Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»
ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ
КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

На правах рукописи
УДК 004.???

ГЕРИЛИВ РОМАН СТЕПАНОВИЧ

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ПЛАНИРОВАНИЯ ВЫПОЛНЕНИЯ ПРЫЖКОВ С ПАРАШЮТОМ

Выпускная квалификационная работа бакалавра

Направление подготовки 09.03.01 Информатика и вычислительная техника

Выпускная квалификационная
работа защищена
«___»_____2021 г.
Оценка _____
Секретарь ГЭК _____

г. Москва
2021

Студент-дипломник	_____	/ Герилив Р.С. /
Руководитель работы	_____	/ Овчаренко Е.С. /
Рецензент	_____	/ Хубаева Ю.Х. /
Заведующий кафедрой №12	_____	/ Иванов М.А. /

АННОТАЦИЯ

Выпускная квалификационная работа посвещена разработки мобильного приложения для планирования выполнения прыжков с парашютом.

Работа состоит всего из 90 листов. Пояснительная записка состоит из введения, трёх глав, заключения и приложения в виде ссылки на GitHub с приложением. Всего в пояснительной записке имеется 44 рисунка, 3 диаграммы, 4 таблиц и 8 формул.

В первой части проводится общий анализ тематики парашютного спорта. Проводится обзор используемых технологий у конкурирующих приложений и также обзор возможных к использованию при создании приложения. Обосновывается актуальность выбранной тематики в диаграммах путём количественного сравнения запросв в сети интернет за промежутки времени.

Во второй части определяются функциональные требования к приложению. Описывается стех используемых технологий и особенности каждой из выбранной технологии. Рассматривается проблематика определения физических динамических характеристик и выводится некоторые формулы для расчётов. Разрабатываются алгоритмы работы интерфейсов и их дальнейшего тестирования.

В третьей части пояснительной записки приводится результыт разработки. Деманстрируются скриншоты работающего приложения и приводится описание работы с активностями. Краско описан результат тестирования и отладки приложения по определённым заранее алгоритмам процессов тестировки.

В заключении подводятся итоги проделанной работы и финализируется информация о разработанном мобильном приложении.

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»

ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ

КАФЕДРА «КОМПЬЮТЕРНЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ»

Направление 09.03.01 Группа Б17-В71

«УТВЕРЖДАЮ»
Заведующий кафедрой
_____ / М.А. Иванов /
"____"_____2020 г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ (ДИПЛОМНЫЙ ПРОЕКТ)

Фамилия, имя, отчество студента: **Герилив Роман Степанович**

Тема работы: **Разработка мобильного приложения для планирования выполнения прыжков с парашютом**

Срок сдачи студентом готовой работы: **15 января 2021 г.**

Руководитель работы: **Овчаренко Евгений Сергеевич, старший преподаватель НИЯУ МИФИ, Отделение интеллектуальных кибернетических систем офиса образовательных программ (415) / Институт интеллектуальных кибернетических систем**

Место выполнения: **НИЯУ МИФИ**

1. Исходные данные:

Приложение предназначено для прогнозирования параметров прыжка с парашютом для дальнейшей оценки опасности. Приложение является мобильным блокнотом парашютиста с интерфейсной возможностью отметок количества прыжков.

Приложение должно:

1. Выполнять возложенный функционал по прогнозированию параметров прыжка.
2. Предоставлять доступ пользователю к данным о погоде.
3. Обеспечивать защиту данных от несанкционированного доступа.
4. Предлагать пользователю удобный web-интерфейс.
5. Выполнять имиджевый и информационный функционал.

2. Содержание задания:

а) обзорная часть:

Провести анализ функционала существующих мобильных приложений в сфере парашютного спорта.

б) расчетно-конструкторская, теоретическая, технологическая части:

1. Провести исследования стека используемых технологий.
2. Исследовать математическую модель прыжка.
3. Разработать структуру базы данных.
4. Разработать структуру программного обеспечения серверной части.
5. Разработать структуру программного обеспечения клиентской части.
6. Разработать структуру интерфейса пользователя.
7. Разработать план проведения тестирования.

в) экспериментальная часть:

1. Реализовать разработанную серверную часть приложения.
2. Реализовать разработанную клиентскую часть приложения.
3. Выполнить комплексное тестирование и отладку системы.
4. Составить руководство пользователя.

3. Основная литература:

1. Java Documentation – [Электронный ресурс]: https://docs.oracle.com/en/java/ (Дата обращения: 18.09.2020);
2. Машнин Т. С. Технология Web-сервисов платформы Java – СПб.: БХВ-Петербург, 2012. – 560 с. (Дата обращения: 17.09.2020);
3. Documentation for app developers – [Электронный ресурс]: https://developer.android.com/docs (Дата обращения: 18.09.2020);
4. Б. Брила – Oracle Database 11g. – М.: ЛитРес, 2019. – 864 с.;
5. Database Documentation – [Электронный ресурс]: https://docs.oracle.com/en/database/index.html (Дата обращения: 18.09.2020).

4. Отчетный материал:

пояснительная записка;

макетно-экспериментальная часть:

1. Листинги отлаженной программы на электронном носителе.
2. Дистрибутив приложения на электронном носителе.
3. Руководство пользователя.

Дата выдачи задания: 1 октября 2020 г.

Руководитель _____ / Е.С. Овчаренко /

Задание принял к исполнению _____ / Р.С. Герилив /

СОДЕРЖАНИЕ

СПИСОК ТЕРМИНОВ И СОКРАЩЕНИЙ

ОС – операционная система;

ПК – персональный компьютер;

ЭВМ – электронно-вычислительная машина;

ПО – программное обеспечение;

ООП – объектно-ориентированное программирование;

ACID – требования к транзакционной системе: Atomicity (Атомарность), Consistency (Согласованность), Isolation (Изолированность), Durability (Стойкость);

API - application programming interface – набор програмно реализованных способов взаимодействия различных классов, процедур, функций и т.д. с иной программой;

SQL (Structured Query Language) – это язык программирования структурированных запросов;

HTTP - англ. HyperText Transfer Protocol – «протокол передачи гипертекста» - протокол прикладного уровня передачи данных;

XML - англ. eXtensible Markup Language – расширяемый язык разметки;

БД – база данных;

СУБД - Система управления базами данных, (англ. Database Management System, сокр. DBMS) – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных;

Nickname – псевдоним пользователя;

Dropzone (дропзона) – термин, обозначающий площадку – аэродром, на котором выполняются прыжки с парашютом;

Активность – рабочая область приложения с различным интерфейсом.

ВВЕДЕНИЕ

Человек всегда стремился к познанию неизведанного. А начиналось такое познание с изучения окружающего нас мира – мира природы. Природа бесконечно может нас удивлять своими красками, формами жизни и их способностям. Такие способности могут зависеть от места обитания, способа существования и иного неисчислимого количества факторов окружающей среды. Если обратить внимание на животных, то их можно охарактеризовать по-разному – это травоядные или плотоядные животные, морские, сухопутные, земноводные или даже подземные. Эти животные различаются по своим скоростным характеристикам, возможностью выживать в экстремальных условиях. Человек всегда изучал особенности этих животных, но оставалось одно, чьи способности выходили за грань человеческого понимания – это птицы.

Желание человека летать можно проследить даже по древнегреческим мифам, например, миф об Икаре. Решение данной актуальной проблемы ставил перед собой Леонардо да Винчи ещё в 1495 году. Именно этим годом датируются записи в его рукописях о возможности безопасного спуска при помощи накрахмаленного полотна в форме пирамиды. Затем учёный Фауст Веранчино описывал подобную же конструкцию для безопасного спуска человека, уточняя, что размер полотна должен зависеть от тяжести самого человека, который данным устройством пользуется. Однако, по иронии судьбы первым, кому действительно пригодилось подобное устройство был француз Лавен – который бежал в начале 17 века из тюрьмы при помощи сшитых простыней, к концам которых были пришиты верёвки. Но первый аналог современного парашюта появился в 1777 году после испытания, в котором принял участие, приговорённый к смертной казни француз Жан Дёмье, после успешного испытания заключённому даровали жизнь. Далее человек принялся осваивать навигацию на воздушных шары, где парашюты уже получили действительно практическое применение, т.к. могли спасти жизнь своему носителю.

С развитием науки и технологий, люди стали понимать больше принципов аэродинамики, и конструкция парашюта со временем менялась и всё больше стала походить на современные модели круглых куполов. Подходя к тематике парашютного спорта, стоит отметить и первый прыжок с парашютом из самолёта, который совершил капитан Альберт Берри в штате Монтана 1 марта 1912 года, данный прыжок был удачным. Далее парашюты развивались как система для спасения лётчиков, которых становилось всё больше с развитием авиации. Для советской авиации над парашютами трудился Котельников. Первый в СССР спасательный парашют применил летчик-испытатель М.М. Громов 23 июня 1927 года на Ходынском аэродроме. Он преднамеренно ввел машину в штопор, из которого выйти не мог, и на высоте 600 м покинул самолет со спасательным парашютом, который был изготовлен из чистого шёлка, именно этот материал использовался как основной для парашютов в то время. А уже в 26 июля 1930 года под руководством Л.Г. Минова группа военных лётчиков выполнила первый групповой прыжок из многоместного самолёта [8]. Парашюты получили своё применение как в армии – воздушно-десантные войска, так и в гражданской сфере в виде экстремального вида спорта.

Основной целью моей дипломной работы является разработка мобильного приложения для статистического планирования выполнения прыжков с парашютом в максимально комфортных условиях. Под комфортными условиями здесь подразумевается скорее понимание парашютистом того, в каких условиях он будет выполнять прыжок. Также приложение является неким мобильным блокнотом (дневником) парашютиста с интерфейсной возможностью отметок количества прыжков и иных данных об этом. В приложении планируется реализация функционала для примерного определения максимальной скорости свободного падения при прыжках с различной высоты в различном положении парашютиста и расчётом критического времени раскрытия после отделения от самолёта. Несмотря на то, что у каждого парашютиста, который совершает прыжки с большой

высоты должен при себе находится высотомер, а также в парашютах есть системы принудительного раскрытия, оценка данного времени. Как мне кажется, будет весьма кстати перед прыжком.

