Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Архитектура вычислительных систем

К защите допустить:

И.О. Заведующего кафедрой информатики
_____С. И. Сиротко

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

ПРОЕКТИРОВАНИЕ БАЗА ДАННЫХ ДЛЯ ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ ПРОВЕДЕНИЯ МНОГОПОЛЬЗОВАТЕЛЬСКИХ ВИКТОРИН С НЕЧЕТКИМИ КРИТЕРИЯМИ ПРИНЯТИЯ ОТВЕТА

БГУИР КП 1-40 04 01 033 ПЗ

Студент Г. К. Лямцев

Руководитель А. В. Давыдчик

Минск 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном цифровом мире интерактивные формы досуга и обучения, такие как онлайн-викторины, приобретают всё большую популярность. Они используются для образования, корпоративных мероприятий, стриминга и просто как увлекательный способ провести время с друзьями. Однако большинство существующих платформ обладают жёсткой системой проверки ответов, где опечатка, синоним или иной порядок слов могут привести к незасчитанному, по сути, правильному ответу, что снижает удовольствие от игры и ограничивает креативность создателей вопросов.

Целью данного курсового проекта является проектирование и разработка базы данных для веб-приложения «QuizFuzz» – платформы для проведения многопользовательских викторин в реальном времени с поддержкой нечётких критериев принятия ответов. Ключевой особенностью системы является реализация гибкого механизма оценки ответов пользователей, который учитывает возможные опечатки, синонимы, фонетическое сходство и даже семантическую близость, что делает игровой процесс более комфортным и интуитивно понятным.

Основные задачи разработки включают:

1 Анализ предметной области и существующих аналогов для выявления их достоинств и недостатков.

2 Формирование функциональных требований к системе и определение подходящих инструментов для разработки.

3 Проектирование базы данных, включая разработку инфологической и даталогической моделей, адекватно отражающих сложную бизнес-логику приложения.

4 Создание физической модели базы данных с использованием выбранной системы управления базами данных (СУБД).

5 Описание ключевых запросов, обеспечивающих функционирование приложения.

Проектирование базы данных является критически важной частью разработки, так как она должна обеспечивать не только надежное хранение структурированных данных о пользователях, вопросах, игровых сессиях и ответах, но и высокую производительность при выполнении сложных запросов в реальном времени, а также поддерживать механизмы нечёткого поиска и аналитики. Грамотно спроектированная база данных станет основой для создания масштабируемого, производительного и отказоустойчивого приложения, способного выдерживать нагрузку от тысяч одновременных пользователей.

1 ОБЗОР СУЩЕСТВУЮЩИХ АНАЛОГОВ

Прежде чем приступить к проектированию собственной системы, необходимо провести анализ существующих на рынке решений, реализующих схожую функциональность. Это позволяет выявить их сильные и слабые стороны, определить общепринятые практики и выделить ниши для внедрения инноваций. Проведенный обзор поможет сформулировать конкурентные преимущества разрабатываемого приложения QuizFuzz и избежать ошибок, допущенных в существующих продуктах. В качестве аналогов были рассмотрены такие платформы, как JKLM.fun, Jackbox Games, Kahoot! и Quizizz.

1.1 JKLM.fun

JKLM.fun – это минималистичная веб-платформа для многопользовательских игр-викторин в реальном времени, которая стала одним из ключевых ориентиров для QuizFuzz.

Функциональность:

1 Создание приватных комнат по короткой ссылке.

2 Поочерёдные раунды с вопросами, на которые игроки вводят текстовые ответы.

3 Базовый механизм «нечёткого» сопоставления ответов, допускающий небольшие опечатки.

Преимущества:

1 Простота и минимализм пользовательского интерфейса, позволяющие быстро начать игру.

2 Низкий порог входа – не требуется регистрация.

3 Эффективная система приглашений в комнаты.

Недостатки:

1 Крайне ограниченный и неконфигурируемый механизм проверки ответов, не поддерживающий синонимы, фонетику или семантику.

2 Отсутствие системы пользователей, статистики, лидербордов и модерации контента.

3 Невозможность создавать собственные вопросы внутри платформы.

Таким образом, JKLM.fun представляет собой удобный, но функционально ограниченный инструмент, решающий базовую задачу проведения викторины без глубокой проработки гибкости проверки и аналитики.

1.2 Jackbox Games

Jackbox Games – это серия популярных party-игр, включающих викторины, которые транслируются на большом экране, а игроки участвуют со своих смартфонов.

Функциональность:

1 Набор разнообразных игровых пакетов с уникальной механикой.

2 Встроенные системы создания пользовательского контента (в некоторых играх).

3 Высококачественный визуальный ряд и озвучка.

Преимущества:

1 Полировка и высокое качество контента.

2 Интуитивный и весёлый геймплей, ориентированный на вечеринки.

3 Мощные встроенные инструменты модерации и цензуры.

Недостатки:

1 Является коммерческим продуктом с необходимостью покупки игровых пакетов.

2 Закрытая экосистема без возможности самостоятельного создания и модерации вопросов сообществом в единой базе.

3 Отсутствие гибкой системы проверки ответов, аналогичной нечёткому поиску QuizFuzz.

Платформа Jackbox служит отличным примером организации социального геймплея и работы комнат, но её модель не подходит для создания открытой платформы с пользовательским контентом.

1.3 Kahoot! и Quizizz

Kahoot! и Quizizz — это образовательные платформы для проведения викторин, широко используемые в учебных заведениях и корпоративной среде.

Функциональность:

1 Создание викторин с различными типами вопросов (выбор ответа, верно/неверно).

2 Публичные и приватные игровые сессии.

3 Подробная статистика и отчеты по результатам игроков.

4 Обширные библиотеки готового контента.

Преимущества:

1 Мощные аналитические и статистические инструменты.

2 Фокус на образовательном процессе и удобство для преподавателей.

