Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Владимирский государственный университет

им. А.Г. и Н.Г. Столетовых

Кафедра информационных систем и программной инженерии

КУРСОВОЙ ПРОЕКТ

по дисциплине

«Технологии разработки мобильных приложений»

на тему:

«Мобильное приложение для выявления совместимости и интересов. Серверная часть»

Выполнил: ст. гр. ПРИ-117

Емельянов Д.В.

Принял: Салех Х.М.

Владимир, 2020

**Аннотация**

Разработанная система представляют собой серверную часть для мобильного приложения для выявления совместимости и интересов. Система разрабатывалась с использованием ASP.NET Core, PostgreSQL, Firebase.

Курсовой проект представлен на 45 страницах, рисунков - 6, использованных источников - 6, приложений - 2.

The developed system is a back-end for a mobile application to identify compatibility and interests. The system was developed using ASP.NET Core, PostgreSQL, Firebase.The course project is presented on 45 pages, pictures - 6, sources used - 6, attachments - 2.

**Содержание**

[ВВЕДЕНИЕ 5](#_Toc59438354)

[1 АНАЛИЗ ЗАДАЧИ 6](#_Toc59438355)

[1.1. Описание предметной области 6](#_Toc59438356)

[1.2. Цель и задачи работы 6](#_Toc59438357)

[1.3. Функциональные требования к системе 7](#_Toc59438358)

[1.4. Нефункциональные требования к системе 8](#_Toc59438359)

[1.5. Диаграмма прецедентов 8](#_Toc59438360)

[1.6. Словарь предметной области 11](#_Toc59438361)

[2 ПРОЕКТИРОВАНИЕ СЕРВЕРНОЙ ЧАСТИ 12](#_Toc59438362)

[2.1. Диаграммы классов 12](#_Toc59438363)

[2.2. Структуры баз данных 13](#_Toc59438364)

[2.3. Архитектура системы 14](#_Toc59438365)

[3 РЕАЛИЗАЦИЯ СЕРВЕРНОЙ ЧАСТИ СИСТЕМЫ 15](#_Toc59438366)

[3.1. Используемые технологии 15](#_Toc59438367)

[3.2. Используемые библиотеки 15](#_Toc59438368)

[3.3. Спецификация API 15](#_Toc59438369)

[3.4. Настройка Firebase 31](#_Toc59438370)

[3.5. Настройка и реализация Entity Framework Core 31](#_Toc59438371)

[3.6. Реализация контроллеров 32](#_Toc59438372)

[3.7. Настройка логгирования 32](#_Toc59438373)

[3.8. Структура проекта 32](#_Toc59438374)

[4 ТЕСТИРОВАНИЕ СЕРВЕРНОЙ ЧАСТИ 33](#_Toc59438375)

[4.1. Тестирование с помощью SWAGGER 33](#_Toc59438376)

[4.2. Тестирование с помощью xUnit 33](#_Toc59438377)

[4.3. Тестирование с помощью Insomnia Core 33](#_Toc59438378)

[5 ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ СЕРВЕРНОЙ ЧАСТИ 34](#_Toc59438379)

[ЗАКЛЮЧЕНИЕ 35](#_Toc59438380)

[6 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 36](#_Toc59438381)

[ПРИЛОЖЕНИЕ А 37](#_Toc59438382)

[ПРИЛОЖЕНИЕ Б 42](#_Toc59438383)

ВВЕДЕНИЕ

На данный момент мобильные технологии являются неотъемлемой частью жизни человека. Рост рынка разработки мобильных устройств является показателем значимости и актуальности использования мобильных устройств как в развлекательных целях, так и в бизнесе, и для исследования личности с помощью опросников и тестов. Прохождение тестов и опросников с помощью мобильного устройства очень удобно: не нужно лично являться в место проведения опроса, пройти тест может любой желающий, имеющий смартфон на базе Android, а собирать и обрабатывать информацию будет намного проще и быстрее, поскольку не понадобится тратить время на подсчет результатов.

В рамках данного курсового проекта будет разработана серверная часть для мобильного приложения для выявления совместимости и интересов по средствам опроса. Данное приложение может быть использовано в целях выявления предпочтений двух и более людей. Пользователь сможет пройти тестирования и затем сравнить его с другими пользователями. Интерфейс приложения будет дружелюбным, чтобы пользовать на интуитивном уровне понимал, что от него требуется.

1 АНАЛИЗ ЗАДАЧИ

1.1. Описание предметной области

Предметной областью является определение совместимости предпочтений пользователей на основе выбора записей по определенной категории.

Совместимость, в данном контексте - совпадение мнений пользователей по определенным вопросам и представлениям чего-либо. В численном эквиваленте совместимость это одинаковое мнение по вопросам, ответы на которые подразумевают “да” или “нет” по не менее чем половине вопросов. Для проверки совместимости считается численное соотношение ответов “да” и “нет” и их совпадения по тем или иным вопросам. На основе совпадений выявляется совместимы интересы и предпочтения пользователей или нет. Значение совместимости может изменяться у одних и тех же пользователей с течением времени, так как зачастую ответы на вопросы зависят от текущей ситуации в обществе, местонахождения пользователей, настроения каждого пользователя и тому подобное.

В рамках курсового проекта в качестве записей для проверки интересов используются посты социальной сети Reddit, получаемые по открытому API. Однако теоретически в качестве записей может быть использована любая другая предметная область.

1.2. Цель и задачи работы

**Цель работы** – создать сервер для мобильного приложения, предоставляющее пользователю интерактивную игру в формате быстрого опроса для выявления совместимости и интересов 2 и более человек, а также проведения анализа изменений предпочтений одного человека

**Задачи:**

* предоставлять пользователю интерактивную игру в формате быстрого опроса;
* предоставлять пользователю статистику о проведенных играх в виде диаграмм и графиков;
* позволять пользователю настраивать профиль в приложении.

1.3. Функциональные требования к системе

К функциональным требованиям данной системы относятся:

* регистрация и авторизация пользователей;
* предоставл ять возможность аутентификации с помощью аккаунта Google;
* предоставлять пользователю возможность управления профилем;
* создавать и настраивать сессию опросника;
* настраивать время, отведенное на прохождение опросника;
* настраивать количество участников;
* настраивать количество записей, которое нужно выбрать;
* приглашать в сессию опросника других пользователей;
* выбор опросов по определенным категориям записей;
* прохождение опросника по записям;
* сравнение своих опросов с опросами других пользователей;
* прохождение опроса заново и сравнение с прошлым своим результатом;
* отмена прохождения опроса, без сохранения состояния;
* использование пользователями с ролью «администратор» панели «администрирование пользователей» для управления пользователями;
* использование пользователями с ролью «модератор опросов» панели «администрирование опросов» для создания и удаления опросов, добавления новых категорий;
* количество участников в игре от 1 до 6 человек;
* количество записей для выбора в игре от 1 до 35;
* время игры от 1 минуты до 30 минут.

1.4. Нефункциональные требования к системе

К нефункциональным требованиям данной системы относятся:

* Понятный и удобный пользовательский интерфейс;
* быстрое реагирование системы на действия пользователей;
* подключение приложения к сети “Интернет”;
* СУБД: PostgreSQL;
* технологии: .NET Core, Flutter.

1.5. Диаграмма прецедентов

Диаграмма прецедентов представлена на рисунке 1.

