**Содержание**

[Введение 4](#_Toc168334682)

[1. Аналитическая часть 5](#_Toc168334683)

[1.1. Интервью с заказчиком 5](#_Toc168334684)

[1.2. Описание предметной области 7](#_Toc168334685)

[1.3. Описание вариантов использования приложения 9](#_Toc168334686)

[1.4. Классификация пользователей результатов проекта 10](#_Toc168334687)

[1.5. Обоснование выбора средств разработки 11](#_Toc168334688)

[1.6. Описание структуры данных 12](#_Toc168334689)

[1.7. Описание архитектуры приложения 13](#_Toc168334690)

[1.8. Проектирование интерфейса 15](#_Toc168334691)

[2 Технологическая часть 18](#_Toc168334692)

[2.1 Организация данных 18](#_Toc168334693)

[2.2 Разработка дизайна приложения 19](#_Toc168334694)

[2.3 Разработка логики приложения 36](#_Toc168334695)

[2.4 Разработка API 41](#_Toc168334696)

[2.5 Протоколы тестирования 43](#_Toc168334697)

[2.6 Разграничение прав доступа 45](#_Toc168334698)

[3.Экономическая часть 46](#_Toc168334699)

[3.1 Описание качества программного продукта 46](#_Toc168334700)

[3.2 Управление процессом разработки 47](#_Toc168334701)

[3.3 Предварительная оценка трудоемкости разработки методом PERT 47](#_Toc168334702)

[Заключение 51](#_Toc168334703)

[Использованные источники информации 52](#_Toc168334704)

## Введение

В связи с распространением смартфонов среди студентов и преподавателей появляется возможность проверять знания студентов в удобном формате и без использования компьютерных классов.

Тема актуальна в связи с распространением смартфонов и доступностью интернета, образуется возможность проводить тестирование на предмет знаний студентов по выбранным дисциплинам.

Целью дипломного проектирования является создание мобильного приложения для составления и проведения тестирования по преподаваемым дисциплинам среди студентов.

Предметом дипломного проектирования является разработка мобильного приложения.

Объектом исследования является тестирование студентов.

Для достижения цели нужно выполнить следующие задачи:

1. Анализировать предметную область;

2. Реализовать пользовательский интерфейс;

3. Реализовать возможности тестирования;

При разработке мобильного приложения были выявлены следующие

функциональные требования:

1. Регистрация и авторизация пользователя;

2. Возможность присваивания роли пользователю;

3. Возможность составлять тестирования;

4. Возможность прохождение тестирования;

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. Каковы общие цели и назначение этого приложения для тестирования студентов?

* Целью и назначением данного приложения для тестирования студентов является оценка знаний и навыков студентов по определенной предметной области.

1. Какие основные функциональные возможности вы ожидаете от приложения для тестирования студентов?

* Создание тестовых заданий.
* Проведение тестирования
* Анализ результатов:
* Персонализация и адаптивность:
* Управление и организация:
* Хранение и доступность данных:

1. Какие типы вопросов и форматы тестов вам необходимы?

При создании данного приложения будут реализованы следующие типы тестов:

* множественный выбор,
* соответствие,
* открытый вопрос

1. Нужно ли вам возможность генерировать случайные тесты или адаптировать их для конкретных групп студентов?

Нет, поскольку мы реализуем приложение, которое позволит тестировать студентов преподавателям по предметам, которые припадает отдельный преподаватель

1. Какую информацию вы хотели бы видеть в отчетах по результатам тестирования студентов?

Результат в процентах, выставление примерной оценки

1. Какие требования по безопасности и доступу к данным вы имеете в виду?

* Уровень доступа: разграничение доступа между администраторами, преподавателями и студентами. Каждая группа пользователей должна иметь доступ только к необходимым функциональностям и данным.
* Аутентификация и авторизация: требуется обеспечить безопасный и надежный механизм аутентификации для всех пользователей приложения. Также необходимо реализовать систему авторизации, удостоверяющую правильность и разрешение каждого доступа к данным.

