**Введение**

В ходе дипломного проекта необходимо разработать программное средство для учебно-образовательных викторин. Данное программное средство является комплексным и состоит из приложения для мобильной операционной системы Андроид и веб-приложения для использования с помощью поддерживаемого интернет-браузера.

Основной целью дипломного проекта является разработка двух приложений-частей программного средства для учебно-образовательных викторин.

Для достижения основной цели поставлены следующие задачи:

* создать приложение для мобильной операционной системы Андроид, способное отображать и загружать викторины из сети интернет, дать пользователю возможность интерактивно взаимодействовать с викторинами в ходе работы с приложением;
* создать веб-приложение для использования через интернет-браузер вне зависимости от операционной системы. Главным требование для совместного взаимодействия частей программного средства для учебно-образовательных викторин является интернет-подключение. Веб-приложение должно предоставлять пользователям функционал для создания викторин;
* разработать эффективный алгоритм, выполняющий поставленные задачи в полном объёме;
* спроектировать базу данных для приложений таким образом, чтобы она удовлетворяла правилам нормализации.

Пояснительная записка к дипломному проекту включает в себя сведения по разрабатываемому программному средству для учебно-образовательных викторин, она содержит пять основных разделов:

* постановка задачи;
* проектирование задачи;
* разработка программного средства;
* тестирование;
* применение.

Раздел постановки задачи в свою очередь содержит базовое исследование предметной области, выделение основных (ключевых) абстракций, обоснование необходимости компьютерной обработки информационных средств и процессов. Также в него входит описание информационной базы решаемой задачи, общее определение основных функций и задач, которые должна решать разрабатываемая программа.

Раздел проектирования задачи отражает этап проектирования задачи. Он включает в себя такие пункты как:

* проектирование пользовательского интерфейса, основываясь на общепринятых для данной области методах;
* определение основных путей взаимодействия с пользовательским интерфейсом;
* разработка модели, которая даёт возможность понять структуру разрабатываемой системы, процесс её функционирования, область применимости;
* сбалансирование требований;
* определение алгоритма решения задачи.

Раздел разработки программного средства содержит описание исходного кода программы, а также описание реализаций задачи с помощью конкретных выбранных для использования инструментов разработки, языков программирования и вспомогательных информационных средств. В данном разделе описывается как физическая структура разработанной программы в целом, так и её отдельных модулей, типичных для программ подобного типа и размера.

Раздел тестирования включает в себя проведение проверки работоспособности системы при наличии определенной совокупности исходных данных и при использовании различных режимов эксплуатации, то есть различных их конфигураций. В данном разделе также присутствует список действий, проводимых для определения работоспособности системы.

В последнем разделе – разделе применения – рассмотрены базовые сценарии использования, а также описано предполагаемое назначение и использование приложений, область применения разрабатываемого программного средства, пошаговое описание базового использования (руководство пользователя). Также в нём перечислены необходимые программные и аппаратные ресурсы для запуска и функционирования приложения, а также последующей комфортной работы с программным средством.

* 1. Постановка задачи

1. Описание предметной области

Всё большее значение приобретает самостоятельная работа обучающихся, создающая необходимые условия для формирования готовности использовать различные средства информации для поиска необходимого знания. Последнее становится особенно актуальным в связи с развитием новых форм образовательного процесса как выстраивания индивидуальной траектории самообучения.

Происходит переориентирование самостоятельной работы обучающихся с традиционной цели усвоения знаний, приобретения умений и навыков, опыта творческой деятельности на развитие способности к самоуправлению собственной учебно-познавательной деятельностью, активно преобразующего отношения к получаемой учебной информации [1].

Таким образом, основным признаком самостоятельной работы обучающихся становится не то, что они занимаются ею без непосредственного участия преподавателя, а то, что в их учебно-познавательной деятельности сочетается функция перевода информации в знания и умения с функцией управления этой деятельностью.

Значительная роль в развитии у будущих специалистов способности к самоуправлению учебно-познавательной деятельностью отводится учебным материалам – средствам обучения, которые используются педагогом и обучающимися как инструменты достижения образовательных целей. Учебные материалы являются ведущим элементом системы средств самостоятельной работы обучающихся как по учебным дисциплинам, так и по предметным областям, связанных с получением развивающей информации.

Существует большое количество видов носителей учебной информации, использование электронных образовательных ресурсов позволяет в значительной степени удовлетворить, формата самостоятельного освоения предлагаемой преподавателем информации, способствует формированию информационной компетентности в таких областях как:

* умение самостоятельно выбирать источники информации;
* принятие решений в условиях неопределённости;
* осуществление самоконтролирующих действий;
* приобретение опыта интерактивного общения.

Учащиеся должны быть способны к самостоятельному конструированию собственного образовательного маршрута, хорошо ориентироваться в информационном пространстве, мыслить творчески, уметь принимать решения и нести за них ответственность. Этому способствует не только классическое преподавание дисциплин в учебных заведениях, но также и использование вспомогательных учебно-игровых или учебно-образовательных материалов. Это позволяет должно способствовать формированию способности адаптироваться в изменяющемся мире и находить пути решения различных жизненных и учебных задач.

В процессе обучения учащимся следует приобретать определенные качества, которые базируются на собственном опыте и знаниях, полученных ранее, то есть быть компетентными во многих вопросах. Формирование у учащихся компетентности является на сегодняшний день одной из наиболее актуальных проблем образования.

При использовании программных учебно-образовательных средств внимание акцентируется на результатах обучения, в качестве которых рассматриваются не сумма усвоенных знаний, умений и навыков, а способность учащихся продуктивно действовать в различных жизненных ситуациях.

Инновационные технологии взаимосвязаны, взаимообусловлены и составляют определенную дидактическую систему, направленную на воспитание таких ценностей как открытость, честность, доброжелательность, сопереживание, взаимопомощь и обеспечивающую образовательные потребности каждого ученика в соответствии с его индивидуальными особенностями.

Среди многообразия учебно-образовательных материалов, соответствующих поставленным целям особое место занимают викторины.

Викторина – учебно-образовательное состязание, заключающееся в ответах на устные или письменные вопросы из различных областей знания.

Викторины как особый вид занятий обладают исключительной занимательностью и являются богатой почвой для формирования стойкого познавательного интереса и развития логического мышления.

Одним из важных требований к выбору методов обучения является необходимость активизации учебно-образовательной деятельности учащегося. Активная мыслительная и практическая деятельность в учебном процессе является важным фактором повышения освоения изучаемого материала.

При создании викторин в общем учитываются следующие требования:

* сочетание посильности и определенной трудности;
* постепенное усложнение заданий;
* включение операций сравнения, сопоставления, обобщения, ведущих к активизации аналитической деятельности учащихся;
* связанность заданий с интересами учащихся.

Исходя из вышеизложенного, предметная область программного средства для учебно-образовательных викторин является комплексной.

1. Обзор существующих аналогов

Для разрабатываемого программного средства для учебно-образовательных викторин отсутствуют известные полноценные аналоги, но существуют системы, обладающие схожей функциональностью. Вот некоторые из них:

* Quizlet;
* Kahoot.

Не являясь заменой разрабатываемому приложению, Quizlet предлагает возможности по созданию тестов, где ученики будут выбирать верные ответы из предложенных, сопоставлять изображения и информацию или вписывать собственные варианты.

Лучший способ приступить к работе в Quizlet – это выполнить поиск по учебным модулям, созданным другими пользователями. Ученики и учителя постоянно создают новые модули, поэтому существует большое количество модулей на разные темы.

Пользоваться данным приложением очень легко – в интерфейсе можно разобраться буквально за пару минут. Пример интерфейса приложения Quizlet показан на рисунке 1.1.



Рисунок 1.1 – Пример интерфейса Quizlet

Однако в данном приложении полностью отсутствует поддержка русского языка, возможности бесплатной версии сильно ограничены. Также отсутствует функционал передачи викторин другим пользователям с возможностью многократного скачивания и хранения на стороне сервера.

Kahoot представляет собой игровую обучающую платформу, используемую в качестве образовательной технологии в школах и других учебных заведениях. Этот учебно-образовательный контент представлен в виде игр, которые являются викториной со множеством выборов, которые позволяют генерировать пользователей и доступны через веб-браузер. Пример использования платформы Kahoot показан на рисунке 1.2.

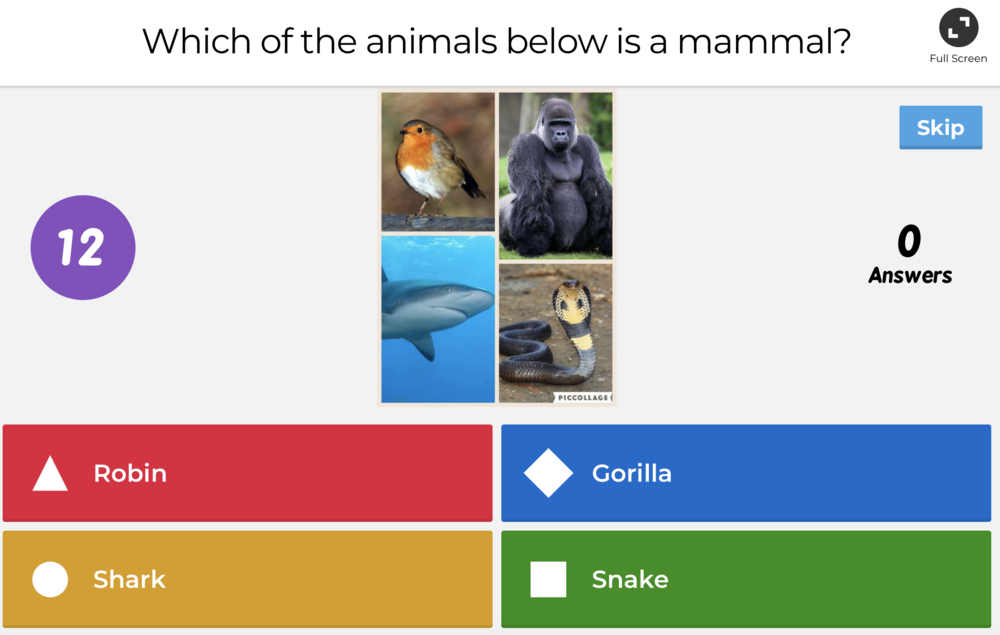


Рисунок 1.2 – Пример использования Kahoot

Учащиеся со своего компьютера или смартфона переходят по ссылке, ранее сгенерированной сервисом, вводят код игры, который генерируется автоматически. Далее учащиеся вводят свое имя и когда все учащиеся вошли под своим именем в игру, учитель запускает тест, нажав на кнопку начала игры.

Сервис Kahoot обладает рядом преимуществ перед Quizlet: наличие мультиплатформенности и бесплатность. Минусы Kahoot заключаются в его узкой специализации: он предназначен для использования в аудиториях и классных комнатах для единовременного пользования большим количеством людей и имеет соответствующие инструменты, направленные на интеграцию с учащимися, отображение их результатов и нацеленный в том числе на социальную составляющую.

Таким образом, как сервис Kahoot, так и приложение Quizlet – это инструменты проверки усвоения знаний некоторой группой людей и взаимодействия учащихся с преподавателем, не приспособленный для удобного составления полноценных викторин, но не полноценный аналог разрабатываемого приложения, имеющие разрабатываемые функции и особенности.

Основные плюсы разрабатываемого программного средства для учебно-образовательных викторин:

* полностью бесплатное обеспечение доступа ко всем функциональным возможностям;
* наличие русского языка;
* использование без необходимости прохождения процедур регистрации;
* ориентированность на самостоятельное прохождение викторин;
* доступ к прохождению викторин на мобильном устройстве с ОС Андроид, создание викторин в веб-приложении, доступном в любом браузере.

Предполагаемые недостатки разрабатываемого программного средства:

* отсутствие социальной составляющей;
* учебно-образовательный материал представлен в виде викторин, а не интерактивных игр;
* невозможность прохождения викторин с мобильных устройств на базе операционных систем Windows Phone или iOS.

Исходя из всего вышеперечисленного, программное средства для учебно-образовательных викторин имеет целый ряд характерных отличий от подобных приложений, что положительно сказывается на востребованности и применимости.

1. Информационная база задачи

Информационная база задачи является совокупностью упорядоченной информации, которая используется для функционирования системы и делится на внешнюю и внутреннюю машинную базу. Она включает в себя входные, выходные и постоянные данные.

Входные данные – это данные величины, которые задаются до начала работы алгоритма или определяются динамически во время его работы. Входные данные берутся из определенного набора объектов и используются для решения определенных задач.

Выходные данные – результат работы определенного алгоритма или процесса, представленный в виде изображения, переменной, текста, логического выражения, документов, изменений в базе данных и так далее.

Постоянные данные – это данные, которые сохраняются после завершения работы программы в течение длительного периода времени и многократно используется для расчетов и других операций в ходе работы программы.

Информационная база программного средства для учебно-образовательных викторин будет представлена в виде входных, выходных и постоянных данных.

В качестве входных данных будет выступать:

* список викторин, полученный с сервера;
* список викторин, предварительно загруженный и сохраненный;
* данные о выборе пользователем конкретной викторины;
* выбор вариантов ответов на представленные вопросы.

Постоянные данные представлены в виде:

* информация о загруженных викторинах;
* результат прохождения викторин пользователем;
* список пройденных викторин.

Выходные данные:

* информация о результатах прохождения теста, отображаемая пользователю после прохождения викторины;
* изменения в базе данных после совершения действий;
* результат поиска викторины;
* загруженная с сервера информация о викторине.

Чтобы создать стабильную интерактивную систему, необходимо понять взаимодействие посредников во внешней среде. Основные участники внешней среды:

* веб-приложение – клиент-серверное приложение, в котором пользователь взаимодействует с сервером при помощи браузера;
* браузер – прикладное программное обеспечение, предназначенное для просмотра веб страниц;
* база данных – структурированный набор информации, хранимый в памяти вычислительной машины;
* персональный компьютер – устройство или система, способная выполнять заданную, чётко определённую, изменяемую последовательность операций;
* смартфон с ОС Андроид – мобильный телефон, дополненный функциональностью карманного персонального компьютера;
* пользователь – лицо, которое использует действующую систему для выполнения конкретной функции.

Взаимодействуя между собой и используя перечисленные типы данных, основные участники внешней среды обеспечивают полноценное функциональное использование системы по назначению.

1. Функциональное назначение

При последующей разработке программного средства для учебно-образовательных викторин следует принимать во внимание изложенные в данном пункте требуемые функциональные возможности системы.

В рамках дипломного проекта поставлена задача разработать программное средство для учебно-образовательных викторин, которое будет состоять из:

* мобильного приложения – созданное для мобильной платформы Андроид, его основной целью является само отображение викторин и предоставление возможностей для их прохождения. Сами викторины необходимо загружать из сети интернет с сервера через это же мобильное приложение, по умолчанию вместе с ним в комплекте планируется поставлять только викторину для демонстрации основных функциональных возможностей по прохождению викторин и последующего сохранению результата на примере;
* веб-приложения – приложения, доступного для использования с помощью интернет-браузера (требует наличие сети интернет). В случае программного средства – системы для учебно-образовательных викторин – будет состоять из двух частей: клиентской и серверной, работающих вместе.

Android-приложение необходимо для скачивания и запуска викторин, оно должно обладающее всеми необходимыми операциями для полнофункционального взаимодействия с викторинами;

Клиентская часть веб-приложения, должна:

* отображать меню по созданию вопросов для викторин, ответов на созданные вопросы;
* осуществлять занесение вопросов в викторины;
* давать пользователю возможность редактирования уже созданных викторин;
* давать возможность редактирования дополнительных сведений о викторине;
* в удобной для пользования форме отображать список уже созданных викторин;
* осуществлять проверку на валидность перед отправкой запроса к серверной части о добавлении викторины в базу данных.

Серверную часть веб-приложения, принимающую и обрабатывающую информацию о созданной викторине с последующей записью в базу данных и выдачей нужного файла викторины по запросу исходящего из Android-приложения.

Android-приложение следует протестировать не менее чем на трёх устройствах с операционной системой версии не ниже 5.0 в различных условиях.

Клиентская часть веб-приложения должна быть протестирована во всех популярных на данный момент браузерах последних версий (Google Chrome, Mozilla Firefox, Opera, Safari). Допускаются минимальные различия во внешнем виде страниц, не влияющие на функционирование системы.

Серверная часть перед привязкой к клиентской части должна быть проверена с помощью статических средств проверки кода на предмет возможных ошибок. Также требуется провести работы по устранению возможных утечек информации.

Также серверная часть будет использоваться мобильным приложением (к ней планируется делать запросы по сети интернет, отправляющиеся с мобильного устройства) с целью получения списка доступных для загрузки викторин, непосредственной загрузки викторин и последующего их запуска, как в случае, если викторина изначально шла вместе с разрабатываемым приложением и не требовала загрузки. Веб-приложение предоставляет возможности по созданию викторин в режиме онлайн, а также последующему редактированию уже существующих в системе викторин, создавая копии на их основе, но не изменяя их.

После запуска Андроид-приложения пользователь сможет получить доступ, как к справочной информации, так и к списку викторин, что является одной из основополагающих функций для разрабатываемого мобильного приложения. При наличии интернет-подключения список викторин пополняется и теми, что ещё не были загружены на устройство с сервера, таким образом, данные для отображения будут получены сразу из нескольких источников, но отображены в едином потоке информации.

При выборе из списка викторины, которая ещё не была загружена на устройство, в области уведомлений появится соответствующее уведомление, отображающее прогресс скачивания выбранной викторины. Это позволит пользователю приложения в реальном времени отслеживать загрузку и видеть процесс, а после его успешного окончания начать прохождение.

