Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ЛЕСОСИБИРСКИЙ ПЕДАГОГИЧЕСКИЙ ИНСТИТУТ –
филиал Сибирского федерального университета

Высшей математики, информатики, экономики и естествознания
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
Л.Н. Храмова
подпись инициалы, фамилия
«14» июня 2024 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии
код-наименование направления

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ПО ГОРНОЛЫЖНЫМ КУРОРТАМ КРАСНОЯРСКОГО КРАЯ

Руководитель 14.06.24 доктор технических наук А.П. Мохирев
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник 14.06.2024 А.А. Вагонис
подпись, дата инициалы, фамилия

Нормоконтролер 14.06.2024 Е.В. Киргизова
подпись, дата инициалы, фамилия

Лесосибирск 2024

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка мобильного приложения по горнолыжным курортам Красноярского края» содержит 55 страницы текстового документа, 12 иллюстраций, 3 таблицы, 40 использованных источников.

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ANDROID, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, МОБИЛЬНАЯ РАЗРАБОТКА, KOTLIN.

Цель исследования – теоретически обосновать и разработать мобильное приложение по горнолыжным курортам Красноярского края.

Объект исследования – процесс разработки мобильного приложения.

Предмет исследования – процесс разработки мобильного приложения по курортам Красноярского края.

Для достижения поставленной цели необходимо решить следующие задачи исследования:

− провести анализ предметной области и сделать обзор существующих мобильных решений;

− выделить функциональные требования к разрабатываемому мобильному приложению;

− разработать мобильное приложение по курортам Красноярского края под операционную систему Android.

В ходе написания выпускной квалификационной работы автор принимал участие в конференциях и конкурсах. К публикации принята статья.

Во время разработки мобильного приложения изучены основы языка программирования Kotlin, взаимодействие клиента приложения с Android API.

В результате выполнения выпускной квалификационной работы разработано мобильное приложение по курортам Красноярского края, которое было протестировано.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С развитием информационных технологий и мобильных устройств стало возможным создание приложений, значительно упрощающих и улучшающих пользовательский опыт в различных сферах жизни. Одним из таких секторов является горнолыжный спорт, который в последние годы приобретает все большую популярность. В связи с этим возникает актуальная потребность в разработке мобильных приложений, направленных на улучшение условий пребывания и практики горнолыжного спорта.

Цель исследования – теоретически обосновать и разработать мобильное приложение по горнолыжным курортам Красноярского края.

Объект исследования – процесс разработки мобильного приложения.

Предмет исследования – процесс разработки мобильного приложения по курортам Красноярского края.

Для достижения поставленной цели необходимо решить следующие задачи исследования:

− провести анализ предметной области и сделать обзор существующих мобильных решений;

− выделить функциональные требования к разрабатываемому мобильному приложению;

− разработать мобильное приложение по курортам Красноярского края под операционную систему Android.

Методы исследования:

− теоретические: анализ учебной и научно-технической литературы по теме исследования; обобщение; сравнительный анализ;

− эмпирические: наблюдение; беседа; моделирование; тестирование программного продукта.

Результаты исследования представлены на научных мероприятиях:

1. VII Всероссийском научно-практической конференции «Актуальные

проблемы преподавания дисциплин естественнонаучного цикла» (г. Лесосибирск, ЛПИ–филиал СФУ, 22 ноября 2023 г.)

2. III Всероссийском молодёжном научном форуме «Современное педагогическое образование: теоретический и прикладной аспекты» (г. Лесосибирск, ЛПИ – филиал СФУ, 8-13 апреля 2024 г.).

По результатам исследования приняты к публикации статьи:

1. Вагонис, А. А. Исследование и анализ сферы разработки мобильного приложения с помощью конструктора для последующей разработки приложения / А. А. Вагонис // III Всероссийском молодежном научном форуме «Современное педагогическое образование: теоретический и прикладной аспекты» / ЛПИ – филиал СФУ – Лесосибирск.

Разработка мобильного приложения по горнолыжным курортам Красноярского края имеет широкий спектр практических преимуществ, способствуя улучшению условий пребывания и опыта пользователей, развитию туристической индустрии и повышению безопасности на курортах.

Структура работы – работа состоит из введения, двух глав, заключения, списка литературы, включающего 40 наименований. Результаты работы представлены в 3 таблицах, 12 рисунках. В 6 приложениях представлены организационная структура организации и листинги кода модулей приложения. Общий объем работы – 55 страницы.

1 Проектирование мобильного приложения

1.1Анализ предметной области

Красноярский край, расположенный в сердце Сибири, известен своими разнообразными природными ландшафтами и благоприятными условиями для зимних видов спорта, в частности, для горных лыж. Развитие горнолыжных курортов в этом регионе делает его привлекательным для туристов и спортсменов, однако актуальная и удобная информация о курортах часто оказывается недоступной или разрозненной. В этой связи актуальной становится задача разработки мобильного приложения, которое бы собрало и предоставило пользователям всю необходимую информацию о горнолыжных курортах Красноярского края.

Красноярский край имеет большое количество горнолыжных массивов и ландшафта, подходящего для туризма. Проанализированный потенциал региона поможет определить востребованность разрабатываемого приложения.

Проанализировав ландшафт Красноярского края и поняв, что Красноярский край подходит для горнолыжного туризма, встаёт следующая проблема инфраструктура услуг, которая поможет выявить потребности, которые можно устранить или минимизировать с помощью разработки удобного сервиса мобильного приложения.

В приложение должна отображена информация о трассе, погодных условиях, ценах на подъёмник, возможность отслеживать мероприятия, все эти функции помогают определить функциональные требования приложения.

Анализ уже использующихся приложений определяет конкретные преимущества разрабатываемого приложения.

Основными пользователями горнолыжных курортов являются туристы, спортсмены и местные жители, интересующиеся зимними видами спорта. Основные проблемы, с которыми сталкиваются посетители:

– недостаток информации – пользователи испытывают трудности в поиске актуальных данных о состоянии трасс, погодных условиях и работе подъемников;

– отсутствие единой платформы – информация о курортах часто разбросана по различным источникам, что усложняет планирование поездок;

– неудобство планирования – отсутствие интегрированных сервисов для бронирования жилья, покупки ски-пассов и аренды оборудования.

На текущий момент существует несколько приложений и веб-сайтов, предоставляющих информацию о горнолыжных курортах. Однако большинство из них не учитывают специфики Красноярского края или не предлагают полноценного функционала.

На основе анализа проблем были выделены основные требования к разрабатываемому мобильному приложению:

– актуальность информации – приложение должно предоставлять оперативные данные о состоянии трасс, погодных условиях и работе инфраструктуры курортов;

– единая платформа – интеграция информации о всех основных курортах Красноярского края;

– удобство использования – интуитивно понятный интерфейс, возможность бронирования услуг;

– дополнительные функции – карта курортов, система уведомлений о изменениях в работе курортов, возможность делиться отзывами и оценками.

Анализ предметной области показывает, что разработка мобильного приложения для горнолыжных курортов Красноярского края является актуальной задачей, способной существенно улучшить опыт пользователей. Приложение должно решать текущие проблемы, с которыми сталкиваются любители зимних видов спорта, и предоставлять удобный доступ к актуальной информации и сервисам.

1.2 UMl диаграммы для проектирования приложения

Для успешного проектирования мобильного приложения по горнолыжным курортам Красноярского края необходимо создать UML диаграммы, которые помогут визуализировать и структурировать все аспекты системы.

Проектирование мобильного приложения начинается с выделения и применения нескольких UML-диаграмм. Как отмечает Крег Ларман [13, с. 103], UML-диаграммы являются одним из средств проектирования для любого вида приложений.

В соответствии с вышесказанным спроектируем для мобильного приложения диаграммы и приведём примеры с их обоснованием,

Диаграмма вариантов (Use Case) – это диаграмма, которая основывается на определении сценария и распределении роли участников, также показывает взаимодействие между пользователем приложения и разрабатываемой системой [15, с. 45].

Диаграмма классов – это диаграмма, которая используется для визуального преставления системы, она показывает классы, из которых состоит система и отношения классов к системе [23, с. 25].

На рисунке 1 представлена диаграмма, которая показывает такие классы как сообщение о происшествии – этот класс представляет собой сообщение о происшествии, которое может быть отправлено пользователем, в случае чрезвычайного происшествия, информация о состоянии трасс – класс показывает информацию о состоянии склона, интегрируется он для начинающих и любителей горнолыжного спорта. Информация о состоянии подъемников – класс показывает информацию о стоимости подъёма и очереди на подъёмник, также этот класс помогает распределить людей по различным подъёмникам. Связь трасс и подъемников класс, который показывает информацию о заполненности трассы и уровни сложности её прохождения, что тоже помогает взаимодействовать приложению с новыми пользователями.

На рисунке 1 представлена диаграмма классов.

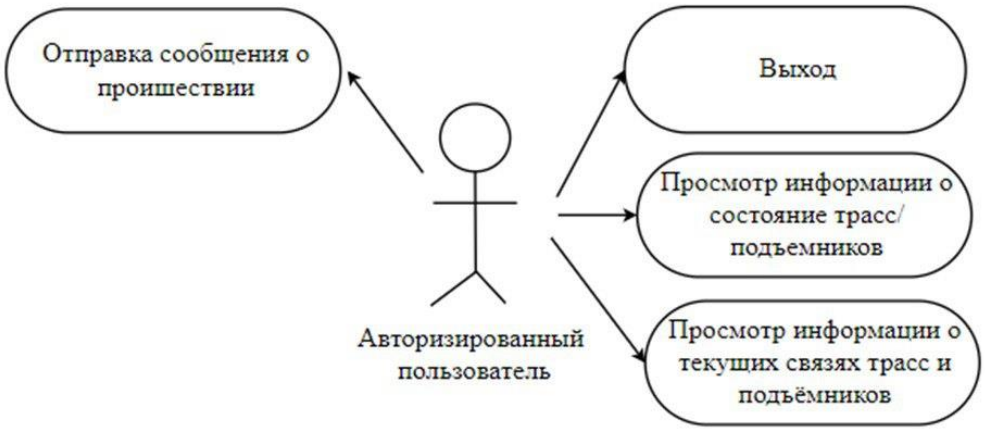

Рисунок 1 – Диаграмма классов

Диаграмма последовательности – показывает последовательность взаимодействия между объектами при выполнении конкретного сценария использования [12, с. 34].