## 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 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:

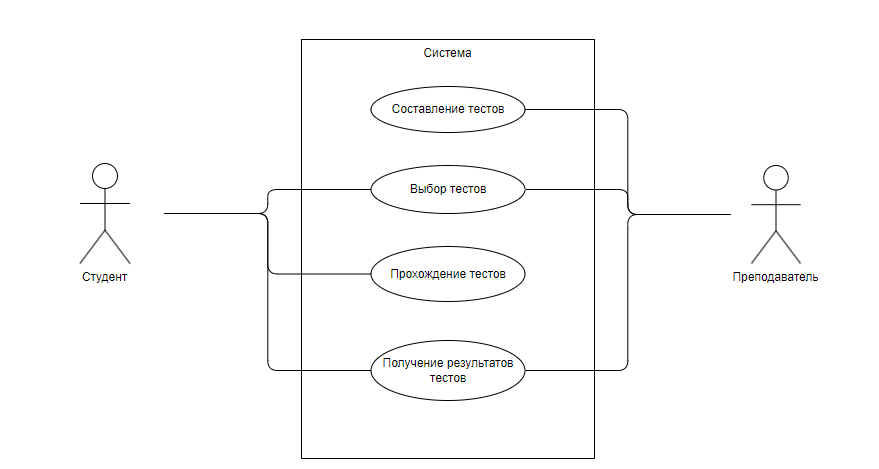


Рисунок 1. Use-case диаграмма.

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: Регистрация/ Авторизация пользователя |
| Результаты (постусловия) | Система выводит список вопросов с варинтами ответа пользователю студенту |

## 1.3. Описание вариантов использования приложения

На основе выявленных прецедентов в Use-case диаграмме был составлены основные успешные сценарии, представленный в таблице 2-3:

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 пользователям. |

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.

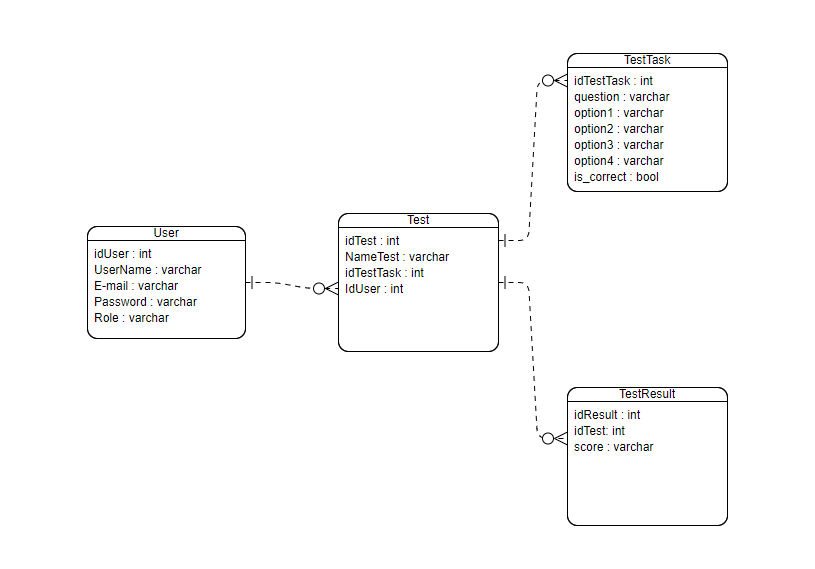


Рисунок 2 - ER – Диаграмма

Данная структура данных позволяет хранить информацию о студентах, тестах, вопросах, ответах и оценках, а также обеспечивает связи между этими сущностями. Это дает возможность эффективно управлять процессом тестирования студентов и анализировать его результаты.

## 1.7. Описание архитектуры приложения

Архитектура клиент-сервер-база данных представляет собой распределенную модель, в которой клиент (пользовательское приложение) взаимодействует с сервером, а сервер, в свою очередь, взаимодействует с базой данных. Вот основные компоненты этой архитектуры:

1. Клиент:

* Пользовательское приложение, такое как веб-браузер или мобильное приложение.
* Отвечает за отображение интерфейса пользователя и обработку пользовательских запросов.
* Отправляет запросы на сервер.

1. Сервер:

* Программное обеспечение, работающее на удаленном сервере.
* Получает запросы от клиента, обрабатывает их и взаимодействует с базой данных.
* Возвращает обработанные данные обратно клиенту.

