МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40 05 01 03 «Информационные системы и технологии»

Специализация «Издательско-полиграфический комплекс»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ**

по дисциплине «Программирование в Internet»

Тема: Web-приложение «МедТестЧек»

Исполнитель

|  |  |  |
| --- | --- | --- |
| cтудент 4 курса группы 1  Руководитель |  | \_\_\_\_\_\_\_\_\_\_\_\_ Д. C. Шкабров  подпись, дата |
| ассистент |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. С. Кантарович  подпись, дата |
| Курсовой проект защищен с оценкой |  | \_\_\_\_\_\_\_\_\_\_\_\_ |
| Руководитель \_\_\_\_\_\_\_\_\_\_ |  | В. С. Кантарович |

подпись

Минск 2023

Оглавление

[Введение 5](file:///C:\Users\petra\Downloads\Telegram%20Desktop\Пояснительная%20записка%20Пирейко%20(2).docx#_Toc149479588)

[1 Постановка задач и обзор аналогов 6](file:///C:\Users\petra\Downloads\Telegram%20Desktop\Пояснительная%20записка%20Пирейко%20(2).docx#_Toc149479589)

[1.1 Постановка задач 6](file:///C:\Users\petra\Downloads\Telegram%20Desktop\Пояснительная%20записка%20Пирейко%20(2).docx#_Toc149479590)

[1.2 Обзор аналогов 6](file:///C:\Users\petra\Downloads\Telegram%20Desktop\Пояснительная%20записка%20Пирейко%20(2).docx#_Toc149479591)

[1.2.1 Веб-приложение «Kahoot!» 6](file:///C:\Users\petra\Downloads\Telegram%20Desktop\Пояснительная%20записка%20Пирейко%20(2).docx#_Toc149479592)

[1.2.2 Веб-приложение «Quizizz» 7](file:///C:\Users\petra\Downloads\Telegram%20Desktop\Пояснительная%20записка%20Пирейко%20(2).docx#_Toc149479593)

[1.3 Вывод по разделу 9](file:///C:\Users\petra\Downloads\Telegram%20Desktop\Пояснительная%20записка%20Пирейко%20(2).docx#_Toc149479594)

**Введение**

В современной медицине постоянно появляются новые открытия и возможности, поэтому очень важно идти в ногу со временем и для этого надо постоянно обновлять свои знания. Тесты помогают восполнить пробелы и повысить уровень знаний, в соответсвии с современенностью.

Схемы автоматизации помогают управлять системой тестов и карточек, тем самым проверять знания у студентов-медиков, действующих врачей и даже тех, у кого был большой перерыв в работе.

Поэтому для составляния тестов, важно учесть базовые знания и новые тенденции, появляющиеся в медицине. Так пользователи приложения вспомнят то, что учили и узнают о своих пробелах.

Задачи таких тестов заинтересовать людей восполнять пробелы в своих знаниях и стремиться узнавать новое.

1. **Постановка задач и обзор аналогов**
   1. **Постановка задач**

Целью данного курсового проекта является создание веб-приложения для проверки знаний в виде тестов и других вариантов. Для реализации проекта необходимо установить список задач, выполнение которых будет отображать прогресс:

* обеспечивать реализацию 3 ролей: администратора, преподавателя и студента;
* обеспечивать регистрацию и авторизацию пользователей;
* обеспечивать возможность администратору управлять пользователями и предметами;
* обеспечивать преподавателю управление тестами: добавление, удаление, редактирование тестов;
* обеспечивать преподавателю и студенту результаты тестов;
* обеспечивать студенту выбор предмета, а затем выбор вида теста;
* обеспечивать студенту возможность создания или удаления теста по темам;
* обеспечивать возможность студенту видеть таймер во время ответов на вопросы;
  1. **Обзор аналогов**
     1. **Веб-приложение «Kahoot!»**

Kahoot! — игровая обучающая платформа, используемая в школах и других учебных заведениях. На сайте предоставлен каталог игр — «Kahoots» — каждая из которых является викториной, содержащей вопросы с несколькими вариантами ответов. Kahoot! был основан в 2012 сообща студентами и преподавателями Норвежского университета естественных и технических наук.

Kahoot! был разработан для групповых занятий. Учащиеся собираются вокруг общего экрана — например, интерактивной доски, проектора, монитора или экрана, передаваемого виртуально, например, через VNC, Jitsi, Skype или Google Hangouts. Игровой процесс простой: все игроки одновременно отвечают на вопросы на своих устройствах. Вопросы выводятся ученикам на экран по одному. Участники набирают очки за каждый правильный ответ. В конце викторины на экран выводится количество очков всех участников, набранных во время ответа на её вопросы.

Для участия в игре не требуется обязательной регистрации. Викторины можно выбрать из каталога на сайте. Также можно создавать новые викторины — часто эту задачу выполняет ведущий урок, например, учитель. Время ответа на каждый вопрос ограничено примерно 30-60 секундами.

Главная страница сайта «Kahoot!» представлена на рисунке 1.1.

