**РЕФЕРАТ**

**РЕЗЮМЕ**

Тема дипломной работы «Веб-приложение для интерактивного изучения английского языка в сфере IT».

В процессе выполнения данной дипломной работы был разработан программный продукт, который позволяет нам проходить викторины и управлять ими, просматривать статистику и уровень знаний. А также данный комплекс позволяет получать различного рода отчёты.

Объектом разработки является программный продукт, направленный на интелектуальное развитие пользователя.

Поставленная задача была выполнена в полном объеме.

**РЭЗЮМЭ**

Тэма дыпломнай працы – «Вэб-дадатак для інтэрактыўнага вывучэння англійскай мовы ў сферы IT».

У працэсе выканання дадзенай дыпломнай працы быў распрацаваны пра-граммный прадукт, які дазваляе нам праходзіць віктарыны і кіраваць імі, праглядаць статыстыку і ўзровень ведаў. А таксама дадзены комплекс дазваляе атрымліваць рознага роду справаздачы.

Аб'ектам распрацоўкі з'яўляецца праграмны прадукт, накіраваны на интелектуальное развіццё карыстальніка.

Пастаўленая задача была выканана ў поўным аб'ёме.

**summary**

The subject of degree work – "Web-application for interactive learning English in IT".

In the process of completing this thesis, a software product was developed that allows us to go through quizzes and manage them, view statistics and the level of knowledge. And also, this complex allows you to receive various kinds of reports.

The object of development is a software product aimed at the intellectual development of the user.

The task has been carried out successfully and completely.

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 4](#_Toc38568341)

[**1** **АНАЛИТИЧЕСКИЙ ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ НА ОСНОВЕ ИЗУЧЕНИЯ ЛИТЕРАТУРНЫХ И ДРУГИХ ИСТОЧНИКОВ ПО ТЕМЕ ДИПЛОМНОЙ РАБОТЫ** 5](#_Toc38568342)

[1.1 Обзор аналогов 7](#_Toc38568343)

[1.2 Постановка задачи на дипломное проектирование 14](#_Toc38568344)

[1.3 Анализ подхода к реализации технического задания 14](#_Toc38568345)

[1.4 Анализ стека технологий для технического задания 16](#_Toc38568346)

[**2** **РАЗРАБОТКА АЛГОРИТМОВ И ТЕХНОЛОГИИ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ** 23](#_Toc38568347)

[2.1 Архитектура программного комплекса 23](#_Toc38568348)

[2.2 Алгоритм работы программного комплекса 28](#_Toc38568349)

[2.3 Проектирование базы данных и структуры программного комплекса 32](#_Toc38568350)

[**ЗАКЛЮЧЕНИЕ** 35](#_Toc38568351)

[**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ** 36](#_Toc38568352)

# **ВВЕДЕНИЕ**

Чем занять себя когда скучно? Первое что приходит на ум: фильмы, сериалы, видеоигры. Такие способы само развлечения часто бывают утомительными, надоедающими и шумными. Из бесшумных вариантов приходит на ум чтение книг, но часто для людей тяжело начать читать целую книгу в эпоху цифровизации, когда информации много и получается она короткими порциями (википедия, новости, мемы и др). Где баланс между беззаботным отдыхом и непрерывным потреблением информации. Есть такой вид развлечения, как викторины.

Викторина — игра, заключающаяся в ответах на устные или письменные вопросы из различных областей знания.

Викторины в основном отличаются друг от друга правилами, определяющими очерёдность хода, тип и сложность вопроса, порядок определения победителей, вознаграждение за правильный ответ.

Существуют настольные викторины с заранее подготовленными вопросами. Очень часто на рынке настольных игр одновременно представлено несколько версий одной и той же игры, различающихся набором (часто уровнем сложности) вопросов. Встречаются также дополнительные наборы вопросов, продающихся отдельно от игры.

Исторически викторины применяются в качестве развлекательного времяпровождения соревновательного плана. C 1975 года викторины используются как игры в телевизионных программах для соревнований и получения призов. Также викторины повсеместно применяются для развлечений рекламного характера для привлечения аудитории товаров или событий. Кроме того, существуют примеры (IQ-тесты, обучающие викторины) использования ответов игроков для оценки знаний, исходя из сложности вопросов, скорости ответов, тематик и прочих направлений.

С появлением Интернета стало возможным создавать специализированные сайты-викторины, позволив викторине проходить в режиме реального времени. На некоторых из них предусмотрены денежные призы для победителей.

Многие тематические сайты размещают странички с викторинами, призванные развлекать посетителей и популяризовать направленность сайта. Участие в этих викторинах обычно бесплатное, а среди правильно ответивших могут разыгрываться призы.

С расцветом эры смартфонов викторина стала доступна и там, позволяя играть в любом месте и в любое время.

Но что, если викторины использовать как корпоративную разработку. Что если пользователями приложения будут работники, и на их результаты будут смотреть работодатели или менеджеры. Например, работник упорно потрудился и поднял свой уровень английского в приложении. Отчет сразу идет менеджерам, и они могут подбирать проекты под уровень языка работника.

# **1 АНАЛИТИЧЕСКИЙ ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ НА ОСНОВЕ ИЗУЧЕНИЯ ЛИТЕРАТУРНЫХ И ДРУГИХ ИСТОЧНИКОВ ПО ТЕМЕ ДИПЛОМНОЙ РАБОТЫ**

Тема дипломного проекта “Веб-приложение для интерактивного изучения английского языка в сфере IT”.

Начать анализ стоит с понятия веб-приложение. Веб-приложение — это любой сайт с элементами интерактива. Это значит, что посетитель может взаимодействовать с материалом, функциями: нажимать кнопки, заполнять формы, запрашивать прайс, совершать покупки.

Технически это интернет-приложение с архитектурой «клиент-сервер». Чтобы понять принцип, давайте вспомним основные элементы такой архитектуры. Клиентом служит браузер, сервером — [веб-сервер](https://semantica.in/blog/chto-takoe-veb-server.html). Связь происходит посредством сети. Представьте, что web-приложение состоит изначально из страниц с частично либо полностью неопределенным содержимым. Итоговое содержание веб-страниц сформируется тогда, когда конкретный пользователь отправит запрос.

Страницы, которые мы видим в браузере, могут быть статическими и динамическими. Статическая web-страница отображается для всех посетителей одинаково. Как это работает:

1. человек вводит в адресной строке запрос или адрес страницы;
2. браузер отправляет его на веб-сервер;
3. веб-сервер анализирует запрос, определяет, что никаких особых признаков и инструкций нет;
4. веб-сервер отправляет веб-страницу браузеру без изменения каких-либо данных на ней. Например, это новостной материал, общая стандартная информация.

В случае с динамическими страницами схема выглядит так:

1. браузер отправил запрос на веб-сервер. К примеру, при этом поступила информация, что у этого пользователя есть набор признаков, при наличии которых для него нужно показывать определенную информацию, значит страница будет динамической;
2. веб–сервер пересылает ее на сервер приложений, где специальное ПО применит правила и инструкции для добавления особых переменных. Например, человек авторизован в системе. Ему может показаться страница с ФИО и другой релевантной именно для него информацией;
3. сервер забирает готовую веб-страницу, отдает браузеру, который показывает ее посетителю, создавшему запрос.

Для того чтобы понимать, что конкретно нужно разработать, нужно проанализировать текущий рынок подобных приложений. Перебирая сайты в интернете, можно подчеркивать плюсы и минусы подходов и понять, что будет включено в разрабатываемое приложение.

Существует множество вариантов создания викторин в интернете, от создания викторин уже в существующих системах и управление доступом на созданные викторины между людьми в вашей компании, друзьях, родственников до разработки собственной платформы под ключ, где можно навязывать собственную политику разработки и не зависите от разработчиков сторонних систем.

## 1.1 Обзор аналогов

В интернете достаточно веб-приложений, на которых можно пройти викторины по английскому языку. Возьмем, например, первые страницы по запросу "Quizzes english":

* <https://accent-center.ru/ru/education/quizzes/viktoriny-na-angliyskom-yazyke/>
* <https://www.englishclub.com/esl-quizzes/>
* <https://www.usingenglish.com/quizzes/>

Эти сайты не развиваются в интернете, их написали и возможно вносят новый функционал по потребности, они устаревают и не похожи на те приложения куда хочется заходит больше одного раза. В индустрии развлечений через викторины нет монополии, как например в поисковых системах, таких как Google Search - во всем мире и Яндекс Поиск - в странах СНГ, это говорит о том, что индустрия либо конкурентноспособная, либо не привлекает инвесторов, глядя на представленные сайты кажется, их сделали люди-волонтеры или корпорации которые вкладываются в интеллектуальное развитие общества или государственные организации. Дизайн разработан лет десять назад, когда ориентировались не на удобность и понятность, а на функциональную часть - выглядит они весьма функционально, но какие возможности дают эти сайты.