3 Высокая стабильность и масштабируемость.

Недостатки:

1 Проверка ответов основана исключительно на выборе заранее заготовленных вариантов, что полностью исключает текстовый ввод и, как следствие, возможность реализации нечёткого сопоставления.

2 Отсутствие механик, характерных для развлекательных викторин (подсказки, динамическое начисление очков за скорость).

Данные платформы идеально решают задачи образования, но их игровая механика слишком ограничена для создания гибкой развлекательной системы с открытыми ответами.

1.4 Выводы

Проведенный анализ показал, что существующие решения либо предлагают простоту и минимализм в ущерб функциональности (JKLM.fun), либо являются закрытыми коммерческими продуктами с фиксированным контентом (Jackbox), либо ориентированы исключительно на формат тестов с множественным выбором (Kahoot!, Quizizz). Ни одна из рассмотренных платформ не сочетает в себе возможность пользовательского генерирования контента, мощную систему модерации, глубокую аналитику и, что не менее важно, гибкий механизм нечёткой проверки текстовых ответов. Это подтверждает актуальность и новизну разрабатываемого веб-приложения QuizFuzz, призванного заполнить выявленную рыночную нишу.

2 ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ И ИНСТРУМЕНТЫ РАЗРАБОТКИ

2.1 Функциональные требования к системе

Разрабатываемая система «QuizFuzz» представляет собой комплексное веб-приложение, предназначенное для организации и проведения многопользовательских викторин в реальном времени. Его ключевой особенностью является поддержка нечётких критериев оценки текстовых ответов, что значительно расширяет стандартные возможности подобных платформ.

Основные функциональные требования к системе включают следующие модули:

1 Модуль аутентификации и авторизации обеспечивает безопасный доступ пользователей к системе. Он должен предоставлять функционал регистрации по электронной почте и паролю, входа в систему и безопасного выхода, а также восстановления доступа к учетной записи. В рамках данного модуля реализуется ролевая модель управления доступом, включающая роли Гостя, Пользователя, Модератора и Администратора. Для защиты API необходимо использовать JWT-токены, а также предусмотреть механизмы защиты сессий и ограничения частоты запросов на аутентификацию.

2 Модуль управления игровыми комнатами отвечает за организацию виртуального пространства для проведения игр. Он позволяет пользователям создавать, редактировать и закрывать игровые комнаты. При создании комнаты должны настраиваться её ключевые параметры: тип доступа (публичный или приватный), лимит участников, условие завершения игры (по достижению определённого количества очков или числа раундов), а также система тегов для подбора вопросов. Для приватных комнат предусматривается генерация кодов доступа или уникальных ссылок-приглашений. Модуль должен предоставлять интерфейс для просмотра списка публичных комнат с возможностью их поиска и фильтрации.

3 Модуль игрового процесса является ядром системы и управляет проведением викторины. Его функционал включает организацию игровых сессий, состоящих из последовательных раундов. Система должна поддерживать различные типы вопросов: текстовые, с изображениями и аудио. Для каждого раунда устанавливается таймер, а также система динамических подсказок, которые открываются в заданные моменты времени. Игроки имеют возможность отправлять ответы неограниченное количество раз до истечения времени раунда, при этом модуль должен осуществлять защиту от флуда. Все события в реальном времени транслируются участникам с использованием технологии SignalR.

4 Модуль нечёткой проверки ответов реализует инновационный подход к оценке ответов игроков. Он включает конвейер предварительной обработки введённого текста, включающий нормализацию, приведение к нижнему

регистру, удаление пунктуации и транслитерацию. Модуль поддерживает несколько стратегий проверки: точное совпадение без учёта регистра, вычисление редакционного расстояния по алгоритмам Левенштейна и Дамерау, проверку по заранее заготовленным алиасам и синонимам, фонетическое сравнение с использованием алгоритмов Double Metaphone и их аналогов для кириллицы, а также семантический анализ на основе векторных представлений слов и фраз. Реализуется механизм агрегации результатов от разных стратегий с назначением весов и порогов уверенности, а также логирование процесса проверки для последующего анализа.

5 Модуль управления контентом и модерации регулирует процесс наполнения системы вопросами. Пользователи получают возможность подавать собственные вопросы на модерацию через специальную форму, указывая тип вопроса, контент, правильный ответ, подсказки и теги. Для модераторов и администраторов предоставляется интерфейс очереди модерации, где вопросы проверяются, редактируются, утверждаются или отклоняются. Каждое действие модератора фиксируется в истории изменений. Модуль также включает управление системой тегов для категоризации контента.

6 Модуль статистики и аналитики собирает и представляет данные об активности пользователей. Он формирует персональную статистику для каждого игрока, включая количество сыгранных и выигранных игр, общую точность ответов, количество уникально угаданных вопросов, среднее время ответа и предпочтительные категории. На основе этих данных строятся глобальные, недельные и сезонные лидерборды. Пользователи имеют доступ к детальной истории своих игр с возможностью просмотра результатов по каждому раунду.

7 Административный модуль предоставляет инструменты для управления всей системой. В его функционал входит управление пользователями: назначение ролей, блокировка аккаунтов. Администраторы осуществляют управление системой тегов, настройку системных лимитов и параметров работы приложения. Модуль также обеспечивает просмотр аудит-логов, в которых фиксируются все значимые действия пользователей и администраторов в системе.

2.2 Выбор СУБД

Для успешной реализации проекта QuizFuzz, учитывая его требования к хранению сложноструктурированных данных, обеспечению целостности, поддержке транзакций и высокой производительности при одновременной работе тысяч пользователей, был выбран подход, основанный на использовании реляционной модели данных. Данный подход обеспечивает логическую целостность информации, удобство представления данных в виде таблиц и эффективность выполнения сложных запросов, необходимых для работы системы игровых сессий, статистики и нечёткого поиска.