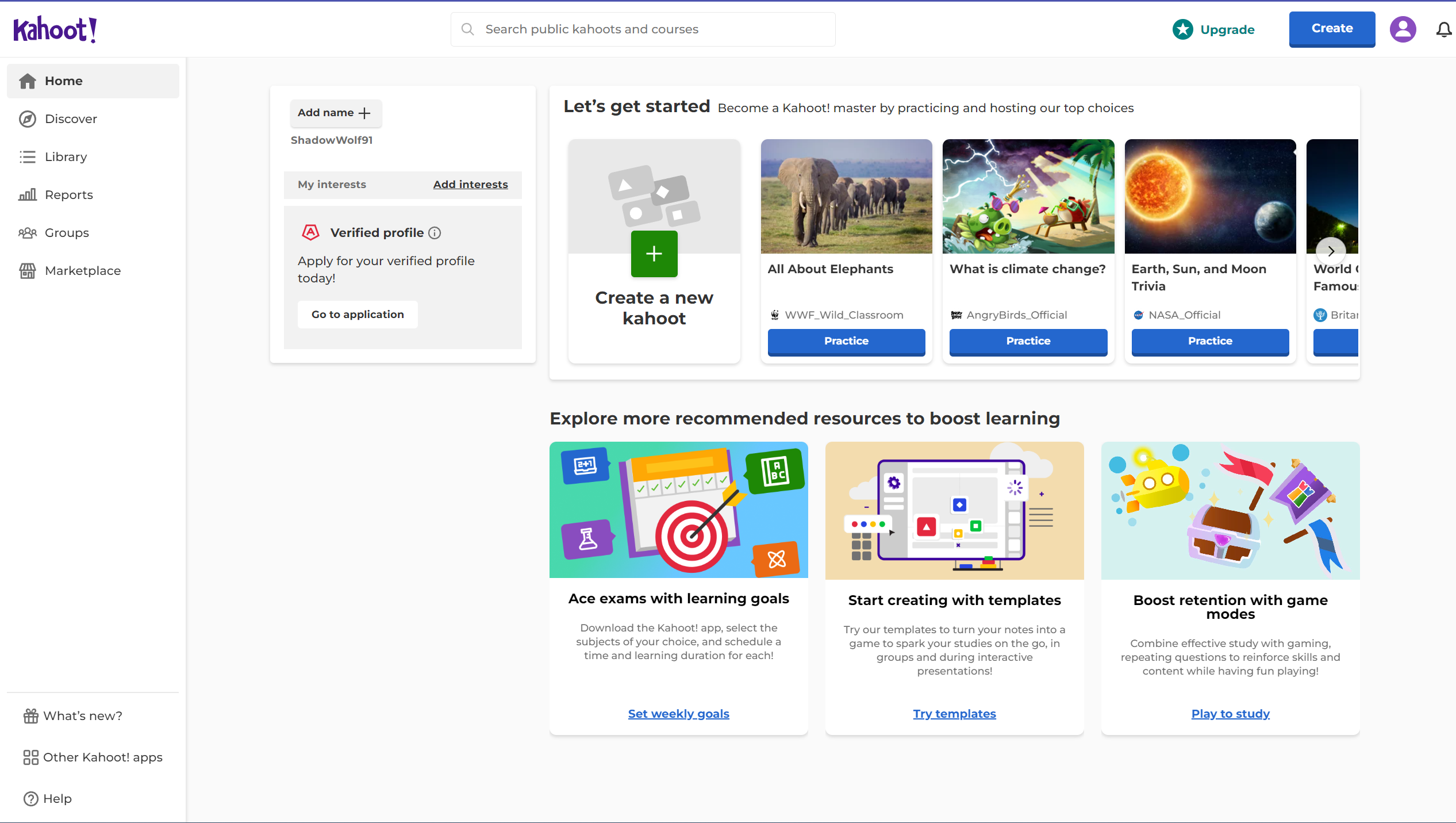


Рисунок 1.1 – Интерфейс «Kahoot!»

Страница с видом теста «Kahoot!» представлена на рисунке 1.2.

