**Содержание**

[Введение 2](#_Toc163205813)

[1. Аналитическая часть 3](#_Toc163205814)

[1.1. Интервью с заказчиком 3](#_Toc163205815)

[1.2. Описание предметной области 4](#_Toc163205816)

[1.3. Описание вариантов использования приложения 6](#_Toc163205817)

[1.4. Классификация пользователей результатов проекта 7](#_Toc163205818)

[1.5. Обоснование выбора средств разработки 7](#_Toc163205819)

[1.6. Описание структуры данных 9](#_Toc163205820)

[1.7. Описание архитектуры приложения 9](#_Toc163205821)

[1.8. Проектирование интерфейса 11](#_Toc163205822)

[2. Технологическая часть 16](#_Toc163205823)

[2.1. Организация данных 16](#_Toc163205824)

[2.2 Разработка дизайна приложения – разметка + листнинг 17](#_Toc163205825)

[2.3 Разработка API 21](#_Toc163205826)

[2.4 Протоколы тестирования 22](#_Toc163205827)

[2.5 Разграничение прав доступа 23](#_Toc163205828)

[3. Экономическая часть 25](#_Toc163205829)

[3.1 Описание качества программного продукта 25](#_Toc163205830)

[3.2. Управление процессом разработки 26](#_Toc163205831)

[3.3. Предварительная оценка трудоемкости разработки 26](#_Toc163205832)

[Заключение 31](#_Toc163205833)

[Использованные источники информации 32](#_Toc163205834)

## Введение

Медицина занимает важную роль в жизни человека. Во многих медицинских учреждениях, главной задачей является обеспечение быстрого обслуживания пациентов.

Актуальность создания мобильного приложения для записи к врачу заключается в необходимости использования постоянно увеличивающегося объема информации при решении различных медицинских задач.

Целью курсового проекта является создание мобильного приложения, которое позволит улучшить процесс работы медицинских учреждений, а также обслуживание пациентов.

Объектом разработки является процесс обслуживания населения в медицинской сфере.

Предметом исследования является система для автоматизации процесса обслуживания пациентов.

Реализация выше поставленной цели будет возможна после решения ряда задач:

1. Анализировать предметную область;

2. Реализовать пользовательский интерфейс;

3. Реализовать возможности записи к врачу;

Функциональные требования, выявленные при разработке мобильного приложения:

1. Регистрация и авторизация пользователя;

2. Возможность присваивания роли пользователю;

3. Возможность поиска врачей;

4. Возможность записи на прием;

5. Возможность получения результата.

Основанием для разработки проекта является учебный план специальности 09.02.07 «Информационные системы и программирование»

## 1. Аналитическая часть

## 1.1. Интервью с заказчиком

Интервью с заказчиком, это процесс беседы с представителем заказчика или ответственным за проект лицом, во время которого собирается информация о требованиях, целях, ожиданиях и предпочтениях заказчика относительно приложения для тестирования студентов. Вот несколько вопросов, которые могут быть заданы в интервью с заказчиком:

1. Что является главной причиной для создания приложения для записи к врачу?

Мы хотим улучшить процесс работы медицинских учреждений, а также обслуживание пациентов.

1. Какие основные функции и возможности вы хотели бы видеть в приложении?

Во-первых, нам нужна функция поиска врачей, чтобы пациенты могли найти нужного специалиста по имени, специализации. Во-вторых, нужна возможность просмотра информации о каждом враче. И, конечно же, самая главная функция – возможность записаться на прием у выбранного врача.

1. Какую информацию вы хотели бы получать о пользователях и записях на прием?

Понадобится такая информация как: имя, фамилия, адрес электронной почты, номер телефона.

1. Какую платформу вы предпочли бы для разработки приложения – IOS или Android?

Нам было бы предпочтительнее иметь приложение для обеих платформ, чтобы охватить максимальное количество пользователей.

1. Какое взаимодействие должно быть между приложением и вашей базой данных о врачах и пациентах?

Нам нужно, чтобы приложение было интегрировано с нашей существующей базой данных, чтобы пациенты могли видеть актуальную информацию о врачах.

1. Важны ли для вас какие-либо дополнительные функции или требования?

Нам важна защита данных пациентов и их конфиденциальность.

## 1.2. Описание предметной области

Сейчас трудно представить человека без смартфона. Но, не во всех учебных заведениях есть возможность оборудовать достаточное количество компьютерных классов. На помощь приходит смартфон. Он не требует использовать его в компьютерном классе, тем самым сможет упростить этапы обучения, как и для студента, так и для преподавателя.

Мобильное приложение, представляющее собой максимально удобный инструмент для тестирования преподавателями студентов по разным преподаваемым дисциплинам, не используя компьютерный класс, но зато используя смартфон.