Диаграмма состояний – описывает различные состояния, в которых может находиться объект или система, и переходы между этими состояниями. Эта диаграмма подходит для моделирования прикладных процессов [11, с. 22].

Диаграмма вариантов использования – это тип диаграммы UML, используемый для визуального представления функциональных требований к системе. Диаграмма показывает, кто взаимодействует с системой и что происходит в результате этого взаимодействия.

Пользователь, не прошедший верификацию данных акаунта это пользователь, который может быстро и легко посмотреть информацию о состоянии трассы и подъемника, а также информацию о текущей количестве людей на трассе и подъемнике.

Авторизованные пользователи – пользователи, которые могут выполнять те же операции, что и неавторизованные пользователи, но могут регистрироваться и пополнять баланс карты для доступа к бугельным и

кресельным подъемникам [12, с. 24].

Диаграмма прецендентов представлена на рисунке 2.

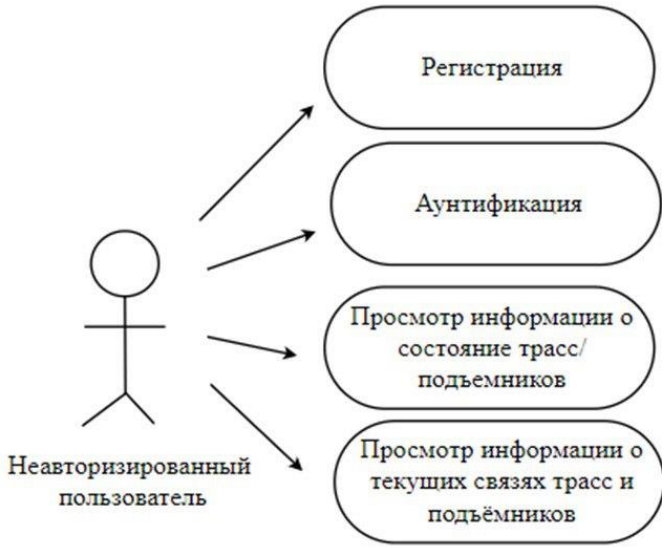

Рисунок 2 – Диаграмма прецедентов

Диаграмма действий – это тип uml диаграммы, используемой для визуализации процесса выполнения задачи [7, с. 104]. На ней показаны действия, которые может выполнять сотрудник лыжного патруля. Сотрудник следит за тем, чтобы информация о трассах или подъемниках была надежной и полной. Диаграмма деятельности представлена на рисунке 3.

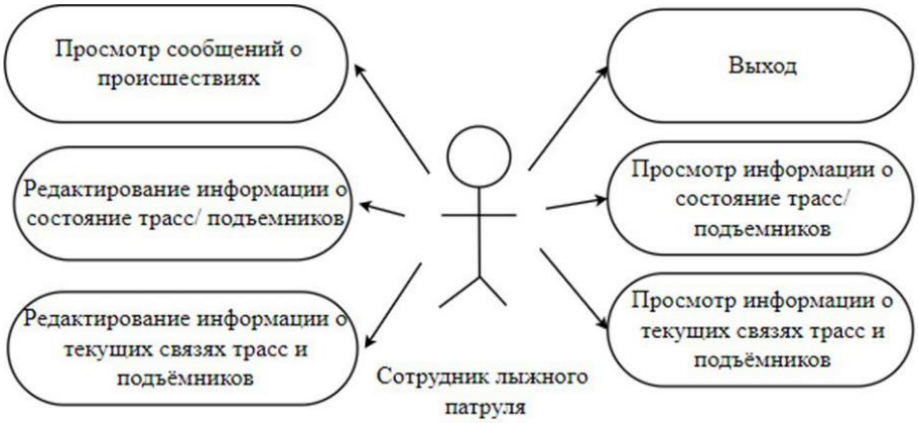

Рисунок 3 – Диаграмма деятельности

На диаграмме представлены действия, совершаемые сотрудником лыжного патруля:

1. получение отчета о происшествии действие, позволяющее получить отчет о происшествии от пользователя, и позволяет быстро отреагировать;

2. редактирование информации о состоянии трассы действие, которое проверяет состояние снежного полотна и отправляет отчёт о необходимой информации в базу данных для дополнения информации в приложении;

3. редактирование информации о связях трасс и подъёмников действие, совершаемое сотрудником для наполненности и достоверности информации о количестве людей на склоне и подъёмнике;

4. проверить состояние склона действие проверяет состояние склона, чтобы избежать несчастных случаев;

5. проверить состояние подъемника действие проверяет состояние подъемника, на котором произошел инцидент;

6. информация о склоне и подъемнике действие предоставляет информацию об уровне сложности склона и помогает новичкам распределить свою физическую подготовку и навыки.

Информация о подъемниках помогает пользователям понять загруженность трасс, избежать заторов и распределить людей в соответствии с количеством подъемников. Обновление информации о состоянии трасс также помогает избежать заторов на трассах.

Создание UML – диаграмм является ключевым этапом в проектировании мобильного приложения по горнолыжным курортам Красноярского края, так как они помогают четко структурировать и визуализировать систему. Используя диаграммы, можно эффективно определить функциональные требования и процессы, обеспечивая надежность и полноту информации, что способствует созданию удобного и надежного инструмента для планирования и бронирования горнолыжных поездок.

1.3 Определение требований к мобильному приложению

Определение требований является одним из ключевых этапов разработки мобильного приложения, так как позволяет четко установить, какие функции и возможности должны быть реализованы для удовлетворения нужд пользователей.

Чтобы проанализировать доступные мобильные приложения для горнолыжных курортов Красноярского края и определить наиболее важные требования, следует рассмотреть те функции и возможности, которые полезны и удобны для пользователей. Эти требования можно разделить на функциональные и нефункциональные.

Функциональные требования к приложению: приложение должно иметь регистрацию и авторизацию пользователя, создать учетную запись и иметь возможность входа в систему. Интеграция с социальными сетями (Яндекс, Google) для удобной регистрации и входа. Также должен быть профиль пользователя для управления личной информацией (имя, адрес электронной почты, контактная информация). К функциональным требования относятся:

– информация о курорте: список всех горнолыжных курортов региона с подробной информацией (название, описание, расположение, контактная информация). Карты курортных зон с указанием расположения склонов, подъемников и инфраструктуры;

– информация о трассах: список трасс с подробной информацией (уровень сложности, длина, текущий статус, открыт/закрыт). Карты дорог с маршрутами и уровнями сложности. Информация о погодных условиях: актуальная информация о погоде на курортах (температура, дождь, скорость ветра). Прогноз погоды на несколько дней;

– события и акции: календарь событий и развлекательных площадок (турниры, фестивали, вечеринки). Информация о текущих акциях и специальных предложениях;

– бронирование и оплата: возможность резервирования услуг (аренда оборудования, покупка билетов на подъемники, бронирование номеров). Поддерживает различные способы оплаты (банковские карты, электронные кошельки);

– обзоры и рейтинги: Возможность оставлять отзывы и оценки курортов и частных услуг. Смотрите рейтинги и отзывы других пользователей;

– навигация и GPS: интеграция GPS для навигации по курорту. Строительство дорог, ведущих к курортам и автомагистралей. Уведомления и напоминания:

– уведомления о предстоящих событиях, изменении погоды и других важных событиях. Напоминания о бронировании и акциях;

Неприменимые требования приложения, к ним относятся: простота использования, интуитивно понятный интерфейс с простыми и логичными разделами навигации. Адаптация под разные устройства (смартфоны, планшеты). К неприменимым приложениям относятся:

– производительность: быстрая загрузка данных и оперативность ввода данных пользователем. Оптимизирован для работы в условиях плохого интернет-соединения;

– безопасность и надежность хранения личной информации: защита персональных данных пользователей. Безопасность транзакций и платежных данных;

– стабильная работа приложения без сбоев и ошибок;

– регулярные обновления и поддержка.

Вышеперечисленные требования помогут создать удобное, надежное и функциональное мобильное приложение, которое удовлетворит потребности пользователей горнолыжных курортов Красноярского края и упростит их времяпровождение на курортах. На сегодняшний день существует не так много Android-приложений, позволяющих отслеживать зоны катания Красноярского края, из которых три основных приложения — «Шерегеш», «Красноярские

поляны» и «GornoList». Эти приложения предоставляют информацию о погодных условиях, состоянии склонов, уровнях снега и навигации по трассам, что делает их незаменимыми помощниками для любителей горных лыж и сноуборда в регионе.

Анализ существующих приложений, представлен в таблице 1.

Таблица 1 – Анализ существующих приложений

Название приложения	**Преимущества**	**Недостатки**	**Функционал**
Шерегеш	– Актуальная и достоверная информация – Позволяет купить ски-пасс онлайн – Удобный интерфейс	– Ограничено информацией о самом Шерегеше – Нет функций бронирования жилья и других услуг	– Карта трасс с актуальным статусом (открыто/закрыто) – Информация о скидках – Онлайн-табло с очередями на подъемники
Красноярские столбы	– Ограничено информацией о самом Шерегеше – Нет функций бронирования жилья и других услуг	– Не оптимизировано для горнолыжников и сноубордистов. – Нет информации о подъемниках и ски-пассах	– Карта скальных массивов и трасс – GPS-навигация – Прогноз погоды – Фото и видео отчеты пользователей – Форум для общения
GornoList	– Широкий охват курортов – Дополнительные функции (бронирование, поиск попутчиков)	– Не такая подробная информация о каждом курорте, как в официальных приложениях – Могут быть неточности в данных	– Информация о более чем 100 курортах России, включая курорты Красноярского края – Карты трасс, прогноз погоды, веб-камеры – Бронирование жилья и услуг

На данный момент нет единого приложения, которое бы полностью удовлетворяло все потребности горнолыжников и сноубордистов Красноярского края. В целом, рынок мобильных приложений для горнолыжных курортов Красноярского края имеет большой потенциал для развития. Ожидается, что в ближайшие годы появятся новые приложения, которые будут предлагать лыжникам и сноубордистам еще более удобные и функциональные сервисы.

Определение требований к мобильному приложению для горнолыжных

курортов Красноярского края помогает четко установить, какие функции и возможности должны быть реализованы для удовлетворения нужд пользователей. Анализ существующих приложений показывает, что наиболее востребованы функции, обеспечивающие удобство пользования и актуальность информации.