Приложение должно реализовывать следующий функционал: имиджевый – привлекать новый контингент в парашютный спорт; практический – определение динамических характеристик прыжка с учётом внесённых параметров парашютиста и его системы и планируемого способа отделения от самолёта – или определение математической модели прыжка, удобность статистики прыжков за период; информационный – ознакомление с историей парашютного спорта и погружение в тематику этого направления.

В процессе разработки приложения была проведена большая работа по оценки оптимальных решений и программных средств. Результатом работы должно стать интуитивно понятное приложение в тематике парашютного спорта, которое действительно может быть полезным и будет заключать в себе весь необходимый функционал как для парашютистов любого класса и уровня подготовки, так и для людей, просто увлекающихся данной тематикой.

Парашютный спорт – это спортивная дисциплина, в которой человек или группа людей, экипированный(-ых) парашютом, отделяется от летательного аппарата (самолета, аэростата и т.п.) и в свободном падении или в планировании под куполом парашюта выполняет ряд упражнений, с последующим приземлением [8]. В самом свободном падении парашютист не может находиться слишком долго, однако уже есть факт прыжка из стратосферы, который выполнил австрийский скайдайвер Феликс Баумгартнер, развив сверхзвуковую скорость. Однако, средняя скорость парашютиста, что будет демонстрироваться в третьей части пояснительной записки, варьируется от 50 до 100 м/с, при такой скорости парашютист в свободном падении может передвигаться в воздухе управляя потоком воздуха, меняя положение рук или ног. Если парашютист прыгнет с высоты 4 километра, то без учёта веса его парашютной системы и его собственного веса,

так же, как и особенностей одежды и иных атмосферных явлений, на падение уйдёт всего 60 секунд до момента безопасного раскрытия парашюта [8].

Все скриншоты рабочего приложения, которые будут продемонстрированы в пояснительной записке представлены с реального смартфона. Начальная отладка и тестирование проходили на дополнительно настроенном в Android Studio эмуляторе смартфона на платформе Android.

1. ОБЗОРНАЯ ЧАСТЬ

В обзорной части будет проведёт анализ имеющихся приложений, связанных с парашютным спортом. Будут рассмотрены различные мобильные сетевые и офлайн приложения, которые являются конкурентами текущей мобильной разработке или сходны своим функционалом. Необходимо проанализировать то, как устроены данные приложения, оценить решения в интерфейсной части, обозначить и улучшить для своей разработки необходимый набор решений из конкурентных приложений.

Немаловажной частью любого приложения является его архитектурная организация – то есть то, как связаны между собой все технические и логические составляющие. Обзор применяемых для создания мобильных приложения архитектур также будет входить в состав обзорной главы дипломного проекта.

Наравне с архитектурой, выбираемой для разработки мобильного приложения, необходимо определиться со стеком используемых технологий. К подобным технологиям относятся языки программирования (низкоуровневые и высокоуровневые). К технологиям также относятся базы данных, которые делятся на реляционные и не реляционные базы данных. СУБД – системы управления базами данных, среди которых также необходимо выбрать оптимальную для реализации мобильного приложения.

И последняя, крайне необходимая часть обзорной главы – это обоснование актуальности разработки мобильного приложения в выбранной тематике. В качестве основной оценки актуальности будет проведён анализ популярности парашютного спорна и направления в целом через количественную оценку запросов в сети интернет.

Далее предлагаю вашему вниманию детализацию всех вышеперечисленных этапов обзорной части дипломного проекта по теме: «Разработка мобильного приложения для планирования выполнения прыжков с парашютом».

1.1.Анализ функционала существующих мобильных приложений по тематике парашютного спорта

С приходом мобильных технологий и огромного множества инструментов для их создания стали появляться приложения, значительно упрощающие нашу жизнь. Изначально такие приложения использовались на персональных компьютерах (далее ПК). Это различные сложные приложения статистические приложения, приложения для контроля различных системных параметров. Сфер, в которых необходимо применения и создания приложений мониторинга или иного контроля за показателями бесчисленное множество. С усовершенствованием технологий встал вопрос мобильности тех устройств, с помощью которых будет производится тот или иной мониторинг. И огромное множество приложений были перестроены под мобильный формат, т.е. теперь ими удобно можно пользоваться с любого смартфона. И это очень удобно, поэтому такую сферу, как парашютный спорт, не могла обойти стороной тенденция на создание мобильных приложений, в которых можно отслеживать свою статистику по прыжкам и вести иную удобную и собранную в одном месте информацию.

В текущем разделе я рассмотрю основной функционал и принципы работы некоторых приложений, которые являются конкурентами для моей разработки. Однако, хочу заметить, что стек технологий, который используется в этих приложениях открыт частично или скрыт полностью их автором, с целью предотвращения копирования приложения и его решений. Основной целью данного анализа будет оценка интуитивности приложений и их практического функционала. Результат анализа будет представлен во второй части пояснительной записки (пункт 2.1 и 2.7) в виде определения необходимого функционала приложения и различных решений в разработке интерфейса.

Первым рассматриваемым конкурентом будет иностранное приложение *Boogie – Skydiving logbook*. Страной происхождения данного приложения является США, там же находится и штаб-квартира Boogie. Данное приложение

адаптировано под различные операционные системы и является кроссплатформенным. Кроссплатформенность даёт большой перевес в такой категории как удобность для пользователя. Ведь сегодня у пользователя может быть смартфон с ОС Android, а завтра он приобретёт себе смартфон на базе iOS, а решение с кроссплатформенностью позволит сохранить все данные пользователя и пользоваться ими с различных устройств, включая ПК и ноутбуки. Приложение *Boogie – Skydiving logbook* является клиент-серверным, имеет возможность регистрации и хранения информации о пользователе. Есть возможность выгрузки статистических данных из приложения в Excel файл. Внутри себя приложение имеет несколько активностей, часть из которых будет продемонстрирована на рисунках ниже [9]:

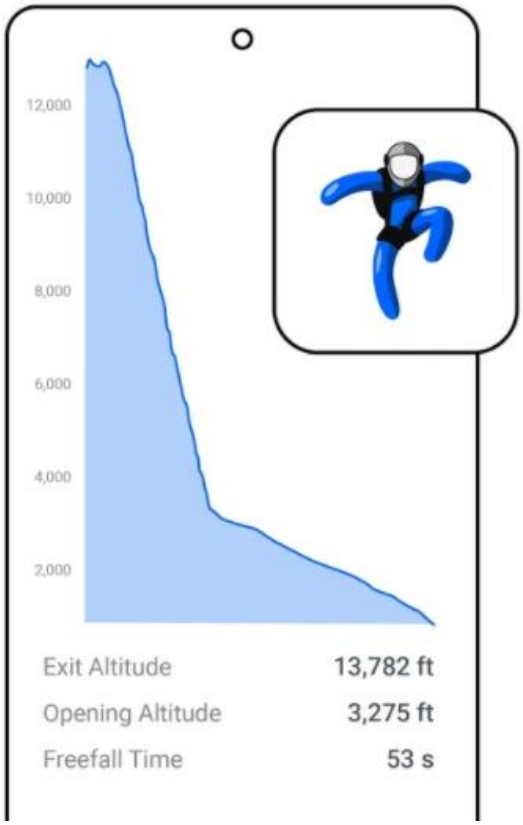

Рисунок 1.1 Пример демонстрации динамики прыжка из приложения Boogie – Skydiving logbook

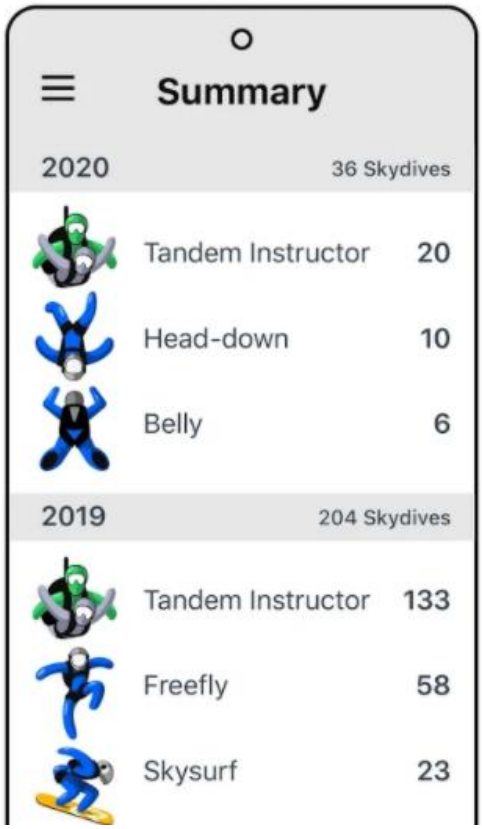

Рисунок 1.2 Пример демонстрации статистики парашютиста по количеству и типам совершённых прыжков

Приложение имеет приятный интерфейс с обилием различных иконок в тематике парашютного спорта. В приложении можно видеть динамику прыжка, однако у данной функции есть необходимость в дополнительном оборудовании, а именно: необходим электронной высотомеру, а сама функция будет доступна уже после прыжка, но является отличным примером интеграции технологии в спорт (рисунок 1.1). Есть возможность, как видно из рисунка 1.2, отмечать количество совершённых прыжков с указанием типа их исполнения в разрезе года. Это довольно интересный и правильный подход. Однако, у приложения есть небольшой недостаток – это отсутствие руссификации, и по отзывам русскоязычных пользователей именно это отталкивает их от пользования данным приложением.

Вторым рассматриваемым конкурентом будет приложение со сходным с предыдущим названием: *Skydiving Logbook*. Это также иностранное мобильное приложение, которое не имеет русского языка в своём интерфейсе.