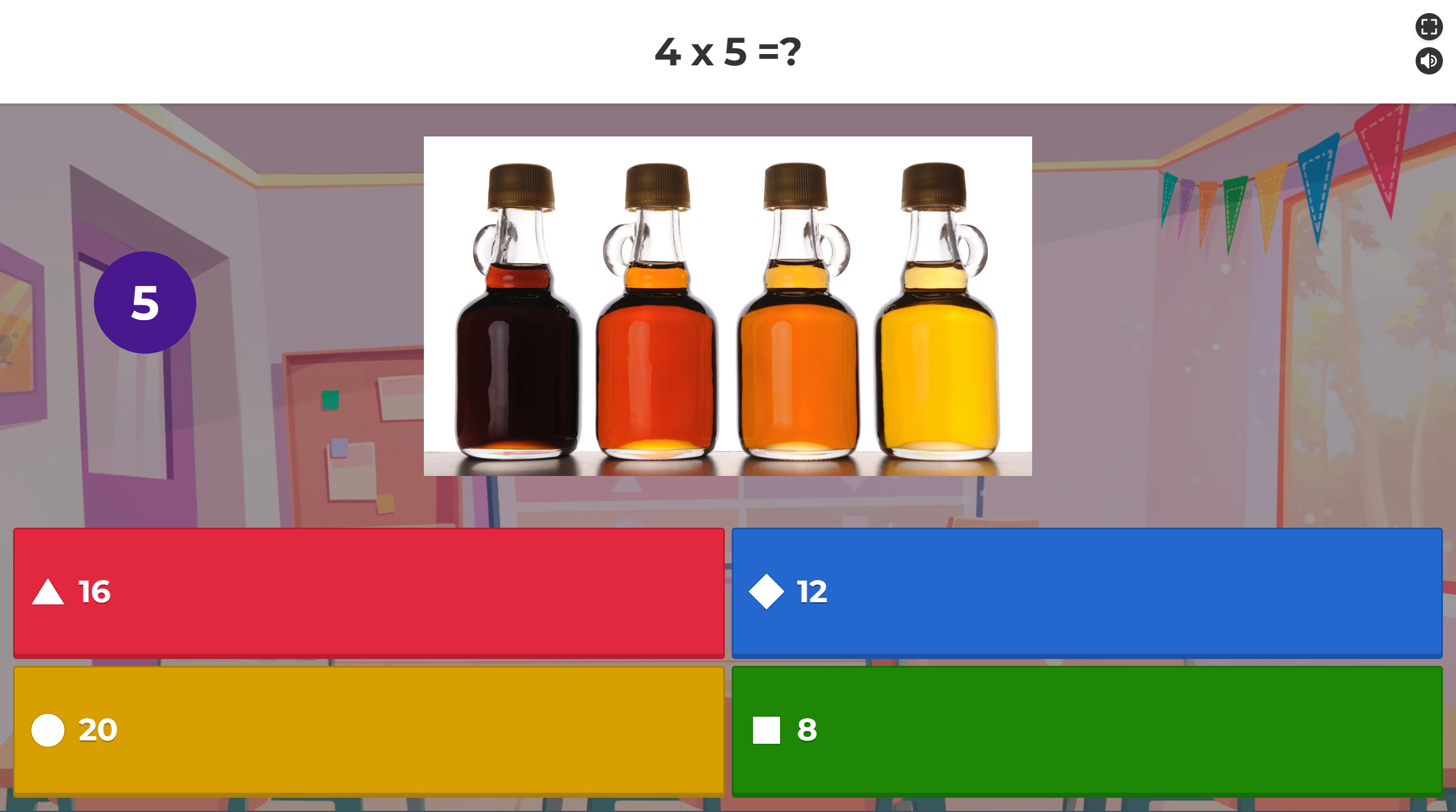


Рисунок 1.2 – Решение теста в «Kahoot!»

* + 1. **Веб-приложение «Quizizz»**

Quizizz — это игровая платформа для взаимодействия со студентами, предлагающая множество функций, которые сделают занятия в классе веселыми, интерактивными и увлекательными.

Как учитель, вы можете проводить формативное оценивание, задавать домашние задания и осуществлять другие увлекательные взаимодействия со своими учениками (для всех классов).

Возможности викторины:

* Уроки/викторины, проводимые инструктором: Учителя контролируют темп; весь класс вместе просматривает каждый вопрос.
* Уроки/тесты, ориентированные на учащихся: учащиеся продвигаются в своем темпе, и вы видите таблицу лидеров и живые результаты по каждому вопросу или уроку.
* Доступ к миллионам викторин: импортируйте любые общедоступные викторины как есть, редактируйте и настраивайте их.
* (Викторина + урок) Редактор: выберите один из 5 различных типов вопросов, чтобы добавлять изображения, видео и аудио к телепортируемым вопросам из других викторин и уроков.
* Отчеты: получайте подробную информацию на уровне класса и учащихся по каждому тесту.

Главная страница сайта «Quizizz» представлена на рисунке 1.3.