1.4 Структурная схема мобильного приложения

Структурная схема мобильного приложения представляет собой организацию его компонентов и взаимосвязей между ними [24, с. 34]. Для создания структурной схемы мобильного приложения по горнолыжным курортам Красноярского края, важно представить основные компоненты приложения и их взаимодействие. Структурная схема поможет визуализировать архитектуру приложения и определить основные модули и их связи.

Структурная схема мобильного приложения представляет собой иерархическое отображение его основных компонентов и их взаимосвязей [8, с. 233]. Она является важным инструментом для разработки и проектирования приложения, так как позволяет:

– визуализировать архитектуру приложения: структурная схема наглядно демонстрирует, как различные компоненты приложения взаимодействуют друг с другом;

– определить функциональные модули: структурная схема помогает разделить приложение на функциональные модули, каждый из которых отвечает за определенную задачу;

– облегчить разработку и тестирование: структурная схема служит ориентиром для разработчиков и тестировщиков, помогая им лучше понимать приложение и его работу;

– обеспечить масштабируемость: структурная схема должна быть разработана таким образом, чтобы приложение можно было легко расширять и

добавлять новые функции.

Рассмотрим общие элементы структурной схемы:

– модули – это основные функциональные блоки приложения;

– декомпозиция модулей – это декомпозиция модулей на более мелкие функциональные единицы [31, с. 44];

– данные – это информация, используемая приложением;

– интерфейсы – способы взаимодействия между модулями, подмодулями и данными;

– внешние системы – системы и сервисы, с которыми взаимодействует приложение.

Пример построенной структурной схемы представлен на рисунке 4.

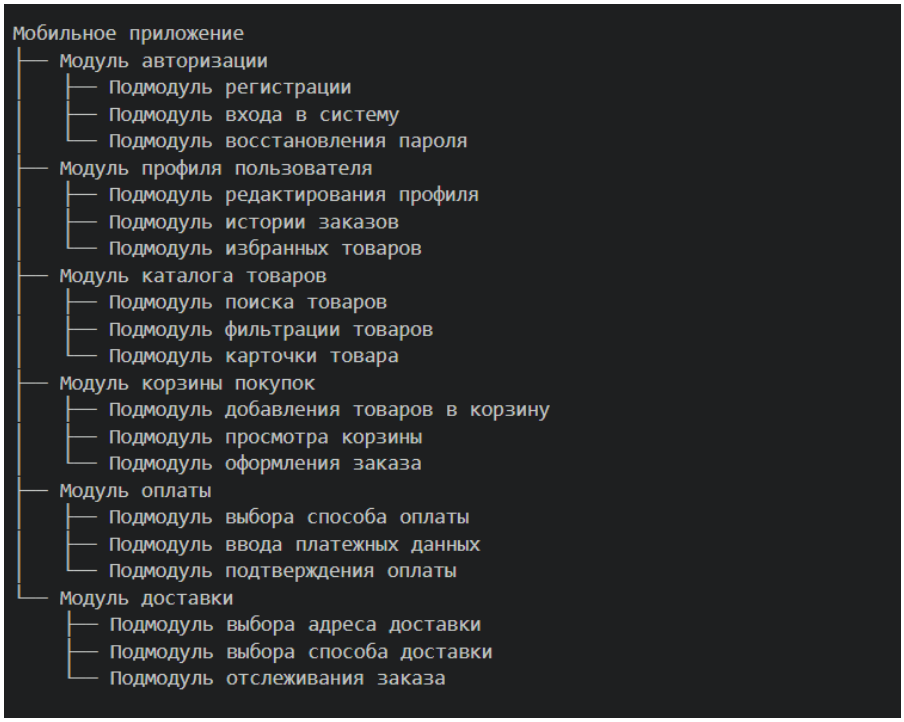

Рисунок 4 – Структурная схема по горнолыжным курортам красноярского Края

Пользователь взаимодействует с приложением через интерфейс пользователя (UI), который включает в себя различные экраны для доступа к функциям приложения. Интерфейс пользователя передает действия и запросы в

Логику приложения (Business Logic), где происходит обработка данных и выполнение бизнес-логики. Логика приложения взаимодействует с Сетевыми запросами (Networking) для получения данных от внешних API и с Данными (Data Layer) для хранения и извлечения информации. Сетевые запросы обеспечивают взаимодействие с внешними сервисами и API для получения актуальной информации о курортах, погоде, бронированиях и отзывах. Данные включают базы данных, в которых хранится информация о пользователях, курортах, трассах, мероприятиях, бронированиях и отзывах.

Структурная схема помогает четко представить архитектуру мобильного приложения, разделяя его на основные компоненты и модули, что упрощает процесс разработки, тестирования и поддержки приложения [38, с. 23].

В мобильной разработке API (Application Programming Interface) – это набор функций, методов и протоколов, предоставляемых операционной системой Android или сторонним сервисом [40, с 54].

API позволяют создавать мобильные приложения и взаимодействовать с различными функциями устройства и внешними сервисами.

Следующим этапом является создание диаграммы компонентов. Диаграмма компонентов – это статическое представление архитектуры мобильного приложения для горнолыжных курортов [31, с. 31]. Где показаны основные компоненты приложения, их взаимосвязи и зависимости. На диаграмме компонентов хорошо видна структура приложения. На диаграмме видно, что приложение состоит из пяти основных компонентов, которые взаимодействуют друг с другом через интерфейс.

Компоненты в схеме представляют пользовательский интерфейс (UI) отвечает за взаимодействие пользователя с приложением. Бизнес-логика реализует функциональность приложения. Управление данными (Data Management) работает с данными приложения. Сетевые запросы (Networking): обеспечивают связь приложения с сетью.

Внешние сервисы приложения – это сервисы, которые позволяют

приложению подключить дополнительную информацию, такие как новости, карты, взаимодействие с пользователем для улучшения и упрощения пользования приложением.

Диаграмма компонентов представлена на рисунке 5.

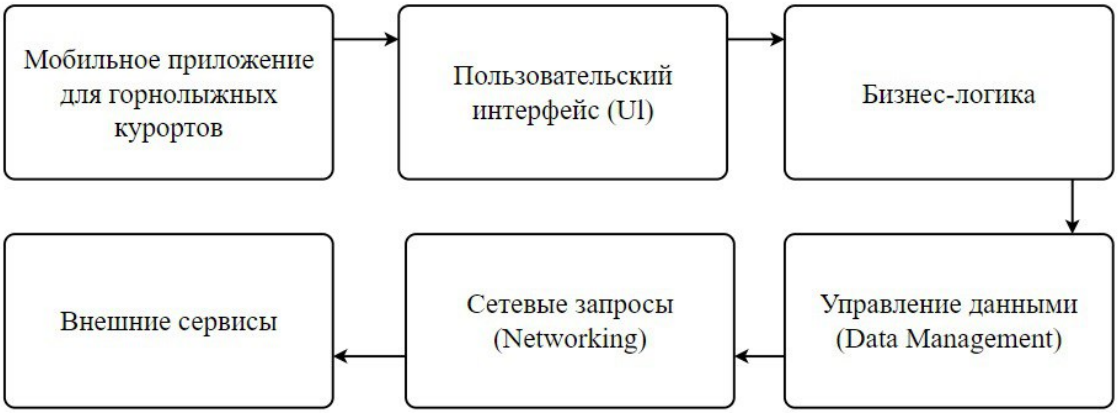

Рисунок 5 – Диаграмма компонентов

Структурная схема мобильного приложения представляет собой важный инструмент для разработки и проектирования приложения, позволяющий визуализировать его архитектуру и определить основные модули.

В ходе изучения и анализа различных аспектов разработки мобильного приложения для горнолыжных курортов Красноярского края были рассмотрены ключевые этапы проектирования, определены функциональные и нефункциональные требования к приложению, разработана структурная схема и описаны основные компоненты приложения.

2 Разработка и тестирование мобильного приложения

2.1 Выбор средств разработки

Структурная схема мобильного приложения представляет собой организацию его компонентов и взаимосвязей между ними. Для создания структурной схемы мобильного приложения по горнолыжным курортам Красноярского края.

Структурная схема мобильного приложения представляет собой иерархическое отображение его основных компонентов и их взаимосвязей. Она является важным инструментом для разработки и проектирования приложения, так как позволяет визуализировать архитектуру приложения, определить функциональные модули, облегчить разработку и тестирование [37, с. 34].

Что бы создать мобильное приложение для горнолыжных курортов Красноярского края необходимо провести анализ различных средств разработки. Для этого следует учитывать такие параметры как: поддержка различных платформ, удобство разработки, производительность приложения и возможности для интеграции с внешними сервисами.

Существующие решения такие как: React Native, Flutter, Xamarin, Native Development проанализированы и представлены в таблице 2.

Таблица 2 – Сравнительная таблица существующих решений

Критерии	**React Native**	**Flutter**	**Xamarin**	**Native Development**	**Android Studio**
Язык программирования	JavaScript (TypeScript)	Dart	C#	Платформа зависимый	Java, Kotlin
Уровень производительности	Высокий	Высокий	Высокий	Самый высокий	Высокий
Уровень сложности	Средний	Средний	Средний	Высокий	Низкий
Размер сообщества	Большой	Большой	Средний	Платформа зависимый	Большой
Набор функций	Богатый	Богатый	Богатый	Самый богатый	Богатый (Android)
Стоимость разработки	Относительно низкая	Относительно низкая	Относительно высокая	Самая высокая	Низкая (Android)

Окончание таблицы 2

Критерии	React Native	Flutter	Xamarin	Native Development	Android Studio
Поддержка платформ	iOS, Android	iOS, Android	iOS, Android Windows	iOS, Android, Windows	Android
Время разработки	Быстрое	Быстрое	Среднее	Медленное	Быстрое
Поддержка Hot Reload	Да	Да	Нет	Нет	Нет
Размер приложения	Средний	Средний	Средний	Платформа зависимый	Средний (Android)
Нативность	Высокая	Высокая	Высокая	Самая высокая	Высокая (Android)

React Native – это кроссплатформенная, быстро развивающееся решение с большим комьюнити, которое позволяет создавать мобильные приложения для Android используя JavaScript. Приложения, созданные с помощью React Native, обладают нативным интерфейсом и производительностью, близкой к нативным приложениям. React Native позволяет легко разделять проект на независимые модули, что упрощает разработку, тестирование и поддержку кода. Это достигается благодаря использованию нативных компонентов и оптимизации JavaScript-кода. Благодаря большому сообществу разработчиков и множеству готовых компонентов и библиотек, можно ускорить процесс разработки и быстро внедрить новые функции.

