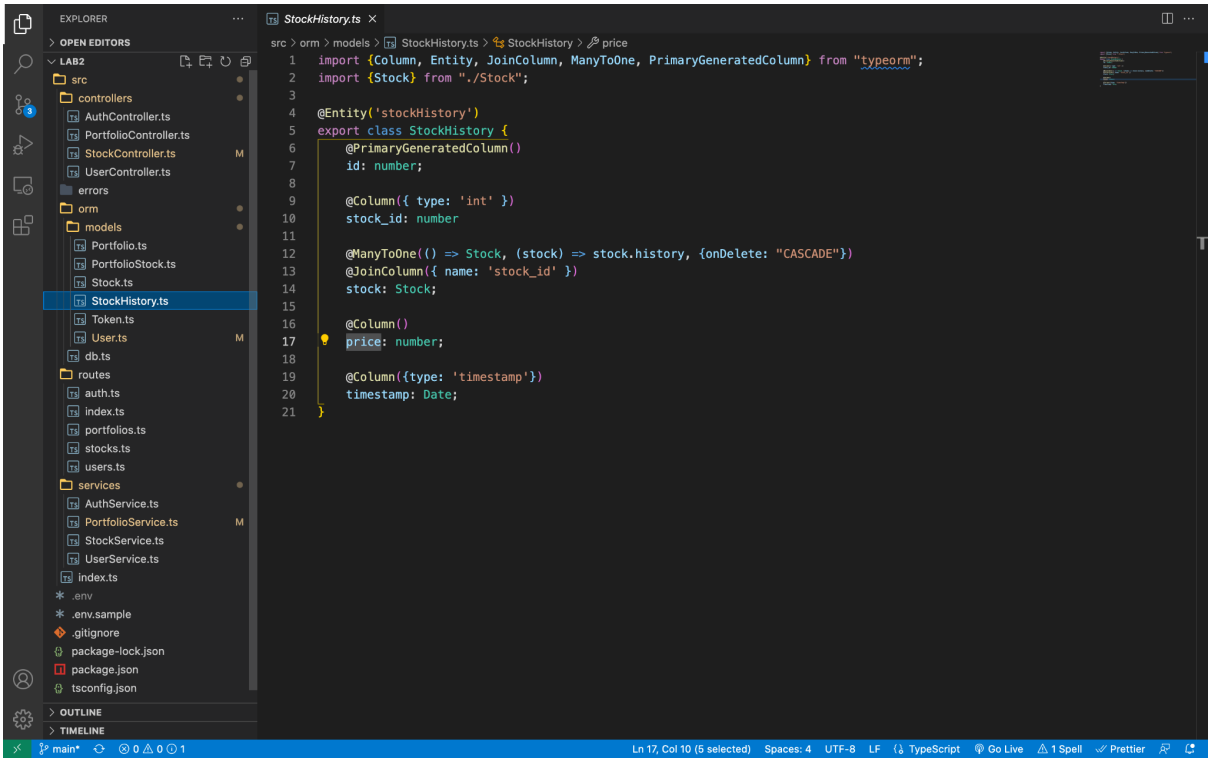
1. Структура приложения

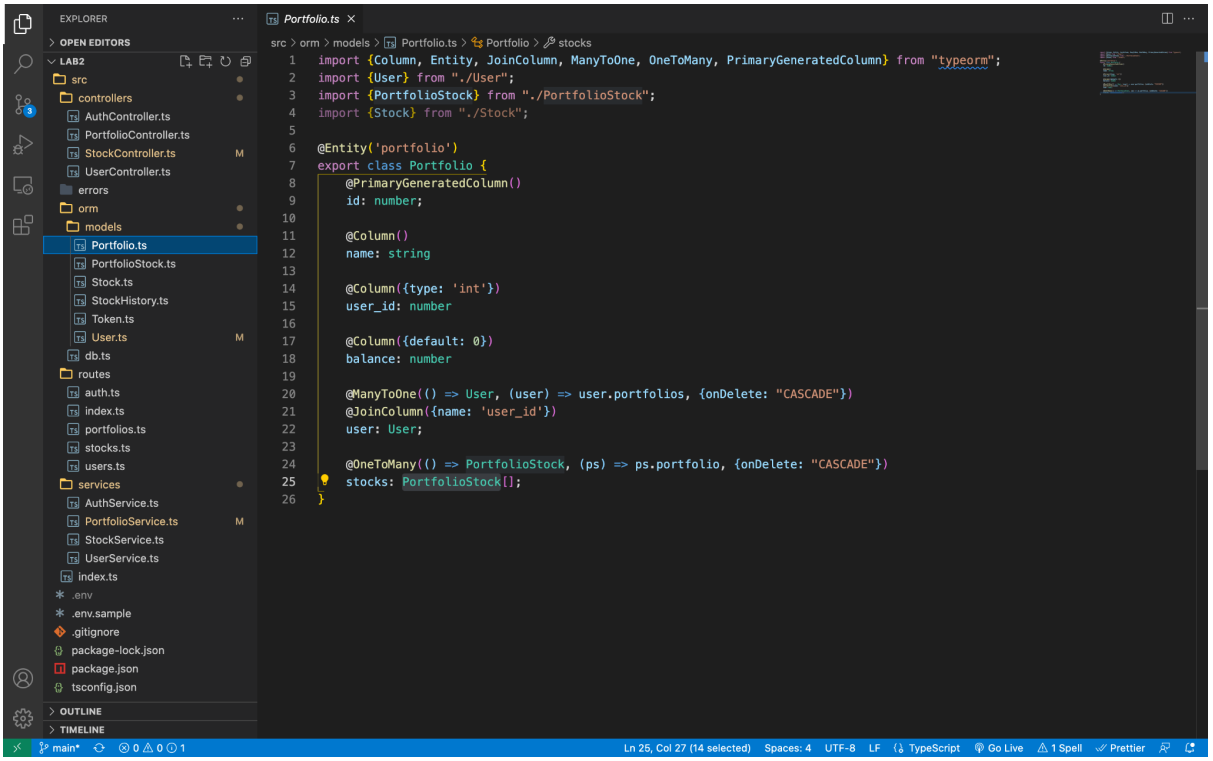
2. Модели

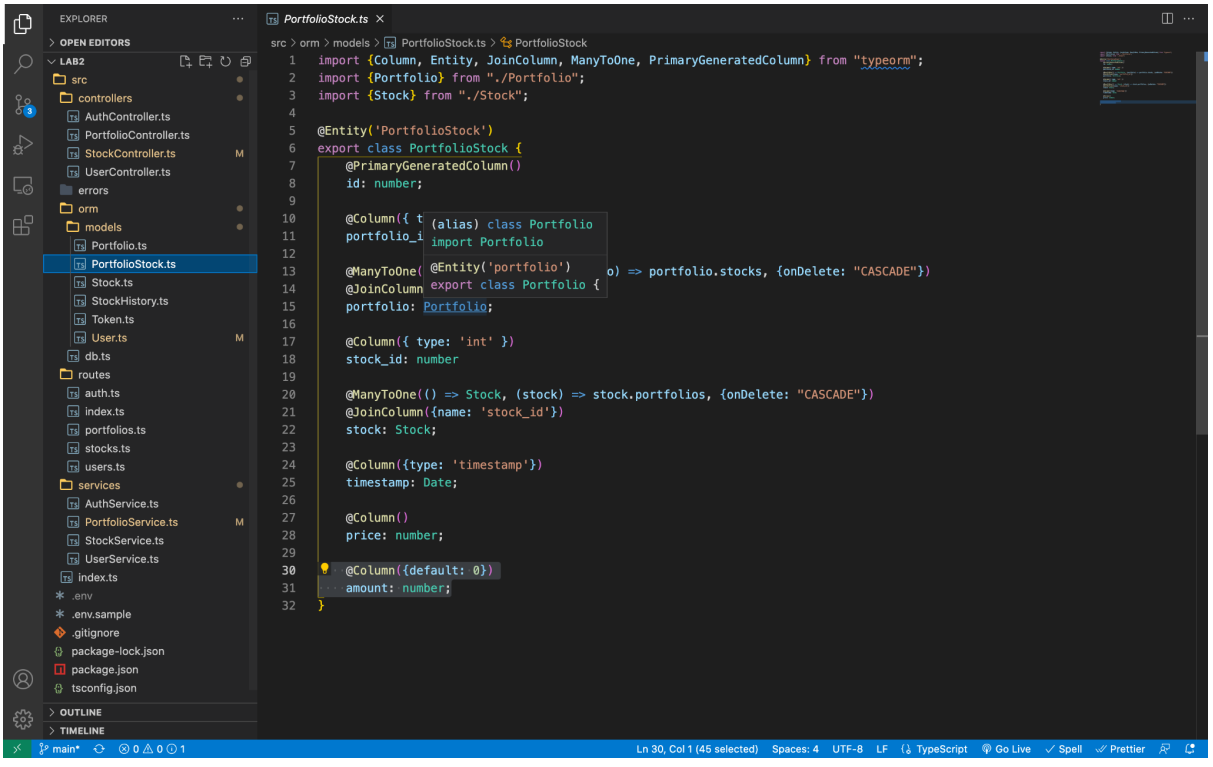

EXPLORER
OPEN EDITORS
LAB2
src
controllers
AuthController.ts
PortfolioController.ts
StockController.ts M
UserController.ts
errors
orm
models
Portfolio.ts
PortfolioStock.ts
Stock.ts
StockHistory.ts
Token.ts
User.ts M
db.ts
routes
auth.ts
index.ts
portfolios.ts
stocks.ts
users.ts
services
AuthService.ts
PortfolioService.ts M
StockService.ts
UserService.ts
index.ts
.env
.env.sample
.gitignore
package-lock.json
package.json
tsconfig.json
OUTLINE
TIMELINE
User.ts M
src > orm > models > User.ts > User
1 import {Column, Entity, OneToMany, OneToOne, PrimaryGeneratedColumn} from "typeorm";
2 import {Portfolio} from "./Portfolio";
3 import { AuthToken } from "./Token";
4
5 @Entity('User')
6 export class User {
7 @PrimaryGeneratedColumn()
8 id: number;
9
10 @Column({ unique: true })
11 email: string;
12
13 @Column()
14 password: string;
15
16 @OneToMany(() => Portfolio, (portfolio) => portfolio.user, {onDelete: "CASCADE"})
17 portfolios: Portfolio[];
18
19 @OneToOne(() => AuthToken, (token) => token.user, {onDelete: "CASCADE"})
20 authToken: AuthToken
21 }
main*
0 0 1
Ln 21, Col 2
Spaces: 4
UTF-8
LF
TypeScript
Go Live
1 Spell
Prettier


