**Реферат**

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*8*

*ДП 00.00.ПЗ*

*Разраб.*

*Славин Н.В.*

*Провер.*

*.*

*Сахонь Е.С.*

Консульт.

*Н. Контр.*

Нистюк О.А.

*Утверд.*

*Смелов В. В.*

Реферат

*Лит.*

*Листов*

*1*

БГТУ 74217075, 2021

Пояснительная записка дипломного проекта содержит 74 страницы, 19 рисунков, 10 таблиц, 10 источников литературы, 5 приложений.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, КВЕСТ, ANDROID, JAVA, ANDROID STUDIO, RETROFIT, VISUAL STUDIO, ASP.NET CORE, C#, SQL, REST

Целью дипломного проекта является разработка мобильного приложения для выполнения заданий и получения вознаграждения «Квест».

В первой главе проводится обзор теоретического материала и аналогичных решений, а также патентный поиск по тематике дипломного проекта

Вторая глава посвящена обоснованию выбора средств разработки и содержит описание технологий, использованных во время выполнения проекта.

В третьей главе описывается процесс разработки, принципы функционирования и назначение созданных компонент проекта.

В четвертой главе описывается проведение тестирования программного средства.

В пятой главе описано руководство пользователя, позволяющее подробно понять интерфейс программного средства.

В шестой главе проводится анализ информационной безопасности приложения.

В седьмой главе приводится расчет экономических параметров и себестоимости программного средства, разработанного в рамках дипломного проекта.

Объем графической части дипломного проекта составляет 5 листов А3.

**Abstract**

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*8*

*ДП 00.00.ПЗ*

*Разраб.*

*Славин Н.В.*

*Провер.*

*.*

*Сахонь Е.С.*

Консульт.

*Н. Контр.*

Нистюк О.А.

*Утверд.*

*Смелов В. В.*

Реферат

*Лит.*

*Листов*

*1*

БГТУ 74217075, 2021

The explanatory note of the thesis project contains 74 pages, 19 figures, 10 tables, 10 sources of literature, 5 appendices.

MOBILE APPLICATION, QUEST, ANDROID, JAVA, ANDROID STUDIO, RETROFIT, VISUAL STUDIO, ASP.NET CORE, C #, SQL, REST

The purpose of the diploma project is to develop a mobile application for completing assignments and receiving a Quest reward.

The first chapter provides an overview of theoretical material and similar solutions, as well as patent search on the topic of the diploma project.

The second chapter is devoted to the rationale for the choice of development tools and contains a description of the technologies used during the project.

The third chapter describes the development process, principles of operation and the purpose of the created project components.

The fourth chapter describes how to test a software tool.

The fifth chapter describes a user guide that provides a detailed understanding of the software interface.

The sixth chapter analyzes the information security of the application.

The seventh chapter provides the calculation of the economic parameters and the cost of the software developed within the framework of the diploma project.

The graphic part of the diploma project is 5 sheets A3.

**Содержание**

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*8*

*ДП 00.00.ПЗ*

*Разраб.*

*Славин Н.В.*

*Провер.*

*.*

*Сахонь Е.С.*

Консульт.

*Н. Контр.*

Нистюк О.А.

*Утверд.*

*Смелов В. В.*

Содержание

*Лит.*

*Листов*

*3*

БГТУ 74217075, 2021

[Введение 7](#_Toc73479047)

[1 Обзор теоретического материала и аналогичных решений 8](#_Toc73479048)

[1.1 Постановка задачи 8](#_Toc73479049)

[1.2 Актуальность задачи 8](#_Toc73479050)

[1.3 Основные сведения по теме «Фриланс» 9](#_Toc73479051)

[1.4 Обзор фриланс-биржи «FL.ru» 9](#_Toc73479052)

[1.5 Обзор фриланс-биржи «ALOT.PRO» 10](#_Toc73479053)

[1.6 Обзор фриланс-биржи «Freelancer» 12](#_Toc73479054)

[1.7 Патентный поиск по теме дипломного проекта 13](#_Toc73479055)

[1.7.1 «Электронная доска объявлений и почтовый сервер» 14](#_Toc73479056)

[1.7.2 «Система электронной доски объявлений» 14](#_Toc73479057)

[1.8 Выводы по разделу 15](#_Toc73479058)

[2 Обоснование выбора используемых технологий 16](#_Toc73479059)

[2.1 Язык программирования Java 16](#_Toc73479060)

[2.2 Платформа .NET 17](#_Toc73479061)

[2.3 Платформа ASP.NET Core 17](#_Toc73479062)

[2.4 СУБД Microsoft SQL Server 18](#_Toc73479063)

[2.5 Структурированный язык запросов SQL 19](#_Toc73479064)

[2.6 Технология Entity Framework Сore 20](#_Toc73479065)

[2.7 Язык программирования C# 21](#_Toc73479066)

[2.8 Выводы по разделу 22](#_Toc73479067)

[3 Проектирование и разработка программного средства 23](#_Toc73479068)

[3.1 Проектирование программного средства 23](#_Toc73479069)

[3.2 Файловая структура 24](#_Toc73479070)

[3.2.1 Структура Web-API 24](#_Toc73479071)

[3.2.2 Структура мобильного приложения 24](#_Toc73479072)

[3.3 Структура базы данных 25](#_Toc73479073)

[3.4 Разработка Web-API 27](#_Toc73479074)

[3.5 Разработка мобильного приложения 31](#_Toc73479075)

[3.6 Выводы по разделу 37](#_Toc73479076)

[4 Тестирование программного средства 39](#_Toc73479077)

[4.1 Авторизация пользователя 39](#_Toc73479078)

[4.2 Регистрация пользователя 40](#_Toc73479079)

[4.3 Создание квеста 41](#_Toc73479080)

[4.4 Изменение квеста 42](#_Toc73479081)

[4.5 Выводы по разделу 43](#_Toc73479082)

[5 Руководство пользователя 44](#_Toc73479083)

[5.1 Авторизация пользователя 44](#_Toc73479084)

[5.2 Регистрация пользователя 45](#_Toc73479085)

[5.3 Раздел «Все квесты» 45](#_Toc73479086)

[5.4 Меню приложения 47](#_Toc73479087)

[5.5 Раздел «Профиль» 48](#_Toc73479088)

[5.6 Создание нового квеста 49](#_Toc73479089)

[5.7 Раздел «Рейтинг пользователей» 50](#_Toc73479090)

[5.8 Выводы по разделу 52](#_Toc73479091)

[6 Анализ информационной безопасности приложения 53](#_Toc73479092)

[6.1 Определение информационной безопасности 53](#_Toc73479093)

[6.2 Хеширование паролей 54](#_Toc73479094)

[6.2.1 Определение хеширования 54](#_Toc73479095)

[6.2.2 Реализация хеширования в программном средстве 54](#_Toc73479096)

[6.3 Протокол HTTPS 55](#_Toc73479097)

[6.4. Обфускация кода 56](#_Toc73479098)

[6.4.1 Определение обфускации 56](#_Toc73479099)

[6.4.2 Реализация обфускации в программном средстве 56](#_Toc73479100)

[6.5 Запрет на запись видео и создание снимка экрана 57](#_Toc73479101)

[6.6 Выводы по разделу 57](#_Toc73479102)

[7 Экономическое обоснование проекта 58](#_Toc73479103)

[7.1 Общая характеристика разрабатываемого программного средства 58](#_Toc73479104)

[7.2 Исходные данные для проведения расчётов 58](#_Toc73479105)

[7.3 Объем программного средства 59](#_Toc73479106)

[7.4 Основная заработная плата 61](#_Toc73479107)

[7.5 Дополнительная заработная плата 61](#_Toc73479108)

[7.6 Отчисления в Фонд социальной защиты населения 61](#_Toc73479109)

[7.7 Расходы на материалы 62](#_Toc73479110)

[7.8 Расходы на оплату машинного времени 62](#_Toc73479111)

[7.9 Прочие прямые затраты 63](#_Toc73479112)

[7.10 Накладные расходы 63](#_Toc73479113)

[7.11 Сумма расходов на разработку программного средства 64](#_Toc73479114)

[7.12 Расходы на сопровождение и адаптацию 64](#_Toc73479115)

[7.13 Полная себестоимость 64](#_Toc73479116)

[7.14 Определение цены, оценка эффективности 65](#_Toc73479117)

[7.15 Вывод по разделу 66](#_Toc73479118)

[Заключение 68](#_Toc73479119)

[Список использованных источников 69](#_Toc73479120)

[ПРИЛОЖЕНИЕ А 70](#_Toc73479121)

[ПРИЛОЖЕНИЕ Б 71](#_Toc73479122)

[ПРИЛОЖЕНИЕ В 72](#_Toc73479123)

[ПРИЛОЖЕНИЕ Г 73](#_Toc73479124)

[ПРИЛОЖЕНИЕ Д 74](#_Toc73479125)

# Введение

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*8*

*ДП 00.00.ПЗ*

*Разраб.*

*Славин Н.В.*

*Провер.*

*.*

*Сахонь Е.С.*

Консульт.

*Н. Контр.*

Нистюк О.А.

*Утверд.*

*Смелов В. В.*

Введение

*Лит.*

*Листов*

*1*

БГТУ 74217075, 2021

Появление онлайн-бизнеса стало одним из важнейших этапов развития Интернета и бизнеса в целом. Благодаря *E-commerce* сформировалась новая рыночная система, кардинально отличающаяся от классической, и тысячи компаний по всей планете получили доступ к многомиллионной аудитории Сети.

Интернет превратился в универсальную платформу, позволяющую пользователям общаться, получать информацию, учиться и тратить деньги одновременно. Следующим логичным этапом его развития стала удалённая работа. Это способствовало появлению фриланса.

Платформами для удалённой работы стали биржи фриланса. Они могут стать для студентов первым источником дохода. Это позволяет в свободное от учёбы время не только получить дополнительный заработок, но и повысить практические навыки в рамках своей специальности. Для этого достаточно иметь при себе собственный ПК или ноутбук и постоянное интернет-соединение.

Целью данного дипломного проекта является разработка мобильного приложения для выполнения заданий и получения вознаграждения «Квест».

Для достижения поставленной цели необходимо решить следующие задачи:

* спроектировать и создать базу данных;
* разработать *Web-API*;
* разработать функционал регистрации и авторизации пользователей;
* обеспечить защиту данных пользователей;
* создать дизайн пользовательского интерфейса;
* разработать мобильное приложение под операционную систему *Android*.

# Обзор теоретического материала и аналогичных решений

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*8*

*ДП 01.00.ПЗ*

*Разраб.*

*Славин Н.В.*

*Провер.*

*.*

*Сахонь Е.С.*

Консульт.

*Н. Контр.*

Нистюк О.А.

*Утверд.*

*Смелов В. В.*

Обзор теоретического материала и аналогичных решений

*Лит.*

*Листов*

*8*

БГТУ 74217075, 2021

## 1.1 Постановка задачи

Целью данного дипломного проекта является разработка мобильного приложения для выполнения заданий и получения вознаграждения под операционную систему *Android*. Исходя из поставленной цели были спроектированы задачи по созданию интуитивно понятного интерфейса и программной части приложения.

Мобильное приложение должно содержать функционал:

* авторизации и регистрации пользователей;
* просмотра и изменения данных аккаунта;
* создания, удаления и редактирования квестов;
* отправки запросов на выполнение квестов;
* выбора исполнителя для созданных квестов;
* просмотра рейтинга пользователей по числу выполненных квестов;
* просмотра информации о пользователях;

## 1.2 Актуальность задачи

Изменения на современном рынке интеллектуального труда, вызванные глобализацией, привели к появлению нового сегмента мирового рынка – фриланса.

Стабильная занятость, гарантированный доход, социальные гарантии и постоянное рабочее место, отвечали требованиям и условиям, складывающимся на рынке труда в XX веке. Сегодня, в связи с развитием информационных технологий, занятость всё больше тяготеет к менее жёстким формам, распространение получают удалённая занятость, самозанятость и некоторые другие явления, общей чертой которых является снижение зависимости работника от работодателя.

Удалённая работа – это реальный шанс получать достойный доход для людей даже с ограниченными физическими возможностями, что усиливает актуальность инклюзивного образования в вузах.

Таким образом, можно сделать вывод о том, что фриланс является ресурсоэффективной технологией, позволяющей разрешить ряд организационных и управленческих проблем, актуальных для большинства современных организаций.

## 1.3 Основные сведения по теме «Фриланс»

Фрилансер – свободный работник, который выполняет разовые задачи для разных заказчиков. Фрилансер чаще всего сам предлагает свои услуги – на специализированных онлайн-ресурсах, через газетные объявления или пользуясь «сарафанным радио», то есть личными связями.

Фриланс — это механизм, суть которого заключается в том, что некое частное лицо или фирма нанимает для выполнения определенной задачи человека, не зачисляя его в штат фирмы. Работник может находиться в другом городе или даже в другой стране, но может работать и в офисах заказчика. Широкое распространение фриланс получил с развитием интернета: сеть и сопутствующие информационные и банковские технологии позволили некоторым категориям работников уменьшить частоту появления в офисах, а то и полностью перейти на надомную/удалённую работу.

Чтобы найти работу фрилансером на дому, нужно просматривать вакансии на различных площадках. Это могут быть и доски объявлений, и биржи, и социальные сети.

В поисках заказов помогут онлайн-доски объявлений. Например, Авито, Юла. Такой способ не очень эффективен, но онлайн ресурсы этого типа – неплохие места, где можно найти работу фрилансеру бесплатно, не выплачивая никаких комиссий. Со временем у специалиста устанавливается определенный процент заказов, которые он получает с подобных площадок. На таких сайтах можно найти постоянного заказчика.

Биржа – посредник между исполнителем и заказчиком. Она обеспечивает и первой, и второй стороне безопасную сделку и гарантированный результат. Для новичка биржи хороши тем, что на них можно найти первых постоянных клиентов и собрать портфолио.

Найти работу можно и на специализированных сайтах. Некоторые опытные фрилансеры и не думают о том, что найти работу можно на сайте HeadHunter. Здесь ищут не только продавцов в розничные магазины, но и копирайтеров, таргетологов, программистов. Фрилансер вправе составить резюме и выставить на этом сайте. Можно использовать и другие аналогичные ресурсы.

Когда у удаленного работника уже имеется опыт и портфолио, он может создать собственный сайт. Такая страничка станет визитной карточкой фрилансера. Важно постоянно работать над развитием сайта, наполнять его новыми готовыми работами, и по возможности создать блог на тематику фриланса.

