Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Головной учебно-исследовательский и методический центр профессиональной реабилитации лиц с ограниченными возможностями здоровья (инвалидов)

КАФЕДРА Системы обработки информации и управления

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:
Разработка мобильного приложения для ГУИМЦ

Студент ИУ5Ц-102Б _____ М.В. Яровенко
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКРБ _____ К.С. Мышенков
(Подпись, дата) (И.О.Фамилия)

Нормоконтролер _____ Ю.Н. Кротов
(Подпись, дата) (И.О.Фамилия)

2024 г.

РЕФЕРАТ

РПЗ 74 с., 9 рис., 5 табл., 15 источн., 3 прил.

ПРОЕКТИРОВАНИЕ, РАЗРАБОТКА, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, МОДЕЛЬ, АЛГОРИТМЫ, ANDROID, KOTLIN.

Объектом разработки является мобильное приложение для ГУИМЦ.

Цель работы состоит в упрощении образовательного процесса студентам ГУИМЦ путем создания мобильного приложения для ГУИМЦ, которое предоставляет важную информацию студентам, в том числе предоставляет расписание учебной группы и возможность подать заявки сотрудникам факультета.

В процессе выполнения квалификационной работы бакалавра было разработано мобильное приложение для ГУИМЦ. Для этого было произведено описание и исследование предметной области, включающей в себя функционирование факультета ГУИМЦ МГТУ им Н.Э. Баумана. Были обозначены функциональные требования системы, который включают в себя предоставление персональных данных авторизированным студентам, информации о расписании учебной группы и возможности подать заявку. Был произведен анализ существующих аналогов, а также были выбраны технологии, которые будут использованы при создании программного продукта. Помимо этого, была описана архитектура, на которой строится приложение. Спроектирована структура базы данных, которая будет использоваться в проектируемом мобильном приложении. Также после разработки программного продукта было произведено его описание и тестирование.

Графическая часть РПЗ представлена в приложении А. Техническое задание на ВКРБ приведено в приложении Б. Программа и методика испытаний ВКРБ приведена в приложении В.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 5
ВВЕДЕНИЕ 6
1 ПОСТАНОВКА ЗАДАЧ РАЗРАБОТКИ 8
1.1 Описание предметной области 8
1.2 Постановка задач системы 9
1.3 Функциональные требования 9
1.4 Анализ существующих аналогов 10
1.5 Обзор используемых технологий для реализации продукта 11
1.5.1 Выбор операционной системы 11
1.5.2 Выбор языка программирования 12
1.5.3 Выбор среды разработки 14
1.6 Выводы по первой главе 18
2 КОНСТРУКТОРСКО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 20
2.1 Разработка приложения 20
2.1.1 Разработка модели базы данных 20
2.1.2 Разработка архитектуры приложения 25
2.1.2.1 Макеты 26
2.1.2.2 Фрагменты и навигация 29
2.1.2.3 Меню Anroid-приложений 30
2.1.2.4 Модели представлений и фабрики моделей представлений 31
2.1.2.5 Связывание данных 33
2.1.2.6 База данных Room 34
2.1.2.7 Управление зависимостями 37
2.1.3 Описание и тестирование функционала приложения 38
2.2 Выводы по второй главе 44
ЗАКЛЮЧЕНИЕ 45
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 47
ПРИЛОЖЕНИЕ А ГРАФИЧЕСКАЯ ЧАСТЬ 49

ПРИЛОЖЕНИЕ Б ТЕХНИЧЕСКОЕ ЗАДАНИЕ 62
ПРИЛОЖЕНИЕ В ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ 69

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей расчетно-пояснительной записке применяют следующие сокращения и обозначения:

– СУБД – система управления базами данных.

– Фреймворк – программная платформа, определяющая структуру программной системы, предоставляющая функционал для облегчения разработку и интеграции различных компонентов крупной системы.

– БД – база данных.

– ГУИМЦ – Головной учебно-исследовательский и методический центр.

– UI – User interface.

– МГТУ – Московский государственный технический университет.

– ОС – Операционная система.

– KMM – Kotlin Multiplatform Mobile.

– SDK - Software development kit.

– IDE – Integrated development environment.

ВВЕДЕНИЕ

В современном мире технологии играют важную роль в нашей повседневной жизни. Мы используем компьютеры, умные устройства и иную технику во многих областях жизнедеятельности. В частности, самым распространённым устройством, которое использует большинство, является смартфон. Сегодня он есть практически у каждого человека. Мобильные приложения не только упрощают выполнение множества задач, но и открывают новые возможности для коммуникации, обучения, развлечений и ведения бизнеса. От проверки электронной почты и социальных сетей до управления финансами и заказа товаров – мобильные приложения охватывают широкий спектр сфер деятельности, делая нашу жизнь удобнее и продуктивнее. Их значимость подтверждается растущим числом пользователей и постоянным развитием рынка, что подчеркивает их важность в современном обществе.

Обратим свое внимание на широкий спектр возможностей мобильного устройства в сфере образования. В наше время обучение в высших заведениях невозможно без цифровых технологий. Так, например, в МГТУ имени Н.Э. Баумана учебный процесс построен на использовании такой электронной инфраструктуры, как Электронный Университет, электронная почта Samoware для студентов и сотрудников, сайт университета, сайт библиотеки МГТУ имени Н.Э. Баумана, сайты кафедр университета и многое другое. Эти ресурсы востребованы и в течение дня к ним обращаются тысячи студентов и сотрудников МГТУ имени Н.Э. Баумана. И принимая во внимание распространенность мобильных устройств, разработка мобильного приложения для того или иного цифрового ресурса выглядит актуально и востребовано.

В данной работе рассматриваемой предметной областью является факультет ГУИМЦ МГТУ имени Н.Э. Баумана. Он имеет собственные цифровые ресурсы, такие как сайт и каналы в социальных сетях, которые используются в учебном процессе как студентами, так и сотрудниками. Но на момент написания данной работы у факультета ГУИМЦ отсутствовали какие-либо приложения на

мобильных устройствах, которые бы позволили упростить учебу студентам и их взаимодействие с сотрудниками факультета. В связи с этим возникает потребность в разработке мобильного приложения, функционал которого будет заточен по учебную деятельность рассматриваемого факультета.

В рамках выпускной квалифицированной работы ставится задача разработки мобильного приложения для ГУИМЦ под операционную систему Android (как самую распространенную и доступную). Приложение должно предоставлять авторизованному пользователю информацию о нем, о его учебной группе, о текущем расписании студента, а также давать возможность оформить студентом заявку для сотрудников по тому или иному вопросу. Вся информация, которая обрабатывается приложением должна будет храниться в базе данных.

Таким образом, все вышеперечисленное подчеркивает актуальность разработки мобильного приложения для ГУИМЦ. Такое приложение будет полезно для всех студентов факультета, а также упростит рабочий процесс и его сотрудникам.

1 ПОСТАНОВКА ЗАДАЧ РАЗРАБОТКИ

1.1 Описание предметной области

Предметной областью разрабатываемого мобильного приложения является факультет ГУИМЦ МГТУ имени Н.Э. Баумана. На этом факультете обучаются студенты, каждый из которых входит в определенную учебную группу, которая имеет собственное расписание образовательного процесса.

Рассмотрим структуру расписания более подробно. Образовательный процесс осуществляется в течение рабочей шестидневной недели. Самая первая пара может начинаться в 8:30, а самая последняя – в 19:10. Недели разделяются на числитель и знаменатель и чередуются в течение семестра. Сами занятия подразделяются на семинары, лекции и лабораторные работы.

Также факультет хранит и обрабатывает персональные данные своих студентов. Эта информация может включать в себя данные о гражданстве студента, о статусе военного обучения, о статусе проживания в общежитии и другие сведения. Все данные, которые есть у факультета должны быть доступны студенту для ознакомления.

Все студенты, обучающиеся на факультете ГУИМЦ могут запросить различного рода документы, например, справку об обучении, и сотрудники университета должны её подготовить. Помимо прочего уникальной особенностью факультета ГУИМЦ является работа сурдопереводчиков для слабослышащих студентов. И каждый нуждающийся в сурдопереводе студент может попросить предоставить переводчика на учебное занятие.

Таким образом, мы рассмотрели основную функциональность факультета ГУИМЦ. Для упрощения учебного процесса многие функции предметной области будут реализованы в мобильном приложении. Это позволит упростить взаимодействие студентов с сотрудниками факультета, ведь с помощью приложения они могут узнавать всю необходимую информацию, в частности расписание учебной группы, а также оставлять заявки.

1.2 Постановка задач системы

Основная цель разработки заключается в создании мобильного приложения для факультета ГУИМЦ, которое упростит студентам решение многих организационных вопросов обучения.

Для достижения искомой цели необходимо выполнить следующие задачи:

– исследовать предметную область;
– провести анализ существующих аналогов;
– определить функциональные требования приложения;
– провести анализ средств для разработки приложения;
– разработать и реализовать способ хранения данных предметной области;
– разработать интерфейс пользователя приложения;
– разработать алгоритмы, требуемые для реализации приложения;
– разработать и реализовать программную часть приложения;
– провести отладку приложения.

1.3 Функциональные требования

Приложению выдвигаются следующие функциональные требования:

– осуществлять авторизацию пользователя при помощи логина и пароля;
– демонстрировать авторизованному пользователю его персональные данные;
– отображать расписание учебной группы пользователя;
– осуществлять запись в базу данных введенной пользователем заявки;
– выводить на экран список ранее введенных пользователем заявок;
– обрабатывать работу кнопки возврата телефонного меню и выход из приложения.

1.4 Анализ существующих аналогов

В настоящее время существует несколько мобильных приложений, которые помогают в организации учебного процесса. Рассмотрим некоторые из них.

1) *Мой МГТУ*. Мобильное приложение, которое разработано НОЦ Электронный университет МГТУ им. Н.Э. Баумана. Оно интегрировано в систему университета и позволяет авторизованным студентам получить доступ к своей успеваемости, пропуску, а также расписанию учебной группы. Существенным недостатком данного продукта для студентов факультета ГУИМЦ является тот факт, что данное приложение не учитывает в полной мере существование факультета ГУИМЦ МГТУ имени Н.Э. Баумана. Это проявляется в невозможности студентом ГУИМЦ при просмотре расписания выбрать свой факультет и учебную группу данного факультета.

У других университетов также есть мобильные приложения подобные приложению «Мой МГТУ», их функционал подобен рассмотренному.

2) *Расписание для студентов*. Мобильное приложение позволяет любому пользователю гибко настроить расписание учебного процесса, которое в последствие будет отображаться при использовании приложения. Но данное приложение независимо и не интегрировано в систему университета, что является существенным недостатком, так как все изменения в расписании пользователю необходимо менять вручную.

Таким образом, видно, что на рынке мобильных приложений отсутствует такой продукт, который бы полностью отвечал запросам студентов факультета ГУИМЦ МГТУ имени Н.Э. Баумана. Существующие приложения не полностью удовлетворяют потребностям студентов ГУИМЦ, хотя и у таких приложений есть свои сильные положительные стороны. Принимая все это во внимание, приходим к выводу, что наилучшим продуктом для целевой аудитории (студенты ГУИМЦ) будет разработанное в ходе данной работы мобильное приложение для ГУИМЦ.

1.5 Обзор используемых технологий для реализации продукта

1.5.1 Выбор операционной системы

Перед разработкой мобильного приложения необходимо определить какие технологии будет наиболее выгодно использовать для достижения результата. Первым важным вопросом, который необходимо решить – это выбор операционной системы, под которую будет разрабатываться мобильное приложение.