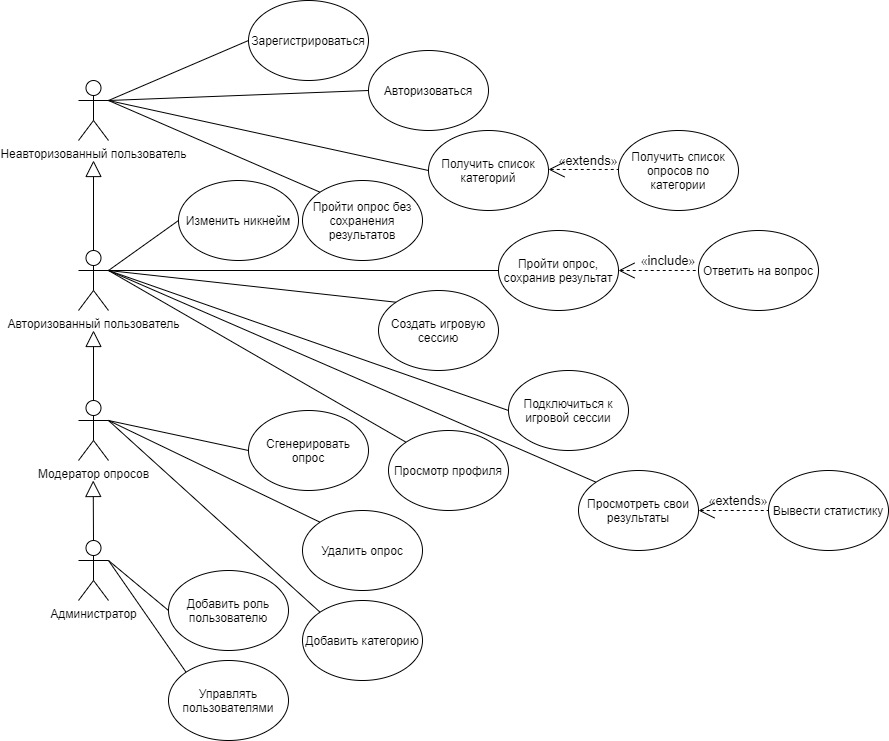


Рисунок 1 - Диаграмма прецедентов

Описание основных прецедентов:

Название «Изменить никнейм»

Актер: Авторизованный пользователь.

Основной поток: Пользователь изменяет свой никнейм в системе.

Постусловие: Никнейм изменился.

Название «Получить список категорий».

Актер: Любой пользователь.

Основной поток: Пользователь желает получить список, он нажимает на кнопку получения списка категорий, ему отображается данный список.

Название «Пройти опрос без сохранения результатов»

Актер: Любой пользователь

Предусловие: Пользователь находится на странице с опросами.

Основой поток: Пользователь выбирает опросник, отвечает на вопросы, не сохраняя ответы.

Название «Пройти опрос, сохранив результат»

Актер: Авторизованный пользователь.

Предусловие: Пользователь авторизировался в приложении.

Основной поток: Пользователь выбирает опросник, отвечает на вопросы, сохраняя ответы на вопросы.

Постусловие: Опросник с ответами сохранен в истории пользователя.

Название «Создать игровую сессию»

Актер: Авторизованный пользователь.

Предусловие: Пользователь авторизировался в приложении.

Основной поток: Пользователь выбирает опросник, выбирает настройки сессии: время, отведённое на опросник, количество выбранных записей, которое нужно набрать каждому пользователю сессии, категорию прохождения опросника, максимальное количество людей в сессии.

Постусловие: Созданная сессия с номером, которым пользователь может поделиться с другими пользователями.

Название «Подключиться к игровой сессии»

Актер: Авторизованный пользователь.

Предусловие: Известен номер созданной другим пользователем игровой сессии.

Основной поток: Пользователь вводит номер сессии, к которой он хочет подключиться.

Постусловие: Пользователь проходит опрос из сессии.

Название «Удалить опрос»

Актер: Модератор опросов.

Предусловие: Выбран опрос.

Основной поток: Модератор удаляет опрос, все его упоминания также удаляются.

Постусловие: Опрос удален.

Название «Сгенерировать опрос»

Актер: Модератор опросов.

Предусловие: Выбрана категория.

Основной поток: Модератор выбрал категорию, заполняет настройки опросника и генерирует его.

Постусловие: Сгенерирован новый опросник.

1.6. Словарь предметной области

**Опросник** **-** это выяснение мнения определенного пользователя о записях в определенной категории.

**Категория** - группа записей, воспринимаемые пользователями как взаимосвязанные и взаимозаменяемые.

**Запись** - элемент опроса, который вы выбираете. Запись обычно состоит из изображения и подписи к нему. Запись может быть связана с каким-либо товаром, событием, шуткой и т.п.

**Свайп** - специальный жест, когда вы кладете палец на экран смартфона и ведете его в нужном направлении по экрану (также глагол “смахнуть”).

**Сессия опросника** - прохождение определенного опроса пользователем по определенной категории, возможно, с участием других пользователей.

**Выбранная запись** - запись, которую пользователь смахнул вправо (или нажал кнопку “Да”).

**Совпадение записей** - ситуация, когда во время сессии опросника все участвующие пользователи выбрали определенную запись.

Максимальное время игры - время, отведённое на прохождение опросника. Если до истечения этого времени пользователь не выбрал необходимого числа записей, то в зачёт идут только те записи, что он успел выбрать.

# 2 **ПРОЕКТИРОВАНИЕ СЕРВЕРНОЙ ЧАСТИ**

При проектировании серверной части использовались требования из первой части.

2.1. Диаграммы классов

Диаграмма классов-моделей уровня анализа представлена на рисунке 2.

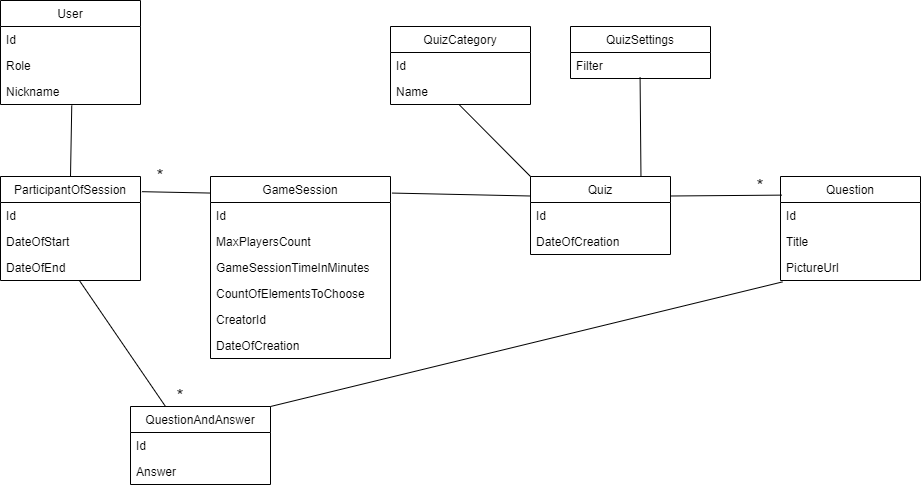


Рисунок 2 - Диаграмма классов-моделей уровня анализа

Quiz(опрос) имеет набор Question(вопросов). GameSession(игровая сессия) может иметь несколько ParticipantOfSession(участник сессии). ParticipantOfSession(участник сессии) может иметь несколько QuestionAndAnswer(ответы на вопросы).

