

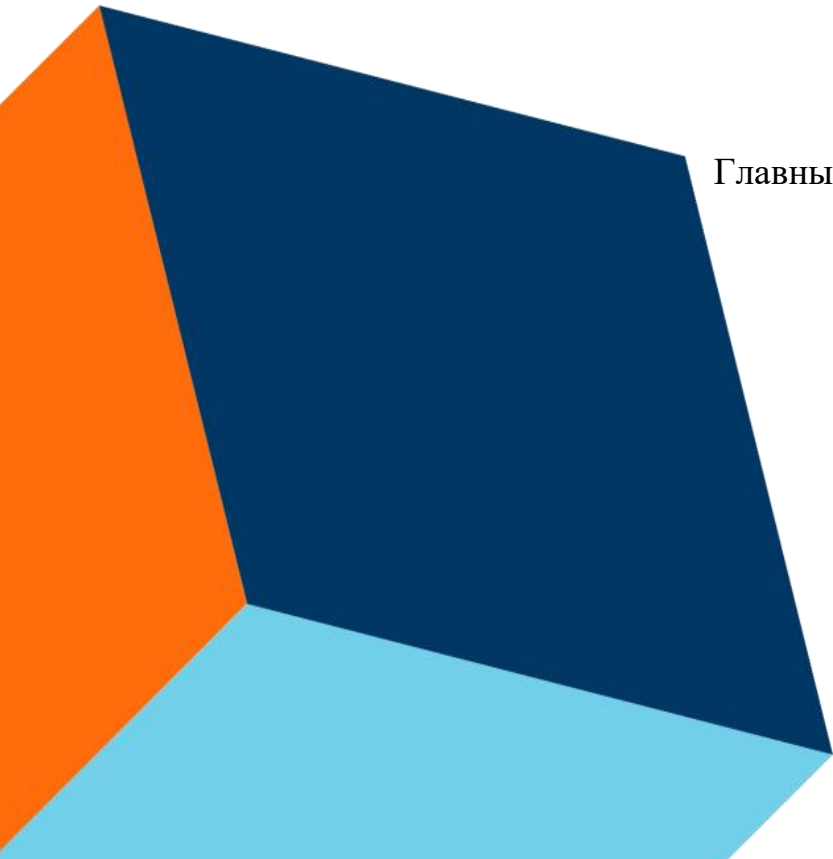


Конкурсное задание

городского конкурса профессионального
мастерства WorldSkills Astana 2023

по компетенции Веб технологии

Module A: Back-end



Главный эксперт

Попов Денис

Время выполнения: 4 часа

Содержание

ВВЕДЕНИЕ	3
ОПИСАНИЕ ПРОЕКТА И ЗАДАЧ	3
МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ	4
ПОРТАЛ АДМИНИСТРАТОРА	5
REST API	7
ИНСТРУКЦИИ ДЛЯ УЧАСТНИКА	22

ВВЕДЕНИЕ

Недавно основанная компания ищет разработчиков полного стека для создания платформы для онлайн игр в браузерах. Разработчики игр могут загружать свои игры на платформу, и пользователи могут играть в них онлайн в браузере.

На платформе есть три части:

- Портал разработчиков игр: веб-приложение для разработчиков игр с возможностью загрузки своих игр на платформу.
- Портал администратора: веб-приложение для администраторов с возможностью управления на платформе пользователями и играми.
- Игровой портал: веб-приложение позволяющее пользователям играть онлайн в браузере.

Компания хочет создать минимальный жизнеспособный продукт (MVP) для платформы. MVP уже должен содержать вышеупомянутые части, на приемлемом уровне, но портал разработчиков игры и портал администратора еще не доработаны. Игровой портал должен быть полностью функциональным, чтобы пользователи могли играть в игры онлайн в браузере.

Проект разделен на два этапа:

- Первый этап (1 день) создание API и страниц портала администратора с использованием PHP фреймворка и базы данных MySQL
- Второй этап (2 день) разработка SPA приложения с использованием HTML/CSS и фреймворка JavaScript, использующего API, разработанного на первом этапе

Для второго этапа будет предоставлена реализация API из первого этапа, поэтому вы можете сосредоточиться на разработке SPA приложения.

ОПИСАНИЕ ПРОЕКТА И ЗАДАЧ

Этот этап требует, чтобы вы реализовали бэкэнд платформы. Бэкэнд должен быть реализован с использованием фреймворка PHP и базы данных MySQL.