EXPLORER
OPEN EDITORS
LAB2
src
controllers
AuthController.ts
PortfolioController.ts
StockController.ts M
UserController.ts
errors
orm
models
Portfolio.ts
PortfolioStock.ts
Stock.ts
StockHistory.ts
Token.ts
User.ts M
db.ts
routes
auth.ts
index.ts
portfolios.ts
stocks.ts
users.ts
services
AuthService.ts
PortfolioService.ts M
StockService.ts
UserService.ts
index.ts
.env
.env.sample
.gitignore
package-lock.json
package.json
tsconfig.json
OUTLINE
TIMELINE
Token.ts
src > orm > models > Token.ts > AuthToken
1 import {Column, Entity, OneToOne, JoinColumn, PrimaryGeneratedColumn} from "typeorm";
2 import { User } from "./User";
3
4 @Entity('authToken')
5 export class AuthToken {
6 @PrimaryGeneratedColumn()
7 id: number;
8
9 @Column({ type: 'int'})
10 userId: number
11
12 @OneToOne(() => User, (user) => user.authToken, {onDelete: "CASCADE"})
13 @JoinColumn({ name: "userId" })
14 user: User
15
16 @Column()
17 token: string
18 }
main*
0 0 1
Ln 15, Col 1
Spaces: 4
UTF-8
LF
TypeScript
Go Live
Spell
Prettier

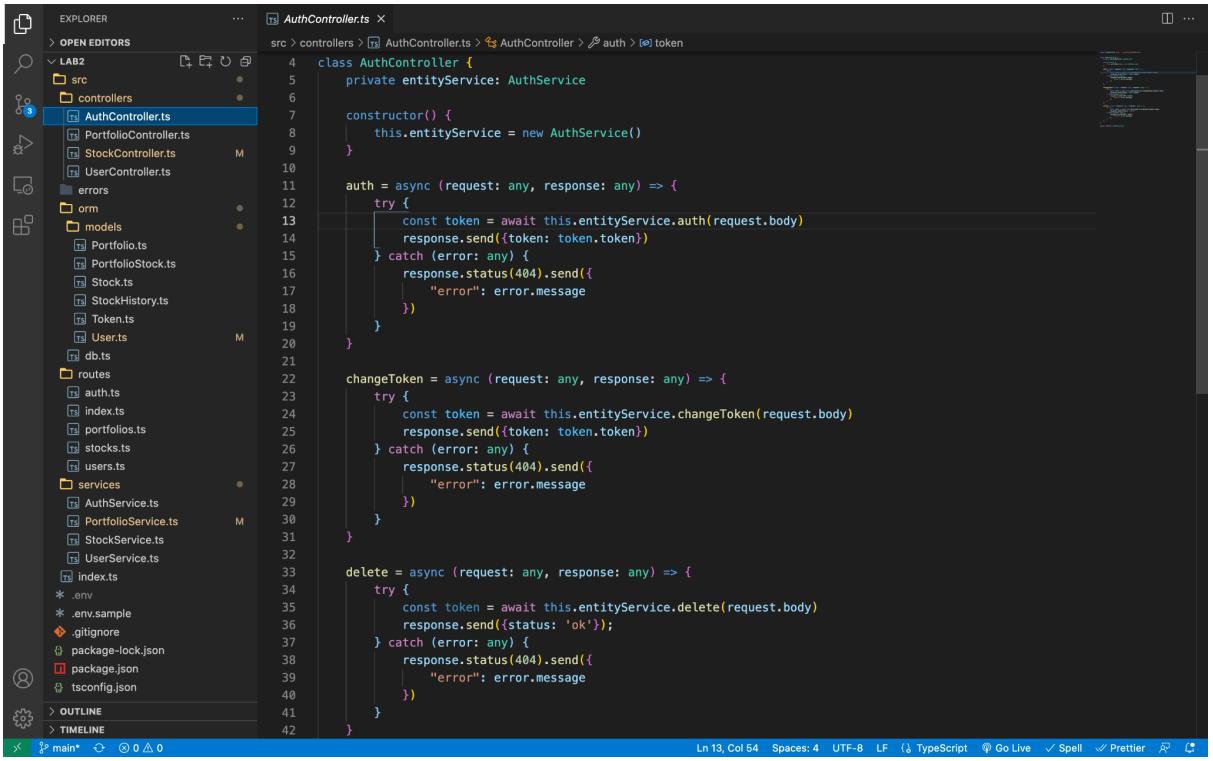

EXPLORER
OPEN EDITORS
LAB2
src
controllers
AuthController.ts
PortfolioController.ts
StockController.ts M
UserController.ts
errors
orm
models
Portfolio.ts
PortfolioStock.ts
Stock.ts
StockHistory.ts
Token.ts
User.ts M
db.ts
routes
auth.ts
index.ts
portfolios.ts
stocks.ts
users.ts
services
AuthService.ts
PortfolioService.ts M
StockService.ts
UserService.ts
index.ts
.env
.env.sample
.gitignore
package-lock.json
package.json
tsconfig.json
OUTLINE
TIMELINE
Stock.ts
src > orm > models > Stock.ts > Stock
3 import {PortfolioStock} from "./PortfolioStock";
4 import {Portfolio} from "./Portfolio";
5
6 @Entity('stock')
7 export class Stock {
8 @PrimaryGeneratedColumn()
9 id: number;
10
11 @Column()
12 name: string;
13
14 @Column()
15 description: string;
16
17 @Column({type: 'timestamp'})
18 created_at: Date;
19
20 @Column()
21 lastPrice: number;
22
23 @OneToMany(() => StockHistory, (stockHistory) => stockHistory.stock, {onDelete: "CASCADE"})
24 history: StockHistory[];
25
26 @OneToMany(() => PortfolioStock, (ps) => ps.stock, {onDelete: "CASCADE"})
27 portfolios: PortfolioStock[];
28 }
main*
0 0 1
Ln 22, Col 1
Spaces: 4
UTF-8
LF
TypeScript
Go Live
1 Spell
Prettier

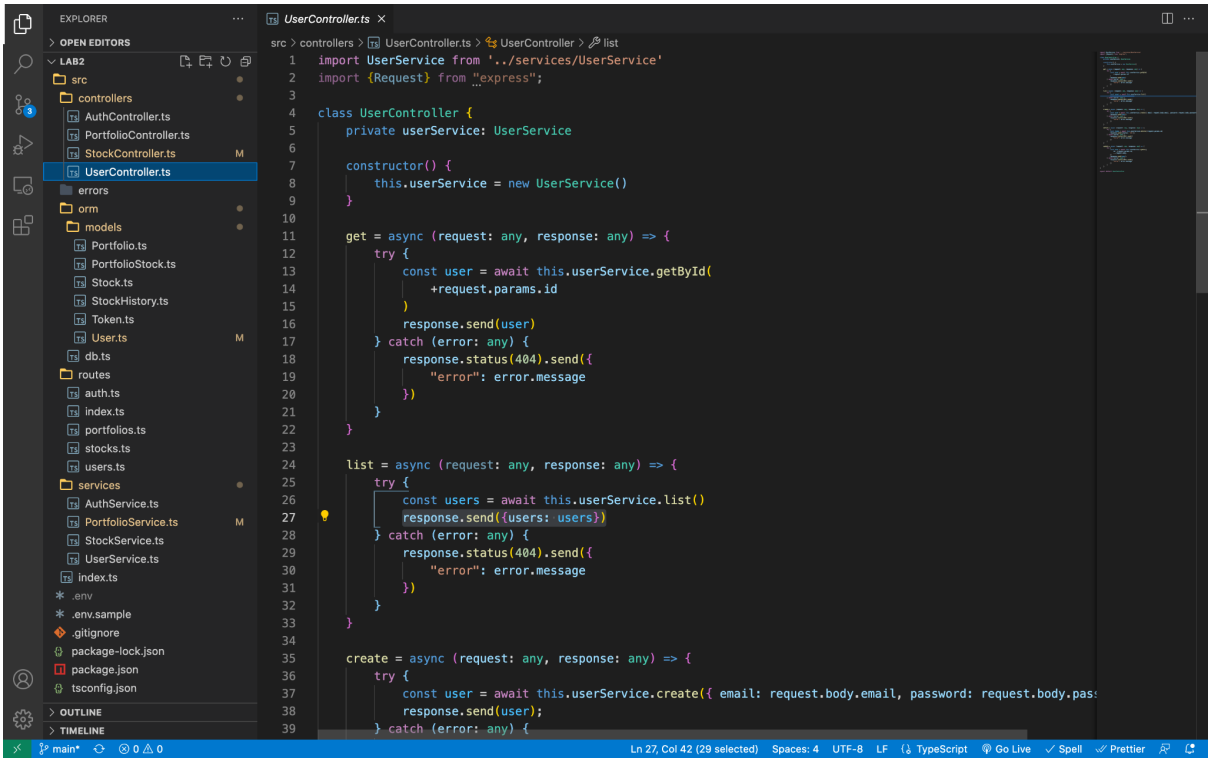

EXPLORER
OPEN EDITORS
LAB2
src
controllers
AuthController.ts
PortfolioController.ts
StockController.ts M
UserController.ts
errors
orm
models
Portfolio.ts
PortfolioStock.ts
Stock.ts
StockHistory.ts
Token.ts
User.ts M
db.ts
routes
auth.ts
index.ts
portfolios.ts
stocks.ts
users.ts
services
AuthService.ts
PortfolioService.ts M
StockService.ts
UserService.ts
index.ts
.env
.env.sample
.gitignore
package-lock.json
package.json
tsconfig.json
OUTLINE
TIMELINE
StockHistory.ts
src > orm > models > StockHistory.ts > StockHistory > price
1 import {Column, Entity, JoinColumn, ManyToOne, PrimaryGeneratedColumn} from "typeorm";
2 import {Stock} from "./Stock";
3
4 @Entity('stockHistory')
5 export class StockHistory {
6 @PrimaryGeneratedColumn()
7 id: number;
8
9 @Column({ type: 'int' })
10 stock_id: number
11
12 @ManyToOne(() => Stock, (stock) => stock.history, {onDelete: "CASCADE"})
13 @JoinColumn({ name: 'stock_id' })
14 stock: Stock;
15
16 @Column()
17 price: number;
18
19 @Column({type: 'timestamp'})
20 timestamp: Date;
21 }
main*
0 0 1
Ln 17, Col 10 (5 selected)
Spaces: 4
UTF-8
LF
TypeScript
Go Live
1 Spell
Prettier