Диаграмма классов-логики уровня анализа представлена на рисунке 3.

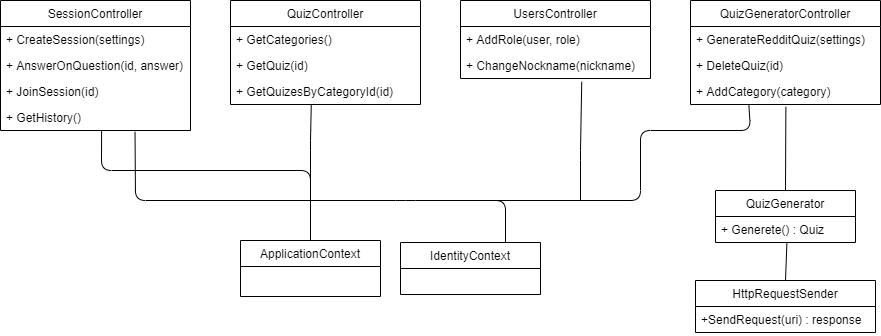


Рисунок 3 - Диаграмма классов- логики уровня анализа

Приложении будет содержать несколько контекстов для обращения к базам данных (ApplicationContext и IdentityContext), ими будут пользоваться контроллеры, которые обрабатывают запросы пользователей. Также будет класс для обращения к сторонним сервисам, используя http протокол, им будет пользоваться класс для генерации опросников (QuizGenerator).

2.2. Структуры баз данных

Во время проектирования системы было принято решение разделить базу данных на две. Первая – это база данных для пользователей, их ролей и никнеймов, так как Firebase не подходит для этого, вторая – это база данных для всей логики, связанной с опросниками. Структура базы данных для пользователей представлена на рисунке 4. Структура базы данных для логики опросников представлена на рисунке 5.

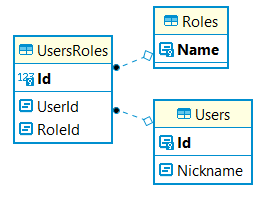


Рисунок 4 - Структура базы данных для пользователей

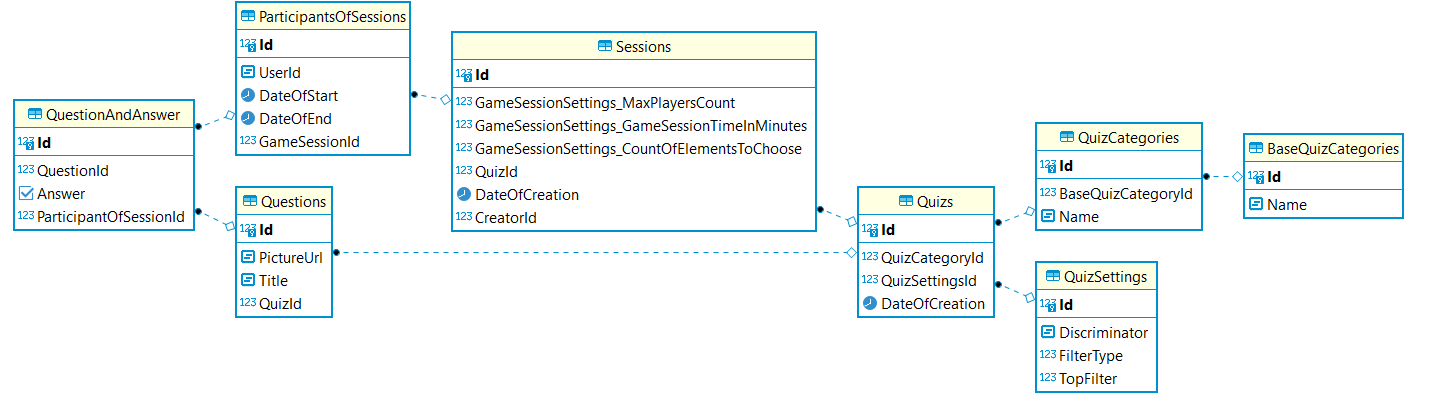


Рисунок 5 - Структура базы данных для логики опросников

2.3. Архитектура системы

Трехзвенная архитектура системы представлена на рисунке 6. Она состоит из: клиента – 1 звено, основного сервера приложения – 2 звено, стороннего сервера аутентификации – 3 звено и двух баз данных.

В качестве стороннего сервиса аутентификации был выбран Firebase.

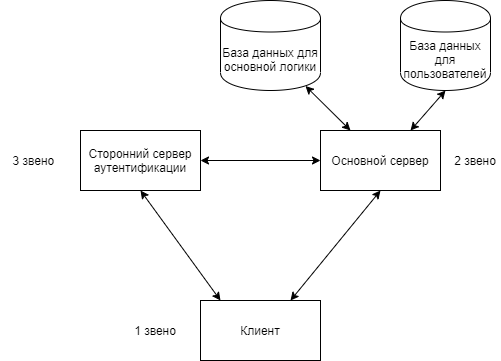


Рисунок 6 - Архитектура системы

3 РЕАЛИЗАЦИЯ СЕРВЕРНОЙ ЧАСТИ СИСТЕМЫ

3.1. Используемые технологии

В качестве платформы для разработки серверной части приложения был выбран ASP.NET Core благодоря его высокопроизводительности, кроссплатформенности [5].

В качестве СУБД была выбрана PostgreSQL — свободная объектно-реляционная система управления базами данных. Существует в реализациях для множества UNIX-подобных платформ, включая AIX, различные BSD-системы, HP-UX, IRIX, Linux, macOS, Solaris/OpenSolaris, Tru64, QNX, а также для Microsoft Windows.

В качестве сервиса аутентификации был выбрал Firebase предоставляющий возможность регистрироваться в системе с помощью email, google аккаунта, телефона и др.

3.2. Используемые библиотеки

Newtonsoft.Json – библиотека для сериализации и десериализации объект в JSON.

Serilog – сторонний логгер для приложения.

Swashbuckle.AspNetCore – библиотека для подключения swagger’а в проект.

Entity Framework Core - современный модуль сопоставления "объект — база данных" для .NET.

xUnit.net - инструмент тестирования.

3.3. Спецификация API

Все запросы к API начинаются со /api. Описание основных методов представлено в таблице 1.

Запросы, которые требуют авторизации, должны иметь заголовок вида: *Authorization: Bearer <User-token>*

Запросы, которые имеют тело, должны иметь заголовок вида:

*Content-type: Application/json*

При ошибке авторизации возвращается ошибка 401 и текстом «Unauthorized».

При ошибке доступа возвращается ошибка со статусом 403 и текстом «Forbidden».

При ошибке наполнения тела запроса возвращается ответ со статусом 400 и описанием ошибки в объекте errros.

При ошибке исполнения запроса, связанной с данными из тела, возвращается ответ со статусом 500 и описанием ошибке в объекте detail.