Мобильное приложение тестирования дает возможность преподавателю составлять тесты для дальнейшего прохождения их студентами, также дает возможность выбора тестов для дальнейшего прохождения, по завершению прохождения тестирования система выдает результаты тестирования студентам и преподавателю.

Также данное мобильное приложение имеет функции регистрации/авторизации с выбором роли (студент или преподаватель), возможность сохранения результатов тестирования уже пройденных тестов.

Исходя из выявленных функциональных требований была построена Use-case диаграмма, представленная на рисунке 1:

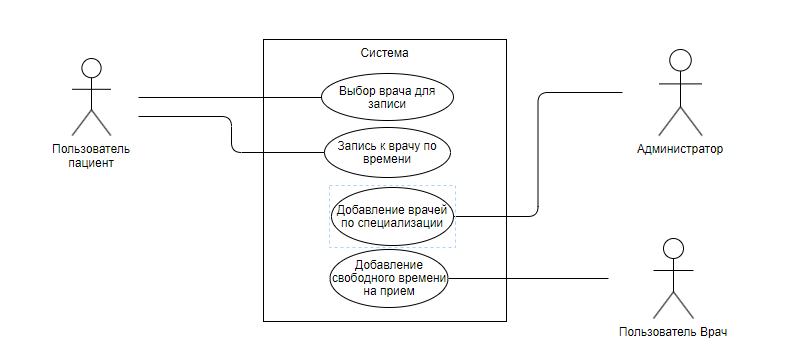


Рисунок 1. Use-case диаграмма.

На основе составленной Use-case диаграммы было выделено 4 прецедента. Рассмотрим прецедент “Выбор тестов”:

Таблица 1 — Прецедент “Выбор тестов”

|  |  |
| --- | --- |
| Название прецедента | Выбор тестов. |
| Цель | Выбор теста студентом составленного преподавателем. |
| Актеры | Пользователь студент(инициатор), Пользователь преподаватель, Система. |
| Ссылки (предусловия) | Реализованы прецеденты: Получение результатов тестирования. |
| Результаты (постусловия) | Пользователь студент заканчивает тестирование. Система определяет количество правильных и неправильных ответов. После этого система отправляет результат студенту и преподавателю для выставления оценки |

## 1.3. Описание вариантов использования приложения

На основе выявленных прецедентов в Use-case диаграмме был составлен основной успешный сценарий, представленный в таблице 2:

Таблица 2 — Основной вариант использования для студента

|  |  |
| --- | --- |
| Действия актера | Отклик системы |
| 1. Пользователь студент выбирает тест составленный преподавателем.  2. Пользователь студент проходит тестирование.  4. Пользователь студент завершает тестирование. | 3. Система ведет подсчет верных и неверных ответов студента и отправляет результат пользователю студенту и пользователю преподавателю.  5. Система отправляет результат тестирования. |

На основе успешного варианта использования можно составить альтернативный поток событий:

1. Произошла ошибка в системе. Система выдает сообщение об ошибке.

3. Данные отсутствуют в системе. Система выдает сообщение об ошибке.

4. Данные отсутствуют в системе. Система выдает сообщение об ошибке.

Таблица 3 — Основной вариант использования для преподавателя

|  |  |
| --- | --- |
| Действия актера | Отклик системы |
| 1. Пользователь преподаватель создает вариант тестирования.  2. Пользователь преподаватель заполняет тест вопросами и ответами.  4. Пользователь преподаватель закрывает доступ к тестированию. | 3. Система ведет подсчет верных и неверных ответов студента и отправляет результат пользователю студенту и пользователю преподавателю.  5. Система отправляет результат тестирования. |

На основе успешного варианта использования можно составить альтернативный поток событий:

1. Произошла ошибка в системе. Система выдает сообщение об ошибке.

3. Данные отсутствуют в системе. Система выдает сообщение об ошибке.

4. Данные отсутствуют в системе. Система выдает сообщение об ошибке.

## 1.4. Классификация пользователей результатов проекта

Классификация пользователей результатов проекта для приложения тестирования студентов может быть следующей:

1. Администраторы/преподаватели: Эта группа пользователей будет ответственна за создание и управление тестами, добавление вопросов, назначение тестов студентам, а также просмотр и анализ результатов тестирования. Они имеют специальные права доступа для администрирования и управления системой тестирования.

2. Студенты: В эту группу входят сами студенты, которые будут проходить тесты. Они имеют доступ для выполнения тестов, выбора ответов на вопросы и отправки своих ответов.

# 1.5. Обоснование выбора средств разработки

Выбор средств разработки Kotlin может быть обоснован следующими факторами:

1. Мультиплатформенность: Kotlin предоставляет возможность разработки как для платформы Java, так и для платформы Android. Это позволяет максимально использовать код и ресурсы и повышает эффективность разработки приложений для разных платформ.