У каждой викторины должна быть указана информация о ней, вид которой динамически изменяется после изменения внутреннего состояния викторины (заменяются надписи и пиктографическая иконка). Здесь пользователю приложения на выбор следует предлагать несколько вариантов операций с любой из викторин в списке. Например, при одиночном нажатии на карточку с викториной, пользователь должен перейти на основной игровой экран. При долгом же нажатии требуется произвести показ диалога, спрашивающего о том, хочет ли пользователь удалить выбранную из списка викторину или нет. Это необходимо для предотвращения случайных нажатий, ведущих к непроизвольным потерям данных. При утвердительном ответе запись о викторине должна исчезнуть из базы данных, а соответственно потеряется вся информация о её прохождении, после чего совершенные изменения должны мгновенно отобразиться (например, викторина должна исчезнуть из списка).

На главном игровом экране будут присутствовать такие элементы как: название викторины, текущий вопрос, прогресс по прохождению викторины, прогресс по количеству набранных правильных ответов, кнопки для выбора вариантов ответов.

Последовательность выбора вопросов для отображения – случайная, что не позволит использующему приложение запоминать места для нажатия на экран и их порядок для успешного прохождения викторины. Через некоторое время (это необходимо, чтобы пользователь успел посмотреть, был ли выбран им правильный вариант ответа или нет) будет осуществляться переход к следующему вопросу. Для этого вся игровая область должна приводиться в изначальное состояние, заполняться обновленными данными надписи на кнопках выбора ответов и сам текст вопроса. Также должны будут обновиться трекеры (отслеживатели) количества пройденных вопросов и количества верных ответов на представленные вопросы. После окончания викторины результат о лучшем прохождении сохранится в базу данных. Если викторина ещё ни разу не была пройдена, то следует делать соответствующую отметку в заметном для пользователя месте.

Программа после дачи ответов на все предоставленные вопросы осуществляет переход на экран отображения результатов прохождения викторины.

Серверная часть веб-приложения будет отвечать за выполнение сразу нескольких функций, таких как:

* хранение созданных викторин в базе данных;
* серверная часть логики по редактированию и сохранению новых учебно-образовательных викторин, в том числе, повторная валидация (проверка на подлинность) данных, переданных из клиентской части;
* отдача всей необходимой информации о викторине (название, список вопросов, справочная информация) мобильному приложению для последующего использования в нём;
* отдача в качестве ответа на запрос готового документа (веб-страницы) клиентской части.

Следовательно, серверная часть будет являться многофункциональной, так как ей следует выполнять необходимые функции одновременно как для мобильной программы, так и для клиентской части веб-приложения.

Клиентская часть должна отвечать за отображение документа, переданного серверной частью веб-приложения, а также за корректную обработку пользовательского ввода и последующую валидацию, отправку запросов к серверу с целью получения необходимых данных.

Функция валидации полученных данных должна присутствовать и в клиентской, и в серверной части. Существует ряд веб-приложений, выполняющих проверку данных на корректность только на одной из перечисленных частей приложения. Основным преимуществом такого подхода является сокращение времени на разработку. Однако, выполнение функции валидации в обеих частях веб-приложения позволяет:

* существенно уменьшить количество ошибок, допущенных пользователями в связи с наличием человеческого фактора путём информирования о допущенных ошибках с предложением внести правки в введенные данные;
* устранить возможность повреждения/создания невалидных (неверных с точки зрения системы) данных злоумышленниками;
* сократить количество возможных ошибок в ходе выполнения разрабатываемого мобильного приложении, так как оно берёт данные о викторинах из серверной части веб-программы.

Одной из самых важных функций клиентской части веб-приложения программного средства для учебно-образовательных викторин является предоставление удобного и интуитивно понятного интерфейса для пользования системой.

Интуитивность – восприятие, понимание особенностей бессознательно, неосознанно. Это одно из главных требований к интерфейсу любого программного продукта, предназначенного для пользования как опытными пользователями системы, так и впервые использующими продукт.

Отсутствие интуитивности негативным образом влияет на пользование программным средством, а, следовательно, основной функционал приложения должен быть размещен таким образом, чтобы даже незаинтересованный в длительном чтении справочной информации пользователь на бессознательном уровне понимал основные концепции, заложенные в интерфейс программы.

Корректное исполнение проанализированных функциональных возможностей программным средством является залогом качества системы.

**Проектирование задачи**

**2.1 Разработка пользовательского интерфейса**

Ясность является первой и самой главной задачей любого интерфейса. Чтобы интерфейс эффективно помогал людям добиваться своих целей, он должен обладать следующими характеристиками:

* быть узнаваемым;
* иметь очевидное для пользователя назначение;
* предсказуемость во время взаимодействия;
* отсутствие путанности, двузначности в элементах;

Благодаря ясности, у пользователей возникает ощущение уверенности и готовность продолжать работу с интерфейсом, интерфейсы существуют, чтобы люди могли в удобной форме взаимодействовать с приложением. Через интерфейс проясняется взаимосвязь, пользователя и предоставляемых приложением функций [2].

Интерфейсы выполняют конкретные задачи, их эффективность измерима. Однако они могут выходить за рамки чисто практического применения и рассматриваться как отдельный опыт для пользователя. Отсутствие раздражающих в интерфейсе факторов повышает статистику пользования.

Человек чувствует себя максимально комфортно, когда чувствует контроль над собой и своим окружением. Непродуманный интерфейс лишает пользователей контроля: людям навязываются неожиданные взаимодействия, запутанные процессы и непредсказуемые результаты, чего следует всячески избегать при создании графических интерфейсов.

Необходимо дать пользователям ощущение контроля: регулярно информировать их о состоянии системы, показать причинно-следственные связи и подсказать, чего ожидать на каждом этапе.

Очень легко переборщить и добавить в интерфейс больше слоев, чем нужно: перегруженные кнопки, хром, графика, опции, предпочтения, окна, вложения и прочее. Из-за этого внимание пользоваться переключается на управление элементами интерфейса, вместо того, чтобы перейти к содержимому приложения.

Необходимо стремиться к максимально естественному интерфейсу, интерфейс должен быть настолько незаметным, чтобы у пользователя сохранялось ощущение прямого управления объектом его внимания.

В основе каждого экрана интерфейса должно лежать одно значимое для пользователя действие. Такой интерфейс проще изучать и использовать, а также его проще дополнить и расширить (при необходимости).

В основе любого экрана в дизайне должно лежать одно сильное действие, объясняющее существование. Когда в основе экрана лежит одно основное действие, добавляются несколько второстепенных действий ,  но они должны выглядеть как второстепенные и не забирать внимание пользователя на себя.

Необходимо уменьшать визуальный вес второстепенных элементов интерфейса, либо показывать их после того, как основное действие достигнуто.

Все экраны должны иметь следующий шаг, подсказка пользователям следующего шага повышает очевидность интерфейса.

Людям комфортно, когда все происходит согласно ожиданиям. В этом контексте элементы дизайна должны выглядеть соответственно их поведению. Это означает, что внешний вид элемента должен говорить о том, как он себя поведет при взаимодействии. Если элемент выглядит как кнопка, то он должен иметь функциональность кнопки.

Элементы интерфейса не должны выглядеть согласованными, если их поведение не согласовано и наоборот: если элементы работают одинаково, то и выглядеть должны одинаково.

Сильная визуальная иерархия  –  очевидность для пользователя, в каком порядке следует рассматривать визуальные элементы на экране.

Визуальная иерархия задает последовательность и плавно направляет взгляд пользователя с одного элемента интерфейса на другой. При слабой визуальной иерархии интерфейс выглядит перегруженным и непонятным  –  взгляд всегда находится в напряжении.

Сложно сохранить сильную визуальную иерархию, если интерфейс постоянно меняется: когда все выделено жирным, считайте что ничего не выделено. Как только в иерархической системе появляется новый элемент, дизайнеру приходится редактировать визуальные веса всех остальных элементов, чтобы снова добиться сильной иерархии. Большинство людей вообще не замечают визуальной иерархии, но она легко способна усилить (или ослабить) дизайн.

Организация элементов в дизайне  –  это иллюстрация их взаимосвязей. Необходимо группировать похожие элементы, располагать их в пространстве так, чтобы были понятны их связи. Когда контент правильно организован, снижается когнитивная нагрузка на пользователя: ему не нужно думать, как элементы связаны между собой.

Цвет не может быть единственной отличительной чертой объектов. Если пользователь будет долго взаимодействовать с дизайном (например, читать) , то следует  использовать приглушенный фоновый цвет. Также необходимо показывать на каждом экране только то, что нужно в данный момент.

В идеальном интерфейсе пользователю не требуется помощь – все и так понятно. Помощь должна быть органично встроена в интерфейс: скрыта, когда все идет нормально, но легко доступна при необходимости. Если система не предлагает помощь автоматически, пользователю приходится самому искать ответ на свой вопрос.

Чтобы помочь пользователям быстро освоиться в дизайне, важно качественно спроектировать пустое состояние –  состояние, в котором ничего еще не произошло. На этом этапе важно направить пользователя, подсказать, как использовать дизайн максимально эффективно. Очень часто у пользователей бывают проблемы именно на этапе пустого состояния.

Следуя вышеперечисленным правилам построения пользовательского интерфейса, был создан макет сайта, изображенный на рисунке 2.1.

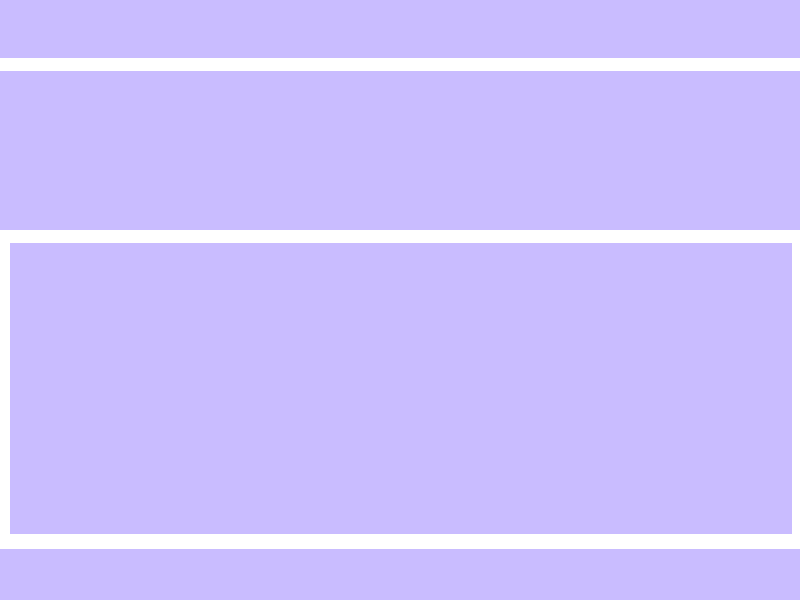


Рисунок 2.1 – Макет пользовательского интерфейса веб-приложения

Пользовательский интерфейс условно разделён на четыре блока. Верхний из них (header) предназначен для ссылок на наиболее часто посещаемые страницы сайта, а также для названия веб-приложения. Нижний блок (footer) предназначен для отображения справочной информации о сайте. Остальные блоки используются непосредственно для контента сайта [3].

Для Андроид-приложений разметка определяет визуальную структуру пользовательского интерфейса. Установить разметку можно создав элементы управления программно в коде Java, либо объявив элементы интерфейса напрямую в XML.

Макет пользовательского интерфейса для разрабатываемого мобильного приложения изображен на рисунке 2.2.

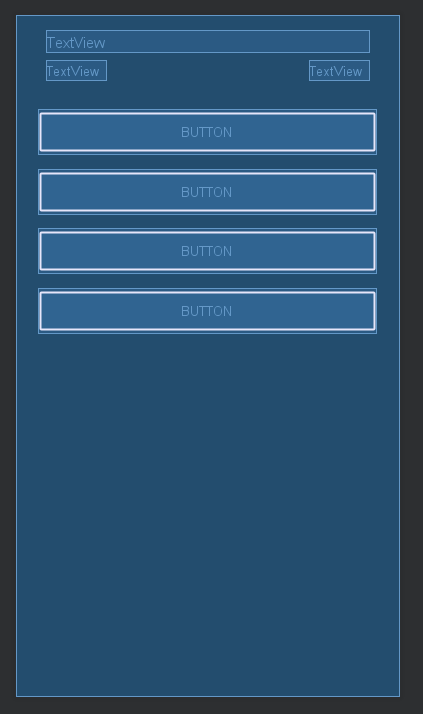


Рисунок 2.2 – Макет пользовательского интерфейса приложения

Макет создан с учётом того, что для приложений на базе операционной системы Андроид существуют специальные единицы измерения dp и sp – это относительные единицы, которые используются для размеров экранных компонентов и текста. Благодаря их использованию экранные элементы выглядят примерно одинаково на экранах разных размеров.

Пиксель, не зависящий от плотности (dp) – относительная единица измерения, которая берет за основу плотность экрана устройства. Примерно cоответствует одному пискелю на устройстве с плотностью экрана 160 точек на дюйм. Отношение меняется в соответствии с плотности экрана, но не в строгой пропорции и зависит от категории экрана, а не точного значения плотности. Рекомендуется применять для корректного отображения элементов на устройствах с различной плотностью экрана.

Пиксель, не зависящий от масштабирования (sp) – как и dp, зависит от плотности экрана, но результат будет масштабироваться в соответствии с настройками размера шрифта пользователя. Рекомендуется для задания размеров шрифтов, чтобы текст подстраивался как под плотность экрана, так и под настройки, заданные пользователем

**2.2 Выбор и обоснование инструментов разработки**

Язык программирования – формальный язык, предназначенный для записи компьютерных программ.

За исключением неполнофункциональных (узкоспециализированных, не предназначенных для написания программ в виду отсутствия базовых необходимых функций) и эзотерических (языков программирования, разработанных с целью исследования границ возможностей разработки, для доказательства какой-либо идеи) языки программирования делятся на следующие категории:

* аспектно-ориентированные – основанная на идее разделения функциональности для улучшения разбиения программы на модули (AspectC++, AspectLua, AspectJ);
* структурные – представляют программы в виде иерархической структуры блоков (Basic, Pascal, Fortran);
* процедурные – языки, в которых последовательно выполняемые операторы можно собрать в подпрограммы, то есть более крупные целостные единицы кода (Lua, C, Golang);
* логические – основаны на теории и аппарате математической логики с использованием математических принципов резолюций (Mercury, Prolog);
* объектно-ориентированные – основаны на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования (PHP, Java, JavaScript, C++);
* функциональные – языки, в которых процесс вычисления трактуется как вычисление значений функций в математическом понимании последних (в отличие от функций как подпрограмм (F#, Haskell, Lisp);
* мультипарадигмальные – язык, возможности которого изначально предполагалось унаследовать от нескольких, чаще всего не родственных языков (C#, Delphi, Erlang).

Не все из вышеперечисленных типов языков программирования можно использовать для разработки мобильного приложения. Существует большое количество решений для использования различных языков при программировании приложения для операционной системы Андроид, к ним относятся такие языки как:

* Java;
* Kotlin;
* C/C++;
* Python;
* Basic;
* Lua.

Языки Lua, Python и Basic используются реже остальных из-за своей неэффективности на данной платформе. Для них существуют специальные системы, позволяющие запускать приложения, написанные на этих языках на платформе Android, но официально они не поддерживаются и не могут в полной мере получить доступ ко всем возможностям этой мобильной операционной системы, из-за чего данные языки чаще всего даже не рассматриваются при выборе языка программирования для ОС Android.

С/С++ – это более низкоуровневые языки, которые также поддерживаются при условии использовании Java NDK или Android NDK. Это позволяет писать нативные для платформы приложения, что может быть полезным при создании игр или других ресурсоёмких программ.

Написанный код будет запускаться не через Java Virtual Machine, а в обход её, то есть непосредственно через саму систему, что даёт больше контроля над такими элементами, как память, сенсоры, жесты, а также даёт возможность более гибкого управления Android-устройством, ресурсами [4]. Это также означает, что при написании кода будут использоваться библиотеками, также написанные на C или C++.

Следовательно, данные языки программирования сложны в настройке и написании кода, его написание занимает намного большее время и требует большего уровня знаний, так как работа ведётся непосредственно с самой системой, где допустить ошибку намного проще. Поэтому рекомендуется использовать эти языки только для написания модулей программы, где необходимо быстро производить сложные операции, такие как обработка и рендеринг графики, больших объёмов или сложных в строении данных, видео и сложных 3D-моделей.

Язык программирования Kotlin в своей типичной реализации является надстройкой над языком Java, созданной для сокращения количества кода, повышения удобства его написания и ограждения от некоторых ошибок.

В свою очередь, этот язык является менее читабельным по сравнению с языком Java, редактирование уже написанного кода может занимать больше времени в связи с временными затратами на разбор существующего кода, а также многие продвинутые концепции языка сложны и редки в использовании.

В связи с вышеперечисленным, для разработки мобильного приложения будет использоваться язык программирования Java. – строго типизированный объектно-ориентированный язык программирования. Изначально новый язык программирования назывался Oak и разрабатывался для бытовой электроники, но впоследствии был переименован в Java, расширил спектр своих возможностей и стал использоваться для написания приложений и другого программного обеспечения.

Программы, написанные на Java, обычно считаются медленнее, чем написанные на языке C. Тем не менее, скорость выполнения программ, написанных на языке Java, была существенно улучшена с выпуском так называемого JIT-компилятора в дополнение к другим особенностям языка для поддержки лучшего анализа кода. Пример кода на языке Java можно увидеть на рисунке 2.3.