Приложение предоставляет следующие возможности своему пользователю [10]:

- Запись прыжков: №, дата, дропзона, воздушное судно, снаряжение, тип прыжка (на животе/групповая акробатика, фрифлай и т. п.), высота отделения/раскрытия (в футах или метрах), задержка, отцепка, заметки и диаграмма;
- Лицензированные парашютисты могут подписывать записи прыжков в журнале;
- Просмотр статистики общего количества прыжков, общего времени падения и отцепок;
- Управление снаряжением, включая компоненты/серийные номера и напоминания о техобслуживании (например, переукладка на следующей неделе);
- Управления дропзонами, с возможностью установки домашней;
- Управление воздушными суднами (около дюжины уже встроено в программу);
- Расчёт загрузки крыла, желаемого размера купола и дополнительного веса для достижения желаемой загрузки;
- Импорт и экспорт для резервного копирования, передачи и печати Вашего журнала;
- Раздел истории для ввода предыдущего времени падения и количества отцепок.

Как можно видеть, приложение несёт в себе большой функционал разной направленности. Приложение не имеет версии доя ПК, однако, также есть версии под ОС Android и iOS. *Skydiving Logbook* – это клиент-серверное приложение с возможностью регистрации, хранения и резервного копирования информации. Однако в приложении нет такой же интересной функции интеграции с внешним устройством, как в ранее рассмотренном. Зато есть возможность добавлять и хранить в облаке фотографии с прыжков, не

перегружая память смартфона. Далее, на рисунках 1.3 и 1.4 будут предоставлены скриншоты интерфейса мобильного приложения:

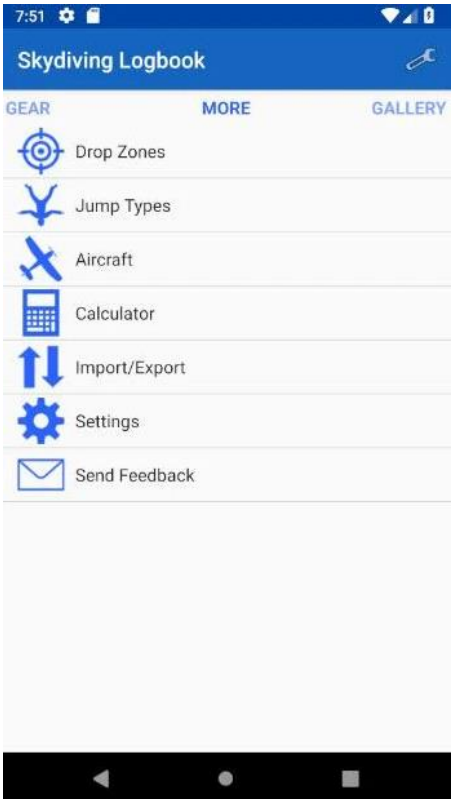

Рисунок 1.3 Главное меню приложения Skydiving Logbook

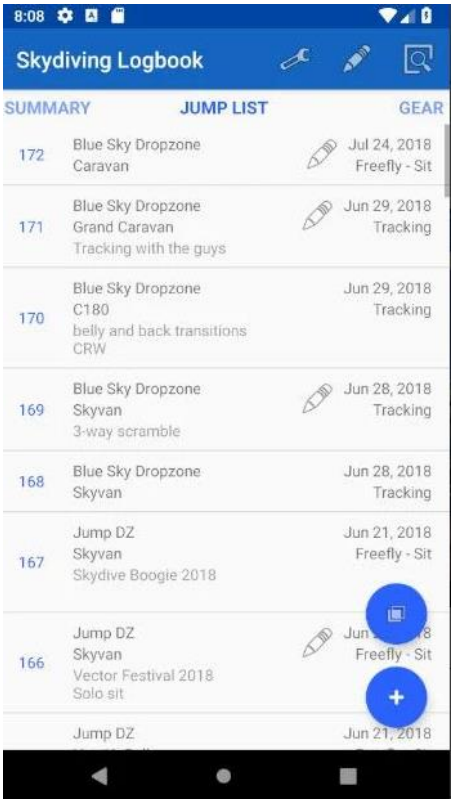

Рисунок 1.4 Список совершённых прыжков с пометкой о количественном номере прыжка

Приложение имеет интуитивно понятный интерфейс, не перегружено иконками, цветовая гамма выдержанная, но простая. Также плюсом является возможность выгрузки данных о прыжках в отдельный документ.

Третьим рассматриваемым приложением будет обучающее приложение *Rhythm Skydiving 401*. Также иностранное приложение, не имеет перевода на русский язык, имеет версии для ОС Android и iOS и несёт в себе следующий функционал [10]:

- Статьи и иная информация о том, как научиться различным способам отделения, как формировать и планировать команду, советы по выступлениям, техники бодифлайта;
- Планы непрерывности ритма с цветовой кодировкой слота для новичков и продвинутые блочные методы, включая ключевые промежуточные изображения различных фигур;
- Интерактивные флеш-карты, которые помогут вам изучить построения фигур в воздухе.

Приложение является довольно интересным для профессионалов в парашютном спорте, но будет абсолютно бесполезно для новичков. Данное приложение будет удобно использовать перед планированием групповых затяжных прыжков с парашютом, но оно не имеет иного функционала, не имеет возможности регистрации и хранения необходимой информации на сервере. Вся необходимая информация хранится именно на клиенте с возможностью перехода по вложенным ссылкам на полезные статьи. Имеет интуитивно понятный интрефейс, пример которого будет предоставлен на рисунке 1.5 ниже:

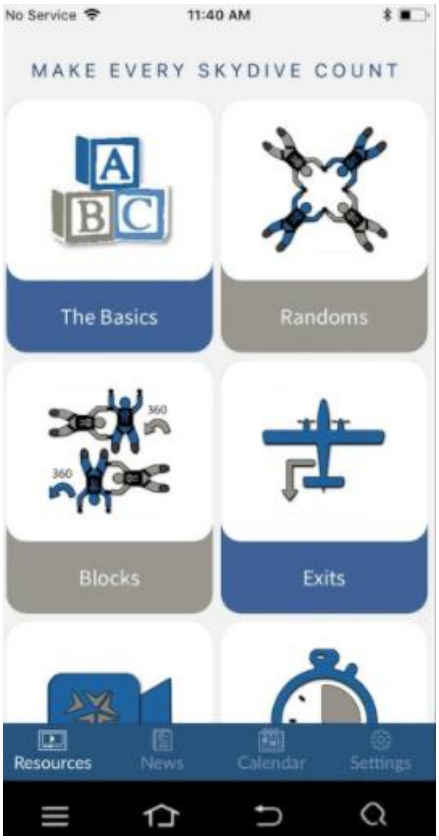

Рисунок 1.5 Главное меню приложения Rhythm Skydiving 401

Четвёртым и последним рассматриваемым конкурентом будет российское приложение «*Моделирование движения парашютиста до этапа раскрытия парашюта*».

Данное приложение рассчитано для использования только на ЭВМ. Язык программирования, на котором написано приложение – Maple (программный пакет, система компьютерной алгебры, продукт компании Waterloo Maple Inc.). Из описания приложения следует, что оно предназначено для моделирования движения парашютиста на трёх этапах его прыжка: первый – падение с отделением его от самолёта, второй – снижение на стабилизационном парашюте, третий – момент наполнения основного купола. Программой производится расчёт скорости парашютиста, траектории полёта и оказываемой нагрузки. Основные расчёты проводятся в рамках системы линейных и нелинейных дифференциальных уравнений на основе реальных исследований движения парашютистов. В программе необходимо задавать начальные условия для прыжка, такие как начальная скорость и высота. Программа представлена только на русском языке. К сожалению, не удалось

найти полноценной программы для оценки её функционирования, но ниже, на рисунке 1.6, будет предоставлен скриншот результирующего графика из научной статьи авторов программы [11]:

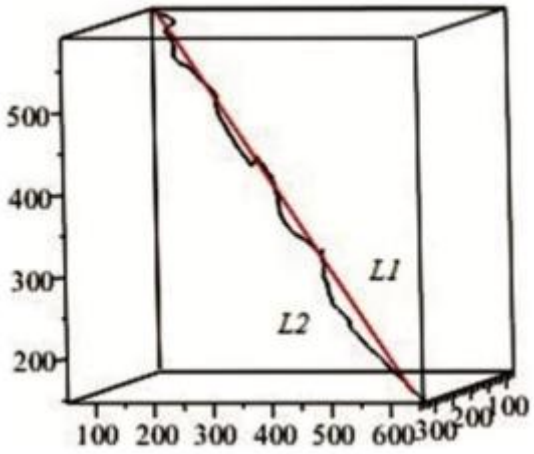

Рисунок 1.6 Фактическая (кривая) и теоретическая (прямая) траектория движения парашютиста

В ходе анализа было рассмотрено четыре различных приложения в тематике парашютного спорта и предназначенных для различных целей использования и имеющих разные степени удобности и актуальности для парашютистов разного уровня подготовки.

Опираясь на полученные знания о структурах и интерфейсах, имеющихся мобильных и стационарных приложений, теперь стало возможным определить минимальный набор к функционалу и возможной компоновки разрабатываемого мобильного приложения (подробно об этом будет рассмотрено в пункте 2.1).

1.2.Обзор архитектуры для реализации приложения

Перед началом разработки любого приложения или даже устройства необходимо проанализировать все процессы, которые необходимо будет реализовать в приложении. Такие процессы можно разделить на требования, а именно: технологические требования и бизнес требования. Технологические требования складываются из сферы применения приложения, необходимости интеграций с другими приложениями или устройствами, из развёртываемости

приложения на различных устройствах – к этому требованию как раз можно отнести опоминающуюся ранее кроссплатформенность, а также к технологическим требованиям относится специфика функционала необходимого для реализации. Именно от такого функционала и будет зависеть основной стек технологий, который будет использоваться далее. А что же такое бизнес требования? Если у проекта нет какого-либо заказчика, и сам проект является индивидуальной целью разработчика, то в любом случае, есть сфера практического применения у данной разработки – это и будет бизнес требованием к приложению. В настоящий момент, есть чёткое понимание о том, что приложение, которому посвящён дипломный проект – является мобильным приложением в тематике парашютного спорта, а значит появляются довольно чёткие рамки по, как минимум, необходимому подходу для разработки компактного и удобного интерфейса для пользователя. Но всё ещё стоит вопрос о выборе архитектуры разрабатываемого приложения. Ведь от этого выбора будет зависеть и стек технологий, и расширяемость приложения, и дальнейшая его оптимизация и возможности других разработчиков внести свою лепту в проект.