Требования к СУБД для проекта QuizFuzz:

1 Поддержка сложных структур данных и связей. Данные системы включают множество взаимосвязанных сущностей (пользователи, вопросы, комнаты, сессии, ответы, оценки), которые требуют соблюдения целостности связей и сложных JOIN-запросов.

2 Гарантия целостности и согласованности данных (ACID). Операции, такие как подсчет очков, завершение раунда и обновление лидербордов, должны быть атомарными и согласованными, что исключает возможность возникновения конфликтов и потерь данных при высокой параллельной нагрузке.

3 Высокая производительность при обработке запросов. Система требует быстрого выполнения операций выборки и агрегации данных для формирования лидербордов, статистики пользователей и подбора вопросов для раундов.

4 Масштабируемость. Архитектура БД должна позволять горизонтальное и вертикальное масштабирование для поддержки растущего числа пользователей и игровых сессий.

5 Надёжность и отказоустойчивость. СУБД должна обеспечивать стабильную работу, механизмы репликации и резервного копирования для минимизации простоев.

6 Поддержка расширенных типов данных. Для реализации семантического поиска в будущем может потребоваться поддержка векторных типов данных (эмбеддингов).

7 Безопасность. Необходимы механизмы разграничения доступа, шифрования данных и обеспечения конфиденциальности пользовательской информации.

Анализ кандидатов:

Были рассмотрены следующие реляционные СУБД: PostgreSQL, MySQL и SQLite.

2.2.1 PostgreSQL – это мощная, свободная объектно-реляционная система управления базами данных. Её разработка началась в 1986 году в Калифорнийском университете в Беркли, и с тех пор она прошла долгий путь, став одним из самых продвинутых и надежных решений в мире открытых СУБД. PostgreSQL строго следует стандартам SQL и поддерживает множество современных возможностей.

Преимущества:

1 Передовая функциональность. PostgreSQL предлагает богатый набор типов данных, включая нативные массивы, JSON/JSONB, XML, геопространственные данные и пользовательские типы. Особенно важно наличие расширения pgvector, которое позволяет эффективно хранить и выполнять поиск по векторным представлениям (эмбеддингам). Это критически важно для будущей реализации семантического модуля нечёткого поиска в QuizFuzz, где ответы можно будет сравнивать по смысловой близости.

2 Высокая производительность и оптимизация сложных запросов. PostgreSQL оснащен sophisticated оптимизатором запросов, который эффективно выполняет сложные JOIN-операции и агрегации, необходимые для формирования статистики пользователей и лидербордов. Поддержка расширенных методов индексирования (таких как GIN, GiST, BRIN) позволяет ускорить поиск по полнотекстовым данным, JSONB-полям и другим сложным структурам, что напрямую влияет на скорость подбора вопросов и анализа ответов.

3 Строгое соблюдение ACID и мощная система конкурентности. Для многопользовательской системы, такой как QuizFuzz, где десятки игроков могут одновременно отправлять ответы в одном раунде, механизм многоверсионного управления конкурентным доступом (MVCC) в PostgreSQL гарантирует, что транзакции по начислению очков и обновлению состояния игры будут изолированными и согласованными без блокировок на уровне таблицы.

4 Надёжность и отказоустойчивость. Поддержка синхронной и асинхронной репликации, механизмы point-in-time recovery (PITR) и активное сообщество делают PostgreSQL надежным выбором для проектов, где простои недопустимы.

Недостатки:

1 Требует более высокой квалификации администраторов. Для тонкой настройки производительности под конкретную нагрузку (настройка параметров shared_buffers, work_mem, эффективное использование партиционирования) требуются глубокие знания системы.

2 Потребление ресурсов. По умолчанию PostgreSQL может потреблять больше памяти и CPU, чем некоторые конкуренты, однако эта "прожорливость" является платой за её мощь и надежность, и для проекта уровня QuizFuzz эти затраты являются оправданными.

2.2.2 MySQL – одна из самых популярных в мире открытых реляционных СУБД, изначально разработанная компанией MySQL AB и сейчас принадлежащая Oracle Corporation. Благодаря своей простоте, производительности и тесной интеграции со стеком LAMP (Linux, Apache, MySQL, PHP/Python/Perl), она стала стандартом де-факто для многих веб-приложений.

Преимущества:

1 Простота использования и настройки. MySQL известна своим быстрым временем развертывания и относительно низким порогом входа для администраторов. Огромное сообщество и обилие документации упрощают поиск решений типичных проблем.

2 Высокая производительность на операциях чтения. Для сценариев, преимущественно состоящих из операций SELECT (например, чтение вопросов, отображение списка комнат), MySQL с движком InnoDB демонстрирует очень хорошую скорость.

3 Репутация и распространённость. Широкая известность технологии снижает риски, связанные с её выбором, и упрощает поиск квалифицированных кадров для поддержки проекта.

Недостатки:

1 Ограниченная функциональность по сравнению с PostgreSQL. Исторически MySQL менее строго следовала стандартам SQL. Хотя ситуация улучшается, такие функции, как рекурсивные запросы (WITH RECURSIVE), развитые оконные функции и работа со сложными типами данных, в PostgreSQL часто реализованы более зрело и эффективно. Это могло бы ограничить возможности для сложной аналитики в QuizFuzz.

2 Ограничения в механизмах параллелизма. Хотя движок InnoDB значительно улучшил ситуацию, в сценариях с экстремально высокой конкурентной записью (когда сотни игроков одновременно отправляют ответы) MySQL в некоторых конфигурациях может показывать менее стабильную производительность по сравнению с PostgreSQL из-за различий в реализации MVCC.

2.2.3 SQLite – это встраиваемая, автономная, бессерверная, транзакционная СУБД с нулевой конфигурацией. Её код находится в общественном достоянии, что делает её чрезвычайно популярной для встраивания в приложения, мобильные системы и браузеры.