На май 2024 года рынок мобильных операционных систем по всему миру представляет собой следующую картину: 71,5% приходится на ОС Android, 27,73% занимает ОС iOS, а оставшиеся доли занимают менее популярные операционные системы [1]. При таком раскладе выбор следует осуществить между ОС Android и ОС iOS. Проведем сравнение основных характеристик операционных систем Android и iOS, заполнив таблицу 1.

Таблица 1 – Сравнение ОС Android и iOS

Характеристика	Android	iOS
Разработчик	Google	Apple
Открытость системы	Открытая (с элементами закрытости)	Закрытая
Устройства	Множество производителей	Только Apple (iPhone, iPad)
Персонализация	Высокая	Ограниченная
Магазин приложений	Google Play Store	Apple App Store
Контроль качества приложений	Меньше контроля	Строгий контроль
Интеграция с другими устройствами	Хорошая (особенно с устройствами Google)	Отличная (с устройствами Apple)
Ценовой диапазон устройств	От бюджетных до флагманских	В основном флагманские
Обновления ОС	Зависит от производителя устройства	Регулярные, для всех устройств одновременно
Документация и ресурсы	Обширная документация и ресурсы	Ограниченные ресурсы для разработчиков

Из данных таблицы видно, что ОС iOS выигрывает с точки зрения контроля качества и безопасности, но уступает ОС Android в возможностях разработчика и распространенности устройств. Таким образом, считая более важными критериями для разрабатываемого мобильного приложения для ГУИМЦ доступность большему числу пользователей, а также более широкий спектр возможностей для разработчика, было принято решение использовать операционную систему Android.

1.5.2 Выбор языка программирования

Следующим важным вопросом после выбора операционной системы мобильного устройства становится выбор языка программирования, с помощью которого будет создаваться мобильное приложение. Для ОС Android самыми популярными языками мобильной разработки являются языки Java и Kotlin. Сравним их.

В начале обратимся к рейтингу языков программирования, чтобы понять насколько актуален тот или иной язык. Рассмотрим несколько источников.

1) TIOBE (данные за май 2024): Java — 4-е место, Kotlin — 19-е место. Считается по поисковым запросам в Google, YouTube, Bing и других посещаемых площадках [2].

2) PYPL (данные за май 2024): Java — 2-е место, Kotlin — 13-е место. Считается по поисковым запросам технической документации языков программирования в Google [3].

3) Stack Overflow (данные за май 2023): Java — 7-е место, Kotlin — 15-е место. Считается по результатам опроса разработчиков сообщества Stack Overflow Survey [4].

Из полученных данных видно, что Java популярнее Kotlin в среде разработчиков, но такая статистика объясняется тем, что язык Kotlin гораздо моложе. Но тем не менее он стремительно набирает популярность среди

разработчиков мобильных приложений. Это подтверждает тот факт, что уже в 2019 году Google сделал Kotlin основным языком под Android [5].

Сравним языки программирования Java и Kotlin между собой по различным аспектам.

– *Тип языка программирования*. Kotlin и Java оба являются статически типизированными языками программирования, что означает, что типы данных переменных должны быть определены до компиляции.

– *Парадигма*. Kotlin поддерживает как объектно-ориентированное, так и функциональное программирование, предоставляя разработчикам большую гибкость в выборе стиля кодирования. Java же традиционно считается объектно-ориентированным языком.

– *Синтаксис*. Синтаксис Kotlin более сжатый и выразительный по сравнению с Java, что позволяет уменьшить количество шаблонного кода и упростить чтение и написание кода.

– *Управление памятью*. Оба языка используют автоматическое управление памятью с помощью сборщика мусора, что уменьшает риск утечек памяти и упрощает управление ресурсами.

– *Нулевая безопасность*. Kotlin имеет встроенную поддержку для предотвращения ошибок NullPointerException благодаря системе типов, которая различает nullable и non-nullable типы. В Java же разработчики должны явно проверять объекты на null, чтобы избежать таких ошибок.

– *Совместимость*. Kotlin полностью совместим с Java, что позволяет использовать существующие Java-библиотеки и фреймворки без проблем. Java, в свою очередь, не поддерживает код Kotlin напрямую.

– *Кроссплатформенность*. Kotlin поддерживает многоплатформенную разработку через Kotlin Multiplatform, что позволяет создавать приложения для различных платформ, используя один и тот же код. Java же ограничена JVM и Java-платформами.

– *Инструменты разработки*. Kotlin тесно интегрирован с IntelliJ IDEA и Android Studio, что обеспечивает удобную разработку приложений, особенно для

Android. Java поддерживается в широком спектре IDE, включая Eclipse, NetBeans и IntelliJ IDEA, что делает ее доступной для большего числа разработчиков.

– *Сообщество и поддержка*. Сообщество Kotlin активно растет, и язык активно поддерживается компанией JetBrains, которая его создала. Java имеет огромное сообщество и обширную поддержку, благодаря чему доступно множество ресурсов и библиотек [6].

По итогам сравнения можно сказать, что нет категорических аспектов в пользу одного или другого языка, так что выбор между Java и Kotlin больше зависит от предпочтений разработчика. Отталкиваясь от более современных возможностей языка Kotlin и более простого синтаксиса в сравнении с языком Java было принято решение разрабатывать мобильное приложение для ГУИМЦ на языке программирования Kotlin.

1.5.3 Выбор среды разработки

Завершающим штрихом в выборе средств разработки будет анализ сред разработки под Android и выбор наилучшей из них.

При создании собственных приложений для Android выбор правильной среды разработки имеет решающее значение для обеспечения исключительного пользовательского опыта. Нативные приложения созданы специально для платформы Android и обеспечивают более высокую производительность, более простой доступ к функциям устройства и более плавные переходы в пользовательском интерфейсе [7]. Рассмотрим некоторые из лучших платформ разработки приложений для Android в 2024 году:

1) *Android Studio*. Android Studio — это официальная интегрированная среда разработки (IDE) для платформы Android, разработанная компанией Google. Она предоставляет разработчикам комплексный набор инструментов для создания приложений на Android, включая:

– *Редактор кода*: поддерживает языки программирования Kotlin, Java и C++, предлагая удобные функции, такие как автодополнение, рефакторинг и анализатор кода.

– *Эмулятор*: позволяет тестировать приложения в различных конфигурациях устройств и версиях Android без необходимости использования реального устройства.

– *Профилировщик производительности*: инструменты для анализа производительности приложений, включая использование CPU, памяти и сетевых запросов.

– *Система сборки Gradle*: автоматизирует процесс сборки приложений и управления зависимостями.

– *Android Virtual Device (AVD) Manager*: управляет конфигурациями виртуальных устройств для тестирования.

– *Layout Editor*: позволяет визуально создавать интерфейсы приложений, используя drag-and-drop компоненты.

– *Android SDK Manager*: управляет инструментами SDK, платформами и другими компонентами, необходимыми для разработки.

– *Support for Jetpack Compose*: поддержка нового фреймворка для создания нативных UI на Kotlin.

Android Studio постоянно обновляется и улучшается, предлагая разработчикам новые возможности и инструменты для упрощения процесса разработки и повышения качества конечных продуктов. Она тесно интегрирована с другими сервисами и инструментами Google, такими как Firebase и Google Cloud, что позволяет легко внедрять облачные сервисы и использовать мощные инструменты аналитики.

Для начала работы с Android Studio достаточно скачать её с официального сайта Android Developers и установить на поддерживаемую операционную систему, такую как Windows, macOS или Linux. После установки и настройки окружения разработчики могут приступать к созданию новых проектов и разработке приложений для Android [8].

2) *Kotlin Multiplatform Mobile (KMM)*. Kotlin Multiplatform Mobile (KMM) — это SDK, разработанный JetBrains, который позволяет разработчикам использовать Kotlin для создания кросс-платформенных мобильных приложений. С помощью KMM можно писать бизнес-логику приложения один раз на Kotlin и использовать её как на Android, так и на iOS, что значительно сокращает время и ресурсы, необходимые для разработки приложений на обе платформы.

Определим ключевые особенности KMM:

– *Общий код*: разработчики могут писать общую бизнес-логику на Kotlin, которая будет работать на обеих платформах, избегая дублирования кода.

– *Платформенно-специфический код*: для доступа к функциям, специфичным для каждой платформы, можно использовать ожидаемые/фактические декларации (expect/actual declarations) в Kotlin.

– *Интеграция с существующими проектами*: KMM можно интегрировать в существующие проекты Android и iOS, позволяя постепенно переносить части приложения на KMM.

– *Поддержка IDE*: JetBrains предоставляет плагин для Android Studio, который упрощает разработку KMM-проектов, включая создание новых проектов и отладку общего кода.

– *Совместимость с библиотеками*: KMM поддерживает использование многих популярных библиотек Kotlin и позволяет создавать собственные мультиплатформенные библиотеки.

KMM продолжает развиваться, и JetBrains активно работает над улучшением инструментов и поддержкой сообщества. Для разработчиков, которые хотят использовать Kotlin для создания кросс-платформенных приложений, KMM представляет собой мощный инструмент, который может упростить разработку и поддержку приложений. Однако, как и любая технология, KMM имеет свои ограничения и лучше всего подходит для проектов, где большая часть бизнес-логики может быть общей для обеих платформ [9].

3) *React Native*. React Native — это популярный фреймворк для разработки кросс-платформенных мобильных приложений, созданный Facebook. Он позволяет разработчикам использовать React, библиотеку JavaScript для построения пользовательских интерфейсов, в сочетании с нативными платформенными возможностями. Вот несколько ключевых особенностей React Native:

– *Кросс-платформенность*: код, написанный на React Native, может быть использован для создания приложений как для iOS, так и для Android, что уменьшает время разработки и упрощает поддержку.

– *Компонентный подход*: React Native использует компоненты React, что делает структуру приложения модульной и упрощает разработку.

– *Горячая перезагрузка (Hot Reloading)*: эта функция позволяет разработчикам видеть изменения в коде в реальном времени без необходимости пересобирать приложение.

– *Нативные модули и плагины*: если в приложении требуются специфические платформенные возможности, можно использовать нативные модули и плагины для доступа к API и функциям устройства.

– *Сообщество и экосистема*: React Native имеет большое и активное сообщество, а также обширную экосистему библиотек и инструментов.

React Native продолжает развиваться, и его сообщество регулярно вносит улучшения и новые возможности. Это делает его одним из предпочтительных выборов для разработчиков, стремящихся создать качественные кросс-платформенные мобильные приложения с использованием знакомых инструментов веб-разработки. Однако, как и любой инструмент, React Native имеет свои ограничения и может не подходить для всех типов проектов, особенно если требуется глубокая интеграция с платформенными возможностями или высокая производительность. В таких случаях может потребоваться использование нативной разработки или других кросс-платформенных решений [10].

Итак, мы рассмотрели некоторые из самых популярных сред разработки мобильных приложений. Учитывая, что создание мобильного приложения для ГУИМЦ заточено под конкретную платформу (Android) и отдавая предпочтение производительности, а не простоте программирования, было принято решение проводить разработку продукта с использованием Android Studio.

1.6 Выводы по первой главе

В первой главе была подробно рассмотрена предметная область разрабатываемого мобильного приложения в виде факультета ГУИМЦ МГТУ имени Н.Э. Баумана. Была проанализирована его работа и выделены те аспекты функциональной деятельности во время учебного процесса, которые необходимо реализовать в создаваемом продукте.

На основе предметной области были сформулированы цели разработки и задачи, которые необходимо выполнить в процессе создания приложения. Цель направлена на создание такого продукта, который облегчит студентам решение многих организационных вопросов обучения. А среди поставленных задач можно выделить следующие: описание предметной области, проведение анализа существующих аналогов, определение функциональных требований, разработка алгоритмов работы приложения и другие.