2. Легкая интеграция: Kotlin может работать существующим Java-кодом, что делает его простым в использовании в существующих проектах или при создании новых. Он также имеет отличную совместимость с мощной экосистемой инфраструктуры Java.

3. Безопасность: Kotlin предоставляет набор функций и возможностей, которые помогают предотвратить ошибки и сбои программы, такие как проверки на null, безопасные вызовы и различные проверки типов. Это способствует созданию более надежного и стабильного кода.

4. Краткость и выразительность: Kotlin имеет простой и понятный синтаксис, который делает код более читабельным и удобным для понимания. Он также предоставляет множество удобных функций, таких как функции расширения и функциональное программирование, что упрощает разработку и сокращает количество кода.

5. Активная поддержка и сообщество: Kotlin активно развивается и поддерживается компанией JetBrains, что гарантирует обновления и исправление ошибок. Он также имеет развитое сообщество разработчиков, которые предоставляют полезные ресурсы и помощь во время разработки.

В целом, выбор Kotlin в качестве средства разработки может быть обоснован его множеством преимуществ, таких как мультиплатформенность, легкая интеграция, безопасность, краткость и выразительность, а также активная поддержка и сообщество.

# 1.6. Описание структуры данных

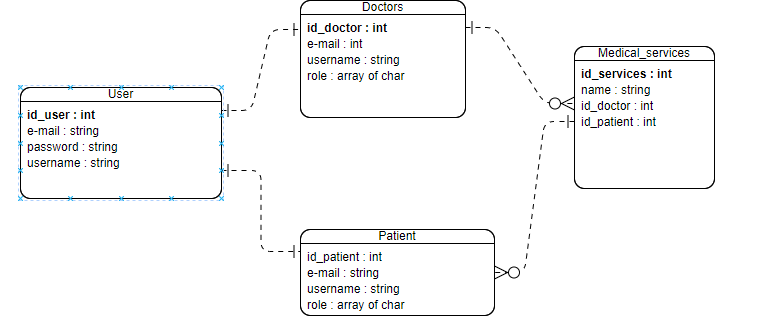
По результатам описания предметной области была составлена ERD диаграмма, представленная на рисунке 2.

Рисунок 2. ER – Диаграмма

# 1.7. Описание архитектуры приложения

Архитектура приложения тестирования для студентов может быть организована в соответствии с принципами модульности и разделения ответственности. Вот возможное описание основных компонентов архитектуры приложения:

1. Клиентский интерфейс: Это часть приложения, которая взаимодействует с пользователями - студентами. Она включает в себя пользовательский интерфейс (UI), который позволяет студентам пройти тесты, выбирать ответы на вопросы и просматривать результаты.

2. Бизнес-логика: Этот компонент отвечает за обработку и управление данными, логикой бизнес-процессов и выполнением операций с тестами и результатами. Он включает в себя функции для создания, редактирования и удаления тестов и вопросов, а также обработки ответов студентов и расчета результатов.

3. Серверная инфраструктура: Эта часть архитектуры отвечает за хранение данных приложения и взаимодействие с базой данных. Она включает в себя серверное API, которое обрабатывает запросы от клиента, а также модули доступа к данным, которые обеспечивают взаимодействие с базой данных.

4. База данных: В этой части архитектуры хранятся все данные, связанные с тестами, вопросами, результатами и студентами. База данных должна быть спроектирована таким образом, чтобы обеспечить эффективное хранение и извлечение данных, а также обеспечить целостность данных с помощью соответствующих ограничений и отношений.

5. Аутентификация и безопасность: Это компоненты, отвечающие за аутентификацию и авторизацию студентов, а также за защиту данных приложения. Безопасность может включать в себя шифрование данных, управление правами доступа и механизмы защиты от несанкционированного доступа.

При проектировании приложения была приведена следующая архитектура “Клиент-сервер”, представленная на рисунке 3

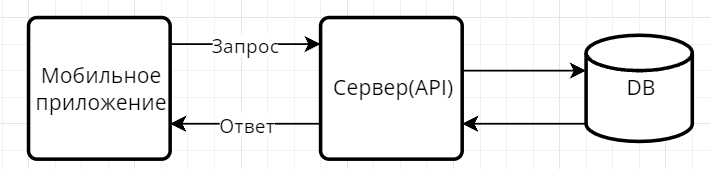


Рисунок 3. Схема архитектуры приложения

# 1.8. Проектирование интерфейса

При анализе предметной области был спроектирован проект интерфейса приложения, представленный на рисунках 4-10. Каждая форма выполняет определенную функцию.

