|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  высшего образования |
| **«Национальный исследовательский ядерный университет «МИФИ»**  **(НИЯУ МИФИ)** |

**Отчет**

**по результатам выполнения задания**

**демонстрационного экзамена**

|  |  |
| --- | --- |
| Студент | Тарасов Артём Александрович |
| Организация | НИЯУ МИФИ (Москва) |
| Группа | Б22-525 |
|  |  |
| Дата | 25.09.2024 |

Москва 2024

**Оглавление**

[**I Итоговый ассесмент** 3](#_Toc171668961)

[**II Инструкция по запуску и ручной проверке** 3](#_Toc171668962)

[**III Алгоритм обработки** **Error! Bookmark not defined.**](#_Toc171668963)

[***Заключение*** 8](#_Toc171668964)

# **I Итоговый ассесмент**

# 

# **II Приложение на основе Spring Boot**

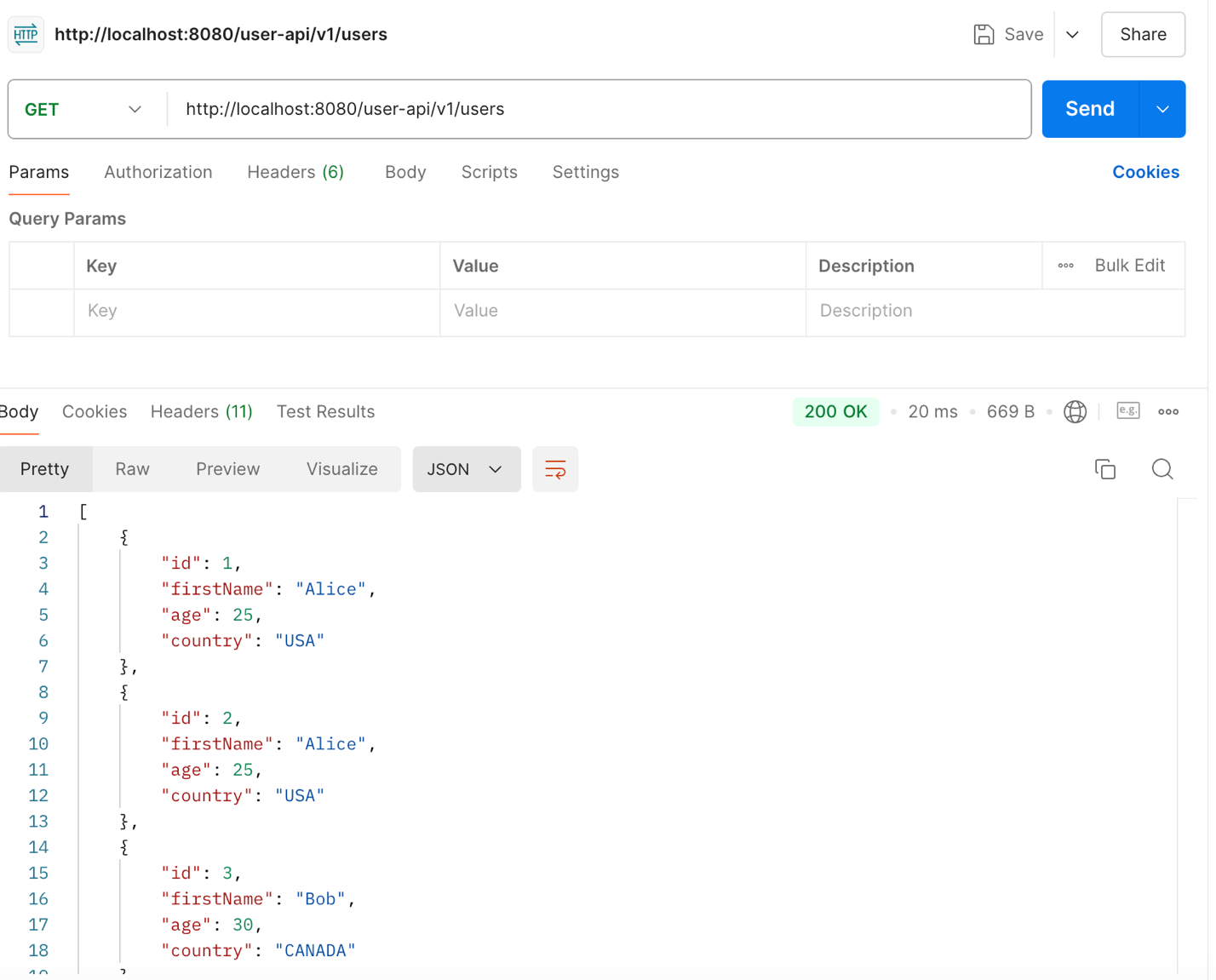
*Всю реализацию, с пояснениями можно посмотреть в моём репозитории:*