## 1.4 Обзор фриланс-биржи «FL.ru»

FL.ru – крупнейшая русскоязычная фриланс биржа. Она помогает найти талантливых исполнителей на любое задание. С помощью неё можно опубликовать бесплатное задание и выбрать исполнителя среди миллионов фрилансеров. Доступна возможность использования «Безопасной сделки» как удобного и легального способа работать с фрилансерами из других стран, как для организаций, так и для физических лиц. Также она гарантирует возврат денег, если исполнитель не справился с заказом или сорвал сроки.

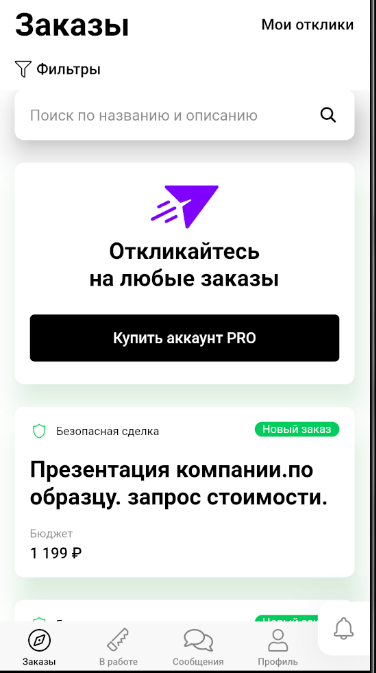


Рисунок 1.1 – Экран главной страницы в «FL.ru»

При регистрации приложение предлагает выбрать роль «Заказчик» или «Фрилансер». Для каждой роли отдельный аккаунт. Фрилансер может брать заказы, но для более лучших заказов потребуется купить PRO аккаунт. Та же ситуация и с заказчиком. Можно публиковать заказы, но для хороших заказов нужен PRO аккаунт.

К плюсам можно отнести удобный и понятный интерфейс, простоту использования и большое количество заказов.

К минусам относится то, что для хорошего заработка придётся покупать PRO аккаунт.

## 1.5 Обзор фриланс-биржи «ALOT.PRO»

Агрегатор вакансий Alot.Pro — продукт у которого нет аналогов. Начиная свой путь фриланса, люди тратят основную часть времени не на работу, а на её поиски. В подавляющем большинстве случаев это биржи фриланса и группы в социальных сетях.

Не имея отзывов, это делать ещё сложнее. Отсюда возникает проблема: чтобы иметь шансы на получение вакансии и конкурировать с другими фрилансерами, необходимо быстро реагировать на появившийся проект. Как следить за всеми сайтами сразу? Для этого и существует данная платформа, которая собирает данные со всех бирж и даже из групп «Вконтакте», чтобы вы первыми увидели свежую вакансию.

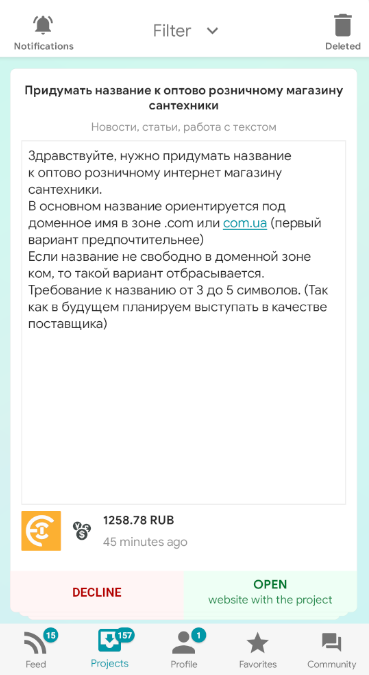


Рисунок 1.2 – Экран главной страницы в «ALOT.PRO»

Простота интерфейса как у мобильного приложения, так и у сайта позволяют сразу приступить к работе, без длительного изучения функционала.

Все собрано в 5 разделов:

1. Лента — раздел, где фрилансер найдет всю полезную информацию, свежие новости и подсказки по работе с сервисом
2. Проекты — здесь представлен весь список проектов, который формируется на фоне ваших фильтров.
3. Профиль — вся информация о вас, как о пользователи, а также, небольшой список достижений, за которые можно получить внутрисистемную валюту и потратить её на оплату Pro-версии.
4. Избранное — место, где хранятся все проекты, которые вы сюда поместили.
5. Форум — где вы всегда можете задать интересующий вас вопрос и быстро получить ответ.

Для работы на платформе вам не понадобиться тратить время на настройку и даже на регистрацию. Вам сразу предложат выбрать интересующие вас категории, после запуска приложения. После этого вы сразу сможете приступить к поиску вакансий. Если захотите синхронизировать аккаунты с разных устройств вам достаточно авторизоваться через любую из социальных сетей. Всё очень удобно.

Для доступа к сайтам проектов необходимо приобрести доступ. Стоимость месячной подписки – всего 265 рублей, а полного доступа навсегда – 1999 рублей. За эти деньги перед вами откроется возможность работать более чем с 20 сайтами в одном приложении, искать вакансии по ключевым словам и многое другое. Подключается в разделе «Профиль».

## 1.6 Обзор фриланс-биржи «Freelancer»

Freelancer, безусловно, является одним из крупнейших фриланс-сайтов в мире, и миллионы пользователей выбирали его на протяжении многих лет.

Функции и правила работы с сервисом Freelancer кому-то могут нравиться, кому-то — нет. Тем не менее, нельзя не согласиться с тем фактом, что именно у Freelancer они появились впервые, что именно Freelancer изобрел многое из того, что сегодня считается стандартом для фриланс-платформ. Тонны спама? Неадекватные отклики от неадекватных исполнителей? Freelancer по-прежнему процветает, оставляя не у дел менее “зубастых” конкурентов.

Различные баджи, уровни и сложная система опыта, используемые для ранжирования исполнителей, это уникальное изобретение Freelancer, которое многие другие сайты позаимствовали и улучшили, чтобы обойти Frelancer.com.

Для начинающего пользователя меню навигации сервиса Freelancer может показаться запутанным, однако на самом деле тут все организовано очень удобным образом. У Freelancer так много предложений для клиентов, что едва ли сервис мог обойтись меньшим количеством ссылок и кнопок. Регистрация очень простая, ее даже можно пропустить и авторизоваться с помощью учетной записи Facebook. Зарегистрироваться можно в качестве фрилансера, заказчика и в обеих этих ипостасях сразу.

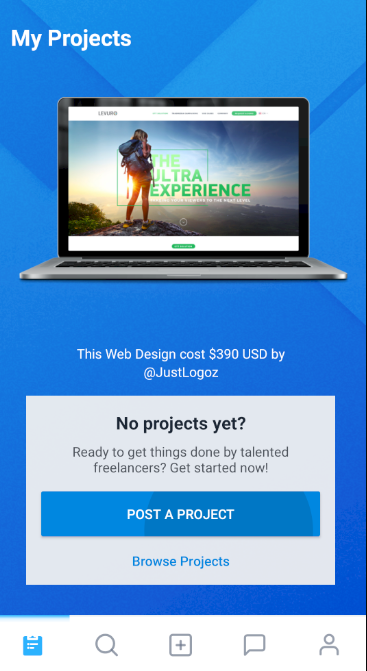


Рисунок 1.3 – Экран главной страницы в «Freelancer»

Вы начинаете работать с абсолютно бесплатным аккаунтом, но чтобы получить доступ к новым возможностям и уменьшить размер комиссии вам придется платить. Даже тесты для фрилансеров платные (от 5 долларов и выше). Вы можете заплатить, и тогда ваш проект или отклик на проект будет отображен на первом месте.

## 1.7 Патентный поиск по теме дипломного проекта

В соответствии с темой дипломного проекта был проведен патентный поиск в области приложений для размещения объявлений.

Патентный поиск – это процесс отбора соответствующих запросу документов или сведений по одному или нескольким признакам из массива патентных документов или данных, при этом осуществляется процесс поиска из множества документов и текстов только тех, которые соответствуют теме или предмету запроса.

Патентный поиск осуществляется посредством информационно-поисковой системы и выполняется вручную или с использованием соответствующих компьютерных программ, а также с привлечением соответствующих экспертов.

Предмет поиска определяют исходя из конкретных задач патентных исследований категории объекта (устройство, способ, вещество), а также из того, какие его элементы, параметры, свойства и другие характеристики предполагается исследовать.

При патентном поиске сравниваются выражения смыслового содержания информационного запроса и содержания документа.

Изобретение – это основной объект промышленной собственности.

Патент на изобретение – это документ, выдаваемый компетентным государственным органом и удостоверяющий: приоритет изобретения, авторство и исключительное право на изобретение. Действует в пределах территории того государства, ведомство которого его выдало.

Под патентоспособностью понимают юридическое свойство объекта промышленной собственности, определяющее его способность охраняться документом исключительного права (патентом) на территории конкретной страны в течение срока действия патента.

Решение является патентоспособным если:

* это способ или вещество;
* обладает новизной;
* имеет изобретательский уровень;
* является промышленно применяемым.

В качестве аналогов можно выделить: «Электронная доска объявлений и почтовый сервер» и «Система электронной доски объявлений».

## 1.7.1 «Электронная доска объявлений и почтовый сервер»

Первый патент на изобретение носит название «Электронная доска объявлений и почтовый сервер», изобретателем которого является Кикугава Сатору. Номер регистрации (свидетельства): 2001105187/09. Дата регистрации: 27.01.2005.

Изобретение относится к электронной доске объявлений и почтовому серверу. Технический результат заключается в том, что электронная доска объявлений обменивается информацией с компьютером пользователя посредством сети и служит посредником для обмена информацией между ними с помощью функции электронной доски объявлений. Электронную доску объявлений снабжают базой данных в виде совокупности известных слов, хранящей множество слов, которые выбраны соответствующим образом.

## 1.7.2 «Система электронной доски объявлений»

Второй патент на изобретение носит название «Система электронной доски объявлений», изобретателем которого является Кузнецов Юрий Дмитриевич. Номер регистрации (свидетельства): 99126436/09. Дата регистрации: 10.09.2003.

Изобретение относится к обработке информации и может применяться в сети системы электронной доски объявлений. Его использование позволяет заранее автоматически проверять содержание сообщения и автоматически сообщать о результатах проверки диспетчеру электронной доски объявлений. Технический результат достигается путем проверки сообщения, посланного от компьютера пользователя, при помощи списка запрещенных к отправке слов и списка слов, требующих осторожности. Если сообщение, которое требуется отправить, не включает запрещенных к отправке слов, сообщение отправляется на электронную доску объявлений.

## 1.8 Выводы по разделу

Данный раздел содержит информацию о аналогах разрабатываемого программного средства, был приведен краткий обзор к каждому из них, рассмотрены их возможности и недостатки, выделены плюсы и минусы.

Каждый аналог был подробно описан, и было показано, для чего предназначено то или иное программное обеспечение.

Как видно из представленного краткого обзора, большинство ресурсов имеют много общего в своих функциональных возможностях. В первую очередь это касается базовых функций: регистрация, создание заказов, взятие заказов в работу.

В итоге был сделан вывод, что некоторые особенности схожих по функционалу уже созданных программных средств недостаточны для достижения требуемого результата, в результате чего было принято решение на создание собственной разработки в рамках дипломного проекта.

# 2 Обоснование выбора используемых технологий

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*8*

*ДП 02.00.ПЗ*

*Разраб.*

*Славин Н.В.*

*Провер.*

*.*

*Сахонь Е.С.*

Консульт.

*Н. Контр.*

Нистюк О.А.

*Утверд.*

*Смелов В. В.*

Обоснование выбора используемых технологий

*Лит.*

*Листов*

*7*

БГТУ 74217075, 2021

## 2.1 Язык программирования Java

*Java* является объектно-ориентированным языком. Он поддерживает полиморфизм, наследование, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

На сегодняшний момент язык *Java* является одним из самых распространенных и популярных языков программирования. *Java* задумывался как универсальный язык программирования, который можно применять для различного рода задач. Текущей версией является *Java* 16, которая вышла в марте 2021 года. *Java* превратилась из просто универсального языка в целую платформу и экосистему, которая объединяет различные технологии, используемые для целого ряда задач: от создания десктопных приложений до написания крупных веб-порталов и сервисов. Кроме того, язык *Java* активно применяется для создания программного обеспечения для множества устройств: обычных ПК, планшетов, смартфонов и мобильных телефонов и даже бытовой техники.

Ключевой особенностью языка *Java* является то, что его код сначала транслируется в специальный байт-код, независимый от платформы. А затем этот байт-код выполняется виртуальной машиной *JVM* (*Java* *Virtual* *Machine*). В этом плане *Java* отличается от стандартных интерпретируемых языков как *PHP* или *Perl*, код которых сразу же выполняется интерпретатором. В то же время Java не является и чисто компилируемым языком, как *С* или *С*++.

Подобная архитектура обеспечивает кроссплатформенность и аппаратную переносимость программ на *Java*, благодаря чему подобные программы без перекомпиляции могут выполняться на различных платформах - *Windows*, *Linux*, *Mac* *OS* и т.д. Для каждой из платформ может быть своя реализация виртуальной машины *JVM*, но каждая из них может выполнять один и тот же код.

### 2.2 Платформа .NET

*.NET Framework* – это программная платформа, выпущенная компанией Microsoft, которая подходит для разных языков программирования. Основой платформы является общеязыковая среда исполнения *Common* *Language* *Runtime* (*CLR*), которая подходит для различных языков программирования. Функциональные возможности *CLR* доступны в любых языках программирования, использующих эту среду.

Программа для *.NET Framework*, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для *.NET* промежуточный байт-код *Common* *Intermediate* *Language* (*CIL*) (ранее назывался *Microsoft* *Intermediate* *Language*, *MSIL*). Затем код либо исполняется виртуальной машиной *Common* *Language* *Runtime* (*CLR*), либо транслируется утилитой *NGen*.*exe* в исполняемый код для конкретного целевого процессора. В случае использования виртуальной машины *CLR* встроенный в неё *JIT*-компилятор «на лету» (*just* *in* *time*) преобразует промежуточный байт-код в машинные коды нужного процессора. Современная технология динамической компиляции позволяет достигнуть высокого уровня быстродействия. Виртуальная машина *CLR* также сама заботится о базовой безопасности, управлении памятью и системе исключений.

### 2.3 Платформа ASP.NET Core

*ASP.NET Core* – свободно-распространяемый кроссплатформенный фреймворк для создания веб-приложений с открытым исходным кодом. Данная платформа разрабатывается компанией Майкрософт совместно с сообществом и имеет большую производительность по сравнению с *ASP.NET*.