Преимущества:

1 Простота развёртывания. SQLite не требует установки и администрирования отдельного серверного процесса. База данных представляет собой единственный файл на диске, что делает её идеальной для локальных приложений и прототипирования.

2 Нулевая нагрузка на администрирование. Отсутствие необходимости настраивать пользователей, права доступа, репликацию или резервное копирование на уровне СУБД (резервное копирование — это просто копирование файла).

Недостатки:

1 Отсутствие масштабируемости и сетевая архитектура. SQLite не поддерживает сетевые подключения и предназначена для использования одним приложением на одной машине. Эта архитектурная особенность делает её абсолютно непригодной для многопользовательского веб-приложения, такого как QuizFuzz, где тысячи клиентов должны одновременно обращаться к данным через веб-сервер.

2 Ограниченная производительность при высокой конкурентной записи. SQLite использует блокировку на уровне всей базы данных при операциях записи. Это означает, что если один игрок отправляет ответ и БД заблокирована на запись, все остальные игроки, пытающиеся отправить ответы в этот момент, будут вынуждены ждать. Это создает "бутылочное горлышко" и полностью исключает возможность реального времени для сотен одновременных пользователей.

2.2.4 Обоснование выбора. Для высоконагруженного многопользовательского веб-приложения QuizFuzz, требующего сложных запросов, абсолютной целостности данных и возможности глубокого масштабирования, выбор пал на PostgreSQL.

Это решение обусловлено её беспрецедентной надёжностью, строгим соблюдением стандартов ACID, мощной поддержкой сложных типов данных и запросов, а также наличием критически важного расширения pgvector для будущей реализации семантического поиска. Её архитектура, основанная на многоверсионности (MVCC), идеально подходит для сценариев с высокой конкурентностью, характерных для игровых сессий. Хотя её настройка может быть сложнее, чем у MySQL, её производительность, функциональность и отказоустойчивость в условиях высокой нагрузки полностью соответствуют амбициозным требованиям проекта.

SQLite, в свою очередь, была сразу исключена из рассмотрения из-за фундаментальной архитектурной непригодности для многопользовательского клиент-серверного приложения.

Таким образом, PostgreSQL обеспечивает необходимый технологический фундамент для построения производительной, надежной и масштабируемой системы QuizFuzz, соответствующей всем заявленным в ТЗ требованиям как на текущий момент, так и для будущего развития.

3 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Проектирование базы данных является критически важным этапом создания системы QuizFuzz, поскольку от выбранной структуры хранения данных напрямую зависят производительность, надежность и масштабируемость всего приложения. В данном разделе представлены последовательные этапы проектирования БД: инфологическая модель, описывающая предметную область на концептуальном уровне, и даталогическая модель, определяющая структуру данных в терминах выбранной СУБД.

3.1 Инфологическая модель

Инфологическая модель представляет собой абстрактное описание предметной области, независимое от конкретной реализации в системе управления базами данных. Она фиксирует ключевые сущности, их атрибуты и взаимосвязи, отражая бизнес-логику приложения QuizFuzz.

Пользовательская подсистема:

1 User (Пользователь): Центральная сущность, содержащая учетные данные (email, хеш пароля) и основную информацию (имя, дата регистрации, последний вход).

2 Role (Роль): Справочник возможных ролей в системе: Admin, Moderator, User.

3 UserRole (Связь пользователя и роли): Обеспечивает назначение нескольких ролей одному пользователю.

Подсистема контента и модерации:

1 Question (Вопрос): Основная сущность, хранящая данные викторины: текст вопроса, тип (TEXT, IMAGE, AUDIO), правильный ответ, сложность, статус модерации.

2 QuestionType (Тип вопроса): Справочник допустимых типов контента.

3 Tag (Тег): Система категоризации для вопросов (например, "История", "Кино", "Наука").

4 QuestionTag (Связь вопроса и тега): Позволяет присваивать вопросу несколько тегов.

5 MediaAsset (Медиафайл): Хранит метаданные медиафайлов (изображений, аудио), привязанных к вопросу (URL, тип, размер, контрольная сумма).

6 Hint (Подсказка): Подсказки, связанные с вопросом, с указанием времени их открытия во время раунда.

7 FuzzyAlias (Алиас/Синоним): Альтернативные варианты правильного ответа, заданные автором вопроса для нечеткого сопоставления.

Подсистема модерации контента:

1 ModerationQueue (Очередь модерации): Фиксирует факт подачи вопроса на модерацию, связывая вопрос, пользователя-автора и назначая статус (PENDING, APPROVED, REJECTED).

2 ModerationAction (Действие модератора): Журналирует все действия модераторов (утверждение, отклонение, редактирование) с указанием причины и сохранением снимка изменений.

Игровая подсистема:

1 Room (Комната): Виртуальное пространство для игры. Хранит настройки: название, приватность, лимит игроков, условие победы, режим выбора тегов.

2 RoomTagSelection (Выбор тегов комнаты): Определяет, какие теги и с каким весом используются для подбора вопросов в конкретной комнате.

3 Invitation (Приглашение): Хранит коды и ссылки для доступа в приватные комнаты.

4 GameSession (Игровая сессия): Сущность, представляющая собой одну полную игру внутри комнаты, от старта до завершения.

5 PlayerInRoom (Игрок в комнате): Регистрирует участие пользователя в конкретной игровой сессии.

6 GameRound (Раунд игры): В рамках сессии хранит данные каждого раунда: выпавший вопрос, временные рамки.

7 PlayerAnswer (Ответ игрока): Фиксирует каждую попытку ответа пользователя в раунде с временной меткой.

Подсистема проверки ответов и аналитики:

1 AnswerEvaluation (Оценка ответа): Ядро системы нечеткого сопоставления. Хранит результат проверки каждого ответа: был ли он

засчитан, какая стратегия использовалась (EXACT, LEVENSHTEIN, SEMANTIC), уверенность алгоритма и начисленные очки.