Возьмем первый сайт из списка. На странице (рисунок 1.1) видим список подготовленных викторин.

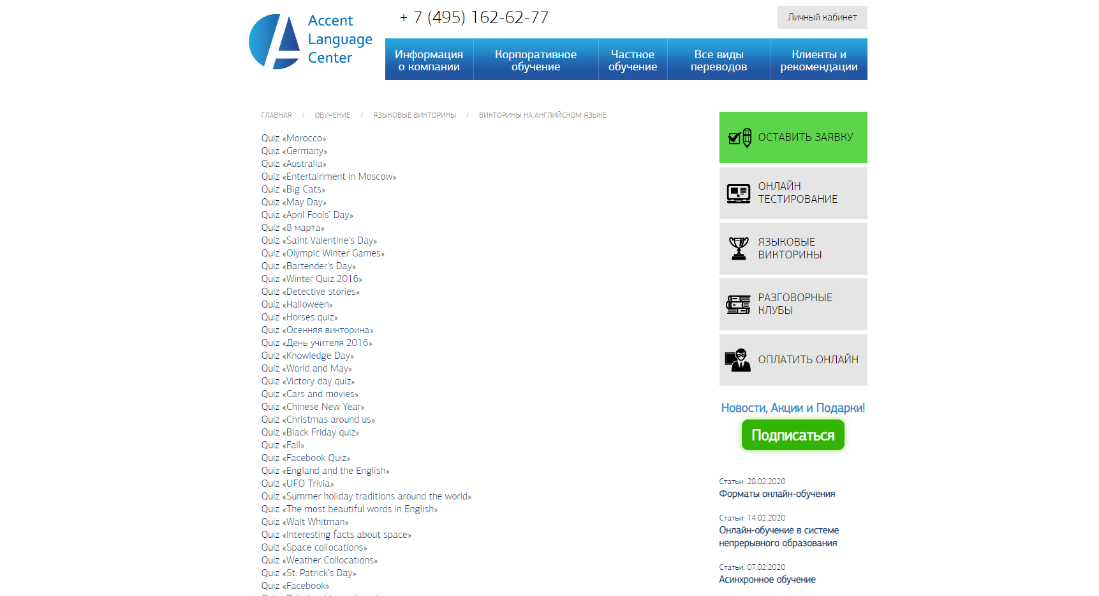


Рисунок 1.1 - Главная страница сайта Accent Language Center

Пройдем тест на сайте “Accent Language Center”. Тут есть тесты разного плана, не только английская грамматика, или правильность речи, так же просто разные тесты, например, на географию на английском языке. Эта реализация выглядит интересно. Пользователю может надоесть учтить грамматику, транскрипции и произношение.

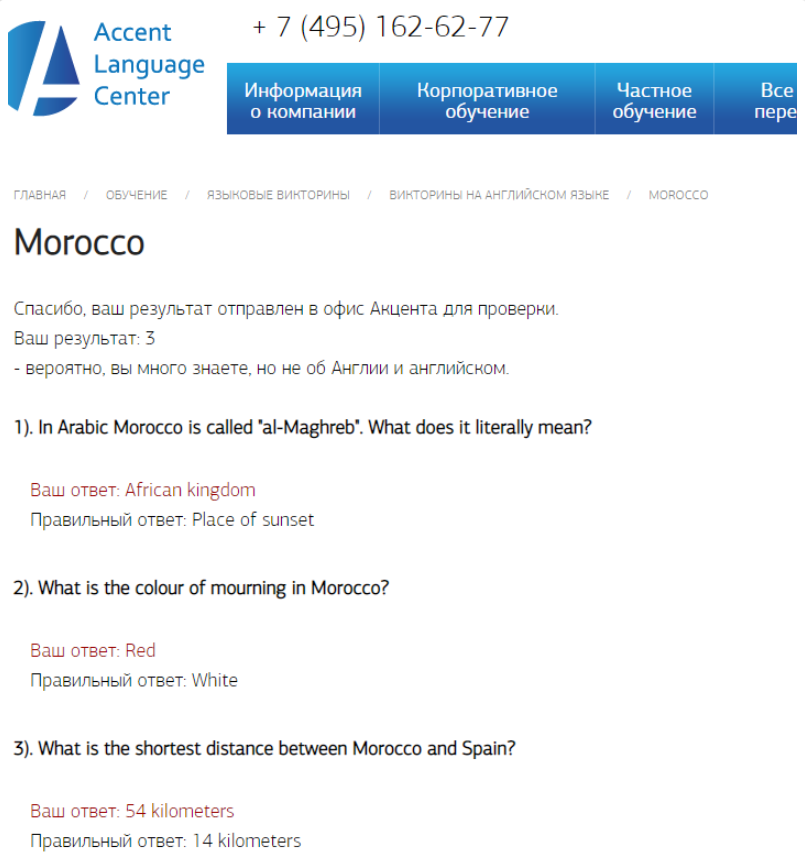


Рисунок 1.2 – Страница результатов сайта Accent Language Center

Эта платформа отображает итоговый результат (рисунок 1.2), ваши ответы и правильные. Никакой истории и статистики тут не найдено. То есть, это одноразовая платформа, зайти пройти несколько викторин по нужной вам теме и выйти.

Следующее веб-приложение предоставляет тесты, специализированные для обучения английскому и его практике. На главной странице мы видим похожий дизайн как на предыдущем сайте (рисунок 1.3).



Рисунок 1.3 – Главная страница сайта ESL Quizzes

После прохождения теста на этой платформе не ощущается, что я перешел на другое приложение, стиль и структура самих тестов почти одинаковая. Нумерация вопроса, вопрос, ответы, результат. Из минусов - возможность изменить результат теста прямо на странице. Посмотрим (рисунок 1.4) на результаты тесты. Набрано 455%. Такой результат не получиться отправить преподавателю, нанимателю и другим заинтересованным лицам.

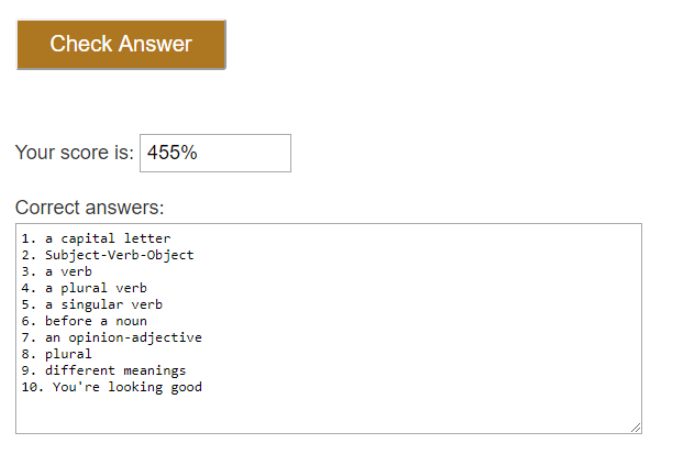


Рисунок 1.4 – Страница результатов сайта ESL Quizzes

Третий аналог приложения с викторинами сразу производит приятные ощущения, сразу виден другой подход к организации приложения и его логики. На рисунке 1.5 показана главная страница.



Рисунок 1.5 – Главная страница сайта UsingEnglish

На рисунке 1.6 показана викторина на сайте “Using English”. Выглядит интересно.

