**Аутентификационный REST-контроллер**

**Описание**

Этот REST-контроллер предоставляет API-эндпоинты для аутентификации и регистрации пользователей.

**Пути**

**1. Регистрация пользователя**

URL: /api/v2/auth/registration

Метод: POST

Описание: Регистрирует нового пользователя с указанными данными.

Параметры запроса:

* email (тип: строка, обязательный) - адрес электронной почты пользователя.
* password (тип: строка, обязательный) - пароль пользователя.
* first\_name (тип: строка, обязательный) - имя пользователя.
* last\_name (тип: строка, обязательный) - фамилия пользователя.

Пример запроса:

POST /api/v2/auth/registration Content-Type: application/json { "email": "example@example.com", "password": "password123", "first\_name": "John", "last\_name": "Doe" }

Пример успешного ответа:

HTTP/1.1 200 OK Content-Type: application/json {

"email": "user@mail.com",

"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyQG1haWwuY29tIiwicm9sZSI6IkFETUlOIiwiaWF0IjoxNjg1MzEyNDE4LCJleHAiOjE2ODU5MTcyMTh9.FxQmo9ObXonMNLuTVPDBJJt501HGtUZG-EPPYrI6r0Y"

}

**2. Аутентификация пользователя**

URL: /api/v2/auth/login

Метод: POST

Описание: Аутентифицирует пользователя с помощью предоставленных учетных данных.

Параметры запроса:

* email (тип: строка, обязательный) - адрес электронной почты пользователя.
* password (тип: строка, обязательный) - пароль пользователя.

Пример запроса:

POST /api/v2/auth/login Content-Type: application/json { "email": "example@example.com", "password": "password123" }

Пример успешного ответа:

HTTP/1.1 200 OK Content-Type: application/json { "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c", "expires\_in": 3600 }

**3. Выход пользователя**

URL: /api/v2/auth/logout

Метод: POST

Описание: Выполняет выход текущего пользователя.

Пример запроса:

POST /api/v2/auth/logout

Пример успешного ответа:

HTTP/1.1 204 No Content

**Зависимости**

Для работы этого контроллера необходимо наличие следующих зависимостей:

* spring-boot-starter-web - предоставляет функциональность для разработки веб-приложений с использованием Spring MVC.
* spring-boot-starter-security - обеспечивает безопасность приложения и аутентификацию пользователей.
* spring-boot-starter-data-jpa - предоставляет функциональность для работы с базой данных через Spring Data JPA.
* spring-boot-starter-validation - предоставляет поддержку валидации данных запросов.

# 

# **Контроллер валюты**

## **Описание**

Этот контроллер предоставляет API-эндпоинты для работы с данными о валюте.

## **Пути**

### **1. Получение истории валют**

URL: /api/v2/currency/gethistory

Метод: GET

Описание: Возвращает список всех сохраненных записей о валюте.

Требования доступа: Требуется наличие прав доступа "developers:read".

Пример запроса:

GET /api/v2/currency/gethistory

Пример успешного ответа:

HTTP/1.1 200 OK Content-Type: application/json [ { "usdSale": "1.23", "usdBuy": "1.21", "eurSale": "1.42", "eurBuy": "1.39" }, { "usdSale": "1.25", "usdBuy": "1.23", "eurSale": "1.40", "eurBuy": "1.37" }, ... ]

### **2. Добавление новой записи о валюте**

URL: /api/v2/currency/addnew

Метод: POST

Описание: Сохраняет новую запись о валюте с указанными данными.

Требования доступа: Требуется наличие прав доступа "developers:write".

Параметры запроса:

· usdSale (тип: строка, обязательный) - курс продажи USD.

· usdBuy (тип: строка, обязательный) - курс покупки USD.

· eurSale (тип: строка, обязательный) - курс продажи EUR.

· eurBuy (тип: строка, обязательный) - курс покупки EUR.

Пример запроса:

POST /api/v2/currency/addnew Content-Type: application/json { "usdSale": "1.23", "usdBuy": "1.21", "eurSale": "1.42", "eurBuy": "1.39" }

Пример успешного ответа:

HTTP/1.1 200 OK Content-Type: application/json { "usdSale": "1.23", "usdBuy": "1.21", "eurSale": "1.42", "eurBuy": "1.39" }

### **3. Получение последней записи о валюте**

URL: /api/v2/currency/getlast

Метод: GET

Описание: Возвращает последнюю сохраненную запись о валюте.