Бэкэнд предоставляет:

- Портал администратора (используя рендеринг на стороне сервера, без JavaScript, кроме фильтрации игр)
- REST API для портала разработчиков и игрового портала

МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ

Платформа имеет следующую модель предметной области:

- Администратор: Может войти в портал администратора и использовать всю его функциональность.
 - username
 - password
 - registered timestamp
 - last login timestamp
- Пользователь платформы: Может зарегистрироваться и войти в качестве игрока и разработчика игр на портал и использовать всю его функциональность.
 - username
 - password
 - registered timestamp
 - last login timestamp
 - game scores (relates to a set of game scores)
 - uploaded games (relates to a set of games)
- Игровые баллы: Каждый раз, когда пользователь играет в игру, создается игровой счет.
 - user (relates to a platform user)
 - game version (relates to a game version)
 - timestamp
 - score (number of points, signed, [-2147483648, 2147483647])
- Игра: Игра в которую можно сыграть на платформе.
 - title
 - description
 - optional thumbnail

- slug (a unique and human-readable name that can be used as part of an URL to identify the game, based on the game title)
 - author (relates to a platform user)
- Версии игр: Версия игры в которую можно сыграть на платформе. Разработчики могут загружать новые версии, но не удалять старые версии.
 - game (relates to a game)
 - version timestamp
 - path to game files

ПОРТАЛ АДМИНИСТРАТОРА

Портал администратора является веб-приложением для администраторов с возможностью управления на платформе пользователями и играми.

Вход на портал должен быть доступен по пути /XX-module-a/admin.

Доступ к порталу можно получить только путем входа в систему, используя имя пользователя и пароль. Имя пользователя и пароль хранятся в базе данных. Пользователи администратора не привязаны к пользователям игрового портала. Это означает, что пользователи администратора могут иметь такое же имя пользователя, что и пользователи игрового портала. Не существует регистрации для администраторов, вместо этого они добавляются через прямой доступ к базе данных (во время создания MVP).

Функциональные возможности портала администратора:

- Список администраторов:
 - Администратор может видеть всех администраторов в базе данных с именем пользователя, временной меткой создания и временной меткой последнего входа в систему.
- Управление пользователями платформы:
 - Администратор может видеть всех пользователей платформы с именем пользователя, временной меткой регистрации, временной меткой последнего входа в систему и ссылкой на страницу профиля пользователя.
 - Формат ссылки: XX-module-a/user/:username, где :username заменяется именем пользователя.

- Администратор может блокировать и разблокировать пользователей, выбирая причину.
 - Заблокированные пользователи больше не могут войти в систему и вместо этого должна быть показана причина, когда они пытаются войти в систему.
 - Баллы заблокированного пользователя больше не отображаются ни в одном из рейтингов.
 - Профиль заблокированного пользователя больше не доступен и вместо этого обрабатывается, как будто его не было (HTTP Status 404 NOT FOUND).
 - Если пользователь уже был вошел в систему, ему будет недоступны функции портала и показана причина в следующий раз, когда он попытается обращаться к API, т.е. любой запрос на бэкэнд, который требует аутентификации, вместо этого вернется HTTP статус 401 UNAUTHORIZED с JSON в теле ответа содержащим причину.
 - Причины могут быть одной из: "You have been blocked by an administrator", "You have been blocked for spamming", "You have been blocked for cheating".
- Управление играми:
 - Администратор может видеть все игры с заголовком, описанием, миниатюрой (если есть), автором, временные метки всех версий и ссылку на страницу игры.
 - Формат ссылки: XX-module-a/game/:slug, где :slug заменяется слагом игры.
 - Администратор может фильтровать игры на странице, осуществляя поиск по заголовку, описанию, автору игры. Примечание: JavaScript разрешен здесь.
 - Администратор может пометить игру как удаленную в базе данных. Пользователи платформы (игроки или разработчики) больше не увидят игру. Администратор по-прежнему видит её со статусом «deleted».
 - Администратор может видеть все счета всех игроков по играм на каждую версию.

- Администратор может сбросить счета для игры. Это означает, что все счета, связанные с любой из версий игры, удалены из базы данных, и игроки больше не увидят их.
- Администратор может удалить отдельный счет для игрока в игре.
- Администратор может удалить все счета одного игрока для данной игры (для всех игровых версий).