1. База данных:

* Хранилище данных, обычно реляционная СУБД (SQL) или NoSQL база данных.
* Сервер взаимодействует с базой данных для хранения, извлечения и обновления данных.

Взаимодействие между компонентами:

1. Клиент отправляет запрос на сервер, например, запрос на получение списка продуктов.
2. Сервер получает этот запрос, обрабатывает его и отправляет запрос к базе данных для получения необходимых данных.
3. База данных возвращает данные серверу.
4. Сервер обрабатывает полученные данные и отправляет ответ обратно клиенту.
5. Клиент получает ответ от сервера и отображает данные пользователю.

Преимущества этой архитектуры:

- Разделение ответственности: клиент, сервер и база данных выполняют разные задачи, что повышает гибкость и масштабируемость системы.

- Безопасность: пользовательские данные хранятся на сервере, а не на клиенте, что обеспечивает большую защиту.

- Управление данными: база данных отвечает за хранение и управление данными, что позволяет легко масштабировать хранилище.

- Централизованная обработка: сервер обрабатывает запросы от множества клиентов, что повышает производительность.

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

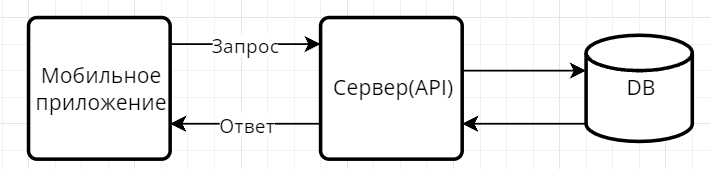


Рисунок 3 - Схема архитектуры приложения

Описание архитектуры показывает, что приложение разделено на ключевые компоненты, которые взаимодействуют друг с другом через четко определенные интерфейсы. Это позволяет поддерживать гибкость и масштабируемость системы.

## 1.8. Проектирование интерфейса

При анализе предметной области был спроектирован проект интерфейса приложения, представленный на рисунках 4-12. Каждая форма выполняет определенную функцию.

|  |  |
| --- | --- |
| Рисунок 4 - Форма “Авторизация” | Рисунок 5 - Форма “Регистрация” |
| Рисунок 6 - Окно “Выбор теста” | Рисунок 7 - Окно “Вопросы |

|  |  |
| --- | --- |
| Рисунок 8 - Окно “Результаты тестирования для студента” | Рисунок 9 - Окно “Результата тестирования для преподавателя” |
| Рисунок 10 - Окно “Результаты тестирования для студента” | Рисунок 11 - Окно “Результата тестирования для преподавателя” |

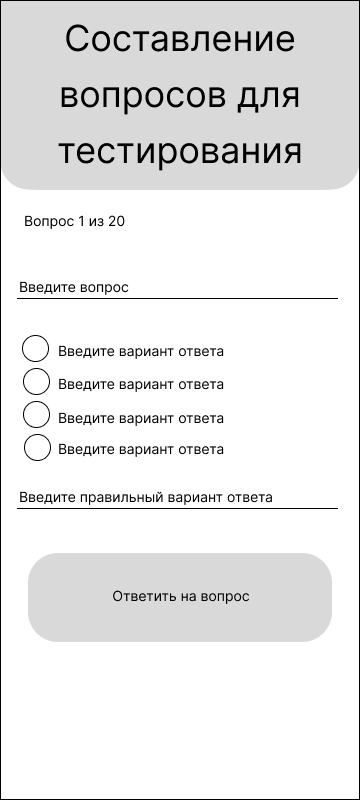


Рисунок 12. Окно “Составление вопросов для тестирования”

После создания проекта интерфейса была создана схема перемещений по приложению, представленная на рисунке 13:

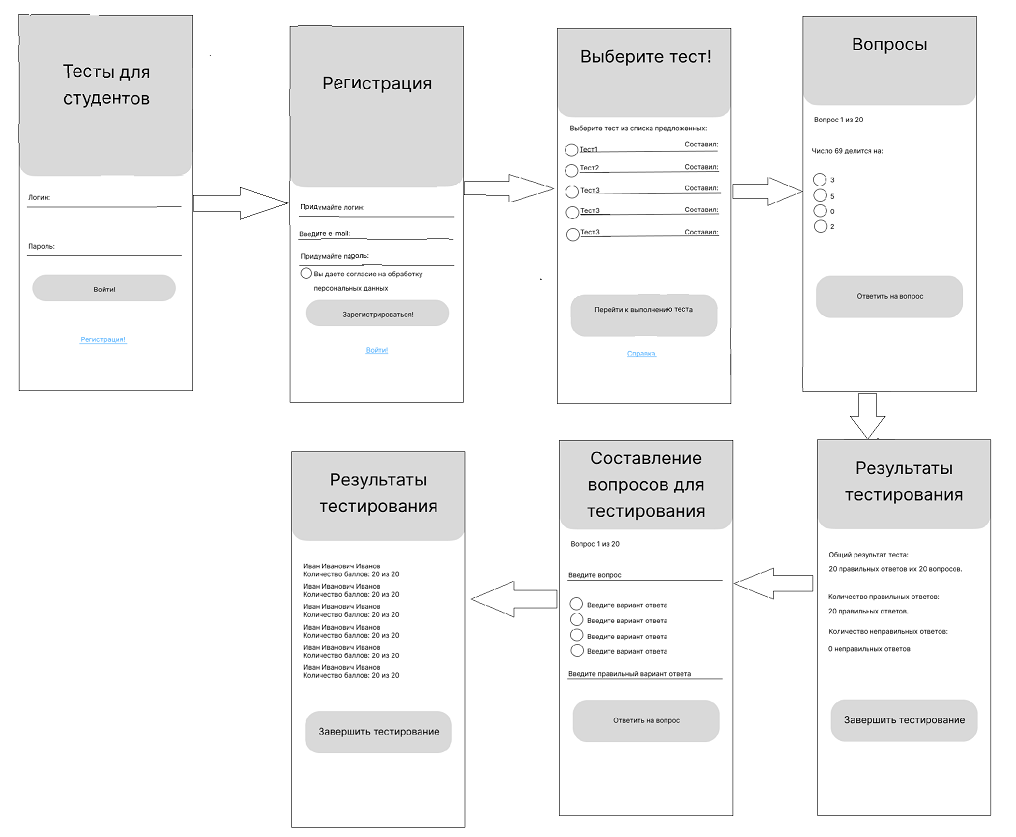


Рисунок 13. Окно “ Схема перемещений по приложению”

На рисунке ясно видно, как происходит работа в приложени

## 2 Технологическая часть

## 2.1 Организация данных

Организация данных в приложении “Тестирование студентов ”играет важную роль в эффективной работе приложения. На следующем рисунке представлена схема базы данных реализованная в Supabase:

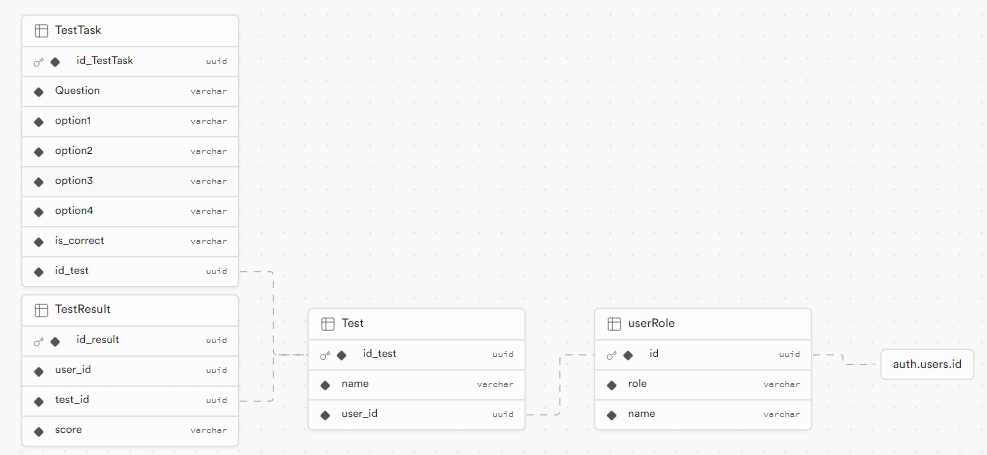


Рисунок 14 – Схема базы данных

Данная структура базы данных позволяет эффективно хранить и управлять всеми необходимыми данными для функционирования приложения. Она обеспечивает целостность, безопасность и масштабируемость системы, а также предоставляет удобный интерфейс для работы с информацией.