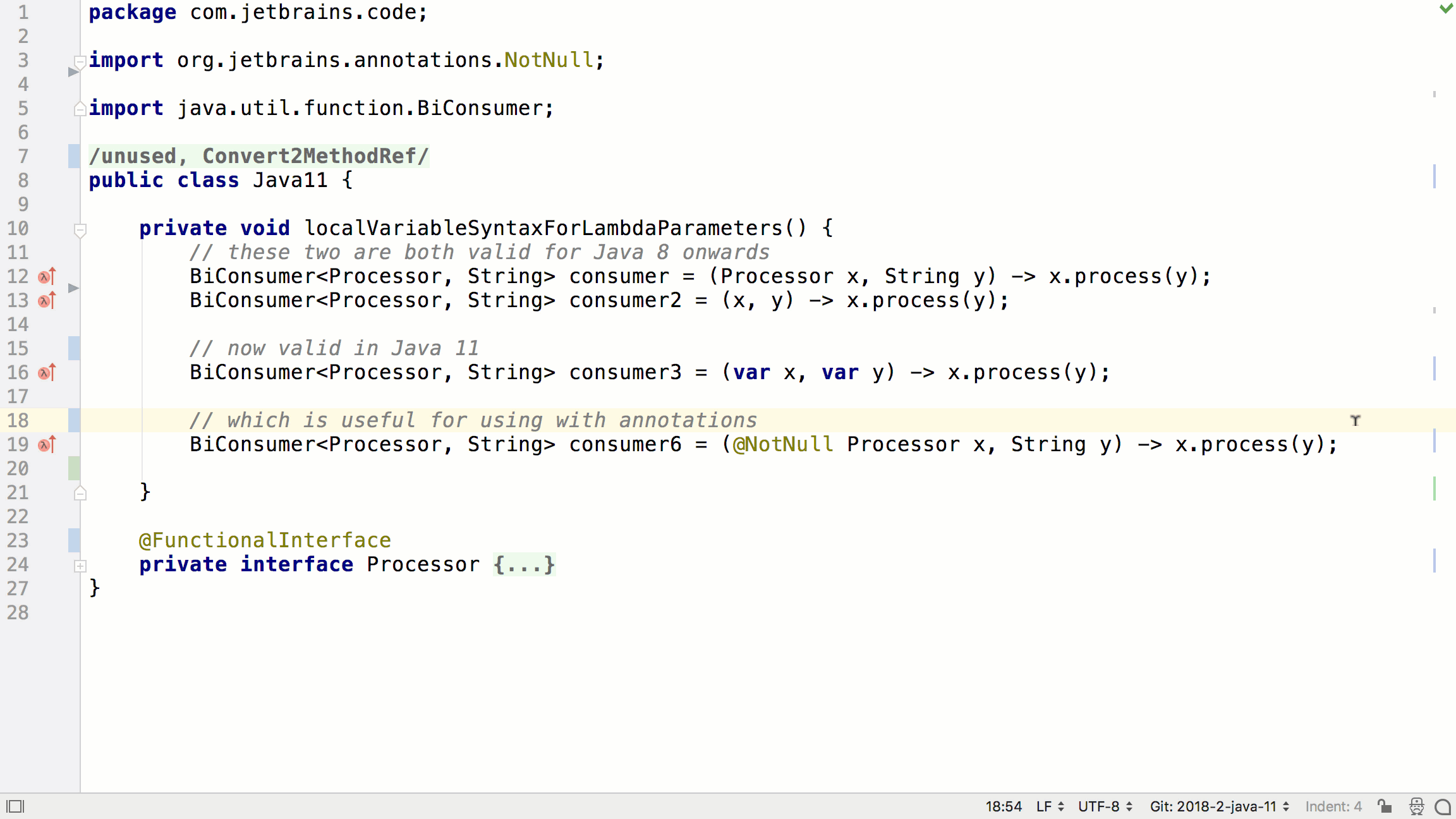


Рисунок 2.3 – Пример кода на языке Java

Для веб-приложения были выбраны такие языки как JavaScript (клиентская часть), Python (серверная часть). Также для клиентской части в качестве языков разметки были выбраны HTML и CSS.

JavaScript – это легковесный, интерпретируемый или JIT-компилируемый, объектно-ориентированный язык с функциями первого класса. Наиболее широкое применение находит как язык сценариев веб-страниц, но также используется и в других программных продуктах. JavaScript это прототипно-ориентированный, мультипарадигменный язык с динамической типизацией, который поддерживает объектно-ориентированный, императивный и декларативный (например, функциональное программирование) стили программирования.

Стандартом языка JavaScript является ECMAScript. Все современные браузеры полностью поддерживают ECMAScript 5.1, однако старые версии браузеров поддерживают, по крайней мере, ECMAScript 3.

Разработчиками написано большое количество инструментов поверх основного языка JavaScript, которые разблокируют огромное количество дополнительных функций с очень небольшим усилием. К ним относятся:

* программные интерфейсы приложения, встроенные в браузеры, обеспечивающие различные функциональные возможности, такие как динамическое создание HTML и установку CSS стилей, захват и манипуляция видеопотоком, работа с веб-камерой пользователя или генерация графики;
* сторонние API позволяют разработчикам внедрять функциональность в свои сайты от других разработчиков;
* сторонние фреймворки и библиотеки, повышающие скорость создания сайтов и приложений.

Пример код на языке JavaScript можно увидеть на рисунке 2.4.



Рисунок 2.4 – Пример кода на языке JavaScript

Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций. Python – активно развивающийся язык программирования, новые версии с добавлением/изменением языковых свойств выходят примерно раз в два с половиной года. Язык не подвергался официальной стандартизации, роль стандарта выполняет CPython, разрабатываемый под контролем автора языка.

Наиболее часто Python по применимости для разработки веб-серверов сравнивают с Perl и Ruby. Эти языки также являются интерпретируемыми и обладают примерно одинаковой скоростью выполнения программ. Как и Perl, Python может успешно применяться для написания скриптов (сценариев) [5]. Как и Ruby, Python является хорошо продуманной системой для ООП.

Однако по совокупности параметром, он превосходит вышеперечисленные языки, в первую очередь потому что это стабильный и распространённый язык. Он используется во многих проектах и в различных качествах: как основной язык программирования или для создания расширений и интеграции приложений. На Python реализовано большое количество проектов, также он активно используется для создания прототипов будущих программ. Python используется во многих крупных компаниях.

HTML – стандартизированный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML. Является единственным языком разметки пользовательских веб-страниц, поддерживаемым (в разных спецификациях) всеми существующими веб-браузерами. Язык HTML интерпретируется браузерами; полученная веб-страница отображается на экране монитора компьютера или мобильного устройства.

Изначально язык HTML был задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения (отображения). В идеале, текст с разметкой HTML должен был без стилистических и структурных искажений воспроизводиться на оборудовании с различной технической оснащённостью (цветной экран современного компьютера, монохромный экран органайзера, ограниченный по размерам экран мобильного телефона или устройства и программы голосового воспроизведения текстов). Однако современное применение HTML очень далеко от его изначальной задачи и теперь он не имеет ни одного аналога, который поддерживался бы весомой частью веб-браузеров, прост в использовании и изучении.

CSS – формальный язык описания внешнего вида документа, написанного с использованием языка разметки (HTML), применяется к элементам документа, чтобы изменить их внешний вид.

CSS представляет собой мощную систему, расширяющую возможности дизайна и верстки веб-страниц. Прямых аналогов CSS, поддерживаемых всеми основными браузерами не существует, некоторые веб-страницы могут отказаться от его использования и обходиться базовыми средствами HTML, но такие веб-страницы обычно очень просты по содержанию и неудобны в использовании, пример представлен на рисунке 2.5.

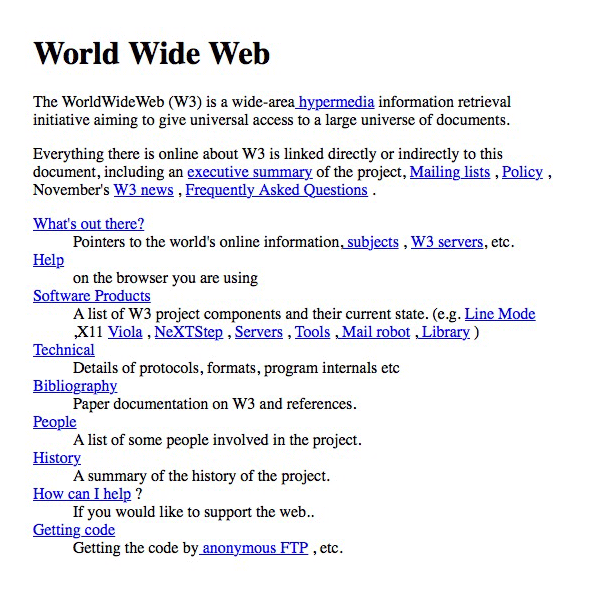


Рисунок 2.5 – Пример сайта без использования CSS

В связи с размером разрабатываемого приложения для платформы Android и его сложностью, была выбрана полноценная интегрированная среда разработки (а не обычный текстовый редактор) от компании Google – Android Studio. Существует также среда разработки Eclipse, которая на данный момент признана устаревшей и официально не дорабатывается для работы с приложениями для ОС Андроид. В ней используется устаревшая система сборки, она обладает неинтуитивным графическим интерфейсом и менее богатым функционалом по сравнению с Android Studio, отсутствие продвинутых вспомогательных инструментов, входящих в состав этой IDE.

Android Studio – это интегрированная среда разработки (IDE) для работы с платформой Android. Основанная на программном обеспечении IntelliJ IDEA от компании JetBrains, – официальное бесплатное средство разработки Android приложений. Данная среда разработки доступна для Windows, OS X и Linux, то есть является мультиплатформенной.

Среда Android Studio предназначена как для небольших команд разработчиков мобильных приложений (даже в количестве одного человека), так и для крупных международных организаций.

Решения для Android разрабатываются в Android Studio с использованием Java или C++. В основе рабочего процесса Android Studio заложен концепт непрерывной интеграции, позволяющий сразу же обнаруживать имеющиеся проблемы, что является неоспоримым преимуществом при выборе инструмента разработки мобильного приложения для операционной системы Андроид.

Продолжительная проверка кода обеспечивает возможность эффективной обратной связи с разработчиками.

С помощью средств оценки производительности определяется состояние файла с пакетом прикладных программ. Визуализация графики дает возможность протестировать, соответствует ли приложение стандартам по скорости и отзывчивости отображения элементов пользовательского интерфейса.

С помощью инструмента для визуализации памяти присутствует возможность в реальном времени отслеживать, когда приложение будет использовать слишком много оперативной памяти и когда произойдет процесс очистки используемых данных («сборка мусора»). Инструменты для анализа батареи позволяют просмотреть, какая нагрузка приходится на используемое в ходе тестирования устройство.

Android Studio позволяет создавать приложения, используя такие языки программирования как Java, Kotlin, С/C++ [6]. Пример работы с Android Studio отображен на рисунке 2.6.

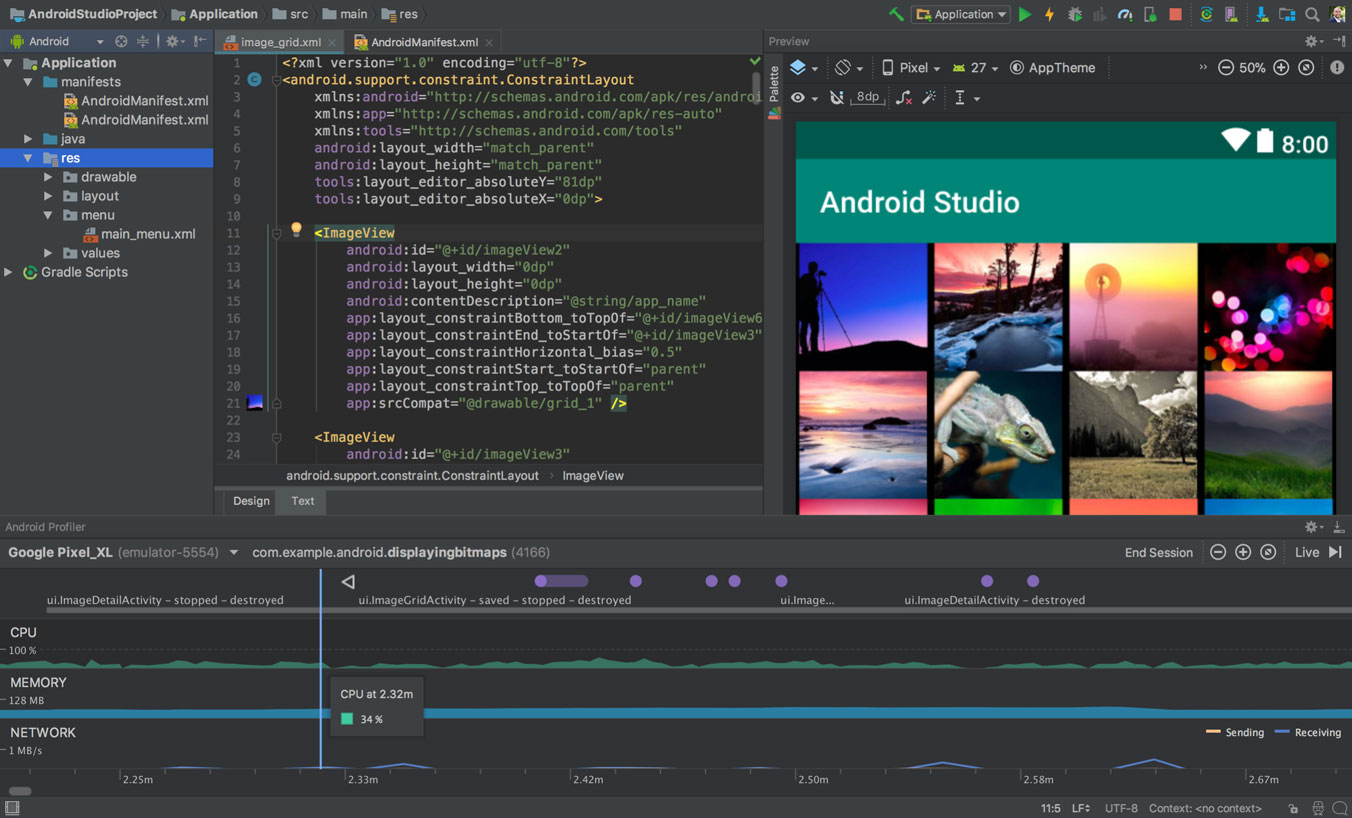


Рисунок 2.6 – Пример работы с Android Studio

Eclipse – свободная интегрированная среда разработки модульных кроссплатформенных приложений.

Eclipse служит в первую очередь платформой для разработки расширений, чем он и завоевал популярность: любой разработчик может расширить Eclipse своими модулями.

Профессиональные программисты умеют быть продуктивными, используя для изменения кода те рефакторинги, которые предлагает их среда разработки. Большинство современных сред разработки предлагают большой выбор возможностей рефакторинга, однако и в этом Eclipse полностью уступает Android Studio, в которой существует большое количество быстрых команд для редактирования кода в процессе его написания.

Главный экран Eclipse представлен рисунке 2.5.

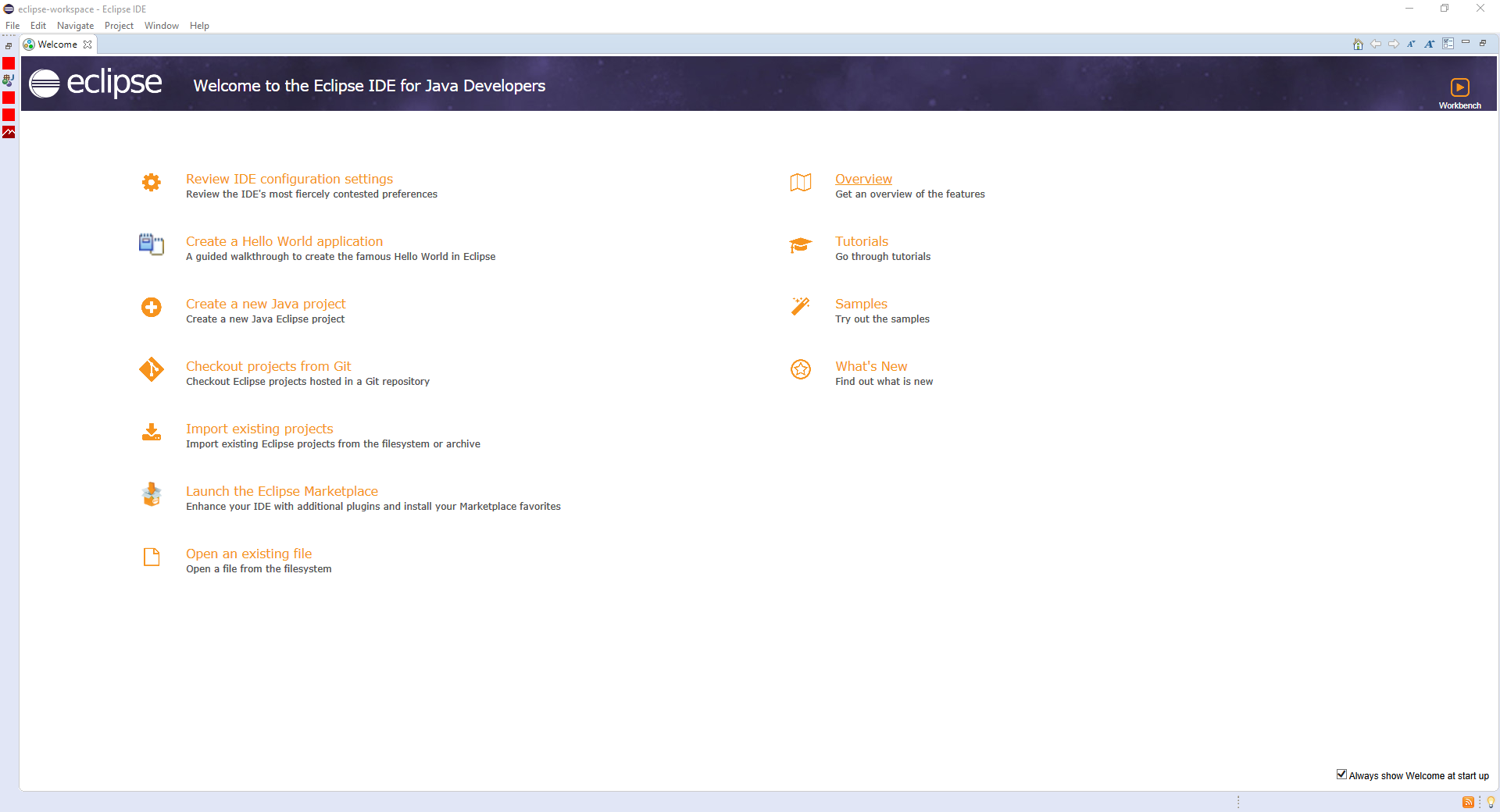


Рисунок 2.7 – Главный экран Eclipse

Также преимуществом Eclipse перед Android Studio является небольшое количество требуемых для корректной работы системных ресурсов, однако, исходя из всех вышеперечисленных недостатков, ни это, ни открытость программного обеспечения не является приоритетным и в качестве IDE была выбрана Android Studio.