Таблица 1 - Спецификация API

|  |  |
| --- | --- |
| **Получение категорий опросников** | |
| Метод | URL |
| GET | /quiz/getcategories |
| Авторизация | Не требуется |
| Тело запроса | |
| - | |
| Ответ | |
| Статус | Тело |
| 200 | [  {  "BaseCategory":{  "Id":1,  "Name":"Reddit"  },  "QuizCategories":[  {  "Id":1,  "BaseQuizCategory":{  "Id":1,  "Name":"Reddit"  },  "Name":"pics"  },  {  "Id":2,  "BaseQuizCategory":{  "Id":1,  "Name":"Reddit"  },  "Name":"memes"  }  ]  }  ] |
| **Получение списка опросников по id категории** | |
| Метод | URL |
| GET | /quiz/bycategory/{categoryId} |
| Авторизация | Не требуется |
| Тело запроса | |
| - | |
| Ответ | |
| Статус | Тело |
| 200 | [  {  "Id": 2,  "QuizSettings": {  "FilterType": 0,  "TopFilter": null,  "Id": 2  },  "DateOfCreation": "2020-12-20T14:10:21.360663"  },  {  "Id": 3,  "QuizSettings": {  "FilterType": 1,  "TopFilter": null,  "Id": 3  },  "DateOfCreation": "2020-12-20T14:10:35.024049"  }  ] |
| **Получение опросника по id** | |
| Метод | URL |
| GET | /quiz/{quizId} |
| Авторизация | Не требуется |
| Тело запроса | |
| - | |
| Ответ | |
| Статус | Тело |
| 200 | {  "Id": 1,  "QuizCategory": {  "Id": 2,  "BaseQuizCategory": null,  "Name": "memes"  },  "QuizSettings": {  "FilterType": 1,  "TopFilter": null,  "Id": 1  },  "DateOfCreation": "2020-12-19T19:45:38.639799",  "CountOfQuestions": 3,  "Questions": [  {  "Id": 1,  "PictureUrl": "https://i.redd.it/0bobdyzm67661.jpg",  "Title": "Top 10 Korean Pranks"  },  {  "Id": 2,  "PictureUrl": "https://i.redd.it/0f0njh9e57661.jpg",  "Title": "ending 2020 with rickrolls"  },  {  "Id": 3,  "PictureUrl": "https://i.redd.it/2bpqbked57661.jpg",  "Title": "I seriously jumped when a body just flew into my face for no reason"  }  ]  } |
| **Генерация опросника** | |
| Метод | URL |
| POST | /quizgenerator/reddit |
| Авторизация | Требуется |
| Тело запроса | |
| {  "QuizCategoryId":1,  "FilterType":1,  "TopFilter":null  } | |
| Ответ | |
| Статус | Тело |
| 200 | {  "Id": 3,  "QuizCategory": {  "Id": 1,  "BaseQuizCategory": {  "Id": 1,  "Name": "Reddit"  },  "Name": "pics"  },  "QuizSettings": {  "FilterType": 1,  "TopFilter": null,  "Id": 3  },  "DateOfCreation": "2020-12-20T14:10:35.024049Z",  "CountOfQuestions": 69,  "Questions": [  {  "Id": 138,  "PictureUrl": "https://i.imgur.com/AaHiphL.jpg",  "Title": "Central Park NYC"  },  {  "Id": 174,  "PictureUrl": "https://i.redd.it/xsxw43p3kc661.jpg",  "Title": "Can someone please shut down the algorithm that produces these Christmas movies?"  },  {  "Id": 175,  "PictureUrl": "https://i.imgur.com/13F1B5zg.jpg",  "Title": "Sandra Bullock kissing Scarlett Johansson at MTV Movie Awards (2010)"  }  ]  } |
| **Изменение никнейма** | |
| Метод | URL |
| POST | /users/changenick |
| Авторизация | Требуется |
| Тело запроса | |
| {  "value": "newnickname"  } | |
| Ответ | |
| Статус | Тело |
| 200 | - |
| **Создание игровой сессии** | |
| Метод | URL |
| POST | /session /create /{quizId} |
| Авторизация | Требуется |
| Тело запроса | |
| {  "maxPlayersCount": 0,  "gameSessionTimeInMinutes": 0,  "countOfElementsToChoose": 0  } | |
| Ответ | |
| Статус | Тело |
| 200 | {  "Id": 6,  "GameSessionSettings": {  "MaxPlayersCount": 3,  "GameSessionTimeInMinutes": 5,  "CountOfElementsToChoose": 10  },  "Quiz": {  "Id": 1,  "QuizCategory": {  "Id": 2,  "BaseQuizCategory": null,  "Name": "memes"  },  "QuizSettings": {  "FilterType": 1,  "TopFilter": null,  "Id": 1  },  "DateOfCreation": "2020-12-19T19:45:38.639799",  "CountOfQuestions": 70,  "Questions": [  {  "Id": 1,  "PictureUrl": "https://i.redd.it/0bobdyzm67661.jpg",  "Title": "Top 10 Korean Pranks"  },  {  "Id": 2,  "PictureUrl": "https://i.redd.it/0f0njh9e57661.jpg",  "Title": "ending 2020 with rickrolls"  },  {  "Id": 3,  "PictureUrl": "https://i.redd.it/2bpqbked57661.jpg",  "Title": "I seriously jumped when a body just flew into my face for no reason"  },  {  "Id": 4,  "PictureUrl": "https://i.redd.it/thbgju3757661.png",  "Title": "I want both to be stuck in my head"  }  ]  },  "DateOfCreation": "2020-12-20T15:04:41.5669794Z",  "CreatorId": 10,  "Participants": [  {  "Id": 10,  "UserId": "bnaXDmVlygcCWfz2UzlfTdLoXhs1",  "User": {  "Id": "bnaXDmVlygcCWfz2UzlfTdLoXhs1",  "Nickname": null,  "UsersRoles": [  {  "Id": 2,  "UserId": "bnaXDmVlygcCWfz2UzlfTdLoXhs1",  "RoleId": "User"  }  ]  },  "DateOfStart": null,  "DateOfEnd": null,  "QuestionsAndAnswers": null  }  ]  } |
| **Присоединиться к игровой сессии** | |
| Метод | URL |
| POST | /session /join |
| Авторизация | Требуется |
| Тело запроса | |
| {  "value": 6  } | |
| Ответ | |
| Статус | Тело |