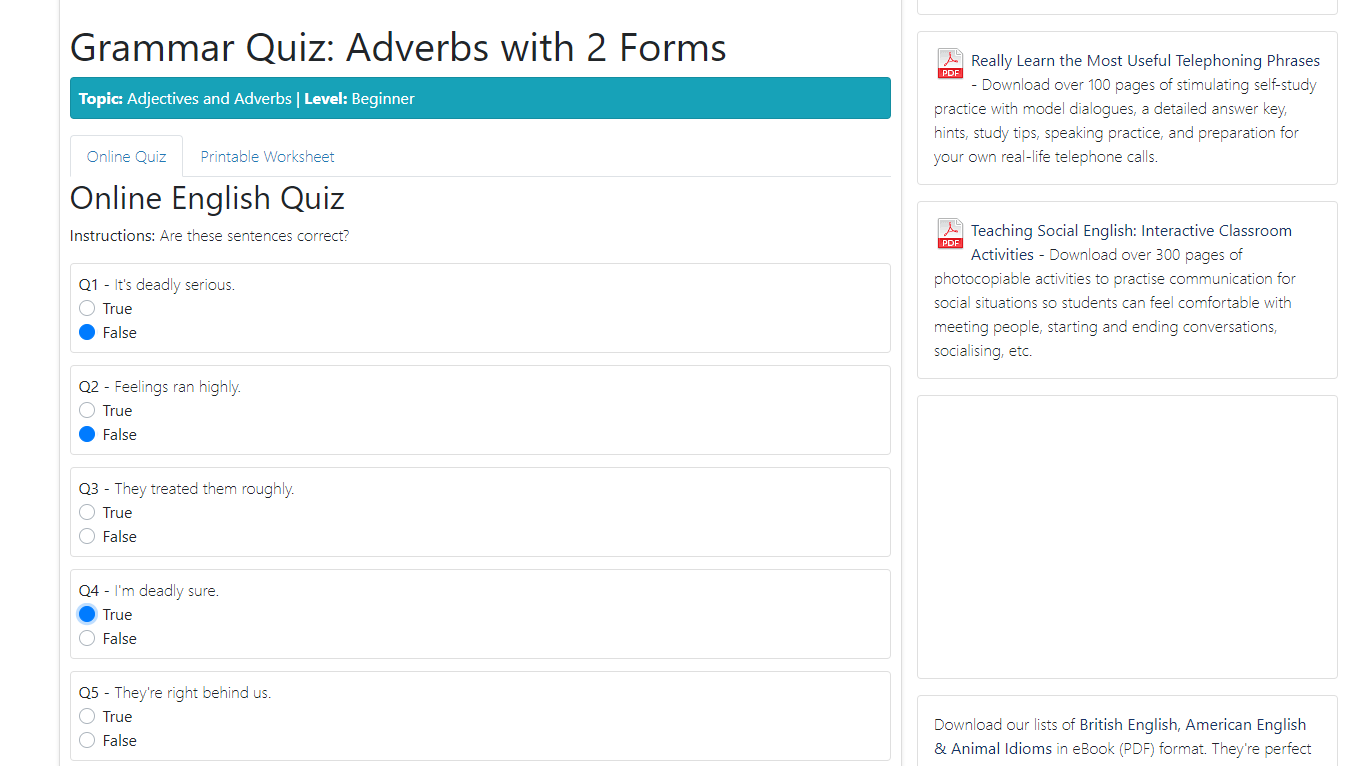


Рисунок 1.6 - Страница результатов сайта UsingEnglish

Положительные моменты:

* возможность выгрузить тесты в формат для печати для того, чтобы раздавать эти тесты не в онлайн формате;
* приятные элементы дизайна: шрифт, цветовая палитра, расстояние между текстом;
* возможность поделиться результатами викторины в Twitter и Facebook.

В результатах не отображают правильные ответы, только неверные — это не относиться к минусам или плюсам. можно сделать это настраиваемым параметром.

Перейдем к более серьёзным решениям с новым дизайном, ориентированием на сферу развлечений и обучения, большим выбором типов викторин, настроек и другого интересного функционала.

Приложение “Quizlet”. Перейдя по ссылке, получаем (рисунок 1.7).

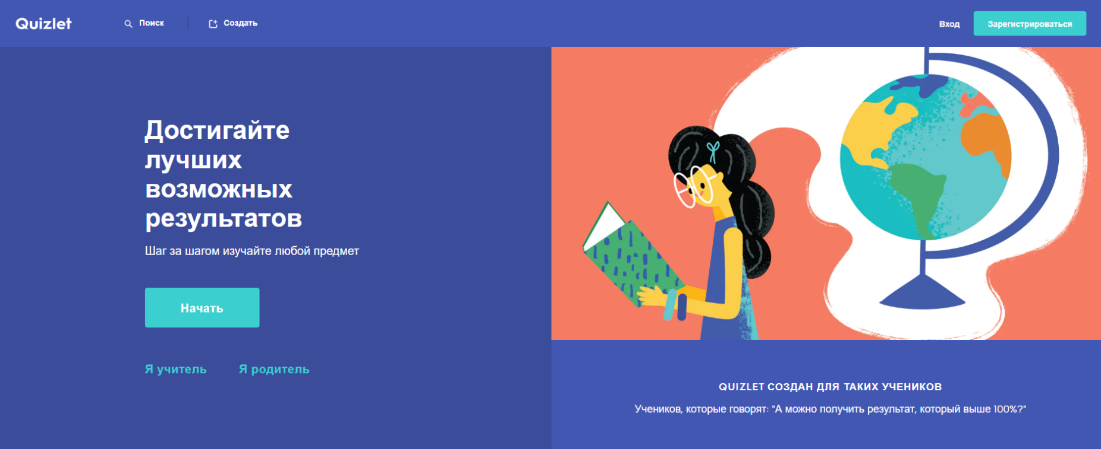


Рисунок 1.7 – Главная страница сайта Quizlet

Сразу можно заметить дизайн, выглядит просто и понятно. Присутствует возможность входа через сторонние сервисы, такие как Google и Facеbook.

Суть этого сервиса в том, что пользователь составляет свои наборы слов или берёт уже готовые и дает им название. На рисунке 1.8 показана страница набора слов.

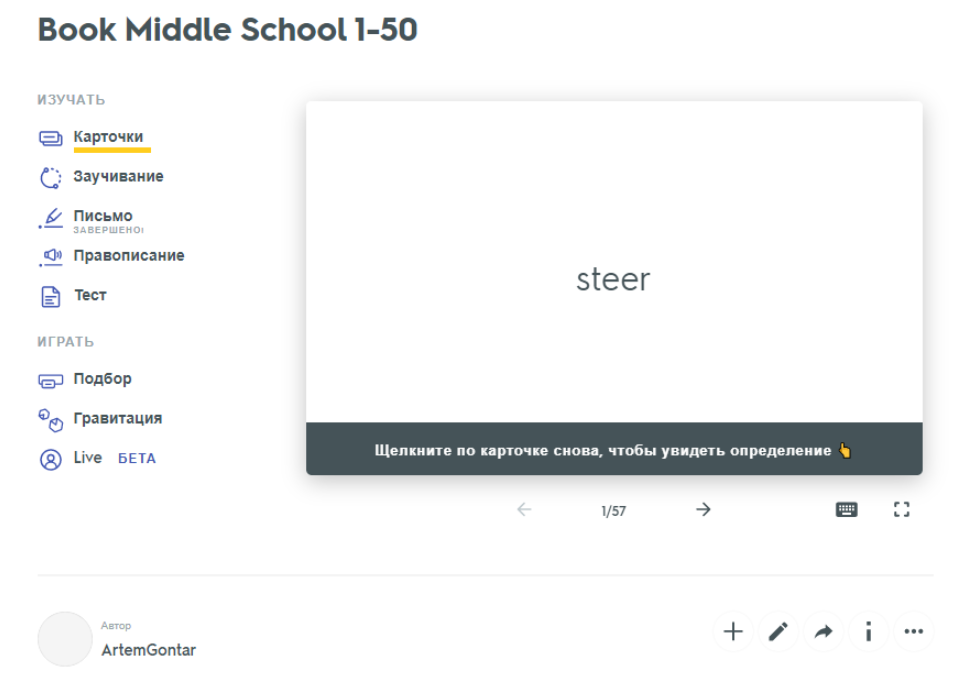


Рисунок 1.8 - Страница набора слов

Если вам понадобиться выучить или повторить этот набор - нужно выбрать режим, в котором удобно учить слова в данный моммент. Например, если вы в такси, можно выбрать письмо или тест, если есть с собой наушники можно выбрать правописание, где вам диктуют слова.

Режимы:

* карточки - просмотр слов на карточках, запоминание слов;
* письмо - показано слово, нужно написать перевод;
* правописание - звучит слово, нужно записать его без ошибок;
* тест - стандартный тест;
* заучивание - комбинированная викторина, в которую входят карточки, письмо, правописание, тест;
* подбор - игровой режим, нужно подобрать пары слово-перевод всем словам на экране;
* гравитация - игровой режим, падают астероиды со словами, чтобы не проиграть нужно как можно быстрее перевести слово на нём;
* live - бета версия.

Есть возможность просмотра истории прохождения, статистика по пользователю, ночной режим (рисунок 1.9).