И вот, наконец, необходимо обозначить некоторые из существующих подходов в архитектурной реализации приложения. Именно с анализа и возможных шаблонов архитектур начинается любая разработка – это так называемый каркас приложения, в рамках которого будет проходить вся разработка.

Что же такое архитектурный шаблон приложения? **Архитектурный шаблон** – это обобщённое решение той или иной потребности в разработке с возможностью переиспользования в различных проектах и с различным стеком технологий [13].

Я буду рассматривать четыре более интересных на мой взгляд архитектурных шаблона, результатом данного обзора будет выбор конкретной архитектуры для разработки мобильного приложения во второй части

пояснительной записки и подробное описание реализации этой архитектуры в приложении в третьей части пояснительной записки.

Многоуровневый шаблон – этот шаблон используется для компоновки или структурирования программ и подпрограмм, которые можно разделить на группы различных подзадач, находящихся на определённых уровнях абстракции. Каждый подобный слой предоставляет службы для более высокого слоя и так далее. Обычно выделяют следующие слои: слой представления (интерфейсы), слой приложения (сервис), слой бизнес-логики (область применения) и слой доступа к данным (базы данных). Такой шаблон может применяться в различных приложениях, предназначенных для ПК и иных web-приложениях. На рисунке ниже будет представлена общая схема шаблона.

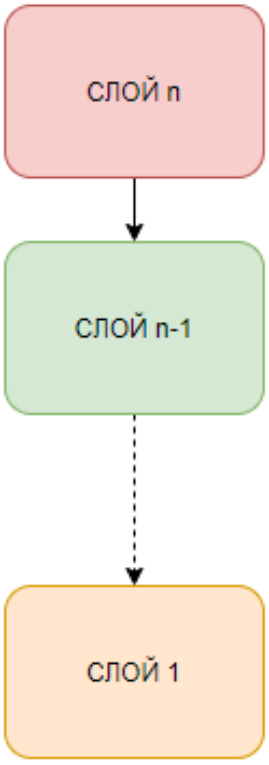

Рисунок 1.7 Общая схема архитектурного «многоуровневого» шаблона

Клиент-серверный шаблон – крайне популярный шаблон. Состоит из двух частей: сервера и клиента, при этом клиентов может быть большое количество множеств. Серверная часть данной архитектуры предоставляет службы клиентским компонентам, а доступы для клиентов, можно

ограничивать или в принципе закрывать. Со стороны клиентской части должен поступить запрос на необходимость доступа к той или иной службе, и в случае положительного call back (обратной связи с сервером), клиент получает запрашиваемый ресурс или активность. В зависимости от реализации серверной или клиентской части может меняться взаимодействие между этими сторонами, но основополагающий принцип – запрос-ответ будет работать при подобной архитектуре всегда. Клиент-серверный шаблон можно использовать в любых сетевых мобильных или десктопных приложениях. Из банальных примеров подобных приложений: электронная почта, социальные сети и подобные им приложения с возможностью регистрации, общения и возможности обмениваться через систему файлами. На рисунке ниже будет представлена общая схема шаблона.

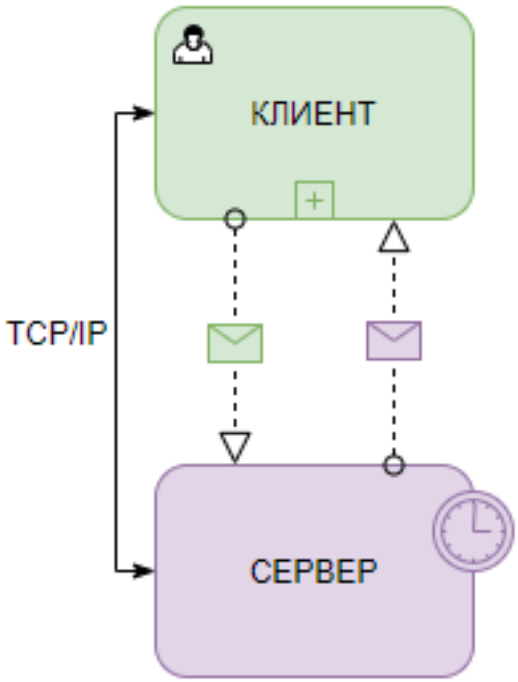

Рисунок 1.8 Общая схема «клиент-серверного» архитектурного шаблона

Шаблон ведущий-ведомый – как следует из названия этого шаблона, для его реализации необходимо два участника: ведущий и ведомый. Ведущий элемент архитектуры выполняет задачу распределения задач между ведомыми элементами и вычисляет финальный результат на основании всех полученных.

Подобный шаблон используется в репликации баз данных, при условии, что есть некая основная (главная) БД, которая является первоисточником для всех синхронизируемых с ней баз данных, подобное явление чаще можно встретить при работе с нереляционными базами данных, при осуществлении перегрузки данных (обогащённых или нет) из источника в иной рабочий слой посредством ETL потоков. В качестве примера в аппаратной среде, можно привести любые периферийные устройства, которые связаны с шиной компьютера. На рисунке ниже будет представлена общая схема шаблона.

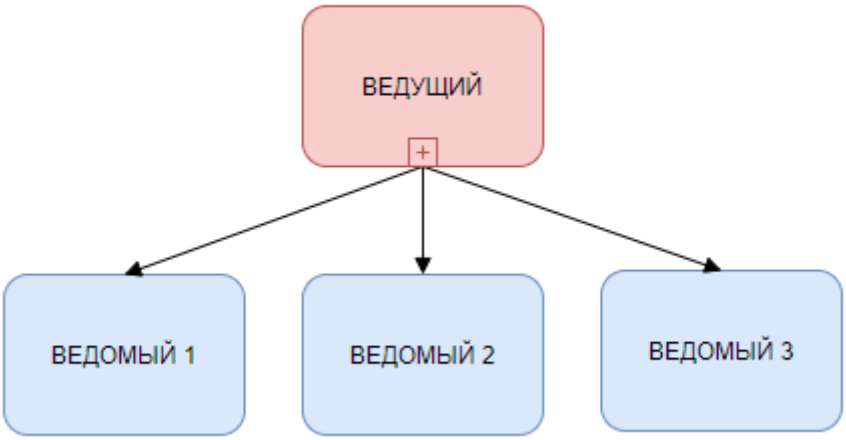

Рисунок 1.9 Общая схема архитектурного шаблона «ведущий-ведомый»

Шаблон посредника – данный шаблон необходим для возможности структурировать распределённых систем с разрозненными и несвязанными друг с другом компонентами. Взаимодействие компонентом может осуществляться, например, через удалённый вызов службы. Компонент, именуемый посредником (Broker), реализует взаимодействие и координацию всех компонентов друг с другом, предоставляя на клиент полноценный набор возможностей. Сервер, которых может быть несколько, передаёт свои компоненты в управление посреднику, а далее происходит процесс схожий с клиент-серверной архитектурой: клиент отправляет запрос уже не на сервер, а брокеру, который в свою очередь перенаправляет клиента к необходимой службе или компоненту на нужный сервер. Могут использоваться брокеры

сообщений: Apache Kafka, Apache ActiveMQ, RabbitMQ и другие. На рисунке ниже будет представлена общая схема шаблона.

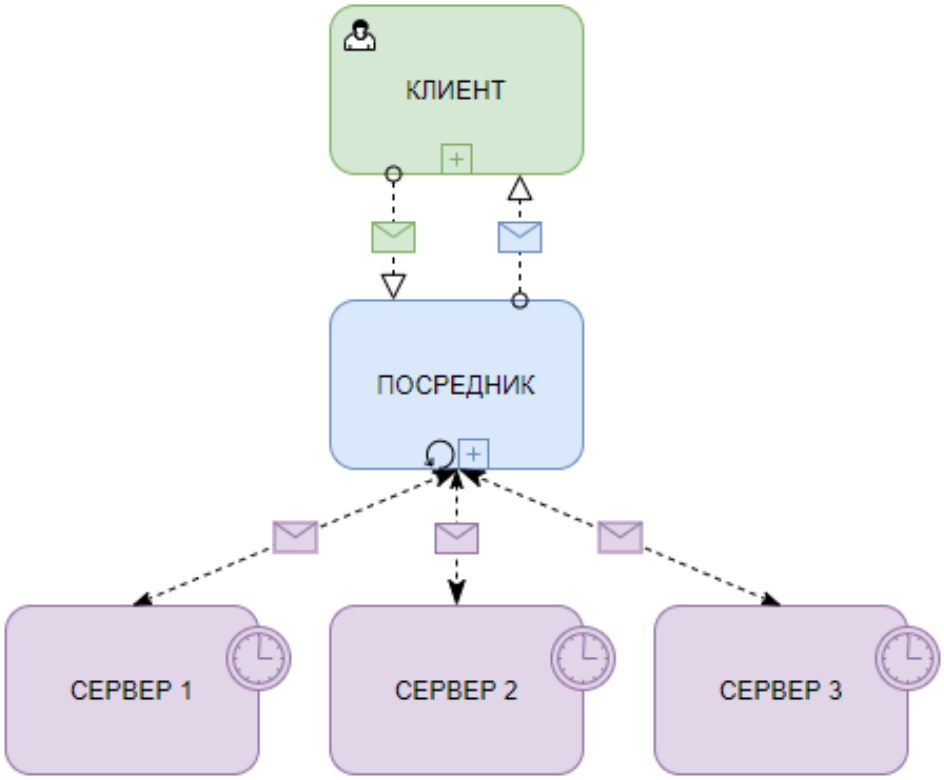

Рисунок 1.10 Общая схема архитектурного шаблона «посредник»

Подводя итоги обзора, хотелось бы отметить основные плюсы и минусы всех рассмотренных архитектурных решений для разработки приложений. Общая информация представлена ниже в таблице 1.1.