**2.3 Разработка графической документации**

В ходе дипломного проекта в качестве графической документации необходимо составить диаграмму деятельности и диаграмму вариантов использования.

Диаграмма деятельности – это блок-схема, которая показывает, как поток управления переходит от одной деятельности к другой, при этом внимание фиксируется на результате деятельности. Результат может привести к изменению состояния системы или возвращению некоторого значения.

Диаграмма деятельности имеет следующие отличия от традиционной блок-схемы:

– обладание более высоким уровнем абстракции;

– наличие возможности представления с помощью диаграмм деятельности управления параллельными потоками наряду с последовательным управлением.

Одно из основных направлений использования диаграмм деятельности – отображение внутрисистемной точки зрения. Диаграммы деятельности применяют для описания шагов, которые должна предпринять система её появления.

Разработка диаграммы деятельности обычно преследует следующие цели:

– детализирование особенности алгоритмической и логической реализации;

– выделение последовательных и параллельных потоков управления;

– подготовка детальной документации для взаимодействия разработчиков системы с ее заказчиками и проектировщиками.

Диаграмма деятельности представлена на рисунке 2.8.

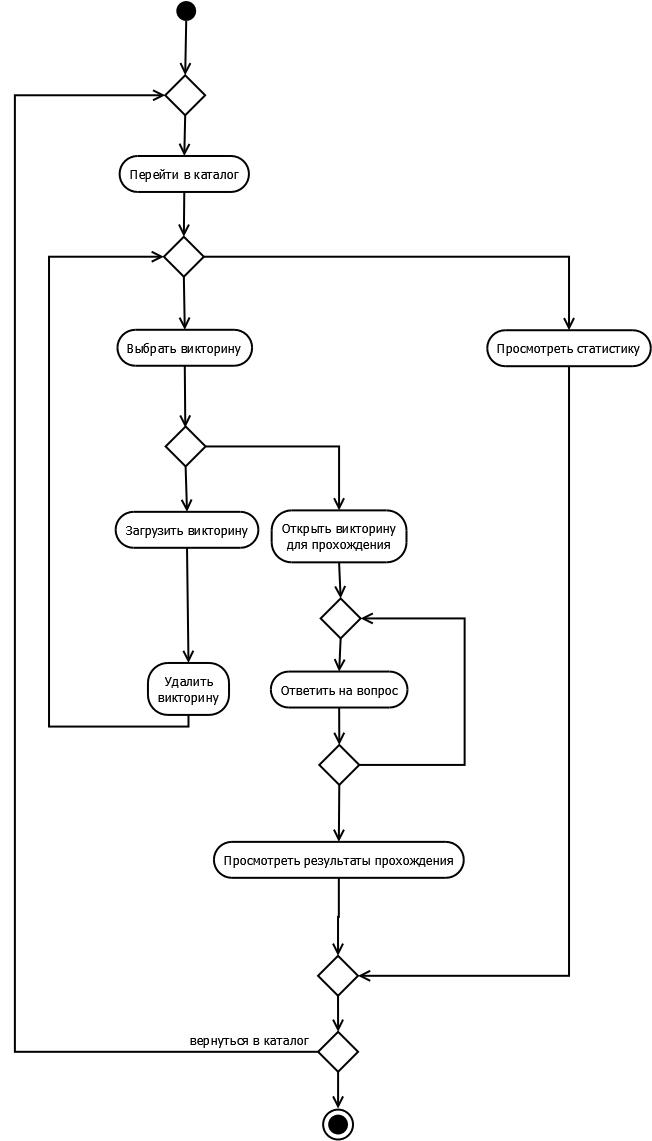


Рисунок 2.8 – Диаграмма деятельности

Процесс работы пользователя с приложением начинается с перехода в каталог викторин, далее пользователю предстоит выбор между проверкой статистики и выбором викторин. При выборе викторин у пользователя есть возможность загрузить викторины для использования офлайн или пройти викторину. После прохождения викторины пользователю будут предоставлены его результаты.

Диаграмма вариантов использования показывает некоторые варианты использования в системе некоторых действующих лиц, и отношения между ними. Она представляет высокоуровневое описание системы (архитектуру системы).

Для достижения этих целей вначале строится модель в форме так называемой диаграммы вариантов использования, которая описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования. Разработка диаграммы вариантов использования преследует следующие цели:

– определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;

– сформулировать общие требования к функциональному поведению проектируемой системы;

– подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Диаграмма вариантов использования (рисунок 2.9) в дипломном проекте содержит два действующих лица: пользователь и составитель.

* режиме пользователя включен вариант просмотра каталога викторин, просмотр самих викторин и их статистику, загружать викторины, также есть возможность прохождения викторин и просмотр собственных результатов.

Работа в режиме составителя включает возможность добавления викторин.

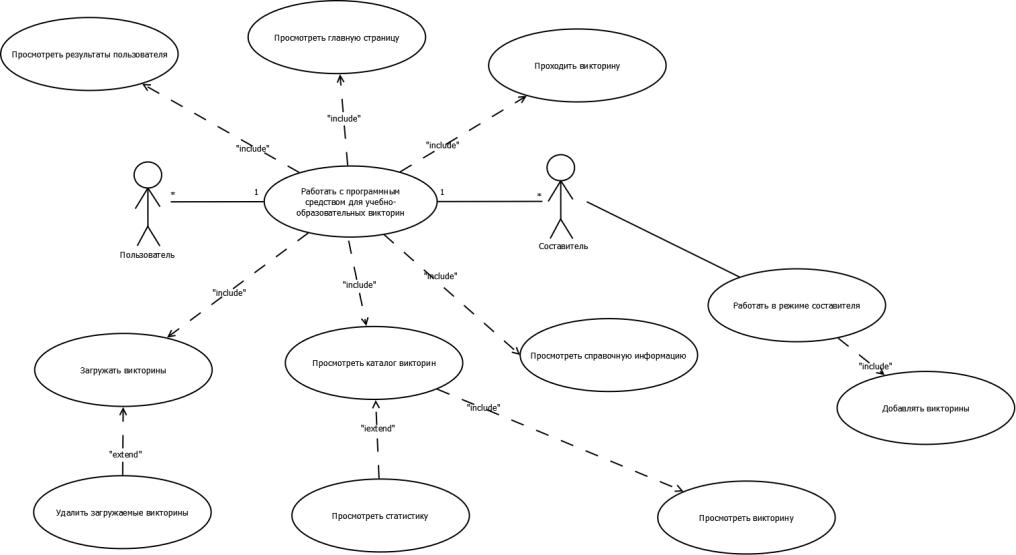


Рисунок 2.9 – Диаграмма вариантов использования

3 Программная реализация

**3.1 Физическая структура**

База данных – это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации, преимущественно больших объемов. Базы данных активно используются для динамических сайтов со значительными объемами данных, такими как интернет-магазины, порталы, корпоративные сайты. Динамические сайты разработаны с помощью серверного языка программирования или на основе CMS и не имеют готовых страниц с данными по аналогии с HTML-сайтами. Страницы динамических сайтов формируются в результате взаимодействия скриптов и баз данных после соответствующего запроса клиента к веб-серверу.

Логическая структура базы данных – структура для пользователя, физическая – структура базы данных для ЭВМ. Физическая структура определяет, тип и свойства данных, которые будут записаны в память компьютера.

Физическая структура данных независима от прикладных программ. К примеру, реляционная модель освобождает пользователей от взаимодействия с физической структурой данных.

В отличие от логических моделей, физическая модель данных связана со способами организации данных на носителях методами доступа к данным (также называемых системами хранения).

Атрибут – свойство сущности.

Домен атрибута – множество допустимых значений, которые может принимать атрибут.

Кортеж – конечное множество взаимосвязанных допустимых значений атрибутов, которые вместе описывают некоторую сущность.

Отношение — конечное множество кортежей (таблица).

Правила перехода к физической модели следующие: каждое отношение превращается в файл базы данных, каждый столбец – в поле файла, каждый кортеж – в запись файла. Этап физического моделирования базы данных включает в себя определение состава файлов и их заполнение исходными данными в соответствии с ограничениями, допущениями и особенностями предметной области.

Пример разделения на физическую и логическую структуру можно увидеть на рисунке 3.1.

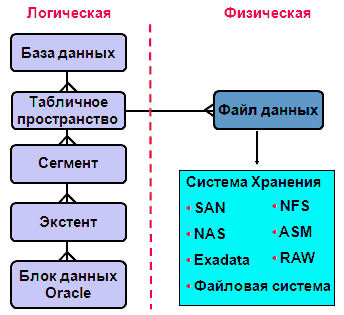


Рисунок 3.1 – Схема разделения на физическую и логическую структуру

В процессе физического проектирования БД необходимо присвоить имена таблицам, а также их полям.

Основными средствами физического моделирования в БД являются:

* структура хранения данных;
* поисковые структуры (для сокращения времени поиска данных);
* язык описания данных.

Структуры хранения в базе данных – структуры СУБД, обеспечивающие хранение данных, и, как правило, независимые от логической структуры данных. Структура хранения может быть изменена без затрагивания кода приложения и не влияет на семантику запросов. В редких случаях знание структуры хранения позволяет дополнительно оптимизировать запросы. Под структурой хранения понимается привязка структуры данных к её реализации, которая может быть другой структурой данных.

Способность программных средств управления данными отделить физическую структуру данных от представления пользователей или от логической организации данных дает пользователям возможность представлять логическую структуру данных независимо от их физической реализации.

При построении физической структуры данных в нее закладывается избыточность с целью обеспечения надежности управления данными.

Нормальная форма – это требование, которое предъявляется к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами.

Основной целью нормализации является исключение избыточного дублирования данных, возникших при добавлении, редактировании и удалении кортежей.

Таблица находится в первой нормальной форме, если в ней отсутствуют повторяющиеся записи. Вторая нормальная форма включает в себя условия первой нормальной формы, а также любое из не ключевых полей должно однозначно идентифицироваться набором ключевых.

Таблица находится в третьей нормальной форме, если она удовлетворяет требованиям второй нормальной формы, а также ни одно из не ключевых полей таблицы не идентифицируется с помощью другого не ключевого поля.

В таблице 3.1 представлена информация о структуре таблицы «Quizzes» базы данных веб-приложения, хранящей информацию о представленных викторинах.

Таблица 3.1 – Quizzes

|  |  |  |
| --- | --- | --- |
| Наименование атрибута | Описание | Тип данных |
| id | Первичный ключ | bigint |
| name | Имя викторины | varchar |
| downloads | Количество загрузок викторины | bigint |
| section\_id | Внешний ключ, идентификационный номер раздела | bigint |

В таблице 3.2 представлена структура таблицы «Questions», отвечающей за хранение данных о всех вопросах викторин.

Таблица 3.2 – Questions

|  |  |  |
| --- | --- | --- |
| Наименование атрибута | Описание | Тип данных |
| id | Первичный ключ | bigint |
| text | Текст вопроса | varchar |
| quiz\_id | Внешний ключ, идентификационный номер викторины | bigint |

В таблице 3.3 описана таблица «Answers», которая хранит все возможные варианты ответов на вопросы для викторин.

Таблица 3.3 – Answers

|  |  |  |
| --- | --- | --- |
| Наименование атрибута | Описание | Тип данных |
| id | Первичный ключ | bigint |
| text | Текст ответа | varchar |
| question\_id | Уникальный идентификатор вопроса | bigint |

В таблице 3.4 представлена структура таблицы «Topics», которая предназначена для хранения различных тем для викторин.

Таблица 3.4 – Topics

|  |  |  |
| --- | --- | --- |
| Наименование атрибута | Описание | Тип данных |
| id | Первичный ключ | bigint |
| name | Название раздела | varchar |

Таблица «Logs», структура которой отражена в таблице 3.5, необходима для хранения данных о совершенных скачиваниях согласно правилам третьей формы нормализации.

Таблица 3.5 – Logs

|  |  |  |
| --- | --- | --- |
| Наименование атрибута | Описание | Тип данных |
| id | Первичный ключ | bigint |
| quiz\_id | Уникальный идентификатор викторины | bigint |
| device\_type | Уникальный идентификатор устройства | bigint |
| created\_on | Дата и время создания записи | datetime |

В таблице 3.6 представлена структура таблицы «DeviceTypes», представляющей хранения различных тем для викторин.

Таблица 3.6 – DeviceTypes

|  |  |  |
| --- | --- | --- |
| Наименование атрибута | Описание | Тип данных |
| id | Первичный ключ | bigint |
| name | Название типа устройства | varchar |

На рисунке 3.2 представлена структура папки, содержащей в себе файлы разметки HTML. Они предназначены для указания базовой разметки страницы при помощи тегов.

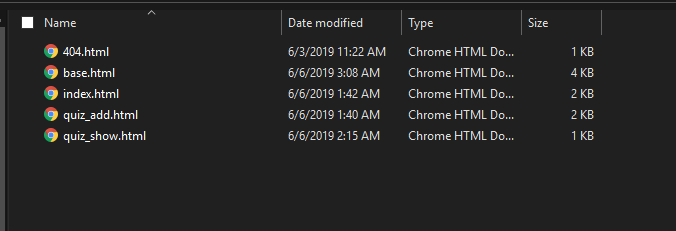


Рисунок 3.2 – Структура папки с файлами разметки

На рисунке 3.3 представлена структура папок, содержащих в себе файлы кода на языке JavaScript, предназначенные для добавления интерактивности на веб-страницы, создания динамически изменяемого содержимого.

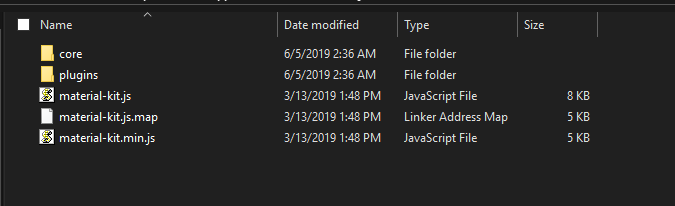


Рисунок 3.3 – Структура папок с файлами кода на языке JavaScript

Структура папок, содержащих файлы с описанием каскадных таблиц стилей, представлена на рисунке 3.4. Каскадные таблицы стилей (CSS) предназначены для задания цветов, шрифтов, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью CSS является разделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое с момента создания производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом.

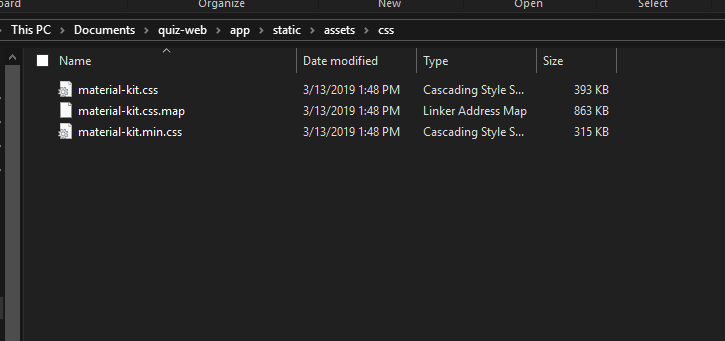


Рисунок 3.4 – Структура папки с CSS файлами

Все перечисленные таблицы соответствуют третьей нормальной форме, а файловая структура основных файлов проекта является оптимальной.

**3.2 Описание разработанных модулей**

Для разрабатываемого приложения был выбран специальный паттерн проектирования под названием МВП (модель, вид, представление).

Шаблон или паттерн проектирования – повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого повторяющегося контекста.

МВП – шаблон проектирования пользовательского интерфейса, который облегчает модульное тестирование и позволяет более чётко разграничивать ответственность логики приложения и кода по отображению и работе с пользовательским интерфейсом.

Каждый из видов классов отвечает за определенную работу в программе:

* модель – хранит в себе бизнес-логику по работе с данными;
* вид – реализует отображение данных из модели, обращается к представителю за обновлениями;
* представитель – реализует часть логики по взаимодействию модели и вида, берёт на себя функциональность посредника.