| 200 | {  "Id": 6,  "GameSessionSettings": {  "MaxPlayersCount": 3,  "GameSessionTimeInMinutes": 5,  "CountOfElementsToChoose": 10  },  "Quiz": {  "Id": 1,  "QuizCategory": {  "Id": 2,  "BaseQuizCategory": null,  "Name": "memes"  },  "QuizSettings": {  "FilterType": 1,  "TopFilter": null,  "Id": 1  },  "DateOfCreation": "2020-12-19T19:45:38.639799",  "CountOfQuestions": 70,  "Questions": [  {  "Id": 1,  "PictureUrl": "https://i.redd.it/0bobdyzm67661.jpg",  "Title": "Top 10 Korean Pranks"  },  {  "Id": 2,  "PictureUrl": "https://i.redd.it/0f0njh9e57661.jpg",  "Title": "ending 2020 with rickrolls"  },  {  "Id": 3,  "PictureUrl": "https://i.redd.it/2bpqbked57661.jpg",  "Title": "I seriously jumped when a body just flew into my face for no reason"  },  {  "Id": 4,  "PictureUrl": "https://i.redd.it/thbgju3757661.png",  "Title": "I want both to be stuck in my head"  }  ]  },  "DateOfCreation": "2020-12-20T15:04:41.5669794Z",  "CreatorId": 10,  "Participants": [  {  "Id": 10,  "UserId": "bnaXDmVlygcCWfz2UzlfTdLoXhs1",  "User": {  "Id": "bnaXDmVlygcCWfz2UzlfTdLoXhs1",  "Nickname": null,  "UsersRoles": [  {  "Id": 2,  "UserId": "bnaXDmVlygcCWfz2UzlfTdLoXhs1",  "RoleId": "User"  }  ]  },  "DateOfStart": null,  "DateOfEnd": null,  "QuestionsAndAnswers": null  },  {  "Id": 11,  "UserId": "B8RskiAdPETrhUvsOnn8L9SSn9a2",  "User": {  "Id": "B8RskiAdPETrhUvsOnn8L9SSn9a2",  "Nickname": null,  "UsersRoles": [  {  "Id": 1,  "UserId": "B8RskiAdPETrhUvsOnn8L9SSn9a2",  "RoleId": "Administrator"  }  ]  },  "DateOfStart": null,  "DateOfEnd": null,  "QuestionsAndAnswers": null  }  ]  } |
| **Ответить на вопрос** | |
| Метод | URL |
| POST | /session /answer |
| Авторизация | Требуется |
| Тело запроса | |
| {  "QuestionId":1,  "Answer":true,  "ParticipantOfSessionId":4  } | |
| Ответ | |
| Статус | Тело |
| 200 | - |
| **Получить историю опросов** | |
| Метод | URL |
| POST | /session/gethistory |
| Авторизация | Требуется |
| Тело запроса | |
| - | |
| Ответ | |
| Статус | Тело |
| 200 | [  {  "Id": 1,  "GameSessionSettings": {  "MaxPlayersCount": 2,  "GameSessionTimeInMinutes": 3,  "CountOfElementsToChoose": 5  },  "Quiz": {  "Id": 1,  "QuizCategory": {  "Id": 2,  "BaseQuizCategory": null,  "Name": "memes"  },  "QuizSettings": {  "FilterType": 1,  "TopFilter": null,  "Id": 1  },  "DateOfCreation": "2020-12-19T19:45:38.639799",  "CountOfQuestions": 70,  "Questions": [  {  "Id": 1,  "PictureUrl": "https://i.redd.it/0bobdyzm67661.jpg",  "Title": "Top 10 Korean Pranks"  },  {  "Id": 2,  "PictureUrl": "https://i.redd.it/0f0njh9e57661.jpg",  "Title": "ending 2020 with rickrolls"  },  {  "Id": 3,  "PictureUrl": "https://i.redd.it/2bpqbked57661.jpg",  "Title": "I seriously jumped when a body just flew into my face for no reason"  },  {  "Id": 4,  "PictureUrl": "https://i.redd.it/thbgju3757661.png",  "Title": "I want both to be stuck in my head"  },  {  "Id": 5,  "PictureUrl": "https://i.redd.it/09uoxl8w47661.png",  "Title": "r/whatisthisthing be like"  },  {  "Id": 6,  "PictureUrl": "https://i.redd.it/m6wknbi557661.jpg",  "Title": "Restricted Christmas smh"  },  {  "Id": 7,  "PictureUrl": "https://i.redd.it/1ik6bwi457661.jpg",  "Title": "Restricted Christmas smh"  },  {  "Id": 8,  "PictureUrl": "https://i.redd.it/bofsdj1257661.jpg",  "Title": "TABS is a great game"  },  {  "Id": 9,  "PictureUrl": "https://i.imgur.com/4Npg00j.jpg",  "Title": "You all know it's true"  },  {  "Id": 10,  "PictureUrl": "https://i.redd.it/bhc27wry47661.jpg",  "Title": "Is it a general fiction or just"  },  {  "Id": 11,  "PictureUrl": "https://i.redd.it/9jrwb54y47661.jpg",  "Title": "Is this what it is?"  }  ]  },  "DateOfCreation": "2020-12-20T15:30:25",  "CreatorId": 1,  "Participants": [  {  "Id": 1,  "UserId": "B8RskiAdPETrhUvsOnn8L9SSn9a2",  "User": {  "Id": "B8RskiAdPETrhUvsOnn8L9SSn9a2",  "Nickname": null,  "UsersRoles": [  {  "Id": 1,  "UserId": "B8RskiAdPETrhUvsOnn8L9SSn9a2",  "RoleId": "Administrator"  }  ]  },  "DateOfStart": "2020-12-20T15:30:35",  "DateOfEnd": "2020-12-20T15:31:25",  "QuestionsAndAnswers": [  {  "Id": 1,  "Question": {  "Id": 2,  "PictureUrl": "https://i.redd.it/0f0njh9e57661.jpg",  "Title": "ending 2020 with rickrolls"  },  "Answer": true  },  {  "Id": 2,  "Question": {  "Id": 3,  "PictureUrl": "https://i.redd.it/2bpqbked57661.jpg",  "Title": "I seriously jumped when a body just flew into my face for no reason"  },  "Answer": true  },  {  "Id": 3,  "Question": {  "Id": 4,  "PictureUrl": "https://i.redd.it/thbgju3757661.png",  "Title": "I want both to be stuck in my head"  },  "Answer": true  },  {  "Id": 4,  "Question": {  "Id": 5,  "PictureUrl": "https://i.redd.it/09uoxl8w47661.png",  "Title": "r/whatisthisthing be like"  },  "Answer": true  },  {  "Id": 5,  "Question": {  "Id": 6,  "PictureUrl": "https://i.redd.it/m6wknbi557661.jpg",  "Title": "Restricted Christmas smh"  },  "Answer": true  }  ]  },  {  "Id": 2,  "UserId": "bnaXDmVlygcCWfz2UzlfTdLoXhs1",  "User": {  "Id": "bnaXDmVlygcCWfz2UzlfTdLoXhs1",  "Nickname": "туц тшслтфьу фвывф",  "UsersRoles": [  {  "Id": 2,  "UserId": "bnaXDmVlygcCWfz2UzlfTdLoXhs1",  "RoleId": "User"  }  ]  },  "DateOfStart": "2020-12-20T15:31:45",  "DateOfEnd": "2020-12-20T15:32:25",  "QuestionsAndAnswers": [  {  "Id": 6,  "Question": {  "Id": 7,  "PictureUrl": "https://i.redd.it/1ik6bwi457661.jpg",  "Title": "Restricted Christmas smh"  },  "Answer": true  },  {  "Id": 7,  "Question": {  "Id": 8,  "PictureUrl": "https://i.redd.it/bofsdj1257661.jpg",  "Title": "TABS is a great game"  },  "Answer": true  },  {  "Id": 8,  "Question": {  "Id": 9,  "PictureUrl": "https://i.imgur.com/4Npg00j.jpg",  "Title": "You all know it's true"  },  "Answer": true  },  {  "Id": 9,  "Question": {  "Id": 10,  "PictureUrl": "https://i.redd.it/bhc27wry47661.jpg",  "Title": "Is it a general fiction or just"  },  "Answer": true  },  {  "Id": 10,  "Question": {  "Id": 11,  "PictureUrl": "https://i.redd.it/9jrwb54y47661.jpg",  "Title": "Is this what it is?"  },  "Answer": true  }  ]  }  ]  }  ] |