2 Scoreboard (Таблица очков): Аккумулирует общие результаты пользователя в рамках одной игровой сессии.

3 UserStats (Статистика пользователя): Агрегированные персональные метрики игрока за все время (сыграно игр, точность, среднее время ответа).

4 LeaderboardEntry (Запись в рейтинге): Рассчитываемые позиции пользователей в глобальных, недельных и сезонных рейтингах.

Связи между ключевыми сущностями:

1 Пользователь (User) может создавать вопросы (Question) и участвовать в игровых сессиях (GameSession) как игрок (PlayerInRoom).

2 Вопрос (Question) проходит жизненный цикл через очередь модерации (ModerationQueue), где модераторы (User) совершают действия (ModerationAction).

3 Игровая сессия (GameSession) состоит из раундов (GameRound), в каждом из которых игроки (PlayerInRoom) дают ответы (PlayerAnswer), которые оцениваются (AnswerEvaluation) и влияют на общий счет (Scoreboard).

Данная инфологическая модель адекватно отражает все бизнес-процессы приложения QuizFuzz и служит основой для перехода к даталогическому проектированию.

3.2 Даталогическая модель

Даталогическая модель описывает структуру данных на уровне базы данных, в том числе типы данных, ограничения и связи между таблицами. Она отвечает за определение точных типов данных для каждого атрибута сущности, установление ограничений на значения и обеспечение целостности данных. В данном разделе представлена детальная спецификация всех сущностей, их атрибутов и типов данных, которые будут использоваться в базе данных системы. Модель строится на основе инфологической модели, уточняя типы данных и устанавливая точные форматы хранения информации для эффективного функционирования системы.

Описание таблиц и их атрибутов:

1 User (Пользователь):

– id (UUID, PRIMARY KEY) – уникальный идентификатор пользователя;

– email (VARCHAR(255), UNIQUE) – электронная почта;

– username (VARCHAR(100), UNIQUE) – имя пользователя;

– password_hash (TEXT) – хэш пароля;

– created_at (TIMESTAMPTZ) – дата регистрации;

– last_login_at (TIMESTAMPTZ) – дата последнего входа;

– is_banned (BOOLEAN) – статус блокировки.

2 Role (Роль):

– id (SMALLSERIAL, PRIMARY KEY) – идентификатор роли;

– name (VARCHAR(50), UNIQUE) – название роли (Admin, Moderator, User).

3 UserRole (Связь пользователя и роли):

– user_id (UUID, FOREIGN KEY – User) – идентификатор пользователя;

– role_id (SMALLINT, FOREIGN KEY – Role) – идентификатор роли;

PRIMARY KEY (user_id, role_id).

4 QuestionType (Тип вопроса):

– id (SMALLSERIAL, PRIMARY KEY) – идентификатор типа;

– code (VARCHAR(20), UNIQUE) – код типа (TEXT, IMAGE, AUDIO);

– description (TEXT) – описание типа.

5 Question (Вопрос):

– id (UUID, PRIMARY KEY) – уникальный идентификатор вопроса;

– type_id (SMALLINT, FOREIGN KEY – QuestionType) – тип вопроса;

– title (TEXT) – заголовок вопроса;

– prompt_text (TEXT) – текст вопроса;

– correct_answer (TEXT) – правильный ответ;

– difficulty (SMALLINT) – сложность (1-5);

– status (VARCHAR(20)) – статус модерации;

– author_user_id (UUID, FOREIGN KEY – User) – автор вопроса;

– created_at (TIMESTAMPTZ) – дата создания;

– updated_at (TIMESTAMPTZ) – дата обновления.

6 Tag (Тег):

– id (SERIAL, PRIMARY KEY) – идентификатор тега;

– name (VARCHAR(50), UNIQUE) – название тега;

– description (TEXT) – описание тега;

– is_active (BOOLEAN) – активен ли тег.

7 QuestionTag (Связь вопроса и тега):

– question_id (UUID, FOREIGN KEY – Question) – идентификатор вопроса;

– tag_id (INTEGER, FOREIGN KEY – Tag) – идентификатор тега;

PRIMARY KEY (question_id, tag_id).

8 MediaAsset (Медиафайл):

– id (UUID, PRIMARY KEY) – идентификатор медиафайла;

– question_id (UUID, FOREIGN KEY– Question) – связанный вопрос;

– media_type (VARCHAR(20)) – тип медиа (IMAGE, AUDIO, VIDEO);

– url (TEXT) – ссылка на файл;

– storage_provider (VARCHAR(50)) – провайдер хранилища;

– checksum (TEXT) – контрольная сумма файла;

– duration_sec (INTEGER) – длительность (для аудио/видео);

– width (INTEGER) – ширина (для изображений);

– height (INTEGER) – высота (для изображений);

– is_sensitive (BOOLEAN) – пометка "чувствительный контент".