Портал администратора должен быть реализован с использованием рендеринга на стороне сервера. Это означает, что бэкэнд должен отображать HTML-страницы, и каждое взаимодействие с бэкэнд должно привести к полной перезагрузке страницы. JavaScript не допускается, за исключением фильтрации игр.

Страницы не должны придерживаться хороших практик дизайна, но должны быть функционально удобными.

REST API

API предназначено для фронтенда с использованием запросов AJAX. API не сохраняет состояния и использует JSON для обмена данными.

API предоставляет:

- Аутентификацию
 - Регистрация пользователя
 - Вход пользователя
 - Выход пользователя
- Возможности разработчика:
 - CRUD (create, read, update, delete) операции для авторов игр
- Возможности игрока:
 - Поиск доступных игр
 - Загрузка игр для игры
 - Сохранение игровых счетов
 - Просмотр информации пользователя
 - Имя, дата и время регистрации
 - Счет за игру
 - Загруженные игры

- Видеть самые высокие счета каждого игрока в игре

Файлы игры

Разработчики могут загружать новые версии игр, предоставляя zip-файл. Файл ZIP должен содержать хотя бы файл index.html.

По желанию также может содержать файл thumbnail.png, который отображается для администраторов и пользователей.

Каждая версия игры получает уникальный идентификатор и доступна по определенному пути.

База данных

База данных должна быть настроена вами. Вам не предоставляется схема. Вы должны использовать базу данных для хранения данных, упомянутых в модели предметной области. Вы можете определить типы, добавить таблицы и столбцы, как вы считаете нужным. Ожидается, что ваша схема будет нормализована (все атрибуты (столбцы базы данных) функционально зависят исключительно от первичной ключа; третья нормальная форма или выше приемлема).

Вы также должны заполнить базу данных следующим примером данных для оценки. Поэтому, пожалуйста, убедитесь, что вы все верно добавили.

Вы можете сделать разумные предположения для значений временной метки, которые здесь не указаны.

В конце, помимо настройки базы данных, вы должны предоставить ER-диаграмму и SQL дампа, который создает схему и вставляет данные ниже в базу данных. Сохраните его под именем dump.sql в вашей рабочей папке.

Администраторы

username	password
admin1	hellouniverse1!
admin2	hellouniverse2!

Пользователи

username	password
player1	helloworld1!
player2	helloworld2!
dev1	hellobyte1!
dev2	hellobyte2!

Игры

title	slug	description	author
Demo Game 1	demo-game-1	This is demo game 1	dev1
Demo Game 2	demo-game-2	This is demo game 2	dev2

Версии игр

game	files
Demo Game 1	upload files provided to you under media files named demo-game-1-v1/ (this will be the first version, not shown to anyone)
Demo Game 1	upload files provided to you under media files named demo-game-1-v2/ (this should be the latest version)
Demo Game 2	upload files provided to you under media files named demo-game-2-v1/

Счета

user	game version	score
player1	first version of "Demo Game 1"	10.0
player1	first version of "Demo Game 1"	15.0
player1	latest version of "Demo Game 1"	12.0
player2	latest version of "Demo Game 1"	20.0
player2	latest version of "Demo Game 2"	30.0
dev1	latest version of "Demo Game 1"	1000.0
dev1	latest version of "Demo Game 1"	-300.0
dev2	latest version of "Demo Game 1"	5.0
dev2	latest version of "Demo Game 2"	200.0

Аутентификация

Аутентификация администраторов и пользователей платформы выполняется с использованием имени пользователя и пароля, однако механизм отличается.

Аутентификация администратора

Администраторы входят в систему и получают сессию cookie. Сессия хранится в файловой системе сервера (stateful) и истекает автоматически после продолжительности настроенной для сессии (убедитесь, что это ≥ 1 час). Сессия cookie отправляется с каждым запросом на бэкэнд.

Аутентификация API REST

Когда пользователь платформы входит в систему, API возвращает токен сеанса. Затем клиент отвечает за хранение токена и отправку его с каждым запросом на бэкэнд в качестве заголовка

в HTTP-запросе. На бэкэнде хранится токен в базе данных с временной меткой истечения срока действия. Токен не имеет состояния и истекает автоматически через один час.

Если пользователь выходит из системы, токен сеанса удаляется из базы данных.