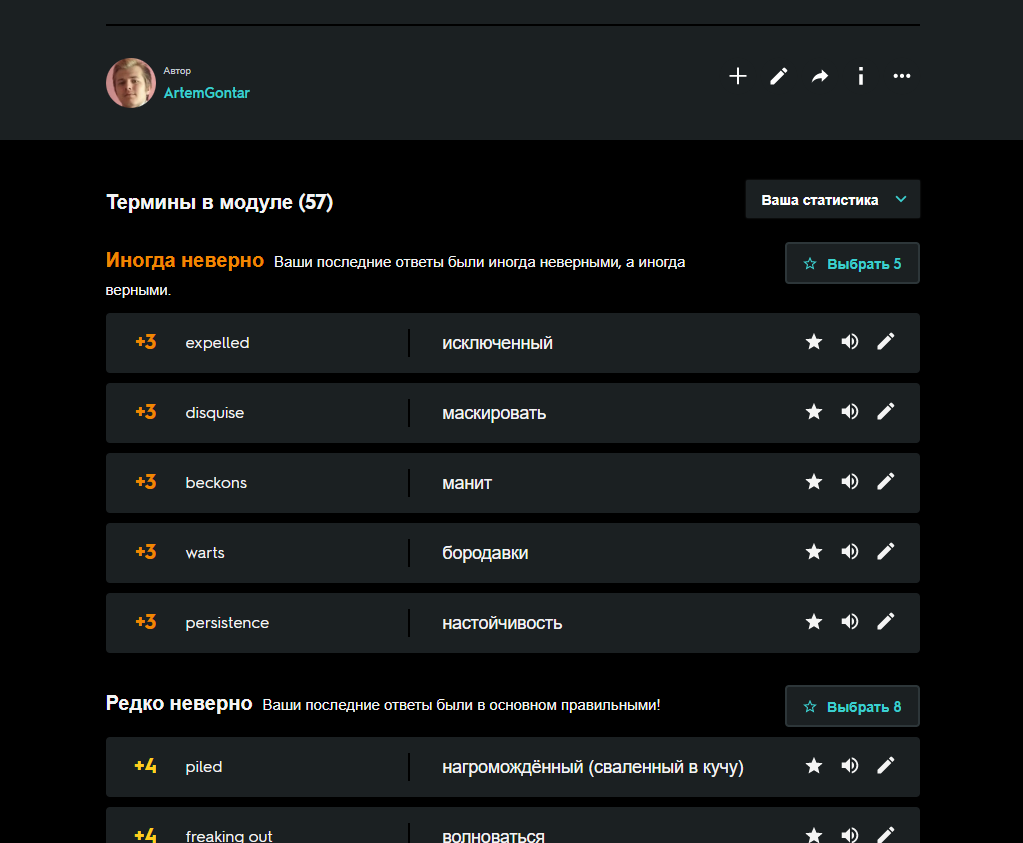


Рисунок 1.9 — Статистика и ночной режим в Quizlet

Не хватает спарингов между пользователями и статистики лучший игроков на платформе для мотивации.

## 1.2 Постановка задачи на дипломное проектирование

Целью дипломного проекта является разработка веб-приложения для интерактивного изучения английского языка в сфере IT.

Приложение должно иметь следующую структуру и функциональность:

* использовать инфраструктуру платформ облачных веб-сервисов;
* иметь базу данных, хранящую информацию: пользователи и их права, викторины. предоставляющую информации о статистике, рейтингах, отчетах;
* графический интерфейс для отображения, ввода, и изменения информации, хранящейся в базе данных;
* производить автоматические отчеты для пользователей с ролью "PM";
* формировать плановую отчетность на базе хранящейся информации;
* реализовывать ролевые политики для пользователей, связанных с подготовкой данных, администраторов, пользователей системы.

В системе будут представлены следующие роли:

* Администратор - чтоб был
* Модератор - управляет викторинами
* Пользователь - проходит викторины
* PM - проектный менеджер, смотрит статистику, отчеты, рейтинги

## 1.3 Анализ подхода к реализации технического задания

Рассмотрим аналоги облачных сервисов, которые существуют и выберем наиболее подходящий под нашу архитектуру и экономическую выгоду. Для рассмотрения возьмём три самые распространенные облачные решения:

* Amazon Web Services. Созданная в 2006 году, облачная платформа Amazon стала первооткрывателем в данной области, благодаря чему завоевала немалый рынок. С постоянными нововведениями и улучшениями на протяжении многих лет, AWS представила более 70 услуг с широким спектром покрытия по всему миру. Серверы доступны в 14 географических регионах. Рыночная доля компании неуклонно растет, во втором квартале 2016 года облачные технологии охватывали 31% рынка.
* Microsoft Azure. Система была запущена в 2010 году и развивается очень быстрыми темпами. Microsoft Azure сейчас представляет собой многогранную сложную систему, которая обеспечивает поддержку множества различных услуг, языков программирования и фреймворков. В составе облака более 60 служб и центров обработки данных в 38 различных географических регионах. В настоящее время Microsoft Azure занимает 11% рынка.
* Google Cloud Platform. Представленная в 2011 году, Google Cloud Platform является самой молодой облачной платформой и, в первую очередь, удовлетворяет потребности поиска Google и Youtube. В настоящее время у компании представлено более 50 услуг и 6 глобальных центров обработки данных. Google Cloud Platform на рынке облачных услуг имеет 5% долю.[1]

Нас интересует четыре составляющие, которые планируется использовать в облачной инфрастуктуре:

* Компьютерные вычисления. EC2 в AWS стал главным синонимом для понятия «масштабируемые вычисления по требованию». Для того, чтобы еще более тщательно планировать и снижать расходы при запуске проектов, компания ввела новые подсервисы, такие как AWS Elastic Beanstalk, Amazon EC2 Container Service. На данный момент AWS поддерживает 7 различных семейств экземпляров и 38 типов экземпляров. Он одновременно предлагает и региональную поддержку, и поддержку зоны. Основа вычислительных систем **Microsoft Azure** – это классические виртуальные машины. Azure Virtual Machine включает 4 различных семейства, 33 типа экземпляров, которые вы можете развернуть в разных регионах. Но поддержка определенной зоны региона пока не поддерживается. **Google Cloud Platform** использует сервис Compute Engine для обработки вычислительных процессов. Одним из главных недостатков является ценообразование, оно менее гибкое по сравнению с AWS и Azure.
* Хранение данных. AWS Simple Storage Service, известный как S3, в значительной степени является промышленным стандартом. S3 создало понятие объектно-ориентированного хранения данных, а для архивации данных был создан отдельный сервис Amazon Glacier. В Azure и Google Cloud существует имеют также достаточно надежные и мощные средства хранения.
* Сеть. В Amazon Web Services вы можете использовать Virtual Private Cloud для создания VPN с настройкой подсети, таблицы маршрутизации, частных диапазонов IP-адресов и сетевых шлюзов. Кроме того, есть сервис Route 53 для реализации веб-службы DNS. Microsoft Azure также предлагает обширные инструменты настройки сетей. Виртуальная сеть (VNET) позволяет установить VPN, настроить публичный IP, подключить гибридное облако, а также активизировать межсетевой экран и DNS. Предложения Google Cloud Platform не столь обширны. Платформа пока имеет только виртуальную сеть Cloud с поддержкой подсетей Public IP, собственный брандмауэр, и необходимые настройки DNS.
* Ценообразование. AWS использует несколько моделей оплаты. По требованию - платёж только за ресурсы и услуги, которые вы используете. Резервирование - необходимое количество ресурсов, которые вы хотите заказать авансом от одного до трех лет и оплачиваете на основе использования. Часто на такие предложения действуют скидки до 75 %. Округление в Amazon Web Services работает на основе часов использования. Microsoft Azure использует более гибкую систему ценообразования, оплата идет за пользование облачными ресурсами, с округлением по минутам. Вы можете использовать готовые подписки MSDN с определенным количеством денежных средств на облачные вычисления, либо оплачивать ресурсы обычным способом с ежемесячным выставлением счетов. Скидки также работают на основе объема заказанных услуг. Google Cloud Platform имеет схожую систему выставление счетов как и Azure, но с округлением использования ресурсов за период в течение 10 минут.

По всем пунктам наиболее подходящим является сервис Amazon Web Services. Для построения нашей архитектуры будем использовать его мощность, гибкость и маштабируемость.

Следующим шагом нужно выбрать брокер сообщений. Тут так же очень много вариантов выбора, будем отталкиваться от брокеров, у которых большие сообщества пользователей и высокая степень поддержки. Сравнивать будем:

* Amazon SQS. Передайте любой объем данных с любым уровнем пропускной способности, не теряя сообщений и не требуя, чтобы другие службы были всегда доступны. С SQS вы можете снять административную нагрузку с эксплуатации и масштабирования высокодоступного кластера обмена сообщениями, заплатив при этом низкую цену только за то, что вы используете.
* Kafka. Распределенный, многораздельный, реплицируемый сервис журнала фиксации. Он обеспечивает функциональность системы обмена сообщениями, но с уникальным дизайном. Ориентирована на Java-приложения.
* RabbitMQ. Предоставляет вашим приложениям общую платформу для отправки и получения сообщений, а ваши сообщения получают безопасное место для жизни до получения.