Затем были отражены основные функциональные требования к мобильному приложению, которые описывают функции необходимые для полноценного функционирования конечного продукта. К важным из них относятся такие функции как: авторизация пользователя по заранее полученному от сотрудников факультета логину и паролю, отображение расписания учебной группы пользователя с учетом знаменателя и числителя, возможность занести пользователем в базу данных нужную заявку – а также отмечена иная функциональность приложения.

Также в этой главе был проведен анализ существующих аналогов среди мобильных приложений под операционную систему Android. По его итогам

выявлены недостатки представленных продуктов в виде отсутствия гибких настроек их функциональности под работу факультета ГУИМЦ МГТУ имени Н.Э. Баумана. Другими словами, была продемонстрирована необходимость в создании собственного продукта, заточенного под работу рассматриваемой предметной области.

И под конец, были проанализированы технологии, которые будут задействованы в создании мобильного приложения для ГУИМЦ. Операционной системой, под которую будет создаваться конечный продукт, стала ОС Android, а языком программирования мобильной разработки был выбран язык Kotlin. Таким образом, осуществлена вся необходимая деятельность перед началом разработки приложения.

2 КОНСТРУКТОРСКО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

2.1 Разработка приложения

2.1.1 Разработка модели базы данных

Прежде чем разрабатывать архитектуру мобильного приложения, необходимо подготовить базу данных, структура которой будет реализовать функционал для хранения информации об студенте, расписании учебной группы, о созданных студентом заявках, а также хранить данные необходимые для авторизации пользователя (логин и пароль).

Чтобы построить базу данных, требуется создать модель IDEF1X, включающую информацию о различных сущностях и связях между ними. В этой модели каждая сущность представлена прямоугольником на диаграмме, который разделен на две части: верхнюю, где указаны основные атрибуты (ключевые поля), и нижнюю, содержащую неключевые поля. Сущности могут соединяться двумя типами связей: идентифицирующей – когда родительский элемент однозначно задан для дочернего, и неидентифицирующей – когда такая связь не однозначна. Идентифицирующие связи изображаются сплошной линией с точкой у дочерней сущности, в то время как неидентифицирующие — пунктирной линией с точкой [11].

Модель IDEF1X для разрабатываемой системы приведена на рисунке А.1. Она содержит в себе 4 сущности: «Студент», «Расписание», «Регистрация», «Заявки».

Сущность «Студент» содержит в себе информацию о студенте, а именно те данные о нем, которые обрабатываются факультетом ГУИМЦ. Сущность включает в себя следующие атрибуты: «Код_студента», «Имя_студента», «Группа_студента», «Пол_студента», «Гражданство_студента», «Статус_обучения_студента», «Форма_обучения_студента», «Статус_военного_обучения_студента», «Статус_проживания_в_общежитии_студента». Все столбцы должны быть

заполнены, то есть они имеют атрибуты NOT NULL. Типы данных атрибутов сущности «Студент» приведены в таблице 2.

Таблица 2 – Типы данных атрибутов сущности «Студент»

Поле	Тип данных	Ключ
Код_пользователя	Целочисленный	Первичный ключ
Имя_студента	Текстовый	
Группа_студента	Текстовый	
Пол_студента	Текстовый	
Гражданство_студента	Текстовый	
Статус_обучения_студента	Текстовый	
Форма_обучения_студента	Текстовый	
Статус_военного_обучения_студента	Текстовый	
Статус_проживания_в_общежитии_студента	Текстовый	

Сущность «Расписание» содержит в себе информацию о занятиях в университете. Данная сущность включает в себя следующие атрибуты: «Код_занятия», «Учебная_группа», «Название_занятия», «Время_занятия», «Тип_недели», «День_недели». Все столбцы должны быть заполнены, то есть они имеют атрибуты NOT NULL. Типы данных атрибутов сущности «Расписание» приведены в таблице 3.

Таблица 3 – Типы данных атрибутов сущности «Расписание»

Поле	Тип данных	Ключ
Код_занятия	Целочисленный	Первичный ключ
Учебная_группа	Текстовый	
Название_занятия	Текстовый	

Продолжение таблицы 3

Поле	Тип данных	Ключ
Время_занятия	Текстовый	
Тип_недели	Текстовый	
День_недели	Текстовый	

Сущность «Регистрация» содержит в себе регистрационные данные необходимые для авторизации. Сущность включает в себя следующие атрибуты: «Код_регистрации», «Логин», «Пароль». Все столбцы должны быть заполнены, то есть они имеют атрибуты NOT NULL. Также атрибут «Логин» обладает требованием уникальности, чтобы исключить конфликты при авторизации. Типы данных атрибутов сущности «Регистрация» приведены в таблице 4.

Таблица 4 – Типы данных атрибутов сущности «Регистрация»

Поле	Тип данных	Ключ
Код_регистрации	Целочисленный	Первичный ключ
Логин	Текстовый	
Пароль	Текстовый	

Сущность «Заявки» содержит в себе информацию о заявках, поданных авторизованным пользователем. Данная сущность включает в себя следующие атрибуты: «Код_заявки», «Имя_автора_заявки», «Группа_автора_заявки», «Текст_заявки». Все столбцы должны быть заполнены, то есть они имеют атрибуты NOT NULL. Отметим, что данная таблица не имеет изначального заполнения и заполняется пользователем мобильного приложения для ГУИМЦ. Типы данных атрибутов сущности «Заявки» приведены в таблице 5.

Таблица 5 – Типы данных атрибутов сущности «Заявки»

Поле	Тип данных	Ключ
Код_заявки	Целочисленный	Первичный ключ
Имя_автора_заявки	Текстовый	
Группа_автора_заявки	Текстовый	
Текст_заявки	Текстовый	

Все связи на диаграмме являются идентифицирующими.

Представленная модель была создана и заполнена с помощью программы DB Browser for SQLite. Связь между базой данной и мобильным приложением была реализована с помощью библиотеки Room. Более подробно это рассмотрено ниже.

Пример заполнения таблицы «Студент» в БД представлен на рисунке 1. Пример заполнения таблицы «Расписание» в БД представлен на рисунке 2. Пример заполнения таблицы «Регистрация» в БД представлен на рисунке 3. Таблица «Заявки» изначальна пустая и заполняется пользователем при работе с приложением.

	studentId	name	group	gender	citizenship	status_education	form_education	army_education	dormitory
	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	Яровенко Максим Васильевич	ИУ5Ц-102Б	Мужской	Гражданин РФ	Обучается	Бюджетная	Не обучается	Не проживает
2	2	Свинцов Артемий Николаевич	ИУ5Ц-102Б	Мужской	Гражданин РФ	Обучается	Бюджетная	Не обучается	Проживает
3	3	Дмитриева Мария Дмитриевна	ИУ5Ц-101Б	Женский	Гражданин РФ	Обучается	Платная	Не обучается	Не проживает
4	4	Каунов Артем Федорович	ИУ5Ц-103Б	Мужской	Гражданин РБ	Обучается	Платная	Обучается	Проживает
5	5	Пылаев Богдан Алексеевич	ИУ5Ц-102Б	Мужской	Гражданин РБ	Обучается	Бюджетная	Обучается	Проживает

Рисунок 1 – Пример заполнения таблицы «Студент»

	scheduleId	group	subject	time	week	day_week
	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	ИУ5Ц-102Б	БЭ 502	08:30-10:05	Числитель	Понедельник
2	2	ИУ5Ц-102Б	Экономика 423ю	08:30-10:05	Знаменатель	Понедельник
3	3	ИУ5Ц-102Б	Беспроводные сети 432 (лекция)	10:15-11:50	Числитель	Понедельник
4	4	ИУ5Ц-102Б	Беспроводные сети 432 (лекция)	10:15-11:50	Знаменатель	Понедельник
5	5	ИУ5Ц-102Б	Экономика 515ю (лекция)	12:00-13:35	Числитель	Понедельник
6	6	ИУ5Ц-102Б	Инженерная этика 515ю (лекция)	12:00-13:35	Знаменатель	Понедельник
7	7	ИУ5Ц-102Б	Инженерная этика 502	13:50-15:25	Числитель	Понедельник
8	8	ИУ5Ц-102Б	МППР 395	13:50-15:25	Знаменатель	Понедельник
9	9	ИУ5Ц-102Б	Безопасность жизнедеятельности ...	08:30-10:05	Числитель	Среда
10	10	ИУ5Ц-102Б	Безопасность жизнедеятельности ...	08:30-10:05	Знаменатель	Среда
11	11	ИУ5Ц-102Б	Технология конструирования ЭВМ ...	10:15-11:50	Числитель	Среда
12	12	ИУ5Ц-102Б	Технология конструирования ЭВМ ...	10:15-11:50	Знаменатель	Среда
13	13	ИУ5Ц-102Б	Технология конструирования ЭВМ ...	12:00-13:35	Числитель	Среда
14	14	ИУ5Ц-102Б	МППР 395 (ЛР)	13:50-15:25	Знаменатель	Четверг
15	15	ИУ5Ц-102Б	МППР 395 (ЛР)	15:40-17:15	Знаменатель	Четверг
16	16	ИУ5Ц-102Б	Средства проектирования 903 (ЛР)	17:25-19:00	Числитель	Четверг
17	17	ИУ5Ц-102Б	Беспроводные сети 306э (ЛР)	17:25-19:00	Знаменатель	Четверг
18	18	ИУ5Ц-102Б	Средства проектирования 903 (ЛР)	19:10-20:45	Числитель	Четверг
19	19	ИУ5Ц-102Б	Беспроводные сети 306э (ЛР)	19:10-20:45	Знаменатель	Четверг
20	20	ИУ5Ц-102Б	ИМ 903 (ЛР)	10:15-11:50	Числитель	Пятница
21	21	ИУ5Ц-102Б	ИМ 903 (ЛР)	12:00-13:35	Числитель	Пятница
22	22	ИУ5Ц-102Б	Средства проектирования 432 ...	13:50-15:25	Числитель	Пятница
23	23	ИУ5Ц-102Б	Средства проектирования 432 ...	13:50-15:25	Знаменатель	Пятница
24	24	ИУ5Ц-102Б	МППР 515ю (лекция)	15:40-17:15	Числитель	Пятница

Рисунок 2 – Пример заполнения таблицы «Расписание»

	registrationId	login	password
	Фильтр	Фил...	Фильтр
1	1	stud1	1
2	2	stud2	2
3	3	stud3	3
4	4	stud4	4
5	5	stud5	5

Рисунок 3 – Пример заполнения таблицы «Регистрация»

2.1.2 Разработка архитектуры приложения

Архитектура мобильного приложения — это структурированный план, который определяет, как приложение будет работать и взаимодействовать с различными компонентами. Приведем ниже основные компоненты, которые обычно включаются в архитектуру мобильного приложения и которые были задействованы при создании мобильного приложения для ГУИМЦ:

1) *Пользовательский интерфейс (UI)*: это все, что видит пользователь. Он должен быть интуитивно понятным и обеспечивать легкость использования. В коде приложения за пользовательский интерфейс отвечают макеты. Ниже они будут рассмотрены более подробно.

2) *Навигация*: осуществляет перемещение между различными элементами в приложении. В мобильном приложении это реализуется с помощью фрагментов и компонента навигации.

3) *Бизнес-логика*: сердце приложения, где обрабатываются данные и выполняются основные функции. Это может включать в себя валидацию, вычисления, обработку бизнес-правил и другое. Данный код реализуется в моделях представлений и их вспомогательных компонентах – фабриках моделей представлений.