Таблица 1.1 Плюсы и минусы рассмотренных архитектурных решений

Название архитектуры	**Плюсы**	**Минусы**
Многоуровневый шаблон	• Одним низким слоем могут пользоваться разные слои более высокого ранга. • Слои данного шаблона упрощают стандартизацию, т.к. четко определяются уровни. • Изменения вносятся внутри какого-то	• Данный шаблон не универсален. • Существуют ситуации в которых возможен пропуск некоторых слоев.

	одного слоя, при этом остальные слои остаются без изменений.	
Клиент-сервер	• Подходит для моделирования и реализации набора различных служб, которые смогут запрашивать клиенты.	• Запросы обычно могут выполняться в отдельных потоках на сервере. • Взаимодействие между процессами повышает ресурсозатратность, т.к. разные клиенты имеют различные технические показатели возможности обработки и отправки запросов.
Ведущий - ведомый	• Можно выделить точность, т.к. выполнение службы делегируется разным ведомым с разной реализацией.	• Все ведомые изолированы и у них нет общего состояния. • Период ожидания в коммуникации «ведущий-ведомый» (заметно в системах реального времени) • Подходит только для тех проблем, решение которых можно разложить на части.
Посредник	• Возможно динамическое изменение, добавление, удаление и перемещение объектов. Данный шаблон позволяет делать процесс распределения прозрачным для разработчика.	• Необходима стандартизация описаний всех служб-участников процесса.

Исходя из вышеперечисленных плюсов и минусом, я бы выбрал для себя архитектуру, которая была клиент-серверной, но при этом обладала возможностями архитектуры «посредник», при этом с «посредником» можно сравнить использование API в рамках того или иного приложения, но при этом

сама архитектура останется клиент-серверной. Далее, во второй части пояснительной записки я обозначу основную концепцию разрабатываемого мобильного приложения и обосную выбор той или иной архитектуры приложения.

1.3.Обзор технологий для создания мобильного приложения

После определения архитектуры разрабатываемого приложения, необходимо обозначить стек технологий. Описание технологий и принципы их применения в моём мобильном приложении будут рассмотрены в пункте 2.2. А сейчас я предлагаю рассмотреть тот набор технологий и решений, который в настоящий момент может быть применён в современной разработке. Предлагаю рассмотреть несколько языков программирования, на которых возможно реализовать мобильное приложение. Рассмотрим различные СУБД, чтобы уже в пункте 2.4 можно было бы уделить внимание особенностям выбранной СУБД и разработать структуру БД для проекта. Также обратим внимание на такие технологии как фреймворк (рассмотрим некоторые популярные) и API.

Язык программирования – это основной, но не единственный, инструмент программиста-разработчика, который объединяет в себе набор лексических, логических и синтаксических правил. Как и любой разговорный язык, языки программирования могет отличатся по виду и принципам написания программ. Языки программирования различаются также по своему назначени, из можно разделить на три основных уровня: машинные, машинно-ориентированные, машинно-независимые. Первые два относятся к языкам низкого уровня, а вот последний – это язык высокого уровня. Именно в рамках языков высокого уровня будет проходить наш обзор.

Каждый год появляются новые языки программирования, но лишь малая часть из них завоёвывает популярность среди разработчиков. Однако, большинство из этих языков являются упрощёнными версиями своих предшественников. В них проще описаны различные библиотеки, и они имеют

больший объём уже встроенных стандартных методов, что позволяет в разы сократить объём кода, когда применяется какая-либо стандартная обработка данных или рвсчёт. Предлагаю рассмотреть следующе языки программирования: Java, JS, C++, Python, Delphi. Каждый из этих языков имеет свою аудиторию из разработчиков и имеет своё особое назначение, т.е. ту сферу применения, где удобней было бы использовать какой-то из перечисленных языков.

Java - это общий термин, который используется для обозначения программного обеспечения и его компонентов, включающих в себя такие компоненты, как "Среда выполнения Java" (JRE), "Виртуальная машина Java" (JVM) и "Внешний модуль". Java – это объектно-ориентированный язык программирования общего назначения. Это значит, что он подходит для написания как серверной части, так и клиентской. У данного языка есть множество версий, которые постоянно улучшаются и пополняются новыми необходимыми библиотеками и иным функционалом. Некоторые из возможностей языка [1]:

- Возможность написания ПО адаптированного к различным платформам;
- Создание сетевых приложений, имеющих доступ к обширному спектру веб-служб;
- Возможность разработки мобильных приложений;
- Возможность разработки backend (серверная часть) приложений;
- Возможность объединения приложений или набора служб в единую многоцелевую разработку при использовании языка Java;
- Создание многофункциональных и эффективных мобильных приложений с широкими возможностями в части frontend, а также с возможностью реализации интеграций различных внешних устройств и приложений (например, датчиков самого смартфона или иных устройств через беспроводное соединение).

Java Script (JS) – это объектно-ориентированный язык программирования. JS чаще всего используется для написания серверных частей приложения. Т.е. язык отлично подходит для backend разработки. Иго применяют, например, для создания сценариев веб-страниц. Java Script – прототипно-ориентирован, при этом может поддерживать функциональное программирование. Некоторые особенности языка:

- Объекты имеют возможность запрашивать свой тип и структуру во время выполнения программы (интроспекция);
- Функции в языке обладают возможностями передачи в качестве параметра;
- В языке присутствует автоматическое приведение типов;
- Реализована автоматическая сборка мусора –управления неиспользуемой более памятью;
- Анонимные функции – функции без уникального идентификатора;
- В языке нет стандартной библиотеки, т.е. нет управления потоками ввода-вывода, базовых типов для бинарных данных и возможности интерфейсно работать с файловыми системами;
- Нет стандартных интерфейсов для работы с БД;
- По синтаксису язык схож с C и Java, но не повторяет её.

C++ - это объектно-ориентированный типизированный язык программирования общего назначения. Используется рядом разработчиков для создания программного обеспечения различного уровня: операционные системы, драйверы различных устройств, для серверов и также на C++ разрабатывают некоторые компьютерные игры. Является потомком языка C (Си), поэтом основные свои особенности синтаксиса наследует именно от него. Некоторые особенности:

- Наличие встроенной стандартной библиотеки с обширным набором разделов;
- Наличие шаблонов;

- Совместимость с языком C;
- У языка присутствуют проблемы с вычислительной эффективностью;
- Есть возможность программного обхода ограничений;
- Сложная совместимость с другими языками программирования.

Python – это объектно-ориентированный язык программирования с довольно простым, но при этом эффективным подходом к реализации задумок разработчика. Python легко изучается и поэтому с ним возможно отходить от стандартных подходов к программированию. В настоящее время этот язык программирования часто применяется при создании и обучении нейро-сетей или написании любых программ для сбора и обработки статистики. Язык можно использовать в качестве расширения для настраиваемых приложений. В самом же языке реализованы все принципы ООП. Помимо всего прочего, питон поддерживает структурное, функциональное и обобщённое программирование. Некоторые из особенностей [6]:

- Имеет обширную стандартную библиотеку в свободном доступе;
- Способен расширяться функциями и типами данных, реализованными на C++ и C;
- Применение для работы со статистикой и нейро-сетями;
- Удобная оптимизация приложения в случае необходимости;
- Недостатком является невозможность модификации и изменения встроенных классов;
- Сильное отличие синтаксиса от привычного за счёт его упрощения.

Delphi – структурированный объектно-ориентированный язык. Основные задачи, которые выполняются разработчиками с помощью данного языка – это создание прикладных приложений. Также в языке есть модули для создания мобильных приложений под системы iOS и Android. Некоторые особенности языка:

- Дружелюбный графический интерфейс при создании приложений, т.е. построение активностей не сопровождается их дополнительным

описанием, оно генерируется автоматически, при использовании нового компонента;

- Данный язык имеет изначально объёмную начальную библиотеку;
- Проблема в быстродействии при загруженности программы;
- Лёгкая встроенная интеграция с различными БД;
- Отсутствие возможности расширения созданного приложения из-за непопулярности языка в настоящее время.

Есть ещё большое множество языков программирования, более широкой направленности или более узкой. Языки отличаются своей популярностью и использованием в крупных государственных и коммерческих проектах. Невозможно сразу охватить все имеющиеся языки программирования, но основной выбор ляжет на тот, с помощью которого можно будет максимально оптимизировать приложение и сделать его модульным с возможностью «аккуратной» доработки – подразумевается доработка, которая никак не затронет уже работающие элементы приложения.

База данных – это некая организованная структура, которая предназначена для хранения и обработки больших связанных или несвязанных друг с другом объёмов информации. Базы данных делятся на два основных типа: реляционные и нереляционные базы данных.

Реляционные БД (SQL, PL/SQL) – это набор данных, который организован в виде различных по своему бизнес-значению сущностей с предопределёнными заранее логическими связями и зависимостями.

Нереляционные БД (NoSQL) – используют разнообразные модели данных, при этом эти данные могут быть никак не связанны друг с другом по бизнес-логике. Хорошо подходит для работы с большими объёмами данных, т.к. данные чаще всего хранят в JSON или в бинарном представлении.

Система управления базами данных – это комплекс программных средств с использованием языка программирования для управления, получения доступа и создания баз данных [4].

В нашем случае основой для рассмотрения будут реляционные базы данных. Рассмотрим некоторые из СУБД:

Oracle Database – это объектно-реляционная СУБД. Одна из самых популярных и оптимизированных реляционных баз данных в среде коммерческих разработок. Языком программирования для данной СУБД является PL/SQL. Некоторые особенности СУБД [5]:

- Автоматическое управление хранением файлов базы данных;
- Связывание переменных через кеш – значительно повышает скорость выполнения запросов;
- Наличие подсказок по оптимизации при выводе плана выполнения запросов;
- Пустые строки не хранятся в памяти и не перегружают её;
- Возможность архивации журнальных файлов
- Наличие битовой индексации помимо имеющихся индексов в виде B-дерева;
- Наличие индекс-таблиц. Данные в таких таблицах лежат непосредственно в индекс-дереве первичного ключа;
- Отвечает требованиям ACID;
- Наличие кластерных и хэш-таблиц.