Имеет модульную структуру и совместима с такими операционными системами как *Windows*, *Linux* и *MacOS*.

*ASP.NET Core* включает в себя фреймворк *MVC*, который объединяет функциональность *MVC*, *Web API* и *Web Pages*. В предыдущих версиях платформы данные технологии реализовались отдельно и поэтому содержали много дублирующей функциональности. Сейчас же они объединены в одну программную модель *ASP.NET Core MVC*. А *Web Forms* полностью ушли в прошлое. Благодаря модульности фреймворка все необходимые компоненты веб-приложения могут загружаться как отдельные модули через пакетный менеджер *Nuget*. Кроме того, в отличие от предыдущих версий платформы нет необходимости использовать библиотеку *System.Web.dll*. Некоторые особенности *ASP.NET*:

* распространение пакетов платформы через *NuGet*;
* интегрированная поддержка для создания и использования пакетов *NuGet*;
* единый стек веб-разработки, сочетающий *Web UI* и *Web API*;
* конфигурация для упрощенного использования в облаке;
* встроенная поддержка для внедрения зависимостей;
* поддержка *CRUD*-операций при работе с таблицами через *GridView*;
* возможно создание веб-приложений, которые реализуют шаблон *Model-View-Controller* (*ASP.NET MVC Framework*).

### 2.4 СУБД Microsoft SQL Server

*Microsoft* *SQL* *Server* – система управления реляционными базами данных (СУБД), разработанная корпорацией *Microsoft*. Основной используемый язык запросов – *Transact*-*SQL*, создан совместно *Microsoft* и *Sybase*. *Transact*-*SQL* является реализацией стандарта *ANSI*/*ISO* по структурированному языку запросов (*SQL*) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз, данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Сервер баз, данных *Microsoft* *SQL* *Server* в качестве языка запросов использует версию языка *SQL*, получившую название *Transact*-*SQL* (сокращённо *T*-*SQL*). Язык *T*-*SQL* является реализацией *SQL*-92 (стандарт *ISO* для языка *SQL*) с множественными расширениями. *T*-*SQL* позволяет использовать дополнительный синтаксис для хранимых процедур и обеспечивает поддержку транзакций (взаимодействие базы данных с управляющим приложением).

*SQL* *Server* является одной из наиболее популярных систем управления базами данных (СУБД) в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

*SQL* *Server* был создан компанией *Microsoft*. Первая версия вышла в 1987 году. А текущей версией является версия 16, которая вышла в 2016 году и которая будет использоваться в текущем руководстве.

*SQL* *Server* долгое время был исключительно системой управления базами данных для *Windows*, однако начиная с версии 16 эта система доступна и на Linux.

*SQL* *Server* характеризуется такими особенностями как:

* *SQL* *Server* работает очень быстро;
* *SQL* *Server* предоставляет шифрование данных;
* с данной СУБД относительно легко работать и вести администрирование.

Центральным аспектом в *MS* *SQL* *Server*, как и в любой СУБД, является база данных. База данных представляет хранилище данных, организованных определенным способом. Нередко физически база данных представляет файл на жестком диске, хотя такое соответствие необязательно. Для хранения и администрирования баз, данных применяются системы управления базами данных (*database* *management* *system*) или СУБД (*DBMS*). И как раз *MS* *SQL* *Server* является одной из такой СУБД.

Для организации баз данных *MS* *SQL* *Server* использует реляционную модель. Эта модель баз данных была разработана еще в 1970 году Эдгаром Коддом. А на сегодняшний день она фактически является стандартом для организации баз данных.

Реляционная модель предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта

### 2.5 Структурированный язык запросов SQL

*SQL* (*Structured Query Language* – Структурированный язык запросов) – язык управления базами данных для реляционных баз данных. Сам по себе *SQL* не является тьюринг-полным языком программирования, но его стандарт позволяет создавать для него процедурные расширения, которые расширяют его функциональность до полноценного языка программирования.

Язык был создан в 1970х годах под названием «*SEQUEL*» для системы управления базами данных (СУБД) *System R*. Позднее он был переименован в «*SQL*» во избежание конфликта торговых марок. В 1979 году *SQL* был впервые опубликован в виде коммерческого продукта *Oracle V2*.

*SQL* создавался как простой стандартизированный способ извлечения и управления данными, содержащимися в реляционной базе данных. Позднее он стал сложнее, чем задумывался, и превратился в инструмент разработчика, а не конечного пользователя. В настоящее время SQL (по большей части в реализации Oracle) остается самым популярным из языков управления базами данных, хотя и существует ряд альтернатив.

*SQL* состоит из четырех отдельных частей.

Язык определения данных (*DDL*) используется для определения структур данных, хранящихся в базе данных. Операторы *DDL* позволяют создавать, изменять и удалять отдельные объекты в БД. Допустимые типы объектов зависят от используемой СУБД и обычно включают базы данных, пользователей, таблицы и ряд более мелких вспомогательных объектов, например, роли и индексы.

Язык манипуляции данными (*DML*) используется для извлечения и изменения данных в БД. Операторы *DML* позволяют извлекать, вставлять, изменять и удалять данные в таблицах. Иногда операторы select извлечения данных не рассматриваются как часть *DML*, поскольку они не изменяют состояние данных. Все операторы DML носят декларативный характер.

Язык определения доступа к данным (*DCL*) используется для контроля доступа к данным в БД. Операторы *DCL* применяются к привилегиям и позволяют выдавать и отбирать права на применение определенных операторов *DDL* и *DML* к определенным объектам БД.

Язык управления транзакциями (*TCL*) используется для контроля обработки транзакций в БД. Обычно операторы *TCL* включают *commit* для подтверждения изменений, сделанных в ходе транзакции, *rollback* для их отмены и *savepoint* для разбиения транзакции на несколько меньших частей.

### 2.6 Технология Entity Framework Сore

*Entity Framework Core* (*EF* *Core*) представляет собой объектно-ориентированную, легковесную и расширяемую технологию от компании *Microsoft* для доступа к данным. *EF Core* является *ORM*-инструментом (*object-relational mapping* - отображения данных на реальные объекты). То есть *EF Core* позволяет работать базами данных, но представляет собой более высокий уровень абстракции: *EF Core* позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными независимо от типа хранилища. Если на физическом уровне пользователь оперирует таблицами, индексами, первичными и внешними ключами, то на концептуальном уровне, который предлагает *Entity Framework*, пользователь уже работает с объектами. Центральной концепцией *Entity Framework* является понятие сущности или *entity*. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами. *Entity Framework Core* поддерживает множество различных систем баз данных. Таким образом, можно с помощью *EF Core* работать с любой СУБД, если для нее имеется нужный провайдер.

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

Также стоит отметить, что *EF* *Core* предоставляет универсальный *API* для работы с данными. И если, к примеру, пользователь решит сменить целевую СУБД, то основные изменения в проекте будут касаться прежде всего конфигурации и настройки подключения к соответствующим провайдерам. А код, который непосредственно работает с данными, получает данные, добавляет их в БД и т.д., останется прежним.

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

При этом сущности могут быть связаны ассоциативной связью один-ко-многим, один-ко-одному и многие-ко-многим, подобно тому, как в реальной базе данных происходит связь через внешние ключи.

Отличительной чертой *Entity Framework Core* является использование запросов *LINQ* для выборки данных из БД. С помощью *LINQ* можно не только извлекать определенные строки, хранящие объекты, из бд, но и получать объекты, связанные различными ассоциативными связями.

Другим ключевым понятием является *Entity Data Model*. Эта модель сопоставляет классы сущностей с реальными таблицами в БД.

*Entity Data Model* состоит из трех уровней: концептуального, уровень хранилища и уровень сопоставления (маппинга).

На концептуальном уровне происходит определение классов сущностей, используемых в приложении.

Уровень хранилища определяет таблицы, столбцы, отношения между таблицами и типы данных, с которыми сопоставляется используемая база данных.

Уровень сопоставления (маппинга) служит посредником между предыдущими двумя, определяя сопоставление между свойствами класса сущности и столбцами таблиц.

Таким образом, можно через классы, определенные в приложении, взаимодействовать с таблицами из базы данных.

*Entity Framework* предполагает три возможных способа взаимодействия с базой данных:

– *database first: Entity Framework* создает набор классов, которые отражают модель конкретной базы данных;

– *model first*: сначала разработчик создает модель базы данных, по которой затем *Entity Framework* создает реальную базу данных на сервере;

– *code firs*t: разработчик создает класс модели данных, которые будут храниться в бд, а затем *Entity Framework* по этой модели генерирует базу данных и ее таблицы.

### 2.7 Язык программирования C#

*C#* – язык программирования, сочетающий объектно-ориентированные и контекстно-ориентированные концепции. Разработан в 1998–2001 годах группой инженеров под руководством Андерсa Хейлсбергa в компании *Microsoft* как основной язык разработки приложений для платформы *Microsoft .NET*. Компилятор с *C*# входит в стандартную установку самой .*NET*, поэтому программы на нем можно создавать и компилировать даже без инструментальных средств вроде *Visual Studio.*

*C*# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и *Java*. Язык имеет строгую статическую типизацию, поддерживает полиморфизм, перегрузку операторов, указатели на функции-члены классов, атрибуты, события, свойства, исключения, комментарии в формате *XML*. Переняв многое от своих предшественников – языков C++, *Delphi, Modula* и *Smalltalk* – С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем: так, C# не поддерживает множественное наследование классов (в отличие от C++) или вывода типов (в отличие от *Haskell*). С помощью языка C# можно создавать обычные приложения *Windows*, *XML*-веб-службы, распределенные компоненты, приложения «клиент-сервер», приложения баз данных и т. д.

## 2.8 Выводы по разделу

В разделе проектирования были рассмотрены основные технологии, необходимые для разработки дипломного проекта. После изучения всех преимуществ этих технологий, для разработки программного средства были выбраны языки программирования С# и *Java* для написания мобильного приложения и *Web-API*. Данные языки способствуют качеству написания кода, отличаются широким спектром операционных систем, которые могут быть использованы для разработки, позволяют разрабатывать приложение с высоким уровнем быстродействия, а также отвечают всем современным стандартам качества разработки.

В качестве СУБД был выбран *MS SQL Server*, позволяющий в рамках проекта взаимодействовать с приложением. В рамках работы могут использоваться любые версии сервера, так как в *Entity Framework Core*, встроена работа со всеми вариантами данной базы данных.

# Проектирование и разработка программного средства

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*8*

*ДП 03.00.ПЗ*

*Разраб.*

*Славин Н.В.*

*Провер.*

*.*

*Сахонь Е.С.*

Консульт.

*Н. Контр.*

Нистюк О.А.

*Утверд.*

*Смелов В. В.*

3 Проектирование и разработка программного средства

*Лит.*

*Листов*

*16*

БГТУ 74217075, 2021

## Проектирование программного средства

Целью данного дипломного проекта является разработка мобильного приложения для выполнения заданий и получения вознаграждения под операционную систему *Android*. Исходя из поставленной цели были спроектированы задачи по созданию интуитивно понятного интерфейса и программной части приложения.

Мобильное приложение должно содержать функционал:

* авторизации и регистрации пользователей;
* просмотра и изменения данных аккаунта;
* создания, удаления и редактирования квестов;
* отправки запросов на выполнение квестов;
* выбора исполнителя для созданных квестов;
* просмотра рейтинга пользователей по числу выполненных квестов;
* просмотра информации о пользователях.

Разрабатываемое программное средство представляет собой клиент-серверное приложение.

Диаграмма вариантов использования представлена в приложении А.

Клиент представляет собой приложение, установленное на смартфон с операционной системой *Android* с версией 5.0 и выше. Клиент предназначен для взаимодействия пользователя с приложением и отправки запросов на сервер.

Сервер представляет собой *Web-API*, написанное на языке программирования C# с помощью платформы *ASP.NET* и *Entity Framework Core*. Сервер предназначен для обработки запросов, полученных от клиентов и отправки клиенту ответов.

Архитектура программного средства представлена в приложении Б.

## Файловая структура

### 3.2.1 Структура Web-API

Перед началом разработки была продумана файловая структура проекта, позволившая быстрее ориентироваться в коде, вносить изменения, дописывать новый функционал. Скриншот файловой структуры *Web-API* представлен на рисунке 3.1.

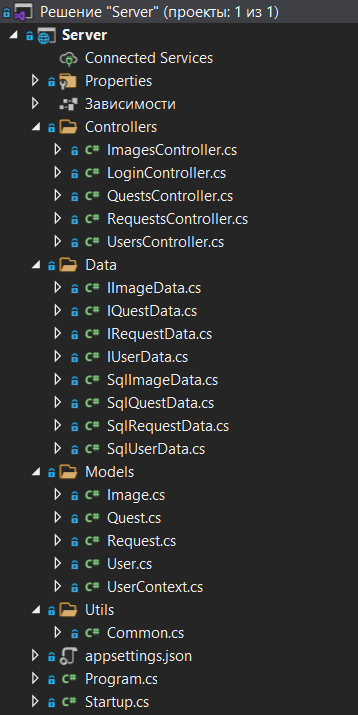


Рисунок 3.1 Файловая структура *Web-API*

Каталог *Models* содержит классы, которые описывают логику используемых данных. *Controllers* – каталог, который содержит файлы функционала приложения. *Data* содержит интерфейсы и классы для работы с БД. *Utils* содержит класс, который обеспечивает хеширование паролей.

### 3.2.2 Структура мобильного приложения

В данном подразделе была описана структура клиентского приложения и показаны некоторые особенности, обусловленные задачей приложения.

Основное решение проекта содержит большое количество папок и файлов, каждая из которых имеет свое назначение.

Скриншот файловой структуры мобильного приложения представлен на рисунке 3.2.

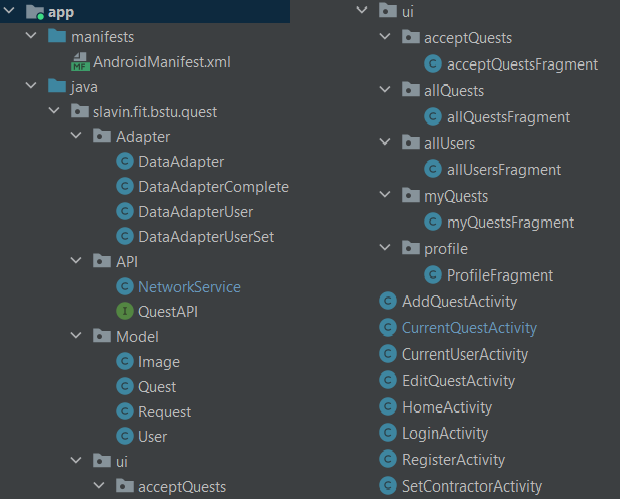


Рисунок 3.2 Файловая структура мобильного приложения

Здесь представлены файлы, описывающие поведение для активностей, но также присутствуют файлы для соединения и работы с сервером.