В связи с этим, были созданы следующие компоненты:

* MainActivity – активность главной страницы, содержит кнопку для перемещения в библиотеку и для перехода к веб-сайту (рисунок 3.5);

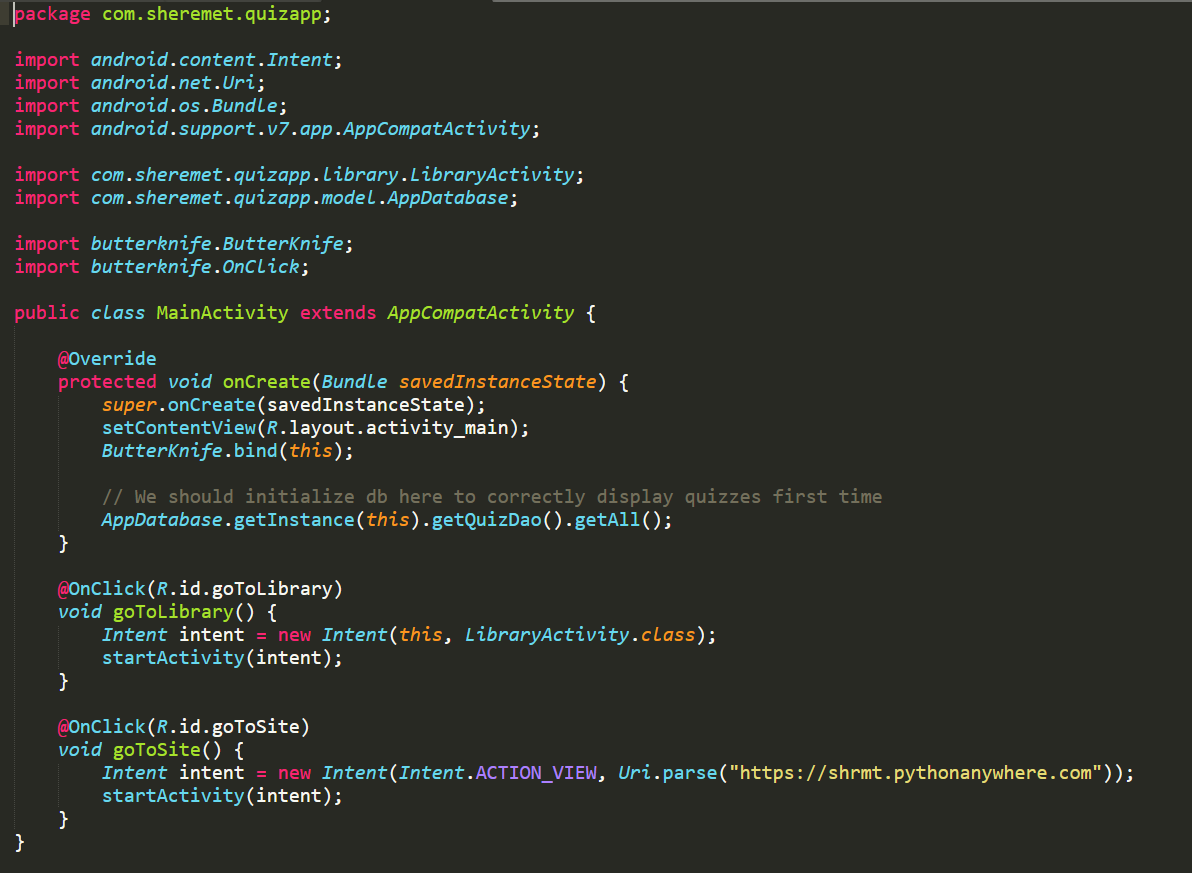


Рисунок 3.5 – Пример реализации MainActivity

* LibraryActivity – активность страницы, отвечающей за отображение списка викторин, предоставляет возможности по открытию или удалению викторины, просмотру информации о викторине (рисунок 3.6);

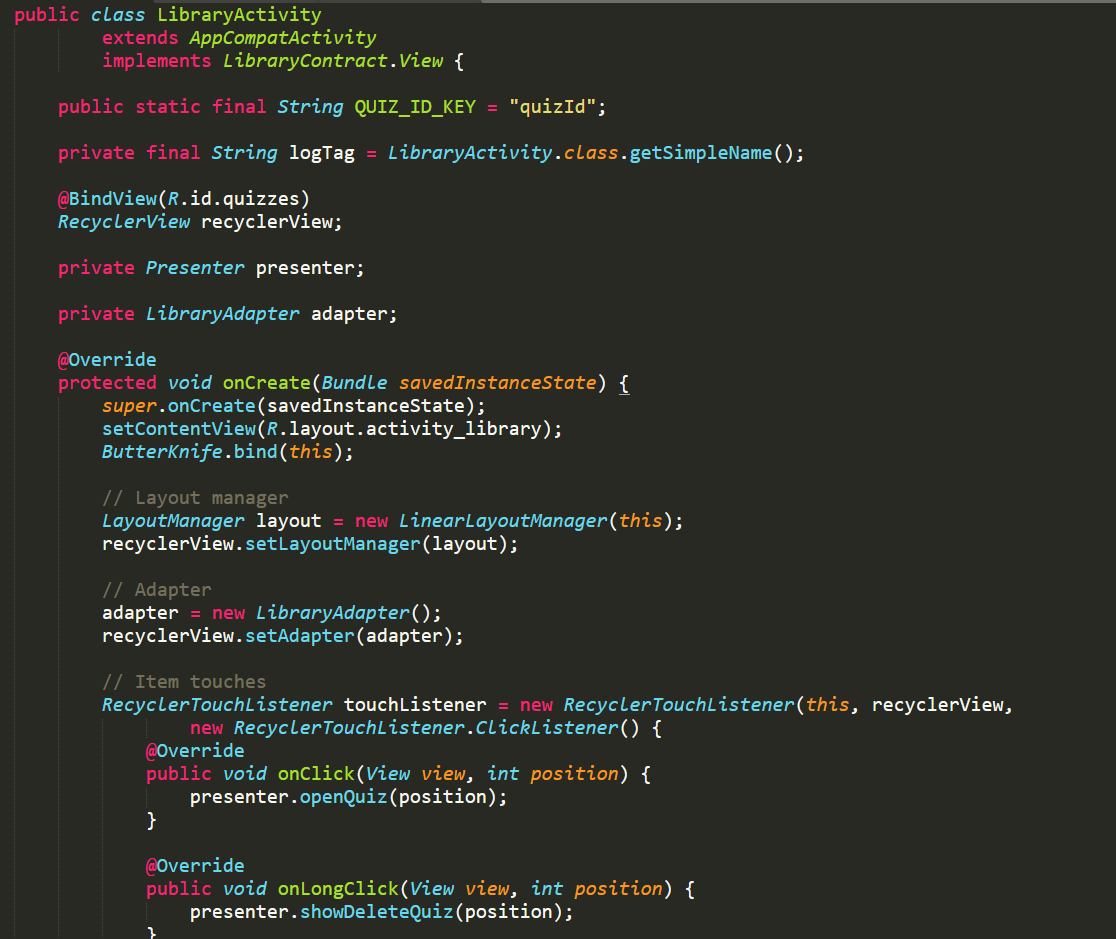


Рисунок 3.6 – Пример реализации LibraryActivity

* GameActivity – активность основной страницы приложения, отвечает за исполнение выбранной в библиотеке викторины, корректный переход между вопросами после ответа на один из них и так далее (рисунок 3.7);

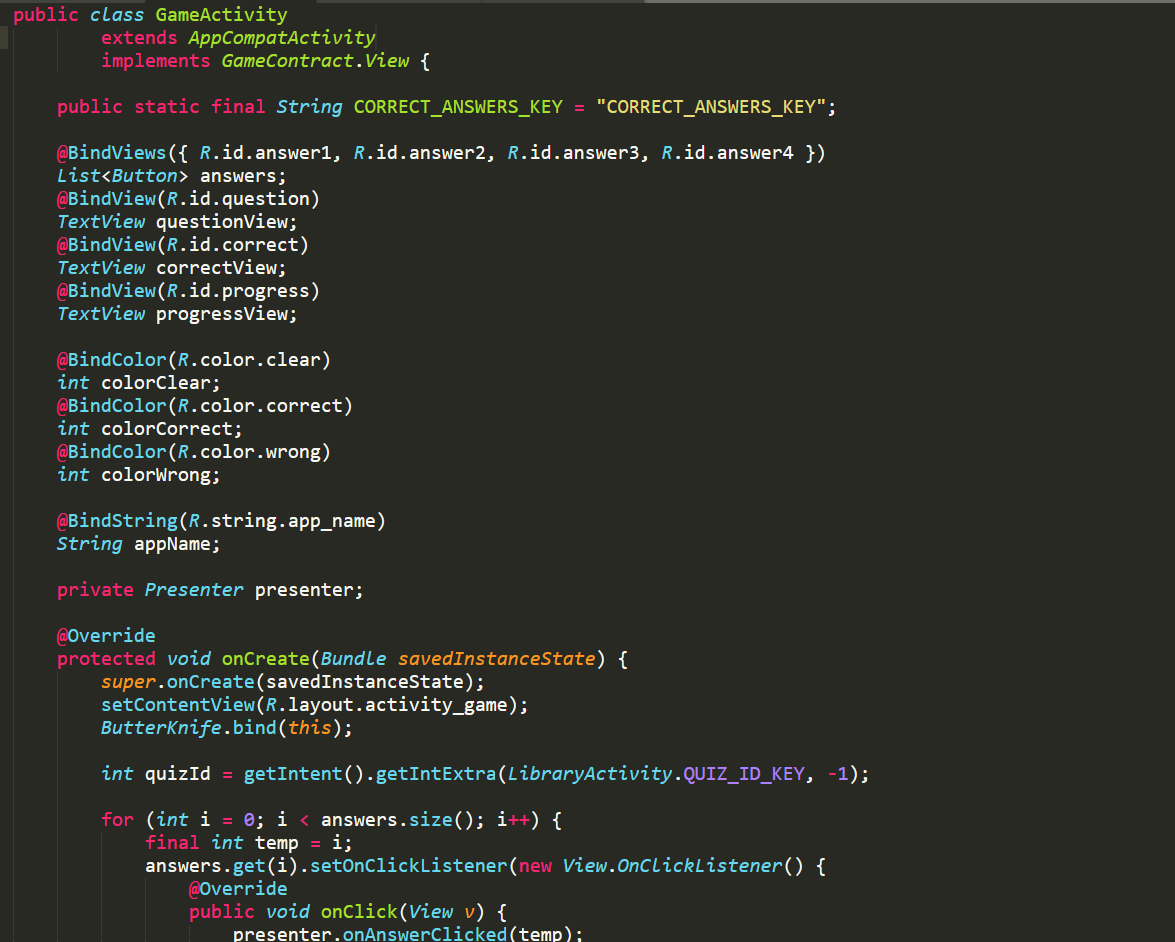


Рисунок 3.7 – Пример реализации GameActivity

* ResultActivity – страница для отображения индивидуального результата после ответа на все вопросы викторины.

Классы Activity в данном приложении, разработанном с применением паттерна МВП являются видом и реализуют необходимые функции по отображению необходимого (содержимое зависит от игрового экрана).

Для каждого из перечисленных Activity был создан класс-представитель, выполняющий связывающую роль в взаимодействии вида и модели. Также используется одна общая модель, позволяющая обращаться к викторинам и загружать необходимую информацию.

**4 Тестирование**

**4.1 Цели и виды тестирования**

Тестирование программного обеспечения – процесс испытания программного продукта, с целью проверки соответствия между реальным поведением программы и её ожидаемым поведением.

Основные виды тестирования:

* функциональное тестирование – это тестирование ПО в целях проверки реализуемости функциональных требований;
* тестирование производительности – (включает в себя нагрузочное тестирование, стресс-тестирование, тестирование стабильности) тестирование, которое проводится с целью определения, как быстро работает вычислительная система или её часть под определённой нагрузкой;
* конфигурационное тестирование;
* юзабилити-тестирование – исследование, выполняемое с целью определения, удобен ли некоторый искусственный объект;
* тестирование безопасности – оценка уязвимости программного обеспечения к различным атакам;
* тестирование локализации – процесс проверки качества адаптации программного обеспечения к культуре какой-либо страны;
* тестирование совместимости – вид нефункционального тестирования, основной целью которого является проверка корректной работы продукта в определенном окружении.

Необходимость этапа тестирования как при разработке веб-приложения, так и при разработке приложения под операционную систему Андроид очевидна – некоторые ошибки могут привести к полному краху системы в дальнейшем, что может повлечь за собой материальные потери.

Тестирование мобильных приложений существенно отличается от тестирования приложений, предназначенных для использования на персональных компьютерах [7].

Мобильные приложения отличаются от настольных, соответственно, в процессе тестирования инженер обязан провести проверки, обусловленные природой мобильных приложений, например, необходимо проводить тестирование установки обновлений.

Операционные системы обновляются довольно часто, что приводит и к обновлению приложений. Пользователь не должен испытывать каких-либо сложностей в процессе обновления. Тестирование должно определить, как реагирует приложение на непредсказуемые действия пользователей.

Еще один вид проверок – проверки различного вида соединения. Такое тестирование проходит в лабораторных условиях, где возможно воссоздать максимально реалистичные условия связи. Такой вид проверки необходим, чтобы понять, как приложение будет вести себя в ситуациях, например, когда сигнал Wi-Fi едва уловим. Телефон быстро запускается, часто уходит в спящий режим, переключается между вышками связи и так далее. Одним из важнейших факторов успеха приложения является тестирование. Что бы ни делали разработчики, как бы ни трудились маркетологи и менеджеры, без контроля качества невозможно создание корректно работающего и конкурентоспособного приложения.

Список необходимых для тестирования пунктов:

* установка и запуск приложения, выход из приложения, повторный вход, удаление приложения с мобильного устройства;
* выключение устройства, разрядка устройства;
* зарядка устройства;
* отключение интернета;
* стабильность;
* адаптация приложения к портретной и альбомной ориентациям устройства;
* входящие и исходящие SMS, MMS, звонки, оповещения других приложений;
* переход устройства в режим ожидания;
* отличия в работе Android и iOS;
* выбор устройств для тестирования;
* особенности, связанные с установкой, удалением и обновлением;
* работа со встроенными кнопками;
* сети и особенности взаимодействия приложения с сетями;
* прерывания и переключения;
* тестирование графического и логического интерфейсов проекта;
* проверка работы с памятью мобильного устройства или телефона;
* локализация;
* потребление питания.

Основные цели проведения мероприятий по тестирования программного средства:

* повысить вероятность того, что приложение, предназначенное для тестирования, будет работать правильно при любых обстоятельствах;
* проверить, что наиболее критические последовательности действий с системой конечного пользователя выполняются верно;
* проверить, что изменения в базах данных не оказывают неблагоприятного влияния на существующие программные модули;
* использовать инструменты автоматизированного тестирования там, где это целесообразно.

Для клиент-серверных приложений конфигурационное тестирование можно разделить на два этапа: серверный и клиентский. На первом «серверном» этапе, тестируется взаимодействие выпускаемого интерактивного веб-приложения с окружением, в котором оно будет установлено.

Основное внимание здесь делается на тестирование с целью определения оптимальной конфигурации оборудования, удовлетворяющего требуемым характеристикам качества.

На следующем, «клиентском» этапе, веб-приложение тестируется с позиции его конечного пользователя и конфигурации его рабочей станции. На этом этапе будут протестированы следующие характеристики: удобство использования, функциональность. Чтобы проверить это, необходимо провести ряд тестов с различными конфигурациями рабочих станций.

Компонентное (модульное) тестирование проверяет функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы по-отдельности.

При интеграционном тестировании проверяется взаимодействие между компонентами системы после проведения компонентного тестирования.

Основной задачей системного тестирования является проверка как функциональных, так и не функциональных требований в системе в целом. При этом выявляются дефекты, такие как неверное использование ресурсов системы, непредусмотренные комбинации данных пользовательского уровня, несовместимость с окружением, непредусмотренные сценарии использования, отсутствующая или неверная функциональность, неудобство использования.

Даже если система удовлетворяет всем требованиям, важно убедиться в том, что она удовлетворяет нуждам пользователя и выполняет свою роль в среде своей эксплуатации, как это было определено в бизнес модели системы. Следует учесть, что и бизнес модель может содержать ошибки. Поэтому так важно провести операционное тестирование как финальный шаг валидации. Кроме этого, тестирование в среде эксплуатации позволяет выявить и нефункциональные проблемы, такие как: конфликт с другими системами, смежными в программных и электронных окружениях; недостаточная производительность системы в среде эксплуатации. Очевидно, что нахождение подобных вещей на стадии внедрения – критичная и дорогостоящая проблема. Поэтому так важно проведение не только верификации, но и валидации, с самых ранних этапов разработки программного обеспечения.

Приемочное тестирование – это формальный процесс тестирования, который проверяет соответствие системы требованиям и проводится с целью определения степени удовлетворения системы критериям.

**4.2 Обнаружение и устранение ошибок**

Сборка сайта при тестировании осуществлялась пошаговым методом сверху вниз. Такой метод сборки предполагает последовательное подключение к набору уже протестированных страниц очередной страницы.

Для клиент-серверных приложений конфигурационное тестирование можно разделить на два этапа: серверный и клиентский.

На первом «серверном» этапе, было протестировано взаимодействие выпускаемого интерактивного веб-приложения с окружением, в котором оно будет установлено.

Во время второго этапа был протестирован пользовательский интерфейс при различных форматах экрана и различных используемых веб-браузерах.

При проведении тестирования была проверена работа всех ссылок, размещенных на сайте. Все переходы осуществляются корректно в соответствии с междустраничными связями. Ссылки, которые имеют не существующий адрес, отсутствуют. В ходе тестирования было выявлено, что программное средство является удобным для пользователя, позволяет беспрепятственно переходить с одной страницы на другую без возвращения на предыдущие страницы, обладает интуитивно понятным пользовательским интерфейсом. В таблице 4.1 отображен чек-лист Android-приложения.

Таблица 4.1 – Чек-лист приложения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Описание | Шаги для воспроизведения | Актуальный результат | Ожидаемый результат |
| *1* | *2* | *3* | *4* | *5* |
| 1 | Переход к списку викторин | В главном меню приложения нажать на кнопку перехода к списку викторин | Появился список викторин из трёх элементов | Появился список викторин из нескольких элементов |
| 2 | Открытие викторины | После перехода к списку викторин, совершить одиночное нажатие по любой из представленных | Появился основной игровой экран, на нём уже отображается первый вопрос | Появился основной игровой экран, на нём первый вопрос |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *1* | *2* | *3* | *4* | *5* |
| 3 | Удаление викторины | После перехода к списку викторин, совершить длительное нажатие по любой из представленных. В диалогов окне подтвердить удаление | После небольшой задержки, викторина полностью исчезла из списка | После небольшой задержки, викторина полностью исчезла из списка |
| 4 | Запуск викторины без вопросов | После перехода к списку викторин, совершить нажатие по тестовой викторине без вопросов | Переход сразу к окну результатов теста без ошибок | Переход сразу к окну результатов теста без ошибок |
| 5 | Поведение кнопки перехода назад на экране результатов | Пройти любую викторину, после появления экрана результатов, нажать на кнопку перехода назад | Вместо перехода к предыдущей странице игры, осуществлен переход к библиотеке викторин | Вместо перехода к предыдущей странице игры, осуществлен переход к библиотеке викторин |

Все тест-кейсы из чек-листа были выполнены как на встроенном в Android Studio эмуляторе с версией Андроид API 21, так и на реальном устройстве с API 28. Ошибок или багов в работе программы за время тестирования найдено не было.