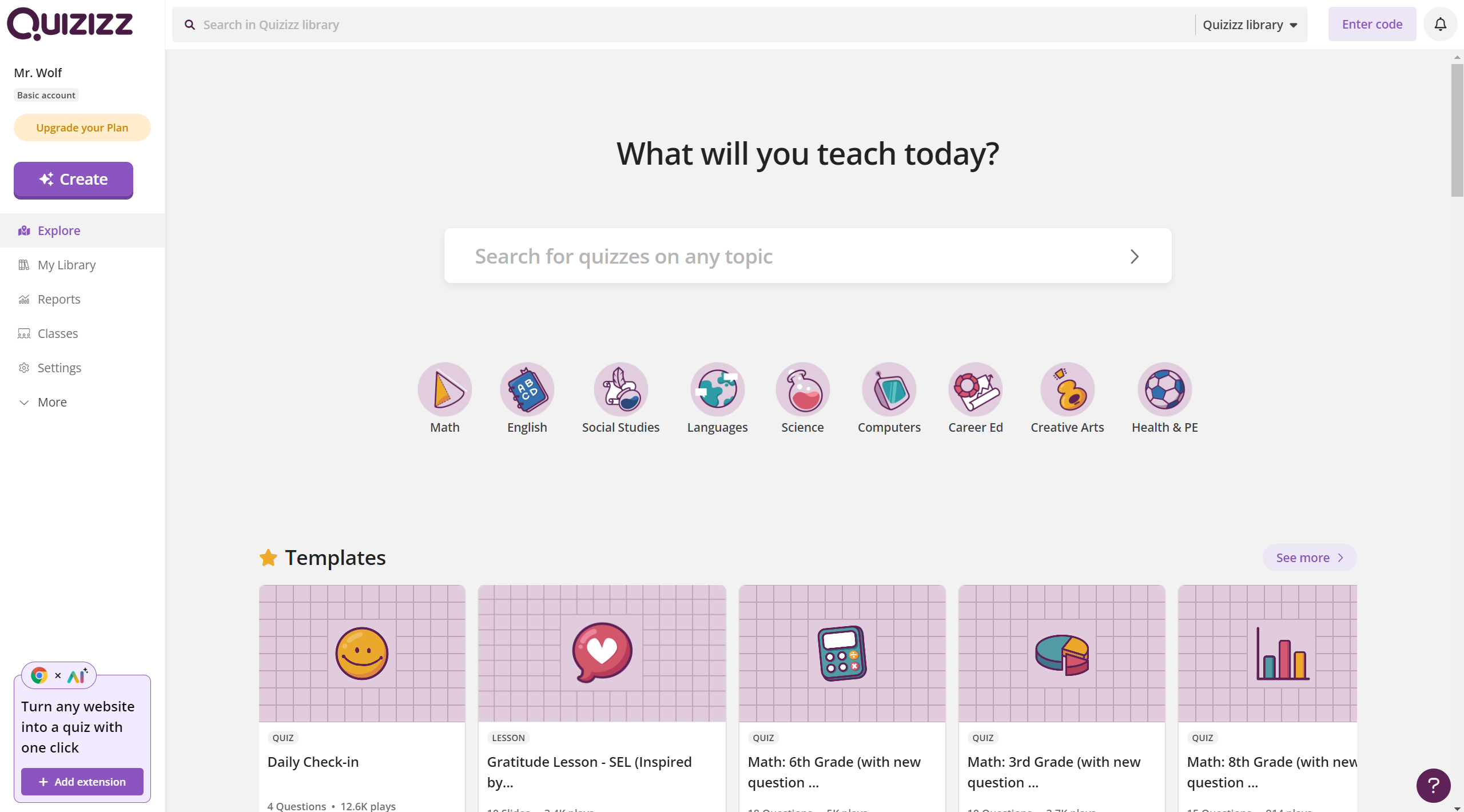


Рисунок 1.3 – Главная страница «Quizizz»

Страница сайта перед выполнением теста «Quizizz» представлена на рисунке 1.4.

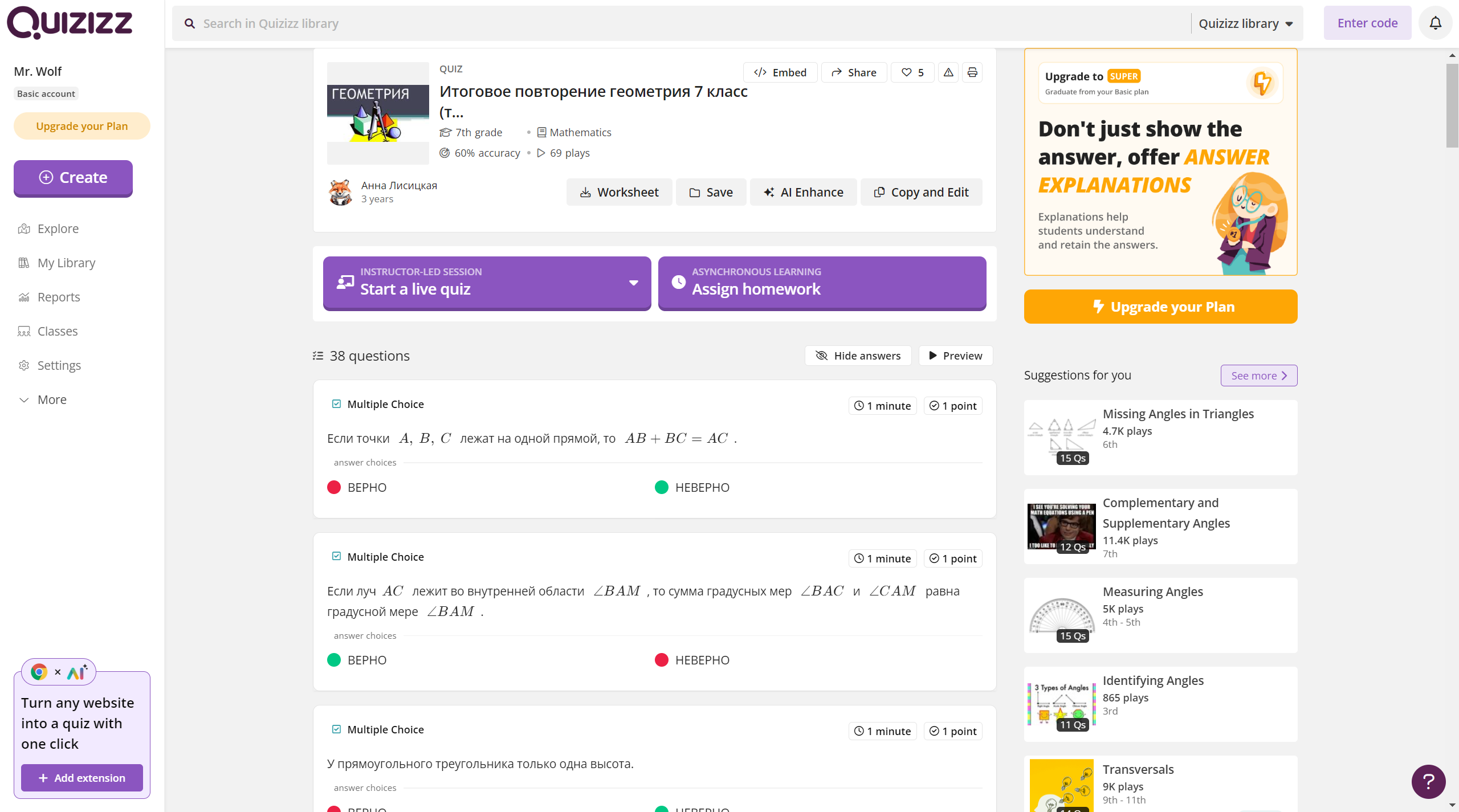


Рисунок 1.4 – Страница сайта с ответами перед выполнением теста «Quizizz»

* 1. **Вывод по разделу**

В ходе анализа существующих аналогов, была выявлена целесообразность разработки и определен основной спектр функциональных возможностей, которые необходимо реализовать при разработке данного программного продукта. Также, на основании использования приложений-аналогов были выявлены основные критерии для построения дизайна приложения, позволяющего осуществлять наиболее простое и интуитивное его использование.