|  |  |
| --- | --- |
| Рисунок 4. Форма “Авторизация” | Рисунок 5. Форма “Регистрация” |

|  |  |
| --- | --- |
| Рисунок 6. Окно “Выбор врача” | Рисунок 7. Окно “Запись к врачу” |

|  |  |
| --- | --- |
| Рисунок 8. Окно “Добавление услуги и свободного времени для приема” | Рисунок 9. Окно “Список записанных” |

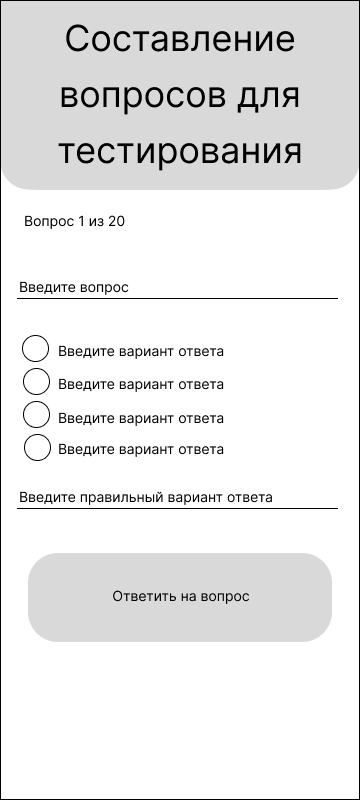


Рисунок 10. Окно “Составление вопросов для тестирования”

После создания проекта интерфейса была создана схема перемещений по приложению, представленная на рисунке 11:

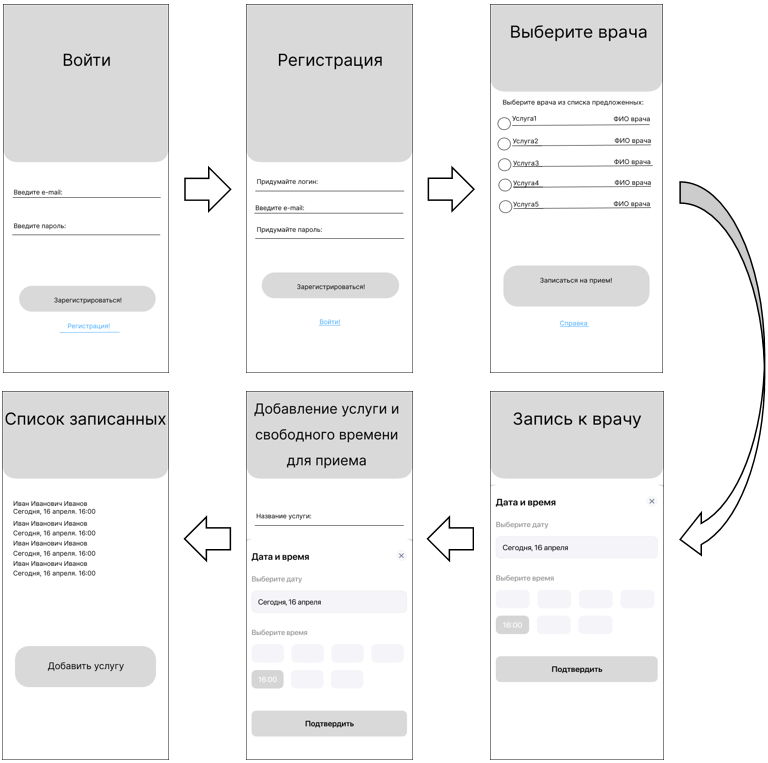


Рисунок 11. Окно “ Схема перемещений по приложению”

На рисунке ясно видно, как происходит работа в приложении.

# 2. Технологическая часть

## 2.1 Организация данных

Организация данных в приложении для тестирования студентов играет важную роль в эффективной работе приложения. На следующем рисунке представлена схема базы данных:

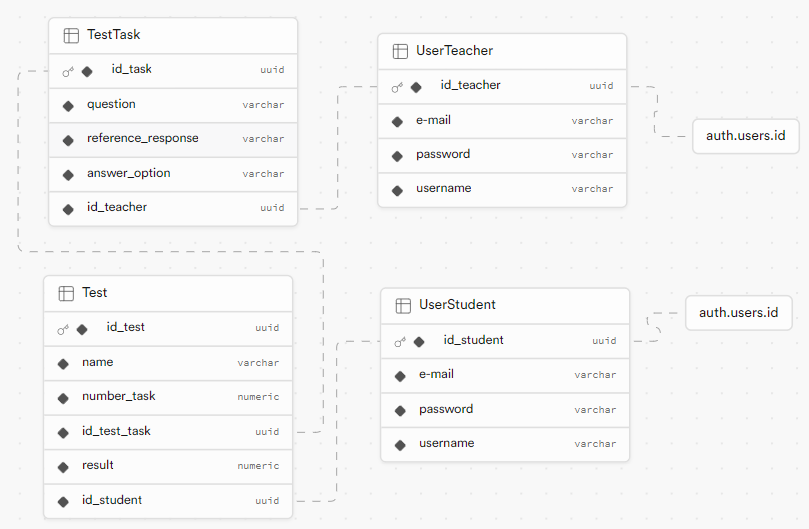


Рисунок 12. Схема базы данных

2.2 Разработка дизайна приложения – разметка + листнинг

Разработка интерфейса мобильного приложения для тестирования студентов требует внимательного планирования и проектирования. При разработке будет реализовано семь окон интерфейса представленные на рисунках 13-20.