Amazon SQS идеально подходит для архитектуры в AWS, но это платный сервис с закрытым исходным кодом. Для приложения выберем RabbitMQ для разработки с возможностью переключения в дальнейшей разработке на Amazon SQS путем создания абстракции для брокера, нужно будет изменить только саму реализацию не затрагивая приложение.

## 1.4 Анализ стека технологий для технического задания

В качестве платформы для разработки был выбран .Net Core 3.1 с его фреймворком ASP.NET. Это самая новая версия .Net Core. Разработчики проделали большую работу по оптимизации .Net Framework и перешли в кроссплатформенность убрав все зависимости от Windows, открыли свой код для сообщества тем самым разрешив сообщиству вносить правки в фреймфорк или самостоятельно его модернизировать под их нужды.

.NET Core основана на .NET Framework. Платформа .NET Core отличается от неё модульностью, кроссплатформенностью, возможностью применения облачных технологий, и тем, что в ней произошло разделение между библиотекой CoreFX и средой выполнения CoreCLR.

.NET Core — модульная платформа. Каждый её компонент обновляется через менеджера пакетов NuGet, а значит можно обновлять её модули по отдельности, в то время как .NET Framework обновляется целиком. Каждое приложение может работать с разными модулями и не зависит от единого обновления платформы.[2]

Исходя из постановки задачи нам потребуется база данных для предоставления информации и составлении отчетов и хранения пользователей в системе.

Для аутентификации и авторизации выберем библиотеку IdentityServer - эта библиотека состоит из всевозможных способов аутентификации и авторизации таких как:

* двухфакторная аутентификация;
* OAuth2, Basic, OpenID и другие стандарты аутентификации;
* Jwt, Bearer и другие токены доступа;
* Single Sign-On;
* Client Credentials - позволяет авторизовывать сервисы.

Ссылка на github - <https://github.com/IdentityServer>.

Для согласованности и встроенной безопасности нам навязывают выбрать реляционную базу данных в качестве хранения пользователей, ролей и остальных таблиц для авторизации. в качестве базы данных выберем MySql. MySql встроена как одна из движков внутри Amaozon RDS, что позволяет легко перейти в инфраструктуру Amazon Cloud Services.

MySQL не единственная СУРБД на рынке, но она является одной из самых популярных и уступает только Oracle Database, когда оценивается с использованием таких важных параметров, как количество упоминаний в результатах поиска, профессиональных профилей в LinkedIn и частоты технических дискуссий на интернет-форумах. Тот факт, что многие крупные технологические гиганты полагаются на него, ещё больше укрепляет заслуженную позицию. Вот причины:

* гибкость и простота в использовании - можно изменить исходный код, чтобы он соответствовал собственным ожиданиям, и не нужно ничего платить за этот уровень свободы, включая варианты обновления до расширенной коммерческой версии. Процесс установки относительно прост и не должен занимать более 30 минут;
* высокая производительность - широкий спектр кластерных серверов поддерживает MySQL. Независимо от того, храниться ли большие объёмы данных электронной коммерции или выполняете тяжелую бизнес-аналитику, MySQL может помочь вам с оптимальной скоростью;
* промышленный стандарт - отрасли используют MySQL в течение многих лет, а это означает, что для опытных разработчиков имеются обильные ресурсы. Пользователи MySQL могут рассчитывать на быструю разработку программного обеспечения, а эксперты-фрилансеры, желающие работать в меньшей степени, будут нуждаться в них;
* безопасность - безопасность ваших данных должна быть вашей главной задачей при выборе правильного программного обеспечения СУРБД. С помощью системы доступа и управления учётными записями MySQL устанавливает высокий уровень безопасности. Доступна проверка на основе хоста и шифрование пароля.[3]

В качестве провайдера данных к MySQL выбран EntityFramework Core.

Entity Framework Core представляет собой объектно-ориентированную, легковесную и расширяемую технологию от компании Microsoft для доступа к данным. EF Core является ORM-инструментом (object-relational mapping - отображения данных на реальные объекты). То есть EF Core позволяет работать базами данных, но представляет собой более высокий уровень абстракции: EF Core позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работаем с объектами.

Entity Framework Core поддерживает множество различных систем баз данных. Таким образом, мы можем через EF Core работать с любой СУБД, если для нее имеется нужный провайдер.

По умолчанию на данный момент Microsoft предоставляет ряд встроенных провайдеров: для работы с MS SQL Server, для SQLite, для PostgreSQL. Также имеются провайдеры от сторонних поставщиков, например, для MySQL.

Центральной концепцией Entity Framework является понятие сущности или entity. Сущность определяет набор данных, которые связаны с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их коллекциями.

Любая сущность, как и любой объект из реального мира, обладает рядом свойств. Например, если сущность описывает человека, то мы можем выделить такие свойства, как имя, фамилия, рост, возраст. Свойства необязательно представляют простые данные типа int или string, но могут также представлять и более комплексные типы данных. И у каждой сущности может быть одно или несколько свойств, которые будут отличать эту сущность от других и будут уникально определять эту сущность. Подобные свойства называют ключами.[4]

Так как у нас будет отдельный сервис для аутентификации, авторизации и работы с пользователем можно спроектировать приложение, как микросервисное, что позволяет независимо разворачивать сервисы нашего приложения в облачной архитектуре AWS.

Глядя на постановку задачи объявим еще один сервис Quiz. Он будет хранить и возвращать данные о темах, викторинах, вопросах в викторинах. В качестве базы данных выберем MongoDb.

MongoDВ может возвращать конкретные поля документов и пользовательские JavaScript-функции. Поддерживается поиск по регулярным выражениям. Также можно настроить запрос на возвращение случайного набора результатов. Имеется поддержка индексов. MongoDВ может работать с набором реплик, то есть содержать две или более копии данных на различных узлах. Каждый экземпляр набора реплик может в любой момент выступать в роли основной или вспомогательной реплики. Все операции записи и чтения по умолчанию осуществляются с основной репликой. Вспомогательные реплики поддерживают в актуальном состоянии копии данных. В случае, когда основная реплика дает сбой, набор реплик проводит выбор, которая из реплик должна стать основной. Второстепенные реплики могут дополнительно являться источником для операций чтения.

Система масштабируется горизонтально, используя технику сегментирования (англ. sharding) объектов баз данных — распределение их частей по различным узлам кластера. Администратор выбирает ключ сегментирования, который определяет, по какому критерию данные будут разнесены по узлам (в зависимости от значений хэша ключа сегментирования). Благодаря тому, что каждый узел кластера может принимать запросы, обеспечивается балансировка нагрузки.[5]

В приложении будет использоваться MongoDriver, как провайдер данных для MongoDb.

Ссылка на гитлаб: <https://github.com/mongodb/mongo-csharp-driver>.

Следующий микросервис будет отвечать за статистику, будет собирать данные о пользователе и его прохождении викторин и выдавать показатели и сервис, который запускается по времени для отправки дневных, недельных, месячных отчетов проектным менеджерам.

Тут есть два варианта либо не использовать базу данных для этого сервиса и получать данных из других сервисов, либо сделать отдельную базу данных и хранить там данные уже подготовленный к выгрузке.

Из плюсов первого подхода не нужно иметь базу данных что позволяет сэкономить память в процессе разработки и деньги, потому что не придется платить за Amazon RDS Instance. Из плюсов второго подхода, скорость формирования статистики и отчетов, н нужно будет забирать данные из других сервисов и обрабатывать их в момент выполнения запроса. Выберем второй вариант, так как ожидание может повлиять на желание пользователей оставаться в системе.

Для мониторинга состояния системы будет создан отдельный микросервис Monitoring. Микросервис занимается тестовым опрашиванием других сервисов и нацелен на проверку, отвечает ли сервис или нет. На рисунке 2.1 показан его примерный интерфейс.