Требования доступа: Требуется наличие прав доступа "developers:write".

Пример запроса:

GET /api/v2/currency/getlast

Пример успешного ответа:

HTTP/1.1 200 OK Content-Type: application/json { "usdSale": "1.23", "usdBuy": "1.21", "eurSale": "1.42", "eurBuy": "1.39" }

## **Автоматическое обновление данных**

Данный контроллер содержит метод saveNewDateFromApi(), который запускается автоматически по расписанию. Он вызывает метод fetchAndSaveCurrencyData() сервиса валюты для получения новых данных о валюте с внешнего API и сохранения их в базе данных. Расписание выполнения метода задается с помощью аннотации @Scheduled.

## **Авторизация**

Для доступа к определенным эндпоинтам требуется наличие определенных прав доступа, указанных в аннотации @PreAuthorize. Например, эндпоинты /gethistory и /getlast требуют прав доступа "developers:read", а эндпоинт /addnew требует прав доступа "developers:write".

# **Контроллер новостей**

## **Описание**

Этот контроллер предоставляет API-эндпоинты для работы с новостями.

## **Пути**

### **1. Получение новости по идентификатору**

URL: /api/v2/news/id

Метод: GET

Описание: Возвращает новость по указанному идентификатору.

Параметры запроса:

· id (тип: Long, обязательный) - идентификатор новости.

· Authorization (тип: строка, обязательный) - заголовок авторизации с токеном пользователя.

Пример запроса:

GET /api/v2/news/id?id=123 Authorization: Bearer <token>

Пример успешного ответа:

HTTP/1.1 200 OK Content-Type: application/json { "id": 123, "title": "Заголовок новости", "content": "Содержание новости", ... }

### **2. Получение списка новостей по параметру**

URL: /api/v2/news/newsbyparam

Метод: GET

Описание: Возвращает список новостей, отфильтрованных по указанным параметрам.

Параметры запроса:

· category (тип: строка, необязательный) - категория новости.

· author (тип: строка, необязательный) - автор новости.

· country (тип: строка, необязательный) - страна новости.

Пример запроса:

GET /api/v2/news/newsbyparam?category=politics&author=john&country=USA

Пример успешного ответа:

HTTP/1.1 200 OK Content-Type: application/json [ { "id": 123, "title": "Заголовок новости 1", "content": "Содержание новости 1", ... }, { "id": 456, "title": "Заголовок новости 2", "content": "Содержание новости 2", ... }, ... ]

### **3. Обновление статуса новости для пользователя**

URL: /api/v2/news/edit/{articleId}

Метод: PUT

Описание: Обновляет статус новости для пользователя (например, помечает новость как прочитанную или понравившуюся).

Параметры запроса:

· articleId (тип: Long, обязательный) - идентификатор новости.

· Authorization (тип: строка, обязательный) - заголовок авторизации с токеном пользователя.

· status (тип: NewsStatus, необязательный) - новый статус новости.

· liked (тип: Boolean, необязательный) - флаг, указывающий, понравилась ли новость пользователю.

Пример запроса:

PUT /api/v2/news/edit/123 Authorization: Bearer <token> Content-Type: application/json { "status": "READ" }

Пример успешного ответа:

HTTP/1.1 200 OK Content-Type: text/plain UserNewsStatus updated successfully.

### **4. Получение последних новостей по категории**

URL: /api/v2/news/set/{category}

Метод: GET

Описание: Возвращает последние новости по указанной категории.

Параметры запроса:

· category (тип: строка, обязательный) - категория новости.

· country (тип: строка, необязательный) - страна новости.

· countries = "gb", "ua", "us";

· categories = "business", "entertainment", "science", "sports", "technology";

Пример запроса:

GET /api/v2/news/set/politics?country=UA

Пример успешного ответа:

HTTP/1.1 200 OK Content-Type: application/json [ { "id": 123, "title": "Заголовок новости 1", "content": "Содержание новости 1", ... }, { "id": 456, "title": "Заголовок новости 2", "content": "Содержание новости 2", ... }, ... ]

### **5. Получение новостей по категории с пагинацией**

URL: /api/v2/news/set/{category}/{offset}/{count}

Метод: GET

Описание: Возвращает новости по указанной категории с пагинацией.