**5 Применение**

* 1. **Назначение и условия применения**

Программное средство для учебно-образовательных викторин включает в себя: приложение для мобильной операционной системы Андроид, веб-приложения для использования с помощью браузера.

Одним из главных требований для полноценного использования обеих частей программного средства является наличие доступа к интернету, так как веб-приложение должно предоставлять пользователям функционал для создания и редактирования викторин, а мобильное приложение – для скачивания и воспроизведения.

Минимальные системные требования для сервера:

* процессор двухъядерный с тактовой частотой 2 ГГц или лучше;
* оперативная память не менее 512 Мб;
* свободное место на жестком диске не менее 100 Мб;
* операционная система Microsoft Windows Server 2003 или более поздняя.

Минимальные требования для установки Андроид-приложения:

* оперативная память не менее 512 Мб;
* свободное место на накопителе не менее 10 Мб;
* операционная система Android 6.0 или более поздняя.

Минимальные системные требования для устройства для пользования веб-приложением:

* веб-браузер Internet Explorer 9 или выше;
* оперативная память не менее 512 Мб.

Возможна установка Андроид-приложения на смартфон путём подключения через USB-кабель с последующим переносом APK-файла. При этом необходимо, чтобы на устройстве был установлен Android Device Bridge. После окончания процесса копирования файла приложения в файловую систему смартфона, необходимо вручную запустить процесс установки.

Доступ к сайту может быть осуществлён путём перехода по адресу веб-приложения.

**5.2 Руководство пользователя**

Приложение «Игра-викторина» является разновидностью учебно-образовательных приложений. Основываясь на сравнении с аналогами, приложение имеет удобный и интуитивный интерфейс, благодаря которому пользователю будет легко адаптироваться к работе с приложением.

Ясность  –  это первая и самая главная задача любого интерфейса. Преставленный в приложении интерфейс полностью справляется с этой задачей. Чтобы интерфейс эффективно помогал пользователям выполнять задачи, он должен быть предсказуемым, узнаваемым, а также, понятным с момента первого пользования. Пример следования данной концепции при создании пользовательского интерфейса изображен на рисунке 5.1.

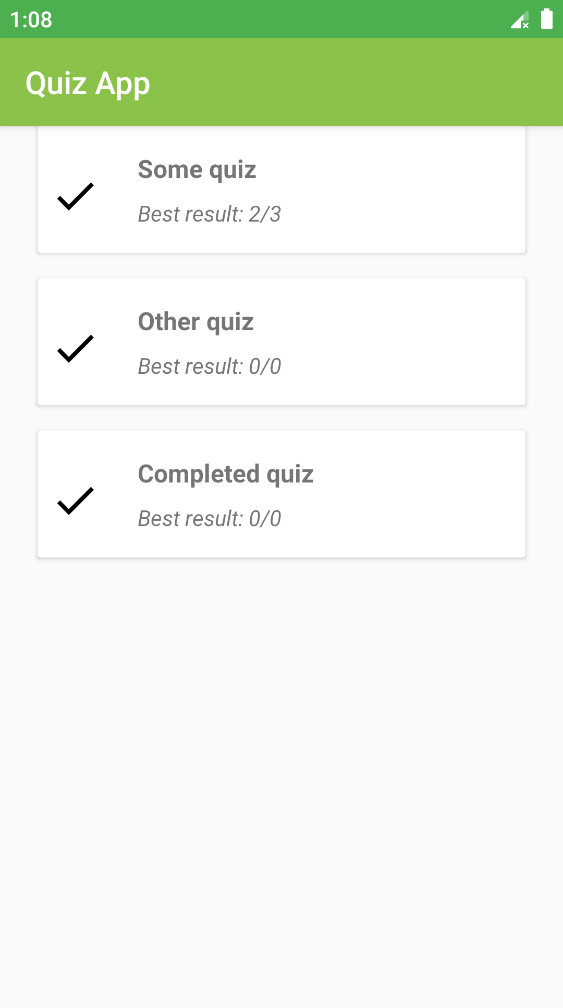


Рисунок 5.1 – Экран мобильного приложения со списком викторин

Основные функции приложения:

* показ списка викторин;
* выбор викторин из списка;
* удаление викторин из списка;
* корректное отображение вопросов и ответов;
* отображение прогресса прямо во время прохождения викторины;
* игровая логика по переходу между вопросами, реакцией на выбор варианта ответа;
* хранение результата о прохождении;
* индивидуальные отчёты о прохождении викторины;
* работа без доступа к интернету;
* стойкость к различным возможным системным ошибкам.

Главная страница предоставляет возможность перехода к списку викторин. После выбора любой из представленных викторин, запускается игровой экран. На нём сразу же будет отображен случайный вопрос из выбранной викторины. После нажатие на ответ, экран становится невосприимчивым к нажатию на некоторое время для предотвращения случайных нажатий.

Нажатая кнопка с ответом начинает подсвечиваться зеленым если выбранный ответ был верным, и красным – если пользователь приложения дал неверный ответ. Пример состояния игрового экрана изображен на рисунке 5.2.

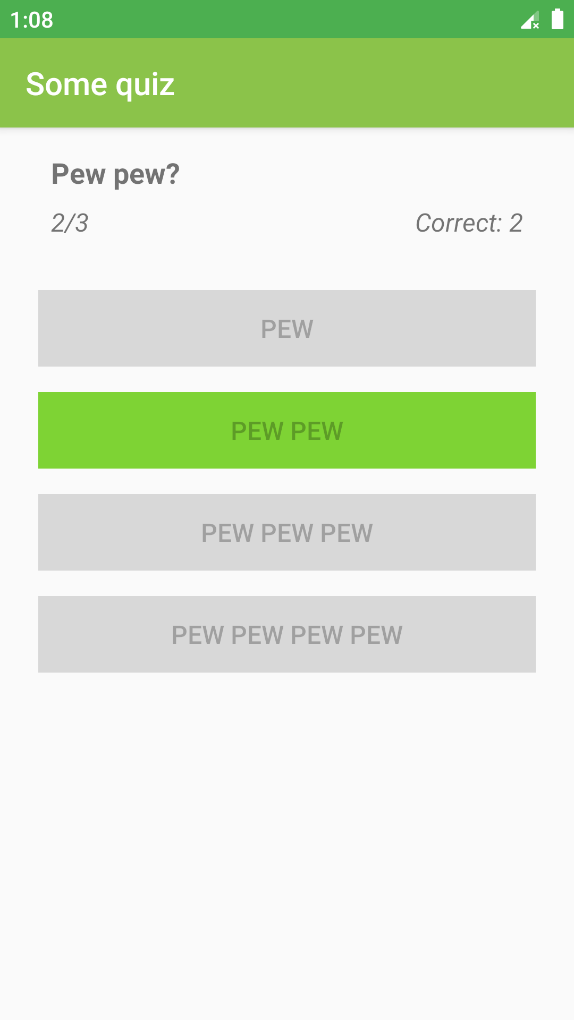


Рисунок 5.2 – Игровой экран разрабатываемого приложения

После выбором пользователем ответов на все предоставленные вопросы (вопросы показываются в случайном порядке), осуществляется переход на экран результатов прохождения. На этом экране осуществляется финальный подсчёт результатов и в зависимости от соотношения правильно отвеченных вопросов к полному их количеству в данном тесте, принимается решение о показе соответствующей фразы и иконки. Пример экрана результатов изображён на рисунке 5.3.



Рисунок 5.3 –Экран результатов при полном прохождении теста

Пользовательский интерфейс веб-приложения был разработан с помощью тех же основных правил. Главная страница сайта одновременно выполняет функцию отображения списка викторин, доступных для просмотра или скачивания через Андроид-приложение (рисунок 5.4).

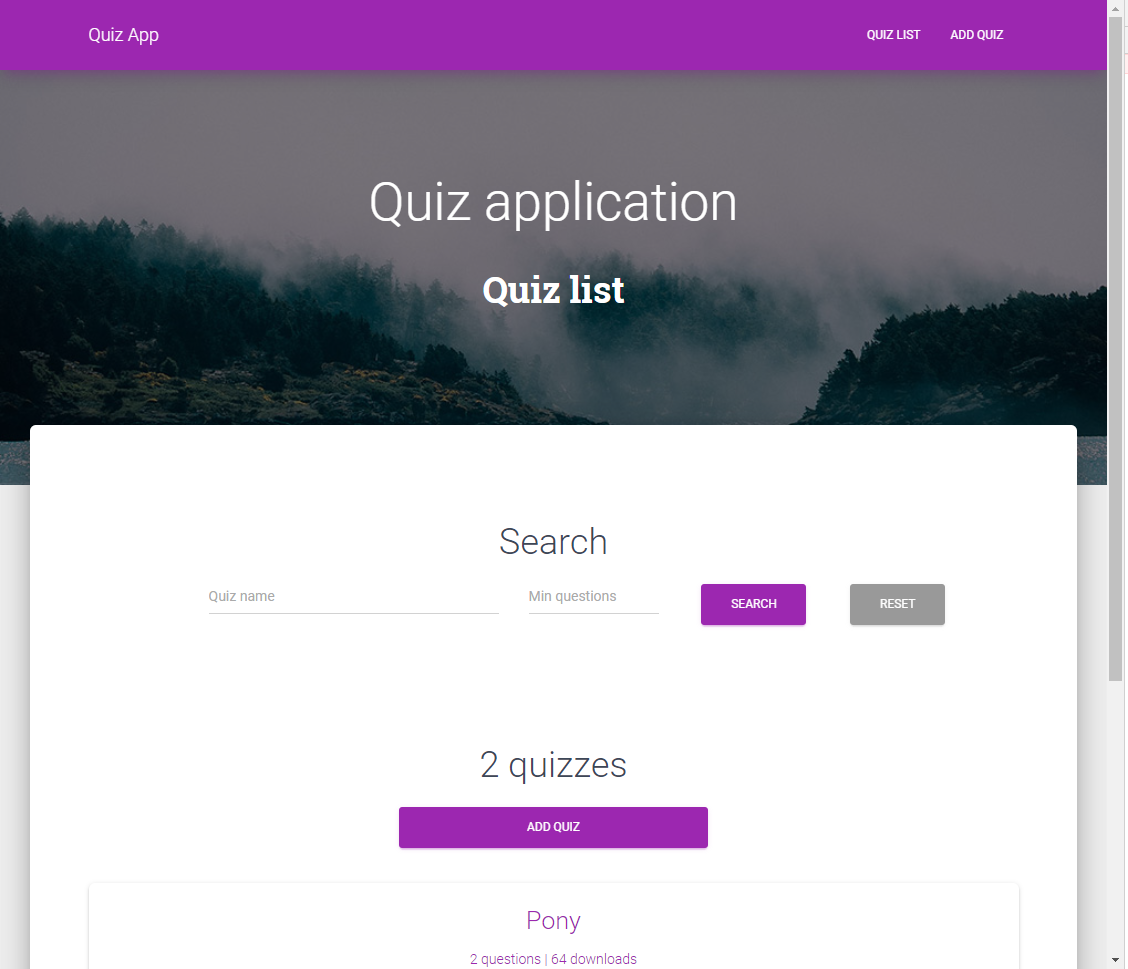


Рисунок 5.4 – Главная страница веб-приложения

После выбора любой из предварительно созданных на сайте викторин, осуществляется переход на страницу просмотра информации о выбранной викторине (рисунок 5.5).

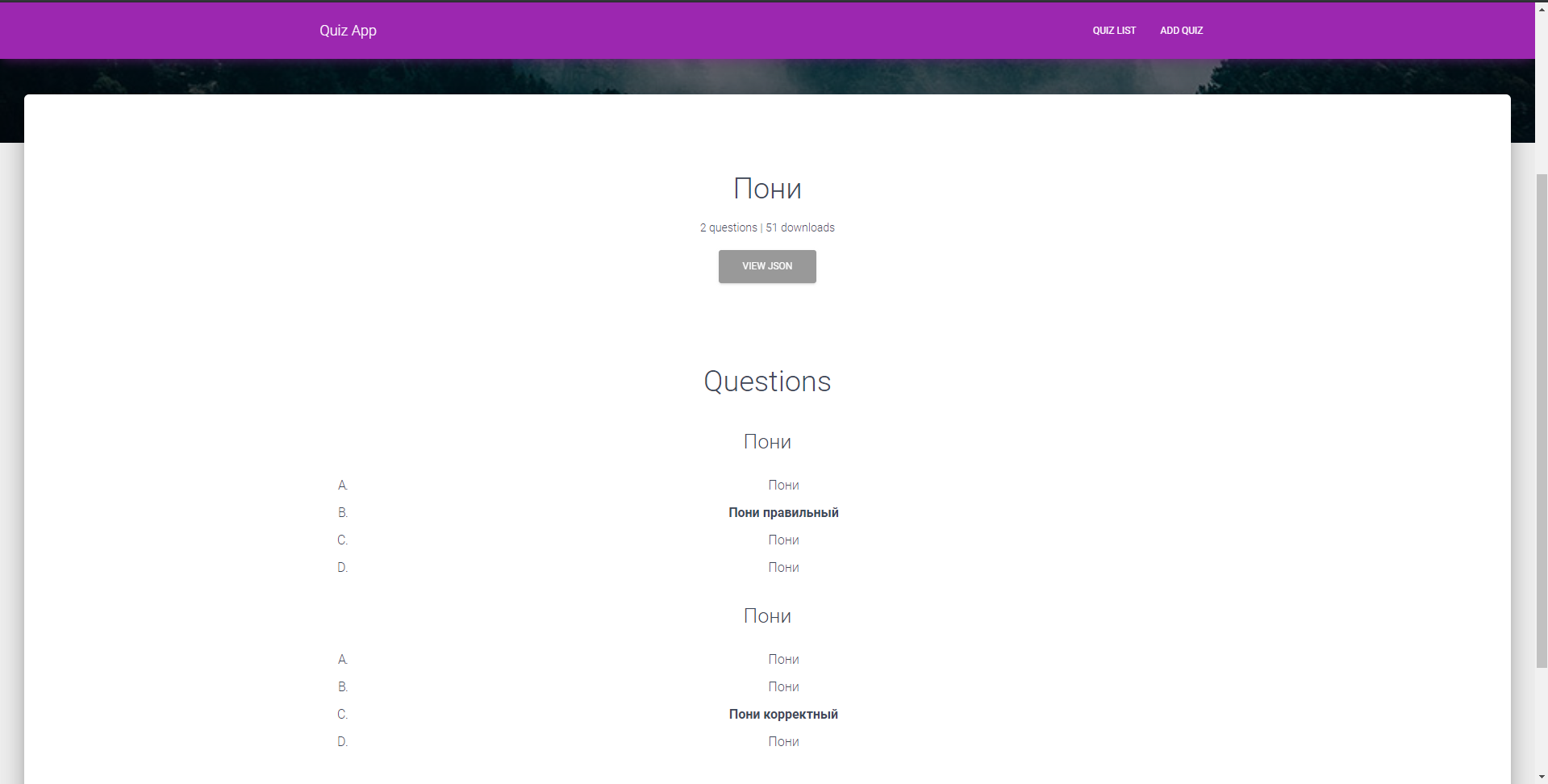


Рисунок 5.5 – Страница просмотра информации о викторине

При нажатии на главной странице либо в правом верхнем углу экрана на кнопку создания викторины, осуществляется переход на страницу, где пользователь может ввести количество вопросов, а после этого перейти непосредственно к заполнению викторины (рисунок 5.6).

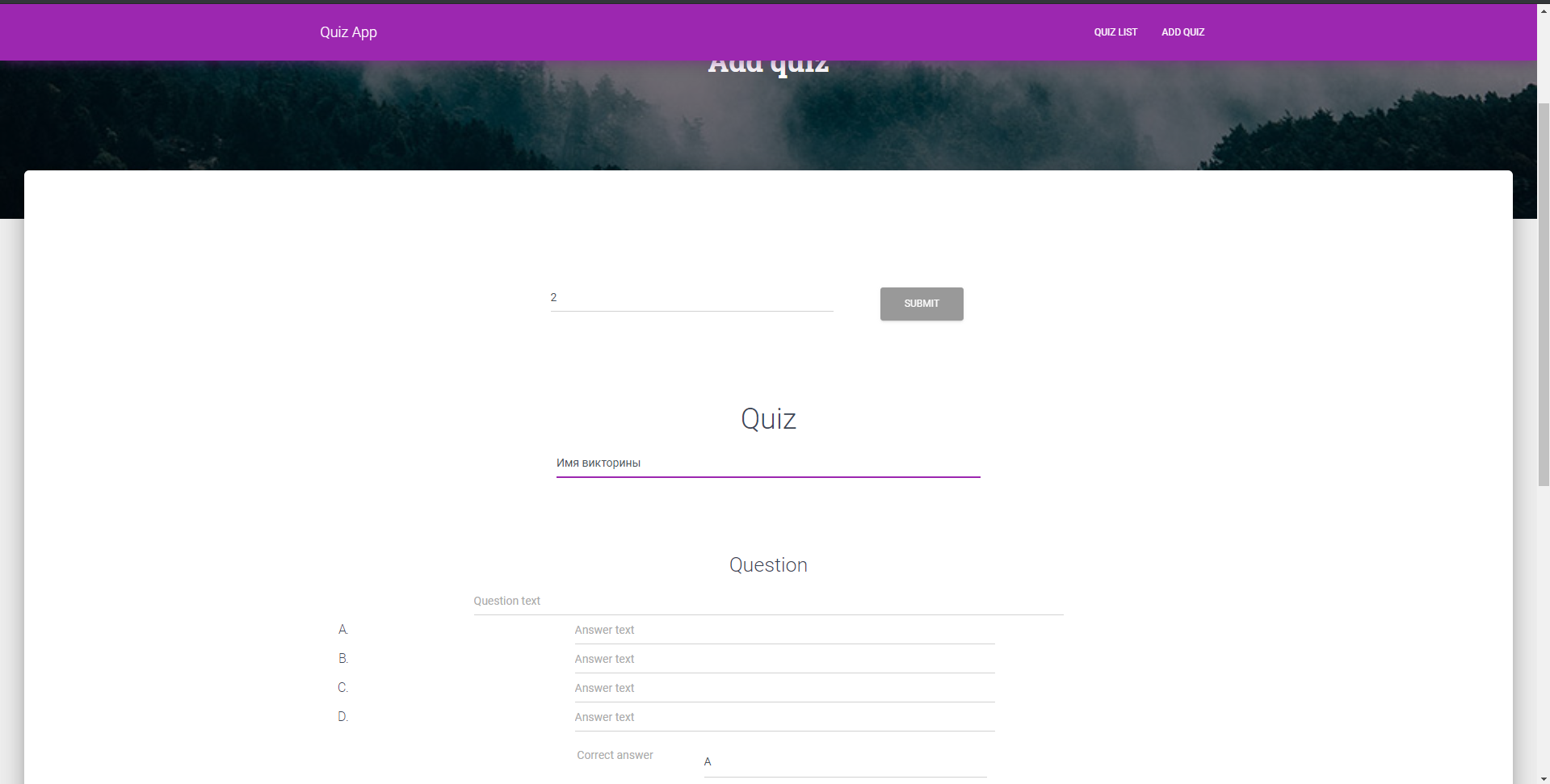


Рисунок 5.6 – Страница создания викторины

Всё вышеперечисленное позволяет пользователю быстро адаптироваться к работе с приложением и всеми его функциями и сразу приступить к работе с ним, без необходимости разбираться в тонкой настройке или проходя через большое количество шагов при каждом запуске или доступе к функциям.