4) *Модель данных*: определяет, как данные структурированы и хранятся. Включает в себя базы данных и схемы данных. В создаваемом приложении работа с базой данных реализована с помощью библиотеки хранения данных Room.

5) *Управление зависимостями*: инструменты и практики для управления библиотеками и фреймворками, от которых зависит приложение.

Надо отметить, что это не все компоненты архитектуры мобильного приложения. В зависимости от сложности и масштаба создаваемого продукта в нем могут использоваться такие компоненты как: сетевые компоненты, управление состоянием, управление безопасностью, мониторинг и аналитика и другие. В связи с тем, что эти компоненты не реализовывались в мобильном

приложении для ГУИМЦ, то они не будут рассмотрены подробно. Но при расширении и введение в эксплуатацию готового продукта эти компоненты будут задействованы.

Ниже приведено подробное описание реализованных в мобильном приложении для ГУИМЦ компонентов.

2.1.2.1 Макеты

Макеты в разработке Android-приложений играют ключевую роль, определяя внешний вид и пользовательский интерфейс приложения. Они представляют собой XML-файлы, которые определяют расположение и размер элементов пользовательского интерфейса на экране. Именно макеты определяют, как приложение будет выглядеть на экране мобильного устройства.

При проектировании макетов необходимо учитывать не только эстетику, но и удобство использования для конечного пользователя. Оптимальный макет должен обеспечивать интуитивно понятный интерфейс, удобный доступ к функциям приложения и эффективное использование пространства экрана.

Основные принципы создания удобного интерфейса:

– *Простота и минимализм*. Интерфейс должен содержать только необходимые элементы и функции. Это позволит пользователю быстро находить нужную информацию и выполнять задачи без лишних усилий.

– *Последовательность и предсказуемость*. Элементы интерфейса должны располагаться в соответствии с общепринятыми стандартами и правилами. Это позволит пользователю легко ориентироваться в приложении и предсказывать его поведение.

– *Адаптивность*. Интерфейс должен адаптироваться под различные размеры экранов и ориентации устройств. Это обеспечит удобство использования приложения на разных устройствах.

– *Интуитивность*. Интерфейс должен быть интуитивно понятным. Пользователь должен иметь возможность легко понять назначение каждого элемента и выполнить нужное действие без дополнительных объяснений.

– *Эстетичность*. Интерфейс должен быть привлекательным и гармоничным. Это создаст положительное впечатление о приложении и повысит его привлекательность для пользователя.

Также важно учитывать особенности целевой аудитории при разработке интерфейса. Например, для детей и пожилых людей могут потребоваться более простые и понятные интерфейсы, а для лиц с ограниченными возможностями для использования приложения могут быть введены специальные режимы, упрощающие взаимодействие пользователя с приложением.

Существует несколько типов макетов. Рассмотрим их:

1) *Линейные макеты (Linear Layouts)*: это один из самых простых типов макетов, в котором элементы располагаются последовательно в одном направлении (вертикально или горизонтально).

2) *Относительные макеты (Relative Layouts)*: эти макеты позволяют размещать элементы относительно друг друга или относительно родительского контейнера.

3) *Фреймовые макеты (Frame Layouts)*: в таких макетах элементы размещаются в виде слоев, каждый из которых может перекрывать другие.

4) *Табличные макеты (Table Layouts)*: эти макеты используются для размещения элементов в виде таблицы с заданным числом строк и столбцов.

5) *Макеты со списками (List View Layouts)*: используются для отображения больших списков данных, где каждый элемент списка имеет одинаковую структуру.

6) *Макеты с ограничениями (ConstraintLayout)*: новый тип макета, который позволяет создавать сложные макеты с использованием ограничений.

Выбор того или иного типа макета обусловлен требованиями создаваемого приложения и потребностями пользователей. Разработчик должен учитывать дизайн и компоновку элементов – разные типы макетов предоставляют разные

способы организации элементов на экране – а также сложность пользовательского интерфейса и производительность, эффективность выбранного типа макета [12].

В разрабатываемом приложении для ГУИМЦ используются линейные макеты (в основном для вспомогательных частей, которых пользователь визуально не видит) и макеты с ограничениями (для отображаемых экранов со сложным пользовательским интерфейсом).

Код линейного макета fragment_internal.xml приведен во фрагменте кода А.2. Он выстраивает композицию из панели инструментов, места для отображения фрагментов приложения и нижнего меню. Этот макет считается вспомогательным, так как с помощью него отображаются другие макеты.

Макеты с ограничениями могут создаваться вручную – когда расположение всех элементов прописывается кодом – или с помощью редактора, предоставляемого Android Studio. На рисунке 4 изображен это редактор, с помощью которого создавался макет с ограничениями для экрана с расписанием.

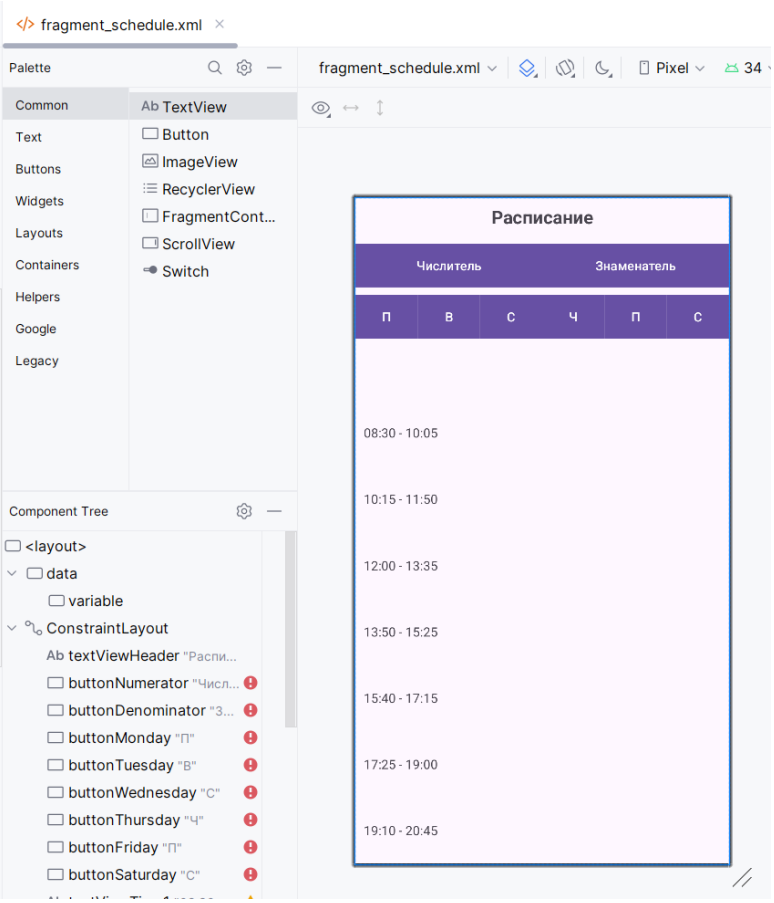

Рисунок 4 – Редактор макетов Android Studio

2.1.2.2 Фрагменты и навигация

Большинство реальных приложений содержит более одного экрана. К примеру, в приложении для электронной почты один экран может быть предназначен для написания новых писем, в то время как другой экран отображает список всех входящих сообщений. А в приложении для управления событиями один экран может демонстрировать общий список запланированных мероприятий, а другой — детализированную информацию о конкретном событии.

Для построения приложений с несколькими экранами используются фрагменты и компонент Navigation.

Фрагменты представляют собой модули пользовательского интерфейса внутри активности, которые позволяют создавать более гибкие и масштабируемые приложения. Они являются частью одного экрана в приложении, который может быть заменен или комбинирован с другими фрагментами в зависимости от действий пользователя или изменений в приложении.

Для управления навигацией между фрагментами в приложении используется компонент навигации (Navigation Component), представляющий собой часть архитектуры Jetpack. Он обеспечивает удобный способ определения навигационного графа приложения и управления переходами между фрагментами [13].

Пример кода графа навигации nav_graph.xml для обеспечения навигации в разрабатываемом приложении для ГУИМЦ приведен во фрагменте кода А.3. С помощью этого кода реализуется переход от экрана регистрации к экрану с персональными данными студента.

2.1.2.3 Меню Anroid-приложений

Для упрощения реализации навигации в мобильном приложении Android Studio предлагает такой инструмент как меню. Как и компонент навигации оно служит для организации переходов между различными частями приложения, но гораздо проще в создании, чем настройка навигации между большим числом экранов вручную. Меню в Android-приложениях играют ключевую роль в улучшении интерактивности и навигации.

Основные типы меню, которые можно реализовать в Android-приложении:

1) *Меню опций (Options Menu)*: это основное меню, доступное через кнопку меню устройства или панель действий. Оно используется для предоставления различных опций, связанных с текущей активностью пользователя.

2) *Контекстное меню (Context Menu)*: появляется при долгом нажатии на элемент интерфейса, например, на элемент списка, и предлагает действия, связанные непосредственно с этим элементом.

3) *Панель навигации (Navigation Drawer)*: выдвижная панель, которая отображается поверх основного содержимого и содержит основные разделы приложения для быстрой навигации. Разновидностью панели навигации является нижняя панель навигации, которая закрепляется в нижней части экрана.

4) *Панель инструментов (Toolbar)*: гибкий элемент, который может использоваться как панель действий, содержащий логотип, название, кнопки действий и другие элементы интерфейса.

5) *Плавающая кнопка действия (Floating Action Button - FAB)*: круглая кнопка, обычно расположенная в нижнем правом углу, предназначенная для основного действия в приложении.

В мобильном приложении для ГУИМЦ реализована нижняя панель навигации, которая состоит из трех кнопок, осуществляющих навигации между экранами с персональными данными студента, расписанием и заявками. Код меню приведен во фрагменте кода А.4.

2.1.2.4 Модели представлений и фабрики моделей представлений

С ростом сложности приложений фрагментам приходится жонглировать все большим количеством объектов. И если не проявить осторожность, это может привести к разбуханию кода, который пытается делать все сразу. Бизнес-логика, навигация, управление пользовательским интерфейсом, обработка изменений в конфигурации – весь код, отвечающий за эти задачи, принято отделять от фрагмента, используя модели представления.

Модель представления (ViewModel) — отдельный класс, который существует параллельно с кодом активности или фрагмента. Он отвечает за все данные, которые должны отображаться на экране, а также может содержать любую бизнес-логику. Каждый раз, когда фрагменту требуется обновить свой макет, он запрашивает у модели представления новейшие значения, которые ему нужно отобразить, а если ему потребуется доступ к бизнес-логике, он вызывает методы, содержащиеся в модели представления [14].

Рассмотрим преимущества использования моделей представления:

1) Отделение бизнес-логики от пользовательского интерфейса: модели представления позволяют разработчикам изолировать код, отвечающий за бизнес-логику, от кода, отвечающего за отображение данных на экране. Это способствует улучшению структуры приложения и облегчает его поддержку и развитие.

2) Управление жизненным циклом: ViewModel сохраняет свое состояние при изменениях конфигурации устройства (например, при повороте экрана), что позволяет сохранять данные и состояние приложения без необходимости повторной загрузки или запроса данных.

3) Облегчение тестирования: изолированная бизнес-логика в моделях представления делает их легко тестируемыми, так как они не зависят от конкретного жизненного цикла активности или фрагмента.

4) Повышение производительности: использование ViewModel позволяет избежать лишней загрузки данных или выполнения запросов к базе данных при повторном создании фрагментов или активностей.