Параметры запроса:

· category (тип: строка, обязательный) - категория новости.

· offset (тип: целое число, обязательный) - смещение (начиная с 0) для пагинации.

· count (тип: целое число, обязательный) - количество новостей для возврата.

· country (тип: строка, необязательный) - страна новости.

Пример запроса:

GET /api/v2/news/set/politics/0/10?country=UA

Пример успешного ответа:

HTTP/1.1 200 OK Content-Type: application/json { "content": [ { "id": 123, "title": "Заголовок новости 1", "content": "Содержание новости 1", ... }, { "id": 456, "title": "Заголовок новости 2", "content": "Содержание новости 2", ... }, ... ], "totalElements": 100, "totalPages": 10, ... }

### **6. Получение сохраненных новостей пользователя**

URL: /api/v2/news/saved

Метод: GET

Описание: Возвращает список сохраненных новостей для пользователя.

Параметры запроса:

· Authorization (тип: строка, обязательный) - заголовок авторизации с токеном пользователя.

Пример запроса:

GET /api/v2/news/saved Authorization: Bearer <token>

Пример успешного ответа:

HTTP/1.1 200 OK Content-Type: application/json [ { "id": 123, "title": "Заголовок новости 1", "content": "Содержание новости 1", ... }, { "id": 456, "title": "Заголовок новости 2", "content": "Содержание новости 2", ... }, ... ]

### **7. Получение понравившихся новостей пользователя**

URL: /api/v2/news/liked

Метод: GET

Описание: Возвращает список понравившихся новостей для пользователя.

Параметры запроса:

· Authorization (тип: строка, обязательный) - заголовок авторизации с токеном пользователя.

Пример запроса:

GET /api/v2/news/liked Authorization: Bearer <token>

Пример успешного ответа:

HTTP/1.1 200 OK Content-Type: application/json [ { "id": 123, "title": "Заголовок новости 1", "content": "Содержание новости 1", ... }, { "id": 456, "title": "Заголовок новости 2", "content": "Содержание новости 2", ... }, ... ]

### **8. Получение сохраненных и понравившихся новостей пользователя**

URL: /api/v2/news/saved-liked

Метод: GET

Описание: Возвращает список сохраненных и понравившихся новостей для пользователя.

Параметры запроса:

· Authorization (тип: строка, обязательный) - заголовок авторизации с токеном пользователя.

Пример запроса:

GET /api/v2/news/saved-liked Authorization: Bearer <token>

Пример успешного ответа:

HTTP/1.1 200 OK Content-Type: application/json [ { "id": 123, "title": "Заголовок новости 1", "content": "Содержание новости 1", ... }, { "id": 456, "title": "Заголовок новости 2", "content": "Содержание новости 2", ... }, ... ]

### **9. Получение статусов новостей пользователя**

URL: /api/v2/news/{id}

Метод: GET

Описание: Возвращает список статусов новостей для пользователя по указанному идентификатору.

Параметры запроса:

id (тип: Long, обязательный) - идентификатор новости.

Пример запроса:

GET /api/v2/news/123

Пример успешного ответа:

HTTP/1.1 200 OK Content-Type: application/json [ { "id": 123, "status": "READ", "liked": true, ... }, ... ]

### 

### **Документация API для LikeController**

Этот

контроллер обрабатывает операции, связанные с предпочтениями пользователя,

такими как отмеченные как понравившиеся и сохраненные статьи.

**Получение количества**

#### **сохраненных статей пользователя**

·

URL: /api/v2/prefer/savedlengs

·

Метод: GET

·

Описание: Возвращает количество статей,

сохраненных пользователем.

·

Заголовок запроса:

·

Authorization: Токен аутентификации пользователя.

·

Ответ:

·

Код состояния: 200 OK

·

Тип содержимого: application/json

·

Тело ответа: Целое число, представляющее

количество сохраненных статей.

**Получение количества статей,**

#### **которые понравились пользователю**

·

URL: /api/v2/prefer/likedlengs

·

Метод: GET

·

Описание: Возвращает количество статей,

которые понравились пользователю.