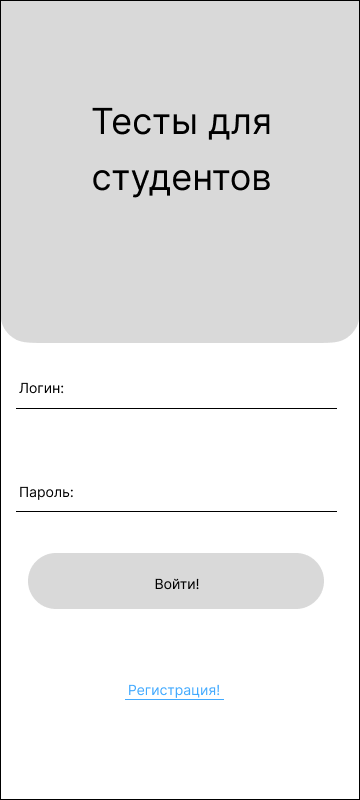


Рисунок 13. Окно авторизации пользователя

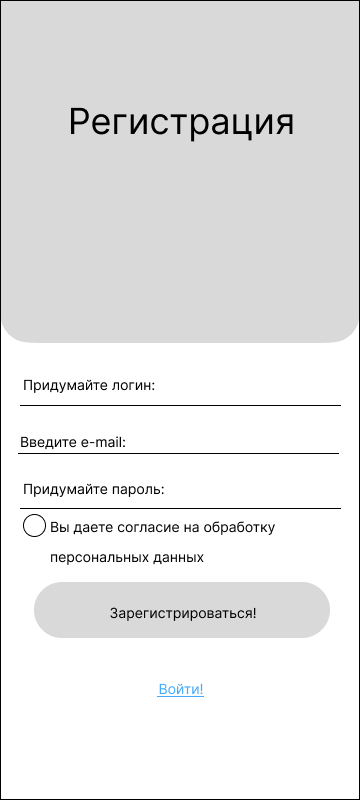


Рисунок 14 Окно регистрации пользователя

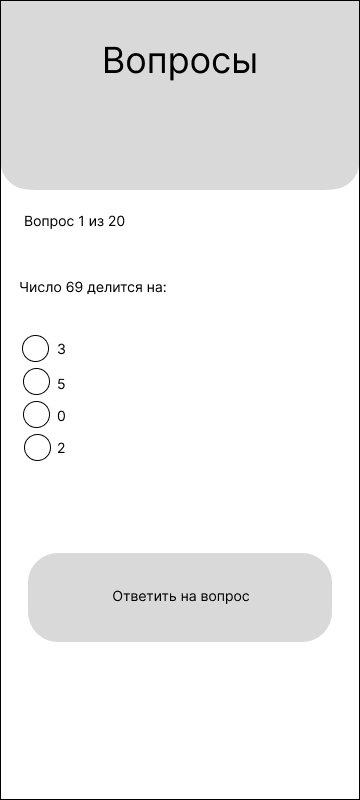


Рисунок 15 Окно показа вопроса с вариантами ответа

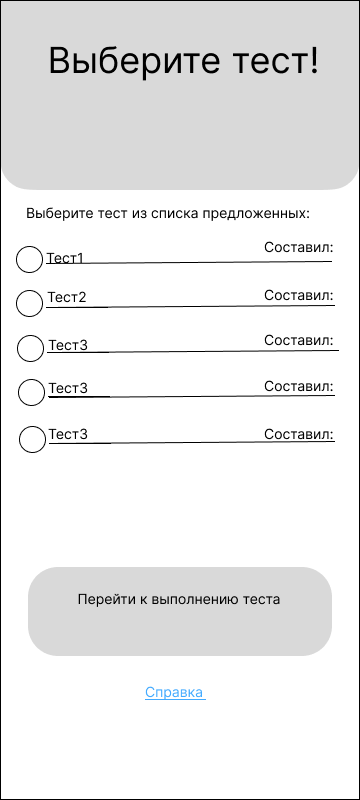


Рисунок 16 Окно выбора тестирования

2.3 Разработка API

При разработки API для мобильного приложения “Тестирование студентов” будет использована оболочка базы данных Supabase основанная на PostgreSQL. PostgreSQL является надежной и расширяемой реляционной базой данных, которая обеспечивает множество возможностей и функций для работы с данными. Имплементация библиотеки Supabase, представлена в листинг 1:

Листинг 1 – имплементирование библиотеки Supabase:

implementation(platform("io.github.jan-tennert.supabase:bom:VERSION"))

implementation("io.github.jan-tennert.supabase:postgrest-kt")

Для реализации функциональных требований были реализованы запросы: регистрация, авторизация, извлечения данных, вставки данных, обновление данных. Данные запросы представленные в листингах 2-5:

Листинг 2 запрос регистрации:

val user = supabase.auth.signUpWith(Email) {

email = "example@email.com"

password = "example-password"

}

Листинг 3 запрос авторизации:

supabase.auth.signInWith(Email) {

email = emailText

password = passwordText

}

Листинг 4 запрос авторизации:

supabase.auth.signInWith(Email) {

email = emailText

password = passwordText

}

Листинг 5 запрос извлечение данных:

val city = supabase.from("user").select().decodeSingle<user>()

Листинг 6 запрос вставки данных:

val test = test(name = "test1", number tasks = RecyclerItem)

supabase.from("cities").insert(city)

2.4 Протоколы тестирования

Протоколы тестирования – это документы, которые определяют последовательность и спецификацию шагов, которые должны быть выполнены для проведения тестов или проверки определенной функциональности. Для тестирования мобильного приложения мы будем использовать Тест-кейс, представленный на таблицах 4-5.

Таблица 4. Тест-кейс регистрация в приложении

|  |  |  |
| --- | --- | --- |
| Название: | Тест регистрации | |
| Предусловие | Ожидаемый результат | Результат теста |
| Открытие приложения на странице регистрации. | Мобильное приложения для тестирования студентов открыто и доступно. |  |
| Шаги теста: | | |
| Заполните форму регистрации в приложении:  “Логин”: Ввести “Test”  “Пароль”: Ввести “test” | Данные успешно введены, и приложение перешло на страницу выбора варианта тестирования. | Данные успешно введены, что подтверждается переходом на страницу выбора варианта тестирования. |

Таблица 5. Тест-кейс авторизации в приложении

|  |  |  |
| --- | --- | --- |
| Название: | Тест регистрации | |
| Предусловие | Ожидаемый результат | Результат теста |
| Открытие приложения на странице авторизации. | Мобильное приложения для тестирования студентов открыто и доступно. |  |
| Шаги теста: | | |
| Заполните форму авторизации в приложении:  “Логин”: Ввести “Test”  “Пароль”: Ввести “test” | Данные успешно введены, и приложение перешло на страницу выбора варианта тестирования. | Данные успешно введены, что подтверждается переходом на страницу выбора варианта тестирования. |

## 2.5 Разграничение прав доступа

Разграничение прав доступа в приложении для тестирования студентов – это важная функциональность, которая позволяет управлять доступом пользователей к различным функциям и данным в приложении на основе их ролей и ответственностей. Это помогает обеспечить безопасность, конфиденциальность и эффективность использования приложения. Вот несколько способов осуществления разграничение прав доступа в таком приложении:

1. Ролевая модель: Определите различные роли пользователей, такие как студент, преподаватель и администратор. Каждая роль должна иметь определенные права доступа. Студентам может быть разрешено проходить тесты и просматривать свои результаты, преподавателям – создавать и назначать тесты, а администраторам – управлять всеми аспектами приложения. 2. Уровень доступа к данным: Ограничьте доступ пользователей к определенным данным на основе их принадлежности к определенным группам или курсам. Например, студенты могут иметь доступ только к данным, связанным с их курсами, преподаватели – к данным своих студентов и курсов, которые они ведут, а администраторы – ко всем данным.

3. Функциональный доступ: Отдельно определите, какие функции приложения доступны для каждой роли. Только преподавателям может быть разрешено создавать и редактировать тесты, а студентам – только проходить тесты.

4. Авторизация и аутентификация: Предоставьте механизм аутентификации пользователя, чтобы убедиться, что только зарегистрированные и аутентифицированные пользователи имеют доступ к приложению. Это также позволяет вам привязывать определенные права доступа к каждому пользователю.

# Экономическая часть

## 3.1 Описание качества программного продукта

Данный программный продукт разработан на основе созданной базы данных «CarService».

Все функции отвечают требованиям поставленной задачи, протестированы и не имеют ошибок.

**Функциональные возможности**

Программный продукт соответствует требованиям поставленной задачи, протестирован и не имеет ошибок.

Только авторизованные пользователи могут вносить изменения.