**2 Проектирование программного средства**

После того как основной функционал был определен, можно начинать непосредственно проектирование приложения. Проектированию подлежит база данных приложения, а также UML-диаграмма и блок-схема алгоритмов работы.

**2.1 Разработка модели базы данных**

Как отмечалось в предыдущих разделах, одним из ключевых моментов при проектировании и создании базы данных является грамотный анализ предметной области приложения. Как следствие – составление такой модели данных, которая будет правильно отражать то, как с этими с данными в общем, и этой моделью, в частности, подразумевается взаимодействовать.

Результатом корректно проведённого анализа, проектирования, и разработки, является модель, способная предоставить функционально все необходимые возможности для пользователя. В частности, для приложения по организации обучающих курсов, можно выделить набор таких операций, как:

* возможность администратору управлять пользователями и предметами;
* возможность преподавателю управлять тестами: добавление, удаление, редактирование тестов;
* возможность студенту выбор предмета, а затем выбор вида теста;
* возможность студенту создания или удаления теста по темам;

Основой инфраструктуры базы данных является грамотно спроектированная модель, которая отображает связь пользовательских таблиц. Правильное и корректное взаимодействие их друг с другом как раз и заключается в схеме базы данных со связями, верно отображающими их положение.

В базе данных реализовано семь таблиц: «*User*», «*Statistics*», «*Subjects*», «*Themes*», «*Tests*», «*OpenQuestions*», «*Cards*». Между таблицами также настроены связи, отражающие их взаимосвязь.

Таблица «*User*» предназначена для хранения данных после прохождения регистрации пользователем. Именно из этой таблицы берутся данные для разграничения ролей и в зависимости от этого, от возможностей, выполняемых в приложении для разных пользователей.

Таблица «*Statistics*» предназначена для хранения данных о статистике пользователя (отметка, набранные баллы, количество правильно отвеченных)

Таблица «*Subjects*» предназначена для хранения информации об учебном предмете: название учебного предмета.

Таблица «*Themes*» предназначена для хранения информации о темах учебного предмета, как название темы, выбор режима прохождения

Таблица «*Tests*» предназначена для хранения данных о тесте и там содержиться такая информация как: вопрос, ответы (A, B, C, D), правильный ответ, какой теме они принадлежат.

Таблица «*OpenQuestions*» предназначена для хранения данных об открытом вопросе и там содержиться такая информация как: вопрос, ответ, правильный ответ, какой теме они принадлежат.

Таблица «*Cards*» предназначена для хранения данных о карточке и там содержиться такая информация как: вопрос, ответ, правильный ответ, какой теме они принадлежат.

Общая структура базы данных представлена на рисунке 2.1 ниже.