ПРИМЕЧАНИЕ. Пользователь может попытаться войти в систему с нескольких устройств/браузеров одновременно и не должен входить в систему при входе в систему с другим устройством/браузером (или используя режим инкогнито).

Администраторы могут отозвать токены сеанса пользователя, чтобы насильно вывести пользователя, когда они хотят заблокировать пользователя.

Вы можете использовать любой формат токена (например, uuid или что-либо, до чего не может догадаться машина в течении разумного времени).

Спецификация REST API

Общая информация:

- Тела ответов в спецификации содержат некоторые статические примеры данных. Должны использоваться динамические данные из базы данных.
- Параметры заполнителей в URL предваряются двоеточием (например :slug или :username).
- Порядок свойств в объектах не имеет значения, но порядок в массивах важен.
- Заголовок Content-Type в ответе должен быть application/json для POST, PUT, PATCH.
- Заголовок Content-Type в ответе всегда должен быть application/json, если явно не указано другое значение.
- Временные метки должны соответствовать формату ISO-8601.
Например 2032-01-31T21:59:35.000Z.
- Указанные URL-адреса относятся к базовому URL-адресу API. Например /api/v1/games это URL, чтобы получить все игры.
- URL-адреса API не должны заканчиваться .php или .html или любого другого расширения файла. Файлы игры являются исключением из этого.
- Токен для закрытых эндпоинтов должен передаваться как Bearer токен в заголовке `Authorization`. Т.е `Authorization: Bearer <token>`

POST /api/v1/auth/signup

Этот эндпоинт создает нового пользователя и возвращает токен сеанса.

Request Body:

```
{
  "username": "testuser",
  "password": "asdf1234"
}
```

Property	Comment
username	required, unique, min length 4, max length 60
password	required, min length 8, max length 2^16

Response:

Successful creation response:

Status Code: 201

Response Body:

```
{
  "status": "success",
  "token": "xxx"
}
```

POST /api/v1/auth/signin

Этот эндпоинт проверяет имя пользователя и пароль для всех пользователей. Если обнаружен, возвращается токен сеанса.

Успешный ответ:

Status Code: 200

Request Body:

```
{
  "username": "testuser",
  "password": "asdf1234"
}
```

Property	Comment
username	required, min length 4, max length 60
password	required, min length 8, max length 2^16

Response Body:

```
{
  "status": "success",
  "token": "xxx"
}
```

Ответ при неверном username/password:

Status Code: 401

Response Body:

```
{
  "status": "invalid",
  "message": "Wrong username or password"
}
```

POST /api/v1/auth/signout

Удаляет токен текущего сеанса.

Успешный ответ:

Status Code: 200

Response Body:

```
{
  "status": "success"
}
```

GET /api/v1/games

Возвращает список игр с возможностью пагинации.

Query Parameters:

Параметр	Описание	По умолчанию
page	Номер страницы. Начинается с 0.	0
size	Количество элементов. Должно быть больше или равно 1.	10
sortBy	Поле для сортировки. Должен быть один из "title", "popular", "uploaddate"	title

Параметр	Описание	По умолчанию
sortDir	Направление сортировки. Должен быть один из "asc" или "desc"	asc

Описание полей сортировки:

Поле	Описание
title	Заголовок игры
popular	Считает общее количество результатов за игру и сортирует по этому количеству
uploaddate	По временной метке загрузки последней версии игры.

В `content`, поля `thumbnail` и `uploadTimestamp` относятся только к последней версии.

Поле `scoreCount` это сумма результатов по всем версиям.

Успешный ответ:

Status Code: 200

Response Body:

```
{
  "page": 0,
  "size": 10,
  "totalElements": 15,
  "content": [
    {
      "slug": "demo-game-1",
      "title": "Demo Game 1",
      "description": "This is demo game 1",
      "thumbnail": "/games/:slug/:version/thumbnail.png",
      "uploadTimestamp": "2032-01-31T21:59:35.000Z",
      "author": "dev1",
      "scoreCount": 5
    }
  ]
}
```

Описание полей в теле ответа:

Поле	Описание
page	Запрашиваемый номер страницы. Начинается с 0.
size	Фактический размер страницы. Должен быть меньше или равен, чем запрошенный размер страницы.
totalElements	Общее количество элементов независимо от страницы.
content	Массив игр на странице.