React Native поддерживает функции горячей перезагрузки и живого обновления, что позволяет мгновенно видеть изменения в коде без необходимости полной перезагрузки приложения [31, с. 67].

React Native является мощным инструментом для разработки кроссплатформенных мобильных приложений, у него есть множество преимуществ, таких как быстрая разработка, использование уже известных технологий и большая поддержка сообщества. Однако его использование может быть ограничено проблемами с производительностью, зависимостью от сторонних библиотек и сложностью в доступе к нативным функциям.

Рассмотрим следующий кроссплатформенный современный фреймворк Flutter от Google для создания мобильных приложений. Возможности Flutter

подходят для разработки приложений с пользовательским интерфейсом и высокой производительностью.

Flutter позволяет использовать единый код для создания приложений для iOS и Android, что существенно сокращает время и ресурсы, необходимые для разработки и поддержки.

Flutter предоставляет широкий выбор готовых виджетов для создания красивых и адаптивных пользовательских интерфейсов. Возможность мгновенно видеть результаты изменений в коде без полной перезагрузки приложения ускоряет процесс разработки и отладки.

Flutter является мощным и гибким решением для разработки кроссплатформенных мобильных приложений, предлагающий множество преимуществ, таких как высокая производительность, быстрая разработка и богатый набор виджетов. Однако его использование может быть ограничено некоторыми минусами, включая большой размер приложений и необходимость изучения нового языка программирования Dart.

Следующее рассмотренное решение – Xamarin, решение, которое позволяет разрабатывать мобильные приложения для iOS, Android и Windows с использованием одного кода на C#. Одним из главных плюсов Xamarin является полный доступ к нативным API и функционалу каждой платформы, что позволяет создавать высокопроизводительные приложения с использованием всех возможностей устройств. Но несмотря на высокую производительность, приложения, написанные на Xamarin, могут быть более медленными из-за огромного потребления ресурсов при выполнении задач и сложной графики. Однако Xamarin обеспечивает быструю разработку и упрощает поддержку кода, что особенно полезно для кроссплатформенных проектов.

Xamarin является хорошим и мощным инструментом для разработки кроссплатформенных мобильных приложений, предлагающим множество преимуществ, включая полную интеграцию с Visual Studio и доступ к нативным API. Однако его использование может быть ограничено проблемами с

производительностью и размером приложения [32, с. 78]. Несмотря на это, Xamarin остается популярным выбором для тех, кто ценит возможность использовать общий код для различных платформ и интеграцию с существующими C# проектами.

Следующее нестандартное решение – это Native Development (нативная разработка). Нативная разработка мобильных приложений предлагает высокую производительность, полный доступ к API платформы и лучший пользовательский опыт [31, с. 65]. Разработанные нативные приложения обеспечивают отличную производительность, так как они разрабатываются специально для конкретной платформы (iOS или Android), используя языки программирования и инструменты, предоставляемые соответствующей платформой, которую выбирает разработчик. Но главным минусом нативной разработки является время, затраченное на разработку, из-за необходимости создания отдельных версий приложения для каждой платформы, что увеличивает затраты на разработку и поддержку. Однако это позволяет максимально использовать возможности каждой платформы и оптимизировать производительность.

Нативная разработка мобильных приложений для Android предлагает высокую производительность и полный доступ к API платформы. Однако это требует больше времени и затрат на разработку и поддержку приложения для Android, что не подходит для разработки мобильного приложения для курортов Красноярского края. Тем не менее, нативная разработка может быть оправдана в случаях, когда требуется максимально высокая производительность и глубокая интеграция с функциями устройства.

Решением для разработки мобильного приложения будет мощная и удобная среда разработки Android Studio, которая обладает широким набором инструментов, позволяющих эффективно создавать и отлаживать приложения. Также в Android Studio интуитивно понятный интерфейс, который позволяет сосредоточиться на процессе разработки, а не на изучении среды разработки.

Дополнительным преимуществом является наличие встроенных инструментов для профилирования и анализа производительности приложений, что помогает разработчикам оптимизировать код и улучшать пользовательский опыт.

Таким образом, Android Studio является самым подходящим решением на основе проведённого анализа существующих решений, так как Android Studio самая удобная среда разработки для создания мобильных приложений под платформу Android. Она обладает множеством преимуществ, включая обширный набор инструментов разработки, мощные инструменты отладки и тестирования, а также интеграцию с Android SDK. Кроме того, Android Studio регулярно обновляется, предлагая новые функции и улучшения, что делает её отличным выбором для долгосрочных проектов.

2.2 Разработка модулей мобильного приложения

Создание мобильного приложения для горнолыжных курортов Красноярского края требует детальной проработки всех его компонентов, чтобы обеспечить функциональность, удобство и надежность использования.

Разработка мобильного приложение состоит из создания модулей для приложения, включает разработку отдельных функциональных блоков, которые могут работать как вместе, так и независимо друг от друга.

Разрабатываемое приложение разделено на модули:

1. модуль Регистрации в приложении;

2. модуль главного экран приложения;

3. окно пополнения баланса, состоящее из двух окон переход к пополнению и само окно пополнения;

4. модуль, который будет добавлен с помощью API и будет интегрировать карты в приложение;

5. модуль афиша, где пользователь сможет просматривать интересующие его мероприятия на курорте.

Разделение приложения на независимые модули упростит его разработку, тестирование и поддержку. Макеты разработанных модулей можно использовать повторно для разработки других модулей, что позволит экономит время. Так же будет проще обслуживание приложения, так как новые функции можно добавлять, ошибки исправлять и обновлять модули без затрагивания всей системы.

Кроме того, модульная структура позволяет легко масштабировать приложение, добавляя новые или изменяя существующие модули. Разные команды разработчиков могут работать над отдельными модулями одновременно, что ускоряет процесс разработки.

Для начала нужно следует определить, на какие функциональные блоки будет разделено приложение. Каждый модуль будет отвечать за определённую задачу. Затем следует установить интерфейсы для взаимодействия модулей друг с другом.

Следующим шагом для разработки служит код для каждого модуля с использованием выбранного языка программирования и среды разработки. После этого каждый модуль проходит тестирование, чтобы убедиться в его правильной работе. Затем все модули интегрируются в единое приложение, которое также тестируется на корректность работы.

Инструменты и примеры для разработки модулей можно использовать такие языки программирования, как Java, Kotlin и Objective-C, а также среду разработки Android Studio [23, с. 84].

Таким образом, модульная разработка мобильных приложений позволяет создать надёжные, масштабируемые и удобные в обслуживании решения.

Следующим этапом разработки является создание модуля регистрации пользователя.

Модуль регистрации пользователя играет ключевую роль в любом мобильном приложении, так как отвечает за создание и управление учетными записями пользователей. Этот компонент состоит из нескольких этапов и

функций, направленных на сбор, проверку и хранение данных пользователей [23 с. 12]. Давайте рассмотрим основные аспекты разработки модуля регистрации пользователя.

Важнейшей частью является форма регистрации, предоставляющая пользователю интерфейс для ввода данных, таких как имя, электронная почта, пароль, номер телефона и другие необходимые сведения. На этом этапе данные проходят валидацию: проверяются корректность и полнота введенной информации, включая формат электронной почты, длину пароля и уникальность имени пользователя.

Безопасность паролей обеспечивается их хешированием перед сохранением в базе данных. После успешной валидации создается учетная запись, и данные сохраняются в базе данных.

Дополнительным этапом является подтверждение регистрации: пользователю отправляется электронное письмо с уникальной ссылкой или кодом для активации учетной записи. Также предусмотрена функция восстановления пароля, включающая отправку инструкций по его сбросу.

Еще одной важной функцией является интеграция с внешними сервисами, такими как Google, через OAuth, что позволяет пользователям регистрироваться с помощью своих существующих аккаунтов. Необходимые требования, которые включают в себя:

– анализ требований: определяются необходимые данные для регистрации, требования к безопасности и функциональности, а также дизайн пользовательского интерфейса;

– проектирование: разрабатывается архитектура модуля, включая структуру базы данных для хранения учетных записей и проектирование интерфейса формы регистрации;

– реализация: создаётся форма регистрации с полями ввода и кнопками отправки, а также реализуется валидация данных на клиентской стороне. Разрабатывается API для обработки регистрационных данных, хеширования

паролей и сохранения данных в базу. На серверной стороне реализуется валидация данных, отправка подтверждающего письма и активация учетной записи;

– тестирование: все функции модуля проходят тестирование для проверки корректности валидации данных, безопасности хранения паролей и успешности регистрации и активации учетной записи;

– документация: создаётся документация для разработчиков и пользователей, описывающая API, структуру данных и инструкции по использованию;

– интеграция: модуль регистрации интегрируется с другими компонентами приложения, такими как модуль аутентификации и личный кабинет пользователя.

Модуль регистрации для приложения на Kotlin для Android представлен на рисунке 6. Листинг модуля – в приложении А.

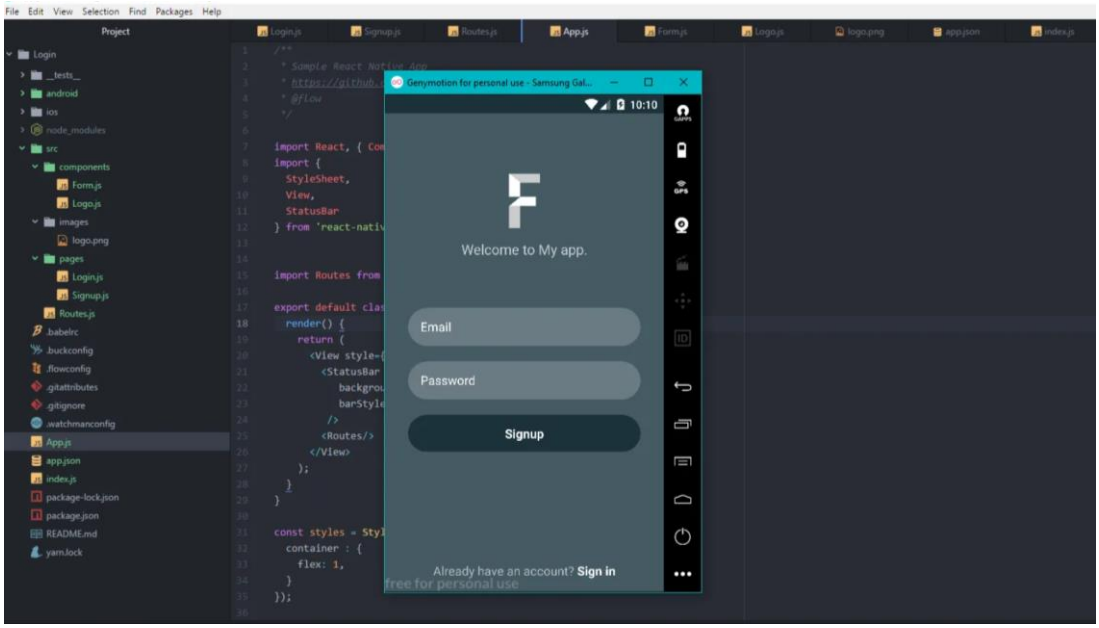

Рисунок 6 – Модуль «Регистрации для приложения»

Модуль регистрации пользователя является важным компонентом любого мобильного приложения. Его правильная реализация обеспечивает

безопасность, удобство использования и эффективное управление учетными записями пользователей.