·

Заголовок запроса:

·

Authorization: Токен аутентификации пользователя.

·

Ответ:

·

Код состояния: 200 OK

·

Тип содержимого: application/json

·

Тело ответа: Целое число, представляющее

количество статей, которые понравились пользователю.

**Получение списка статей,**

#### **которые понравились пользователю**

·

URL:

/api/v2/prefer/liked-articles

·

Метод: GET

·

Описание: Возвращает список статей,

которые понравились пользователю.

·

Заголовок запроса:

·

Authorization: Токен аутентификации пользователя.

·

Ответ:

·

Код состояния: 200 OK

·

Тип содержимого: application/json

·

Тело ответа: Список статей,

соответствующих предпочтениям пользователя.

**Получение списка сохраненных**

#### **статей пользователя**

·

URL:

/api/v2/prefer/saved-articles

·

Метод: GET

·

Описание: Возвращает список статей,

сохраненных пользователем.

·

Заголовок запроса:

·

Authorization: Токен аутентификации пользователя.

·

Ответ:

·

Код состояния: 200 OK

·

Тип содержимого: application/json

·

Тело ответа: Список сохраненных статей,

принадлежащих пользователю.

### 

### **Документация API для NoteController**

Этот

контроллер обрабатывает операции, связанные с заметками пользователя.

#### **Создание новой задачи**

·

URL: /api/v2/usernotes/createtask

·

Метод: POST

·

Описание: Создает новую задачу

пользователя.

·

Параметры запроса:

·

subject: Тема задачи (строка)

·

description: Описание задачи (строка)

·

Заголовок запроса:

·

Authorization: Токен аутентификации пользователя.

·

Ответ:

·

Код состояния: 200 OK

·

Тип содержимого: application/json

·

Тело ответа: Модель UserNotesModel, представляющая созданную задачу

пользователя.

**Получение списка заметок**

#### **пользователя**

·

URL: /api/v2/usernotes/getauthusernotes

·

Метод: GET

·

Описание: Возвращает список заметок

пользователя.

·

Заголовок запроса:

·

Authorization: Токен аутентификации пользователя.

·

Ответ:

·

Код состояния: 200 OK

·

Тип содержимого: application/json

·

Тело ответа: Список моделей UserNotesModel, представляющих заметки пользователя.

**Получение ключа шифрования**

#### **задачи**

·

URL: /api/v2/usernotes/getKey

·

Метод: GET

·

Описание: Возвращает ключ шифрования

задачи по ее идентификатору.

·

Параметры запроса:

·

id: Идентификатор задачи (число)

·

Ответ:

·

Код состояния: 200 OK

·

Тип содержимого: application/json

·

Тело ответа: Ключ шифрования задачи

(строка).

**Получение задачи по ключу**

#### **шифрования**

·

URL: /api/v2/usernotes/getTaskByKey

·

Метод: GET

·

Описание: Возвращает задачу по ключу

шифрования.

·

Параметры запроса:

·

key: Ключ шифрования задачи (строка)

·

Заголовок запроса:

·

Authorization: Токен аутентификации пользователя.

·

Ответ:

·

Код состояния: 200 OK

·

Тип содержимого: application/json

·

Тело ответа: Модель UserNotesModel, представляющая задачу.

#### **Обновление задачи**

·

URL: /api/v2/usernotes/updatetask

·

Метод: PUT

·

Описание: Обновляет задачу пользователя.

·

Параметры запроса (выберите один из

вариантов для идентификации задачи):

·

key: Ключ шифрования задачи (строка)

·

id: Идентификатор задачи (число)

·

Параметры запроса для обновления полей

задачи (выберите нужные параметры):

·

subject: Новая тема задачи (строка)

·

description: Новое описание задачи (строка)

·

noteStatus: Новый статус задачи (строка, одно из

значений: TODO, IN\_PROGRESS, DONE)

·

availability: Новая доступность задачи (строка, одно

из значений: PRIVATE, PUBLIC)

·

Заголовок запроса:

·

Authorization: Токен аутентификации пользователя.

·

Ответ:

·

Код состояния: 200 OK

·

Тип содержимого: application/json

·

Тело ответа: Модель UserNotesModel, представляющая обновленную задачу.

·