3.4. Настройка Firebase

Firebase использует аутентификацию по средством JWT токена, настройка middleware для ASP.NET Core приложения происходит в файле Startup в методе ConfigureServices и выглядит так [6]:

services

.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.Authority = "https://securetoken.google.com/calibration-pri117";

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidIssuer = "https://securetoken.google.com/calibration-pri117",

ValidateAudience = true,

ValidAudience = "calibration-pri117",

ValidateLifetime = true

};

3.5. Настройка и реализация Entity Framework Core

Для взаимодействия с базами данных была настроена ORM Entity Framework Core. В конфигурацию были добавлены строки подключения к базам данных, в проекте были созданы контексты баз данных. Конфигурацию и контексты представлены в приложении A-1. В контекстах есть начальная инициализация базы данных, а точнее создается пользователь-админ и несколько категорий опросников [3].

3.6. Реализация контроллеров

Для обмена данными были реализованы REST-контроллеры. Валидация осуществляется по средствам аннотаций в моделях или проверкой в самом контроллере. Пример контроллера для работы с пользователями приведён в приложении А-2 [1][2][5].

3.7. Настройка логгирования

В качество логгера был использован Serilog, его настройка представлена в предложении А-3.

3.8. Структура проекта

В основе структуре проекта лежит Web-API сервис, к нему подключаются библиотеки, которые были написаны и разделены по назначению, к примеру есть библиотека Logic, в которой хранятся классы с бизнес-логикой приложения, есть LogicDB, в которой классы для работы с базой данных, есть Models, в которой модели.

4 ТЕСТИРОВАНИЕ СЕРВЕРНОЙ ЧАСТИ

4.1. Тестирование с помощью SWAGGER

В приложение была подключена библиотека для работы со SWAGGER, с помощью нее были протестированы некоторые запросы к API. Основные результаты тестирования представлены в приложении Б-1.

4.2. Тестирование с помощью xUnit

Для некоторых компонентов системы были написаны интеграционные тесты с помощью фреймворка xUnit, они представлены в приложении Б-2.

****4.3. Тестирование с помощью Insomnia Core****

Insomnia Core – приложение для отладки REST сервисов. Тестирование с использованием Insomnia представлено в приложении Б-3.

5 ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ СЕРВЕРНОЙ ЧАСТИ

Чтобы запустить сервер необходимо выполнить следующие шаги:

1. Скачать СУБД PostgreSQL с официального сайта (<https://www.postgresql.org/>).
2. Установить ранее скаченную СУБД и во время установки указать логин и пароль для доступа к базам данных.
3. Скачать исходный код проекта с удаленного репозитория (<https://github.com/djd1man/BackendForMobileAppCourseProj>)
4. В файле appsettings.json в проекте BackendMobileApp необходимо указать логин и пароль для доступа к базе данных в строках подключения.
5. Собрать и запустить приложение. Самый просто способ — это сделать через Visual Studio, нажав кнопку запуска приложения.

После выполнения этих шагов создастся и заполнится база данных, а также приложение, которое будет доступно по ссылке http://localhost:5000.

ЗАКЛЮЧЕНИЕ

В рамках выполнения данного курсового проекта была разработана серверная часть для мобильного приложения по выявлению совместимости и интересов людей. Была спроектирована база данных, описана спецификация API и протестирована система.

В ходе выполнения курсового проекта была освоена разработка клиент-серверных приложений на ASP.NET Core с СУБД PostgreSQL.

6 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. — СПб.: Питер, 2013. — 896 с.: ил. — (Серия «Мастер-класс»). ISBN 978-5-496-00433-6

2) Стивен Клири. Конкурентность в C# Питер, 2020 год, 220 стр., 2-е изд., ISBN: 978-5-4461-1572-3

3) C# 8.0 in a Nutshell by Joseph Albahari, Eric Johannsen Released May 2020 Publisher(s): O'Reilly Media, Inc. ISBN: 9781492051138

4) ГОСТ 2.105 – 95. Общие требования к текстовым документам. –  [Текст] : Стандартинформ,  2007. – 13 с.

5) Официальная документация .NET Core. [Электронный ресурс] Режим доступа: <https://docs.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-3.1>

6) Официальная документация Firebase. [Электронный ресурс] Режим доступа: https://firebase.google.com/docs

ПРИЛОЖЕНИЕ А  
**ЛИСТИНГ КОМПОНЕНТОВ СИСТЕМЫ**

1. Реализация Entity Framework Core

Строки подключения в конфигурации:

"ConnectionStrings": {

"Data": "Host=localhost;Port=5432;Database=Data;Username=postgres;Password=root",

"Identity": "Host=localhost;Port=5432;Database=Identity;Username=postgres;Password=root"

}

Контекст для работы с базой данных пользователей:

public class ApplicationContext : DbContext

{

public DbSet<User>? Users { get; set; }

public DbSet<Role>? Roles { get; set; }

public DbSet<UserRole>? UsersRoles { get; set; }

public ApplicationContext(DbContextOptions<ApplicationContext> options)

: base(options)

{

//Database.EnsureDeleted();

Database.EnsureCreated();

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<UserRole>()

.HasOne(ur => ur.User)

.WithMany(p => p!.UsersRoles)

.HasForeignKey(pc => pc.UserId);

modelBuilder.Entity<UserRole>()

.HasOne(pc => pc.Role)

.WithMany(c => c!.UsersRoles)

.HasForeignKey(pc => pc.RoleId);

var roles = new[]

{

new Role(){ Name = "User"},

new Role(){ Name = "Moderator"},

new Role(){ Name = "Administrator"}

};

modelBuilder.Entity<Role>().HasData(roles);

var user = new User()

{

Id = "B8RskiAdPETrhUvsOnn8L9SSn9a2"

};

modelBuilder.Entity<User>().HasData(user);

var userRole = new UserRole()

{

Id = 1,

RoleId = "Administrator",

UserId = "B8RskiAdPETrhUvsOnn8L9SSn9a2"

};

modelBuilder.Entity<UserRole>()

.HasData(userRole);

base.OnModelCreating(modelBuilder);

}

}

Контекст для работы с основной базой данных:

public class MyContext : DbContext

{

public DbSet<GameSession>? Sessions { get; set; }

public DbSet<RedditQuizSettings>? RedditSettings { get; set; }

public DbSet<ParticipantOfSession>? ParticipantsOfSessions { get; set; }

public DbSet<Quiz>? Quizs { get; set; }

public DbSet<QuizCategory>? QuizCategories { get; set; }

public DbSet<BaseQuizCategory>? BaseQuizCategories { get; set; }

public DbSet<Question>? Questions { get; set; }

public MyContext(DbContextOptions<MyContext> options)

: base(options)

{

//Database.EnsureDeleted();

Database.EnsureCreated();

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

var redditCategory = new BaseQuizCategory() { Id = 1, Name = "Reddit" };

modelBuilder.Entity<BaseQuizCategory>().HasData(redditCategory);

modelBuilder.Entity<QuizCategory>().HasData(

new { Id = 1L, BaseQuizCategoryId = redditCategory.Id, Name = "pics" },

new { Id = 2L, BaseQuizCategoryId = redditCategory.Id, Name = "memes" });

}

}

1. Реализация контроллера для взаимодействия с пользователями

[Route("api/users")]

[ApiController]

public class UsersController : ControllerBase

{

public UsersController(

ILogger<UsersController>? logger,

ApplicationContext appContext)

{

\_logger = logger;

\_appContext = appContext

?? throw new ArgumentNullException(nameof(appContext));

\_logger?.LogInformation("The instance is created");

}

[Authorize(Roles = "User,Moderator,Administrator")]

[Route("changenick")]

[HttpPost]

public IActionResult ChangeNick(NewNickname newNickname)

{

if (newNickname.Value is null)