9 Hint (Подсказка):

– id (UUID, PRIMARY KEY) – идентификатор подсказки;

– question_id (UUID, FOREIGN KEY – Question) – связанный вопрос;

– order_index (SMALLINT) – порядковый номер;
– hint_text (TEXT) – текст подсказки;
– media_asset_id (UUID, FOREIGN KEY – MediaAsset) – медиа-подсказка;
– reveal_time_sec (SMALLINT) – время открытия в секундах.
10 FuzzyAlias (Алиас ответа):
– id (UUID, PRIMARY KEY) – идентификатор алиаса;
– question_id (UUID, FOREIGN KEY – Question) – связанный вопрос;
– alias_text (TEXT) – текст алиаса;
– kind (VARCHAR(20)) – тип (SYNONYM, ALT_SPELLING, TRANSLIT).
11 ModerationQueue (Очередь модерации):
– id (UUID, PRIMARY KEY) – идентификатор записи;
– question_id (UUID, FOREIGN KEY – Question) – проверяемый вопрос;
– submitted_by_user_id (UUID, FOREIGN KEY – User) – автор вопроса;
– status (VARCHAR(20)) – статус (PENDING, APPROVED, REJECTED);
– moderator_user_id (UUID, FOREIGN KEY – User) – модератор;
– comment (TEXT) – комментарий модератора;
– created_at (TIMESTAMPTZ) – дата подачи;
– updated_at (TIMESTAMPTZ) – дата решения.
12 ModerationAction (Действие модератора):
– id (UUID, PRIMARY KEY) – идентификатор действия;
– queue_id (UUID, FOREIGN KEY – ModerationQueue) – запись в очереди;
– action (VARCHAR(20)) – действие (APPROVE, REJECT, EDIT);
– moderator_user_id (UUID, FOREIGN KEY – User) – модератор;
– reason (TEXT) – причина действия;
– created_at (TIMESTAMPTZ) – дата действия;
– diff_snapshot (JSONB) – снимок изменений.
13 Room (Комната):
– id (UUID, PRIMARY KEY) – идентификатор комнаты;
– owner_user_id (UUID, FOREIGN KEY – User) – создатель;
– name (VARCHAR(255)) – название комнаты;
– is_private (BOOLEAN) – приватная/публичная;
– access_code_hash (TEXT) – хэш кода доступа;
– max_players (SMALLINT) – макс. игроков;
– victory_condition_type (VARCHAR(20)) – условие победы;
– victory_value (INTEGER) – значение условия;
– selected_tags_mode (VARCHAR(20)) – режим тегов;
– state (VARCHAR(20)) – состояние комнаты;
– created_at (TIMESTAMPTZ) – дата создания;
– updated_at (TIMESTAMPTZ) – дата обновления.
14 RoomTagSelection (Выбор тегов комнаты):
– room_id (UUID, FOREIGN KEY – Room) – комната;
– tag_id (INTEGER, FOREIGN KEY – Tag) – тег;

– weight (SMALLINT) – вес тега;
PRIMARY KEY (room_id, tag_id).
15 Invitation (Приглашение):
– id (UUID, PRIMARY KEY) – идентификатор приглашения;
– room_id (UUID, FOREIGN KEY – Room) – комната;
– code (VARCHAR(50), UNIQUE) – код приглашения;
– created_at (TIMESTAMPTZ) – дата создания;
– expires_at (TIMESTAMPTZ) – срок действия;
– created_by_user_id (UUID, FOREIGN KEY – User) – создатель.
16 GameSession (Игровая сессия):
– id (UUID, PRIMARY KEY) – идентификатор сессии;
– room_id (UUID, FOREIGN KEY – Room) – комната;
– started_at (TIMESTAMPTZ) – время начала;
– ended_at (TIMESTAMPTZ) – время окончания;
– total_rounds_planned (SMALLINT) – планируемое число раундов;
– total_rounds_played (SMALLINT) – сыгранное число раундов.
17 PlayerInRoom (Игрок в комнате):
– id (UUID, PRIMARY KEY) – идентификатор участия;
– session_id (UUID, FOREIGN KEY – GameSession) – игровая сессия;
– user_id (UUID, FOREIGN KEY – User) – игрок;
– joined_at (TIMESTAMPTZ) – время присоединения;
– left_at (TIMESTAMPTZ) – время выхода;
– is_owner_snapshot (BOOLEAN) – был ли создателем.
18 GameRound (Раунд игры):
– id (UUID, PRIMARY KEY) – идентификатор раунда;
– session_id (UUID, FOREIGN KEY – GameSession) – игровая сессия;
– round_index (SMALLINT) – номер раунда;
– question_id (UUID, FOREIGN KEY – Question) – вопрос раунда;
– started_at (TIMESTAMPTZ) – время начала;
– ended_at (TIMESTAMPTZ) – время окончания;
– time_limit_sec (SMALLINT) – лимит времени.
19 PlayerAnswer (Ответ игрока):
– id (UUID, PRIMARY KEY) – идентификатор ответа;
– round_id (UUID, FOREIGN KEY – GameRound) – раунд;
– user_id (UUID, FOREIGN KEY – User) – игрок;
– answer_text (TEXT) – текст ответа;
– answer_time_ms (INTEGER) – время ответа;
– created_at (TIMESTAMPTZ) – время отправки.
20 AnswerEvaluation (Оценка ответа):
– id (UUID, PRIMARY KEY) – идентификатор оценки;
– player_answer_id (UUID, FOREIGN KEY – PlayerAnswer) – оцененный ответ;
– is_correct (BOOLEAN) – правильность;
– match_strategy (VARCHAR(20)) – стратегия совпадения;
– score_awarded (INTEGER) – начисленные очки;