PostgreSQL – мощная объектно-реляционная база с открытым исходным кодом. В настоящее время является популярной среди государственных программных разработок за счёт своей надёжности и функциональности [7]. Некоторые особенности:

- Возможность обеспечивать хранение разных типов сетевых адресов (в отличии от MySQL - данная СУБД не является рассматриваемой, например);
- Поддержка геометрических типов данных;
- Возможность создания нового типа данных, в случае отсутствия необходимого;

- Отвечает требованиям ACID;
- Позволяет обрабатывать разом большие объёмы данных на уровне производительности, сравнимом с нереляционными БД.

MS Access – реляционная СУБД от фирмы Microsoft входит в пакет стандартного Office. Имеет широкий и интуитивно понятный интерфейс и спектр встроенных функций. Является скорее СУБД для первоначального ознакомления с системой запросов и в целом с языком SQL. Некоторые особенности:

- Возможность визуального выстраивания связей между сущностями с выставлением первичных ключей;
- Лёгкая и понятная иерархия внутренних БД;
- Является стандартизированной СУБД;
- Отвечает SQL требованиям, но не полноценно соответствует (ACID);
- Пустые строки увеличивают объём хранимой информации (это явный минус).

Помимо языков программирования и баз данных существует огромный пласт технологий, называемый Framework (Фреймфорк). Это набор вспомогательных элементов, значительно упрощающих жизнь разработчику. Также существуют полноценные вспомогательные сервисы с заложенным набором функций и интерфейсов (API).

Фреймворк – это набор библиотек для автоматизации постоянно повторяющихся наборов действий. Использование фреймворков позволяет значительно сократить объём кода и внести некоторую стандартизацию при совместной разработке крупных проектов. Также упрощаются связи между различными частями/модулями приложения. Сложность и объём фреймворка определяется задачами, которые ему предстоит решать. Практически ни одна современная разработка, а тем более для реализации мобильного приложения или тем более клиент-серверного мобильного приложения не может обойтись

без использования фреймворков. Набор таких фреймворков зачастую определяет именно язык программирования проекта.

API (Application Programming Interface) – интерфейс программирования приложений, который позволяет различным сервисам взаимодействовать, обмениваться и получать доступ к данным как внешним так и внутренним, в зависимости от используемого API и способа его интеграции с приложением. API можно назвать неким посредником между сервисами и приложениями, что частично пересекается с описанной в пункте 1.2 архитектурами приложений (архитектурный шаблон «Посредник»), однако, таковым не является. Данной технологией предоставляется решение в виде классов, функций и т.д. реализованное на стороннем сервисе в рамках среды, в которой проходит разработка. По факту, происходит переиспользование уже имеющейся технологии для нужд разрабатываемого приложения. По типам API можно разделить на следующие Web API:

- RPC (Remote Procedure Call) – удаленный вызов процедур;
- SOAP (Simple Object Access Protocol) – простой протокол доступа к объектам;
- REST (Representational State Transfer) – передача состояния представления.

API бывают внутренними, публичными и партнёрскими. Основными открытыми для пользования API являются таковые от фирм «Яндекс» и «Google». Чаще всего использование того или иного API от данных фирм связано больше с предпочтениями разработчика или заказчика, а не с какими-либо техническими особенностями данного продукта у этих сервисов. Использование API также, как использование фреймворков значительно сокращает количество кода и позволяет оптимизировать огромное количество процессов как в разработке, так и в отладочном тестировании разрабатываемого приложения.

1.4.Актуальность

Мобильное приложение для планирования выполнения прыжков с парашютом является весьма узконаправленным и может охватить лишь аудиторию, которая имеет непосредственный интерес к прыжкам с парашютом или занимается этими прыжками на профессиональном уровне, однако, потенциально может быть интересно и обычным пользователям в том случае, если будет хорошо ранжироваться и индексироваться ресурсе своего размещения. В качестве основных показателе актуальности разработки подобного приложения предлагаю обратиться к статистике «Яндекса». Такая же статистика есть и у «Google», но так как первоначально приложение направленно на СНГ регион, то правильнее было бы использовать именно статистические ресурсы по запросам в такой поисковой системе, как «Яндекс». Ниже я приведу количественные показатели запросов в период за несколько месяцев, а далее, за несколько лет [14]. После чего предлагаю обратить внимание на диаграммах 1.1, 1.2 и 1.3, на которых изображены диаграммы с прослеживаемой динамикой популярности выбранных мною запросов в поисковую систему по тематике парашютного спорта.

Статистика и текущие показатели запросов «Парашют», «Парашютный спорт», «Прыжок с парашютом», «Skydiving logbook» за выбранные промежутки месяцев, согласно сайту https://wordstat.yandex.ru, представлены в таблице 1.2:

Таблица 1.2 Статистика интернет запросов по тематике

Запрос	**Период**	**Количество**
Парашют	01.07.2019 - 31.07.2019	277808
Прыжок с парашютом		104062
Парашютный спорт		7592
Парашют	01.08.2019 - 31.08.2019	293921
Прыжок с парашютом		102535
Парашютный спорт		8612

Парашют	01.07.2020 - 31.07.2020	338921
Прыжок с парашютом		134637
Парашютный спорт		7857
Парашют	01.08.2020 - 31.08.2020	381098
Прыжок с парашютом		146754
Парашютный спорт		9627
Парашют	01.11.2020 - 30.11.2020	280300
Прыжок с парашютом		67185
Парашютный спорт		6982

Выбраны исключительно летние промежутки за последние два года по причине большей активности в парашютном спорте именно в это время года, однако, количественно разница в зимнем периоде ощущается, и в качестве демонстрации приведён актуальный период за ноябрь 2020 года. Также можно обратить внимание на то, что между количественным показателем запроса «парашют» и запроса «парашютный спорт» есть огромная разница. Это связано скорее всего именно с тем, что большое количество людей интересуется данной тематикой не в качестве вида спорта, а как что-то экстремальное и потенциально, хотели бы себя в этом попробовать. Также данное утверждение можно сделать, обратив внимание на запрос «прыжок с парашютом». Эта закономерность будет присуща каждому виду спорта, ведь многие умеют играть в волейбол или в теннис, но мало кто готов связать свою жизнь с профессиональным спортом. Как раз на подобную аудиторию и нацелено в какой-то степени моё приложение.

Ниже, подводя итоги разбора актуальности тематики парашютного спорта, для большей наглядности я приведу диаграммы, которые должны показать, становится ли тематика парашютного спорта популярней со временем, а именно в разрезе двух лет. Также хочу отметить то, что количественный показатель некоторых запросов крайне неплохой, учитывая, что тематика весьма узкой направленности.

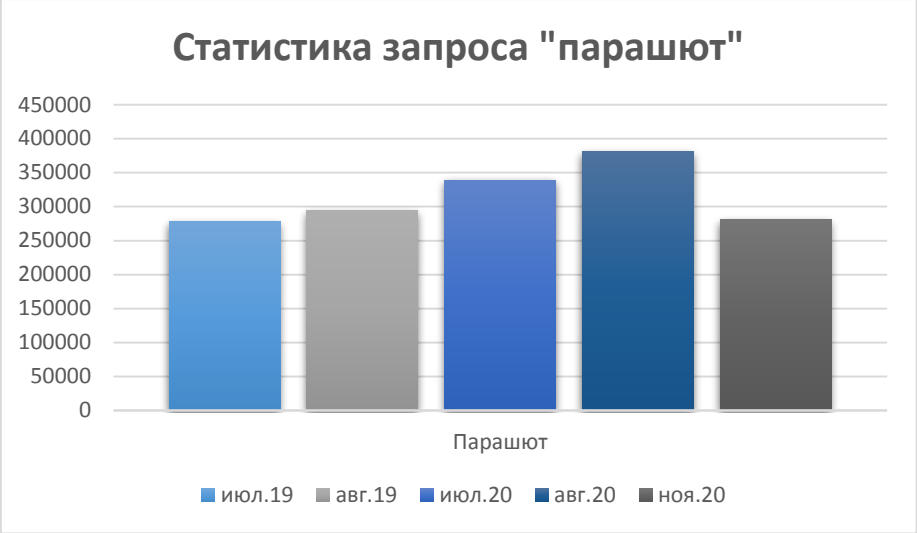

Диаграмма 1.1 Статистика запроса «парашют»

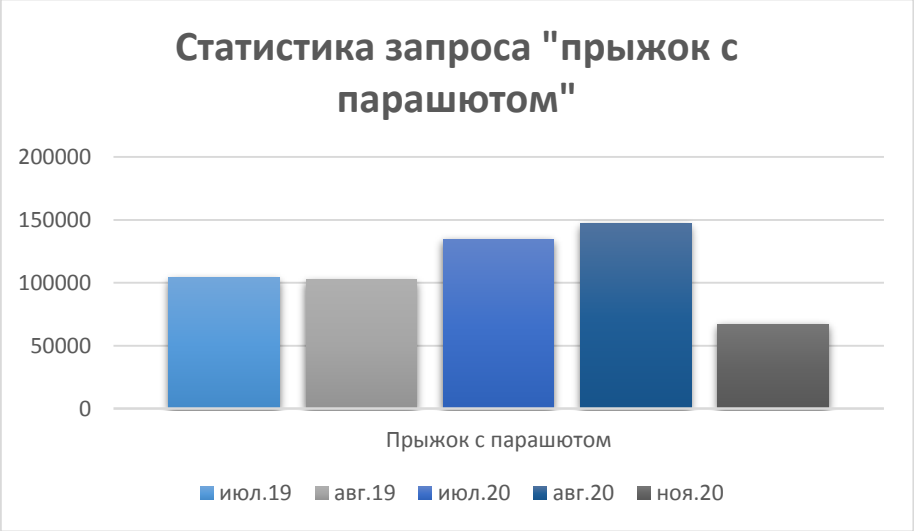

Диаграмма 1.2 Статистика запроса «прыжок с парашютом»