Можно рассчитать, сколько страниц есть:

`pageCount = ceil(totalElements / requestedSize)`

Он также может быть вычислен, если возвращаемая страница является последней страницей, умножая (page+1) по запрошенному размеру страницы и проверяя, меньше ли результата или равна общим элементам.

`isLastPage = (page + 1) * requestedSize >= totalElements`

ПРИМЕЧАНИЕ 1: Если есть игра, в которой пока нет игровой версии, она не включена ни в ответ, ни в общее количество. ПРИМЕЧАНИЕ 2: Если нет миниатюры, поле миниатюры равно null.

POST /api/v1/games

Этот эндпоинт может быть использован для создания игры. Тем не менее, версия игры должна быть загружена отдельным эндпоинтом. Если в игре еще нет версии, она не возвращается этим эндпоинтом.

Request Body:

```
{
  "title": "Demo Game 3",
  "description": "This is demo game 3"
}
```

Property	Comment
title	required, min length 3, max length 60

Property	Comment
description	required, min length 0, max length 200

Response:

Успешный ответ:

Status Code: 201

Response Body:

```
{
  "status": "success",
  "slug": "generated-game-slug"
}
```

Существующий slug:

Если сгенерированный slug не уникален, игра не может быть создана, а вместо этого возвращается следующий ответ.

Response:

Status Code: 400

```
{
  "status": "invalid",
  "slug": "Game title already exists"
}
```

GET /api/v1/games/:slug

Возвращает детали игры.

Response:

Status Code: 200

```
{
  "slug": "demo-game-1",
  "title": "Demo Game 1",
  "description": "This is demo game 1",
  "thumbnail": "/games/:slug/:version/thumbnail.png",
  "uploadTimestamp": "2032-01-31T21:59:35.000Z",
  "author": "dev1",
  "scoreCount": 5,
  "gamePath": "/games/demo-game-1/1/"
}
```

Если нет миниатюры, поле миниатюры равно null.

Поле «gamePath» указывает на URL-пути, который могут использовать браузеры для визуализации игры. Это означает, что это достижимый путь для загрузки.

Загрузка файла игры POST /api/v1/games/:slug/upload

Пользователь может загрузить новую версию игры, если он является автором этой игры.

- Это не эндпоинт REST и скорее он принимает загружаемый файл. Имя параметра `zipfile`.
- Версия игры нумеруется целым числом и автоинкрементно. Первая версия это `1`.
- Пользователь не может управлять версией.
- Токен сеанса должен быть предоставлен как параметр формы `token`.
- Загруженный файл - это zip-файл, который извлекается, а затем предоставляется по публичному пути.
- Путь должен храниться в записи игры, чтобы игроки могли найти игровые файлы.
- Если zip-файл содержал thumbnail.png, его путь хранится как thumbnail в записи игры.

Если загрузка не выполняется из-за одной из этих возможных причин, ответ должен быть простым текстовым объяснением ошибки.

- User is not author of the game
- ZIP file extraction fails
- File size too big
- Unspecified IO error

Предоставление игровых файлов GET /games/:slug/:version/

Файлы игры, которые были загружены, предоставляются под тем путем, который является общедоступным.

Идея заключается в том, что фронт может создать iframe, указывающий на этот путь. Это загружает index.html, который в перспективе может загружать файлы JavaScript и CSS с относительного пути.

PUT /api/v1/games/:slug

Этот эндпоинт позволяет автору игры обновить заголовок и описание игры.

Request Body:

```
{
  "title": "Demo Game 1 (updated)",
  "description": "Updated description"
}
```

Примечание: это не обновляет slug игры.

Response:***Успешный ответ на обновление:***

Status Code: 200

Response Body:

```
{  
  "status": "success"  
}
```

Ответ, если пользователь не является автором игры:

Status Code: 403

Response Body:

```
{  
  "status": "forbidden",  
  "message": "You are not the game author"  
}
```

DELETE /api/v1/games/:slug

Автор может удалить свою игру. Это удаляет игру, все версии и все результаты.

Response:***Успешный ответ на удаление:***

Status Code: 204

Возвращает пустое тело. Заголовок `Content-Type` не обязательно должен быть `application/json`.

Ответ, если пользователь не является автором игры:

Status Code: 403

Response Body:

```
{  
  "status": "forbidden",  
  "message": "You are not the game author"  
}
```

GET /api/v1/users/:username

Возвращает данные пользователя.