{

return Problem("Nickname is null");

}

if (string.IsNullOrWhiteSpace(newNickname.Value))

{

return Problem("Nickname is empty");

}

var userId = User.Claims.First(x => x.Type == "user\_id").Value;

var user = \_appContext.Users!.Find(userId);

user.Nickname = newNickname.Value;

\_appContext.Users!.Update(user);

\_appContext.SaveChanges();

return Ok();

}

[Authorize(Roles = "Administrator")]

[Route("changeusernick")]

[HttpPost]

public IActionResult ChangeNickCurrentUser(ChangeUserNick changeUserNick)

{

if (changeUserNick.NewNickname is null)

{

return Problem("Nickname is null");

}

if (string.IsNullOrWhiteSpace(changeUserNick.NewNickname))

{

return Problem("Nickname is empty");

}

var user = \_appContext.Users!.Find(changeUserNick.UserId);

if (user is null)

{

return Problem("Unknown user");

}

user.Nickname = changeUserNick.NewNickname;

\_appContext.Users!.Update(user);

\_appContext.SaveChanges();

return Ok();

}

[Authorize(Roles = "Administrator")]

[Route("getusers")]

[HttpGet]

public string GetUsers()

{

var users = \_appContext.Users!.ToList();

var usersToReturn = users.Select(u => new { UserId = u.Id, Nickname = u.Nickname});

// return JsonConvert.SerializeObject(usersToReturn);

return JsonConvert.SerializeObject(users);

}

[Authorize(Roles = "User,Moderator,Administrator")]

[Route("getprofile")]

[HttpGet]

public string GetProfile()

{

var userId = User.Claims.First(x => x.Type == "user\_id").Value;

return JsonConvert.SerializeObject(\_appContext.Users!.Find(userId));

}

private readonly ILogger<UsersController>? \_logger;

private readonly ApplicationContext \_appContext;

}

1. Настройка логгирования

Конфигурация:

"Serilog": {

"Using": [ "Serilog.Sinks.Console", "Serilog.Sinks.File" ],

"MinimumLevel": {

"Default": "Information",

"Override": {

"Microsoft": "Warning",

"System": "Warning"

}

},

"WriteTo": [

{

"Name": "Console",

"Args": {

"outputTemplate": "{Timestamp:o} [{Level}] ({Application}/{MachineName}/{ThreadId}) {SourceContext:l} {Scope} {Message}{NewLine}{Exception}"

}

},

{

"Name": "File",

"Args": {

"path": "serilog-basic-.log",

"rollingInterval": "Day",

"outputTemplate": "{Timestamp:o} [{Level}] ({Application}/{MachineName}/{ThreadId}) {SourceContext:l} {Scope} {Message}{NewLine}{Exception}"

}

}

],

"Enrich": [ "FromLogContext", "WithMachineName", "WithThreadId" ]

}

Подключение в Program.cs

public class Program

{

public static async Task Main(string[] args)

{

await CreateHostBuilder(args).Build().RunAsync().ConfigureAwait(false);

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>().UseSerilog(

(hostingContext, loggerConfiguration) => loggerConfiguration.ReadFrom.Configuration(hostingContext.Configuration));

});

}

ПРИЛОЖЕНИЕ Б

**1. Тестирование с помощью SWAGGER**



Рисунок Б.1 - Тестирования получения категорий с помощью SWAGGER



Рисунок Б.2 - Тестирования получения опроса по айди с помощью SWAGGER

**2. Тестирование с помощью xUnit**

public class RedditPostsGetterTests

{

[Fact(DisplayName = nameof(RedditPostsGetter) + " can get posts and they will be different with different params.")]

[Trait("Category", "Integration")]

public async Task CanGetPosts()

{

var httpRequestSender = new HttpRequestSender();

var postsGetter = new RedditPostsGetter(httpRequestSender);

var posts1 = await postsGetter.GetPostsAsync(

"memes",

Reddit.RedditFilterType.Top,

Reddit.RedditTopFilter.AllTime,

CancellationToken.None).ConfigureAwait(false);

var posts2 = await postsGetter.GetPostsAsync(

"memes",

Reddit.RedditFilterType.Top,

Reddit.RedditTopFilter.AllTime,

CancellationToken.None).ConfigureAwait(false);

posts1.Should().BeEquivalentTo(posts2);

posts1.Count.Should().Be(70);

var posts3 = await postsGetter.GetPostsAsync(

"memes",

Reddit.RedditFilterType.Top,

Reddit.RedditTopFilter.ThisMonth,

CancellationToken.None).ConfigureAwait(false);

var posts4 = await postsGetter.GetPostsAsync(

"memes",

Reddit.RedditFilterType.Top,

Reddit.RedditTopFilter.ThisMonth,

CancellationToken.None).ConfigureAwait(false);

posts3.Should().BeEquivalentTo(posts4);

posts3.Count.Should().Be(50);

var posts5 = await postsGetter.GetPostsAsync(

"memes",

Reddit.RedditFilterType.Top,

Reddit.RedditTopFilter.ThisMonth,

CancellationToken.None).ConfigureAwait(false);

var posts6 = await postsGetter.GetPostsAsync(

"memes",

Reddit.RedditFilterType.Top,

Reddit.RedditTopFilter.Today,

CancellationToken.None).ConfigureAwait(false);

posts5.Should().NotBeEquivalentTo(posts6);

var posts7 = await postsGetter.GetPostsAsync(

"pics",

Reddit.RedditFilterType.Hot,

null,

CancellationToken.None).ConfigureAwait(false);

var posts8 = await postsGetter.GetPostsAsync(

"pics",

Reddit.RedditFilterType.Hot,

null,

CancellationToken.None).ConfigureAwait(false);

posts7.Should().BeEquivalentTo(posts8);

posts7.Count.Should().Be(70);

var posts9 = await postsGetter.GetPostsAsync(

"pics",

Reddit.RedditFilterType.Hot,

null,

CancellationToken.None).ConfigureAwait(false);

var posts10 = await postsGetter.GetPostsAsync(

"pics",

Reddit.RedditFilterType.Hot,

null,

CancellationToken.None).ConfigureAwait(false);

posts9.Should().NotBeEquivalentTo(posts10);

}

[Fact(DisplayName = nameof(RedditPostsGetter) + " cannot get posts if subreddit doenst contains posts with pictures.")]