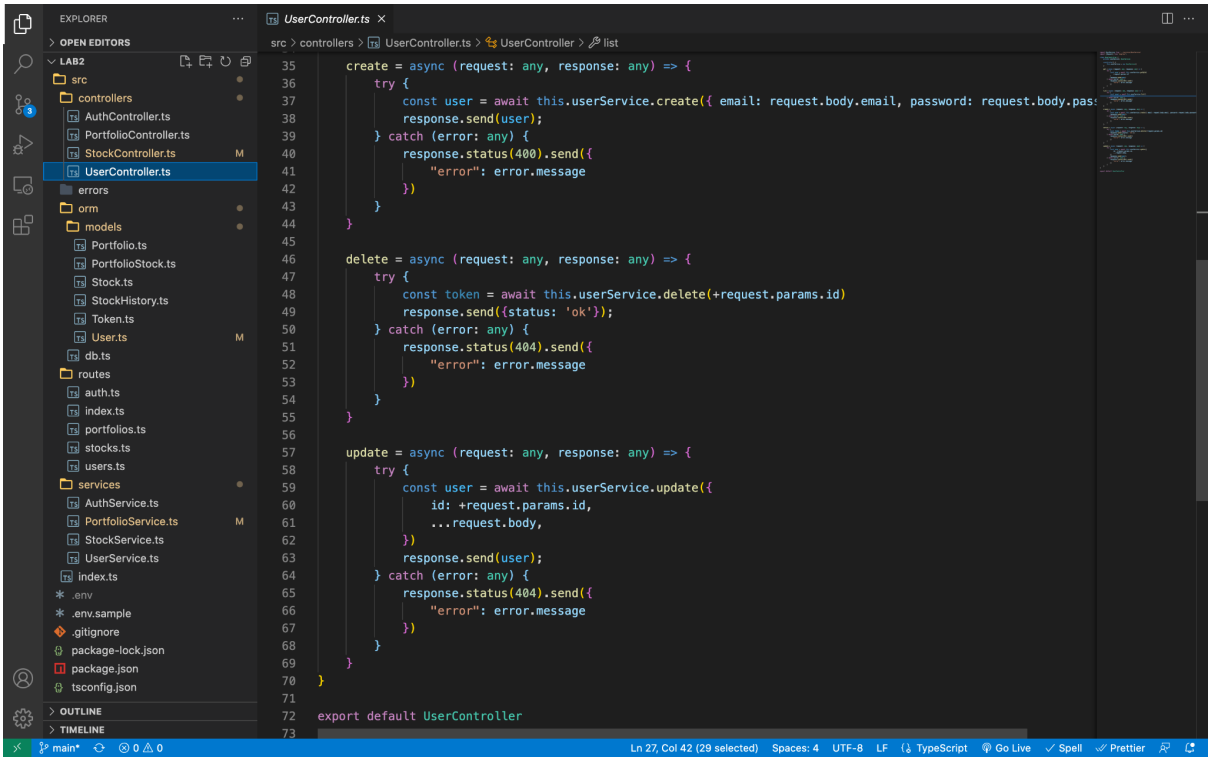


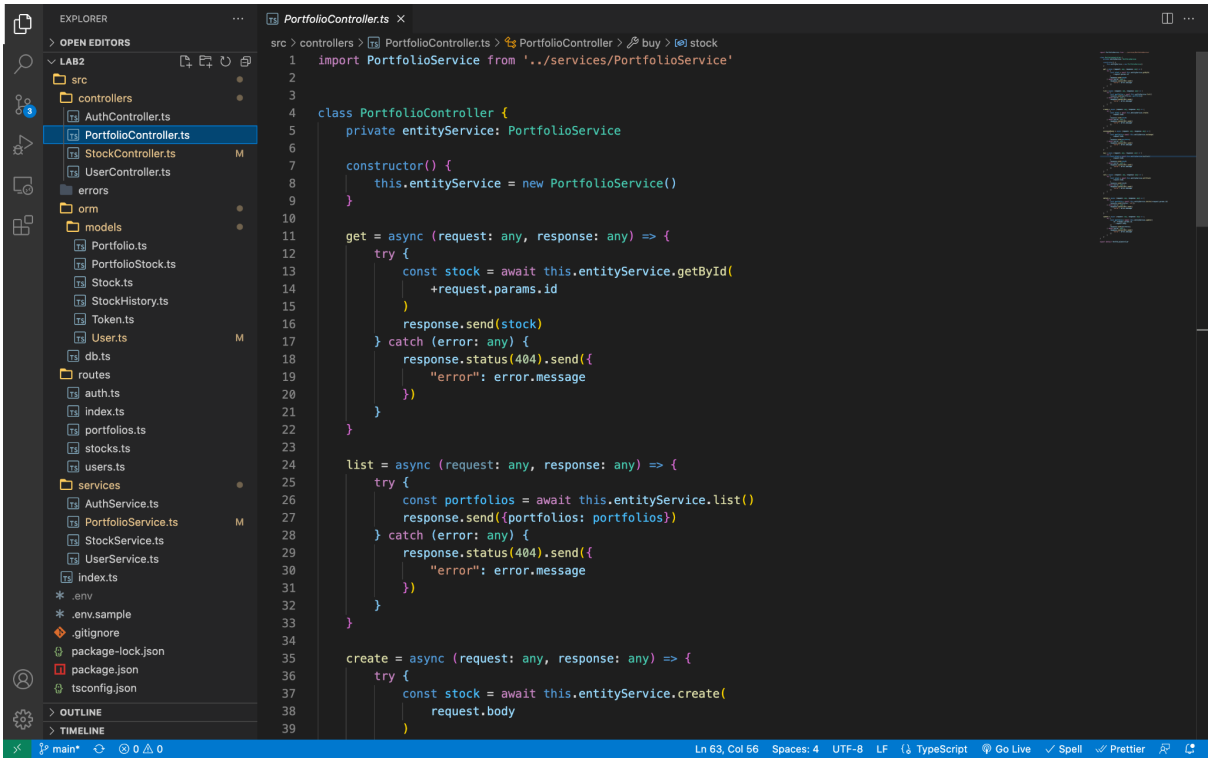


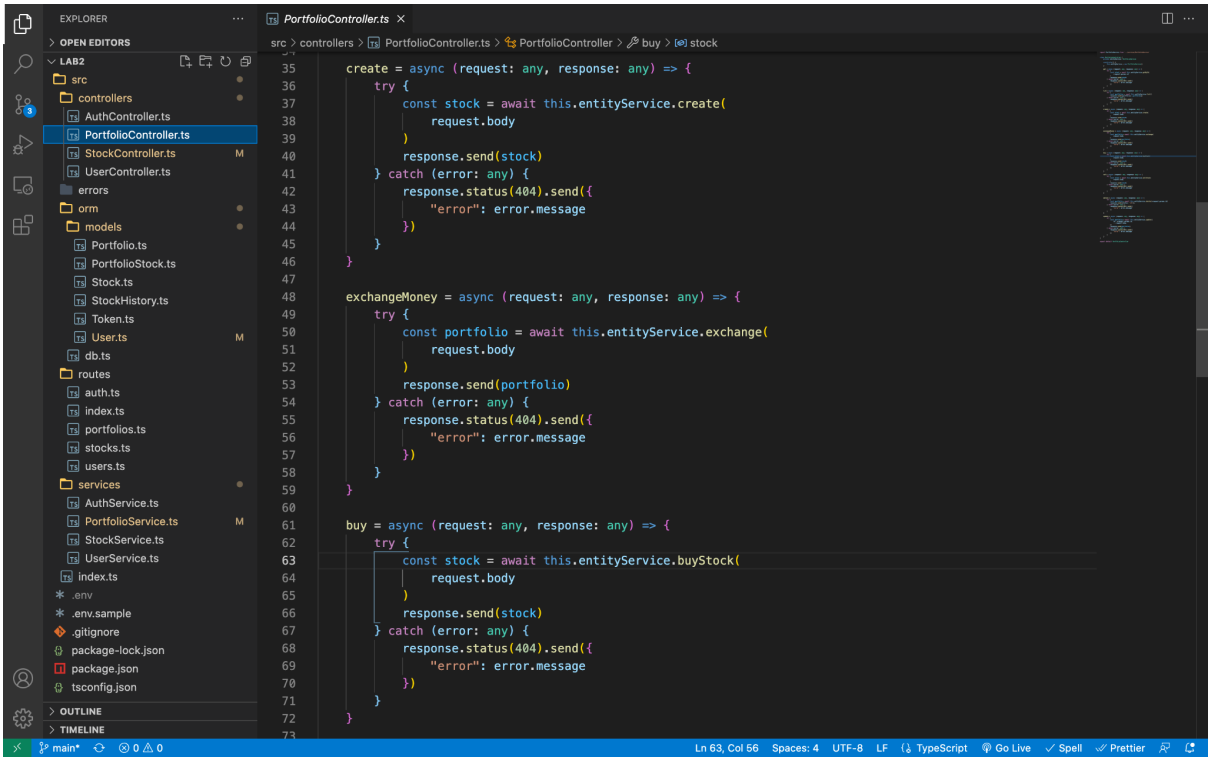
3. Контроллеры

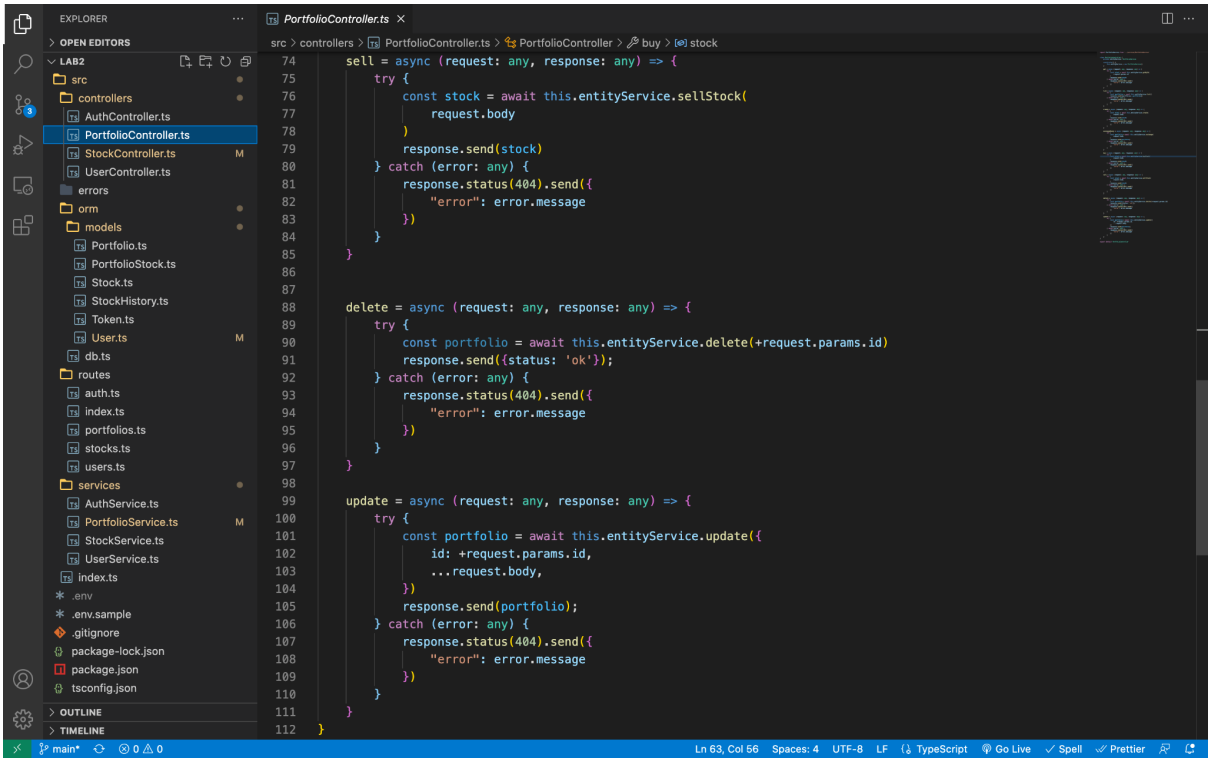

EXPLORER
OPEN EDITORS
LAB2
src
controllers
AuthController.ts
PortfolioController.ts
StockController.ts M
UserController.ts
errors
orm
models
Portfolio.ts
PortfolioStock.ts
Stock.ts
StockHistory.ts
Token.ts
User.ts M
db.ts
routes
auth.ts
index.ts
portfolios.ts
stocks.ts
users.ts
services
AuthService.ts
PortfolioService.ts M
StockService.ts
UserService.ts
index.ts
.env
.env.sample
.gitignore
package-lock.json
package.json
tsconfig.json
OUTLINE
TIMELINE
AuthController.ts
src > controllers > AuthController.ts > AuthController > auth > token
class AuthController {
private entityService: AuthService
constructor() {
this.entityService = new AuthService()
}
auth = async (request: any, response: any) => {
try {
const token = await this.entityService.auth(request.body)
response.send({token: token.token})
} catch (error: any) {
response.status(404).send({
"error": error.message
})
}
}
changeToken = async (request: any, response: any) => {
try {
const token = await this.entityService.changeToken(request.body)
response.send({token: token.token})
} catch (error: any) {
response.status(404).send({
"error": error.message
})
}
}
delete = async (request: any, response: any) => {
try {
const token = await this.entityService.delete(request.body)
response.send({status: 'ok'});
} catch (error: any) {
response.status(404).send({
"error": error.message
})
}
}
main*
0 0
Ln 13, Col 54
Spaces: 4
UTF-8
LF
TypeScript
Go Live
Spell
Prettier