Response:

Status Code: 200

Response Body:

```
{
  "username": "dev1",
  "registeredTimestamp": "2032-01-31T21:59:35.000Z",
  "authoredGames": [
    {
      "slug": "demo-game-1",
      "title": "Demo Game 1",
      "description": "This is demo game 1"
    }
  ],
  "highscores": [
    {
      "game": {
        "slug": "demo-game-1",
        "title": "Demo Game 1",
        "description": "This is demo game 1"
      },
      "score": 15,
      "timestamp": "2032-01-31T21:59:35.000Z"
    }
  ]
}
```

authoredGames - это массив, который возвращает все игры, по крайней мере, с одной версией, для которой данный пользователь является автором. Если пользователь, запрашивающий данные пользователя, является самим пользователем, возвращаются также игры, в которых еще нет версии.

highscores - это массив самых высоких результатов за игры в которые играл пользователь.

GET /api/v1/games/:slug/scores

Возвращает самые высокие баллы каждого игрока, который играл в любую версию игры, отсортированную по баллам (по убыванию).

Response:

Status Code: 200

Response Body:

```
{
  "scores": [
    {
      "username": "player2",
      "score": 20,
      "timestamp": "2032-01-31T21:59:35.000Z"
    },
    {
      "username": "player1",
      "score": 15,
      "timestamp": "2032-01-31T21:59:35.000Z"
    }
  ]
}
```

POST /api/v1/games/:slug/scores

Когда пользователь заканчивает работу игры, счет может быть опубликован этим эндпоинтом.

Request Body:

```
{
  "score": 100
}
```

Игровая версия, связанная со счетом, является последней доступной.

Response:

Успешный ответ на создание:

Status Code: 201

Response Body:

```
{
  "status": "success"
}
```

Невалидные запросы

Если в запросах POST или PUT были невалидные поля, они проверяются, и возвращается ответ со списком нарушений.

Status Code: 400

Response Body:

```
{
  "status": "invalid",
  "message": "Request body is not valid.",
  "violations": {
    "field_name": {
      "message": "required"
    },
    "field_name": {
      "message": "must be at least 4 characters long"
    },
    "field_name": {
      "message": "must be at most 60 characters long"
    }
  }
}
```

В приведенном выше примере показаны все возможные нарушения. Фактические возвращаемые нарушения должны быть только полями, которые фактически были недействительными. Показывается не более одной проверки на поле. Валидации выполняются в порядке появления в примере выше. field_name должен быть заменен фактическим именем поля.

Сообщения для длины должны включать фактическое значение требования длины.

Отсутствует или недействительный заголовок аутентификации

Если пользователь обратиться к эндпоинту, который требует присутствия заголовка auth, это должен быть ответ:

Status Code: 401

Response Body:

```
{
  "status": "unauthenticated",
  "message": "Missing token"
}
```

Если пользователь обратиться к эндпоинту, который требует присутствия заголовка auth, но токен недействительный или нет токена, это должен быть ответ:

Status Code: 401

Response Body:

```
{
  "status": "unauthenticated",
  "message": "Invalid token"
}
```

Если пользователь обратиться к эндпоинту, который требует присутствия заголовка auth, но пользователь заблокирован, это должен быть ответ:

Status Code: 403

Response Body:

```
{
  "status": "blocked",
  "message": "User blocked",
  "reason": "You have been blocked by an administrator"
}
```

Примечание: reason - это динамическое значение, выбранное администратором, который блокирует пользователя.

Следующие методы и шаблоны пути требуют действительного заголовка сеанса:

- POST /api/v1/auth/signout
- POST, PUT, DELETE /api/v1/games/**
- GET /api/v1/users/**

Не существующий путь API

Если пользователь вызовет эндпоинт или ресурс, который не существует, это должен быть ответ:

Status Code: 404

Response Body:

```
{
  "status": "not-found",
  "message": "Not found"
}
```

ИНСТРУКЦИИ ДЛЯ УЧАСТНИКА

- В случае, если вы не выполнили каждое задание, предоставить команде оценки доступ к выполненным заданиям - это выбор и ответственность участника.
- Вы должны учитывать качество кода.
- Пожалуйста, используйте надлежащий контроль версий (ветви и коммиты) во время разработки.