[Trait("Category", "Integration")]

public async Task CannotGetPosts()

{

var httpRequestSender = new HttpRequestSender();

var postsGetter = new RedditPostsGetter(httpRequestSender);

var posts1 = await postsGetter.GetPostsAsync(

"AskReddit",

Reddit.RedditFilterType.Top,

Reddit.RedditTopFilter.AllTime,

CancellationToken.None).ConfigureAwait(false);

posts1.Count.Should().Be(0);

}

}

**3. Тестирование с помощью Insomnia Core**

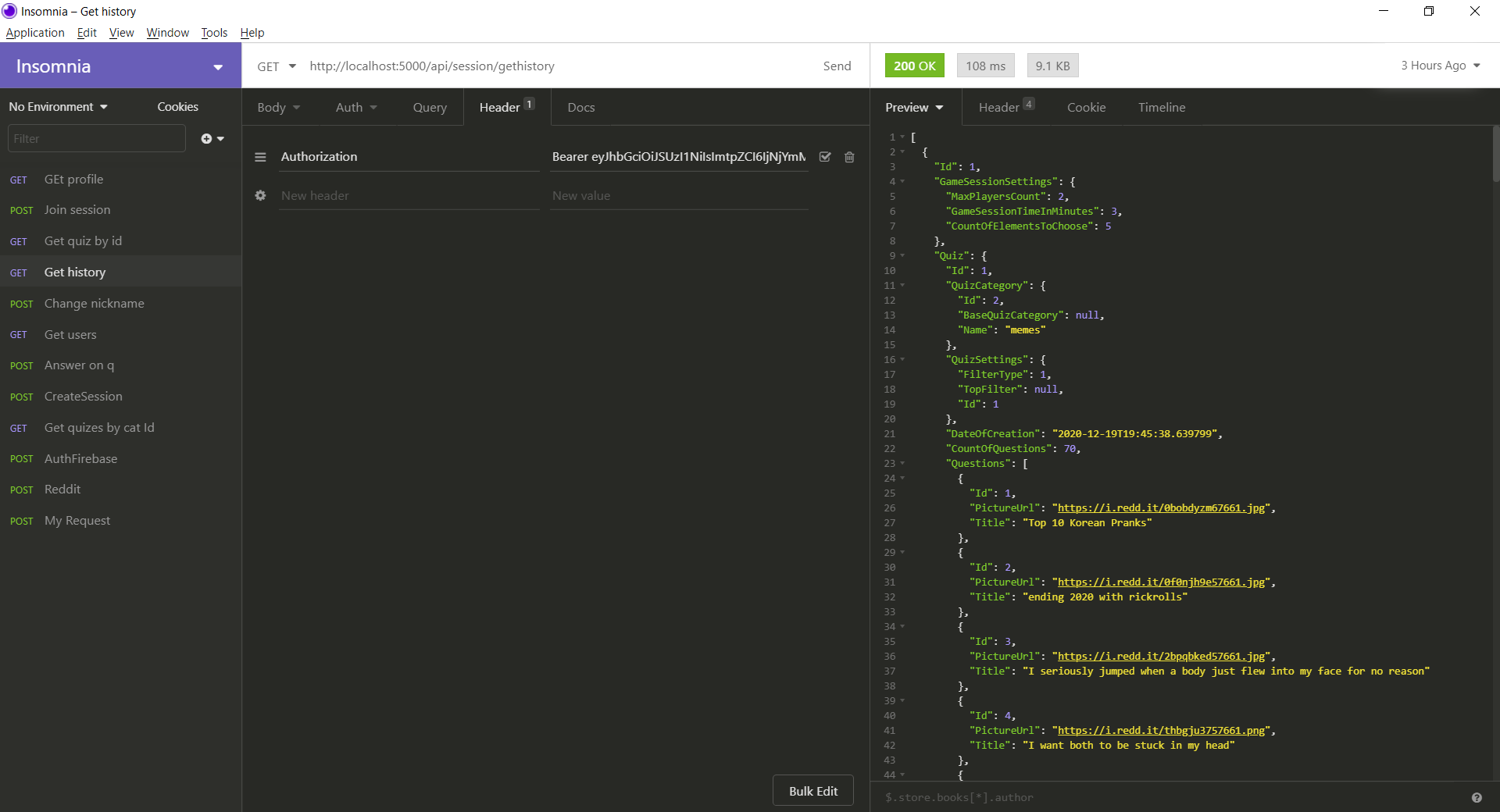


Рисунок Б.3 – Тестирование получения истории с помощью Insomnia Core

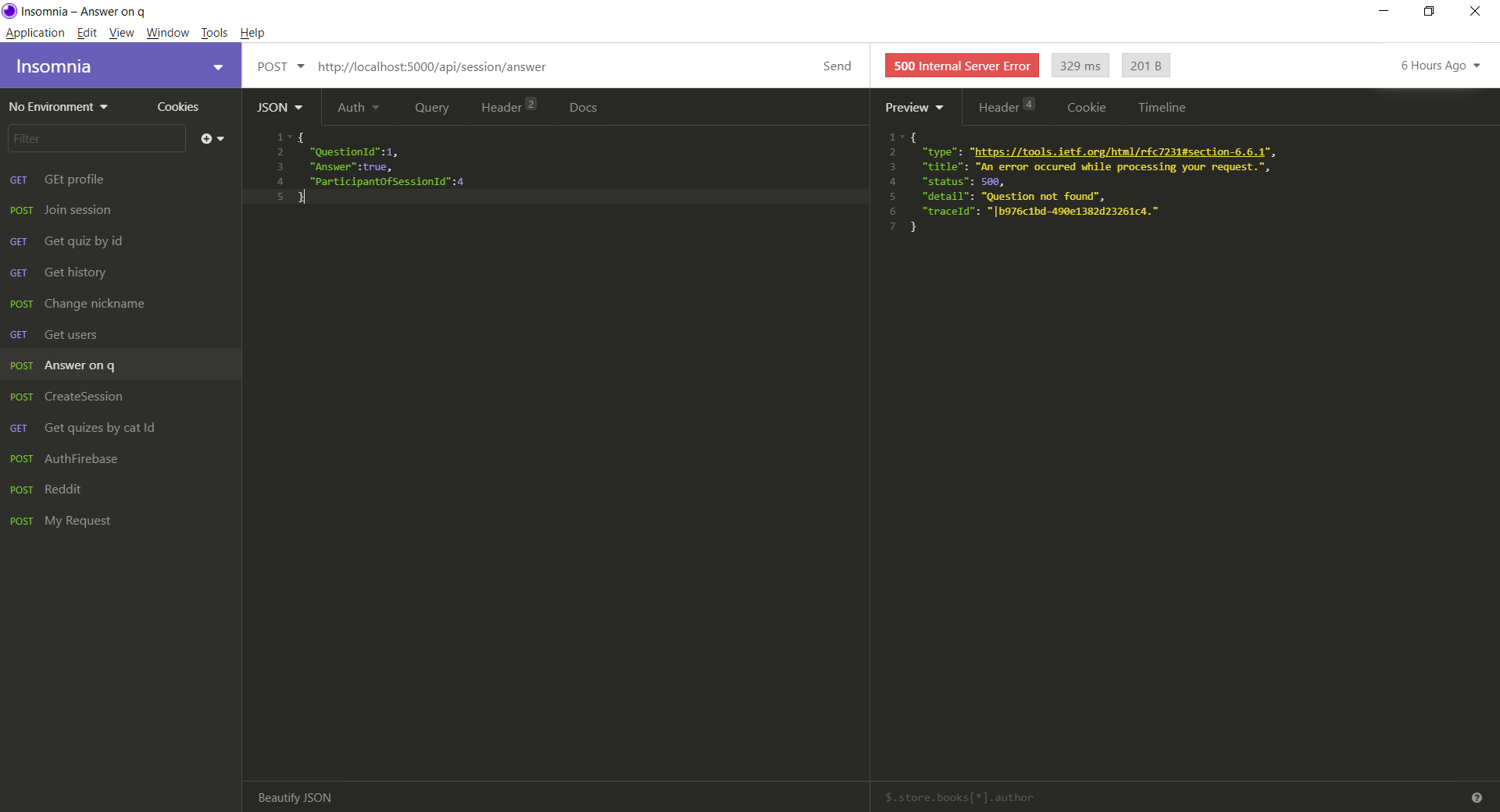


Рисунок Б.4 - Тестирование ответа на вопрос с помощью Insomnia Core