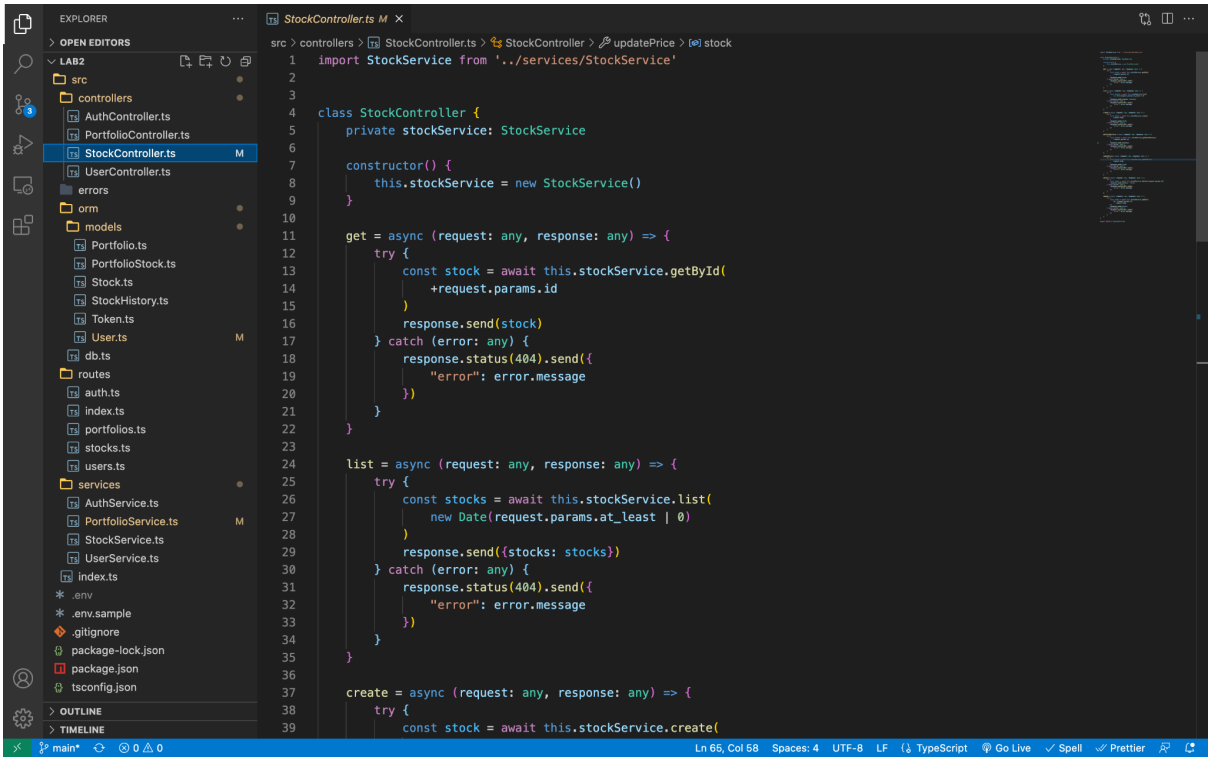

EXPLORER
OPEN EDITORS
LAB2
src
controllers
AuthController.ts
PortfolioController.ts
StockController.ts M
UserController.ts
errors
orm
models
Portfolio.ts
PortfolioStock.ts
Stock.ts
StockHistory.ts
Token.ts
User.ts M
db.ts
routes
auth.ts
index.ts
portfolios.ts
stocks.ts
users.ts
services
AuthService.ts
PortfolioService.ts M
StockService.ts
UserService.ts
index.ts
.env
.env.sample
.gitignore
package-lock.json
package.json
tsconfig.json
OUTLINE
TIMELINE
UserController.ts
src > controllers > UserController.ts > UserController > list
import UserService from '../services/UserService'
import {Request} from "express";
class UserController {
private userService: UserService
constructor() {
this.userService = new UserService()
}
get = async (request: any, response: any) => {
try {
const user = await this.userService.getById(
+request.params.id
)
response.send(user)
} catch (error: any) {
response.status(404).send({
"error": error.message
})
}
}
list = async (request: any, response: any) => {
try {
const users = await this.userService.list()
response.send({users: users})
} catch (error: any) {
response.status(404).send({
"error": error.message
})
}
}
create = async (request: any, response: any) => {
try {
const user = await this.userService.create({ email: request.body.email, password: request.body.pas
response.send(user);
} catch (error: any) {
main*
0 0
Ln 27, Col 42 (29 selected)
Spaces: 4
UTF-8
LF
TypeScript
Go Live
Spell
Prettier


EXPLORER
OPEN EDITORS
LAB2
src
controllers
AuthController.ts
PortfolioController.ts
StockController.ts M
UserController.ts
errors
orm
models
Portfolio.ts
PortfolioStock.ts
Stock.ts
StockHistory.ts
Token.ts
User.ts M
db.ts
routes
auth.ts
index.ts
portfolios.ts
stocks.ts
users.ts
services
AuthService.ts
PortfolioService.ts M
StockService.ts
UserService.ts
index.ts
.env
.env.sample
.gitignore
package-lock.json
package.json
tsconfig.json
OUTLINE
TIMELINE
UserController.ts
src > controllers > UserController.ts > UserController > list
create = async (request: any, response: any) => {
try {
const user = await this.userService.create({ email: request.body.email, password: request.body.pas
response.send(user);
} catch (error: any) {
response.status(400).send({
"error": error.message
})
}
}
delete = async (request: any, response: any) => {
try {
const token = await this.userService.delete(+request.params.id)
response.send({status: 'ok'});
} catch (error: any) {
response.status(404).send({
"error": error.message
})
}
}
update = async (request: any, response: any) => {
try {
const user = await this.userService.update({
id: +request.params.id,
...request.body,
})
response.send(user);
} catch (error: any) {
response.status(404).send({
"error": error.message
})
}
}
}
export default UserController
main*
0 0
Ln 27, Col 42 (29 selected)
Spaces: 4
UTF-8
LF
TypeScript
Go Live
Spell
Prettier

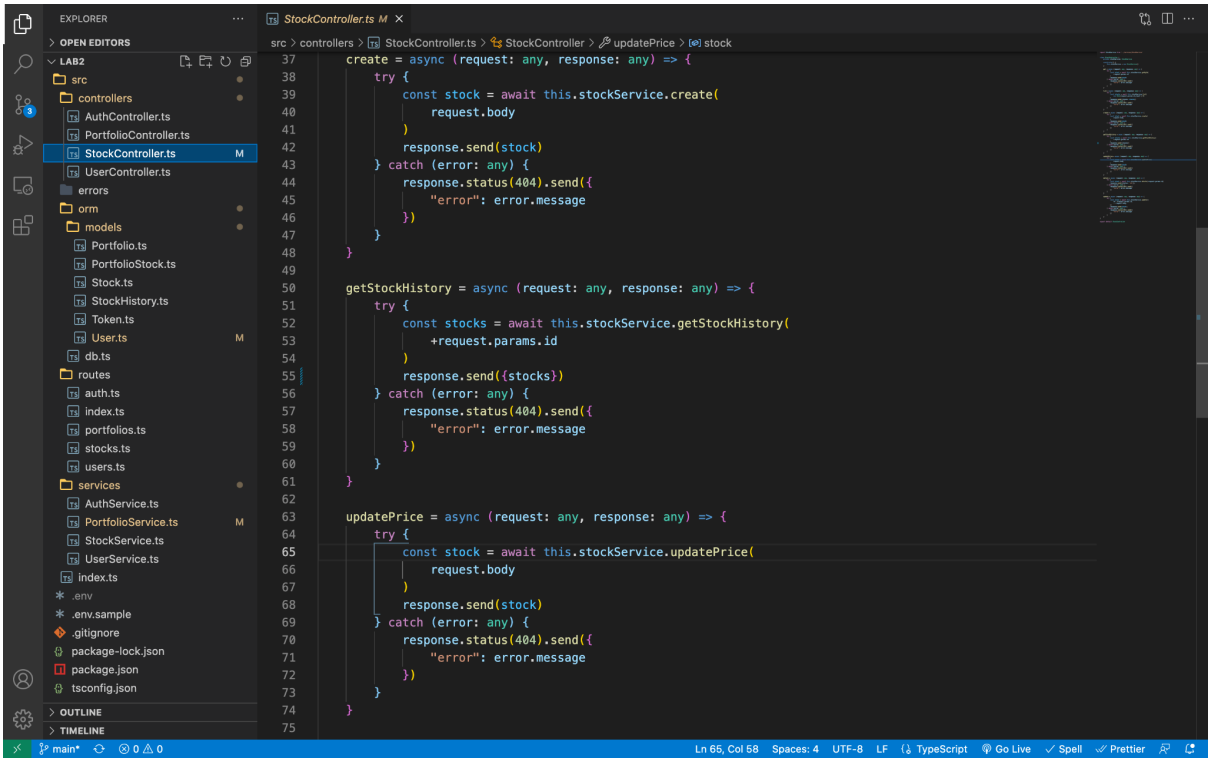

EXPLORER
OPEN EDITORS
LAB2
src
controllers
AuthController.ts
PortfolioController.ts
StockController.ts M
UserController.ts
errors
orm
models
Portfolio.ts
PortfolioStock.ts
Stock.ts
StockHistory.ts
Token.ts
User.ts M
db.ts
routes
auth.ts
index.ts
portfolios.ts
stocks.ts
users.ts
services
AuthService.ts
PortfolioService.ts M
StockService.ts
UserService.ts
index.ts
.env
.env.sample
.gitignore
package-lock.json
package.json
tsconfig.json
OUTLINE
TIMELINE
PortfolioController.ts
src > controllers > PortfolioController.ts > PortfolioController > buy > stock
import PortfolioService from '../services/PortfolioService'
class PortfolioController {
private entityService: PortfolioService
constructor() {
this.entityService = new PortfolioService()
}
get = async (request: any, response: any) => {
try {
const stock = await this.entityService.getById(
+request.params.id
)
response.send(stock)
} catch (error: any) {
response.status(404).send({
"error": error.message
})
}
}
list = async (request: any, response: any) => {
try {
const portfolios = await this.entityService.list()
response.send({portfolios: portfolios})
} catch (error: any) {
response.status(404).send({
"error": error.message
})
}
}
create = async (request: any, response: any) => {
try {
const stock = await this.entityService.create(
request.body
)
main*
0 0
Ln 63, Col 56
Spaces: 4
UTF-8
LF
TypeScript
Go Live
Spell
Prettier