После разработки модуля регистрации переходи к созданию модуля главного экрана.

Модуль главного экрана является центральной точкой взаимодействия пользователей с мобильным приложением. Он представляет собой главный интерфейс, через который пользователи получают доступ к основным функциям и данным приложения [34, с. 122].

Основные функции модуля главного экрана, это отображение ключевой информации: представление основной информации и данных, таких как новости, обновления, личные данные пользователя и информация о курортах:

1. навигация: обеспечение легкого доступа к другим разделам и функциям приложения через меню навигации, вкладки или кнопки;

2. персонализация: настройка отображения контента в зависимости от предпочтений и истории использования пользователя;

3. уведомления: информирование пользователей о новых событиях, сообщениях или обновлениях через push-уведомления или внутренние уведомления приложения;

4. интерактивные элементы: включение элементов управления, таких как кнопки, переключатели и списки, для взаимодействия с приложением.

Шаги разработки модуля главного экрана:

1. анализ требований: определение требований к функциональности и пользовательскому интерфейсу главного экрана на основе потребностей целевой аудитории;

2. проектирование: разработка макетов и прототипов интерфейса, включая расположение элементов, цветовую гамму и навигацию;

3. реализация: создание пользовательского интерфейса с использованием XML для Android или Storyboard. Реализация логики отображения данных и взаимодействия с пользователем. Разработка API для получения данных,

необходимых для отображения на главном экране. Интеграция с базой данных и другими сервисами;

4. тестирование: проведение функционального и пользовательского тестирования для обеспечения корректной работы и удобства использования главного экрана [32, с. 34];

5. документация: создание документации, описывающей функции и элементы главного экрана, а также инструкции по использованию и настройке;

6. интеграция: интеграция главного экрана с другими модулями и компонентами приложения.

Главный экрана мобильного приложения на Kotlin для Android представлен на рисунке 7, листинг страницы – в приложении Б.

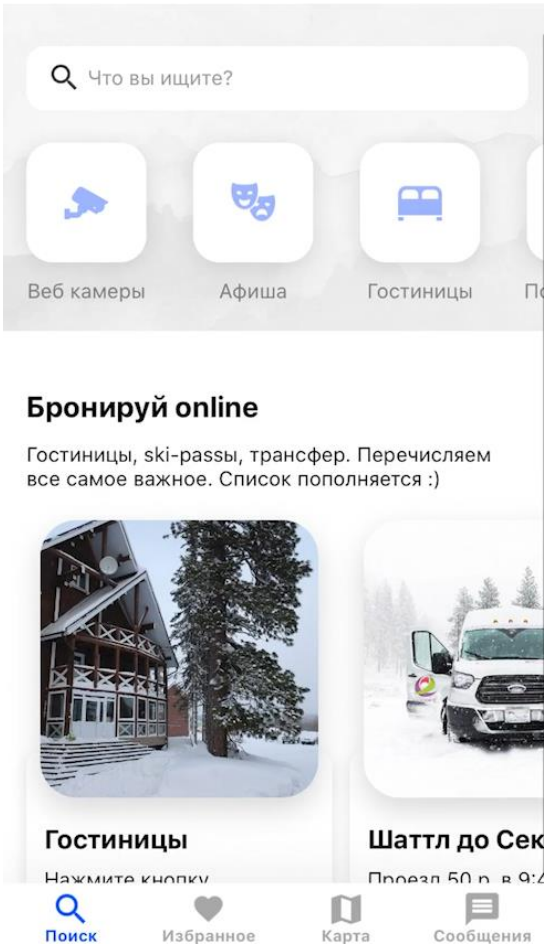

Рисунок 7 – Главный экран приложения

Модуль главного экрана – важная часть мобильного приложения, определяющая первое впечатление пользователя и предоставляющая доступ к ключевым функциям. Правильная разработка и реализация этого модуля обеспечивают удобство использования и эффективное взаимодействие с приложением.

Последним этапом разработки приложения является создание модуля оплаты в приложении.

Модуль пополнения ски-пассов является важной частью мобильного приложения для горнолыжных курортов, позволяя пользователям быстро и удобно пополнять свои ски-пассы через мобильное устройство. Этот модуль должен обеспечивать безопасные и надежные транзакции, а также быть удобным и интуитивно понятным для пользователей.

Основные функции модуля пополнения ски-пассов:

1. ввод данных ски-пасса: пользователь может ввести или сканировать номер своего ски-пасса;

2. выбор тарифа: выбор тарифа или пакета услуг для пополнения, таких как количество дней, время действия и другие опции;

3. подтверждение данных: просмотр и подтверждение введенных данных перед оплатой;

4. платежная система: интеграция с платежными системами для обработки транзакций, поддержка различных методов оплаты (кредитные карты, электронные кошельки и т.д.);

5. история транзакций: Возможность просмотра истории пополнений ски-пассов пользователем;

6. уведомления: информирование пользователя о статусе транзакции (успешное пополнение, ошибки и т.д.).

Шаги разработки модуля пополнения ски-пассов:

1. анализ требований: определение требований к функциональности модуля, методов оплаты и безопасности транзакций;

2. проектирование: разработка архитектуры модуля, проектирование пользовательского интерфейса и определение потоков взаимодействия пользователя;

3. реализация: создание пользовательского интерфейса с полями для ввода данных ски-пасса, выбора тарифа и подтверждения данных. Реализация взаимодействия с платёжной системой. Разработка API для обработки данных ски-пасса, тарифа и оплаты. Интеграция с платёжными системами для обработки транзакций. Обеспечение безопасности данных и транзакций;

4. тестирование: проведение функционального тестирования модуля, включая тестирование платежной системы и безопасности транзакций;

5. документация: создание документации, описывающей функции и элементы модуля, а также инструкции по использованию и настройке;

6. интеграция: интеграция модуля с другими частями приложения, такими как профиль пользователя и история транзакций.

Пример реализации модуля пополнения ски-пассов на Kotlin для Android продемонстрирован в приложении В и его реализация на рисунке 8-9:

Модуль пополнения ски-пассов – это важная часть мобильного приложения для горнолыжных курортов, обеспечивающая удобство и безопасность для пользователей.

Модуль пополнения ски-пассов – это не просто функциональная часть мобильного приложения, но и инструмент, повышающий удобство и удовлетворённость пользователей [35, с. 122].

Введение дополнительных функций и возможностей, таких как сканирование ски-пассов, интеграция с платёжными системами, поддержка скидок и промокодов, обеспечивает более комплексное и приятное взаимодействие с приложением.

Основная цель – создание интуитивно понятного, надёжного и безопасного модуля, который будет полезен и удобен для пользователей, а также позволит курортам эффективно управлять пополнением ски-пассов.

Модули «Переход но окно пополнения» и «Пополнения баланса ски-пасса» представлены на рисунке 8 и 9.

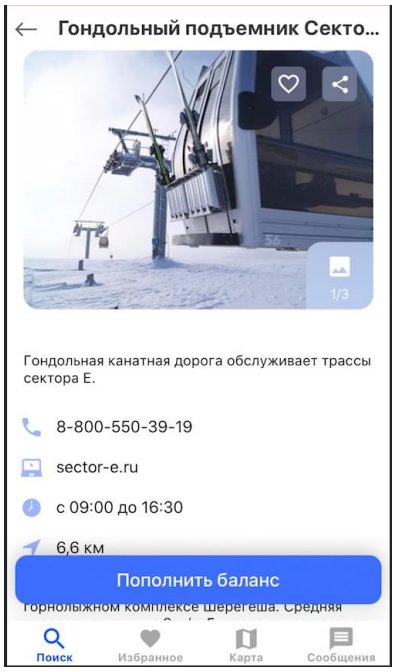

Рисунок 8 – Модуль «Переход на окно пополнения»

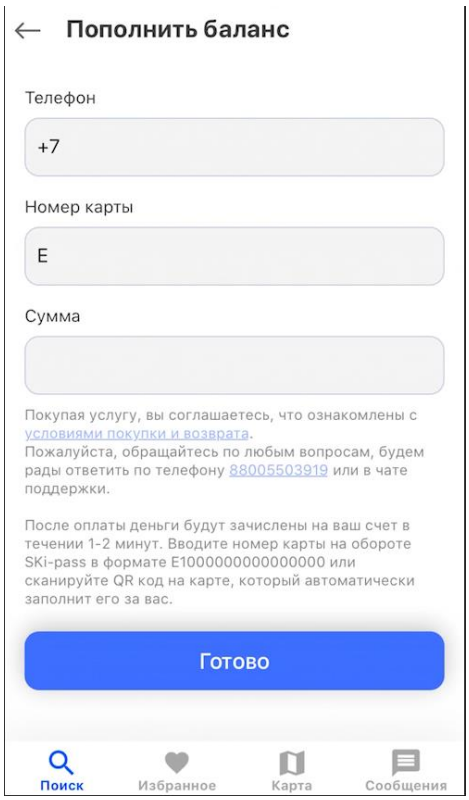

Рисунок 9 – Модуль «Пополнения баланса ски-пасса»

Следующим этапом является создание модуля интеграции в приложение карт Mapbox.

Модуль интеграции карт Mapbox в мобильное приложение для горнолыжных курортов является важным элементом, обеспечивающим визуализацию географической информации, навигацию и предоставление различных данных пользователям.