Каталог *Model* содержит классы, которые описывают логику используемых данных. Каталог *API* содержит в себе интерфейс и класс для соединения с *Web*-*API* и отправки запросов с помощью *Retrofit*. Каталог *ui* содержит в себе классы фрагментов для переходов по пунктам меню. Каталог *Adapter* содержит в себе классы адаптеров для работы элемента *RecyclerView*.

## Структура базы данных

В результате проектирования была создана база данных, которая удовлетворяет поставленным задачам. Данная реализация достаточно проста и универсальна, с ее помощью в дальнейшем можно изменять и дополнять структуру базы данных. Для реализации функций, которые были определены на стадии проектирования, были созданы следующие таблицы:

* *Users* (информация о пользователях);
* *Quests* (квесты пользователей);
* *Requests* (запросы пользователей на исполнение квестов);
* *Images* (фотографии пользователей);

Логическая схема базы данных со всеми связями показана на рисунке 3.3 и в приложении В.

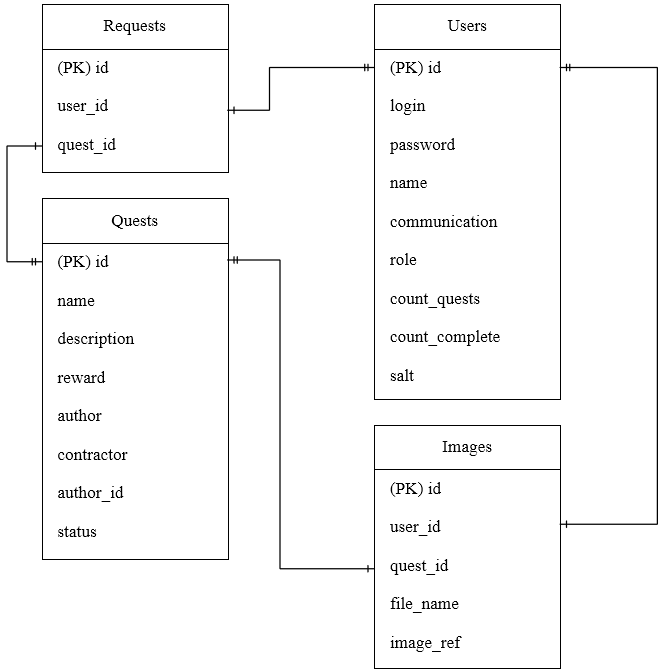


Рисунок 3.3 – Логическая схема базы данных

Для реализации функционала мобильного приложения было создано четыре таблицы баз данных. Далее приведена структура основных таблиц и их описание.

Таблица *Users* содержит *id* пользователя, логин, пароль, имя пользователя, способ связи с ним, роль, количество доступных квестов для создания, количество выполненных квестов и salt для хеширования.

Описание ее полей приведено в таблице 3.1.

Таблица 3.1 – Структура таблицы *Users*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| id | int | Идентификатор пользователя |
| login | nvarchar(50) | Логин пользователя |
| password | nvarchar(50) | Захешированный пароль |
| name | nvarchar(50) | Имя пользователя |
| communication | nvarchar(250) | Способ связи |
| role | nvarchar(50) | Роль |
| count\_quests | int | Доступные квесты для создания |
| count\_complete | int | Выполненные квесты |
| salt | nvarchar(250) | Хэш пароля |

Таблица *Quests* содержит идентификатор квеста, название квеста, описание квеста, вознаграждение за выполнение квеста, автора квеста, исполнителя квеста, идентификатор автора и статус квеста. Описание ее полей приведено в таблице 3.2.

Таблица 3.2 – Структура таблицы *Quests*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| id | int | Идентификатор квеста |
| name | nvarchar(50) | Название квеста |
| description | nvarchar(500) | Описание квеста |
| reward | nvarchar(50) | Вознаграждение |
| author | nvarchar(50) | Логин автора |
| contractor | nvarchar(50) | Логин исполнителя |
| author\_id | int | Идентификатор автора квеста |
| status | nvarchar(50) | Статус квеста |

Таблица *Requests* содержит идентификатор запроса, идентификатор квеста, идентификатор пользователя. Структура представлена в таблице 3.3.

Таблица 3.3 – Структура таблицы *Requests*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| id | int | Идентификатор запроса |
| user\_id | int | Идентификатор квеста |
| quest\_id | int | Идентификатор пользователя |

Таблица *Images* содержит идентификатор фотографии, идентификатор пользователя, идентификатор квеста, название файла и ссылку на изображение. Структура представлена в таблице 3.4.

Таблица 3.4 – Структура таблицы *Images*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| id | int | Идентификатор фотографии |
| user\_id | int | Идентификатор пользователя |
| quest\_id | int | Идентификатор квеста |
| file\_name | nvarchar(50) | Название файла |
| image\_ref | nvarchar(500) | Ссылка на изображение |

Выше были представлены таблицы, которые были созданы для функционирования всей системы.

### 3.4 Разработка Web-API

Основной функционал сервера содержится в контроллерах. Это связано со спецификой разрабатываемой программы. В данном подразделе были описаны используемые контроллеры.

Всего было написано 5 контроллеров, используемых для работы программного средства:

* *QuestsController;*
* *UsersController;*
* *ImagesController;*
* *RequestsController;*
* *LoginController.*

Один из часто используемых контроллеров – это *QuestsController*, который обеспечивает всю работу с квестами.

Пример программного кода контроллера представлен на листинге 3.1.

[ApiController]

public class QuestsController : ControllerBase

{

private IQuestData \_questData;

public QuestsController(IQuestData questData)

{

\_questData = questData;

}

[HttpGet]

[Route("api/[controller]")]

public string GetQuests()

{

return JsonConvert.SerializeObject(\_questData.GetQuests());

}

}

Листинг 3.1 – Программная реализация контроллера *QuestsController*

Основной функционал контроллера:

* получение списка квестов;
* добавление нового квеста;
* получение отдельного квеста;
* удаление отдельного квеста;
* изменение отдельного квеста.

Получение списка всех квестов и получение отдельного квеста происходит при *GET* запросе. Для добавления нового квеста используется *POST* запрос. Для удаления используется *DELETE* запроса. Для изменения уже существующего квеста применяется *PATCH* запрос.

Контроллер взаимодействует с базой данных с помощью запросов *LINQ to Entities*. Все запросы прописаны в классе *SqlQuestData*, который реализует интерфейс *IQuestData*.

Следующий по очереди идёт контроллер *UsersController*. Он обеспечивает работу с таблицей *Users*. Пример программного кода контроллера представлен на листинге 3.2.

[ApiController]

public class UsersController : ControllerBase

{

private IUserData \_userData;

public UsersController(IUserData userData)

{

\_userData = userData;

}

[HttpGet]

[Route("api/[controller]")]

public string GetUsers()

{

return JsonConvert.SerializeObject(\_userData.GetUsers());

}

}

Листинг 3.2 – Программная реализация контроллера *UsersController*

Основной функционал контроллера:

* получение списка пользователей;
* добавление нового пользователя;
* получение отдельного пользователя;
* удаление отдельного пользователя;
* изменение отдельного пользователя.

Контроллер взаимодействует с базой данных с помощью запросов *LINQ to Entities*. Все запросы прописаны в классе *SqlUserData*, который реализует интерфейс *IUserData*.

Контроллер *LoginController* предназначен для авторизации пользователя. В нём идёт сравнение полученных данных с имеющимися в базе данных. Если пользователь с полученными данными существует, то клиенту возвращаются все данные этого пользователя. Пример программного кода контроллера представлен на листинге 3.3

[HttpPost]

[Route("api/[controller]")]

public string Login(User user)

{

List<User> list = \_userData.GetUsers();

foreach (User user1 in list)

{

var client\_post\_hash\_password = Convert.ToBase64String(

Common.SaltHashPassword(

Encoding.ASCII.GetBytes(user.password),

Convert.FromBase64String(user1.salt)));

if (user.login == user1.login && client\_post\_hash\_password.Equals(user1.password))

return JsonConvert.SerializeObject(user1);

}

return JsonConvert.SerializeObject("Пользователь не найден");

}

Листинг 3.3 – Программная реализация контроллера *LoginController*

*LoginController* выполняет хеширование полученного от клиента пароля на основе *salt* хранящегося в базе данных. Если полученный пароль будет совпадать с имеющимся в базе данных, то клиенту вернётся ответ с объектом *User* для дальнейшего взаимодействия с приложением. В другом случае клиент получит сообщение о том, что пользователь не был найден.

Определение всех действий с базой данных происходит в интерфейсах. Всего было написано 4 интерфейса, используемых для работы программного средства:

* *IQuestData;*
* *IUserData;*
* *IImageData;*
* *IRequestData.*

В интерфейсе *IQuestData* определено 5 методов взаимодействия с данными. Эти методы представляют собой *CRUD*-операции, т.е. добавление, удаление, изменение и выборка данных. Программный код интерфейса представлен на листинге 3.4.

public interface IQuestData

{

List<Quest> GetQuests();

Quest GetQuest(int id);

string AddQuest(Quest quest);

void DeleteQuest(Quest quest);

Quest EditQuest(Quest quest);

}

Листинг 3.4 – Программный код интерфейса *IQuestData*

Метод *GetQuests* предназначен для получения списка всех квестов. *GetQuest* используется для получения данных об одном квесте по его идентификатору. *AddQuest* добавляет новый квест в базу данных. *DeleteQuest* удаляет квест из базы данных по идентификатору. Метод *EditQuest* предназначен для изменения данных отдельного квеста.

Все интерфейсы были реализованы, чтобы иметь возможность работать с данными в нашей БД. Для каждого интерфейса был создан файл, реализующий его. Всего было написано 4 класса, реализующих интефейсы:

* *SqlQuestData;*
* *SqlUserData;*
* *SqlImageData;*
* *SqlRequestData.*

*SqlQuestData* – класс, реализующий интерфейс *IQuestData,* в котором идёт реализация методов для работы с таблицей *Quests*.

Пример программного кода класса *SqlQuestData* представлен на листинге 3.5.

public class SqlQuestData : IQuestData

{

private UserContext \_userContext;

public SqlQuestData(UserContext userContext)

{

\_userContext = userContext;

}

public List<Quest> GetQuests()

{

return \_userContext.Quests.ToList();

}

}

Листинг 3.5 – Программный код класса *SqlQuestData*

## 3.5 Разработка мобильного приложения

Основная логика программного средства реализована на стороне клиента, т.е. в мобильном приложении. Для взаимодействия клиентского приложения с сервером использовалась библиотека *Retrofit*. С помощью этой библиотеки можно отправлять запросы на сервер в асинхронном режиме. Для её реализации было создано несколько файлов. Первый файл – это интерфейс *QuestAPI*, который содержит в себе определения всех запросов, которые будут отправляться на сервер.

Пример программного кода интерфейса представлен на листинге 3.6.

public interface QuestAPI {  
 @GET("/api/quests/{id}")  
 @Headers("Content-Type: application/json")  
 public Call<Quest> GetQuest(@Path("id") int id);  
  
 @GET("/api/quests")  
 @Headers("Content-Type: application/json")  
 public Call<List<Quest>> GetQuests();  
  
 @DELETE("/api/quests/{id}")  
 @Headers("Content-Type: application/json")  
 public Call<String> DeleteQuest(@Path("id") int id);  
  
 @POST("/api/quests")  
 @Headers("Content-Type: application/json")  
 public Call<String> AddQuest(@Body Quest quest);  
  
 @PATCH("/api/quests/{id}")  
 @Headers("Content-Type: application/json")  
 public Call<String> EditQuest(@Path("id") int id, @Body Quest quest);

}

Листинг 3.6 – Программный код интерфейса *QuestAPI*

Каждый метод содержит в себе тип данных, который будет получен в качестве ответа от сервера, название, а также параметры, которые передает этот запрос. Также с помощью аннотации указывается тип запроса и его заголовки.

Следующий файл, который был создан для реализации библиотеки *Retrofit* это *NetworkService*. С помощью него устанавливается соединение с сервером, отправляется нужный запрос и возвращается ответ.

Пример программного кода интерфейса представлен на листинге 3.7.

public class NetworkService {  
 private static NetworkService *mInstance*;  
 private static final String *BASE\_URL* = "http://10.0.2.2:32024";  
 private Retrofit mRetrofit;  
  
 private NetworkService() {  
 HttpLoggingInterceptor interceptor = new HttpLoggingInterceptor();  
 interceptor.setLevel(HttpLoggingInterceptor.Level.*BODY*);  
  
 OkHttpClient.Builder client = new OkHttpClient.Builder()  
 .addInterceptor(interceptor);  
  
 mRetrofit = new Retrofit.Builder()  
 .baseUrl(*BASE\_URL*)  
 .addConverterFactory(GsonConverterFactory.*create*())  
 .client(client.build())  
 .build();  
 }  
}

Листинг 3.7 – Программная реализация класса *NetworkService*

Для отправки запросов на сервер часто вместе с запросом посылаются данные в виде объектов, с которыми нужно произвести какие-то действия. Модели данных размещены в папке *Model* и для каждой модели выделен отдельный файл.

Пример программного кода модели *Quest* представлен на листинге 3.8.

public class Quest {  
 @SerializedName("id")  
 private int id;  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
}

Листинг 3.8 – Программный код модели *Quest*

В классе каждой модели присутствуют поля, описывающие данную модель и методы *get* и *set* для возврата значения какого-либо поля определённой модели либо для установки нового значения.

Оставшуюся часть файлов в мобильном приложении составляют файлы определения поведения активностей, их разметок, адаптеры для работы со списками, а также файлы дизайна приложения. Файл *AddQuestActivity* – это активность, которая предназначена для создания нового квеста. В ней происходит создание объекта квеста и отправка *POST* запроса на сервер.

Пример программной реализации класса *AddQuestActivity* представлен на листинге 3.9.