EXPLORER
OPEN EDITORS
LAB2
src
controllers
AuthController.ts
PortfolioController.ts
StockController.ts M
UserController.ts
errors
orm
models
Portfolio.ts
PortfolioStock.ts
Stock.ts
StockHistory.ts
Token.ts
User.ts M
db.ts
routes
auth.ts
index.ts
portfolios.ts
stocks.ts
users.ts
services
AuthService.ts
PortfolioService.ts M
StockService.ts
UserService.ts
index.ts
.env
.env.sample
.gitignore
package-lock.json
package.json
tsconfig.json
OUTLINE
TIMELINE
PortfolioController.ts
src > controllers > PortfolioController.ts > PortfolioController > buy > stock
create = async (request: any, response: any) => {
try {
const stock = await this.entityService.create(
request.body
)
response.send(stock)
} catch (error: any) {
response.status(404).send({
"error": error.message
})
}
}
exchangeMoney = async (request: any, response: any) => {
try {
const portfolio = await this.entityService.exchange(
request.body
)
response.send(portfolio)
} catch (error: any) {
response.status(404).send({
"error": error.message
})
}
}
buy = async (request: any, response: any) => {
try {
const stock = await this.entityService.buyStock(
request.body
)
response.send(stock)
} catch (error: any) {
response.status(404).send({
"error": error.message
})
}
}
main*
0 0
Ln 63, Col 56
Spaces: 4
UTF-8
LF
TypeScript
Go Live
Spell
Prettier

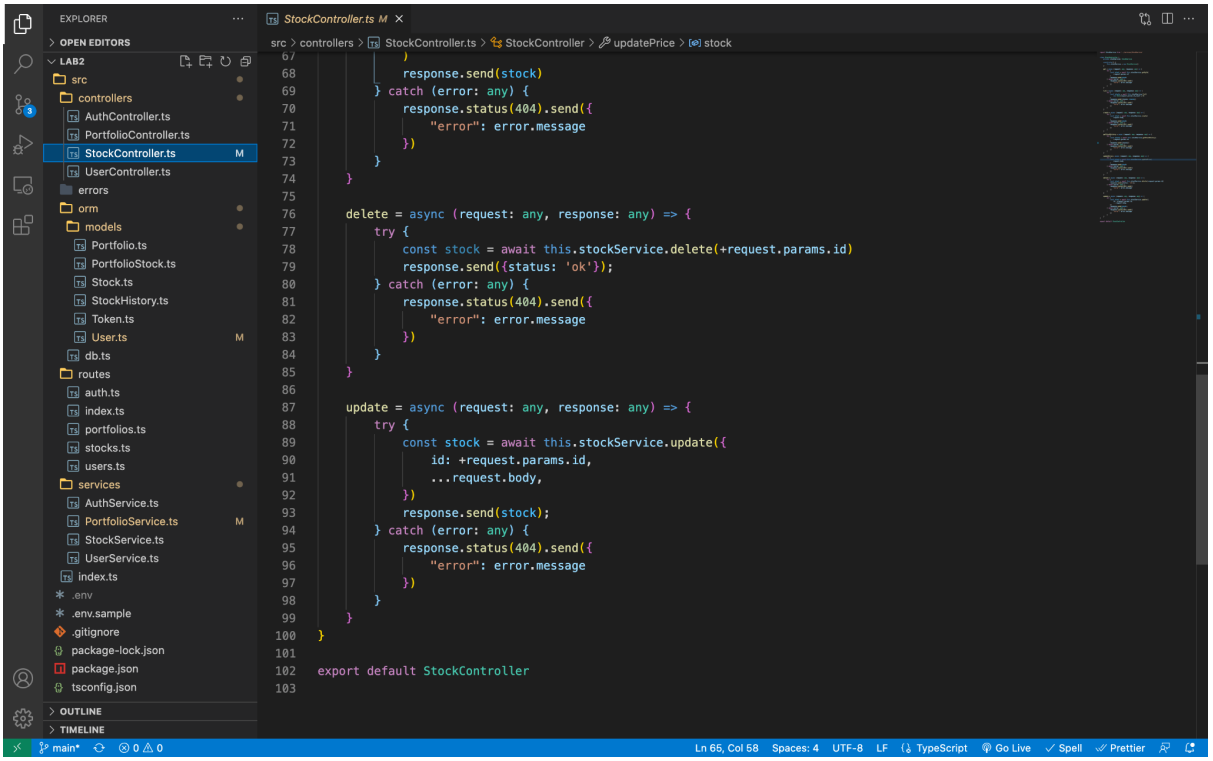

EXPLORER
OPEN EDITORS
LAB2
src
controllers
AuthController.ts
PortfolioController.ts
StockController.ts M
UserController.ts
errors
orm
models
Portfolio.ts
PortfolioStock.ts
Stock.ts
StockHistory.ts
Token.ts
User.ts M
db.ts
routes
auth.ts
index.ts
portfolios.ts
stocks.ts
users.ts
services
AuthService.ts
PortfolioService.ts M
StockService.ts
UserService.ts
index.ts
.env
.env.sample
.gitignore
package-lock.json
package.json
tsconfig.json
OUTLINE
TIMELINE
PortfolioController.ts
src > controllers > PortfolioController.ts > PortfolioController > buy > stock
sell = async (request: any, response: any) => {
try {
const stock = await this.entityService.sellStock(
request.body
)
response.send(stock)
} catch (error: any) {
response.status(404).send({
"error": error.message
})
}
}
delete = async (request: any, response: any) => {
try {
const portfolio = await this.entityService.delete(+request.params.id)
response.send({status: 'ok'});
} catch (error: any) {
response.status(404).send({
"error": error.message
})
}
}
update = async (request: any, response: any) => {
try {
const portfolio = await this.entityService.update({
id: +request.params.id,
...request.body,
})
response.send(portfolio);
} catch (error: any) {
response.status(404).send({
"error": error.message
})
}
}
}
main*
0 0
Ln 63, Col 56
Spaces: 4
UTF-8
LF
TypeScript
Go Live
Spell
Prettier



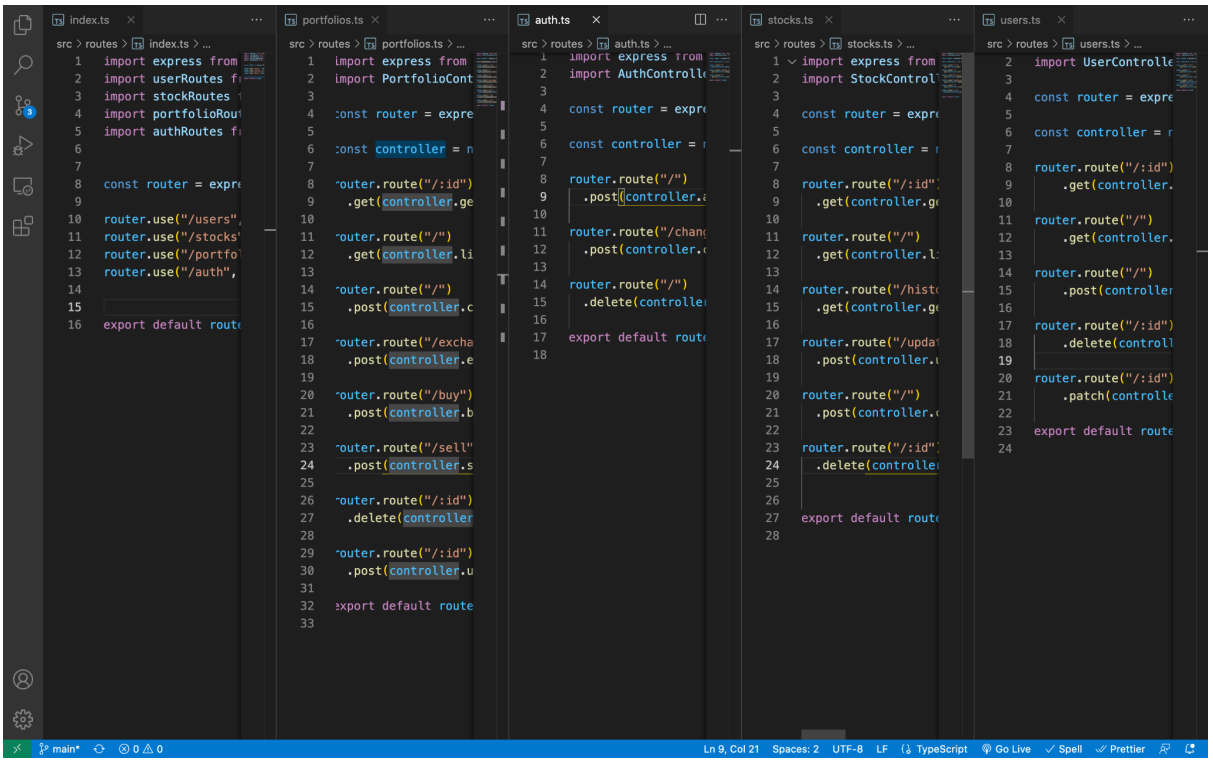

```ts
import StockService from '../services/StockService'


class StockController {
    private stockService: StockService

    constructor() {
        this.stockService = new StockService()
    }

    get = async (request: any, response: any) => {
        try {
            const stock = await this.stockService.getById(
                +request.params.id
            )
            response.send(stock)
        } catch (error: any) {
            response.status(404).send({
                "error": error.message
            })
        }
    }

    list = async (request: any, response: any) => {
        try {
            const stocks = await this.stockService.list(
                new Date(request.params.at_least | 0)
            )
            response.send({stocks: stocks})
        } catch (error: any) {
            response.status(404).send({
                "error": error.message
            })
        }
    }

    create = async (request: any, response: any) => {
        try {
            const stock = await this.stockService.create(
```