## 2.2 Разработка дизайна приложения

Разработка интерфейса мобильного приложения для тестирования студентов требует внимательного планирования и проектирования. При разработке было реализовано семь окон интерфейса представленные на рисунках 13-9.

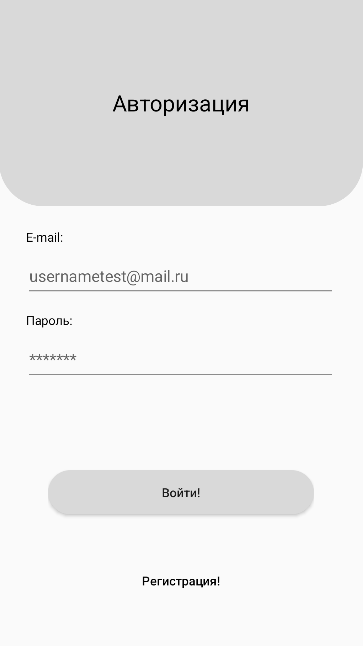


Рисунок 13 – Страница авторизации в приложении

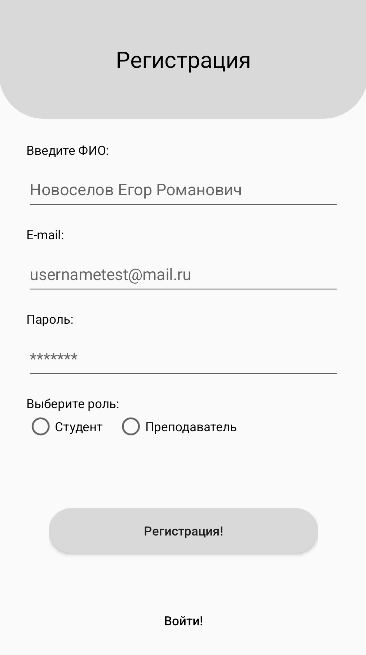


Рисунок 14 – Окно регистрации в приложении

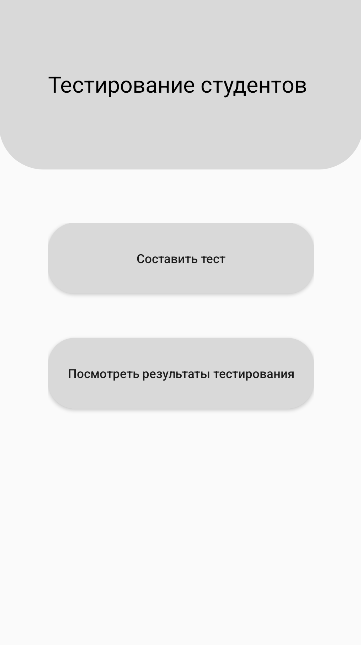


Рисунок 15 – Окно выбора действия для преподавателя

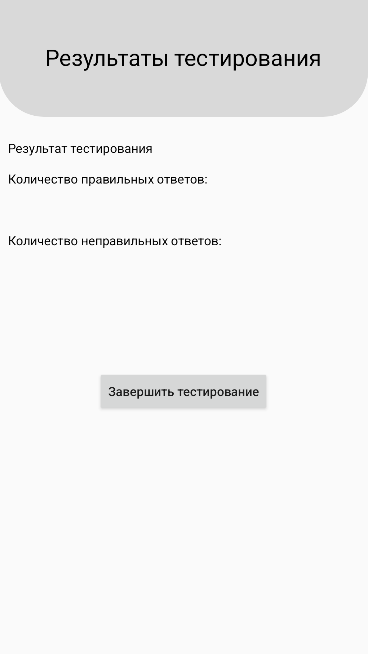


Рисунок 16 – Экран результатов тестирования для студента



Рисунок 17 – Экран создания тестирования

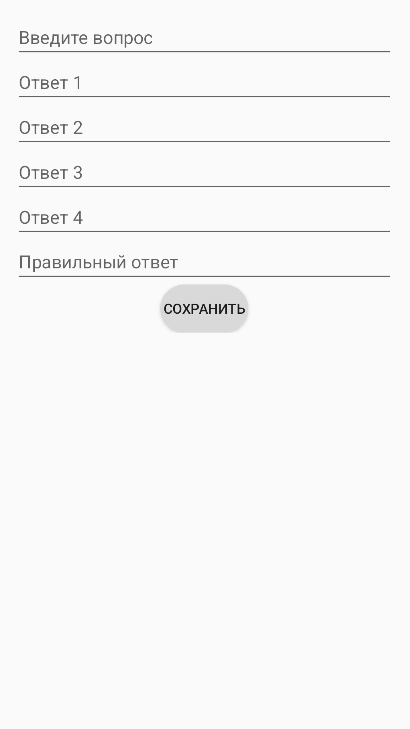
****

Рисунок 18 – Вспомогательный экран создания тестирования

Для реализации окна тестирования было создано вспомогательное окно, с помощью которого можно генерировать неограниченное количество вопросов.

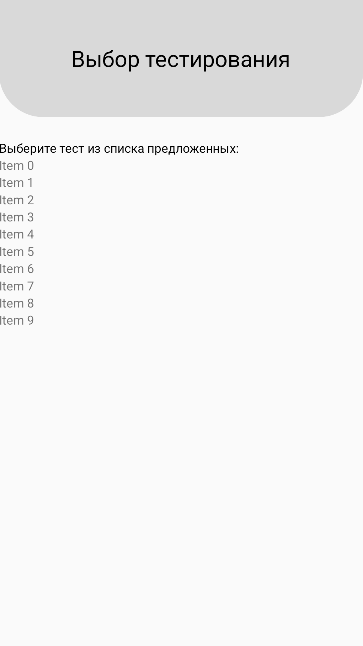
**

Рисунок 19 – Выбор теста студентом

## 2.3 Разработка логики приложения

В данном разделе будет рассмотрена разработка логики приложения, которое представляет собой разрабатываемое мобильное приложение предназначенное для организации и проведения тестирования студентов в образовательных учреждениях. Основной целью приложения является автоматизация процесса тестирования, что позволит повысить эффективность и объективность оценки знаний студентов.

Одним из ключевых компонентов разрабатываемого приложения является модуль авторизации/регистрации пользователя пользователей. Данный модуль отвечает за аутентификацию/регистрацию пользователей. Для реализации авторизации/регистрации требуется валидация вводимых значение на edittext. Рассмотрим валидацию email, представленный в листинг 8:

Листинг 8 – Валидация email при регистрации/авторизации

edit\_email.addTextChangedListener(object: TextWatcher {  
 override fun beforeTextChanged(p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int) {  