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_addquest*);  
 ActionBar actionBar =getSupportActionBar();  
 actionBar.setHomeButtonEnabled(true);  
 actionBar.setDisplayHomeAsUpEnabled(true);  
  
 username = bundle.getString("username");  
 add.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 add();  
 }  
 });  
}

Листинг 3.9 – Программная реализация активности *AddQuestActivity*

Также в данном классе есть другие методы, которые обеспечивают обработку событий кликов на странице. Метод *add* предназначен для создания объектов квеста, изображения и отправки их на сервер. Пример программной реализации метода представлен на листинге 3.10.

public void add() {  
 NetworkService.getInstance().getQuestApi().AddQuest(quest)  
 .enqueue(new Callback<String>() {  
 @Override  
 public void onResponse(@NonNull Call<String> call, @NonNull Response<String> response) {  
 if (response.body().equals("Квест успешно добавлен")) {  
 NetworkService.getInstance().getQuestApi().GetQuests()  
 .enqueue(new Callback<List<Quest>>() {  
 @Override  
 public void onResponse(@NonNull Call<List<Quest>> call, @NonNull Response<List<Quest>> response) {  
 uploadImage();  
 }  
 back();  
 }  
 }  
}

Листинг 3.10 – Программная реализация метода *add*

В данном методе отправляются последовательно 2 запроса на сервер, сначала создание объекта квеста, а затем создание объекта изображения. Также здесь присутствует метод *uploadImage* предназначенный для загрузки изображения в *FireBase Storage,* получения ссылки на изображение и отправки этой ссылки на сервер вместе с объектом *Image.*

*FireBase* – это облачная база данных, которая позволяет пользователям хранить и получать сохраненную информацию, а также имеет удобные средства и методы взаимодействия с ней. Пример программной реализации метода для загрузки изображения в *FireBase* представлен на листинге 3.11.

private void uploadImage() {  
 ref.putFile(filePath).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {  
 public void onSuccess(UploadTask.TaskSnapshot taskSnapshot){   
 imageuri = uri.getResult().toString();  
 NetworkService.getInstance().getQuestApi().AddImage(image)  
.enqueue(new Callback<String>() {  
 public void onResponse(@NonNull Call<String> call, @NonNull Response<String> response)  
}

Листинг 3.11 – Программная реализация метода *UploadImage*

Файл *AllQuestFragment* – это фрагмент, который предназначен для вывода списка всех квестов, созданных пользователями, внутри *RecyclerView.* Для этого производится *GET* запрос на сервер и полученные данные помещаются в адаптер, который затем устанавливается в список.

Пример программной реализации класса *AllQuestFragment* представлен на листинге 3.12.

public class allQuestsFragment extends Fragment {  
  
 public View onCreateView(@NonNull LayoutInflater inflater,  
 ViewGroup container, Bundle savedInstanceState) {  
 root = inflater.inflate(R.layout.fragment\_allquests, container, false);  
  
 Bundle id = getActivity().getIntent().getExtras();  
 username = id.getString("username");  
 userId = id.getInt("id");  
  
 setHasOptionsMenu(true);  
 display();  
  
 return root;  
 }

Листинг 3.12 – Программная реализация класса *AllQuestsFragment*

В данном классе есть несколько функций, которые отвечают за всю логику работы фрагмента. *display –* функция, которая объединяет в себе все остальные функции, предназначена для получения списка квестов от сервера посредством отправки *GET* запроса. Функция *initViews* предназначена для определения элемента *RecyclerView* и установки для него адаптера. Программная реализация функции представлена на листинге 3.13.

private void initViews(){  
 recyclerView = (RecyclerView) root.findViewById (R.id.card\_recycler\_view);  
 recyclerView.setHasFixedSize(true);  
 RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(root.getContext());  
 recyclerView.setLayoutManager(layoutManager);  
 recyclerView.setNestedScrollingEnabled(false);  
 adapter = new DataAdapterComplete(root.getContext(), listQuests, listImages);  
 recyclerView.setAdapter(adapter);  
  
}

Листинг 3.13 – Программная реализация функции *initViews*

Также на данном фрагменте реализована функция *filter* для фильтрации списка по введённому тексту в поиске. Пример программной реализации функции представлен на листинге 3.14.

void filter(String text){  
 List<Quest> temp = new ArrayList();  
 for(Quest quest: listQuests){  
 if(quest.getName().toLowerCase().contains(text.toLowerCase()) || quest.getDescription().toLowerCase().contains(text.toLowerCase())){  
 temp.add(quest);  
 }  
 }  
 //update recyclerview  
 adapter.updateList(temp);  
}

Листинг 3.14 – Программная реализация функции *filter*

Как уже говорилось выше, адаптеры нужны для отображения списка в *RecyclerView.* Класс *DataAdapter* используется для отображения элементов в списке, а также для придания дополнительной логики этим элементам, например, обработки события на клик. Пример программной реализации класса представлен на листинге 3.15.

public class DataAdapter extends RecyclerView.Adapter<DataAdapter.ViewHolder> {  
  
 public DataAdapter(Context context, List<Quest> questsList, List<Image> imagesList) {  
 this.context = context;  
 this.questsList = questsList;  
 this.imagesList = imagesList;  
 }  
  
 public class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener, View.OnCreateContextMenuListener{  
 public ViewHolder(View view) {  
 super(view);  
 view.setOnClickListener(this);  
 view.setOnCreateContextMenuListener(this);  
 }  
 }  
}

Листинг 3.15 – Программная реализация класса *DataAdapter*

Также в приложении присутствует свой дизайн, который был сделан с помощью файлов разметки. Дизайн всех кнопок в приложении был прописан внутри файла *button\_style.xml.* Пример кода дизайна представлен на листинге 3.16.

<?xml version="1.0" encoding="utf-8"?>  
<inset xmlns:android="http://schemas.android.com/apk/res/android"  
 android:insetBottom="6dp"  
 android:insetLeft="4dp"  
 android:insetRight="4dp"  
 android:insetTop="6dp">  
 <ripple android:color="?attr/colorControlHighlight">  
 <item>  
 <shape android:shape="rectangle"  
 android:tint="#46a138">  
 <corners android:radius="10dp"/>  
 <padding android:bottom="4dp"/>  
 </shape>  
 </item>  
 </ripple>  
</inset>

Листинг 3.16 – Код дизайна кнопок в приложении

## 3.6 Выводы по разделу

В результате выполнения дипломного проекта было разработано мобильное приложение для выполнения заданий и получения вознаграждения «Квест». Все поставленные задачи перед проектом были выполнены.

Полученное мобильное приложение не требует специфического программного обеспечения, для его запуска необходимо наличие *Android* 5.0 и выше, а также доступ в сеть *Internet* любым доступным и удобным для пользователя способом. Интерфейс является интуитивно понятным и простым. В программном средстве использованы устойчивые формы записи кода (использование обработчиков ошибок) благодаря чему повышается отказоустойчивость приложения.

Программное средство справляется со всеми поставленными задачами и целями, а именно:

* спроектирована и разработана структура базы данных;
* разработан пользовательский интерфейс приложения;
* реализована авторизация и регистрация пользователей;
* реализован поиск пользователей и квестов;
* реализовано создание, редактирование и удаление квестов;
* разработаны контроллеры для обработки запросов клиентского приложения.

Спроектированная архитектура проекта позволит добавлять и усовершенствовать функциональные возможности и поддерживать работу системы в будущем.

# Тестирование программного средства

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*8*

*ДП 04.00.ПЗ*

*Разраб.*

*Славин Н.В.*

*Провер.*

*.*

*Сахонь Е.С.*

Консульт.

*Н. Контр.*

Нистюк О.А.

*Утверд.*

*Смелов В. В.*

4 Тестирование программного средства

*Лит.*

*Листов*

*5*

БГТУ 74217075, 2021

Тестирование является неотъемлемой частью процесса разработки программного продукта, так как данный этап позволяет найти неисправности в работе приложения и исправить их.

В данной главе будут описаны некоторые тесты, проведенные для выявления и устранения неисправностей разработанного программного продукта.

Для начала представим названия тест-кейсов, которые и будем рассматривать ниже:

* авторизация пользователя;
* регистрация пользователя;
* создание квеста;
* изменение квеста;

## Авторизация пользователя

Для авторизации пользователю необходимо ввести свой логин и пароль, после чего нажать на кнопку «Войти». Поля не могут быть пустыми, иначе появится предупреждение.

После заполнения всех полей происходит запрос на сервер, на котором идёт поиск пользователя с введёнными данными. Если пользователь был найден, то приходит ответ с данными пользователя и происходит вход в приложение. Если пользователь не был найден, то отобразится соответствующая ошибка.

Скриншот тестирования авторизации представлен на рисунке 4.1.

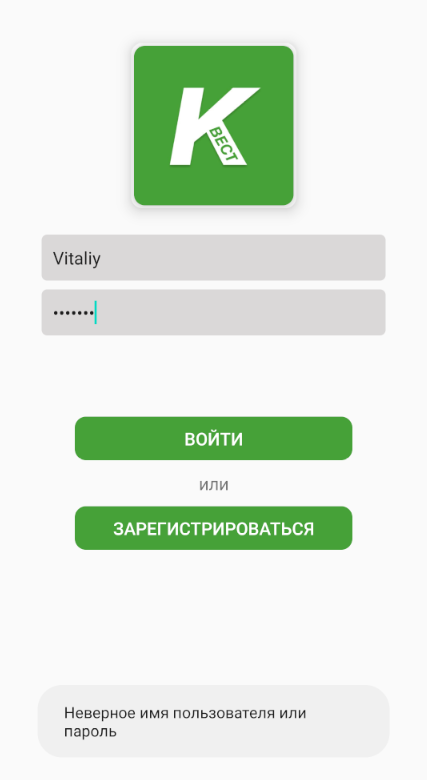


Рисунок 4.1 – Тестирование авторизации пользователя

## Регистрация пользователя

Для регистрации пользователю необходимо ввести логин, пароль, своё имя и способ связи с пользователем. Все поля не должны быть пустыми и длина логина и пароля должна составлять не менее 6 символов, иначе появится предупреждение.

Скриншот тестирования регистрации представлен на рисунке 4.2.

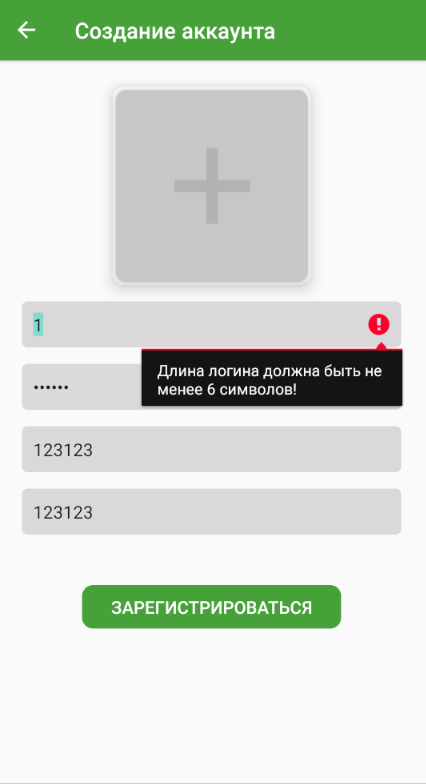


Рисунок 4.2 – Тестирование регистрации пользователя

После ввода всех данных требуется нажать на кнопку «Зарегистрироваться», тогда отправится запрос на сервер и если пользователь с таким логином не существует в базе данных, то аккаунт успешно будет создан.

## Создание квеста

Для создание квеста требуется ввести его название, описание и вознаграждение за его выполнение. Все поля должны быть заполнены и иметь ограничение на максимальное количество символов.

Скриншот тестирования создания квеста представлен на рисунке 4.3.

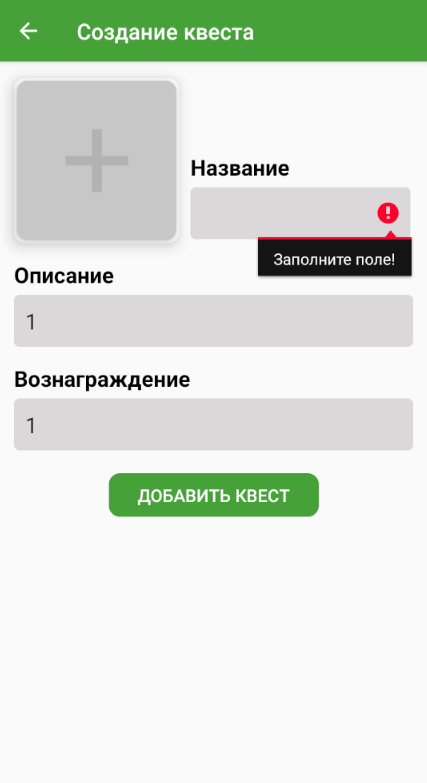


Рисунок 4.3 – Тестирование создания квеста

После успешного создания квеста, его можно будет найти в списке всех квестов или в разделе «Мои квесты».

## Изменение квеста

Для изменения квеста пользователю нужно изменить те данные, которые требуется и нажать на кнопку «Изменить квест». Тут действуют такие же ограничения, как и при создании квеста, а именно поля должны быть заполнены и иметь ограничение на максимальное количество символов.

Скриншот тестирования изменения квеста представлен на рисунке 4.4.

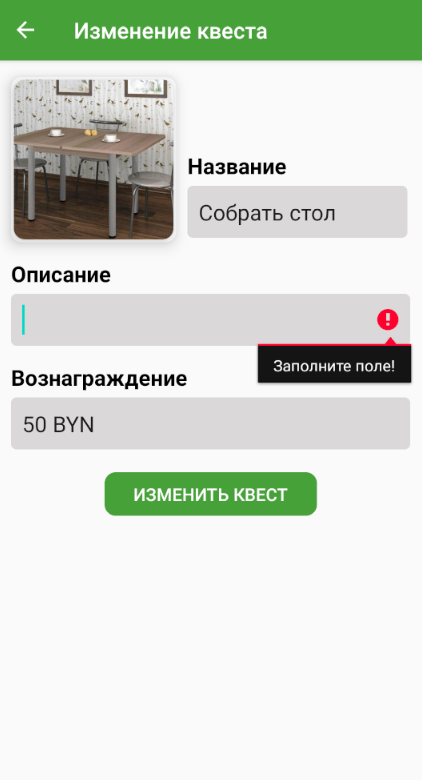


Рисунок 4.4 – Тестирование изменения квеста

## 4.5 Выводы по разделу

В данной главе был выполнен контрольный пример для обнаружения ошибок в поведении системы и выполнен ряд тестов для проверки ее стабильности. В результате проделанной работы установлено, что система стабильно работает, все тесты были выполнены успешно, а именно:

* авторизация пользователя;
* регистрация пользователя;
* создание квеста;
* изменение квеста;

В ходе тестирования было выявлено, что интерфейс обладает достаточной функциональностью и удобством, а дизайн приложения имеет благоприятный для пользования вид.