<https://github.com/tarasovxx/digital_mephi_java>

На страницах далее можно посмотреть, как работают эндпойнты при локальном тестировании через Postman/браузер.

Весь необходимый функционал был реализован, а также к приложению были написаны тесты, которые можно посмотреть также в репозитории.

Приложение было написано по паттернам: репозиторий, сервис, контроллер. Использует Spring, а также in-memory H2-database. Вся сборка производится через **gradle**

****

GET <http://localhost:8080/user-api/v1/users>  
response:

[

{

"id": 1,

"firstName": "Alice",

"age": 25,

"country": "USA"

},

{

"id": 2,

"firstName": "Alice",

"age": 25,

"country": "USA"

},

{

"id": 3,

"firstName": "Bob",

"age": 30,

"country": "CANADA"

},

{

"id": 4,

"firstName": "Charlie",

"age": 22,

"country": "FRANCE"

},

{

"id": 5,

"firstName": "David",

"age": 27,

"country": "GERMANY"

},

{

"id": 6,

"firstName": "Eve",

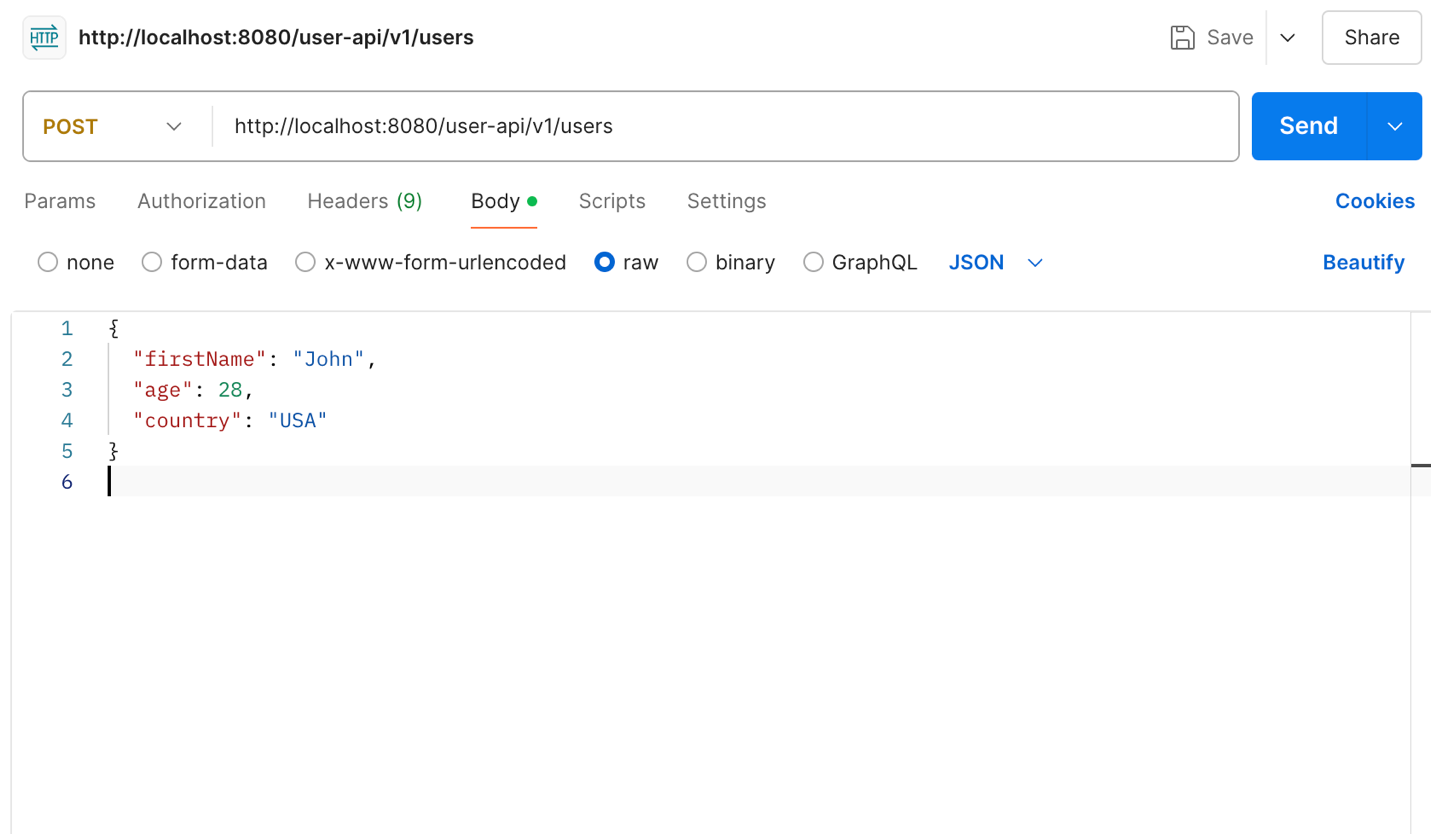
"age": 35,

"country": "JAPAN"

}

]

**Добавление пользователя в базу данных:**

****

POST <http://localhost:8080/user-api/v1/users> with body  
{

"firstName": "John",

"age": 28,

"country": "USA"

}

curl -X POST http://localhost:8080/user-api/v1/users \

-H 'Content-Type: application/json' \

-d '{

"firstName": "John",

"age": 28,

"country": "USA"

}'  
response:

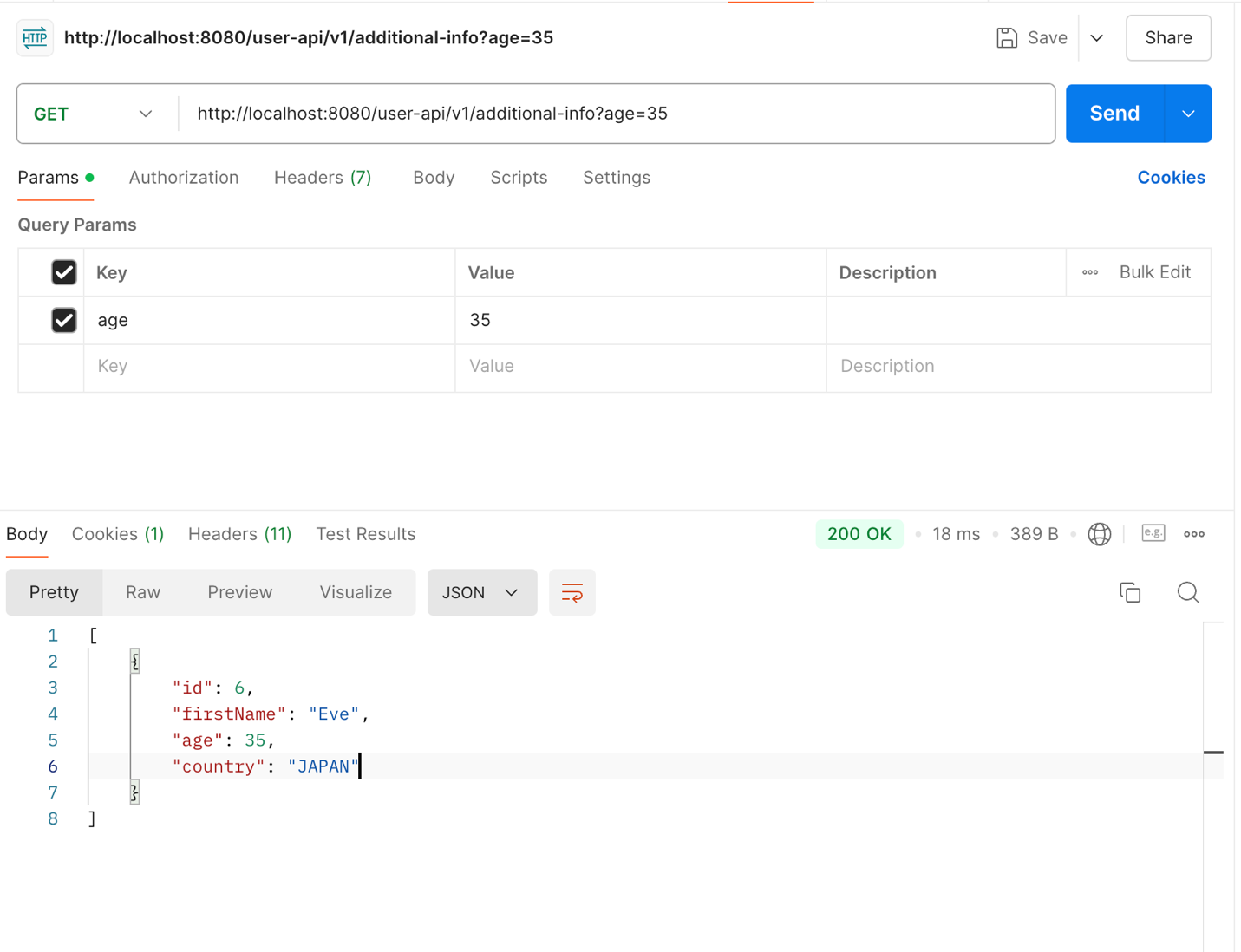
{

"status": "Ok"

}

**Получить пользователей старше определенного возраста** (GET запрос с параметром):

Вернется список пользователей, возраст которых больше или равен 25, отсортированный по имени.

****

**GET** <http://localhost:8080/user-api/v1/additional-info?age=27>

Response:

[

{

"id": 3,

"firstName": "Bob",

"age": 30,

"country": "CANADA"

},

{

"id": 5,

"firstName": "David",

"age": 27,

"country": "GERMANY"

},

{

"id": 6,

"firstName": "Eve",

"age": 35,

"country": "JAPAN"

}

]

**GET** <http://localhost:8080/user-api/v1/additional-info?age=35>

Response

[

{

"id": 6,

"firstName": "Eve",

"age": 35,

"country": "JAPAN"

}

]

# ***Заключение***

*В данной работе было создано небольшое Java web-приложение на основе Spring Boot, реализующее функционал управления пользователями. Приложение поддерживает следующие операции:*

1. Получение списка всех пользователей через эндпоинт "user-api/v1/users" (GET метод).
2. Добавление нового пользователя через эндпоинт "user-api/v1/users" (POST метод).
3. Получение списка пользователей, возраст которых больше либо равен заданному, через эндпоинт "user-api/v1/additional-info" (GET метод).

***Общие выводы***

* Приложение успешно реализовано с использованием паттернов "репозиторий", "сервис" и "контроллер", что обеспечивает хорошую структурированность и поддержку кода.
* Для хранения данных используется in-memory H2 база данных, что упрощает локальное тестирование и разработку.
* Вся сборка и управление зависимостями производится через Gradle, что обеспечивает удобство и гибкость в управлении проектом.