Для создания объекта модели представления и связывания его с активностями и фрагментами используется провайдер модели представления ViewModelProvider — специальный класс, который должен предоставлять модели представлений активностям и фрагментам. Он гарантирует, что новый объект модели представления создается только в том случае, если он не существует.

Провайдер модели представления хранит модель представления, пока активность или фрагмент продолжают существовать. Когда фрагмент отсоединяется или удаляется из своей активности, провайдер разрывает связь с моделью представления фрагмента. Когда провайдер модели представления получает следующий запрос на объект модели представления, он создает новый объект.

Альтернативный способ создания модели представления основан на передаче провайдеру модели представления фабрики модели представлений: отдельного класса, единственным назначением которого является создание и инициализация моделей представлений. Такой подход означает, что провайдеру модели представления не придется беспокоиться о создании модели представления своими силами. Вместо этого он использует фабрику модели представления [14].

Хотя фабрики моделей представлений могут использоваться для любых разновидностей моделей представлений, чаще всего они используются для моделей, конструкторы которых должны получать аргументы. Дело в том, что провайдер модели представления не может передавать аргументы конструктору сам по себе: для этого он должен использовать фабрику модели представления.

Код фабрики модели представления StudentViewModelFactory.kt приведен во фрагменте кода А.5. Данный код используется для создания модели представления. Код этой модели приведен во фрагменте кода А.6. Этот код

отвечает за то, какие данные, взятые из базы данных, будут отображаться на экране персональных данных студента.

2.1.2.5 Связывание данных

В разрабатываемом мобильном приложении для ГУИМЦ используется связывание данных.

Связывание данных (Data binding) — это техника, используемая в программировании и разработке приложений, которая позволяет автоматически связывать данные между источниками данных и пользовательским интерфейсом. Это упрощает процесс отображения и обновления данных на пользовательском интерфейсе, минимизируя необходимость ручного кодирования для синхронизации данных и элементов интерфейса [15].

Основные идеи и возможности, предоставляемые техникой связывания данных, включают в себя:

1) *Автоматическая синхронизация*: Data binding автоматически обновляет пользовательский интерфейс (например, текстовые поля, виджеты и элементы управления) при изменении данных и наоборот, что позволяет отображать актуальные данные и уменьшает вероятность ошибок.

2) *Двунаправленная привязка*: связывание позволяет не только отображать данные на интерфейсе, но и обрабатывать введенные пользователем данные и автоматически обновлять соответствующие данные в источнике.

3) *Уменьшение кода*: с использованием связывания данных уменьшается объем кода, необходимого для обновления и отображения данных на интерфейсе, что улучшает читаемость кода и упрощает его обслуживание.

4) *Изоляция данных и представления*: связывание помогает разделить данные и их отображение, что делает код более модульным и гибким.

5) *Поддержка различных платформ и языков*: Data binding может использоваться в разных программных платформах и языках программирования,

включая Android, iOS, JavaScript, WPF (Windows Presentation Foundation) и другие [15].

В мобильном приложении для ГУИМЦ связывание данных используется для передачи введенных пользователем данных между моделями представлений, где обрабатывается бизнес-логика приложения, и макетами, которые отвечают за внешний вид приложения и отображения всех его элементов.

2.1.2.6 База данных Room

В большинстве приложений используются данные. Но если не позаботиться о том, чтобы эти данные были где-то сохранены, данные будут навсегда потеряны при закрытии приложения. В мире приложений Android информация обычно хранится в базах данных.

Во внутренней реализации многие базы данных Android используют SQLite. SQLite — упрощенная, стабильная, быстрая и оптимизированная для однопользовательского доступа система, и все эти особенности делают ее хорошим кандидатом для приложений Android. Тем не менее написание кода для создания, управления и взаимодействия с базами данных SQLite может быть непростым делом.

Для упрощения задачи в Android Jetpack включена библиотека хранения данных Room, которая работает на базе SQLite. Room открывает доступ ко всем преимуществам SQLite, но с более простым кодом. Например, библиотека поддерживает удобные аннотации, которые позволяют быстро писать код баз данных, менее однообразный и более надежный [14].

Рассмотрим преимущества использования Room:

1) *Простота использования*: Room обеспечивает простой и интуитивно понятный способ работы с базами данных, что позволяет разработчикам быстро создавать и поддерживать сложные структуры данных.

2) *Высокоуровневая абстракция*: Room предоставляет высокоуровневую абстракцию над SQLite, позволяя разработчикам избежать написания многочисленных строк кода для выполнения операций с базой данных.

3) *Безопасность и надежность*: Room предотвращает ошибки времени выполнения за счет использования аннотаций для проверки корректности SQL-запросов на этапе компиляции.

4) *Поддержка асинхронных операций*: Room позволяет выполнять операции с базой данных асинхронно, что повышает производительность и отзывчивость приложения.

Room использует набор интерфейсов и классов с аннотациями для создания баз данных SQLite для вашего приложения. Для этого необходимы три условия:

1) Класс базы данных.

Класс определяет базу данных, включая ее имя и номер версии. Он используется для получения экземпляра базы данных.

Для конфигурации базы данных приложения необходимо разработать абстрактный класс. Этот класс будет содержать информацию о названии базы данных и её версии, а также включать описания классов или интерфейсов, которые устанавливают структуру таблиц и предоставляют методы для работы с данными.

Код, определяющий базу данных «GuimcDatabases», приведен во фрагменте кода А.7. В нем прописываются все таблицы, которые входят в базу данных (студенты, расписание, заявки, регистрация).

2) Классы данных таблиц.

Вся информация в базе данных хранится в таблицах. Каждая таблица определяется при помощи класса данных, который включает аннотации для имени таблицы и ее столбцов.

Для каждого типа данных, который должен храниться в базе данных, создается отдельная таблица. К примеру, в приложении-календаре будет таблица

для хранения событий, тогда как в программе, предсказывающей погоду, найдется таблица для данных о различных географических регионах.

Для каждой таблицы, которая должна быть включена в базу данных, определяется класс данных. Этот класс данных должен включать свойство для каждого столбца таблицы, а аннотации сообщают Room, как должна быть настроена конфигурация таблицы [14].

Класс данных, определяющий таблицу «student_table», приведен во фрагменте кода А.8. В нем задаются следующие столбцы: studentId (первичный ключ), name (имя студента), group (группа студента), gender (пол), citizenship (гражданство), status_education (статус обучения), form_education (форма обучения), army_education (статус военного обучения), dormitory (проживание в общежитии).

3) Интерфейсы доступа к данным.

Для взаимодействия с таблицей используется интерфейс, который определяет методы доступа к данным, необходимые для приложения.

Чтобы определить, как приложение должно обращаться к данным таблицы, создается интерфейс с аннотациями. Интерфейс определяет объект DAO (Data Access Object, то есть «объект доступа к данным») со всеми методами, необходимыми приложению для вставки, чтения, обновления и удаления данных [14].

Код интерфейса доступа к данным таблицы «student_table» приведен во фрагменте кода А.9. В нем определяются методы: Insert – вставка записей, Update – обновление записей, Delete – удаление записей, а также определены методы, возвращающие результат SQL-запроса.

Room использует эти три составляющие для генерирования всего кода, необходимого приложению для создания базы данных SQLite, ее таблиц и методов доступа к данным.

2.1.2.7 Управление зависимостями

Управление зависимостями является ключевым аспектом разработки мобильных приложений. Это процесс, который позволяет разработчикам эффективно управлять и интегрировать внешние библиотеки и фреймворки, которые обеспечивают необходимый функционал для приложения.

Основные принципы управления зависимостями:

1) *Изоляция зависимостей*: каждая зависимость должна быть изолирована таким образом, чтобы изменения в одной не влияли на другие. Это помогает предотвратить "эффект домино", когда ошибка в одной библиотеке может привести к сбоям во всем приложении.

2) *Версионирование*: важно использовать конкретные версии зависимостей, чтобы избежать несовместимостей и неожиданных изменений в поведении приложения после обновления библиотек.

3) *Централизованное управление*: использование инструментов управления зависимостями, таких как Gradle для Android или CocoaPods для iOS, позволяет централизованно управлять всеми зависимостями проекта.

4) *Автоматическое разрешение конфликтов*: современные системы управления зависимостями могут автоматически разрешать конфликты между различными версиями библиотек, выбирая наиболее подходящую версию.

5) *Легкость обновления*: управление зависимостями должно обеспечивать простой процесс обновления библиотек, чтобы разработчики могли легко получать исправления ошибок и новые функции.

6) *Документация и поддержка*: хорошо документированные зависимости с активной поддержкой сообщества упрощают их интеграцию и обслуживание.

7) *Лицензирование*: необходимо учитывать лицензии используемых библиотек, чтобы избежать юридических проблем и соблюдать требования открытого исходного кода.

Управление зависимостями помогает разработчикам сосредоточиться на создании уникального функционала приложения, минимизируя время и усилия,

необходимые для решения проблем совместимости и обслуживания внешних библиотек. Это обеспечивает более стабильную и управляемую среду разработки.

При разработке мобильных приложений под ОС Android используется инструмент управления зависимостями Gradle.

Gradle — это современная система автоматизации сборки, широко используемая в разработке мобильных приложений, особенно на платформе Android. Она предоставляет гибкий способ описания и управления зависимостями, а также процессами сборки и развертывания приложений.

Во фрагменте кода А.10 приведено код файла build.gradle.kts, в котором прописаны все зависимости, использующиеся для сборки мобильного приложения для ГУИМЦ.

2.1.3 Описание и тестирование функционала приложения

Тестирование функционала мобильного приложения будет производиться при помощи мобильного устройства с установленной версией ОС Android 13.0. Стоит отметить, что приложение поддерживает версии ОС Android, начиная с версии 7.0.

После установки приложения на мобильное устройство для его запуска необходимо нажать на иконку приложения. При нажатии открывается стартовый экран, на котором имеются поля для ввода логина и пароля и кнопка, при нажатии которой будет осуществлена проверка введенных пользователем логина и пароля. Если введенные данные верны, то приложение перейдет к экрану с персональными данными студента, в противном случае – на экране отобразится надпись: «Введенный логин или пароль неверный».

На рисунке 5 представлен стартовый экран авторизации мобильного приложения. Вверху изображен логотип факультета ГУИМЦ. Ниже две строки с информацией о том, какую функцию выполняет данный экран. Из этих строк видно, что экран предназначен для регистрации.

Рисунок 5 – Экран авторизации мобильного приложения

На рисунке 6 показана ситуация, когда введен неверный логин или пароль. При нажатии на кнопку «Войти», если логин и пароль введены неверно, то на экране выведется строка, окрашенная в красный цвет, с этой информацией для пользователя. Пользователь может стереть неверно введенные данные и ввести их заново. Если они будут ведены верно, то приложение по кнопке «Войти» откроет экран с персональными данными студента.

Рисунок 6 – Экран авторизации мобильного приложения после нажатия на кнопку «Войти» при неверном логине или пароле

После того как пользователь ввел правильный логин и пароль и нажал на кнопку «Войти», то откроется экран с персональными данными авторизованного студента. На этом экране отображается вся информация, взятая из таблицы «Студент» базы данных, которая относится к авторизованному студенту. Внизу экрана располагается нижняя панель навигации, с помощью которой можно переключаться между другими экранами мобильного приложения. Стоит отметить, что при переходе на другой экран нижняя панель навигации остается на том же месте и с помощью нее можно вернуться на любой другой экран. На

рисунке 7 представлен пример экрана с персональными данными авторизованного студента.