**Надежность**

Средства проверки достоверности исходных данных - вывод производится из базы данных, элементы таблиц которой были загружены ранее.

**Практичность**

Данный проект является продуктом закрытого программного кода, поэтому нет возможности у любого человека редактировать данный программный продукт.

**Эффективность**

Высокая скорость отклика почти на любой программной и аппаратной конфигурации - низкие требования к платформе.

**Сопровождаемость**

Данный ПП обладает легкой расширяемостью. Функции не конфликтуют друг с другом, в связи с этим ошибок при работе с программой не возникает.

**3.2 Управление процессом разработки**

Перед проведением оценки трудоемкости выполнения проекта проведена декомпозиция работ, выполняемых для достижения целей проекта. Проведен анализ обобщенной операционной структуры трудозатрат для программных проектов, расписание проекта не потребовало дополнений.

На рисунке 17 представлена иерархическая структура работ на диаграмме Ганта.

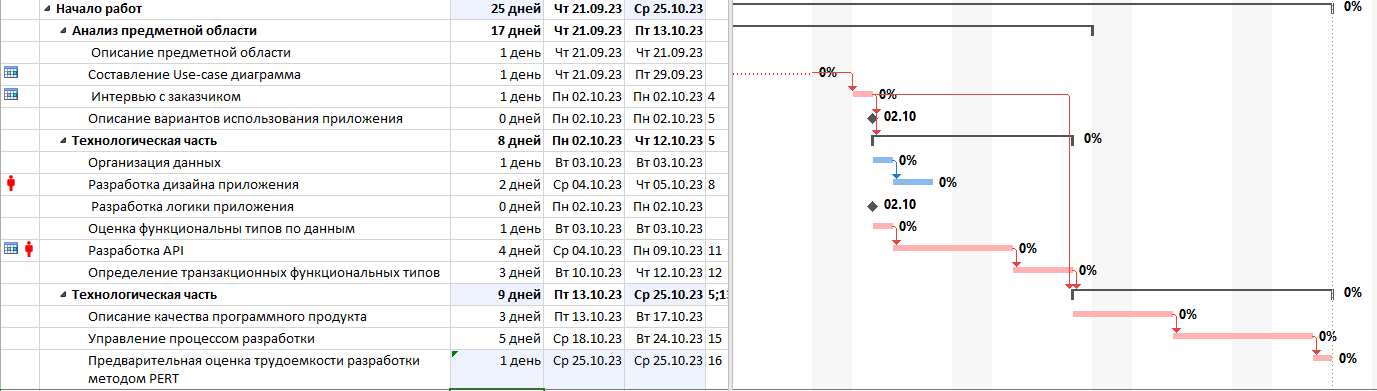


Рисунок 17 – Диаграмма Ганта

## 3.3 Предварительная оценка трудоемкости разработки

На основании перечня функций, выявленных на этапе анализа требований была выполнена экспертная оценка трудоемкости разработки приложения, с учетом его функций, представленная в таблице 3.

Таблица 6 – Экспертная оценка трудоемкости разработки приложения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Функция** | **Лучший случай** | **Наиболее вероятный** | **Худший случай** | **Ожидаемый случай** |
| Составление тестов | 4 | 2 | 6 | 3 |
| Выбор тестов | 2 | 4 | 6 | 4 |
| Прохождение тестов | 2 | 4 | 5 | 4 |
| Получение результатов тестов | 4 | 2 | 8 | 3 |
| **Итого** | **12** | **12** | **25** | **14** |

В предварительной оценке трудоёмкости разработки представлено количество дней, за которое возможно реализовать функции из Use-Case диаграммы используя формулу PERT (табл.3).

Оценку средней трудоемкости по каждому элементарному пакету можно определить по формуле:

(1)

1. Составление тестов.

Oi = 4

Mi = 2

Pi = 6

Средняя трудоемкость рассчитана по формуле (1):

1. Выбор тестов.

Oi = 2

Mi = 4

Pi = 6

Средняя трудоемкость рассчитана по формуле (1):

1. Прохождение тестов.

Oi = 2

Mi = 4

Pi = 5

Средняя трудоемкость рассчитана по формуле (1):

1. Получение результатов тестирования.

Oi = 4

Mi = 2

Pi = 8

Средняя трудоемкость рассчитана по формуле (1):

На основе экспертная оценки трудоемкости разработки мобильного приложения приложения было выявлено, что ожидаемый случай выполнения разработки программного продукта займет 14 дней.

– это среднеквадратичное отклонение Данный показатель можно расчитать по формуле 2.

(2)

1. Для составления тестирования, среднеквадратичное отклонение рассчитано по формуле (2).
2. Для выбора теста, среднеквадратичное отклонение рассчитано по формуле (2).
3. Для прохождения тестирования, среднеквадратичное отклонение рассчитано по формуле (2).
4. Для получения результатов тестирования, среднеквадратичное отклонение рассчитано по формуле (2).