Mapbox – платформа для карт и геолокационных данных, которая предоставляет набор инструментов и API для создания настраиваемых карт, визуализации данных и обработки геолокационной информации. Mapbox используется разработчиками для создания интерактивных карт и приложений, которые могут отображать, анализировать и обрабатывать географическую информацию. Mapbox предоставляет мощный набор инструментов и API для работы с картами, что позволяет создавать интерактивные и настраиваемые карты [21, с. 34].

Mapbox – мощная и гибкая платформа для работы с картами и геолокационными данными, предлагающая широкий спектр инструментов для разработчиков. Она позволяет создавать настраиваемые и интерактивные карты, интегрировать их в веб- и мобильные приложения, а также обрабатывать и анализировать географическую информацию. Mapbox идеально подходит для различных применений, от навигации и логистики до маркетинга и управления городскими ресурсами [21, с. 122].

Основные функции модуля интеграции карт Mapbox: отображение карт, интеграция базовых карт Mapbox, таких как топографические карты, спутниковые снимки и пользовательские стили карт. Основные функции модуля интеграции карт Mapbox, такие как:

– маршруты и тропы: отображение лыжных трасс, маршрутов и троп, включая различную информацию, такую как уровень сложности, текущие условия;

– поиск и навигация: функции поиска мест, таких как подъемники, рестораны, туалеты и другие удобства. Маршрутизация и навигация по курорту, предоставление пошаговых указаний;

– метки и аннотации: добавление пользовательских меток и аннотаций на карту для отображения важной информации и интересных мест;

– реальное время: интеграция данных в реальном времени, таких как погода, статус подъемников и трасс, события;

– оффлайн режим: поддержка оффлайн карт, что позволяет пользователям загружать карты и использовать их без подключения к интернету.

Шаги разработки модуля интеграции карт Mapbox: анализ требований: определение требований к функциональности модуля, таких как типы карт, информация для отображения и особенности навигации. Регистрация в Mapbox: Создание аккаунта в Mapbox и получение доступа к API ключу, который будет использоваться в приложении. Шаги включают в себя:

– настройка проекта: Добавление зависимостей Mapbox в проект и настройка API ключа в конфигурационных файлах;

– реализация: создание пользовательского интерфейса для отображения карт, поиска мест и навигации. Разработка серверной части для обработки данных в реальном времени и интеграции с внешними сервисами (например, погодные данные);

– тестирование: проведение тестирования функций отображения карт, поиска и навигации, а также оффлайн режимов;

– документация: создание документации, описывающей использование модуля и инструкции по его интеграции и настройке;

– интеграция: интеграция модуля с другими частями приложения, такими как профиль пользователя и система уведомлений.

Таким образом, интеграция Mapbox в мобильное приложение для горнолыжных курортов позволяет создать мощный и функциональный инструмент, обеспечивающий высокое качество сервиса и удобство для

пользователей.

Модуль интеграции карт Mapbox на Kotlin для Android представлен на рисунке 10, листинг модуля – в приложении Г.

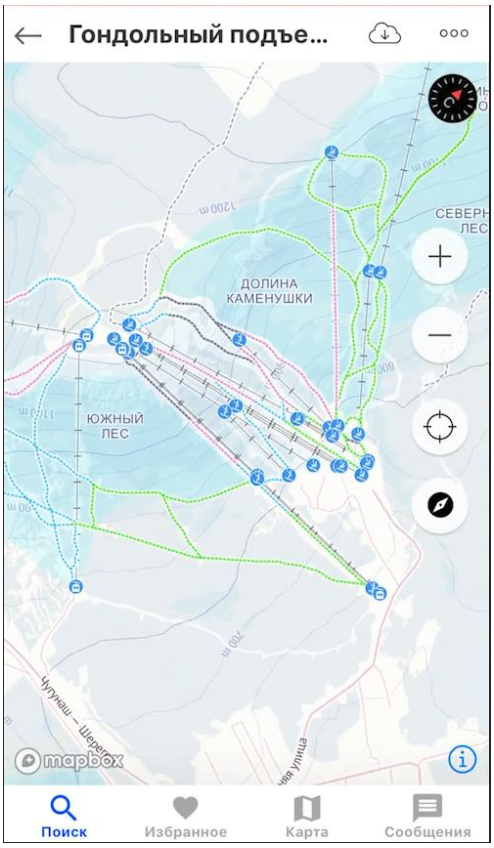

Рисунок 10 – Интеграция спутниковых карт через «Mapbox»

Модуль интеграции карт «Mapbox» – мощный инструмент для создания интерактивных и информативных карт в мобильном приложении для горнолыжных курортов.

Следующим этапом разработки является создание модуля Афиша

Модуль «Афиша» предназначен для отображения предстоящих событий и мероприятий, происходящих на горнолыжных курортах Красноярского края.

Он предоставляет пользователям информацию о различных событиях, таких как вечеринки, концерты, фестивали и другие мероприятия, которые могут быть интересны посетителям курорта.

Основные функции модуля «Афиша»: отображение списка мероприятий:

показ списка предстоящих мероприятий с краткой информацией, такой как название, дата, время и место проведения. Основные функции модуля «Афиша», такие как:

– детали мероприятия: возможность открыть подробную информацию о мероприятии, включая описание, фотографии, условия участия и контактные данные;

– бронирование и покупка билетов: ссылки на бронирование или покупку билетов на мероприятие;

– фильтрация и поиск: фильтры для поиска мероприятий по категориям, дате или местоположению. Поиск, по ключевым словам, для быстрого нахождения интересующих событий;

– добавление в избранное: возможность добавлять мероприятия в избранное для быстрого доступа к ним позже;

– напоминания и уведомления: настройка напоминаний о предстоящих мероприятиях и уведомления о новых событиях.

Шаги разработки модуля «Афиша»: анализ требований, определение типов мероприятий, которые будут отображаться, и информации, которая будет собираться и показываться пользователям. Шаги разработки модуля «Афиша» включают в себя:

– дизайн интерфейса: разработка удобного и привлекательного интерфейса для отображения списка мероприятий и их деталей;

– интеграция с базой данных: Настройка базы данных для хранения информации о мероприятиях. Реализация API для получения данных о мероприятиях и их обновления;

– создание пользовательского интерфейса для отображения афиши с использованием фреймворков, таких как React Native, Kotlin (для Android);

– реализация функций бронирования: интеграция с системами бронирования и оплаты;

– тестирование: проведение тестирования всех функций модуля, включая

отображение списка мероприятий, фильтрацию, поиск и бронирование;

– внедрение и поддержка: интеграция модуля в основное приложение. Регулярное обновление информации о мероприятиях и поддержка пользователей.

Модуль «Афиша» представлен на рисунке 11, листинг модуля – в приложении Д.

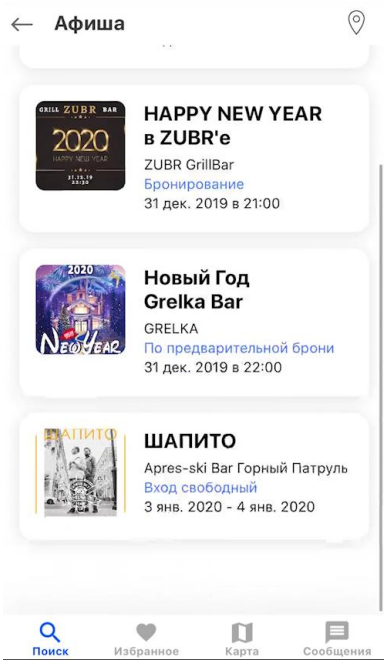

Рисунок 11 – Модуль «Афиша»

Модуль «Афиша» – важная часть мобильного приложения для горнолыжных курортов, обеспечивающая пользователей актуальной информацией о предстоящих мероприятиях и событиях. Его правильная реализация поможет улучшить опыт пользователей, привлечь их внимание к различным мероприятиям и увеличить вовлеченность в события, происходящие на курорте.

Последний этап разработки – модуль «Бары и рестораны».

Модуль «Бары и рестораны» предназначен для предоставления пользователям информации о местах общественного питания на горнолыжных

курортах Красноярского края.

Этот модуль поможет пользователям находить ближайшие бары и рестораны, узнавать о них подробную информацию, просматривать меню и бронировать столики.

Основные функции модуля «Бары и рестораны»:

– отображение списка баров и ресторанов: показ списка заведений с краткой информацией, такой как название, тип кухни, рейтинг, расстояние от текущего местоположения и часы работы;

– детали заведения: возможность открыть подробную информацию о заведении, включая описание, фотографии, меню, контактные данные и адрес;

– бронирование столиков: функция бронирования столиков через приложение или перенаправление на сайт заведения для бронирования;

– фильтрация и поиск: фильтры для поиска заведений по типу кухни, рейтингу, расположению и другим параметрам. Поиск, по ключевым словам, для быстрого нахождения интересующих заведений;

– отзывы и рейтинги: просмотр и добавление отзывов и рейтингов, чтобы пользователи могли делиться своим опытом и помогать другим выбирать лучшие заведения;

– карта: интеграция с картой для отображения местоположения заведений и построения маршрутов к ним;

– избранное: возможность добавлять заведения в избранное для быстрого доступа к ним позже.

Шаги разработки модуля «Бары и рестораны»: анализ требований, определение типов информации, которая будет собираться и отображаться о заведениях. Шаги разработки модуля включает в себя:

– дизайн интерфейса: разработка удобного и привлекательного интерфейса для отображения списка заведений и их деталей;

– интеграция с базой данных: Настройка базы данных для хранения информации о заведениях. Реализация API для получения данных о заведениях

и их обновления;

– создание пользовательского интерфейса для отображения списка баров и ресторанов с использованием фреймворков, таких как React Native, Kotlin (для Android) или Swift (для iOS).

Модуль «Бары и рестораны» для горнолыжных курортов Красноярского края предоставляет пользователям обширную информацию о местных заведениях питания, включая название, тип кухни, рейтинг, часы работы и расстояние. Он позволяет просматривать подробные данные о заведениях, включая описание, фотографии, меню, контактные данные и адрес, а также бронировать столики через приложение или на сайте заведения. Пользователи могут фильтровать и искать заведения по различным параметрам, просматривать и добавлять отзывы и рейтинги, использовать карту для нахождения и прокладки маршрутов, а также сохранять избранные заведения для быстрого доступа.

Модуль «Бары и рестораны» представлен на рисунке 12. Листинг модуля – в приложении Е.