# Руководство пользователя

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*8*

*ДП 05.00.ПЗ*

*Разраб.*

*Славин Н.В.*

*Провер.*

*.*

*Сахонь Е.С.*

Консульт.

*Н. Контр.*

Нистюк О.А.

*Утверд.*

*Смелов В. В.*

5 Тестирование программного средства

*Лит.*

*Листов*

*9*

БГТУ 74217075, 2021

Для работы приложения необходим смартфон с операционной системой *Android* 5.0 и выше, а также наличие соединения с интернетом.

## Авторизация пользователя

При входе в приложение пользователь попадает на страницу авторизации. Для проверки авторизации пользователя используются логин и пароль. Страница авторизации представлена на рисунке 5.1.

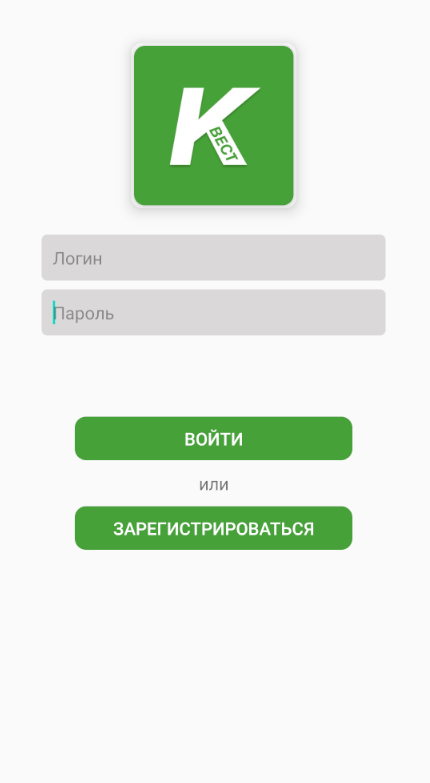


Рисунок 5.1 – Страница авторизации

## 5.2 Регистрация пользователя

Если у пользователя ещё не существует аккаунт, то требуется нажать на кнопку «Зарегистрироваться». После клика запустится новая активность для регистрации. Страница регистрации представлена на рисунке 5.2.

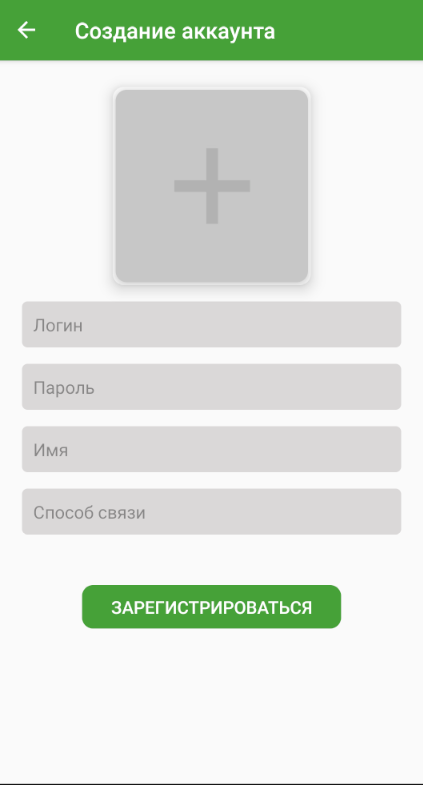


Рисунок 5.2 – Страница регистрации

На странице регистрации пользователю необходимо ввести логин, пароль, имя и способ связи для создания аккаунта. После успешной регистрации активность закроется и вернётся страница авторизации.

## 5.3 Раздел «Все квесты»

После успешной авторизации откроется раздел «Все квесты», в котором выводится список всех созданных квестов. Скриншот раздела представлен на рисунке 5.3.

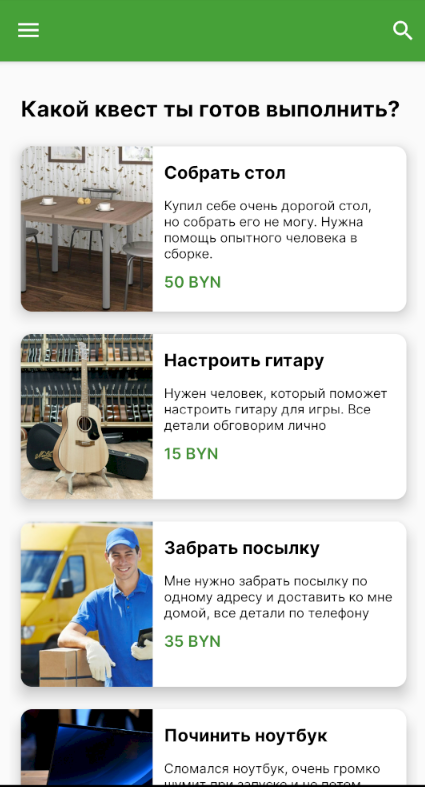


Рисунок 5.3 – Раздел «Все квесты»

На каждый квест можно нажать после чего откроется новая активность, в которой будет представлена подробная информация о квесте. Скриншот подробной информации о квесте представлен на рисунке 5.4.



Рисунок 5.4 – Подробная информация о квесте

На данной странице можно прочитать всю нужную информацию о квесте. А также можно отправить запрос на выполнение выбранного квеста, если вы не являетесь его автором. При условии, что вы автор квеста, можно выбрать исполнителя квеста.

## 5.4 Меню приложения

По нажатию на кнопку в верхнем левом углу либо свайпу вправо откроется меню, в котором представлены все основные разделы приложения. Скриншот меню представлен на рисунке 5.5.

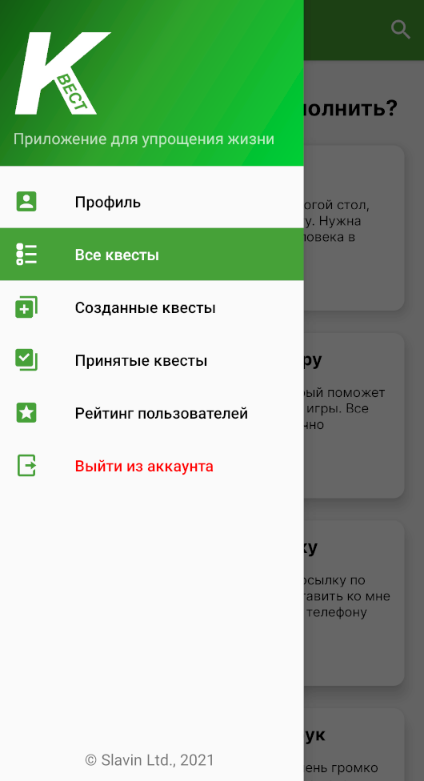


Рисунок 5.5 – Меню приложения

По нажатию на любой из пунктов меню происходит запуск соответствующего фрагмента активности либо выход из аккаунта.

## 5.5 Раздел «Профиль»

Раздел «Профиль» открывает страницу с информацией об аккаунте авторизованного пользователя. Скриншот профиля представлен на рисунке 5.6.

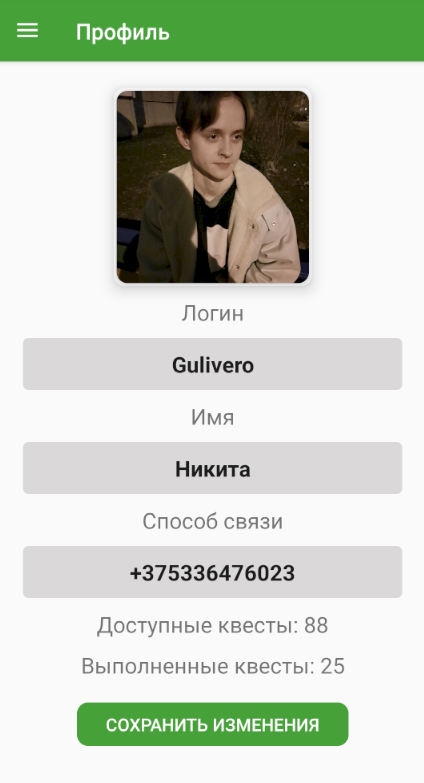


Рисунок 5.6 – Профиль авторизованного пользователя

В данном разделе представлена информация о логине, имени, способе связи и количестве доступных квестов для создания. По желанию, можно изменить некоторые данные пользователя, и чтобы эти изменения сохранились требуется нажать на кнопку «Сохранить изменения».

## 5.6 Создание нового квеста

Раздел «Мои квесты» открывает страницу, на которой выводится список со всеми созданными квестами авторизованного пользователя. На каждый из квестов также можно нажать для открытия подробной информации. Также здесь имеется меню с кнопкой «Новый квест», по нажатию на которую открывается страница создания квеста. Скриншот страницы создания нового квеста представлен на рисунке 5.7.

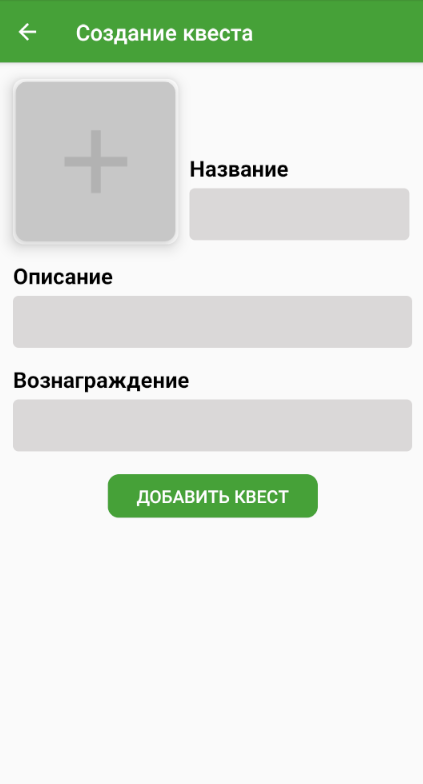


Рисунок 5.7 – Страница создания нового квеста

На данной странице требуется ввести данные о квесте и нажать на кнопку «Добавить квест». Квест будет успешно создан, если у авторизованного пользователя не достигнут лимит создания квестов.

## 5.7 Раздел «Рейтинг пользователей»

Раздел «Рейтинг пользователей» представляет собой страницу со список всех пользователей, зарегистрированных в приложении. В верхнем меню есть строка поиска для того, чтобы найти пользователя с требуемым логином. Скриншот раздела представлен на рисунке 5.8.

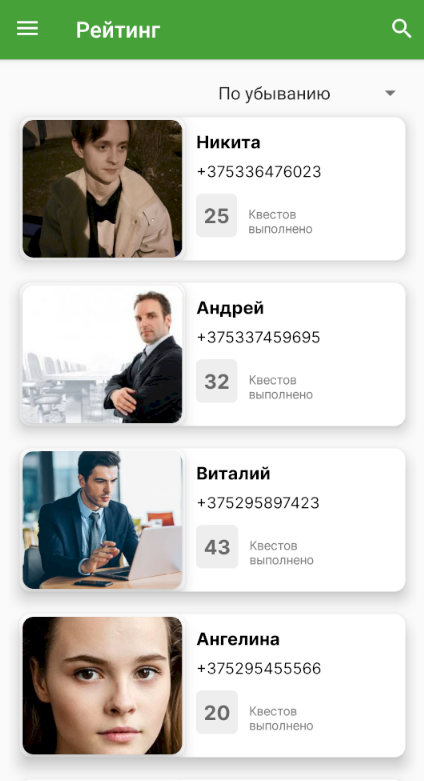


Рисунок 5.8 – Раздел «Рейтинг пользователей»

Также по нажатию на любого пользователя откроется страница с его профилем, где можно просмотреть подробную информацию. Скриншот страницы профиля представлен на рисунке 5.9.

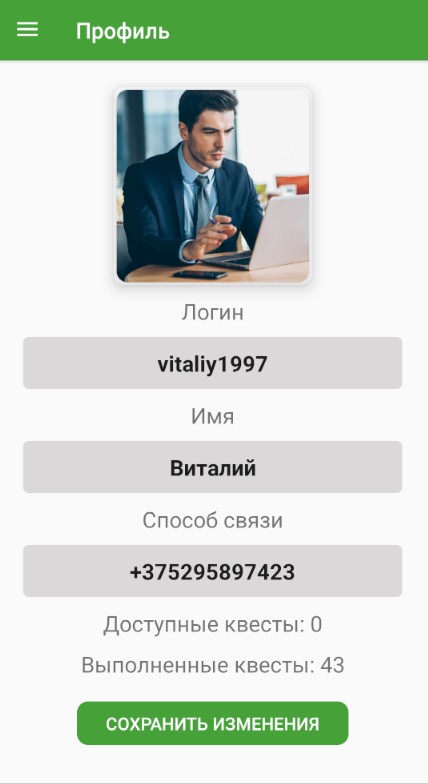


Рисунок 5.9 – Профиль выбранного пользователя

## 5.8 Выводы по разделу

Цель данного раздела заключалась в создании руководства пользователя, чтобы предоставить пользователю возможность самостоятельно решать свои прикладные задачи с помощью данного мобильного приложения. Данное руководство служит кратким введением в предметную область и ознакомлением со всеми возможностями мобильного приложения, а также содержит краткое описание конкретных процедур решения задач.

Данный раздел необходим для ознакомления пользователя с внешним видом, интерфейсом и функционалом мобильного приложения.

Таким образом руководство пользователя полезно при возникновении вопросов касательно использования приложения и навигации между его страниц.

# Анализ информационной безопасности приложения

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*8*

*ДП 06.00.ПЗ*

*Разраб.*

*Славин Н.В.*

*Провер.*

*.*

*Сахонь Е.С.*

Консульт.

*Н. Контр.*

Нистюк О.А.

*Утверд.*

*Смелов В. В.*

6 Анализ информационной безопасности приложения

*Лит.*

*Листов*

*5*

БГТУ 74217075, 2021

## 6.1 Определение информационной безопасности

Информационная безопасность – это сохранение и защита информации, а также ее важнейших элементов, в том числе системы и оборудование, предназначенные для использования, сбережения и передачи этой информации. Другими словами, это набор технологий, стандартов и методов управления, которые необходимы для защиты информационной безопасности.

Цель обеспечения информационной безопасности – защитить информационные данные и поддерживающую инфраструктуру от случайного или преднамеренного вмешательства, что может стать причиной потери данных или их несанкционированного изменения. Информационная безопасность помогает обеспечить непрерывность бизнеса.