 btnLog.isEnabled=false  
 }  
 override fun onTextChanged(p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int) {  
 if (p0.isValidEmail()){  
 edit\_email.error = null  
 btnLog.isEnabled=true  
 }else{  
 edit\_email.error = "Введите E-mail."  
 btnLog.isEnabled=false  
 }  
 }  
 override fun afterTextChanged(p0: Editable?) {  
 mail = edit\_email.text.toString()  
 }  
 })

//Функция проверяющая валидность e-mail адреса  
 fun CharSequence?.isValidEmail():Boolean{  
 return !isNullOrEmpty() && Patterns  
 .EMAIL\_ADDRESS.matcher(this).matches()

Для реализации авторизации/регистрации также требуется валидация вводимых значение на edittext. Рассмотрим валидацию пароля, представленный в листинг 9

Листинг 9 – Валидация пароля при регистрации/авторизации

password\_con\_edit.addTextChangedListener(object: TextWatcher {  
 override fun beforeTextChanged(p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int) {  
 }  
 override fun onTextChanged(p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int) {  
 if (p0 != null) {  
 if (p0.trim().length>0){  
  
 password\_edit.error = null  
 if (p0.trim().length>6){  
 password\_con\_edit.error = null  
 }else {  
 password\_con\_edit.error = "Пароль меньше 6 символов."  
 }  
 }else{  
 password\_con\_edit.error = "Нужно подтвердить пароль"  
 }  
 }  
 }  
 override fun afterTextChanged(p0: Editable?) {  
 pass\_con = password\_con\_edit.text.toString()  
 if (pass == pass\_con){  
 btnReg.isEnabled = true  
 }else{  
 password\_con\_edit.error = "Пароли не совпадают"  
 btnReg.isEnabled = false  
 }  
 }  
})

Также компонент разрабатываемого приложения является модуль управления тестами. Данный модуль предоставляет преподавателям возможность создавать тестовые задания для оценки знаний студентов. Для добавления теста и вопросов была разработана функция добавления вопросов, представленная на листинге 10.