– confidence (NUMERIC(3,2)) – уверенность алгоритма;
– normalized_answer (TEXT) – нормализованный ответ;
– evaluated_at (TIMESTAMPTZ) – время оценки.
21 Scoreboard (Таблица очков):
– id (UUID, PRIMARY KEY) – идентификатор записи;
– session_id (UUID, FOREIGN KEY – GameSession) – игровая сессия;
– user_id (UUID, FOREIGN KEY – User) – игрок;
– score_total (INTEGER) – общий счет;
– correct_count (INTEGER) – число верных ответов;
– unique_correct_count (INTEGER) – уникальные верные ответы.
22 UserStats (Статистика пользователя):
– user_id (UUID, PRIMARY KEY, FOREIGN KEY – User) – пользователь;
– games_played (INTEGER) – сыграно игр;
– games_won (INTEGER) – выиграно игр;
– questions_answered (INTEGER) – дано ответов;
– unique_first_correct (INTEGER) – первых правильных ответов;
– avg_answer_time_ms (INTEGER) – среднее время ответа;
– favorite_tags (JSONB) – любимые теги;
– top_accuracy_tags (JSONB) – теги с лучшей точностью;
– updated_at (TIMESTAMPTZ) – время обновления.
23 LeaderboardEntry (Запись в рейтинге):
– id (UUID, PRIMARY KEY) – идентификатор записи;
– period (VARCHAR(20)) – период (GLOBAL, WEEKLY, SEASON);
– start_at (TIMESTAMPTZ) – начало периода;
– end_at (TIMESTAMPTZ) – конец периода;
– user_id (UUID, FOREIGN KEY – User) – пользователь;
– score (INTEGER) – очки в рейтинге;
– rank (INTEGER) – позиция в рейтинге.
24 AuditLog (Журнал аудита):
– id (UUID, PRIMARY KEY) – идентификатор записи;
– user_id (UUID, FOREIGN KEY – User) – пользователь;
– action (VARCHAR(100)) – действие;
– entity_type (VARCHAR(50)) – тип сущности;
– entity_id (UUID) – идентификатор сущности;
– meta (JSONB) – метаданные;
– created_at (TIMESTAMPTZ) – время события.

Созданная даталогическая модель обеспечивает надежную основу для физической реализации базы данных. Все сущности имеют четко определенные типы данных и ограничения целостности, что гарантирует согласованность хранимой информации. Связи между таблицами реализованы через внешние ключи, что поддерживает реляционную целостность данных на уровне СУБД.

4 РАЗРАБОТКА БАЗЫ ДАННЫХ

Разработка базы данных представляет собой ключевой этап создания программного комплекса QuizFuzz, обеспечивающий основу для хранения, обработки и анализа всей информации системы. На данном этапе осуществляется переход от даталогической модели к физической реализации средствами выбранной СУБД PostgreSQL.

База данных предназначена для централизованного хранения сведений о пользователях, вопросах, игровых сессиях, ответах участников, статистике и модерации контента. Она обеспечивает взаимосвязанность всех сущностей, целостность данных, а также возможность выполнения аналитических запросов и поддержки игрового процесса в реальном времени.

4.1 Физическая модель базы данных

Физическая модель базы данных представляет собой реализацию даталогической модели в конкретной СУБД PostgreSQL. Модель включает таблицы, индексы, последовательности и ограничения целостности, оптимизированные для эффективного выполнения запросов и обеспечения надежности хранения данных.

Ключевые аспекты физической модели:

1 Типы данных – Использование соответствующих типов данных PostgreSQL: UUID для первичных ключей, TIMESTAMPTZ для временных меток, JSONB для гибкого хранения структурированных данных, SMALLINT для оптимизации хранения числовых значений с малым диапазоном.

2 Ограничения целостности – Реализация внешних ключей с каскадными операциями, проверочные ограничения (CHECK), уникальные ограничения (UNIQUE) и ограничения NOT NULL для обеспечения логической целостности данных.

3 Индексация – Создание оптимальных индексов для ускорения выполнения запросов, включая B-tree индексы для часто используемых полей фильтрации и GIN индексы для полнотекстового поиска и работы с JSONB данными.

4.2 Создание таблиц

Создание таблиц выполнялось с помощью SQL-скриптов, реализующих физическую модель данных. Каждая таблица содержит строго типизированные поля с соответствующими ограничениями.

Пример создания основных таблиц:

```
-- Таблица пользователей
CREATE TABLE "User" (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email VARCHAR(255) UNIQUE NOT NULL,
    username VARCHAR(100) UNIQUE NOT NULL,
```

```
    password_hash TEXT NOT NULL,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    last_login_at TIMESTAMPTZ,
    is_banned BOOLEAN DEFAULT FALSE
);

-- Таблица вопросов
CREATE TABLE Question (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    type_id SMALLINT NOT NULL REFERENCES QuestionType(id),
    title TEXT NOT NULL,
    prompt_text TEXT,
    correct_answer TEXT NOT NULL,
    difficulty SMALLINT CHECK (difficulty BETWEEN 1 AND 5),
    status VARCHAR(20) DEFAULT 'DRAFT',
    author_user_id UUID REFERENCES "User"(id),
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Таблица игровых сессий
CREATE TABLE GameSession (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    room_id UUID NOT NULL REFERENCES Room(id),
    started_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    ended_at TIMESTAMPTZ,
    total_rounds_planned SMALLINT NOT NULL,
    total_rounds_played SMALLINT DEFAULT 0
);
```

Пример создания таблиц связей:

```
-- Связь вопросов и тегов
CREATE TABLE QuestionTag (
    question_id UUID NOT NULL REFERENCES Question(id) ON DELETE CASCADE,
    tag_id INTEGER NOT NULL REFERENCES Tag(id) ON DELETE CASCADE,
    PRIMARY KEY (question_id, tag_id)
);

-- Связь комнат и тегов с весами
CREATE TABLE RoomTagSelection (
    room_id UUID NOT NULL REFERENCES Room(id) ON DELETE CASCADE,
    tag_id INTEGER NOT NULL REFERENCES Tag(id) ON DELETE CASCADE,
    weight SMALLINT NOT NULL DEFAULT 1,
    PRIMARY KEY (room_id, tag_id)
);
```