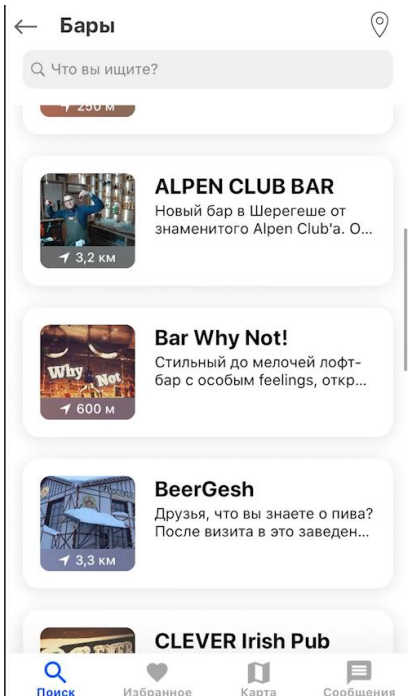

Рисунок 12 – Модуль «Бары и рестораны»

Таким образом, модульная структура мобильного приложения для горнолыжных курортов Красноярского края позволяет эффективно разрабатывать, тестировать и поддерживать его компоненты. Каждый модуль выполняет свою функцию: регистрация пользователей, главный экран, пополнение баланса ски-пассов, интеграция карт Mapbox, афиша мероприятий и информация о барах и ресторанах. Такая организация обеспечивает удобство в использовании, безопасность, масштабируемость и простоту обновлений, а также позволяет разным командам разработчиков работать параллельно над различными частями приложения.

2.3 Тестирование мобильного приложения

Тестирование мобильного приложения для горнолыжных курортов Краснодарского края – важная часть разработки. По словам А.Коротеевой, тесты позволяют проверить функциональность, надежность и соответствие требованиям.

Для тестирования мобильного приложения по горнолыжным курортам Красноярского края используются следующие шаги:

1. функциональное тестирование: убедиться, что приложение корректно отображает информацию о курортах, включая расписание работы, погодные условия, сведения о трассах и лифтах. Проверить работу функции поиска курортов и трасс по различным критериям, таким как расстояние от текущего местоположения, уровень сложности трассы и наличие определенных удобств. Проверить возможность просмотра интерактивных карт курортов с отображением трасс, лифтов, кафе и других объектов;

2. тестирование бронирования и покупки услуг: протестировать процесс бронирования и оплаты проката оборудования, уроков с инструктором, аренды гостиниц и других услуг, предоставляемых курортами. Проверить корректность расчета цен, доступность выбранных услуг и возможность выбора

дополнительных опций;

3. юзабилити-тестирование: оценить пользовательский интерфейс на предмет интуитивно понятности и простоты навигации. Убедиться, что все функции приложения доступны и понятны пользователям с разным уровнем опыта;

4. тестирование совместимости: убедиться, что приложение работает правильно на разных моделях мобильных устройств и операционных системах. Управляйте функциями с помощью различных разрешений экрана и ориентации устройства;

5. тестирование производительности: проверить скорость загрузки приложения и отображение информации. Проверьте работу приложения на низкой скорости Интернета или без сети;

6. тестирование безопасности: обеспечение безопасности пользовательских данных, особенно при оплате и предоставлении личной информации. В таблице 3 представлены результаты функциональных тестов мобильного приложения.

Таблица 3 – Проведение тестирования

Тест	**Описание**	**Результат**
Корректно отображает информацию о курортах	– Вся информация о курорте отображается корректно и полно – Изображения курорта подходят к текстовому описанию – Функции перехода работают корректно	Тест пройден без ошибок
Процесс бронирования и оплаты проката оборудования	– Процесс бронирования и оплаты завершается успешно без ошибок – Пользователь получает подтверждение о бронировании и оплате на экране приложения и по электронной почте – Зарезервированное оборудование отображается в соответствующем разделе приложения – Платежная информация и личные данные пользователя сохраняются конфиденциально и безопасно	Тест пройден без ошибок
Проверка безопасности	– Приложение обеспечивает высокий уровень безопасности для пользовательских данных и предотвращает уязвимости	Тест пройден без ошибок

Окончание таблицы 3

Тест	Описание	Результат
Проверка безопасности	– Не обнаружено никаких серьезных уязвимостей, которые могли бы привести к утечке данных или вредоносным атакам – Приложение соответствует стандартам безопасности и рекомендациям для разработки мобильных приложений	Тест пройден без ошибок
Работа с кроссплатформенностью	– Все основные функции приложения работают корректно на разных android устройствах – Пользовательский интерфейс приложения адаптирован и корректно отображается на устройствах с разными разрешениями экрана – Производительность приложения удовлетворительна на разных устройствах	Тест пройден без ошибок
Проверка скорости загрузки	– Время загрузки приложения разумно и не превышает предельно допустимое значение для данного типа приложения – Время загрузки приложения стабильно и не сильно меняться между разными запусками приложения – Приложение доступно для использования после окончания загрузки, без задержек или ошибок	Тест пройден без ошибок

Разработанное мобильное приложение для горнолыжных курортов Красноярского края успешно прошло все этапы тестирования и продемонстрировало высокий уровень качества и надежности.

Функциональное тестирование подтвердило корректную работу основных функций приложения, включая, бронирование услуг и отображение информации о курортах.

Тестирование кроссплатформенности подтвердило работоспособность приложения на устройствах с операционной системой Android, обеспечивая при этом одинаковый пользовательский опыт для разных устройств.

Проверка скорости загрузки подтвердила быструю и стабильную загрузку приложения на различных устройствах и с разными характеристиками.

Проверка безопасности подтвердила, что приложение обеспечивает надежную защиту конфиденциальных данных пользователей и предотвращает уязвимости, обеспечивая безопасное использование.

В целом, результаты тестирования свидетельствуют о высоком качестве

мобильного приложения для горнолыжных курортов Красноярского края и его готовности к выпуску и использованию конечными пользователями.

В ходе работы по разработке мобильного приложения для горнолыжных курортов Красноярского края выполнены все ключевые этапы: анализ предметной области, определение требований, разработка UML-диаграмм и структурной схемы, выбор средств разработки и реализация модулей, включая регистрацию пользователя, главный экран и оплату. Завершающим этапом стало тестирование, что позволило обеспечить надежность и стабильность работы приложения. Итогом является создание современного, удобного и функционального мобильного приложения, удовлетворяющего потребности пользователей и облегчающего взаимодействие с курортной инфраструктурой.

ЗАКЛЮЧЕНИЕ

В рамках данной выпускной квалификационной работы были исследованы и изучены теоретические основы разработки мобильных приложений для горнолыжных курортов Красноярского края.

В первой главе был проведен анализ предметной области, существующих приложений и определены требования к мобильному приложению. Это позволило выявить необходимые функции и особенности подобных систем. Кроме того, были разработаны UML-диаграммы и структурная схема приложения, что способствовало четкому представлению архитектуры системы.

Во второй главе были выбраны средства разработки и реализованы ключевые модули приложения, включая регистрацию пользователя, главный экран и оплату. Проведенное тестирование обеспечило надежность и стабильность работы приложения.

В результате работы было создано полноценное мобильное приложение для горнолыжных курортов Красноярского края, обладающее удобным интерфейсом и широким функционалом. Разработанное приложение отвечает современным требованиям и может быть успешно использовано в реальных условиях.

Разработка мобильного приложения стала ценным опытом, который позволил применить полученные знания и навыки на практике. Созданный проект имеет потенциал для дальнейшего развития и может стать основой для создания коммерчески успешного приложения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Агеев, П. Основы разработки мобильных приложений / П. Агеев. – Москва : Альфа-Пресс, 2018. – 432 с.

2. Андреев, К. Архитектура мобильных приложений / К. Андреев. – Санкт-Петербург : Питер, 2020. – 384 с.

3. Антонов, В. Программирование для Android / В. Антонов. – Москва : Лаборатория знаний, 2019. – 512 с.

4. Афанасьев, Д. Agile-разработка мобильных приложений / Д. Афанасьев. – Москва : ДМК Пресс, 2021. – 296 с.

5. Ахметов, Р. Android для профессионалов / Р. Ахметов. – Казань : Институт компьютинга, 2022. – 408 с.

6. Баранов, А. Разработка приложений для Android / А. Баранов. – Москва : Альфа-Пресс, 2021. – 432 с.

7. Белоусов, С. Программирование под Android: пошаговое руководство / С. Белоусов. – Санкт-Петербург : Питер, 2020. – 384 с.

8. Березин, В. Основы Android-разработки / В. Березин. – Екатеринбург : У-Фактория, 2019. – 352 с.

9. Богданов, Н. Практическое руководство по созданию Android-приложений / Н. Богданов. – Новосибирск : Сибирское университетское издательство, 2018. – 406 с.

10. Бурнет, Э. Привет, Android! Разработка мобильных приложений / Э. Бурнет. – Санкт-Петербург : Питер – 2012. – 253 с.

11. Гриффитс, Д. Head First. Kotlin / Д. Гриффитс. – Санкт-Петербург : Питер – 2020. – 646 с.

12. Гриффитс, Д. Head First. Программирование для Android / Д. Гриффитс. – Санкт-Петербург : Питер – 2018. – 912 с.

13. Дарвин, Я. Ф. Android. Сборник рецептов: задачи и решения для разработчиков приложений / Я. Ф. Дарвин. – Москва : Диалектика, 2019. – 768 с.

14. Дейтел, П. Android для разработчиков / П. Дейтел. – Санкт-Петербург : Питер, 2016. – 560 с.

15. Колисниченко, Д.Н. Самоучитель. Программирование для Android / Д. Н. Колисниченко. – Москва : БХВ, 2021. – 288 с.

16. Куликов, С. С. Тестирование программного обеспечения. Базовый курс / С. С. Куликов. – Беларусь : Минск, 2020. – 314с.

17. Лазарев, С. Разработка приложений под Android: полное руководство / С. Лазарев. – Москва : Альфа-Пресс, 2020. – 528 с.

18. Ларин, Д. Программирование для Android / Д. Ларин. – Санкт-Петербург : Питер, 2019. – 470 с.

19. Лебедев, И. Основы создания Android-приложений / И. Лебедев. – Москва : ДМК Пресс, 2021. – 384 с.

20. Лещенко, О. Практическое руководство по разработке на Android / О. Лещенко. – Новосибирск : Сибирское издательство, 2018. – 450 с.

21. Литвинов, А. Android для профессионалов / А. Литвинов. – Екатеринбург : У-Фактория, 2022. – 502 с.