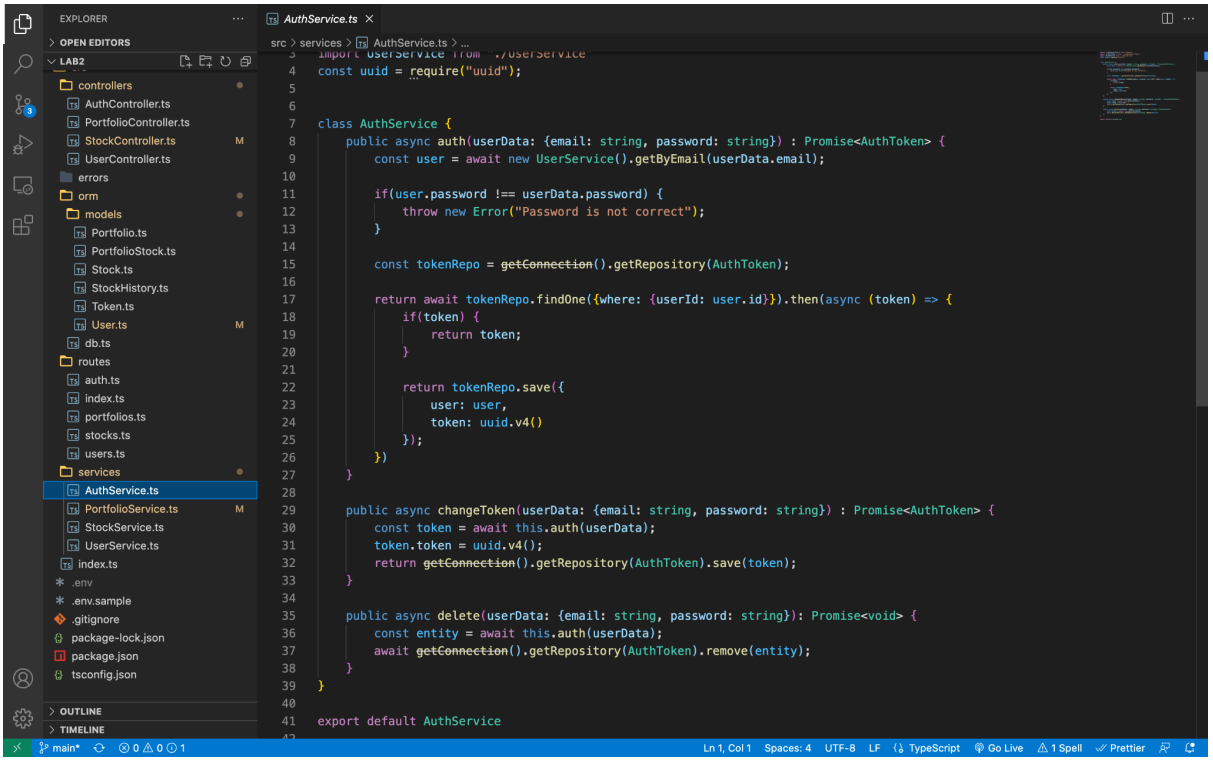

```ts
    create = async (request: any, response: any) => {
        try {
            const stock = await this.stockService.create(
                request.body
            )
            response.send(stock)
        } catch (error: any) {
            response.status(404).send({
                "error": error.message
            })
        }
    }

    getStockHistory = async (request: any, response: any) => {
        try {
            const stocks = await this.stockService.getStockHistory(
                +request.params.id
            )
            response.send({stocks})
        } catch (error: any) {
            response.status(404).send({
                "error": error.message
            })
        }
    }

    updatePrice = async (request: any, response: any) => {
        try {
            const stock = await this.stockService.updatePrice(
                request.body
            )
            response.send(stock)
        } catch (error: any) {
            response.status(404).send({
                "error": error.message
            })
        }
    }