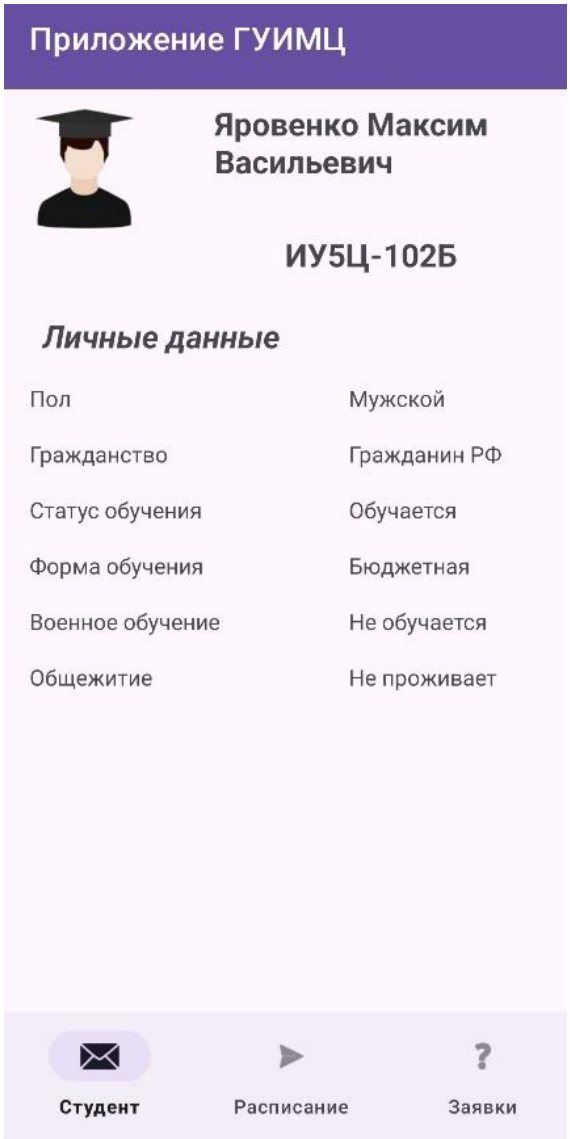

Рисунок 7 – Экран с персональными данными авторизованного студента

При нажатии на кнопку «Расписание» на нижней панели навигации откроется экран с расписание той учебной группы, к которой относится студент. На этом экране отображается расписание выбранного дня недели с учетом знаменателя или числителя. Выбор того или иного дня осуществляется с помощью кнопок в верхней части экрана. При нажатии на кнопку «Числитель» будет отображаться расписание дня недели по числителю, а при нажатии на кнопку «Знаменатель» - по знаменателю. При нажатии на кнопки, соответствующие дням недели, будет открываться расписание

соответствующего дня недели (понедельник, вторник и т.п.). Все отображаемые данные берутся из базы данных. На рисунке 8 приведен пример экрана с расписанием.

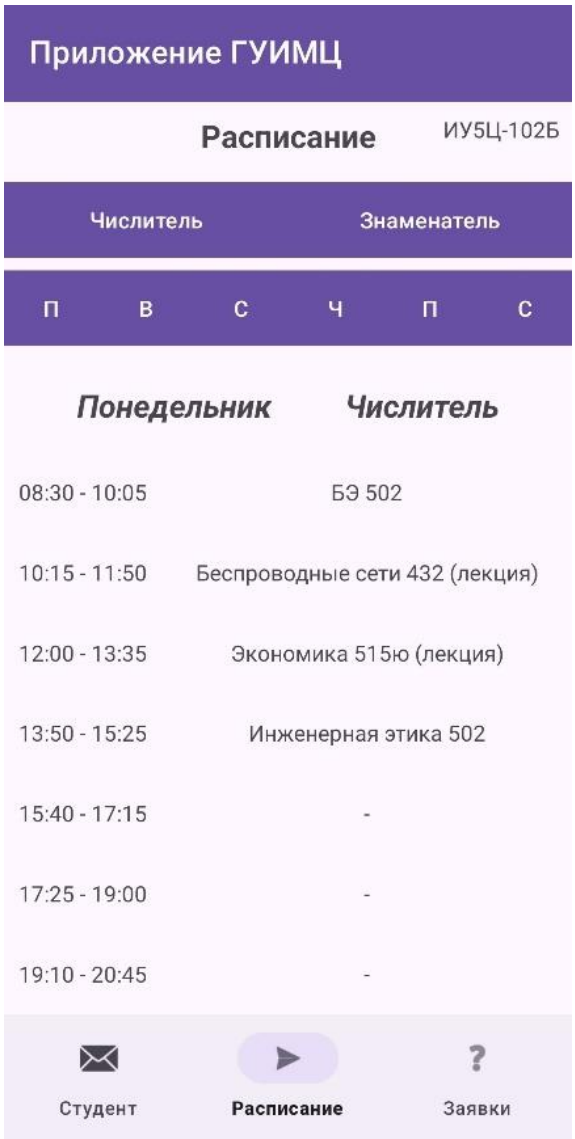

Рисунок 8 – Экран с расписанием

При нажатии на кнопку «Заявки» на нижней панели навигации откроется экран с заявками студента. На этом экране пользователь может ввести в поле заявку и при нажатии на кнопку «Отправить» она внесется в таблицу «Заявки» базы данных. В нижней части экрана отображается список с ранее введенными заявками. Этот список формируется на основе записей в таблице «Заявки» базы данных. На рисунке 9 приведен пример экрана с заявками.

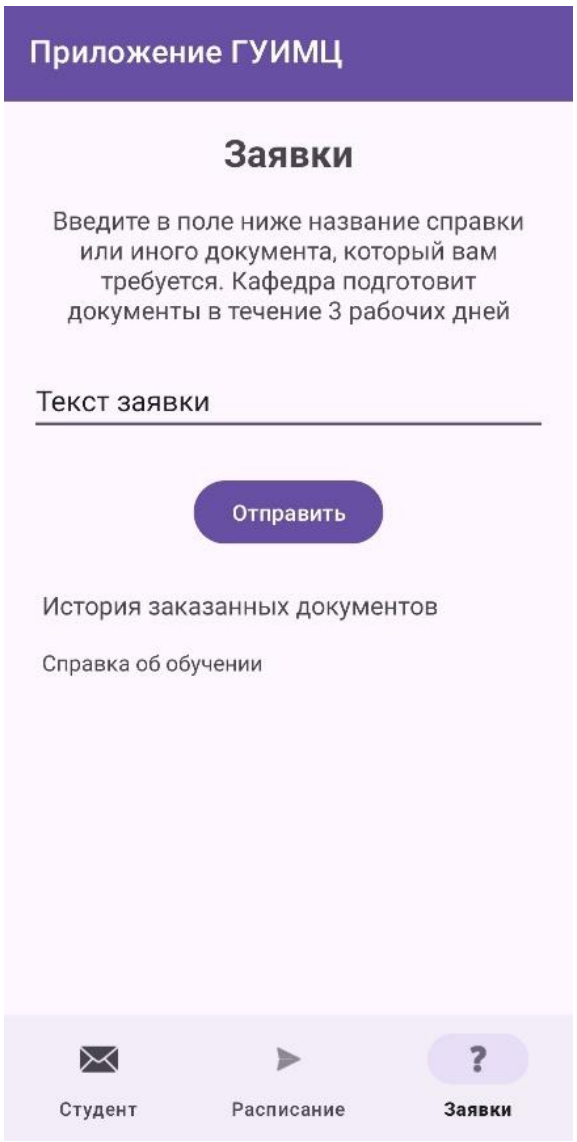

Рисунок 9 – Экран с заявками

Также в мобильном приложении для ГУИМЦ переопределена работа кнопки возврата телефонного меню. При ее нажатии переходит возврат к экрану с персональными данными студента, если на данный момент этот экран не открыт. Если же открыт экран с персональными данными студента или экран с регистрацией, то нажатие на кнопку возврата закроет приложение.

В целях безопасности приложение не сохраняет авторизацию пользователя. При выходе из приложения и повторном его открытии пользователю придется заново ввести логин и пароль от своего аккаунта. При этом, если приложение не закрывалось, а находится в фоновом режиме, то выход из аккаунта не осуществляется.

2.2 Выводы по второй главе

Во второй главе была описана логическая модель базы данных, которая используется в мобильном приложении. Описание модели было представлено в форме диаграммы «сущность-связь». На ней были изображены сущности, выделенные для предметной области («Студент», «Расписание», «Регистрация», «Заявки»), атрибуты, которыми обладают эти сущности, а также связи между ними. Была разобрана каждая из заявленных сущностей: были описаны типы данных атрибутов и приведены примеры их заполнения.

Затем была описана архитектура мобильного приложения и компоненты на которых она основана: макеты, фрагменты с компонентом навигации, модели представлений и фабрики моделей представлений, библиотека базы данных Room, файлы управления зависимостями. Все эти компоненты были разобраны подробно с приведением примеров кода, использованного при создании мобильного приложения.

В завершении проведено было описание рабочего мобильного приложения для ГУИМЦ и осуществлен тест его работоспособности. По его итогам можно говорить о достижении цели разработки и выполнении поставленных задач. Функциональность мобильного приложения отвечала всем заявленным требованиям.

ЗАКЛЮЧЕНИЕ

При разработке мобильного приложения для ГУИМЦ были достигнуты следующие результаты.

Была изучена предметная область в виде образовательной деятельности факультета ГУИМЦ МГТУ имени Н.Э. Баумана. В ходе анализа рабочего процесса, осуществляемого в предметной области, была обоснована актуальность идеи создания мобильного приложения и важность его разработки. Выделяя главное, востребованность создаваемого продукта опирается на его практическую пользу, а именно его использование упрощает образовательный процесс студентам, которые будут пользоваться мобильным приложением для ГУИМЦ.

Далее была сформулирована цель разработки, заключающаяся в облегчении учебного процесса студентам и рабочего процесса сотрудникам факультета. На ее основе были сформулированы задачи, выполнение которых приблизит нас к поставленной цели. К этим задачам отнесли следующее: исследование предметной области, проведение анализа существующих аналогов, определение функциональных требований приложения и его разработка с использованием заранее отобранных средств.

В ходе выполнения данной работы были разработаны функциональные требования к конечному продукту. Среди них можно выделить такие, как авторизация пользователя, отображения расписания учебной группы студента, занесение заявок студента в базу данных и другие.

Перед началом фактической разработки мобильного приложения был проведен разбор аналогов, выявлены их сильные и слабые стороны, на основе которых были сформулированы преимущества именно мобильного приложения для ГУИМЦ. Также не обошлось без тщательного выбора средств разработки и технологий, которые использовались при создании продукта. Стоит отметить, что ответственный подход к выбору инструментов предопределил успешное создание приложения.

Затем была разработана модель базы данных, которая используется в рабочем мобильном приложении. Ее диаграмма и примеры заполненных таблиц также были отражены в данной работе.

Подробное описание архитектуры мобильного приложения и ее компонентов, которые использовались в разработке, занимает основную часть данной работы. Были подробно разобраны такие вещи, как макеты, фрагменты, модели представления, фабрики моделей представлений, а также библиотека базы данных Room. Реализация заявленных компонентов была продемонстрирована фрагментами кода.

Созданное мобильное приложение было тщательно протестировано. Результаты данного этапа демонстрируют отсутствие ошибок в работе продукта и подчеркивают его работоспособность.

По итогам проделанной работы имеется готовое мобильное приложение под ОС Android, которое отвечает всем поставленным требованиям. Можно говорить о достижение заявленной цели и выполнении всех заданных задач. Функционал приложения действительно помогает студентам в ходе образовательного процесса на факультете ГУИМЦ.