22. Машнин, Т. С. Разработка Android – приложений в деталях / Т. С. Машнин. – Екатеринбург : Издательские решения, 2021. – 400 с.

23. Медведев, А. Основы программирования под Android / А. Медведев. – Новосибирск : Сибирское университетское издательство, 2018. – 340 с.

24. Морозов, П. Профессиональная разработка Android-приложений / П. Морозов. – Казань : Институт компьютинга, 2022. – 398 с.

25. Овчинников, И. Разработка мобильных приложений под Android / И. Овчинников. – Москва : Альфа-Пресс, 2020. – 420 с.

26. Уваров, А. Разработка мобильных приложений под Android / А. Уваров. – Москва : Альфа-Пресс, 2019. – 396 с.

27. Устинов, В. Программирование для Android / В. Устинов. – Санкт-Петербург : Питер, 2020. – 372 с.

28. Ушаков, Е. Основы создания Android-приложений / Е. Ушаков. – Екатеринбург : У-Фактория, 2021. – 360 с.

29. Удовенко, М. Android: практическое руководство для разработчиков / М. Удовенко. – Новосибирск : Сибирское университетское издательство, 2018. – 410 с.

30. Умнов, С. Профессиональная разработка на Android / С. Умнов. – Казань : Институт компьютинга, 2022. – 424 с.

31. Чайкин, А. Разработка приложений под Android / А. Чайкин. – Москва : ДМК Пресс, 2019. – 384 с.

32. Чалова, И. Основы программирования для Android / И. Чалова. – Санкт-Петербург : Питер, 2020. – 320 с.

33. Чебанов, В. Android для начинающих / В. Чебанов. – Екатеринбург : У-Фактория, 2018. – 288 с.

34. Чепурной, П. Профессиональная разработка на Android / П. Чепурной. – Новосибирск : Сибирское университетское издательство, 2021. – 412 с.

35. Чистяков, Е. Создание приложений под Android / Е. Чистяков. – Казань : Институт компьютинга, 2022. – 396 с.

36. Шалаева, Е. Разработка мобильных приложений на Kotlin / Е. Шалаева. – Москва : ДМК Пресс, 2021. – 368 с.

37. Шевчук, И. Основы программирования для iOS / И. Шевчук. – Санкт-Петербург : Питер, 2020. – 456 с.

38. Шелухин, А. Создание эффективных мобильных интерфейсов / А. Шелухин. – Москва : Альпина Паблишер, 2019. – 352 с.

39. Шипилова, Н. Введение в разработку мобильных приложений / Н. Шипилова. – Екатеринбург : У-Фактория, 2018. – 400 с.

40. Шумилов, О. Программирование на Swift для начинающих / О. Шумилов. – Новосибирск : Сибирское издательство, 2022. – 512 с.

ПРИЛОЖЕНИЕ А

Листинг модуля регистрации в мобильном приложении

```
class RegistrationActivity : AppCompatActivity() {
    private lateinit var etUsername: EditText
    private lateinit var etEmail: EditText
    private lateinit var etPassword: EditText
    private lateinit var btnRegister: Button
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_registration)
        etUsername = findViewById(R.id.etUsername)
        etEmail = findViewById(R.id.etEmail)
        etPassword = findViewById(R.id.etPassword)
        btnRegister = findViewById(R.id.btnRegister)
        btnRegister.setOnClickListener {
            val username = etUsername.text.toString().trim()
            val email = etEmail.text.toString().trim()
            val password = etPassword.text.toString().trim()
            if (validateInput(username, email, password)) {
                // Call backend API to register the user
                registerUser(username, email, password)
            }
        }
    }
    private fun validateInput(username: String, email: String, password: String):
Boolean {
        if (username.isEmpty() || email.isEmpty() || password.isEmpty()) {
            Toast.makeText(this, "All fields are required",
```

```
Toast.LENGTH_SHORT).show()
                return false
            }
            if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
                Toast.makeText(this, "Invalid email address",
Toast.LENGTH_SHORT).show()
                return false
            }
            if (password.length < 6) {
                Toast.makeText(this, "Password must be at least 6 characters",
Toast.LENGTH_SHORT).show()
                return false
            }
            return true
        }
        private fun registerUser(username: String, email: String, password: String) {
            // Code to call backend API and handle registration logic
        }
    }
```

ПРИЛОЖЕНИЕ Б

Листинг модуля главного экрана мобильного приложения

```
class MainActivity : AppCompatActivity() {
    private lateinit var tvWelcome: TextView
    private lateinit var btnViewProfile: Button
    private lateinit var btnViewNews: Button
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        tvWelcome = findViewById(R.id.tvWelcome)
        btnViewProfile = findViewById(R.id.btnViewProfile)
        btnViewNews = findViewById(R.id.btnViewNews)
        // Set welcome message
        val username = "User" // This should be retrieved from user data
        tvWelcome.text = "Welcome, $username!"
        btnViewProfile.setOnClickListener {
            // Navigate to Profile activity
            val intent = Intent(this, ProfileActivity::class.java)
            startActivity(intent)
        }
        btnViewNews.setOnClickListener {
            // Navigate to News activity
            val intent = Intent(this, NewsActivity::class.java)
            startActivity(intent)
        }
    }
}
```

ПРИЛОЖЕНИЕ В

Листинг модуля окна перехода и окно пополнения ски-пассов

```
class SkiPassTopUpActivity : AppCompatActivity() {
    private lateinit var etSkiPassNumber: EditText
    private lateinit var spinnerTariff: Spinner
    private lateinit var btnConfirm: Button
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_ski_pass_top_up)
        etSkiPassNumber = findViewById(R.id.etSkiPassNumber)
        spinnerTariff = findViewById(R.id.spinnerTariff)
        btnConfirm = findViewById(R.id.btnConfirm)
        // Setup tariff options (this should be dynamically loaded from server)
        val tariffs = arrayOf("1 Day - $50", "3 Days - $130", "7 Days - $300")
        val adapter = ArrayAdapter(this,
android.R.layout.simple_spinner_dropdown_item, tariffs)
        spinnerTariff.adapter = adapter
        btnConfirm.setOnClickListener {
            val skiPassNumber = etSkiPassNumber.text.toString().trim()
            val selectedTariff = spinnerTariff.selectedItem.toString()
            if (validateInput(skiPassNumber, selectedTariff)) {
                // Call backend API to process the payment and top-up
                processTopUp(skiPassNumber, selectedTariff)
            }
        }
    }
    private fun validateInput(skiPassNumber: String, selectedTariff: String):
Boolean {
```

```
            if (skiPassNumber.isEmpty() || selectedTariff.isEmpty()) {
                Toast.makeText(this, "All fields are required",
Toast.LENGTH_SHORT).show()
                return false
            }
            return true
        }
        private fun processTopUp(skiPassNumber: String, selectedTariff: String) {
            // Code to call backend API and handle top-up logic
        }
    }
```

ПРИЛОЖЕНИЕ Г

Листинг модуля интеграции карт в мобильном приложение

```
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import com.mapbox.mapboxsdk.Mapbox
import com.mapbox.mapboxsdk.maps.MapboxMap
import com.mapbox.mapboxsdk.maps.MapView
import com.mapbox.mapboxsdk.maps.OnMapReadyCallback
class MapActivity : AppCompatActivity(), OnMapReadyCallback {
    private lateinit var mapView: MapView
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // Initialize Mapbox
        Mapbox.getInstance(this, getString(R.string.mapbox_access_token))
        setContentView(R.layout.activity_map)
        mapView = findViewById(R.id.mapView)
        mapView.onCreate(savedInstanceState)
        mapView.getMapAsync(this)
    }
    override fun onMapReady(mapboxMap: MapboxMap) {
        // Configure map settings here
    }
    override fun onStart() {
        super.onStart()
        mapView.onStart()
    }
    override fun onResume() {
        super.onResume()
```

```
        mapView.onResume()
    }
    override fun onPause() {
        super.onPause()
        mapView.onPause()
    }
    override fun onStop() {
        super.onStop()
        mapView.onStop()
    }
    override fun onLowMemory() {
        super.onLowMemory()
        mapView.onLowMemory()
    }
    override fun onDestroy() {
        super.onDestroy()
        mapView.onDestroy()
    }
    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        mapView.onSaveInstanceState(outState)
    }
}
```

ПРИЛОЖЕНИЕ Д

Листинг модуля Афиша

```
class EventAdapter(private val events: List<Event>) : RecyclerView.Adapter<EventAdapter.EventViewHolder>() {
    class EventViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val title: TextView = itemView.findViewById(R.id.eventTitle)
        val date: TextView = itemView.findViewById(R.id.eventDate)
        val time: TextView = itemView.findViewById(R.id.eventTime)
        val location: TextView = itemView.findViewById(R.id.eventLocation)
        val image: ImageView = itemView.findViewById(R.id.eventImage)
    }
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): EventViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_event, parent, false)
        return EventViewHolder(view)
    override fun onBindViewHolder(holder: EventViewHolder, position: Int) {
        val event = events[position]
        holder.title.text = event.title
        holder.date.text = event.date
        holder.time.text = event.time
        holder.location.text = event.location
        // Load image with a library like Glide or Picasso
       Glide.with(holder.image.context).load(event.imageUrl).into(holder.image)
    }
    override fun getItemCount() = events.size
}
```

ПРИЛОЖЕНИЕ Е

Листинг кода модуля бары и рестораны

```
class RestaurantAdapter(private val restaurants: List<Restaurant>) : RecyclerView.Adapter<RestaurantAdapter.RestaurantViewHolder>() {
    class RestaurantViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val name: TextView = itemView.findViewById(R.id.restaurantName)
        val type: TextView = itemView.findViewById(R.id.restaurantType)
        val rating: TextView = itemView.findViewById(R.id.restaurantRating)
        val address: TextView = itemView.findViewById(R.id.restaurantAddress)
        val image: ImageView = itemView.findViewById(R.id.restaurantImage)
    }  override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RestaurantViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_restaurant, parent, false)
        return RestaurantViewHolder(view)
    }
    override fun onBindViewHolder(holder: RestaurantViewHolder, position: Int) {  val restaurant = restaurants[position]
        holder.name.text = restaurant.name
        holder.type.text = restaurant.type
        holder.rating.text = restaurant.rating.toString()
        holder.address.text = restaurant.address
        // Load image with a library like Glide or Picasso
Glide.with(holder.image.context).load(restaurant.imageUrl).into(holder.image)
    }
    override fun getItemCount() = restaurants.size}
```