6 Охрана труда

**6.1 Организация охраны труда на предприятии, производящем системы учебно-игрового назначения**

**6.1.1 Организация службы охраны труда и ее права**

Для организации охраны труда и осуществления контроля за соблюдением законодательства об охране труда, наниматель в установленном законодательством порядке создает службу охраны труда, вводит в штат должность специалиста по охране труда или возлагает соответствующие обязанности по охране труда на уполномоченное им должностное лицо либо привлекает юридическое лицо (индивидуального предпринимателя), аккредитованное (аккредитованного) на оказание услуг в области охраны труда, в соответствии с законодательством [8].

Структура и численность службы охраны труда устанавливаются в зависимости от численности работников, характера и степени опасности факторов производственной среды и трудового процесса, наличия опасных производственных объектов, работ с повышенной опасностью.

Республиканские органы государственного управления и иные государственные организации, подчиненные Правительству Республики Беларусь, местные исполнительные и распорядительные органы создают службу охраны труда или вводят в штат должность специалиста по охране труда в пределах установленной численности работников.

Должности специалистов по охране труда в организациях производственной сферы вводятся при численности работников свыше 100 человек, в организациях других сфер деятельности – свыше 200 человек. При меньшей численности работников наниматель может вводить должность специалиста по охране труда или возлагать соответствующие обязанности по охране труда на уполномоченное им должностное лицо, имеющее необходимую подготовку. В случае невозможности исполнения обязанностей специалиста по охране труда работниками организации наниматель может привлекать юридическое лицо (индивидуального предпринимателя), аккредитованное (аккредитованного) на оказание услуг в области охраны труда, в соответствии с законодательством.

Служба охраны труда (специалист по охране труда) подчиняется непосредственно руководителю организации или уполномоченному в соответствии с системой управления охраной труда его заместителю. Служба охраны труда (специалист по охране труда) организует работу по охране труда в соответствии с настоящим Законом, иными актами законодательства. Специалисты по охране труда кроме выполнения своих трудовых функций могут привлекаться только для ликвидации чрезвычайных ситуаций.

Типовое положение о службе охраны труда утверждается Министерством труда и социальной защиты Республики Беларусь.

Нормативы численности специалистов по охране труда разрабатываются и утверждаются Министерством труда и социальной защиты Республики Беларусь.

Исключение из структуры организации службы охраны труда, сокращение должности специалиста по охране труда допускаются только в случаях реорганизации (ликвидации) организации, сокращения численности или штата работников.

**6.1.2 Организация комиссии по охране труда на предприятии**

Комиссия по охране труда может быть создана в организации, где имеется профсоюзный комитет (профсоюз). При отсутствии в организации профсоюза такая комиссия не может быть создана. Комиссия по охране труда создается по инициативе нанимателя и (или) по инициативе профсоюза. В ее состав на паритетной основе входят представители нанимателя и профсоюза. Создание такой комиссии не является обязательным.

Типовое положение о комиссии по охране труда, утвержденное постановлением Министерства труда и социальной защиты Республики Беларусь от 28.11.2013 № 114 (далее - Типовое положение о комиссии), не содержит норм о количестве и должностях (профессиях) работников, которые могут быть включены в состав комиссии. В связи с этим порядок формирования, методы и формы работы и т.п. комиссии по охране труда определяются в самой организации нанимателем и профсоюзом. Целесообразно формировать состав комиссии по охране труда из равного количества представителей нанимателя (например, инженера по охране труда, заместителя руководителя, главного инженера) и профсоюза, определяемого по соглашению сторон. Количество членов комиссии законодательством не установлено.

Руководство комиссией и организация ее работы осуществляется председателем комиссии. Председатель (заместитель председателя) и секретарь комиссии избираются из числа ее членов на первом заседании комиссии.

Комиссия организации по охране труда не может создаваться приказом руководителя организации: это означало бы, что не соблюден принцип партнерских отношений. Поэтому состав представителей нанимателя определяется его приказом, а состав представителей профсоюза - его решением, содержащимся в протоколе. В течение срока полномочий комиссии, изменения ее состава также оформляются приказами руководителя организации и решениями профсоюзного комитета.

Является возможным формировать состав комиссии по охране труда на срок полномочий профсоюзного комитета либо на срок действия коллективного договора.

Обоснованием установления срока полномочий комиссии на срок действия коллективного договора служит то обстоятельство, что указанная комиссия участвует в разработке раздела об охране труда коллективного договора, плана мероприятий по охране труда, прилагаемого к коллективному договору, и иных документов.

Комиссия по охране труда принимает участие:

* в разработке системы управления охраной труда;
* в разработке раздела об охране труда коллективного договора, соглашения;
* в разработке плана мероприятий по охране труда;
* в совместных действиях нанимателя и работников по обеспечению требований по охране труда, предупреждению производственного травматизма и профессиональных заболеваний;
* в проведении проверок условий и охраны труда на рабочих местах и информировании работников об их результатах.

Комиссии по охране труда предоставлено право:

* принимать решения, необходимые для организации работы по охране труда и осуществления контроля за соблюдением законодательства об охране труда;
* проводить анализ соответствия законодательству об охране труда принимаемых ЛНПА по охране труда;
* рассматривать на заседаниях результаты проведенных мероприятий по охране труда;
* заслушивать на заседаниях информацию должностных лиц организации о фактах возникновения угрозы жизни или здоровью работающих, о проводимой работе по профилактике производственного травматизма и т.д.;
* запрашивать и получать от представителей сторон информацию, документы и материалы, необходимые для осуществления деятельности комиссии;
* вносить предложения по совершенствованию работы по охране труда и другим вопросам, отнесенным к компетенции комиссии по охране труда [9].

Заседания комиссии проводятся по мере необходимости, но не реже одного раза в шесть месяцев. Ее решения оформляются протоколом, который подписывается председателем и присутствующими на заседании членами. Решение Комиссии в течение трех рабочих дней после его принятия направляется.

**6.1.3 Основные мероприятия по оптимизации условий труда**

Для обеспечения безопасности труда и предупреждения профессиональных заболеваний, а также в целях охраны здоровья работники, занятые на работах с вредными и (или) опасными условиями труда или на работах, где в соответствии с законодательством есть необходимость в профессиональном отборе, проходят предварительные (при поступлении на работу) и периодические (в течение трудовой деятельности) обязательные медицинские осмотры, а также внеочередные медицинские осмотры при ухудшении состояния здоровья.

Порядок проведения обязательных медицинских осмотров работников устанавливается республиканским органом государственного управления, проводящим государственную политику в области здравоохранения, по согласованию с республиканским органом государственного управления, проводящим государственную политику в области труда.

Работники, занятые на работах с повышенной опасностью, проходят предсменный медицинский осмотр либо освидетельствование на предмет нахождения в состоянии алкогольного, наркотического или токсического опьянения. Наниматель обязан организовать проведение медицинских осмотров в установленном законодательством порядке.

Расходы по проведению медицинских осмотров работников несет наниматель.

На время прохождения предусмотренных настоящей статьей медицинских осмотров за работником сохраняются его место работы (должность) и средний заработок.

Также типовыми мероприятиями являются:

* проведение инструктажей (вводного, первичного, повторного, внепланового, целевого);
* использование средств коллективной защиты;
* применение средств индивидуальной защиты;
* установка агитирующих плакатов специального назначения.

Всё вышеперечисленное в совокупности позволяет добиться необходимой степени соблюдения охраны труда на предприятии.

**7 Экономическая часть. Технико-экономическое обоснование разработки программного средства**

**7.1 Общая характеристика разрабатываемого программного средства**

На основных площадках для размещения приложений не было замечено полноценных аналогов разрабатываемого программного средства для учебно-образовательных викторин, а также в связи с высокой актуальностью и востребованностью учебно-образовательных приложений [10], было принято решение разрабатывать данное программное средство. Направление получения знаний с помощью сети интернет становится всё только более востребованным с течением времени.

Программное средство для учебно-образовательных викторин включает в себя приложение для мобильной операционной системы Андроид и клиент-серверное веб-приложение.

В рамках дипломного проекта поставлена задача разработать программное средство для учебно-образовательных викторин, которое будет состоять из:

* мобильного приложения – созданное для мобильной платформы Андроид, его основной целью является само отображение викторин и предоставление возможностей для их прохождения;
* веб-приложения – приложения, доступного для использования с помощью интернет-браузера (требует наличие сети интернет). Для программного средства для учебно-образовательных викторин – будет состоять из двух частей: клиентской и серверной, работающих вместе.

Викторины необходимо загружать с сервера веб-приложения через мобильное, однако вместе с мобильным приложением идёт в комплекте несколько предварительно созданных викторин. Таким образом, это характеризует разрабатываемое программное средство как пригодное для демонстрации основных функциональных возможностей даже без наличия соединения с сервером веб-приложения.

Android-приложение необходимо для скачивания и запуска викторин, оно должно обладающее всеми необходимыми операциями для полнофункционального взаимодействия с викторинами;

Клиентская часть веб-приложения, должна:

* отображать меню по созданию вопросов для викторин, ответов на созданные вопросы
* предоставлять полноценный пользовательский интерфейс по занесению вопросов в викторины;
* давать пользователю возможность редактирования уже созданных викторин;
* давать возможность редактирования дополнительных сведений о викторине;
* в удобной для пользования форме отображать список уже созданных викторин;
* осуществлять проверку на валидность перед отправкой запроса к серверной части как о добавлении викторины в базу данных, так и во время других запросов.

# 7.2 Расчет затрат на разработку программного средства

# 7.2.1 Затраты на основную заработную плату команды разработчиков

Определяются исходя из состава и численности исполнителей, размеров часовой тарифной ставки, количество часов работы в день, эффективного фонда рабочего времен и коэффициента премирования каждого из участников разработки программного средства.

Расчет основной заработной платы исполнителей осуществляется по формуле:

Зо =, (7.1)

где n – количество исполнителей, занятых разработкой ПС;

Тчi – часовая тарифная ставка i-го исполнителя, тыс. руб.;

Тч – количество часов работы в день, ч.;

Фэi – эффективный фонд рабочего времени i-го исполнителя, дн.;

К – коэффициент премирования.

Зо1 = 633,6,

Зо2 = = 211,2,

Зо3 =.

Зо = Зо1 + Зо2 + Зо3 = 633,6 + 211,2+ 369,6 = 1214,4

Расчет затрат на основную заработную плату рекомендуется осуществлять в форме таблицы 7.1.

Таблица 7.1 – Расчет затрат на основную заработную плату исполнителей

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Исполни тели | Разряд | Тарифный коэффи циент (Тк) | Месячная тарифная  ставка (Тм), руб. | Часовая тарифная  ставка (Тч), руб. | Эффективный фонд рабочего  времени (Фэi), дн. | Коэффи циент премий | Заработная плата, руб. |
| Программист | 11 | 3,98 | 370,14 | 2,2 | 24 | 1,5 | 633,6 |
| Дизайнер | 11 | 3,98 | 370,14 | 2,2 | 8 | 1,5 | 211,2 |
| Тестировщик | 11 | 3,98 | 370,14 | 2,2 | 14 | 1,5 | 369,6 |
| Основная заработная плата (Зо), руб. | | | | | | | 1214,4 |

**7.2.2 Затраты на дополнительную заработную плату исполнителей** включает выплаты, предусмотренные законодательством о труде (оплата трудовых отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей), и определяется по формуле:

Зд = , (7.2)

где Зд – затраты на основную заработную плату, (руб.);

Нд – норматив дополнительной заработной платы, рекомендуется брать в пределах 10-20%.

Зд = = 121,44.

**7.2.3 Отчисления на социальные нужды** (в фонд социальной защиты населения и на обязательное страхование) определяются в соответствии с действующими законодательными актами по формуле:

Зсз = , (7.3)

где Нсз – норматив отчислений на социальные нужды, % (согласно действующему законодательству).

Зсз = = 462,2.

**7.2.4 Расходы по статье «Машинное время» (Рм)** включают оплату машинного времени, необходимого для разработки и отладки ПС, которое определяется по нормативам (в машино-часах) на 100 строк исходного кода машинного времени в зависимости от характера решаемых задач и типа ПЭВМ:

Рм = Цм ∙ Тч ∙ Ср, (7.4)

где Цм – цена одного машинного времени, м-ч, руб;

Тч – количество часов работы в день, дн;

Ср – длительность проекта, дн.

Рм = 1 ∙ 8 ∙ 46 = 368.

Полная сумма затрат на разработку программного обеспечения находится путем суммирования всех рассчитанных статей затрат (см. таблицу 7.2).

Таблица 7.2 – Затраты на разработку программного обеспечения

|  |  |
| --- | --- |
| Статья затрат | Сумма, руб. |
| Основная заработная плата команды разработчиков | 1214,4 |
| Дополнительная заработная плата команды разработчиков | 121,44 |
| Отчисления на социальные нужды | 462,2 |
| Прочие затраты | 368 |
| Общая сумма затрат на разработку | 2166,04 |

# 7.3 Расчёт стоимостной оценки результата

Результатом (Р) использования программного средства является прирост чистой прибыли и амортизационных отчислений.

# 7.3.1 Расчёт прироста чистой прибыли

Прирост чистой прибыли представляет собой экономию эксплуатационных затрат (электроэнергия, затраты на текущий ремонт, затраты на обслуживание), прямых затрат в результате роста производительности труда, материальных затрат.

Применение программного средства позволит получить экономию на прямых затратах, расчёт которых можно осуществить следующим образом.

Прирост прибыли за счёт экономии расходов на заработную плату и отчислений на социальное страхование в результате снижения трудоёмкости.

Эз = 𝑁п ∙ (𝑡𝑐 ∙ 𝑇𝑐 − 𝑡н ∙ 𝑇н) ∙ Кпр ∙ (1 +) ∙ (1 + ) (7.5)

где Nп – плановый объем работ;

tc, tн – трудоемкость выполнения работы до и после внедрения системы, нормо-час;

Тс, Тн – часовая тарифная ставка, соответствующая разряду выполняемых работ до и после внедрения системы, 1,38 руб/час;

Кпр – коэффициент премий (Кпр=1,5 - 2);

Нд – норматив дополнительной заработной платы (Нд=10 - 20%);

Нсз – норматив отчислений от фонда оплаты труда, включаемых в себестоимость Нсз=34%.

Эз = 263 ∙ (5 ∙1,38 − 0,25 ∙ 1,38) ∙ 1,5 ∙ (1 +) ∙ (1 + ) = 3825,83.

Прирост чистой прибыли рассчитывается по формуле:

(7.6)

где n – виды затрат, по которым получена экономия;

Эi – сумма экономии, полученная за счет снижения i-го вида затрат, р.;

Нп – ставка налога на прибыль (Нп=18%).

Прирост чистой прибыли по годам эксплуатации имеет одинаковое значение.

# 7.3.2 Расчёт прироста амортизационных отчислений

Амортизационные отчисления являются источником погашения инвестиций в приобретение нового изделия.

Расчёт амортизационных отчислений осуществляется по формуле:

А = (7.7)

где На – норма амортизации используемой техники (На=8 -15%);

З – затраты на разработку программного обеспечения.

А =

# 7.4 Расчёт показателей экономической эффективности проекта

При оценке эффективности инвестиционных проектов необходимо осуществить приведение затрат и результатов, полученных в разные периоды времени, к расчётному году путём умножения затрат и результатов на коэффициент дисконтирования 𝛼𝑡, который определяется следующим образом:

𝛼𝑡 = (1 + Ен)−𝑡, (7.8)

где Ен – требуемая норма дисконта (Ен=11% );

t – порядковый номер года, затраты и результаты которого приводятся к расчётному году;

t р – расчётный год (t р=1).

𝛼1 = (1 + 0,11)1−1 = 1,

𝛼2 = (1 + 0,11)1−2= 0,9,

𝛼3 = (1 + 0,11)1−3= 0,81,

𝛼4 = (1 + 0,11)1−4= 0,73.

В связи с тем, что систему планируется внедрять на предприятие во второй половине 2019 года, предполагаемая прибыль составит 75% (3137,18 \* 0,75 = 2352,89).

Расчёт чистого дисконтированного дохода за четыре года реализации проекта и срока окупаемости инвестиций представлены в таблице 2.3.