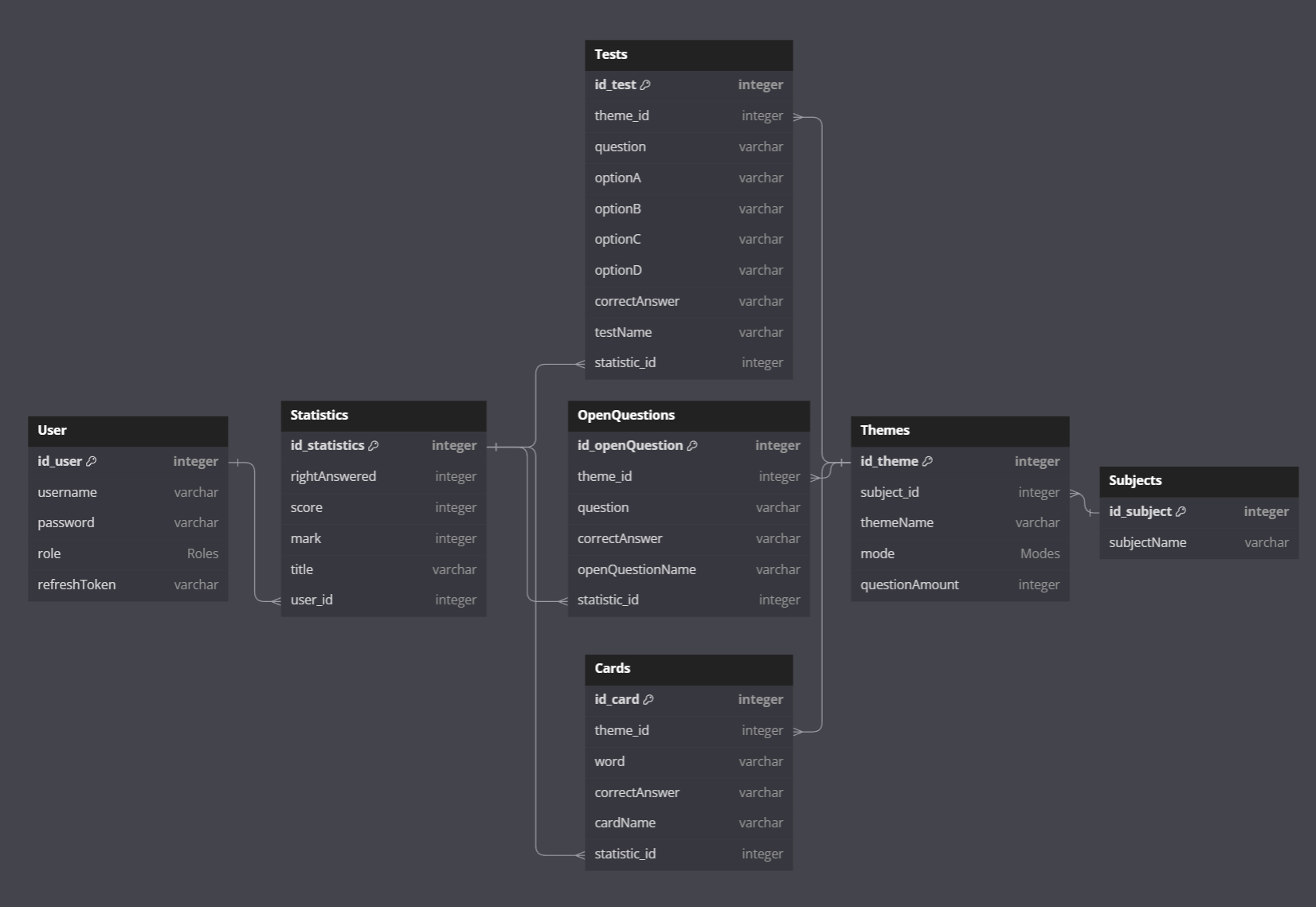


Рисунок 2.1 – Структура базы данных

В качестве примера, того, на чём основывались решения построения тех или иных связей, рассмотрим связи сущностей «*Tests*»/«*OpenQuestions*»/«*Cards*» и «*Statistics*», а также «*Tests*»/«*OpenQuestions*»/«*Cards*» и «*Themes*».

«*Themes*» и «*Tests*»/«*OpenQuestions*»/«*Cards*» являются центральными понятиями в проектируемой системе, так как в совокупности они представляют собой, в некотором смысле, информационное ядро системы, отражающее её основное предназначение – управление видами закрепления знаний и создания для них определенных типов. Связь между этими сущностями определена как «один ко многим». Это объясняется тем, что один тип курсов может содержать некоторое множество конкретных курсов, например, по разным направлениям математики.

Связь между сущностями «*Statistics*» и«*Tests*»/«*OpenQuestions*»/«*Cards*» отражает факт участия в решениях тестов/вопросов/карточек студентов – один студент может как проходить несколько тестов/вопросов/карточек. Соответственно, связь между ними должна быть отражена отношением «один ко многим».

Далее будут расписана каждая таблица для общего понимания, какие данные хранятся в них. Таблица «*Users*» содержит следующие поля:

- *id\_user* – идентификационный номер пользователя;

- *username* – логин пользователя для авторизации в приложении;

- *password* – хешированный пароль пользователя;

- *role* – тип пользователя;

- *refreshToken* – токен пользователя;

Таблица «*Statistics*» основная информация о статистиках пользователей:

- *id\_statistics* – идентификационный номер статистики пользователя;

- *rightAnswered* – количество правильно отвеченных;

- *score* – баллы за ответы;

- *mark* – оценка за ответы;

- *title* - заголовок;

- *user\_id* - идентификационный номер пользователя;

Таблица «*Subjects*» содержит в себе конкретные курсы:

- *id\_subject* – идентификационный номер предмета;

- *subjectName* – название предмета;

Таблица «*Themes*» включает в себя информацию об учителях:

- *id\_theme* – идентификационный номер темы;

- *subject\_id* – идентификационный номер предмета;

- *themeName* – название темы;

- *mode* – режим проверки знаний;

- *questionAmount* – количество вопросов.

Таблица «*Tests*» нужна для получения информации о предметах:

- *id\_test* – идентификационный номер теста;

- *theme\_id* – идентификационный номер темы;

- *question* – вопрос;

- *optionA* – ответ A;

- *optionB* – ответ B;

- *optionC* – ответ C;

- *optionD* – ответ D;

- *correctAnswer* – правильный ответ;

- *testName* – название теста;

- *statistic\_id* – идентификационный номер статистики пользователя.

Таблица «*OpenQuestions*» содержит информацию о руководителях:

- *id\_openQuestion* – идентификационный номер руководителя;

- *theme\_id* – идентификационный номер темы;

- *question* – вопрос;

- *correctAnswer* - правильный ответ;

- *openQuestionName* - название открытого вопроса;

- *statistic\_id* - идентификационный номер статистики пользователя.

Таблицы «*Cards*» содержит с себе данные о студентах:

- *id\_card* – идентификационный номер студента;

- *theme\_id* – идентификационный номер темы;

- *word* - слово;

- *correctAnswer* - правильный ответ;

- *cardName* – название карточки;

- *statistic\_id* - идентификационный номер статистики пользователя.

Остальные примеры связей между сущностями являются довольно тривиальными в сравнении с вышеуказанными, и логика их проектирования не должна вызывать вопросов, исходя только из семантического отражения любой из сущностей.

На рисунке 2.1 проиллюстрированы следующие связи:

* связь «*User-Statistics*»: один ко многим;
* связь «*Statistics-Tests*»: один ко многим;
* связь «*Statistics-OpenQuestions*»: один ко многим;
* связь «*Statistics-Cards*»: один ко многим;
* связь «*Subjects-Themes*»: один ко многим.
* связь «*Themes-Tests*»: один ко многим;
* связь «*Themes-OpenQuestions*»: один ко многим;
* связь «*Themes-Cards*»: один ко многим.

**2.2 Проектирование UML-диаграммы**

При разработке программного обеспечения производится определение схем взаимодействия для представления, какие воздействия на систему смогут оказывать разные пользователи. Схемы взаимодействия — это описание возможного поведения пользователя и определение того, как система будет реагировать на его поведение и приспосабливаться к нему. И для этого используются UML-диаграммы.