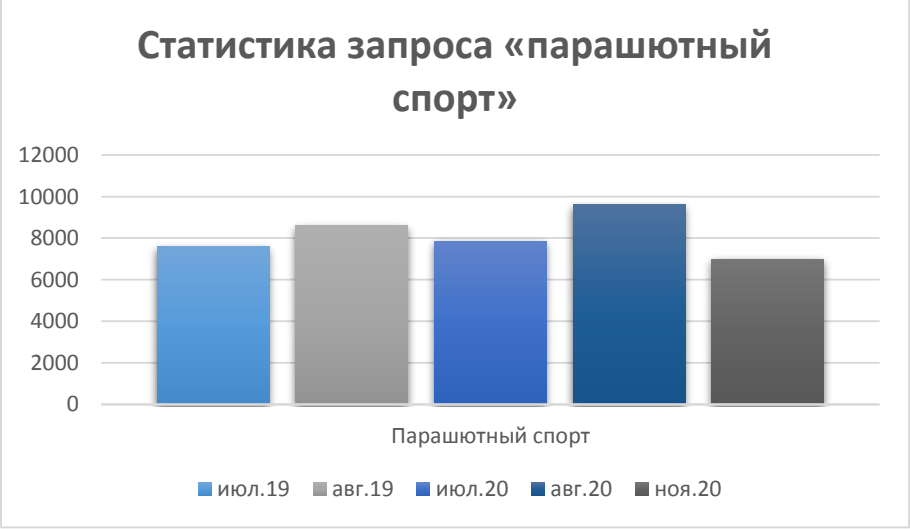

Диаграмма 1.3 Статистика запроса «парашютный спорт»

2. РАСЧЁТНО-КОНСТРУКТОРСКАЯ ЧАСТЬ

В качестве шаблона архитектуры приложения будет выбран шаблон «клиент-сервер». Данная архитектура должна удовлетворить всем требованиям, которые предъявляются к приложению. О самих требованиях я расскажу в пункте 2.1. А пока напомню, чем примечательна клиент серверная архитектура.

Архитектура клиент-сервер – это архитектура, при которой имеется два постоянных участника процесса, один из которых распределяет нагрузку по запросам второго. Но стоит добавить ещё один сегмент системы – это обязательное наличие хранилища данных. Речь сейчас идёт о базе данных, типы которых также разбирались ранее. В самом узком клиент-серверном приложении, если в нём присутствует возможность регистрации для клиентов, значит есть база данных. База данных может хранить различный объём информации, от данных нескольких сотен пользователей, до данных нескольких миллионов пользователей и хранением для каждого из них информации о переписке с кем-либо, хранение аудио и графических файлов, а также различное количество иной текстовой информации, при условии чёткого разграничения доступа к тем или иным ресурсам.

Для разрабатываемого мобильного приложения очень хорошо подходит как раз клиент-серверная архитектура с простым устройством базы данных. Т.е. будет необходимость хранения данных о регистрации пользователей – это необходимо для дальнейшей авторизации в профиле и использования именно своего аккаунта в приложении. Помимо данных о регистрации, таких как: почта, пароль и никнейм аккаунта, необходимо вторая часть хранилища, в которой будет «лежать» именно информация о пользователе, которую он сможет редактировать или удалять в зависимости от своих потребностей. А уже клиент-серверная архитектура приложения позволит каждому пользователю при входе в свой аккаунт получать доступ именно к своим данным без возможности доступа к информации других пользователей мобильного приложения. О самой разработке структуры базы данных и о

выбранной СУБД будет подробно описано в пункте 2.4. Общая схема клиент-серверной архитектуры мобильного приложения для планирования выполнения прыжков с парашютом представлена на рисунке 2.1:

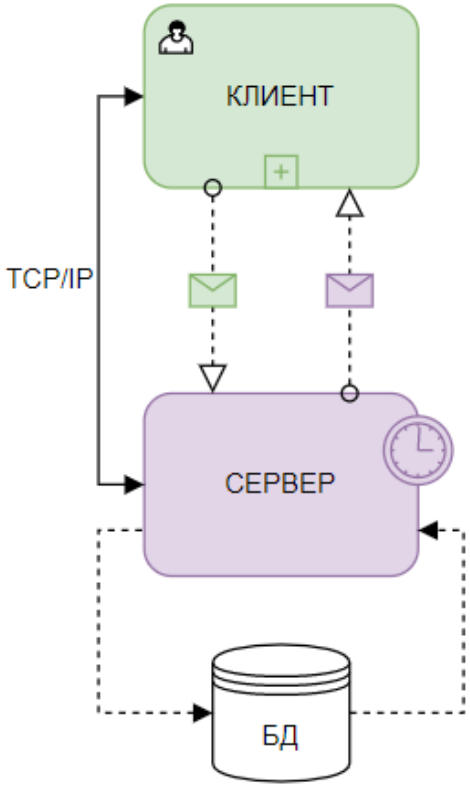

Рисунок 2.1 Общий вид архитектуры мобильного приложения для планирования выполнения прыжков с парашютом

И в качестве основных аргументов по выбору именно такой, опишу некоторые из особенностей и преимуществ данной архитектуры:

- Основным вычислительным и подвергающимся нагрузкам устройством является сервер, следовательно, в клиенсткой части приложения отсутствует код, нагружающий систему или устройство пользователя;
- Такая архитектура безопасна - клиент не имеет прямого доступа к базе данных, следовательно, не сможет воспользоваться данными других клиентов в корыстных целях;
- Код в приложении не дублируется в каком-то смысле, ведь если бы приложение не имело серверной части, то для большого количество

клиентских частей пришлось бы дублировать различный функционал, а не передавать его через распределение серверной части;

- Есть возможность обеспечить практически безотказную работу приложения за счёт использование кластера серверов – при отключении одного из серверов, клиент не теряет связи с другим, следовательно, может как зайти в приложение, так и зарегистрироваться в нём с полным сохранением всех данных;
- При подобной архитектуре система имеет возможность расширения как программного, так и аппаратного – для увеличения своих вычислительных способностей.

2.1.Требования к функционалу приложения

В результате обзора существующих аналогов разрабатываемого мобильного приложения для планирования прыжков с парашютом, мною был выявлен некоторый функционал, который в обязательном порядке должен присутствовать в приложении. Также был выявлен функционал второго плана – это тот, который не несёт какую-то колоссальную ценность, но также может быть полезным. Помимо прочего, был выявлен функционал третьего плана – в него входят те элементы, которые я не планирую реализовывать в текущей версии приложения, но они могли бы стать крайне интересным и функциональным дополнением в следующих версиях моей разработки.

Все требования к функционалу, как первого, так и второго плана, я предлагаю рассматривать одним списком, а уже отдельно рассмотреть функционал третьего плана.

Основные требования к функционалу приложения:

- Разрабатывается сугубо мобильное приложение под операционную систему Android для мобильных устройств, с возможностью поддержки различных версий (от 6 и выше);
- Приложение будет клиент-серверным, в нём будет реализована регистрация клиентов и хранение их персональных данных;

- Дизайн и интерфейс приложения должны соответствовать выбранной тематике, следовательно, должны содержать элементы и атрибутику парашютного спорта;
- В приложении должна быть возможность редактирования и пополнения информации о себе в отдельной активности;
- В приложении необходима реализация расчёта приблизительной математической модели прыжка с парашютом с различных высот, при различных условиях и в различной экипировке, при этом вводимые данные для «дружелюбия» приложения не должны привязываться к конкретному человеку, т.е. физиологические параметры человека вводятся отдельно, как и иные числовые показатели, которые могут меняться;
- В активности математической модели прыжка необходим вывод максимальной скорости, которую разовьёт парашютист и дополнительные комментарии по прыжку, такие как: критическое время раскрытия парашюта и риски, которые возможны при выбранных параметрах;
- В приложении должна присутствовать возможность просмотра дропзон на карте с возможностью комментариев к ним;
- Необходима реализация доступа к прогнозу погоды в рамках общего формата, а также по конкретным дропзонам, для возможности планирования своего «прыжкового дня»;
- В приложение возможна для реализации отдельная активность с историей парашютного спорта и с возможностью перехода на интересующие источники, а также просмотр фото и видео по тематике;
- Хранение всех данных клиентов будет реализовано в реляционной базе данных, которая должна иметь возможность расширения, например, для добавления отдельного хранилища фотографий и видео пользователей или для хранения истории чата.

В качестве функционала, рассматриваемого как потенциальное дальнейшее дополнение приложение хочется отметить такие глобальные, с точки зрения доработок вещи, как:

- Возможность ведения внутреннего чата с другими пользователями (ведёт к дополнению в БД);
- Возможность использовать телефон в качестве «карманного» высотомера, а именно пользоваться датчиками внутри устройства для фиксации скорости и положения, для дальнейшего сравнения с теоретически рассчитанными показателями (ведёт к дополнению в БД) – но, подобный функционал напрямую зависит от возможностей мобильного устройства и не может быть оптимизирован;
- Портирование приложения на iOS.

2.2.Исследование стека используемых технологий

Определившись с требованиями к функционалу мобильного приложения, можно обозначить используемый стек технологий, отметить особенности некоторых из них, для дальнейшего формирования основных алгоритмов работы приложения. Необходимо было выбрать операционную систему, под которую будет разрабатываться приложение, объектно-ориентированный язык программирования, систему управления базой данных, возможные фреймворки при написании серверной или клиентской части приложения и набор используемых API.