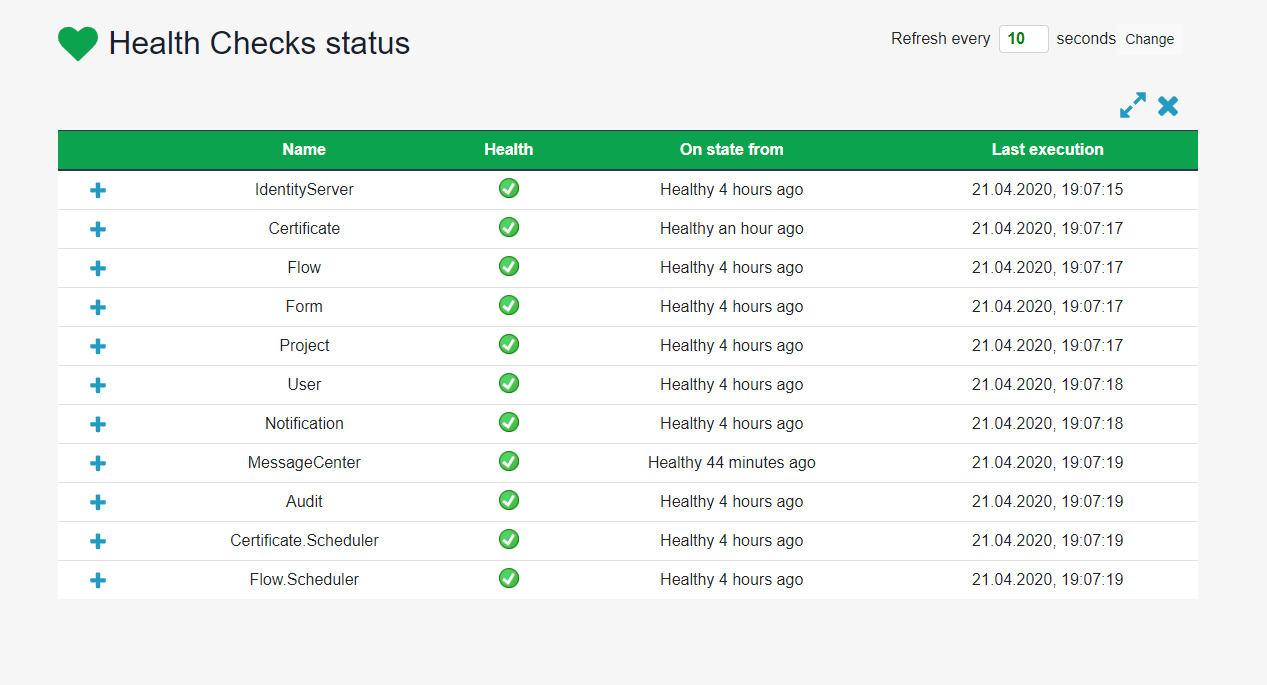


Рисунок 2.1 — Интерфейс микросервиса Monitoring

В качестве CI/CD процессов выбран AWS CodeDeploy. AWS CodeDeploy – это полностью управляемый сервис для автоматизации развертывания программного обеспечения с использованием различных вычислительных сервисов, таких как Amazon EC2, AWS Fargate, AWS Lambda и локальных серверов. AWS CodeDeploy упрощает быстрый выпуск новых возможностей, помогает избежать простоев во время развертывания и берет на себя все сложности при обновлении приложений. AWS CodeDeploy можно использовать для автоматизации развертывания программного обеспечения, устраняя необходимость в подверженных ошибкам ручных операциях. Сервис масштабируется в соответствии с вашими потребностями.[6]

Для логирования в приложение будет интегрирован Elactic Stack. Он состоит из трех основных компонент:

* Logstash – это утилита для обработки лог событий из различных источников, с помощью которой можно выделить поля и их значения в сообщении, также можно настроить фильтрацию и редактирование данных. После всех манипуляций Logstash перенаправляет события в конечное хранилище данных. Утилита настраивается только через конфигурационные файлы;
* Elasticsearch – это решение для полнотекстового поиска, но с дополнительными удобствами, типа легкого масштабирования, репликации и прочего, что сделало продукт очень удобным и хорошим решением для высоконагруженных проектов с большими объемами данных. Elasticsearch является нереляционным хранилищем (NoSQL) документов в формате JSON, и поисковой системой на базе полнотекстового поиска Lucene. Аппаратная платформа — Java Virtual Machine, поэтому системе требуется большое количество ресурсов процессора и оперативки для работы;
* Kibana позволяет искать\брать данные и запрашивать статистику из базы данных elasticsearch, но основе ответов строятся множество красивых графиков и дашбоардов. Также система имеет функционал администрирования базы данных elasticsearch, в последующих статьях мы рассмотрим более подробно данный сервис. А сейчас покажем пример дашбоардов по межсетевому экрану Check Point и сканеру уязвимостей OpenVas, которые можно будет построить;

В качестве механизма разворачивания микросервисов выбран AWS Elastic Container Service, что позволяет разворачивать приложение в Docker-контейнерах. Подобно виртуальной машине докер запускает свои процессы в собственной, заранее настроенной операционной системе. Но при этом все процессы докера работают на физическом host сервере деля все процессоры и всю доступную память со всеми другими процессами, запущенными в host системе. Подход, используемый докером, находится посередине между запуском всего на физическом сервере и полной виртуализацией, предлагаемой виртуальными машинами. Этот подход называется контейнеризацией.

Для хранения и версионирования исходного кода выбрана платформа AWS CodeCommit. AWS CodeCommit - это полностью управляемый сервис [по управлению исходным кодом](https://aws.amazon.com/ru/devops/source-control/), предназначенный для хостинга защищенных репозиториев на основе Git. Он упрощает совместную работу команд с кодом в безопасной системе с высокой масштабируемостью. С сервисом CodeCommit не требуется поддерживать собственную систему управления исходным кодом или беспокоиться о масштабировании соответствующей инфраструктуры. С помощью CodeCommit можно безопасно хранить любые ресурсы, от исходного кода до исполняемых файлов. Сервис эффективно интегрируется с существующими инструментами Git.

Планируется 2 окружения. Development – для разработки и тестирования функционала. Со следующими настройками:

* MySql: 1Gb Ram, 50Gb hard disk;
* MongoDb: 1Gb Ram, 50Gb hard disk, 0 shards, 0 replica-set;
* RabbitMq: 1Gb Ram, 50Gb hard disk.

Staging – для демонстрации разработанного и протестированного функционала функционала. Со следующими настройками:

* MySql: 2Gb Ram, 100Gb hard disk;
* MongoDb: 2Gb Ram, 100Gb hard disk, 0 shards, 0 replica-set;
* RabbitMq: 2Gb Ram, 50Gb hard disk.

Версии используемой инфраструктуры:

* MySql – 10.2;
* MongoDb – 4.0.9;
* ActiveMq – последняя стабильная;
* Docker – 19.0.3.

Для разработки графического интерфейса выбран механизм рендеринга html-разметки Razor Page для микросервиса IdentityServer, и Angular для Quiz WebApi и Statistic WebApi.

Razor Pages предоставляют технологию, альтернативную системе Model-View-Controller. Razor Pages позволяют создавать страницы с кодом Razor, которые могут обрабатывать запросы. В некоторой степени эта функциональности напоминает работу веб-форм, которые представляли страницу с расширением aspx и имели файл логики на C#, связанный с данной страницей. В этом плане Razor Pages представляют альтернативу стандартной модели MVC для построения приложения.

Angular представляет фреймворк от компании Google для создания клиентских приложений. Прежде всего он нацелен на разработку SPA-решений (Single Page Application), то есть одностраничных приложений. В этом плане Angular является наследником другого фреймворка AngularJS. В то же время Angular это не новая версия AngularJS, а принципиально новый фреймворк.

Angular предоставляет такую функциональность, как двустороннее связывание, позволяющее динамически изменять данные в одном месте интерфейса при изменении данных модели в другом, шаблоны, маршрутизация и так далее.

Одной из ключевых особенностей Angular является то, что он использует в качестве языка программирования TypeScript. Поэтому перед началом работы рекомендуется ознакомиться с основами данного языка. Но мы не ограничены языком TypeScript. При желании можем писать приложения на Angular с помощью таких языков как Dart или JavaScript. Однако TypeScript является основным языком для Angular.[7]

# **РАЗРАБОТКА АЛГОРИТМОВ И ТЕХНОЛОГИИ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ**

## 2.1 Архитектура программного комплекса

Посмотрим на архитектуру приложения на рисунке 2.2.