Основная задача приложения — представлять собой единое средство, дающее возможность конечному пользователю использовать всю функциональность приложения и отслеживать поведение системы.

Данный этап является одним из самых значимых по время проектирования, так как именно тут решается, кто и какими ролями будет обладать в данной системе.

В данном приложении существует три роли: администратор, преподователь и студент. Но перед тем, как быть в роли, пользователям нужно пройти регистрацию и в данном случаем диаграмма будет иметь вид, как показано на рисунке 2.2.

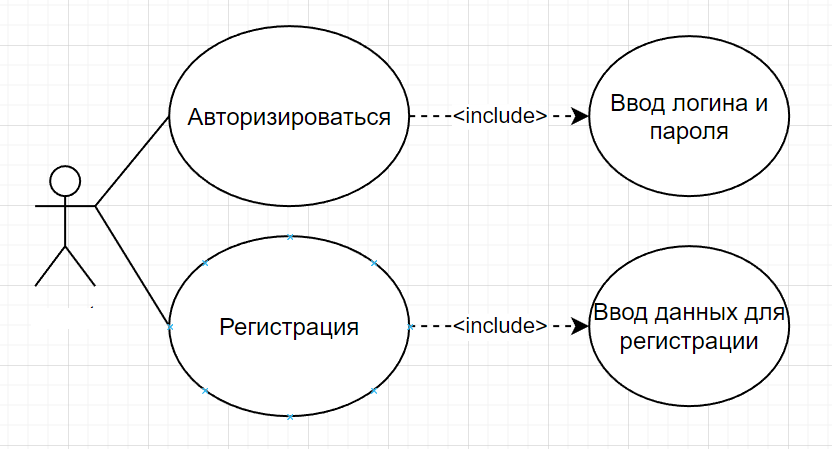


Рисунок 2.2 – UML-диаграмма регистрации и авторизации

В данной диаграмме показывается, что пользователь, который не вошел в систему должен пройти регистрацию и авторизацию и после этого будет доступен функционал приложения.

После регистрации или авторизации пользователь переходит в роль либо студента, либо преподователя, либо администратора, которым доступны функционалы, как показано на рисунке 2.3

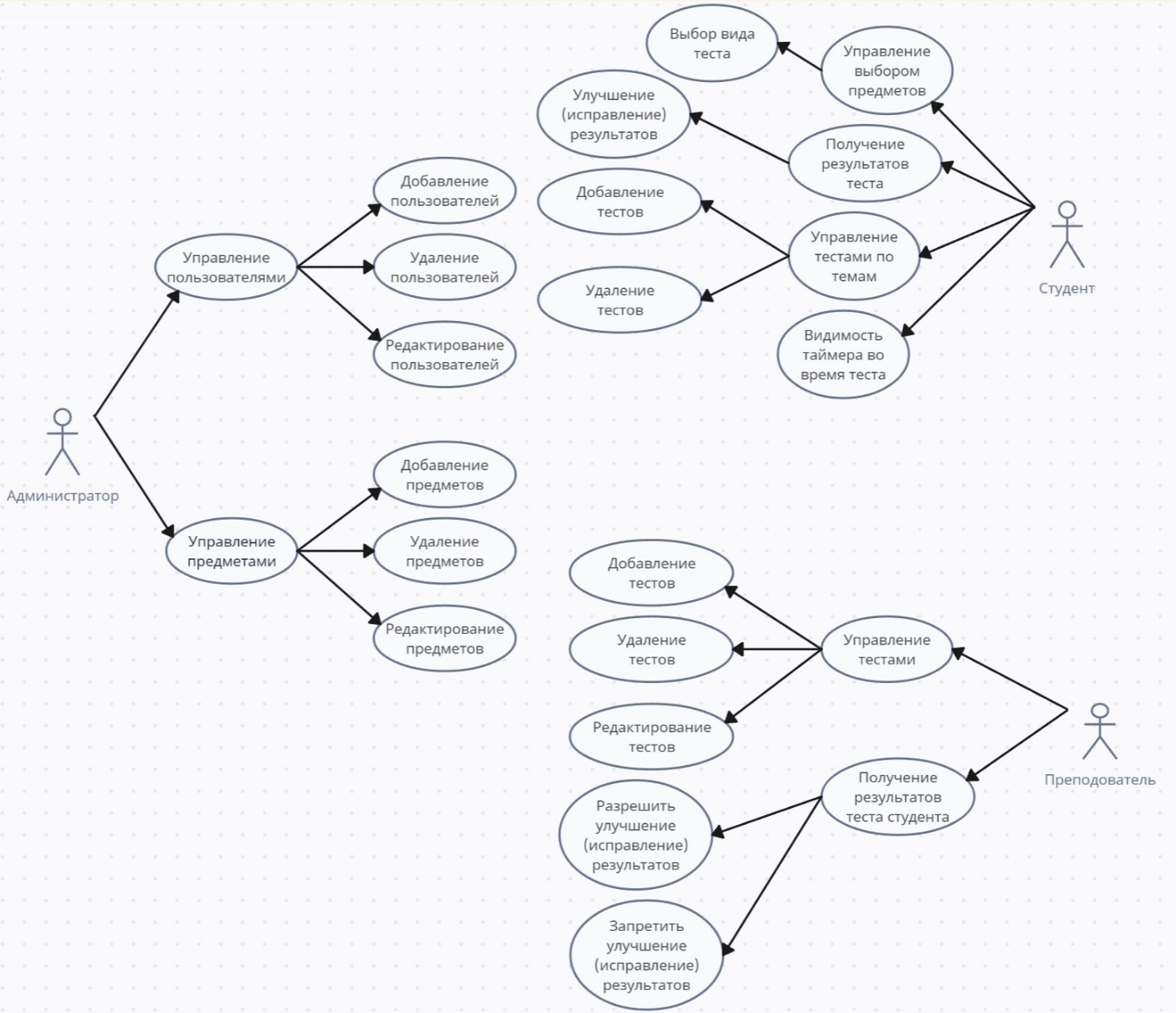


Рисунок 2.3 – UML-диаграмма всех пользователей

Для начала рассмотрим пользователя студента. Он может выполнять выбор уже из готового контента и с ним взатмодействовать.