```

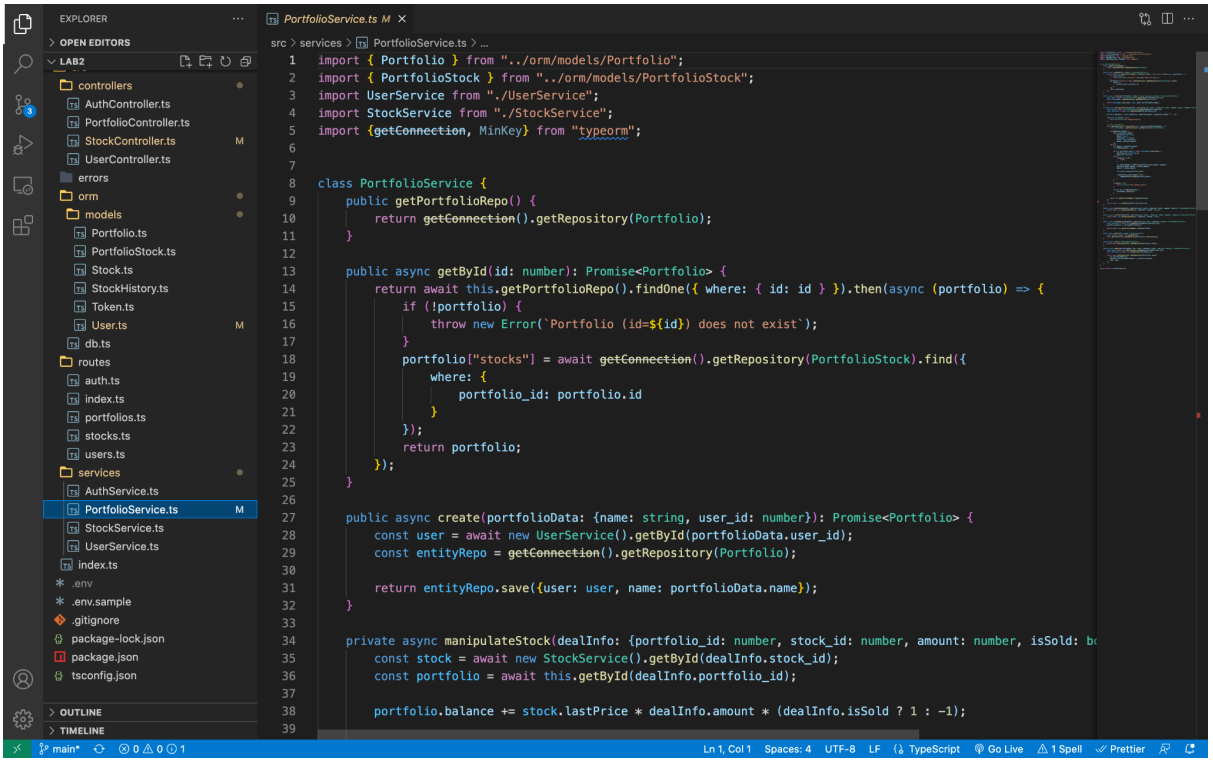


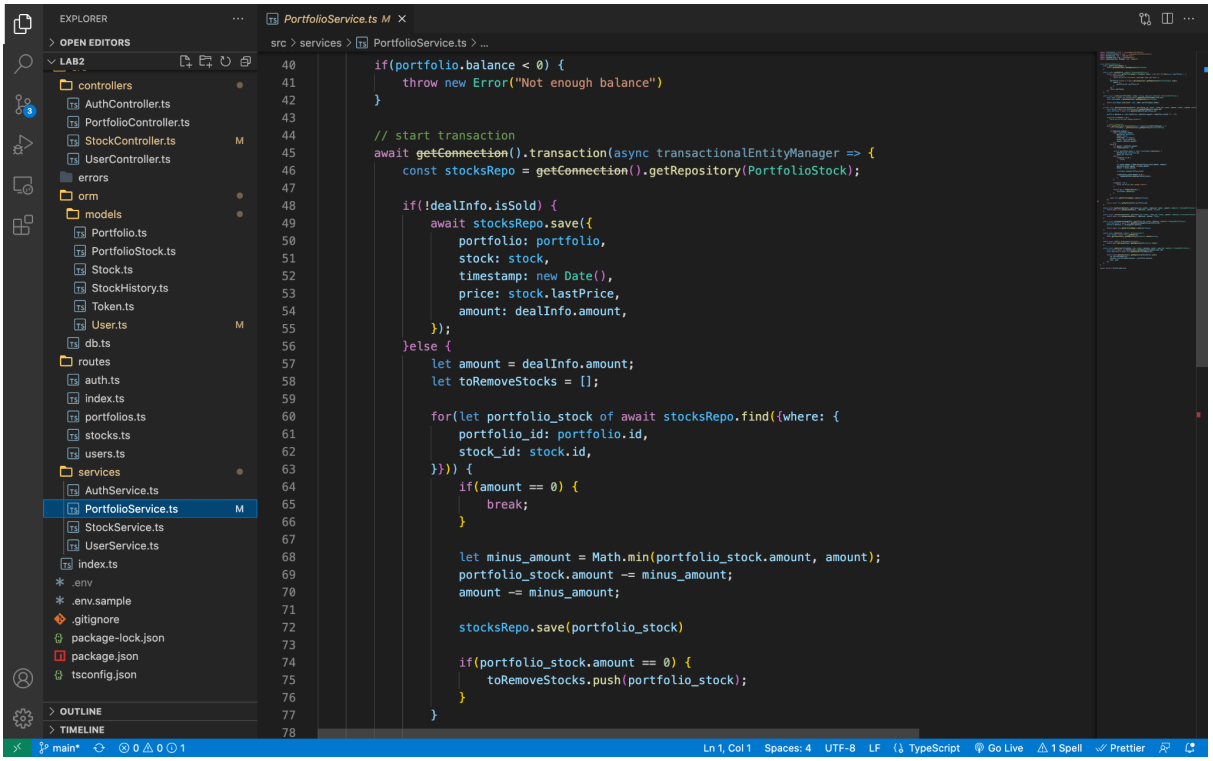
4. Routes

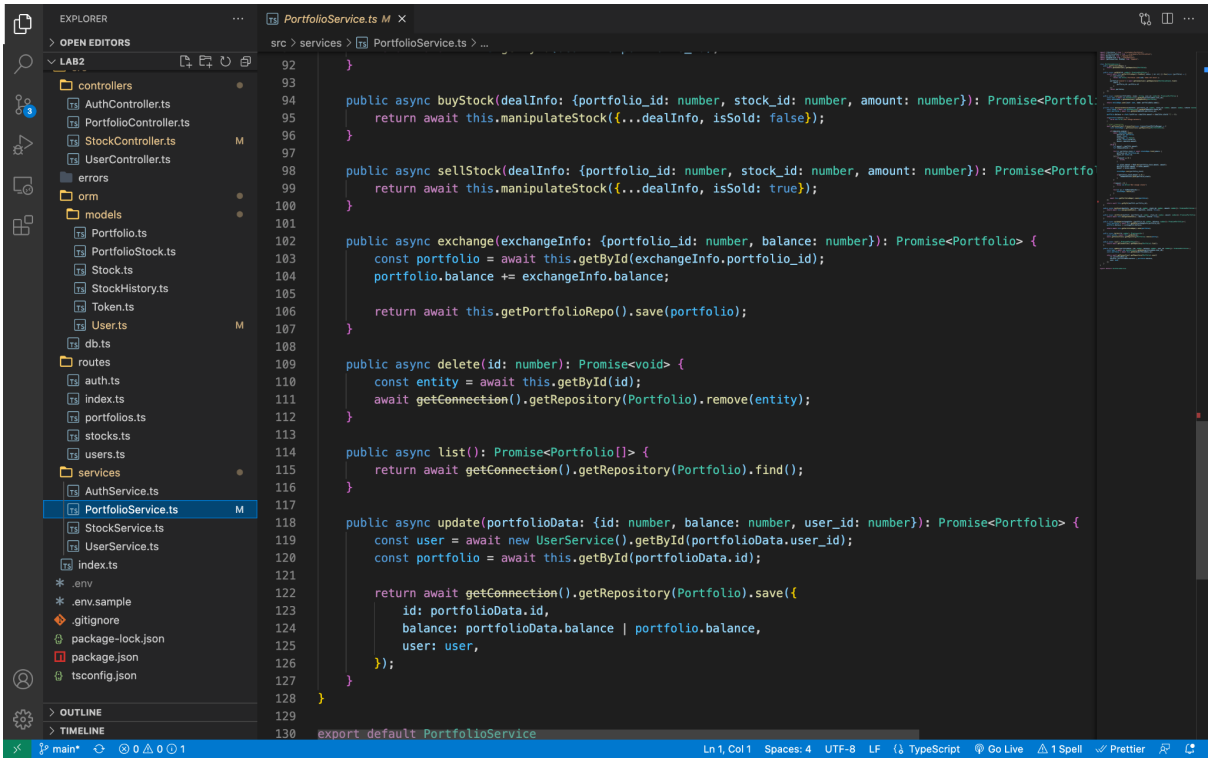


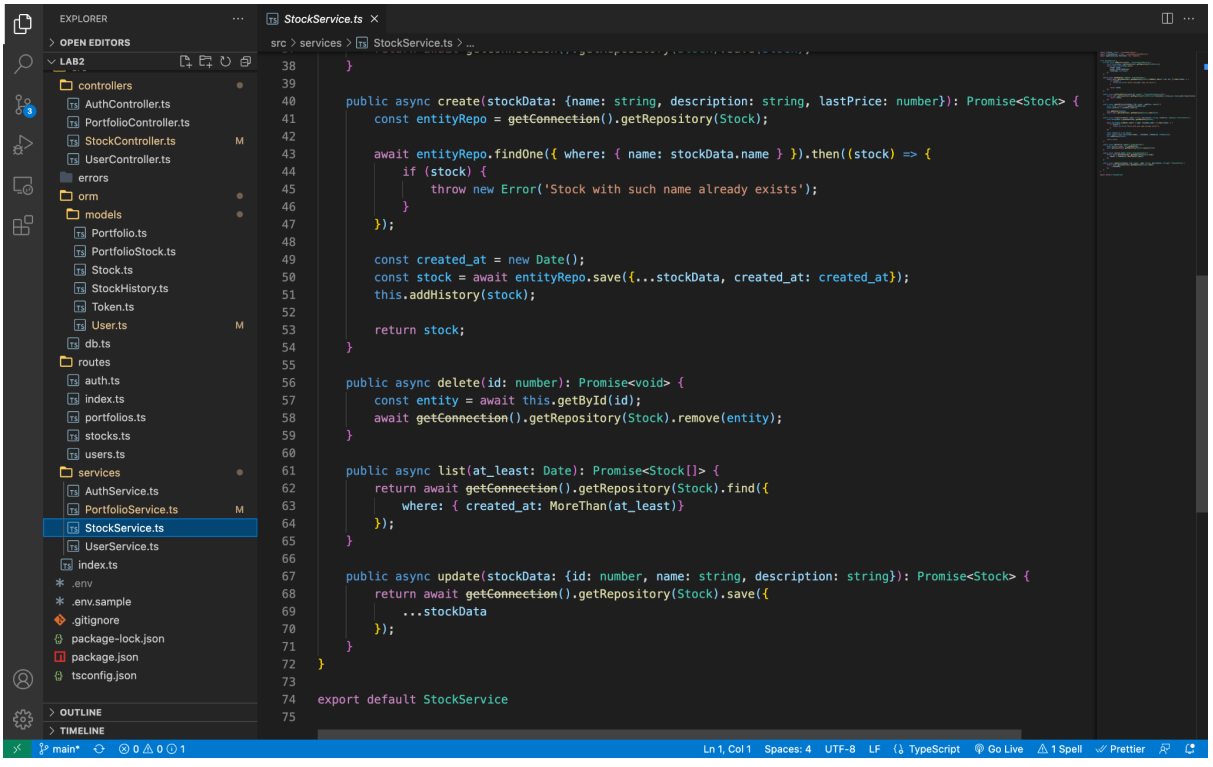
5. Сервисы

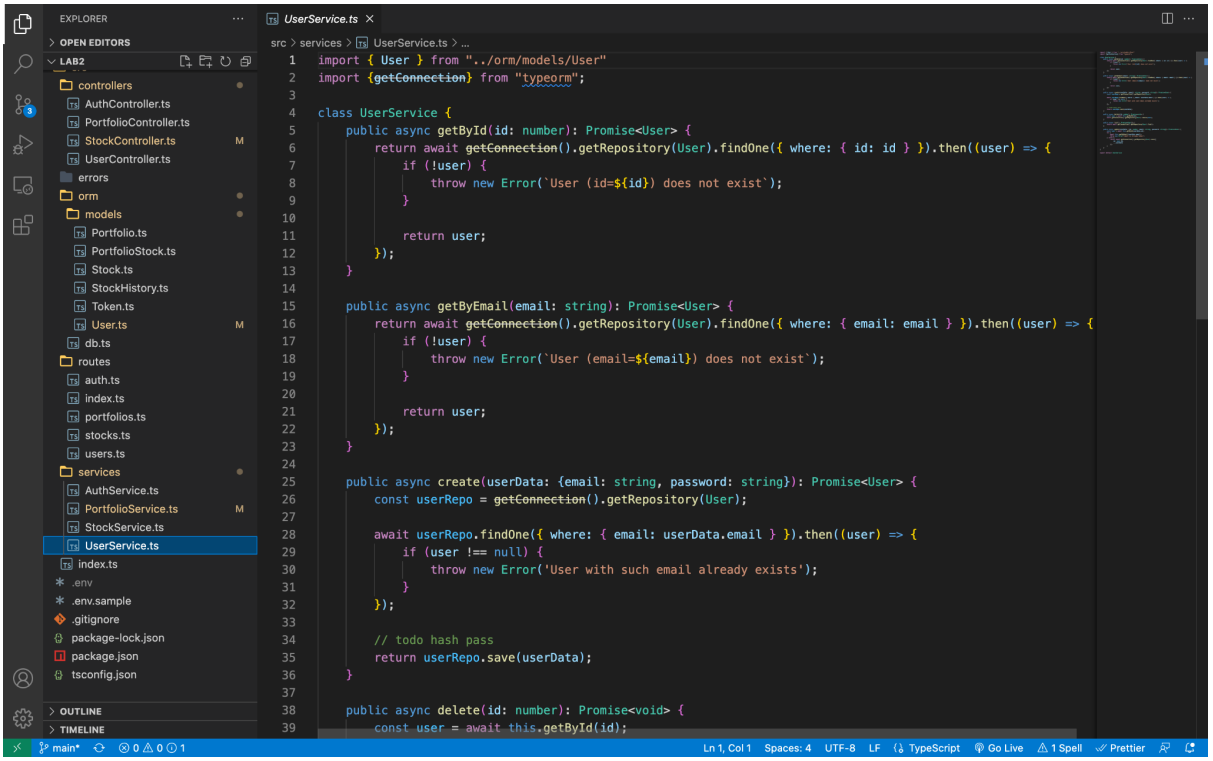

EXPLORER
OPEN EDITORS
LAB2
controllers
AuthController.ts
PortfolioController.ts
StockController.ts M
UserController.ts
errors
orm
models
Portfolio.ts
PortfolioStock.ts
Stock.ts
StockHistory.ts
Token.ts
User.ts M
db.ts
routes
auth.ts
index.ts
portfolios.ts
stocks.ts
users.ts
services
AuthService.ts
PortfolioService.ts M
StockService.ts
UserService.ts
index.ts
.env
.env.sample
.gitignore
package-lock.json
package.json
tsconfig.json
OUTLINE
TIMELINE
AuthService.ts
src > services > AuthService.ts > ...
import userService from "./userService"
const uuid = require("uuid");
class AuthService {
public async auth(userData: {email: string, password: string}) : Promise<AuthToken> {
const user = await new UserService().getByEmail(userData.email);
if(user.password !== userData.password) {
throw new Error("Password is not correct");
}
const tokenRepo = getConnection().getRepository(AuthToken);
return await tokenRepo.findOne({where: {userId: user.id}}).then(async (token) => {
if(token) {
return token;
}
return tokenRepo.save({
user: user,
token: uuid.v4()
});
})
}
public async changeToken(userData: {email: string, password: string}) : Promise<AuthToken> {
const token = await this.auth(userData);
token.token = uuid.v4();
return getConnection().getRepository(AuthToken).save(token);
}
public async delete(userData: {email: string, password: string}): Promise<void> {
const entity = await this.auth(userData);
await getConnection().getRepository(AuthToken).remove(entity);
}
}
export default AuthService
main*
0 0 1
Ln 1, Col 1
Spaces: 4
UTF-8
LF
TypeScript
Go Live
1 Spell
Prettier