Листинг 10 – Создание тестирования:

private fun addQuestion() {  
 val questionView = layoutInflater.inflate(R.layout.question\_view, questionsContainer, false)  
 val questionEditText = questionView.findViewById<EditText>(R.id.questionText)  
 val answer1EditText = questionView.findViewById<EditText>(R.id.answer1)  
 val answer2EditText = questionView.findViewById<EditText>(R.id.answer2)  
 val answer3EditText = questionView.findViewById<EditText>(R.id.answer3)  
 val answer4EditText = questionView.findViewById<EditText>(R.id.answer4)  
 val correctAnswerEditText = questionView.findViewById<EditText>(R.id.CorrectAnswerEdit)  
 questionEditText.setText("")  
 answer1EditText.setText("")  
 answer2EditText.setText("")  
 answer3EditText.setText("")  
 answer4EditText.setText("")  
 correctAnswerEditText.setText("")  
 questionsContainer.addView(questionView)  
 }

Для сохранения теста и вопросов была разработана функция сохранения вопросов тестирования, представленная на листинге 11.

Листинг 11 – Сохранение тестирования:

private fun saveTest() {  
 val questions = mutableListOf<Question>()  
 for (i in 0 until questionsContainer.childCount) {  
 val questionView = questionsContainer.getChildAt(i)  
 val question = getQuestionFromView(questionView)  
 questions.add(question)  
 }  
 Toast.makeText(this, "Test saved", Toast.LENGTH\_SHORT).show()  
 }

Для сохранения теста и вопросов была разработана функция получения данных о вопросах тестирования, представленная на листинге 12

Листинг 11 – Получения данных о тестировании:

private fun getQuestionFromView(questionView: View): Question {  
 val questionText = questionView.findViewById<EditText>(R.id.questionText).text.toString()  
 val answer1 = questionView.findViewById<EditText>(R.id.answer1).text.toString()  
 val answer2 = questionView.findViewById<EditText>(R.id.answer2).text.toString()  
 val answer3 = questionView.findViewById<EditText>(R.id.answer3).text.toString()  
 val answer4 = questionView.findViewById<EditText>(R.id.answer4).text.toString()  
 val correctAnswer = questionView.findViewById<EditText>(R.id.CorrectAnswerEdit).text.toString()  
 val question = getQuestionFromView(questionView)  
 uploadQuestionToSupabase(question)

}

## 2.4 Разработка API

При разработке 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. Для использования Supabase нужно имплементировать библиотеку. Имплементация библиотеки Supabase, представлена в листинг 1.

Листинг 12 – имплементирование библиотеки Supabase:

implementation(platform("io.github.jan-tennert.supabase:bom:2.4.2"))  
implementation ("io.github.jan-tennert.supabase:postgrest-kt")  
implementation ("io.github.jan-tennert.supabase:realtime-kt")

Для использования Supabase нужно имплиментировать K-tor. K-tor - это библиотека для создания серверных веб-приложений на Kotlin. Она предназначена для разработки современных высокопроизводительных и масштабируемых веб-сервисов и API. Имплементирование K-tor представлено в листинг 2.

Листинг 13 – имплементирование K-tor:

implementation("io.ktor:ktor-client-cio:2.3.11")

Также для передачи данных нужно использовать сериализацию. Сериализация в программировании - это процесс преобразования объекта в поток байтов, который может быть сохранен в хранилище или передан по сети. Листинг добавления сериализации представлена в листинге 3.