Для успешного внедрения систем информационной безопасности на предприятии необходимо придерживаться трех главных принципов:

1. Конфиденциальность. Это значит ввести в действие контроль, чтобы гарантировать достаточный уровень безопасности с данными предприятия, активами и информацией на разных этапах деловых операций для предотвращения нежелательного или несанкционированного раскрытия. Конфиденциальность должна поддерживаться при сохранении информации, а также при транзите через рядовые организации независимо от ее формата;
2. Целостность. Целостность имеет дело с элементами управления, которые связаны с обеспечением того, чтобы корпоративная информация была внутренне и внешне последовательной. Целостность также гарантирует предотвращение искажения информации;
3. Доступность. Доступность обеспечивает надежный и эффективный доступ к информации уполномоченных лиц. Сетевая среда должна вести себя предсказуемым образом с целью получить доступ к информации и данным, когда это необходимо. Восстановление системы по причине сбоя является важным фактором, когда речь идет о доступности информации, и такое восстановление также должно быть обеспечено таким образом, чтобы это не влияло на работу отрицательно.

## 6.2 Хеширование паролей

### 6.2.1 Определение хеширования

Криптографические хеш-функции — незаменимый и повсеместно распространенный инструмент, используемый для выполнения целого ряда задач, включая аутентификацию, проверку целостности данных, защиту файлов и даже обнаружение зловредного ПО. Существует масса алгоритмов хеширования, отличающихся криптостойкостью, сложностью, разрядностью и другими свойствами.

Криптографическая хеш-функция, чаще называемая просто хешем, — это математический алгоритм, преобразовывающий произвольный массив данных в состоящую из букв и цифр строку фиксированной длины. Причем при условии использования того же типа хеша длина эта будет оставаться неизменной, вне зависимости от объема вводных данных.

### 6.2.2 Реализация хеширования в программном средстве

Хеширование паролей было реализовано в отдельном файле *Common.cs* на стороне сервера*.* Для этого был использован алгоритм *SHA-256*. Программная реализация хеширования представлена в листинге 6.1.

public static byte[] GetRandomSalt(int length)

{

var random = new RNGCryptoServiceProvider();

byte[] salt = new byte[length];

random.GetNonZeroBytes(salt);

return salt;

}

public static byte[] SaltHashPassword(byte[] password, byte[] salt)

{

HashAlgorithm algorithm = new SHA256Managed();

byte[] plainTextWithSaltBytes = new byte[password.Length + salt.Length];

for (int i = 0; i < password.Length; i++)

{

plainTextWithSaltBytes[i] = password[i];

}

for (int i = 0; i < salt.Length; i++)

{

plainTextWithSaltBytes[password.Length + i] = salt[i];

}

return algorithm.ComputeHash(plainTextWithSaltBytes);

}

Листинг 6.1 – Программная реализация хеширования паролей

Функция *GetRandomSalt* возвращает случайный набор байт заданной длины. Затем функция *SaltHashPassword* совершает хеширование на основе полученного набора байт, введённого пароля и выбранного алгоритма хеширования. После хеширования функция возвращает также набор байт, который является паролем.

В таблице *Users* базы данных специально было выделено 2 столбца для хранения *salt* (набора байт для хеширования)и хешированного пароля.

Авторизация пользователей происходит за счёт ввода логина и пароля в приложении, после чего сервер хеширует введёный пароль на основе *salt,* хранящегося в таблице базы данных и затем сравнивает полученный пароль с имеющимся в таблице.

## 6.3 Протокол HTTPS

*HTTPS* – это расширение протокола *HTTP* для поддержки шифрования в целях повышения безопасности. Данные в протоколе *HTTPS* передаются поверх криптографических протоколов *TLS* или *SSL*. В отличии от *HTTP* с *TCP*-портом 80, для *HTTPS* по умолчанию используется *TCP*-порт 443.

*HTTPS* не является отдельным протоколом. Это обычный *HTTP*, работающий через шифрованные транспортные механизмы *SSL* и *TLS*. Он обеспечивает защиту от атак, основанных на прослушивании сетевого соединения — от снифферских атак и атак типа *man*-*in-the*-*middle*, при условии, что будут использоваться шифрующие средства и сертификат сервера проверен и ему доверяют.

По умолчанию *HTTPS* *URL* использует 443 *TCP*-порт. Чтобы подготовить веб-сервер для обработки *HTTPS*-соединений, администратор должен получить и установить в систему сертификат открытого и закрытого ключа для этого веб-сервера. В *TLS* используется как асимметричная схема шифрования (для выработки общего секретного ключа), так и симметричная (для обмена данными, зашифрованными общим ключом). Сертификат открытого ключа подтверждает принадлежность данного открытого ключа владельцу сайта. Сертификат открытого ключа и сам открытый ключ посылаются клиенту при установлении соединения; закрытый ключ используется для расшифровки сообщений от клиента.

Существует возможность создать такой сертификат, не обращаясь в центр сертификации. Подписываются такие сертификаты этим же сертификатом и называются самоподписанными.

Для обеспечения конфиденциальности данных программного средства, *Web-API* было поставлено на хостинг с установленным сертификатом *SSL.*

## 6.4. Обфускация кода

## 6.4.1 Определение обфускации

Обфускация – приведение исходного текста или исполняемого кода программы к виду, сохраняющему её функциональность, но затрудняющему анализ, понимание алгоритмов работы и модификацию при декомпиляции.

Так же, алгоритмы обфускации активно используются не только для затруднения анализа кода, но и для уменьшения размера программного кода, что, в свою очередь, активно используется при разработке различных веб-сервисов и баз данных.

Цели обфускации:

* затруднение декомпиляции/отладки и изучения программ с целью обнаружения функциональности;
* затруднение декомпиляции проприетарных программ с целью предотвращения обратной разработки или обхода DRM и систем проверки лицензий;
* затруднение взлома программного обеспечения;
* оптимизация программы с целью уменьшения размера работающего кода и ускорения работы.

## 6.4.2 Реализация обфускации в программном средстве

Обфускация была реализована с помощью ProGuard. ProGuard – это инструмент командной строки с открытым исходным кодом, который сжимает, оптимизирует и скрывает Java-код. Также он может оптимизировать байт-код, а также обнаруживать и удалять неиспользуемые конструкции.

Для включения обфускации внутри файла *build.gradle* был написан код.Пример кода представлен на листинге 6.2.

buildTypes {  
 release {  
 shrinkResources true  
 minifyEnabled true  
 proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'  
 }  
}

Листинг 6.2 – Код включения обфускации

Ключевое слово *release* здесь указывает, что данные правила будут применяться к ­­*release*-сборке приложения. *shrinkResources* оптимизирует приложение, удаляя неиспользуемые и ненужные части кода. *minifyEnabled* применяется для обфускации. *getDefaultProguardFile* получает настройки *ProGuard.*

## 6.5 Запрет на запись видео и создание снимка экрана

В мобильном приложении также установлен запрет на запись видео и создание снимка экрана с помощью установки флагов для окна. Пример кода для установки запрел представлен на листинге 6.3.

getWindow().setFlags(WindowManager.LayoutParams.FLAG\_SECURE,  
 WindowManager.LayoutParams.FLAG\_SECURE);

Листинг 6.3 – Код для установки запрета

Функция *getWindow* получает окно, которое сейчас открыто на экране. А функция *setFlags* устанавливает нужные флаги для запрета с помощью параметров *FLAG\_SECURE.*

## 6.6 Выводы по разделу

Крайне важно обеспечить безопасность ценной информации, которую компания хранит и делает доступной с помощью программных приложений и веб-служб, от неавторизованных пользователей и злоумышленников. Но также очень важно, чтобы эти защищенные ресурсы, такие как информация о кредитных картах или номера социального страхования, были немедленно доступны авторизованным, законным пользователям и системам для проведения деловых операций.

Чтобы обеспечить безопасный доступ к информации, приложения и службы могут применять различные меры безопасности. В данном дипломном проекте были реализованы следующие методы:

* использование хеширования паролей;
* использование *HTTPS* протокола;
* применение обфускации;
* установка запрета на запись видео и создания снимка экрана.

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*8*

*ДП 07.00.ПЗ*

*Разраб.*

*Славин Н.В.*

*Провер.*

*.*

*Сахонь Е.С.*

Консульт.

Соболевский А.С.

*Н. Контр.*

Нистюк О.А.

*Утверд.*

*Смелов В. В.*

7 Экономическое обоснование проекта

*Лит.*

*Листов*

*10*

БГТУ 74217075, 2021

# 7 Экономическое обоснование проекта

## 7.1 Общая характеристика разрабатываемого программного средства

Во время разработки дипломного проекта использовались технологии .NET Core, Entity Framework, MS SQL Server и языки программирования Java, C#. Разработанное программное средство представляет собой мобильное приложение для выполнения заданий и получения вознаграждения «Квест», предназначенное для экономии времени пользователей путем публикации квестов и поиска их исполнителей. С помощью данного приложения пользователь может просматривать все квесты, добавлять свои квесты, отправлять заявки на выполнение чужих квестов. Также у приложения есть роль администратор, который может изменять и удалять квесты.

Мобильное приложение «Квест» превосходит аналогичные решения, которые были рассмотрены в данном дипломном проекте: простым и лаконичным дизайном, пользователь может добавлять, изменять и удалять свои квесты, а также выбирать исполнителей квеста из числа подавших заявку.

По результатам анализа применяемых продуктами-аналогами стратегий монетизации следует выбрать стратегию монетизации: реклама в приложении (пользователи активно скачивают бесплатные приложения, а их издатели получают прибыль с показа рекламы).

## 7.2 Исходные данные для проведения расчётов

Источниками исходных данных для данных расчетов выступают действующие законы и нормативно-правовые акты. Исходные данные для расчета приведены в таблице 7.1.

Таблица 7.1 – Исходные данные для расчета

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование показателя | Единица измерения | Условные обозначения | Норматив |
| Численность разработчиков | чел. | Чр | 1,0 |
| Норматив дополнительной заработной платы | % | Ндз | 15,0 |
| Ставка отчислений в Фонд социальной защиты населения | % | Нфсзн | 34,0 |
| Ставка отчислений в БРУСП «Белгосстрах» | % | Нбгс | 0,6 |
| Цена одного машино-часа | руб. | Смч | 0,5 |
| Норматив прочих затрат | % | Нпз | 18,5 |
| Норматив накладных расходов | % | Нобп, обх | 30,0. |
| Норматив расходов на сопровождение и адаптацию | % | Нрса | 17,0 |
| Ставка НДС | % | ННДС | 20,0 |

## 7.3 Объем программного средства

В таблице 7.2 указаны в укрупнённом виде все работы, реально выполненные для создания, указанного в дипломной работе программного средства и количество рабочих дней, реально потраченных для выполнения этих работ.

Таблица 7.2 – Затраты рабочего времени на разработку ПС

|  |  |
| --- | --- |
| Содержание работ | Затраты рабочего времени, дней |
| 1.Построение диаграмм для проектирование дипломного проекта | 5 дня |
| 2.Создание базы данных | 3 дня |
| 3.Разработка серверной части | 10 дней |
| 4.Разработка клиентской части | 17 дней |
| 5.Тестирование внешнего вида | 2 дня |
| 6.Тестирование бизнес логики | 3 дня |
| 7.Тестирование базы данных | 1 день |
| 8.Написание руководства пользователя | 1 день |
| Всего | 42 дня |

Результат (значение «всего») по данной таблице будет использовано далее для расчётов.

Для оценки объёма программного средства, все его функции классифицируются с использованием специального каталога функций, который определяет их объем. Общий объем программного средства *V*o, вычисляется как сумма объёмов *Vi* каждой из *n* его функций по формуле 7.1.

|  |  |  |
| --- | --- | --- |
|  | , | (7.1) |

где – объем *i*-ой функции ПС, условных машинных команд;

*n* – общее число функций.

В таблице 7.3 представлены функции, присутствующие в рассматриваемом программном средстве и соответствующий им объем в условных машино–командах.

Таблица 7.3 – Содержание и объем функций в программном средстве

|  |  |  |
| --- | --- | --- |
| № функции | Содержание функции | Объем, условных машино-команд |
| 101 | Организация ввода информации | 620 |
| 102 | Контроль, предварительная обработка | 350 |
| 111 | Управление вводом/выводом | 800 |
| 202 | Взаимодействие между компонентами системы | 1400 |
| 401 | Взаимодействие с базой данных | 950 |
| 402 | Вспомогательные методы | 700 |
| 506 | Обработка ошибочных и сбойных ситуаций | 450 |
| 707 | Графический вывод результатов | 1500 |

Опираясь на данные таблицы 7.3, можно определить объем программного средства, разработанного в ходе дипломного проектирования:

*Vo* = 620 + 350 + 800 + 1400 + 950 + 700 + 450 + 1500 = 6 770 (маш. команд).

Уточнённый объем программного средства *V*o*/* вычисляется по формуле 7.2.

|  |  |  |
| --- | --- | --- |
|  | , | (7.2) |

где *V*o – объем программного средства, усл. машино-команд;

Кск - коэффициент изменения скорости обработки информации.

Исходя из вычисленного объёма программного средства, можно определить уточненный объем программного средства:

*V*o*/* = 6 770 ⋅ 0,6 = 4 062 (условных машино–команд).

## 7.4 Основная заработная плата

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для специалистов в программирования на Java Android. В итоге было установлено, что средняя месячная заработная плата на позиции junior составляет 1 532 рублей

Согласно таблице 7.2, проект разрабатывался одним человеком на протяжении 42 рабочих дня, что соответствует 2 месяцам. Таким образом, основная заработная плата будет рассчитываться по формуле 7.3:

|  |  |  |
| --- | --- | --- |
|  | , | (7.3) |

где Соз– основная заработная плата, руб.;

Траз – время раработки, месяцев;

Краз – количество разработчиков, человек;

Сзп – средняя месячная заработная плата.

|  |  |  |
| --- | --- | --- |
|  | руб. |  |

В дальнейшем для других расчётов используется основная заработная плата, рассчитанная по указанной выше методике.

## 7.5 Дополнительная заработная плата

Дополнительная заработная плата на конкретное программное средство включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах к основной заработной плате по формуле (7.4):

|  |  |  |
| --- | --- | --- |
|  | , | (7.4) |

где Соз– основная заработная плата, руб.;

Ндз – норматив дополнительной заработной платы, %.

руб.

## 7.6 Отчисления в Фонд социальной защиты населения

Отчисления в Фонд социальной защиты населения (ФСЗН) и по обязательному страхованию от несчастных случаем на производстве и профессиональных заболеваний определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей.