* Реализованы автоматические тесты, которые проверяют корректность работы основных функциональных возможностей приложения. Тесты можно найти в репозитории проекта.
* Приложение успешно проходит тестирование через Postman и браузер, что подтверждает его работоспособность и соответствие поставленным требованиям.

***Реализация***

*Всю реализацию, включая пояснения, можно посмотреть в моём репозитории:*

[*https://github.com/tarasovxx/digital\_mephi\_java*](https://github.com/tarasovxx/digital_mephi_java)

***Основные достижения***

* Приложение было написано с использованием Spring Boot и модулей Spring WEB и Spring Data JPA. Вся функциональность, описанная в техническом задании, была успешно реализована.
* Приложение автоматически создает таблицу users в базе данных и заполняет её минимум пятью записями при запуске.
* Реализованные тесты проверяют основные функциональные возможности приложения, что подтверждает его корректную работу.

***Технические детали***

* Пользователь описывается сущностью User с полями Long id, String firstName, Integer age, Country country. Country — это enum с названиями стран.
* Данные всех пользователей хранятся в таблице users в базе данных.
* Добавление новых пользователей также происходит в таблицу users.
* Для взаимодействия с базой данных используется in-memory H2 database.
* Все операции с пользователями реализованы через JPA репозиторий без написания SQL-запросов.

***Заключение***

Весь необходимый функционал был реализован, а также к приложению были написаны тесты, которые можно посмотреть в репозитории. Приложение соответствует всем обязательным требованиям и готово к использованию для локального тестирования и дальнейшего развития.