Листинг 14 – имплементирование сериализации:

kotlin("plugin.serialization") version "2.0.0-RC3"

Для реализации функциональных требований были реализованы запросы: регистрация, авторизация, извлечения данных, вставки данных, обновление данных. Данные запросы представленные в листингах 4-5:

Листинг 15 запрос регистрации:

lifecycleScope.launch {  
 val user = supabase.auth.signUpWith(Email) {  
 email = mail.toString()  
 password = pass.toString()  
 }}

Листинг 16 – запрос авторизации:

lifecycleScope.launch {  
 supabase.auth.signInWith(Email) {  
 email = mail.toString()  
 password = pass.toString()  
 }}

Метод возвращение пользователя позволяет получить объект user из текущего сеанса для получения самой последней пользовательской информации. Объект user извлекается из базы данных, а не из локального сеанса. Метод представлен в листинг 6

Листинг 17 – возвращение пользователя:

lifecycleScope.launch {  
 val user = supabase.auth.retrieveUserForCurrentSession(updateSession = true)

}

Листинг 18 – запрос извлечение данных:

lifecycleScope.launch {  
 var usersRole: userId? = supabase.from("userRole").select().decodeSingle<userId>()  
}

Листинг 19 – запрос вставки данных:

lifecycleScope.launch {  
 val toUpsert = Message(id = userID.toString(), role = selectedRole.toString(), name = username.toString())  
 supabase.from("userRole").upsert(toUpsert)  
}

## 2.5 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. |

## 

## 2.6 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.

## 3. Экономическая часть

## 3.1 Описание качества программного продукта

Данный программный продукт разработан на основе созданной базы данных.

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 Предварительная оценка трудоемкости разработки методом PERT

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 | 2 | 4 | 6 | 3 |
| Выбор тестов | 2 | 4 | 6 | 3 |
| Прохождение тестов | 2 | 4 | 5 | 3 |
| Получение результатов тестов | 2 | 4 | 8 | 3 |
| **Итого** | **8** | **16** | **25** | **12** |

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.

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

(3)

Суммарная трудоёмкость проекта рассчитывается по формуле 4.

(4)

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

(5)

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

В ходе выполнения дипломной работы был разработан прототип мобильного приложения "Тестирование студентов" на основе Android Studio, Kotlin, Supabase и Visual Paradigm.

Использование Android Studio в качестве интегрированной среды разработки позволило эффективно создавать и тестировать приложение для платформы Android. Выбор языка Kotlin обеспечил современный, лаконичный и безопасный подход к программированию мобильных приложений.

Интеграция с Supabase, платформой для разработки серверной части приложений, упростила процесс создания и управления базой данных, аутентификацией пользователей, а также обеспечила надежное хранение и обработку данных.

Использование Visual Paradigm на этапе проектирования помогло визуализировать архитектуру приложения, создать детальные диаграммы классов, вариантов использования и потоков данных. Это способствовало более структурированному и продуманному подходу к разработке.

В ходе работы были успешно реализованы основные функциональные возможности приложения, включая регистрацию и аутентификацию пользователей, создание и прохождение тестов, просмотр результатов тестирования. Приложение обладает интуитивно понятным интерфейсом и отвечает современным требованиям к мобильным приложениям.

Таким образом, результатом выполнения дипломной работы стала разработка работоспособного прототипа мобильного приложения "Тестирование студентов", который может служить основой для дальнейшего развития и совершенствования системы..

# 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. ГОСТ Р 58412-2019 «Защита информации»;
4. ГОСТ Р ИСО/МЭК 25010-2015 «Системная и программная инженерия»;
5. ГОСТ 34.602-2020 «Комплекс стандартов на автоматизированные системы».
6. Мамедли Р. Э. Базы данных. Лабораторный практикум : учебное пособие для вузов / Р. Э. Мамедли. – Санкт-Петербург : Лань, 2023. – 152 с.;
7. Нестеров С. А. Интеллектуальный анализ данных с использованием SQL Server : учебник для вузов / С. А. Нестеров. – Санкт-Петербург : Лань, 2023. – 160 с.;
8. 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.;
9. 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.;
10. 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.;
11. 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.;
12. 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.;
13. 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.;
14. Старолетов С. М. Основы тестирования программного обеспечения : учебное пособие для СПО / С. М. Старолетов. – 3-е изд., стер. – Санкт-Петербург : Лань, 2023. – 192 с.;
15. 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.;