Стоит отметить, что приложение имеет все возможности для дальнейшего расширения и модифицирования, чтобы становиться все совершеннее с каждым новым обновлением.

Графическая часть РПЗ представлена в приложении А. Техническое задание на ВКРБ приведено в приложении Б. Программа и методика испытаний ВКРБ приведена в приложении В.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Statcounter GlovalStats. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (дата обращения: 01.06.2024).
2. TIOBE. URL: https://www.tiobe.com/tiobe-index/ (дата обращения: 01.06.2024).
3. PYPL Index. URL: https://pypl.github.io/PYPL.html (дата обращения: 01.06.2024).
4. Stack Overflow. URL: https://survey.stackoverflow.co/2023/ (дата обращения: 01.06.2024).
5. Skillbox. URL: https://skillbox.ru/media/code/java_ili_kotlin_chto_vybrat_nachinayushchemu_android_razrabotchiku/ (дата обращения: 01.06.2024).
6. Java. URL: https://www.java.com/ru/ (дата обращения: 01.06.2024)
7. AppMaster. URL: https://appmaster.io/ru/blog/luchshie-instrumenty-dlia-sozdaniia-prilozhenii-dlia-android (дата обращения: 01.06.2024)
8. Android Studio. URL: https://developer.android.com/studio (дата обращения: 01.06.2024)
9. Kotlin. URL: https://kotlinlang.org/docs/multiplatform.html (дата обращения: 01.06.2024)
10. React Native. URL: https://reactnative.dev/ (дата обращения: 01.06.2024)
11. Верников Г. Основы методологии IDEF1. URL: https://www.cfin.ru/vernikov/idef/idef1.shtml (дата обращения: 15.05.2024).
12. OTUS. URL: https://otus.ru/journal/makety-ekranov-android-prilozhenij-opisanie-i-osobennosti/#Виды_макетов (дата обращения: 15.05.2024).
13. Android Developers. URL: https://developer.android.com/ (дата обращения: 15.05.2024).
14. Гриффитс Д., Гриффитс Д. Head First. Программирование для Android на Kotlin. 3-е изд. СПб.: Питер, 2023. 912 с.: ил.

15. Apptractor. URL: https://apptractor.ru/info/articles/chto-takoe-data-binding.html (дата обращения: 01.06.2024)

ПРИЛОЖЕНИЕ А
ГРАФИЧЕСКАЯ ЧАСТЬ

В графическую часть выпускной квалификационной работы входят:

Рисунок А.1 – Диаграмма IDEF1X для системы.

Фрагмент кода А.2 – Код линейного макета.

Фрагмент кода А.3 – Код графа навигации.

Фрагмент кода А.4 – Код нижней панели навигации.

Фрагмент кода А.5 – Код фабрики модели представления.

Фрагмент кода А.6 – Код модели представления.

Фрагмент кода А.7 – Код, задающий базу данных.

Фрагмент кода А.8 – Код, задающий таблицу «Студент».

Фрагмент кода А.9 – Код интерфейса доступа к данным таблицы «Студент».

Фрагмент кода А.10 – Код сборки зависимостей build.gradle.kts.

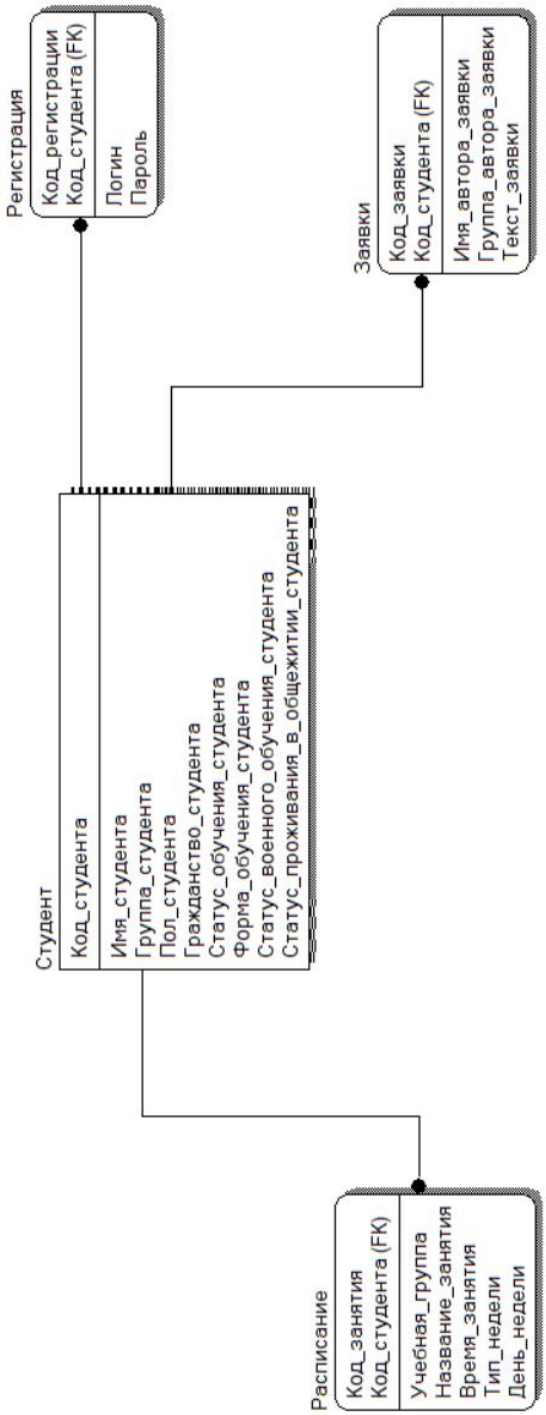

Рисунок А.1 – Диаграмма IDEF1X для системы

Фрагмент кода А.2 – Код линейного макета

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <com.google.android.material.appbar.MaterialToolbar
        android:id="@+id/toolbar"
        style="@style/Widget.MaterialComponents.Toolbar.Primary"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        app:title="Приложение ГУИМЦ" />

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/internal_nav_host_fragment"

android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        app:defaultNavHost="true"
        app:navGraph="@navigation/internal_nav_graph" />

<com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/bottom_nav"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:menu="@menu/main_menu" />
</LinearLayout>
```

Фрагмент кода А.3 – Код графа навигации

```
 <?xml version="1.0" encoding="utf-8"?>

<navigation
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/nav_graph"
    app:startDestination="@id/registrationFragment">

    <fragment
        android:id="@+id/registrationFragment"
        android:name="com.hfad.guimc.RegistrationFragment"
        android:label="fragment_registration"
        tools:layout="@layout/fragment_registration" >
        <action

android:id="@+id/action_registrationFragment_to_internalFragment"
            app:destination="@id/internalFragment"
            app:popUpTo="@id/registrationFragment"
            app:popUpToInclusive="true" />
    </fragment>

    <fragment
        android:id="@+id/internalFragment"
        android:name="com.hfad.guimc.InternalFragment"
        android:label="InternalFragment" />
</navigation>
```

Фрагмент кода А.4 – Код нижней панели навигации

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/studentFragment"
        android:icon="@android:drawable/ic_dialog_email"
        android:title="Студент" />
    <item
        android:id="@+id/scheduleFragment"
        android:icon="@android:drawable/ic_menu_send"
        android:title="Расписание" />
    <item
        android:id="@+id/orderFragment"
        android:icon="@android:drawable/ic_menu_help"
        android:title="Заявки" />

</menu>
```

Фрагмент кода А.5 – Код фабрики модели представления

```
package com.hfad.guimc

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider

class StudentViewModelFactory (private val dao: StudentDao) :
ViewModelProvider.Factory {

    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if
(modelClass.isAssignableFrom(StudentViewModel::class.java)) {
            return StudentViewModel(dao) as T
        }
        throw IllegalArgumentException("Unknown ViewModel")
    }
}
```

Фрагмент кода А.6 – Код модели представления

```
package com.hfad.guimc
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import kotlinx.coroutines.launch
import com.hfad.guimc.RegistrationData
class StudentViewModel (val dao: StudentDao) : ViewModel() {
    var name = MutableLiveData<String>()
    var group = MutableLiveData<String>()
    var gender = MutableLiveData<String>()
    var citizenship = MutableLiveData<String>()
    var statusEdu = MutableLiveData<String>()
    var formEdu = MutableLiveData<String>()
    var armyEdu = MutableLiveData<String>()
    var dormitory = MutableLiveData<String>()
    val studentId = RegistrationData.registrationId
    fun studentData() {
        //Сопрограмма нужна для обращения к базе данных в фоновом
режиме
        viewModelScope.launch {
            name.value = dao.getName(studentId)
            group.value = dao.getGroup(studentId)
            gender.value = dao.getGender(studentId)
            citizenship.value = dao.getCitizenship(studentId)
            statusEdu.value = dao.getStatusEdu(studentId)
            formEdu.value = dao.getFormEdu(studentId)
            armyEdu.value = dao.getArmyEdu(studentId)
            dormitory.value = dao.getDormitory(studentId)
            RegistrationData.studentName = dao.getName(studentId)
            RegistrationData.studentGroup =
dao.getGroup(studentId)
        }
    }
}
```

Фрагмент кода А.7 – Код, задающий базу данных

```
@Database(entities = [Student::class, Schedule::class,
Order::class, Registration::class], version = 3, exportSchema =
true)
abstract class GuimcDatabase : RoomDatabase() {
    abstract val studentDao: StudentDao
    abstract val scheduleDao: ScheduleDao
    abstract val orderDao: OrderDao
    abstract val registrationDao: RegistrationDao
    companion object {
        @Volatile
        private var INSTANCE: GuimcDatabase? = null

        fun getDatabase(context: Context): GuimcDatabase {
            synchronized(this) {
                // Если INSTANCE не null, то возвращаем его,
                // если же оно null, то создаем новую базу данных
                return INSTANCE ?: synchronized(this) {
                    val instance = Room.databaseBuilder(
                        context.applicationContext,
                        GuimcDatabase::class.java,
                        "app_database"
                    )
                        // Указываем название файла базы данных,
который нужно скопировать из assets

.createFromAsset("database/guimc_database.db")
                        .build()
                    INSTANCE = instance
                    instance
                }
            }
        }
    }
}
```

Фрагмент кода А.8 – Код, задающий таблицу «Студент»

```
package com.hfad.guimc
import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "student_table")
data class Student (
    @PrimaryKey(autoGenerate = true)
    var studentId: Int,

    @ColumnInfo(name = "name")
    var studentName: String = "",

    @ColumnInfo(name = "group")
    var studentGroup: String = "",

    @ColumnInfo(name = "gender")
    var studentGender: String = "",

    @ColumnInfo(name = "citizenship")
    var studentCitizenship: String = "",

    @ColumnInfo(name = "status_education")
    var studentStatusEdu: String = "",

    @ColumnInfo(name = "form_education")
    var studentFormEdu: String = "",

    @ColumnInfo(name = "army_education")
    var studentArmyEdu: String = "",

    @ColumnInfo(name = "dormitory")
    var studentDormitory: String = ""
)
```