Таблица 7.3 – Расчет экономического эффекта от использования нового ПО

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование показателей | Усл.  обоз. | По годам производства | | | |
| 1-й | 2-й | 3-й | 4-й |
| Результат | | | | | |
| 1. Прирост прибыли за счет экономии затрат | Пч | 2352,89 | 3137,18 | 3137,18 | 3137,18 |
| 2. Прирост амортизационных отчислений | А | 259,92 | 259,92 | 259,92 | 259,92 |
| 3. Прирост результата | Pt | 2612,81 | 3397,1 | 3397,1 | 3397,1 |
| 4. Коэффициент дисконтирования | 𝛼𝑡 | 1 | 0,9 | 0,81 | 0,73 |
| 5. Результат с учетом фактора времени | Pt𝛼𝑡 | 2162,81 | 3057,39 | 2751,65 | 2479,88 |
| Затраты (инвестиции) | | | | | |
| 6. Инвестиции в разработку программного продукта | З | 2166,04 | - | - | - |
| 7. Инвестиции с учетом фактора времени | Зt𝛼𝑡 | 2166,04 | - | - | - |
| 8. Чистый дисконтированный доход по годам | ЧДДt | 446,77 | 3057,39 | 2751,65 | 2479,88 |
| 9. ЧДД с нарастающим итогом | ЧДД | 446,77 | 3504,16 | 6255,81 | 8735,69 |

В результате технико-экономического обоснования применения программного средства были получены следующие значения показателей их эффективности:

* чистый дисконтированный доход за четыре года производства продукции составит 8735,69;
* Все затраты на разработку программного продукта окупаются на первом-втором году его использования.

Таким образом, внедрение программного средства является эффективным и экономически целесообразным.

**Заключение**

Целью дипломного проекта являлась разработка программного средства для учебно-образовательных викторин, которое состояло бы как из приложения для мобильной операционной системы Андроид, так и из веб-приложения.

В ходе выполнения цели дипломного проекта были выполнены следующие задачи:

* создание приложения для мобильной операционной системы Андроид, способного скачивать и воспроизводить викторины;
* создание веб-приложения, предоставляющее пользователям функционал для создания викторин;
* разработка и описание эффективного алгоритма работы программного средства, взаимодействия его частей;
* проектирование и описание реляционной базы данных программного средства со структурой таблиц, удовлетворяющих требованиям нормализации;
* демонстрация полученных в ходе обучения знаний путем выполнения предыдущих задач.

Также были выполнены такие подзадачи как проведение базового исследования предметной области, выделение основных (ключевых) абстракций, обоснование необходимости компьютерной обработки информационных средств и процессов.

Программное средство реализовано в соответствии с распространенными рекомендациями и установленными стандартами, оно способно корректно выполнять все необходимые задачи, заложенные в алгоритм.

Таким образом, все поставленные в ходе дипломного проекта задачи были выполнены в соответствии с требованиями. Созданное в процессе разработки программное средство является востребованным и в полной мере демонстрирует полученные в ходе обучения знания.

**Список использованных источников**

1 Заварыкина, И. П. Мониторинг востребованности учебно-образовательных средств на основе интегра данных учебных и библиотечно-библиографических процессов / И. П. Заварыкина, П. А. Болдырев –Екатеринбург, 2015. – 59 с.

2 Венедюхин, А. И. Принципы построения пользовательского интерфейса сайтов / А. И. Венедюхин – Москва : Эксмо, 2016. –303 c.

3 Венедюхин, А. И. Создание сайтов / А. И. Венедюхин – Москва : Эксмо, 2011. – 528 c.

4 Особенности Java с точки зрения Android-разработчика [Электронный ресурс] – Режим доступа: https://stfalcon.com – Дата доступа: 12.04.2019.

5 Python [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/python – Дата доступа: 05.05.2019.

6 Android Studio [Электронный ресурс] – Режим доступа: http://wnfx.ru – Дата доступа: 03.05.2019.

7 Особенности тестирования мобильных приложений [Электронный ресурс] – Режим доступа: https:// https://qa-academy.by – Дата доступа: 06.05.2019.

8 Залуцкий, А. И. Охрана труда. Конспект лекций / А. И. Залуцкий, Е. А. Зубелик. – Мн. : Издательство МИУ, 2011. – 325 с.

9 Вредные и опасные производственные факторы [Электронный ресурс] – Режим доступа: https://lektsii.org/ – Дата доступа: 13.05.2019.

10 Андреев В.П. Экономика производства программного обеспечения / В.П. Андреев. – Москва : Эксмо, 2014. – 40 с.

**Приложение А**

(обязательное)

Листинг программы

/\* routes.py \*/

from flask import render\_template, flash, redirect, url\_for, abort, jsonify

from app import app, db

from wtforms import FieldList, FormField

from app.forms import QuizAddForm, QuestionCountForm, QuestionAddForm, QuizSearchForm

from app.models import Answer, Question, Quiz

@app.route('/', methods=['GET', 'POST'])

@app.route('/index', methods=['GET', 'POST'])

def index():

form = QuizSearchForm()

if form.validate\_on\_submit():

name = form.name.data

quizzes = Quiz.query.filter(Quiz.name.contains(name)).limit(50).all()

else:

quizzes = Quiz.query.limit(50).all()

return render\_template('index.html', quizzes=quizzes, form=form, title='Quiz list')

@app.route('/quiz\_list')

def get\_quiz\_list():

ids = Quiz.query.limit(50).with\_entities(Quiz.id).all()

quiz\_list = []

for id in ids:

quiz\_list.append(id[0])

return jsonify(quiz\_list)

@app.route('/quiz/<int:id>')

def show\_quiz(id):

quiz = Quiz.query.get(id)

if not quiz:

abort(404)

return render\_template('quiz\_show.html', quiz=quiz, title='Quiz view')

@app.route('/quiz/<int:id>/json')

def show\_quiz\_json(id):

quiz = Quiz.query.get(id)

if not quiz:

abort(404)

quiz.downloads += 1

db.session.commit()

return jsonify(quiz.to\_dict())

@app.route('/add', methods=['GET', 'POST'])

def add\_quiz(count=None):

form = QuizAddForm()

count\_form = QuestionCountForm()

if count\_form.validate\_on\_submit():

# hack to dynamically set form's min entries count

min\_entries = count\_form.count.data

class HackQuizAddForm(QuizAddForm):pass

HackQuizAddForm.questions = FieldList(FormField(QuestionAddForm), min\_entries=min\_entries, max\_entries=20)

form = HackQuizAddForm()

return render\_template('quiz\_add.html', form=form, count\_form=count\_form, title='Add quiz')

if form.validate\_on\_submit():

quiz = Quiz(name=form.name.data)

db.session.add(quiz)

for j in range(len(form.questions.entries)):

question\_form = form.questions.entries[j]

question = Question(text=question\_form.text.data, quiz=quiz)

db.session.add(question)

for i in range(len(question\_form.answers.entries)):

answer = question\_form.answers.entries[i]

is\_correct = int(question\_form.correct\_answer.data) == i

db.session.add(Answer(text=answer.data, is\_correct=is\_correct, question=question))

db.session.commit()

return redirect(url\_for('show\_quiz', id=quiz.id))

return render\_template('quiz\_add.html', form=form, count\_form=count\_form, title='Add quiz')

@app.errorhandler(404)

def error404(error):

return render\_template('404.html')

/\*models.py\*/

from app import db

class Quiz(db.Model):

id = db.Column(db.Integer, primary\_key=True)

name = db.Column(db.String, index=True, unique=True)

downloads = db.Column(db.Integer, default=0)

questions = db.relationship('Question', backref='quiz', lazy="dynamic")

def \_\_repr\_\_(self):

return '<Quiz {} {}>'.format(self.id, self.downloads)

def to\_dict(self):

questions = []

for question in self.questions:

questions.append(question.to\_dict())

return {

"id": self.id,

"name": self.name,

"downloads": self.downloads,

"questions": questions

}

class Question(db.Model):

id = db.Column(db.Integer, primary\_key=True)

text = db.Column(db.String)

quiz\_id = db.Column(db.Integer, db.ForeignKey('quiz.id'))

answers = db.relationship('Answer', backref='question', lazy="dynamic")

def \_\_repr\_\_(self):

return '<Question {} id: {}>'.format(self.text, self.quiz\_id)

def to\_dict(self):

answers = []

for answer in self.answers:

answers.append(answer.to\_dict())

return {

"text": self.text,

"answers": answers

}

class Answer(db.Model):

id = db.Column(db.Integer, primary\_key=True)

text = db.Column(db.String)

is\_correct = db.Column(db.Boolean, index=True)

question\_id = db.Column(db.Integer, db.ForeignKey('question.id'))

def \_\_repr\_\_(self):

return '<Asnwer {} {}>'.format(self.is\_correct, self.text)

def to\_dict(self):

return {

"text": self.text,

"is\_correct": self.is\_correct

}

/\* forms.py \*/

from flask\_wtf import FlaskForm

from wtforms import StringField, PasswordField, BooleanField, SubmitField, IntegerField, FormField, FieldList, SelectField, HiddenField

from wtforms.validators import DataRequired, NumberRange, Length, Optional

class QuizSearchForm(FlaskForm):

name = StringField('Name', validators=[Length(min=3)])

min\_questions = IntegerField('Min questions', validators=[NumberRange(0, 20), Optional()])

submit = SubmitField('Search')

class QuestionCountForm(FlaskForm):

count = IntegerField('Questions count', validators=[NumberRange(2, 20)])

submit = SubmitField('Submit')

class QuestionAddForm(FlaskForm):

text = StringField('Question', validators=[DataRequired()])

answers = FieldList(StringField('Answer', validators=[DataRequired()]), min\_entries=4, max\_entries=4)

correct\_answer = SelectField('Correct answer', choices=[('0', 'A'), ('1', 'B'), ('2', 'C'), ('3', 'D')], validators=[DataRequired()])

class QuizAddForm(FlaskForm):

name = StringField('Quiz name', validators=[DataRequired()])

questions = FieldList(FormField(QuestionAddForm), max\_entries=20)

submit = SubmitField('Submit')

/\* \_\_init\_\_.py \*/

from flask import Flask

import os

from flask\_sqlalchemy import SQLAlchemy

from flask\_migrate import Migrate

basedir = os.path.abspath(os.path.dirname(\_\_file\_\_))

app = Flask(\_\_name\_\_)

app.config['SECRET\_KEY'] = 'some random string'

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///' + os.path.join(basedir, 'app.db')

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

db = SQLAlchemy(app)

migrate = Migrate(app, db)

from app import routes, models

/\* MainActivity.java \*/

package com.sheremet.quizapp;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import com.sheremet.quizapp.library.LibraryActivity;

import com.sheremet.quizapp.model.AppDatabase;

import butterknife.ButterKnife;

import butterknife.OnClick;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

ButterKnife.bind(this);

// We should initialize db here to correctly display quizzes first time

AppDatabase.getInstance(this).getQuizDao().getAll();

}

@OnClick(R.id.goToLibrary)

void goToLibrary() {

Intent intent = new Intent(this, LibraryActivity.class);

startActivity(intent);

}

@OnClick(R.id.goToSite)

void goToSite() {

Intent intent = new Intent(Intent.ACTION\_VIEW, Uri.parse("https://shrmt.pythonanywhere.com"));

startActivity(intent);

}

}

/\* LibraryActivity.java \*/

package com.sheremet.quizapp.library;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.os.Bundle;

import android.support.v7.app.AlertDialog;

import android.support.v7.app.AppCompatActivity;

import android.support.v7.widget.LinearLayoutManager;

import android.support.v7.widget.RecyclerView;

import android.support.v7.widget.RecyclerView.LayoutManager;

import android.view.View;

import com.sheremet.quizapp.R;

import com.sheremet.quizapp.game.GameActivity;

import com.sheremet.quizapp.library.LibraryContract.Presenter;

import com.sheremet.quizapp.model.Quiz;

import com.sheremet.quizapp.utils.RecyclerTouchListener;

import java.util.List;

import butterknife.BindView;

import butterknife.ButterKnife;

public class LibraryActivity

extends AppCompatActivity

implements LibraryContract.View {

public static final String QUIZ\_ID\_KEY = "quizId";

private final String logTag = LibraryActivity.class.getSimpleName();

@BindView(R.id.quizzes)

RecyclerView recyclerView;

private Presenter presenter;

private LibraryAdapter adapter;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_library);

ButterKnife.bind(this);

// Layout manager

LayoutManager layout = new LinearLayoutManager(this);

recyclerView.setLayoutManager(layout);

// Adapter

adapter = new LibraryAdapter();

recyclerView.setAdapter(adapter);

// Item touches

RecyclerTouchListener touchListener = new RecyclerTouchListener(this, recyclerView,

new RecyclerTouchListener.ClickListener() {

@Override

public void onClick(View view, int position) {

presenter.openQuiz(position);

}

@Override

public void onLongClick(View view, int position) {

presenter.showDeleteQuiz(position);

}

});

recyclerView.addOnItemTouchListener(touchListener);

// Presenter

presenter = new LibraryPresenter(this);

presenter.start();

}

@Override

protected void onDestroy() {

super.onDestroy();

presenter.destroy();

}

@Override

public Context getContext() {

return this;

}

@Override

public void setAdapterContent(List<Quiz> quizzes) {

adapter.setContent(quizzes);

}

@Override

public void addQuiz(Quiz quiz) {

adapter.addQuiz(quiz);

}

@Override

public void showDeleteDialog(final int pos) {

DialogInterface.OnClickListener deleteDialogListener =

new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int option) {

switch (option) {

case DialogInterface.BUTTON\_POSITIVE:

presenter.deleteQuiz(pos);

break;

}

}

};

new AlertDialog.Builder(this)

.setMessage("Do you really want to delete quiz?")

.setPositiveButton("Yes", deleteDialogListener)

.setNegativeButton("No", deleteDialogListener)

.show();

}

@Override

public void updateAdapterContent() {

adapter.notifyDataSetChanged();

}

@Override

public void startGame(int quizId) {

Intent intent = new Intent(this, GameActivity.class);

intent.putExtra(QUIZ\_ID\_KEY, quizId);

startActivity(intent);

}

}

/\* LibraryPresenter.java \*/

package com.sheremet.quizapp.library;

import android.util.Log;

import com.sheremet.quizapp.library.LibraryContract.View;

import com.sheremet.quizapp.model.Answer;

import com.sheremet.quizapp.model.AnswerJson;

import com.sheremet.quizapp.model.ApiCallback;

import com.sheremet.quizapp.model.AppDatabase;

import com.sheremet.quizapp.model.Question;

import com.sheremet.quizapp.model.QuestionJson;

import com.sheremet.quizapp.model.Quiz;

import com.sheremet.quizapp.model.QuizJson;

import com.sheremet.quizapp.model.QuizLoader;

import java.util.ArrayList;

import java.util.List;

public class LibraryPresenter

implements LibraryContract.Presenter {

private View view;

private List<Quiz> quizzes;

private QuizLoader quizLoader;

LibraryPresenter(View view) {

this.view = view;

}

@Override

public void start() {

final AppDatabase db = AppDatabase.getInstance(view.getContext());

quizzes = db.getQuizDao().getAll();

view.setAdapterContent(quizzes);

quizLoader = new QuizLoader();

quizLoader.downloadQuizzes(view.getContext(), new ApiCallback<QuizJson>() {

@Override

public void onReceive(QuizJson quizJson) {

Quiz quiz = quizJson.toQuiz();

if (quiz.getServerId() != 0 && db.getQuizDao().getByServerId(quiz.getServerId()) != null) {

Log.d("LibraryPresenter", quiz.getName() + " (server id: " + quiz.getServerId() + ") already exists. ");

return;

}

quiz.setId((int) db.getQuizDao().insert(quiz));

List<Question> questions = new ArrayList<>();

List<Answer> answers = new ArrayList<>();

for (QuestionJson q: quizJson.questions) {

Question question = q.toQuestion(quiz);

questions.add(question);

question.setId((int) db.getQuestionDao().insert(question));

int correctAnswers = 0;

for (AnswerJson a: q.answers) {

Answer answer = a.toAnswer(question);

if (answer.getCorrect()) {

correctAnswers += 1;

}

answers.add(answer);

}

if (correctAnswers == 0) {

Log.e(quiz.getName(), "No correct answer for question " + question.getQuestionText());

return;

} else if (correctAnswers > 1) {

Log.e(quiz.getName(), correctAnswers + " correct answers for question " + question.getQuestionText());

return;

}

}

db.getAnswerDao().insertAll(answers.toArray(new Answer[0]));

if (questions.size() < 2 || questions.size() > 20) {

Log.e(quiz.getName(), "Wrong amount of questions! " + questions.size());

return;

} else if (answers.size() != questions.size() \* 4) {

Log.e(quiz.getName(), questions.size() + " questions, but " + answers.size() + " answers!");

return;

}

Log.d(quiz.getName(), "Successfully added quiz with " + questions.size() + " questions");

view.addQuiz(quiz);

}

});

}

@Override

public void destroy() {

view = null;

}

@Override

public void openQuiz(int pos) {

Quiz quiz = quizzes.get(pos);

int quizId = quiz.getId();

view.startGame(quizId);

}

@Override

public void showDeleteQuiz(int pos) {

view.showDeleteDialog(pos);

}

@Override

public void deleteQuiz(int pos) {

final AppDatabase db = AppDatabase.getInstance(view.getContext());

Quiz quiz = quizzes.get(pos);

try {

db.beginTransaction();

for (Question question: db.getQuestionDao().getAll(quiz.getId())) {

db.getAnswerDao().deleteAll(question.getId());

}

db.getQuestionDao().deleteAll(quiz.getId());

db.getQuizDao().delete(quiz);

db.setTransactionSuccessful();

quizzes.remove(quiz);

view.updateAdapterContent();

} catch (Exception e) {

Log.e(quiz.getName(), "Error during transaction submit! " + e.getMessage());

} finally {

db.endTransaction();

}

}

}