Отчисления в Фонд социальной защиты населения вычисляются по формуле 7.5

|  |  |  |
| --- | --- | --- |
|  |  | (7.5) |

где – основная заработная плата, руб.;

– дополнительная заработная плата на конкретное ПС, руб.;

– норматив отчислений в Фонд социальной защиты населения, %.

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле 7.6

|  |  |  |
| --- | --- | --- |
|  |  | (7.6) |

Таким образом, общие отчисления в БРУСП «Белгосстрах» составили 21,14 руб., а в фонд социальной защиты населения – 1 198,024 руб.

## 7.7 Расходы на материалы

Сумма расходов на материалы СМ определяется как произведение нормы расхода материалов в расчете на сто строк исходного кода НМ на уточненный объем программного средства *Vo/* , по формуле 7.7

|  |  |  |
| --- | --- | --- |
|  | . | (7.7) |

Учитывая, что норма расхода материалов в расчете на сто строк исходного кода равен 0,46 руб. (по данным, приведенным в приложении 2 таблице П 2.10 «Оценка значений среднего расхода материалов на разработку и отладку 100 строк кода применения ПС» методического пособия), можно определить сумму расходов на материалы.

СМ = 0,46 ⋅ 4 062 / 100 = 18,7 руб.

Сумма расходов на материалы была вычислена на основе данных приведенных в таблице 7.1 данного дипломного проектирования.

## 7.8 Расходы на оплату машинного времени

Сумма расходов на оплату машинного времени Смв определяется как произведение стоимости одного машино-часа Смч на уточненный объем программного средства *Vo/* и на норматив расхода машинного времени на отладку ста строк исходного кода НМВ, по формуле 7.8

|  |  |  |
| --- | --- | --- |
|  | . | (7.8) |

Учитывая, что норматив машинного времени на отладку ста строк исходного кода равен 12 (по данным, приведенным в приложении 2 таблице П 2.11 «Оценка значений среднего машинного времени на отладку 100 строк исходного кода без применения ПС» методического пособия), можно определить сумму расходов на оплату машинного времени.

Смв = 0,5 ⋅ 4 062 ⋅ 12 / 100 = 243,72 руб.

Сумма расходов на оплату машинного времени была вычислена на основе данных приведенных в таблице 7.1 данного дипломного проектирования.

## 7.9 Прочие прямые затраты

Сумма прочих затрат Спз определяется как произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив прочих затрат в целом по организации Нпз, и находится по формуле 7.9.

|  |  |  |
| --- | --- | --- |
|  |  | (7.9) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму прочих затрат.

Спз = 3 064 ⋅ 18,5 / 100 = 566,84 руб.

## 7.10 Накладные расходы

Сумма накладныхрасходовСобп,обх – произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив накладных расходов в целом по организации Нобп,обх, по формуле 7.10.

|  |  |  |
| --- | --- | --- |
|  |  | (7.10) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму накладных расходов.

= 3 064 ⋅ 30 / 100 = 919,2 руб.

## 7.11 Сумма расходов на разработку программного средства

Сумма расходов на разработку программного средства Ср определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, расходов на материалы, расходов на оплату машинного времени, суммы прочих затрат и суммы накладных расходов, по формуле 7.11

|  |  |  |
| --- | --- | --- |
|  | Ср = Соз + Сдз + Сфсзн + Сбгс + См + Смв + Спз + Собп,обх. | (7.11) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму расходов на разработку программного средства.

Ср = 3 064 + 459,6 + 1 198,024 + 21,14 + 18,7 + 243,72 + 566,84 + 919,2 = 6 491,22 руб.

Сумма расходов на разработку программного средства была вычислена на основе данных, рассчитанных ранее в данном разделе и составила 4 465,52 рублей.

## 7.12 Расходы на сопровождение и адаптацию

Сумма расходов на сопровождение и адаптацию программного средства Срса определяется как произведение суммы расходов на разработки на норматив расходов на сопровождение и адаптацию Нрса, и находится по формуле 7.12.

|  |  |  |
| --- | --- | --- |
|  |  | (7.12) |

= 6 491,224 ⋅ 17 / 100 = 1 103,51 руб.

Сумма расходов на сопровождение и адаптацию была вычислена на основе данных, рассчитанных ранее в данном разделе.

Все проведенные выше расчеты необходимы для вычисления полной себестоимости проекта.

## 7.13 Полная себестоимость

Полная себестоимость Сп определяется как сумма двух элементов: суммы расходов на разработку Ср и суммы расходов на сопровождение и адаптацию программного средства Срса.

Полная себестоимость Сп вычисляется по формуле 7.13

|  |  |  |
| --- | --- | --- |
|  |  | (7.13) |

Сп = 6 491,224 + 1 103,51 = 7 594,734 руб.

Полная себестоимость программного средства была вычислена на основе данных, рассчитанных ранее в данном разделе.

## 7.14 Определение цены, оценка эффективности

Продукты-аналоги:

1. *ALOT.PRO* – в приложении автоматически собираются и выкладываются в свободный доступ пользователям задания с разных бирж фриланса. Каких именно, пользователю нужно выбрать самому при регистрации и установке приложения на телефон.
2. *FL.ru* – Приложение позволяет найти исполнителей на любое задание. Опубликовать бесплатный проект и выбирать исполнителя среди фрилансеров. Существует возможность использовать «Безопасную сделку» как удобный и легальный способ работать с фрилансерами из других стран, как для организаций, так и для физических лиц.
3. *Freelancer* – приложение позволяет публиковать проекты на исполнение бесплатно, использовать фиксированную стоимость проекта, имеет надежную систему платежей, позволяет общаться фрилансерам и заказчикам с помощью мгновенного мессенджера. Являясь фрилансером вы сможете предлагать свои услуги работодателям со всего мира и работать, когда вам удобно и где вам удобно.

По результатам анализа применяемых продуктами-аналогами стратегий монетизации следует выбрать стратегию монетизации: контекстная реклама.

Были выбраны следующие характеристики для показателей качества рассматриваемого программного продукта и программного продукта конкурента:

1. Дизайн – то, как приложение выглядит, очень важно для любого современного приложения.
2. Юзабилити – насколько приложение удобно в использовании
3. Количество фич – количество инноваций, внедряемых в приложение
4. Количество багов – наличие несущественных ошибок в работе. Чем меньше - тем лучше.

Расчет показателей качества базового и нового продуктов согласно балловому методу приводится в таблицах 7.4 и 7.5.

Таблица 7.4 – Содержание и объем функций в программном средстве

|  |  |  |  |
| --- | --- | --- | --- |
| Показатель качества | Весовой коэффициент | Квест | ALOT.PRO |
| Дизайн | 0,3 | 7 | 7 |
| Юзабилити | 0,4 | 7 | 7 |
| Количество фич | 0,2 | 6 | 4 |
| Количество багов | 0,1 | 1 | 2 |
| *Всего* | *1* | *6,2* | *5.8* |

Таблица 7.5 – Содержание и объем функций в программном средстве

|  |  |  |  |
| --- | --- | --- | --- |
| Показатель качества | Весовой коэффициент | Freelancer | FL.ru |
| Дизайн | 0,3 | 6 | 6 |
| Юзабилити | 0,4 | 5 | 4 |
| Количество фич | 0,2 | 4 | 3 |
| Количество багов | 0,1 | 2 | 3 |
| *Всего* | *1* | *4,8* | *4.3* |

Расчёт прогнозного количества установок программного средства К1 при монетизации методом размещения рекламы, рассчитывается по формуле 7.14:

|  |  |
| --- | --- |
|  | (7.14) |

где К0 − количество установок ПС конкурента;

T0 – количество лет существования приложения;

ИР – показатель рассматриваемого программного продукта;

ИК – показатель программного продукта конкурента.

К1 = (1 000 000 / 4 \* 6,2) / 5,8 = 267 241,4 (установок в год),

К2 = (5 000 000 / 7 \* 6,2) / 4,8 = 922 619,05 (установок в год),

К3 = (100 000 / 2 \* 6,2) / 4,3 = 72 093,02 (установок в год),

К = (267 241,4 + 922 619,05 + 72 093,02) / 3 = 420 651,2 (установок в год).

По данным исследований в *Android*-приложениях цена за тысячу показов составляет в среднем $4 (11 рублей). Если среднее количество установок 420 651,2 за год, то денежные поступления от показа рекламы в приложении Пост.в год = 4 627,2 рублей за год.

Срок окупаемости приложения Tок вычисляется по формуле 7.15

|  |  |  |
| --- | --- | --- |
|  |  | (7.15) |

где – полная себестоимость, руб.;

Пост.в год – денежные поступления от показа рекламы в приложении за год, руб.;

Ток = 7 594,73 / 4 627,2 = 1,6 года.

## 7.15 Вывод по разделу

В таблице 7.6 представлены результаты расчётов для основных показателей данной главы в краткой форме.

Таблица 7.6 – Результаты расчетов

|  |  |
| --- | --- |
| Наименование показателя | Значение |
| Время разработки, мес. | 2 |
| Количество программистов, чел. | 1 |
| Основная заработная плата, руб. | 3 064 |
| Дополнительная заработная плата, руб | 459,6 |
| Отчисления в Фонд социальной защиты населения и по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний, руб | 1 219,2 |
| Расходы на материалы, оплату машинного времени, прочие, руб | 829,26 |
| Накладные расходы, руб | 919,2 |
| Себестоимость разработки программного средства, руб. | 6 491,22 |
| Расходы на сопровождение и адаптацию, руб. | 1 103,51 |
| Полная себестоимость, руб. | 7 594,73 |
| Годовые денежные поступления от показа рекламы, руб. | 4 627,2 |
| Срок окупаемости, лет | 1,6 |

Разработка программного средства, осуществляемая одним программистом в течении 2 месяцев, при заданных условиях обойдется в 7 594,73 руб. Реализации данного программного средства по будет приносить годовые денежные поступления от показа рекламы в размере 4 627,2 рублей и окупиться через 1,6 года.

Необходимость разработки программного средства, обусловлена все большим ростом популярности мобильных приложений для фриланса. Количество фрилансеров постоянно растёт, а мобильных приложений с удобным интерфейсом и лаконичным дизайном не так много. Чтобы решить данную проблему, было разработано программное средство, которое дает возможность добавлять свои собственные квесты и выполнять квесты других пользователей.

# Заключение

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*8*

*ДП 00.00.ПЗ*

*Разраб.*

*Славин Н.В.*

*Провер.*

*.*

*Сахонь Е.С.*

Консульт.

*Н. Контр.*

Нистюк О.А.

*Утверд.*

*Смелов В. В.*

Заключение

*Лит.*

*Листов*

*1*

БГТУ 74217075, 2021

В ходе выполнения дипломного проекта были рассмотрены основные аналогичные решения, изучены возможности рассматриваемых программных средств. Произведен обзор теоретического материала по теме дипломного проекта.

Были проанализированы и выбраны основные технологии и средства для разработки дипломного проекта. Программное средство реализовано на языках *Java* и C#. В качестве системы управления базой данных использована СУБД *Microsoft SQL Server*.

При разработке программного средства была спроектирована архитектура приложения, а также была модифицирована файловая структура проекта. В качестве клиентской части, было разработано мобильное приложение на языке *Java* под операционную систему *Android.*

При составлении руководства пользователя была подробно описана работа с программным средством. Так же было проведено тестирование, которое показало, что разработанное программное средство соответствует заданным требованиям.

При рассмотрении технико-экономического обоснования разработанного проекта было рассчитано количество денежных затрат и трудозатрат на разработку программного средства, прибыль от реализации разработанного программного продукта.

Целью данного дипломного проекта являлась проектирование и разработка мобильного приложения для выполнения заданий и получения вознаграждения «Квест». Для достижения поставленной цели были выполнены следующие задачи:

* проектирование и создание базы данных;
* разработка *Web-API*;
* разработка функционала регистрации и авторизации пользователей;
* обеспечение защиты данных пользователей;
* создание дизайна пользовательского интерфейса;
* разработка мобильного приложения под операционную систему *Android*.

В рамках дипломного проекта было разработано программное средство, которое соответствует предъявленным требованиям и поставленным задачам.

# Список использованных источников

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*8*

*ДП 00.00.ПЗ*

*Разраб.*

*Славин Н.В.*

*Провер.*

*.*

*Сахонь Е.С.*

Консульт.

*Н. Контр.*

Нистюк О.А.

*Утверд.*

*Смелов В. В.*

Список использованных источников

*Лит.*

*Листов*

*1*

БГТУ 74217075, 2021

1. SQL Server [Электронный ресурс]. – Режим доступа: <https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15>/. – Дата доступа: 14.03.2021.
2. Руководство по MS SQL Server 2019 [Электронный ресурс]. – Режим доступа: <https://metanit.com/sql/sqlserver/>. – Дата доступа: 20.03.2021
3. C# Reference [Электронный ресурс]. – Режим доступа: <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/>. – Дата доступа: 27.03.2021.
4. Рихтер, Дж. CLR via С#. Программирование на платформе Microsoft .NET Framework 4.0 / Дж. Рихтер. – 3-е изд. ¬– СПб.: Питер, 2012. – 928 с.
5. Руководство по ASP.NET Core 3 [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/aspnet5/>. – Дата доступа: 10.04.2021.
6. Введение в Entity Framework [Электронный ресурс]. – Режим доступа: https://professorweb.ru/my/entity-framework/6/level1/. – Дата доступа: 21.04.2021.
7. Программирование под Андроид на Java [Электронный ресурс]. – Режим доступа: <https://metanit.com/java/android/>. – Дата доступа: 30.04.2021
8. Руководство по языку программирования Java [Электронный ресурс]. – Режим доступа: <https://metanit.com/java/tutorial/>. – Дата доступа: 07.05.2021
9. Retrofit documentation [Электронный ресурс]. – Режим доступа: <https://square.github.io/retrofit/>. – Дата доступа: 14.05.2021
10. Библиотека Retrofit [Электронный ресурс]. – Режим доступа: <http://developer.alexanderklimov.ru/android/library/retrofit.php/>. – Дата доступа: 21.05.2021

# ПРИЛОЖЕНИЕ А

Даиграмма вариантов использования

# ПРИЛОЖЕНИЕ Б

Схема архитектуры приложения

# ПРИЛОЖЕНИЕ В

Логическая схема базы данных

# ПРИЛОЖЕНИЕ Г

Блок-схема алгоритма авторизации в приложении

# ПРИЛОЖЕНИЕ Д

Таблица экономических расчётов