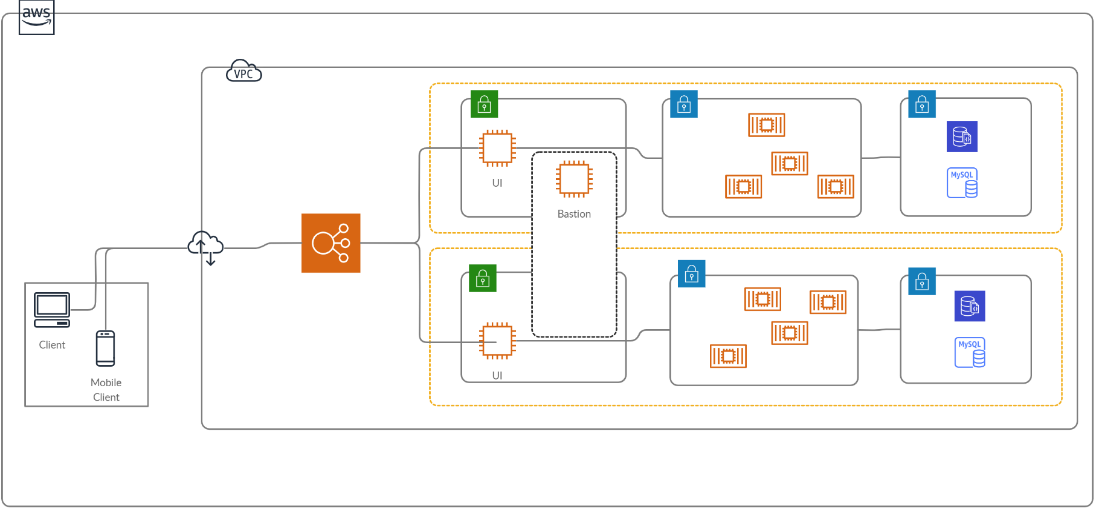


Рисунок 2.2 — Архитектура приложения

Путь от клиента до сервиса, когда пользователь заходит на сайт можно описать следующим образом:

1. запрос проходит через internet gateway - компонент VPC, который обеспечивает связь между экземплярами в вашем VPC и Интернетом;
2. попадает на elastic load balancer – служащий для балансировки нагрузки трафика HTTP и HTTPS и обеспечивает расширенную маршрутизацию запросов. В данной архитектуре для балансировки между avialibility zone - зоны доступности, если один экземпляр выходит из строя, второй экземпляр в другой зоне доступности мог обрабатывать запросы;
3. проходит в public subnet - контур, который имеет выход в интернет и обращается к пользовательскому интерфейсу приложениях;
4. если запрашиваемый контент статический, то возвращается ответ с разметкой или данными, если динамический, то запрос переходит в private subnet - подсеть, в которой находятся микросервисы. К этой подсети есть доступ через public subnet bastion host;
5. сервисы обращаются к базам данных, которые находятся в полной изоляции от интернета и обращатся туда можно только из приложения, чтобы избежать утечки персональных данных;
6. полностью сформированный ответ возвращается клиенту.

Этот путь можно выразить на диаграмме последовательностей следующим образом (рисунок 2.3)

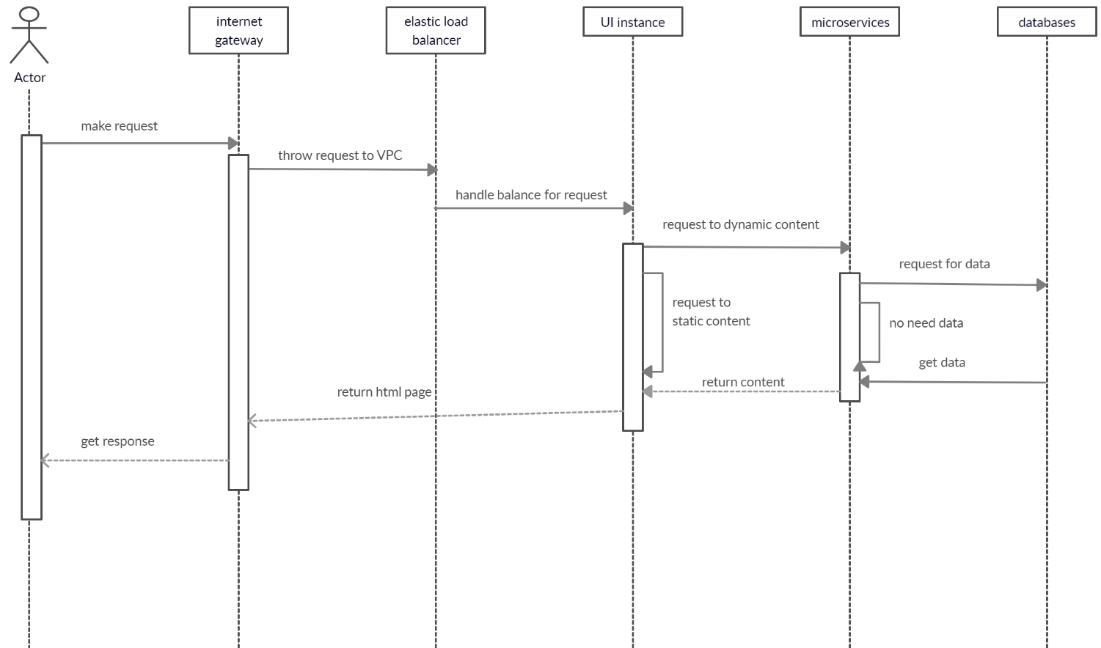


Рисунок 2.3 — Путь запроса клиента по компонентам архитектуры

Рассмотрим подробнее структуру приложения в отрыве от Anazon Web Services. На рисунке 2.4 продемонстрирована карта сервисов.

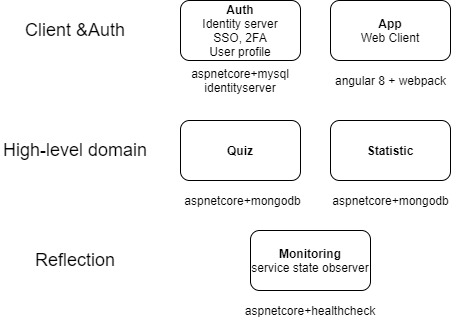


Рисунок 2.4 — Карта сервисов

На карте показано какие будут созданы сервисы, их стек и ответственность. И так рассмотрим каждый из них:

* сервис “Identity” отвечает за аутентификацию и авторизацию пользавтеля и профиль пользователя, в его стек входят asp.net core, реляционная база данных MySql, двухфакторная аутентификация и технология Single Sign-On;
* сервис “Web Client” отвечает за пользовательский интерфейс приложения, в его стек входит angular framework, css, html, webpack;
* сервис “Quiz” отвечает за бизнес логику викторин, их разделов и вопросов, в его стек входит asp.net core, не реляционная база данных mongoDB.
* сервис “Statistic” отвечает за бизнес логику статистики и отчетов, в его стек входит asp.net core, не реляционная база данных mongoDB.
* сервис “Monitoring” отвечает за проверку состояния всех сервисов, в его стек входит asp.net core и health check технология.

По отдельности сервисы описаны и выявлен их стек, теперь стоит рассмотреть их взаимодействие между собой и базами данных. Посмотрим на диаграмму (рисунок 2.5).

Изображение выглядит как текст, карта, знак, улица

Автоматически созданное описание

Рисунок 2.5 — Взаимодействие микросервисов

Микросервисы Identity, Quiz, Statistic, которые взаимодейсвуют между собой через брокера сообщений RabbitMQ, что позволяет сервисам оставаться независимыми и маштабируемыми. Также эти серисы взаимодействуют со своими базами данных. Микросервис Monitoring не имеет базы данных и не слушает сообщения брокера, он обращается по http протоколу ко всем сервисам в системе и проверяет их работоспособность.

Приступим к детальному рассмотрению сервисов и постороению архитекуры решения. В каждом микросервисе будет 5 проектов:

1. Persistence – в котором описывается инфраструктура сервиса, такая как провайдеры базы данных, брокер сообщений и др.;
2. Domain – описываются классы домменной модели, проект не имеет никаких зависимостей;
3. Application – описывает бизнес логику вокруг домменной модели, зависит от Domain и Persistence, используется паттерн программирования CQRS;
4. Api – описывает интерфейс взаимодействия приложения, используется тип проекта Asp.Net Core WebApi;
5. UnitTests – модульное тестирование приложения.

Посмотрим на диаграмму классов домменной модели в микросервисах identity (рисунок 2.6), quiz (рисунок 2.7), statictic (рисунок 2.8).

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 2.6 — Диаграмма домменных классов Identity

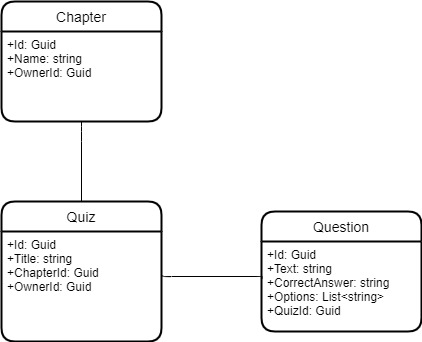


Рисунок 2.7 — Диаграмма домменных классов Quiz

Рисунок 2.8 — Диаграмма домменных классов Statistic

## 2.2 Алгоритм работы программного комплекса

Рассмотрим возможности каждой роли на диаграмме использований. Общие функие которые есть у каждого пользователя это аутентификация и регистрация. Регистрация предполагает ввод электронной почты, пароля и подтверждения пароля. Аутентификация предпологает ввод электорнной почты и пароля для проверки личности и авторизации пользователя в системе. Все пользователи после регистрации имеют роль “Пользователь”. Рассмотрим его функции на диаграмме (рисунок 2.4).