Фрагмент кода А.9 – Код интерфейса доступа к данным таблицы «Студент»

```
package com.hfad.guimc

import androidx.room.Dao
import androidx.room.Delete
import androidx.room.Insert
import androidx.room.Query
import androidx.room.Update

@Dao
interface StudentDao {

    @Insert
    suspend fun insert(student: Student)

    @Update
    suspend fun update(student: Student)

    @Delete
    suspend fun delete(student: Student)

    @Query("SELECT `name` FROM student_table WHERE studentId ==
:id")
    suspend fun getName(id: Int): String

    @Query("SELECT `group` FROM student_table WHERE studentId ==
:id")
    suspend fun getGroup(id: Int): String

    @Query("SELECT `gender` FROM student_table WHERE studentId ==
:id")
    suspend fun getGender(id: Int): String

    @Query("SELECT `citizenship` FROM student_table WHERE
studentId == :id")
```

```
    suspend fun getCitizenship(id: Int): String

    @Query("SELECT `status_education` FROM student_table WHERE
studentId == :id")
    suspend fun getStatusEdu(id: Int): String

    @Query("SELECT `form_education` FROM student_table WHERE
studentId == :id")
    suspend fun getFormEdu(id: Int): String

    @Query("SELECT `army_education` FROM student_table WHERE
studentId == :id")
    suspend fun getArmyEdu(id: Int): String

    @Query("SELECT `dormitory` FROM student_table WHERE studentId
== :id")
    suspend fun getDormitory(id: Int): String
 }
```

Фрагмент кода А.10 – Код сборки зависимостей build.gradle.kts

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.jetbrains.kotlin.android)
    id("kotlin-kapt")
}

android {
    namespace = "com.hfad.guimc"
    compileSdk = 34
    buildFeatures {
        viewBinding = true
        dataBinding = true
    }

    defaultConfig {
        applicationId = "com.hfad.guimc"
        minSdk = 24
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"
        testInstrumentationRunner =
"androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-
optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
```

```
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8
        targetCompatibility = JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = "1.8"
    }
}

dependencies {
    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.appcompat)
    implementation(libs.material)
    implementation(libs.androidx.activity)
    implementation(libs.androidx.constraintlayout)
    implementation(libs.androidx.navigation.fragment.ktx)
    implementation(libs.androidx.navigation.ui.ktx)
    implementation(libs.androidx.lifecycle.viewmodel.compose)
    implementation(libs.androidx.room.common)
    implementation(libs.androidx.room.ktx)
    val room_version = "2.6.1"
    implementation("androidx.room:room-runtime:$room_version")
    annotationProcessor("androidx.room:room-
compiler:$room_version")
    kapt("androidx.room:room-compiler:$room_version")
    implementation("androidx.room:room-ktx:$room_version")
    implementation("androidx.room:room-rxjava2:$room_version")
    implementation("androidx.room:room-rxjava3:$room_version")
    implementation("androidx.room:room-guava:$room_version")
    testImplementation("androidx.room:room-testing:$room_version")
    implementation("androidx.room:room-paging:$room_version")
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
}
```

ПРИЛОЖЕНИЕ Б

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

Утверждаю
Заведующий кафедрой ИУ-5
_____В.И. Терехов
"__"_____2024 г.

Согласовано
научный руководитель
_____ К.С. Мышенков
"__"_____2024 г.

Разработка мобильного приложения для ГУИМЦ

Техническое задание
(вид документа)

писчая бумага
(вид носителя)

6
(количество листов)

ИСПОЛНИТЕЛЬ: _____ Яровенко Максим Васильевич

"__"_____2024 г.

Москва – 2024

1 Наименование

Разработка мобильного приложения для ГУИМЦ.

2 Основание для разработки

Основанием для разработки является задание на выпускную квалификационную работу, подписанное руководителем выпускной работы и утверждённое заведующим кафедрой ИУ5 МГТУ им. Н.Э. Баумана.

3 Исполнитель

Студент пятого курса МГТУ им. Н. Э. Баумана группы ИУ5Ц–102Б Яровенко М. В.

4 Цель работы

Цель работы заключается в повышении удобства обучения студентов факультета ГУИМЦ за счет создания мобильного приложения, функционал которого способствует эффективности взаимодействия с кафедрой студентов факультета ГУИМЦ.

5 Содержание работы

5.1 Задачи

1) Исследовать предметную область, определить функциональные задачи.
2) Провести сравнение и осуществить выбор инструментов и версий программного обеспечения для создания приложения.
3) Разработать архитектуру приложения.
4) Собрать данные предметной области.
5) Структурировать и подготовить данные предметной области.
6) Разработать и реализовать способ хранения данных предметной области.

7) Разработать интерфейс пользователя приложения.

8) Разработать алгоритмы, требуемые для реализации приложения.

9) Разработать и реализовать программную часть приложения.

10) Провести тестирование работы приложения.

11) Провести отладку приложения.

12) Оформить техническую документацию.

5.2 Требования к функциональным характеристикам

Разрабатываемое приложение должно выполнять следующие функции:

1) Авторизация пользователя.

2) Просмотр информации об авторизованном студенте.

3) Просмотр информации о расписании учебной группы студента.

4) Создание авторизованным студентом заявки для кафедры.

5) Вывод на экран списка созданных ранее заявок.

6) Реализация работы кнопки возврата телефонного меню и выход из приложения.

5.3 Требования к входным и выходным данным

5.3.1 Требования к входным данным

Входные данные представляют собой данные, полученные от пользователя и необходимые для авторизации:

1) Логин.

2) Пароль.

5.3.2 Требования к выходным данным

Выходные данные представляют собой данные, выведенные на экран пользователя:

1) Информация о студенте.

2) Расписание учебной группы студента.

3) Список созданных ранее студентом заявок.

Таким образом, пользователь системы может просматривать информацию и получать необходимые сведения.

5.4 Требования к надёжности

Приложение должно надёжно и устойчиво функционировать, при вводе некорректных данных выдавать сообщение. При сбоях восстанавливаться после перезагрузки.

Хранилище данных должно работать так, чтобы в случае возникновения сбоев нельзя было допустить потери или повреждение информации.

5.5 Лингвистические требования

Клиентская часть веб–приложения должна быть русифицирована.

5.6 Требования к составу программных средств

Для работы клиентской части приложения на компьютере пользователя требуется:

1) ОС Windows.

2) Эмулятор Android-приложений на ПК.

Для работы клиентской части приложения на мобильном устройстве пользователя требуется:

1) ОС Android.

5.7 Требования к составу технических средств

Минимальные системные требования для работы приложения на ПК:

1) Процессор с тактовой частотой 1 ГГц.

2) Оперативная память 1 Гб.

3) Видеоадаптер и монитор.

4) Жёсткий диск объёмом 10 Гб.

5) «Мышь» или другое указывающее устройство.

6) Клавиатура.

7) Сетевой адаптер.

Минимальные системные требования для работы приложения на мобильном устройстве:

1) Операционная система: Android 7.0 и выше.
2) Размер ОЗУ: от 2 ГБ.
3) Размер встроенной памяти: от 8 ГБ.

6 Этапы работы

График выполнения отдельных этапов работ приведён в таблице 1 в соответствии с приказом об организации учебного процесса в 2023/2024 учебном году.

Таблица 1 – Этапы разработки

№ п/п	**Наименование этапа и содержание работ**	**Сроки исполнения**
1	Постановка задач разработки, анализ инструментов и программного обеспечения для создания приложения	декабрь 2023 г.
2	Формулирование проблемы, цели и задач работы	январь 2024 г.
3	Разработка мобильного приложения для ГУИМЦ	февраль–апрель 2024 г.
4	Тестирование и отладка	апрель 2024 г.
5	Оформление документации	май–июнь 2024 г.
6	Показ научному руководителю итоговой работы	июнь 2024 г.

7 Техническая документация

По окончании работы предъявляется следующая техническая документация:

- техническое задание;
- рабочий материал по выполняемому проекту;
- программа и методика испытаний;
- руководство пользователя;
- графический материал по проекту в формате презентации.

8 Порядок приёма работы

Приём и контроль программного изделия осуществляется в соответствии с методикой испытаний (см. документ «Программа и методика испытаний»).

9 Дополнительные условия

Данное техническое задание может уточняться в установленном порядке.

ПРИЛОЖЕНИЕ В

ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

Утверждаю	Согласовано
Заведующий кафедрой ИУ-5	научный руководитель
_____В.И. Терехов	_____ К.С. Мышенков
"__"_____2024 г.	"__"_____2024 г.

Разработка мобильного приложения для ГУИМЦ

ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ
(вид документа)

писчая бумага
(вид носителя)

5
(количество листов)

ИСПОЛНИТЕЛЬ: _____ Яровенко Максим Васильевич

"__"_____2024 г.

Москва – 2024

Аннотация

В данном документе описываются последовательность и методы проведения испытаний при тестировании программного изделия, состав и структура технических и программных средств, необходимых для проведения испытаний, а также приводятся требования к предъявляемой документации, характеристикам программы применительно к условиям эксплуатации и требования к информационной и программной совместимости. Описывается тестовый пример и реакция системы на него.

1. Объект испытаний

Мобильное Android приложение для ГУИМЦ.

2. Цель испытаний

Цель испытания – проверка функционирования всех указанных в техническом задании функций программы.

3. Состав предъявляемой документации

На испытания программного продукта предъявляются следующие документы:

1) Техническое задание.

2) Программа и методика испытаний.

4. Технические требования

4.1 Требования к программной документации

Комплектность программной документации должна удовлетворять разделу данного документа "Состав предъявляемой документации".

4.2 Требования к техническим характеристикам

4.2.1 Требования к составу аппаратного обеспечения

Мобильное приложение для ГУИМЦ должно выполняться на мобильном устройстве со следующими характеристиками:

– Версия операционной системы Android 7.0 и выше;

4.2.2 Требование к составу программного обеспечения

Для работы мобильного приложения для ГУИМЦ требований к составу программного обеспечения не предъявляется.

5. Порядок проведения испытаний

Испытания данного программного продукта будут проводиться в следующем порядке:

1) Запуск мобильного приложения для ГУИМЦ.

2) Тестирование функционала мобильного приложения для ГУИМЦ.

5.1 Требования к составу аппаратного обеспечения

Требования к составу аппаратного обеспечения учитываются согласно пункту 4.2.1.

5.2 Требования к составу программного обеспечения

Требования к составу программного обеспечения учитываются согласно пункту 4.2.2.

6. Методы испытаний

Методы испытаний приведены в таблице 1.

Таблица 1 – Методы испытаний

N	N пункта ТЗ	Выполняемые действия	Результат
1	5.2.1 Авторизация	Пользователь: 1. Открывает приложение; 2. Вводит логин и пароль; 3. Нажимает на «Продолжить».	Открывается экран с личными данными студента
2	5.2.2 Просмотр информации об авторизованном студенте	Пользователь нажимает на пункт меню «Студент».	Открывается экран с личными данными студента

N	N пункта ТЗ	Выполняемые действия	Результат
3	5.2.3 Просмотр информации о расписании учебной группы студента	Пользователь нажимает на пункт меню «Расписание».	Открывается экран с расписанием группы студента
4	5.2.3 Просмотр информации о расписании учебной группы студента	Пользователь нажимает на экране с расписанием студента на кнопки «Числитель», «Знаменатель», а также кнопки с днем недели	На экране выводится расписание выбранного дня недели
4	5.2.4 Создание авторизованным студентом заявки для кафедры	Пользователь нажимает на пункт меню «Заявки».	Открывается экран с заявками
5	5.2.4 Создание авторизованным студентом заявки для кафедры	Пользователь: 1. Вводит информацию о нужной заявке; 2. Нажимает на «Отправить».	В базу данных вносится введенная студентом запись.
6	5.2.5 Вывод на экран списка созданных ранее заявок	Вывод на экран списка созданных ранее заявок	На экране «Заявки» в нижней части выводятся все ранее созданные заявки

N	N пункта ТЗ	Выполняемые действия	Результат
7	5.2.6 Реализация работы кнопки возврата телефонного меню и выход из приложения	Пользователь нажимает на кнопку возврата телефонного меню (один или несколько раз)	Пользователь возвращается на предыдущий экран вплоть до выхода из приложения