Пример создания таблиц для системы ответов и оценки:

```
-- Таблица ответов игроков
CREATE TABLE PlayerAnswer (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    round_id UUID NOT NULL REFERENCES GameRound(id) ON DELETE CASCADE,
    user_id UUID NOT NULL REFERENCES "User"(id),
    answer_text TEXT NOT NULL,
    answer_time_ms INTEGER NOT NULL,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Таблица оценки ответов (ядро нечеткого сопоставления)
CREATE TABLE AnswerEvaluation (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    player_answer_id UUID NOT NULL REFERENCES PlayerAnswer(id) ON DELETE
CASCADE,
    is_correct BOOLEAN NOT NULL,
    match_strategy VARCHAR(20) NOT NULL,
    score_awarded INTEGER NOT NULL,
```

```
    confidence NUMERIC(3,2) CHECK (confidence >= 0 AND confidence <= 1),
    normalized_answer TEXT NOT NULL,
    evaluated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Таблица статистики пользователей
CREATE TABLE UserStats (
    user_id UUID PRIMARY KEY REFERENCES "User"(id),
    games_played INTEGER DEFAULT 0,
    games_won INTEGER DEFAULT 0,
    questions_answered INTEGER DEFAULT 0,
    unique_first_correct INTEGER DEFAULT 0,
    avg_answer_time_ms INTEGER DEFAULT 0,
    favorite_tags JSONB,
    top_accuracy_tags JSONB,
    updated_at TIMESTAMPTZ DEFAULT NOW()
);
```

4.3 Запросы

Запросы к базе данных обеспечивают реализацию бизнес-логики приложения и позволяют получать необходимую информацию для отображения в пользовательском интерфейсе и проведения игрового процесса.

Примеры ключевых запросов, используемых в системе:

1 Получение списка активных публичных комнат:

```
SELECT r.id, r.name, r.max_players,
       COUNT(pir.user_id) as current_players,
       u.username as owner_name
FROM Room r
JOIN "User" u ON r.owner_user_id = u.id
LEFT JOIN GameSession gs ON r.id = gs.room_id AND gs.ended_at IS NULL
LEFT JOIN PlayerInRoom pir ON gs.id = pir.session_id AND pir.left_at IS NULL
WHERE r.is_private = false
  AND r.state = 'LOBBY'
GROUP BY r.id, u.username;
```

2 Подбор вопросов для игровой сессии по тегам комнаты:

```
SELECT q.id, q.title, q.prompt_text, q.correct_answer,
       q.difficulty, qt.code as question_type
FROM Question q
JOIN QuestionType qt ON q.type_id = qt.id
JOIN QuestionTag qtag ON q.id = qtag.question_id
JOIN RoomTagSelection rts ON qtag.tag_id = rts.tag_id
WHERE rts.room_id = $1
  AND q.status = 'APPROVED'
  AND q.id NOT IN (
    SELECT gr.question_id
    FROM GameRound gr
    JOIN GameSession gs ON gr.session_id = gs.id
    WHERE gs.room_id = $1
      AND gs.started_at > NOW() - INTERVAL '7 days'
  )
GROUP BY q.id, qt.code
ORDER BY RANDOM()
LIMIT 20;
```

3 Расчет результатов раунда с учетом нечеткого сопоставления:

```
SELECT u.username,
       SUM(ae.score_awarded) as round_score,
       ae.match_strategy,
       pa.answer_text,
       ae.normalized_answer
```

```
FROM PlayerAnswer pa
JOIN AnswerEvaluation ae ON pa.id = ae.player_answer_id
JOIN "User" u ON pa.user_id = u.id
WHERE pa.round_id = $1
  AND ae.is_correct = true
ORDER BY ae.score_awarded DESC, pa.answer_time_ms ASC;
```

4 Обновление статистики пользователя после завершения игры:

```
WITH game_stats AS (
    SELECT
        pir.user_id,
        COUNT(DISTINCT gs.id) as games_played,
        COUNT(DISTINCT CASE WHEN sb.score_total = max_scores.max_score THEN
gs.id END) as games_won,
        COUNT(pa.id) as questions_answered,
        AVG(pa.answer_time_ms) as avg_time
    FROM PlayerInRoom pir
    JOIN GameSession gs ON pir.session_id = gs.id
    JOIN PlayerAnswer pa ON pir.user_id = pa.user_id AND pa.round_id IN (
        SELECT id FROM GameRound WHERE session_id = gs.id
    )
    JOIN Scoreboard sb ON pir.session_id = sb.session_id AND pir.user_id =
sb.user_id
    JOIN (
        SELECT session_id, MAX(score_total) as max_score
        FROM Scoreboard
        GROUP BY session_id
    ) max_scores ON sb.session_id = max_scores.session_id
    WHERE gs.ended_at IS NOT NULL
      AND pir.user_id = $1
    GROUP BY pir.user_id
)
INSERT INTO UserStats (user_id, games_played, games_won, questions_answered,
avg_answer_time_ms, updated_at)
SELECT user_id, games_played, games_won, questions_answered, avg_time, NOW()
FROM game_stats
ON CONFLICT (user_id)
DO UPDATE SET
    games_played = EXCLUDED.games_played,
    games_won = EXCLUDED.games_won,
    questions_answered = EXCLUDED.questions_answered,
    avg_answer_time_ms = EXCLUDED.avg_answer_time_ms,
    updated_at = NOW();
```

В результате проектирования и реализации базы данных на основе PostgreSQL была создана оптимальная, устойчивая и расширяемая структура, которая обеспечивает целостность, быстродействие и удобство работы с данными. Использование реляционной модели позволило эффективно описать все взаимосвязи между элементами системы и обеспечить их согласованность при выполнении любых операций.

ЗАКЛЮЧЕНИЕ

СПИСОК ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИЕ А
(обязательное)
Справка о проверке на заимствование

ПРИЛОЖЕНИЕ Б
(обязательное)
Листинг программного кода

ПРИЛОЖЕНИЕ В
(обязательное)
Функциональная схема алгоритма, реализующего программное средство

ПРИЛОЖЕНИЕ Г
(обязательное)
Блок схема алгоритма,
реализующего программное средство

ПРИЛОЖЕНИЕ Д
(обязательное)
Графический интерфейс пользователя

ПРИЛОЖЕНИЕ Е
(обязательное)
Ведомость документов