Изображение выглядит как карта, текст

Автоматически созданное описание

Рисунок 2.4 — Диаграмма использование для роли “Пользователь”

У данного пользователя есть возможность:

* просмотра викторин, разделов;
* прохождения викторин;
* просмотра результатов;
* просмотра своего профиля;
* просмотра своей статистики.

Роль по умолчанию может изменить модератор или администратор. Рассмотрим диаграммы использования для “Модератора” на рисунок 2.5.

Изображение выглядит как карта, текст

Автоматически созданное описание

Рисунок 2.5 — Диаграмма использование для роли “ Модератор”

Для пользователя с ролью “Модератор” добавлены функции:

* управление викторинами и разделами;
* прохождения викторин;
* просмотра результатов;
* просмотра своего профиля;
* просмотра своей статистики.

Рассмотрим диаграммы использования для “Администратора” на рисунке 2.6.

Изображение выглядит как карта, текст

Автоматически созданное описание

Рисунок 2.6 — Диаграмма использование для роли “Администратор”

Для пользователя с ролью “Администратор” добавлены функции:

* управление пользователем – полный доступ;
* управление викторинами – полный доступ.

Рассмотрим диаграмму использоавния (рисунок 2.7) пользователя с ролью “Проектный менеджер”.

Изображение выглядит как карта, текст, фотография, стол

Автоматически созданное описание

Рис 2.7 — Диаграмма использование для роли “Проектный менеджер”

У данного пользователя есть возможность:

* просмотра отчетов;
* просмотра статистики;
* просмотра профиля пользователей;
* просмотра статистики пользователей.

## 2.3 Проектирование базы данных и структуры программного комплекса

Проектирование базы данных один из самых важных этапов построения архитектуры приложения. Самое сложное определить структуры базы данных, больше всего это относиться к реляционным базам данных, так как не реляционные не имеют как таковой структуры. Начнем с системных таблиц, которые для пользователя не несут логики. На рисунке xx показана диаграмма таблиц, отвечающая за механизм аутентификации и авторизации сервисов IdentityServer. В контексте IdentityServer сервисы называют клиенты.

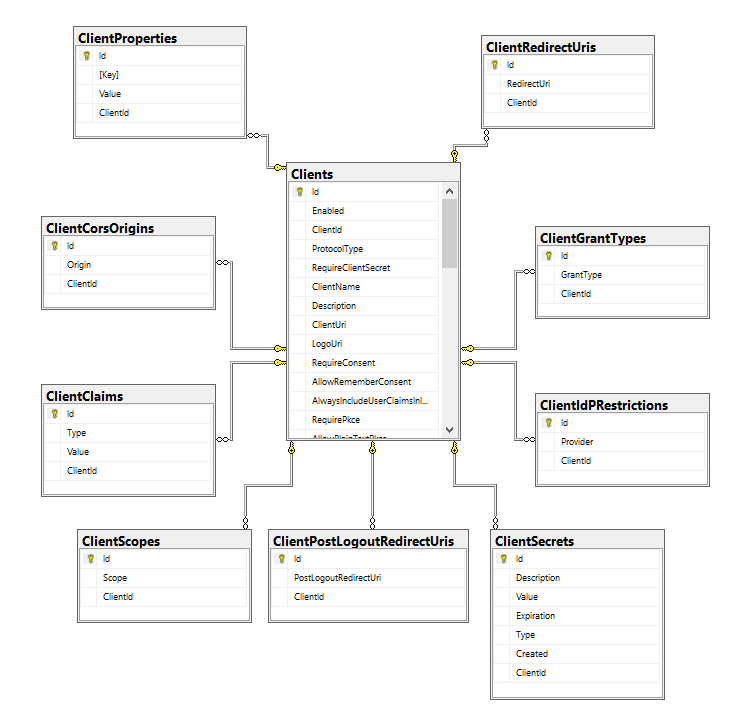


Рис 13 — Схема базы данных для управления клиентами IdentityServer

Следующая системная диаграмма отображает диаграму ресурсов в контексте IdentityServer. Это сервисы, к которым IdentityServer может выдать доступ на обращение.

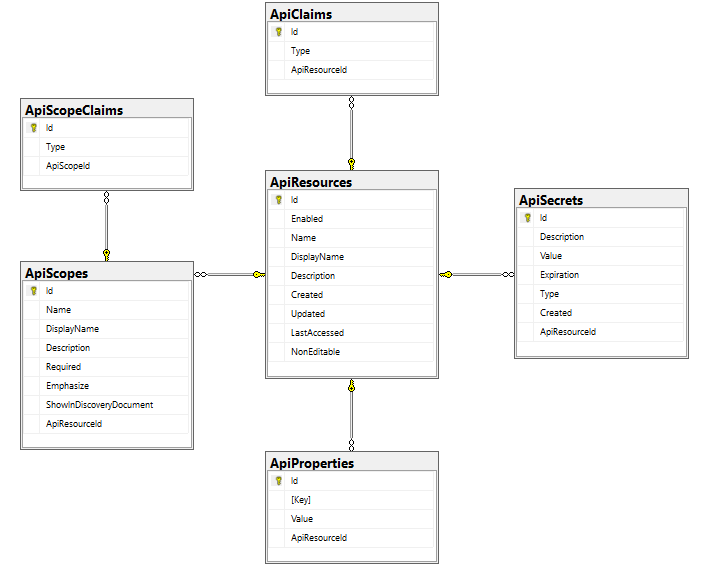


Рис 14 — Схема базы ресурсов IdentityServer

Следующая диаграмма отображает таблицы пользователей AspNetUsers, таблицу ролей AspNetRoles, их связку AspNetUserRoles и другие системные таблицы.

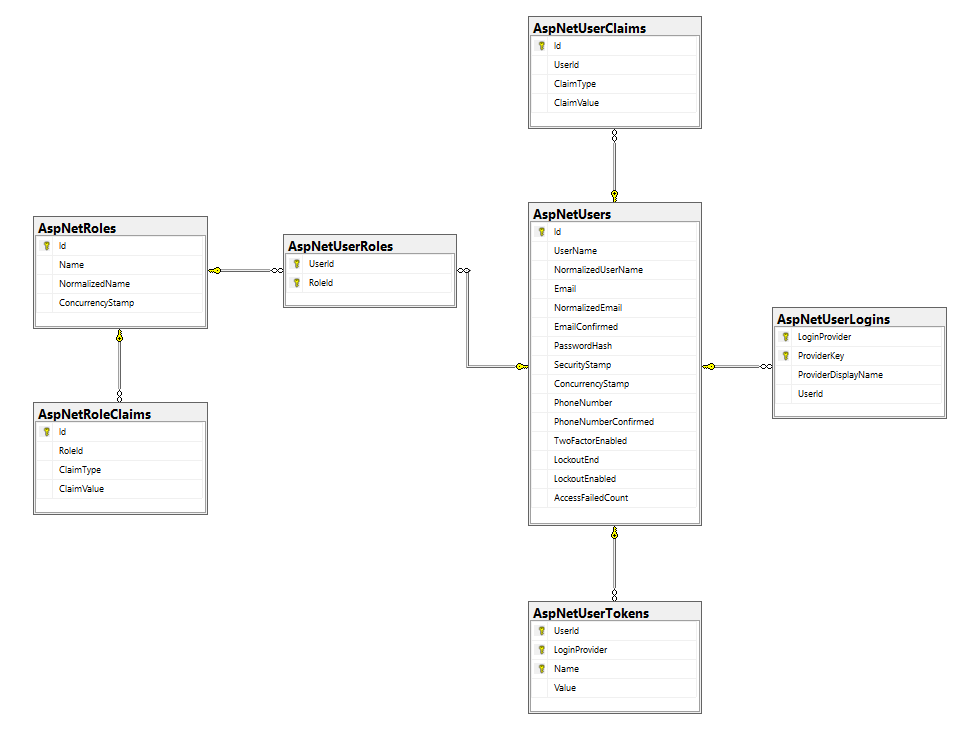


Рис 15 — Схема базы пользовательских данных

# **ЗАКЛЮЧЕНИЕ**

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. <http://la.by/blog/sravnenie-uslug-oblachnyh-provayderov-microsoft-azure-aws-ili-google-cloud>

2. <https://docs.microsoft.com/en-us/dotnet/core/>

3. <https://www.hostinger.ru/rukovodstva/shto-takoje-mysql/>

4. <https://metanit.com/sharp/entityframeworkcore/>

5. <https://ru.wikipedia.org/wiki/MongoDB>

6. <https://aws.amazon.com/ru/codedeploy/>