Как видно из диаграммы преподователь может выполнять множество функций в приложении по созданию разных видов проверки знаний, так же ему доступно и редактирование данных, и их удаление.

Если пользователь в системе авторизируются под особыми правами, детали чего мы опустим, то пользователь будет в роли администратора, которому доступно еще больше функционала.

Как видно, администратор наделен наибольшими правами по сравнению с другими пользователями системы, он может создавать все предметы с темами приложения, изменять их и в случае, если необходимо, то и удалять.

Каждая роль имеет своих участников, которые должны мочь выполнять только то, что предусмотрено для них системой, любое нарушение пределов данных и оговоренных в этом приложении ролей может привести к непредвиденным последствиям, что означает важность правильной реализации данный ролей. Реализация, таким образом, должна предусматривать, что каждый пользователь может совершать определенные действия, согласно его роли. Если пользователь пытается сделать что-то, что находиться вне его полномочий, система должна пресечь его действия, иначе они пагубно отразятся на системе.

**2.3 Проектирование алгоритмов**

Важным этапом при взаимодействии с системой является регистрация пользователя, так как далее происходит процесс авторизации, который позволит использовать весь функционал приложения.

**2.4 Выбор инструментов разработки**

Следующим логическим шагом следует выбор инструментов для разработки.

В качестве языка программирования для создания веб-приложения и интерфейса будет использоваться язык разработки TypeScript, также программное средство должно взаимодействовать с базой данных.

В ходе работы, вся логика, которая предусмотрена тематикой приложения, должна быть реализована на средствах, поставляемых выбранной СУБД и её стандартом языка SQL. В качестве основных инструментов, была выбрана СУБД PostgreSQL среда разработки для неё Prisma. Разработка самого веб-приложения осуществлялась на платформе node.js посредством модели MRCS.

PostgreSQL — свободная объектно-реляционная система управления базами данных. Существует в реализациях для множества UNIX-подобных платформ. Функции являются блоками кода, исполняемыми на сервере, а не на клиенте БД.

ASP.NET Core — свободно-распространяемый кроссплатформенный фреймворк для создания веб-приложений на платформе .NET с открытым исходным кодом.

MRCS — cтруктура каталогов Модель-Маршруты-Контроллеры-Сервисы. Для модульной структуры кода логика должна быть разделена на эти каталоги и файлы.

* Модели (Models) — определение схемы модели.
* Маршруты (Routes) — маршруты API сопоставляются с контроллерами.
* Контроллеры (Controllers) - контроллеры обрабатывают всю логику проверки параметров запроса, запроса и отправки ответов с правильными кодами.
* Службы (Services) - службы содержат запросы к базе данных и возвращают объекты или выдают ошибки.

**2.5 Вывод**

Таким образом в конце данной главы была спроектирована база данных, описаны UML-диаграммы вариантов использования для разных типов пользователей, как для администратора, так и для преподователя и студента, был описан алгоритм регистрации пользователей с хорошей проверкой на корректность введенных данных, а также определен стек технологий, используемый в данном проекте.

Таким образом, заранее известен список всех используемых технологий, сферы деятельности, которые они покрывают и некоторые основные алгоритмы, вокруг и на основе которых потом будут появляться новые механизмы взаимодействия между частями системы.

1. **Разработка программного средства**

Приложение будет разрабатываться посредством языка TypeScrtpt используя для реализации кроссплатформенную технологию node.js и модель MRCS.

MRCS — шаблон проектирования, с помощью которого его компоненты (модель данных приложения, маршруты приложения, службы приложения и взаимодействие с пользователем) разделены на четыре отдельных так, что модификация одного из них оказывает минимальное воздействие на остальные.

**3.1 Разработка серверной части**

Архитектура программного обеспечения — совокупность важнейших решений об организации программной системы. Архитектура включает: выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов; соединение выбранных элементов структуры и поведения, во всё более крупные системы; архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

Структура проекта состоит из папок «*Properties*», «*Controllers*», «*Migrations*», «*Models*», «*Queries*», «*Validation*», «*ViewModels*», «*View*» и классов «*Errors*», «*Program*», «*Startup*», а также файлов «*package.json*» и «*appsettings*».

Структура проекта изображена на рисунке 3.1.

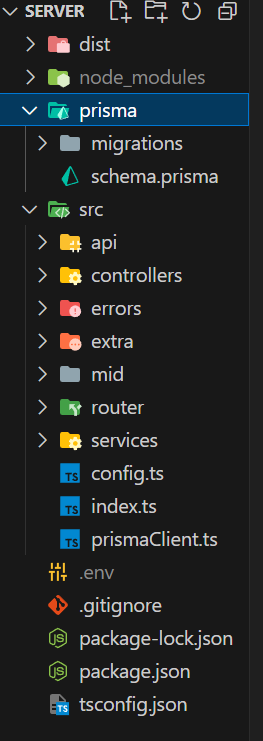


Рисунок 3.1 – Структура проекта