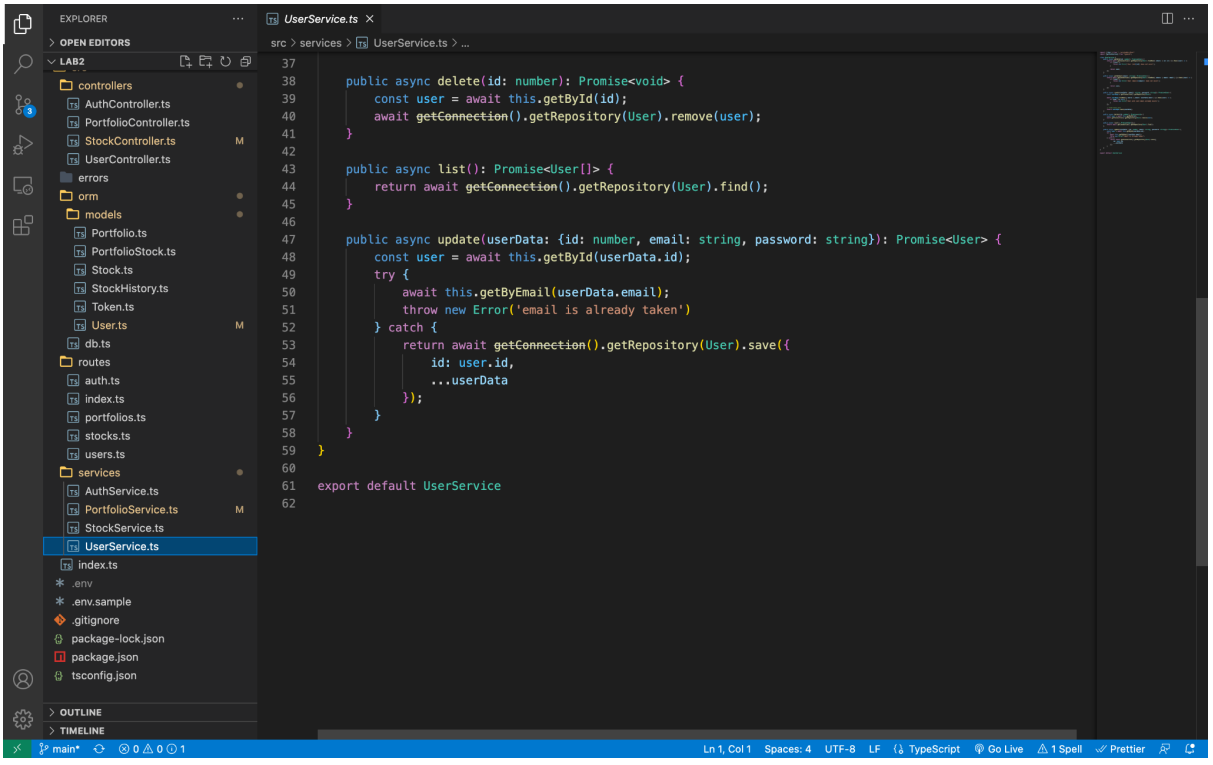

EXPLORER
OPEN EDITORS
LAB2
controllers
AuthController.ts
PortfolioController.ts
StockController.ts M
UserController.ts
errors
orm
models
Portfolio.ts
PortfolioStock.ts
Stock.ts
StockHistory.ts
Token.ts
User.ts M
db.ts
routes
auth.ts
index.ts
portfolios.ts
stocks.ts
users.ts
services
AuthService.ts
PortfolioService.ts M
StockService.ts
UserService.ts
index.ts
.env
.env.sample
.gitignore
package-lock.json
package.json
tsconfig.json
OUTLINE
TIMELINE
PortfolioService.ts M
src > services > PortfolioService.ts > ...
import { Portfolio } from "../orm/models/Portfolio";
import { PortfolioStock } from "../orm/models/PortfolioStock";
import UserService from "./UserService";
import StockService from "./StockService";
import {getConnection, MinKey} from "typeorm";
class PortfolioService {
public getPortfolioRepo() {
return getConnection().getRepository(Portfolio);
}
public async getById(id: number): Promise<Portfolio> {
return await this.getPortfolioRepo().findOne({ where: { id: id } }).then(async (portfolio) => {
if (!portfolio) {
throw new Error(`Portfolio (id=${id}) does not exist`);
}
portfolio["stocks"] = await getConnection().getRepository(PortfolioStock).find({
where: {
portfolio_id: portfolio.id
}
});
return portfolio;
});
}
public async create(portfolioData: {name: string, user_id: number}): Promise<Portfolio> {
const user = await new UserService().getById(portfolioData.user_id);
const entityRepo = getConnection().getRepository(Portfolio);
return entityRepo.save({user: user, name: portfolioData.name});
}
private async manipulateStock(dealInfo: {portfolio_id: number, stock_id: number, amount: number, isSold: b
const stock = await new StockService().getById(dealInfo.stock_id);
const portfolio = await this.getById(dealInfo.portfolio_id);
portfolio.balance += stock.lastPrice * dealInfo.amount * (dealInfo.isSold ? 1 : -1);
main*
0 0 1
Ln 1, Col 1
Spaces: 4
UTF-8
LF
TypeScript
Go Live
1 Spell
Prettier


EXPLORER
OPEN EDITORS
LAB2
controllers
AuthController.ts
PortfolioController.ts
StockController.ts M
UserController.ts
errors
orm
models
Portfolio.ts
PortfolioStock.ts
Stock.ts
StockHistory.ts
Token.ts
User.ts M
db.ts
routes
auth.ts
index.ts
portfolios.ts
stocks.ts
users.ts
services
AuthService.ts
PortfolioService.ts M
StockService.ts
UserService.ts
index.ts
.env
.env.sample
.gitignore
package-lock.json
package.json
tsconfig.json
OUTLINE
TIMELINE
PortfolioService.ts M
src > services > PortfolioService.ts > ...
if(portfolio.balance < 0) {
throw new Error("Not enough balance")
}
// start transaction
await getConnection().transaction(async transactionalEntityManager => {
const stocksRepo = getConnection().getRepository(PortfolioStock);
if(!dealInfo.isSold) {
await stocksRepo.save({
portfolio: portfolio,
stock: stock,
timestamp: new Date(),
price: stock.lastPrice,
amount: dealInfo.amount,
});
}else {
let amount = dealInfo.amount;
let toRemoveStocks = [];
for(let portfolio_stock of await stocksRepo.find({where: {
portfolio_id: portfolio.id,
stock_id: stock.id,
}})) {
if(amount == 0) {
break;
}
let minus_amount = Math.min(portfolio_stock.amount, amount);
portfolio_stock.amount -= minus_amount;
amount -= minus_amount;
stocksRepo.save(portfolio_stock)
if(portfolio_stock.amount == 0) {
toRemoveStocks.push(portfolio_stock);
}
}
main*
0 0 1
Ln 1, Col 1
Spaces: 4
UTF-8
LF
TypeScript
Go Live
1 Spell
Prettier

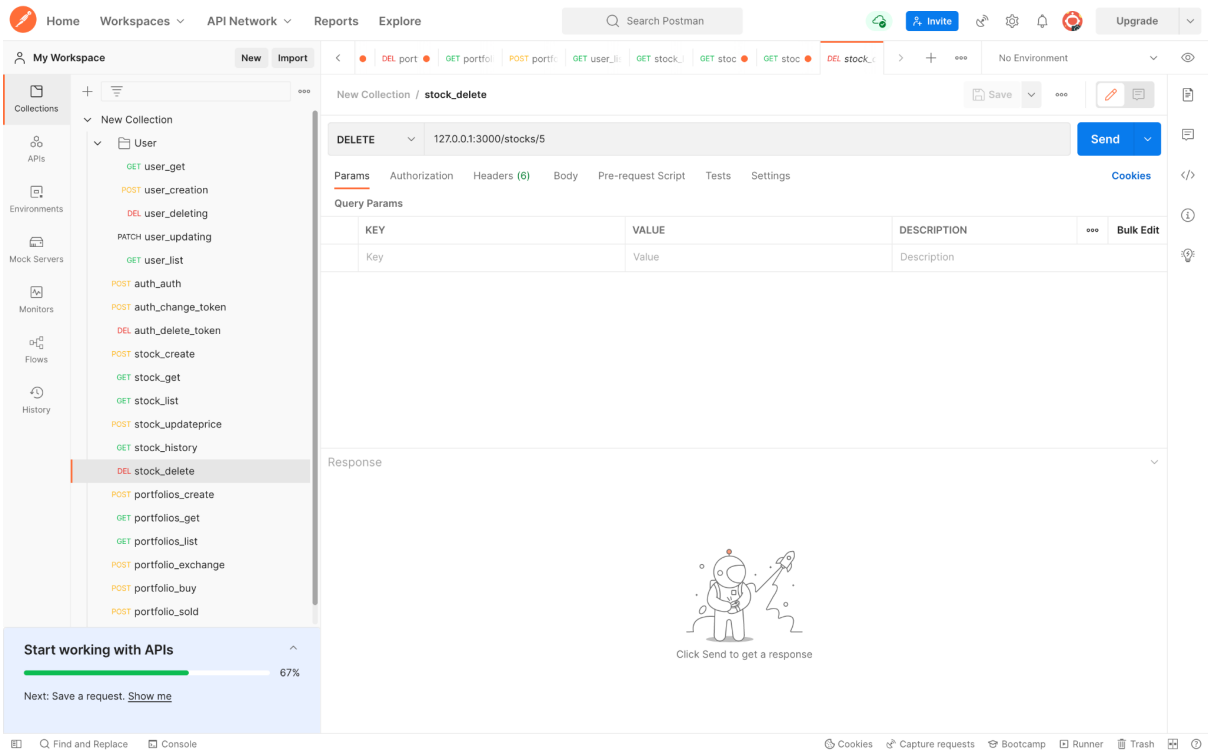

EXPLORER
OPEN EDITORS
LAB2
controllers
AuthController.ts
PortfolioController.ts
StockController.ts M
UserController.ts
errors
orm
models
Portfolio.ts
PortfolioStock.ts
Stock.ts
StockHistory.ts
Token.ts
User.ts M
db.ts
routes
auth.ts
index.ts
portfolios.ts
stocks.ts
users.ts
services
AuthService.ts
PortfolioService.ts M
StockService.ts
UserService.ts
index.ts
.env
.env.sample
.gitignore
package-lock.json
package.json
tsconfig.json
OUTLINE
TIMELINE
PortfolioService.ts M
src > services > PortfolioService.ts > ...
}
public async buyStock(dealInfo: {portfolio_id: number, stock_id: number, amount: number}): Promise<Portfol
return await this.manipulateStock({...dealInfo, isSold: false});
}
public async sellStock(dealInfo: {portfolio_id: number, stock_id: number, amount: number}): Promise<Portfo
return await this.manipulateStock({...dealInfo, isSold: true});
}
public async exchange(exchangeInfo: {portfolio_id: number, balance: number}): Promise<Portfolio> {
const portfolio = await this.getById(exchangeInfo.portfolio_id);
portfolio.balance += exchangeInfo.balance;
return await this.getPortfolioRepo().save(portfolio);
}
public async delete(id: number): Promise<void> {
const entity = await this.getById(id);
await getConnection().getRepository(Portfolio).remove(entity);
}
public async list(): Promise<Portfolio[]> {
return await getConnection().getRepository(Portfolio).find();
}
public async update(portfolioData: {id: number, balance: number, user_id: number}): Promise<Portfolio> {
const user = await new UserService().getById(portfolioData.user_id);
const portfolio = await this.getById(portfolioData.id);
return await getConnection().getRepository(Portfolio).save({
id: portfolioData.id,
balance: portfolioData.balance | portfolio.balance,
user: user,
});
}
}
export default PortfolioService
main*
0 0 1
Ln 1, Col 1
Spaces: 4
UTF-8
LF
TypeScript
Go Live
1 Spell
Prettier


EXPLORER
OPEN EDITORS
LAB2
controllers
AuthController.ts
PortfolioController.ts
StockController.ts
UserController.ts
errors
orm
models
Portfolio.ts
PortfolioStock.ts
Stock.ts
StockHistory.ts
Token.ts
User.ts
db.ts
routes
auth.ts
index.ts
portfolios.ts
stocks.ts
users.ts
services
AuthService.ts
PortfolioService.ts
StockService.ts
UserService.ts
index.ts
.env
.env.sample
.gitignore
package-lock.json
package.json
tsconfig.json
OUTLINE
TIMELINE
StockService.ts
src > services > StockService.ts > ...
38 }
39
40 public async create(stockData: {name: string, description: string, lastPrice: number}): Promise<Stock> {
41 const entityRepo = getConnection().getRepository(Stock);
42
43 await entityRepo.findOne({ where: { name: stockData.name } }).then((stock) => {
44 if (stock) {
45 throw new Error('Stock with such name already exists');
46 }
47 });
48
49 const created_at = new Date();
50 const stock = await entityRepo.save({...stockData, created_at: created_at});
51 this.addHistory(stock);
52
53 return stock;
54 }
55
56 public async delete(id: number): Promise<void> {
57 const entity = await this.getById(id);
58 await getConnection().getRepository(Stock).remove(entity);
59 }
60
61 public async list(at_least: Date): Promise<Stock[]> {
62 return await getConnection().getRepository(Stock).find({
63 where: { created_at: MoreThan(at_least)}
64 });
65 }
66
67 public async update(stockData: {id: number, name: string, description: string}): Promise<Stock> {
68 return await getConnection().getRepository(Stock).save({
69 ...stockData
70 });
71 }
72 }
73
74 export default StockService
75
main*
Ln 1, Col 1
Spaces: 4
UTF-8
LF
TypeScript
Go Live
1 Spell
Prettier





6. Тестирование с помощью Postman



Вывод

В ходе данной лабораторной работы мы реализовали RESTful API приложение сайта криптобиржи.