СКО – это среднеквадратичное отклонение для оценки суммарной трудоёмкости. Данный показатель можно расчитать по формуле 3.

(3)

Среднеквадратичное отклонение для оценки суммарной трудоёмкости рассчитана по формуле (3).

Суммарная трудоёмкость проекта рассчитывается по формуле 4.

(4)

Е – это суммарная трудоёмкость проекта, которая определяется по формуле (4).

– это оценка суммарной трудоёмкости проекта, не превышающая с вероятностью в 95%, которая расчитывается по формуле 5.

(5)

Оценка суммарной трудоёмкости проекта, которую мы не превысим с вероятностью в 95% рассчитана по формуле (5).

Если сотрудник назначен на проект, то тратить он будет 60 – 80% своего рабочего времени. Поэтому, в месяц сотрудник будет работать по проекту, примерно.

Следовательно, трудоёмкость проекта в человеко-месяцах составит, приблизительно.

2596 / 226 ≈ 11,4

# Заключение

До начала реальной части проекта был произведён расчёт трудоёмкости работы. Было выявлено, какое количество человеко-часов необходимо для разработки проекта.

В результате работы была создана логическая модель базы данных в документе Microsoft Visio. Разработаны окна интерфейса, а также схема перемещения между ними. Информационная система состоит из:

ER-диаграммы логической модели базы данных, разработанной в Microsoft Visio;

Окна интерфейса будущего приложения, разработанные в Figma и Android studio.

В программе предусмотрены возможности:

* Просмотра данных;
* Добавления данных;
* Редактирования данных;
* Регистрации и Авторизации.

В ходе работы закреплена технология проектировки баз данных и диаграмм классов в Microsoft Visio, а также планирования создания окон интерфейса и перемещений между ними будущих экранов мобильного приложения. Были изучены и проведены расчёты необходимого времени и количества человек на разработку приложения.

# Использованные источники информации

1. ГОСТ Р ИСО 9000-2008 «Системы менеджмента качества»;
2. ГОСТ Р ИСО/МЭК 12207-99 «Процессы жизненного цикла программных средств»;
3. Мамедли Р. Э. Базы данных. Лабораторный практикум : учебное пособие для вузов / Р. Э. Мамедли. – Санкт-Петербург : Лань, 2023. – 152 с.;
4. Нестеров С. А. Интеллектуальный анализ данных с использованием SQL Server : учебник для вузов / С. А. Нестеров. – Санкт-Петербург : Лань, 2023. – 160 с.;
5. Макшанов А. В. Большие данные. Big Data : учебник для СПО / А. В. Макшанов, А. Е. Журавлев, Л. Н. Тындыкарь. – 3-е изд., стер – Санкт-Петербург : Лань, 2023. – 188 с.;
6. Волк В. К. Базы данных. Проектирование, программирование, управление и администрирование : учебник для СПО / В. К. Волк. – 2-е изд., стер. – Санкт-Петербург : Лань, 2022. – 340 с.;
7. Васильева М. А. Информационное обеспечение систем управления. Проектирование базы данных с заданиями : учебник для вузов / М. А. Васильева, К. М. Филипченко, Е. П. Балакина. – Санкт-Петербург : Лань, 2023. – 200 с.;
8. Васильева М. А. Система контроля версий. Основы командной разработки : учебное пособие для вузов / М. А. Васильева, К. М. Филипченко. – Санкт-Петербург : Лань, 2022. – 144 с.;
9. Заяц А. М. Введение в гибридные технологии разработки мобильных приложений : учебное пособие для СПО / А. М. Заяц, Н. П. Васильева. – 2-е изд., стер. – Санкт-Петербург : Лань, 2022. – 160 с.;
10. Васильев Н. П. Введение в гибридные технологии разработки мобильных приложений : учебное пособие для вузов / Н. П. Васильев, А. М. Заяц. – 3-е изд., стер. – Санкт-Петербург : Лань, 2022. – 160 с.;
11. Старолетов С. М. Основы тестирования программного обеспечения : учебное пособие для СПО / С. М. Старолетов. – 3-е изд., стер. – Санкт-Петербург : Лань, 2023. – 192 с.;
12. Старолетов С. М. Основы тестирования и верификации программного обеспечения : учебное пособие для вузов / С. М. Старолетов. – 3-е изд., стер. – Санкт-Петербург : Лань, 2023. – 344 с.;
13. ГОСТ Р 58412-2019 «Защита информации»;
14. ГОСТ Р ИСО/МЭК 25010-2015 «Системная и программная инженерия»;
15. ГОСТ 34.602-2020 «Комплекс стандартов на автоматизированные системы».