ОС – в качестве основной операционной системой, под которую будет разрабатываться приложение выбрана ОС Android. ОС Android – это операционная система, которая используется в смартфонах, планшетах, телевизоров с системами smart и во многих других электронных девайсов [3]. Данная ОС является открытой, именно этим обусловлена её популярность. Под неё пишется огромное количество приложений, а также различных технологий, таких как API. У данной операционной системы хорошая расширяемость и совместимость с предыдущими своими же версиями – это

позволяет разрабатывать любые мобильные приложения, используя все современные вспомогательные технологии и виджеты, с оглядкой на пользователей, у которых может быть установлена устаревшая версия ОС. Выбирая в качестве основной системы разработки Android сразу устраняется необходимость в выборе среды, в которой будет проходить разработка интерфейсов и иного функционала на клиентской части, т.к. компания предоставляет собственную среду для подобных разработок – Android Studio. Данная среда включает в себя широкий набор библиотек, совместимых с различными версиями ОС, основным языком для написания программ является Java. Сразу имеется широкий набор шаблонных решений, а этот аспект улучшает вид программного кода, т.к. все шаблонные коды и функционалы уже являются оптимизированными. Также в Android Studio есть встроенные эмуляторы смартфонов, которые позволяют проводить визуальное ручное отладочное тестирование без затрат времени на установку ПО на личный смартфон.

Язык ООП – в качестве основного языка программирования будет выбран объектно-ориентируемый язык Java. Данный язык, как уже описывалось ранее, является типизированным ООП. Приложения, которые были написаны на языки Java могут функционировать на различных компьютерных архитектурах, в которых есть реализация виртуальной машины Java. Виртуальная машина Java (JVM) – это основная исполнительная часть системы ООП - Java Runtime Environment (JRE) [2]. Данный язык является одним из самых популярных в настоящее время. Именно его чаще всего используют в крупных фирмах при разработке сложного программного обеспечения, к которому предъявляются высокие требования по надёжности и расширяемости. Кроме популярности, но и не без её участия, для данного языка и на всех платформах, которые его используют, имеется неисчислимое количество дополнительных библиотек и фреймворков, написано огромное количество API, это позволяет значительно сократить объём кода и более

детально погрузиться в задуманные функциональные особенности приложения.

На ООП Java будет реализована как серверная, так и клиентская части мобильного приложения.

СУБД – в качестве основной системы управления базами данных в моём приложении будет использоваться СУБД Oracle Database 11g в рамках проектирования БД в среде SQLDeveloper. Напомню, что Oracle Database – это реляционная система управления базами данных. Конкретно в версию 11g входит огромное количество дополнительных вспомогательных модулей, как логических, так и опциональных. Вот некоторые из них [5]:

- Oracle Active Data Guard – эта опция позволяет разграничивать ресурсоёмкие работы с основной БД, т.е. даёт возможность распределения этих работ на резервные синхронизированные БД;
- Advanced Compression – данная функция позволяет управлять растущими объёмами данных с большей эффективностью за счёт сжатия типов данных (как стандартных текстовых или числовых, так и картинок с видео) – данный функционал будет крайне полезен при расширении приложения и добавлении в него нового функционала при постоянном наращивании аудитории;
- Advanced Security – обеспечивает улучшенное более понятное для разработчика шифрование данных;
- Database Vault – данный функционал управляет доступами к приложению и информации в нём, что позволяет обеспечить безопасность данных от внутренних злоумышленников;
- Oracle Data Mining – это одна из аналитических функций, которая позволяет проанализировать различные закономерности в скриптах и предложить альтернативные более оптимальные варианты;
- Partitioning – данный функционал позволяет индексировать таблицы на более простые компоненты, которыми значительно проще управлять;

- Diagnostic Pack – набор модулей, встроенных в ядро Oracle Database;
- Oracle Programmer – это отдельный встроенный продукт, который обеспечивает программный интерфейс для разработчика в любых продуктах Oracle Database.

Также присутствует огромное количество других встроенных компонентов и модулей, но здесь перечислены, на мой взгляд, более интересные из них. По причине совей надёжности, популярности и расширяемости была выбрана именно СУБД Oracle.

Фреймворк – как уже было описано ранее, в пункте 1.3, это набор некоторых библиотек, которые обеспечивают автоматизации постоянно повторяющихся процессов в рамках разрабатываемого приложения или программы. В рамках разрабатываемого мобильного приложения будут применяться следующие фреймворки: Spring, Maven, Hibernate, Jackson, HTTP Client. Ниже присутствует краткое описание и назначение каждого из перечисленных:

- Hibernate – это фреймворк, обеспечивающий связку реляционной базы данных, в нашем случае это Oracle Database, с объектно-ориентированным языком программирования, для нас таким является Java. Данный фреймворк является весьма популярным в настоящее время, но основная причина его использование – это надёжная и прозрачная работа с описанием подключаемой СУБД. Hibernate максимально структурирован и понятен в использовании, позволяет без особых проблем и излишков кода связывать классы в Java с сущностями из СУБД. Также особенностью этого фреймворка является автоматическая генерация запросов при работе с СУБД, что позволяет работать только в рамках классов и методов в них, не переходя на SQL скрипты.
- Maven – данный фреймворк необходим для автоматизации сборки проектов на основе описания структуры в файлах на основе языка POM из подмножества XML.

- Jackson - это высокопроизводительный процессор JSON для Java;
- Spring Framework – это универсальный фреймворк с открытым кодом для Java, который обеспечивает архитектуру Модель – Отображение – Контроллер;
- HTTP Client – данный фреймворк/библиотека разработана компанией Oracle для использования в рамках отправки и обработки HTTP запросов;
- Retrofit2 – безопасный HTTP-клиент для Android и Java. Является важным элементом для работы с API.

API – как уже было описано ранее, в пункте 1.3, это набор програмно реализованных способов связи разных классов, функций, процедур и т.д. с другими программами. В рамках мобильного приложения для планирования выполнения прыжков с парашютом, исходя из требований к функционалу приложения, будут использоваться API с возможностью подключения модулей географической карты, с определением местоположения и поисковой системой, а также понадобится API погоды. Ниже представлено краткое описание модулей, используемых в моём приложении:

Google Map SDK for Android – стандартное API от компании Google, которое позволяет добавлять в приложение Google карты Сам сервис предоставляется в автоматизированном виде и содержит дополнительные классы и графические компоненты для настройки, например:

- Значки, маркеры точек и подобные;
- Различные разделительные линии;
- Геометрические фигуры;
- Привязанные к определённым местам на карте битовые изображения;
- Наборы картинок для отображения поверх карты.

Weather API – одно из самых распространённых API для работы с прогнозом погоде, используется как приложение по умолчанию на всех

современных смартфонов. С помощью данного API есть возможность просматривать текущие прогнозы погоды и прогнозы на несколько дней вперёд, а также есть возможность просмотра исторической сводки по погоде в указанном регионе. Ещё одна из особенностей Weather API – то, что данное API бесплатное, по сравнению с иными сервисами, которые требуют иногда, например, Яндекс значительную сумму, после перехода лимита запросов на прогноз (данный лимит будет достигнут примерно в течение 1 дня пользования приложением). Однако, и в текущем API если ограничения по запросам, но они значительно выше и не блокируют полностью доступ к сервису.

2.3.Исследование математической модели прыжка с парашютом

Говоря о прыжке с парашютом, всегда приходит на ум какие-то формулы равноускоренного движения с высоты. И в какой-то степени - это правильно, ведь на человека действует сила притяжения, и он действительно равноускорено падает вниз. Но справедливо ли данное утверждение именно к парашютистам? Я уверен, что нет. Ведь данный полёт является осознанным, а, следовательно, подразумевает соблюдение как-то правил и дополнительных мер безопасности, к которым, как это не странно, относится и сам парашют. При этом форма парашютов как в убранном виде, так и в раскрытом может быть абсолютно разной. Отделение человека от самолёта может также отличаться от раза к разу, а ещё и скорость самого самолёта, высота прыжка, влияние различных атмосферных явлений, коэффициенты разрежённости и сопротивления воздуха на разных высотах и ещё огромное количество факторов и даже таких несущественных, как болтающиеся шнурки из ботинок. Учесть абсолютно все факторы при математических расчётах возможно, но спрогнозировать и определить точное значение для массового использования практически невозможно, т.к. придётся заставлять пользователей вводить огромное количество информации. О которой они возможно даже никогда и не слышали.

Чтобы как-то упростить вводимые пользователем данные и обеспечить оптимизированные вычислительные действия с максимально приближённым к реальности значением максимальной скорости и критического времени раскрытия парашюта, мною были проанализированы несколько статей по данной тематике.

При рассмотрении математической модели движения парашютиста при совершении им прыжка с самолёта необходимо учитывать две задачи: баллистико-временные характеристики тела парашютиста при учёте его собственной массы и массы экипировки, и описание движения в процессе совершения самого прыжка для изменения характеристик аэродинамических показателей с учётом управления парашютной системой [12]. Но т.к. в текущей реализации приложения я не затрагиваю работу с датчиками смартфона, то и в качестве расчётов для математической модели прыжка решение второй поставленной задачи я рассматривать не буду.

Первая же задача будет решаться упрощённой математической моделью. Необходимо систематизировать все имеющиеся параметры и также различные постоянные величины. К постоянным параметрам, исходя из исследования Ю.В. Усачёва и В.Н. Курашина можно отнести следующие: H – высота выброса парашютиста; V_0 – скорость самолета; k – вес, рост парашютиста; g – ускорение свободного падения; ρ – плотность воздуха; T – температура воздуха. К временным (переменным) параметрам относятся: t_n – время десантирования, w – скорость ветра; V – скорость парашютиста; u – скорость восходящих (нисходящих) потоков; d – снос (расстояние от проекции на землю точки выброса до точки приземления); C – коэффициент лобового сопротивления десантируемого объекта; F – мидель десантируемого объекта [12].

Сам прыжок с парашютом можно разделить на 4 этапа:

- Отделение от самолёта;
- Свободное падение;
- Раскрытие купола парашюта;

- Снижение и приземление на раскрытом куполе.

В настоящий момент будут рассматриваться первые два этапа – это отделение и свободное падение, по причине того, что весь анализ в текущей реализации строится именно на потенциальных рисках при совершении прыжка и на критических величинах. Также я не учитываю потенциальное наличие стабилизирующего парашюта у прыгающего. Ниже, на картинке 2.2 представлена зависимость относительной плотности воздуха от высоты:

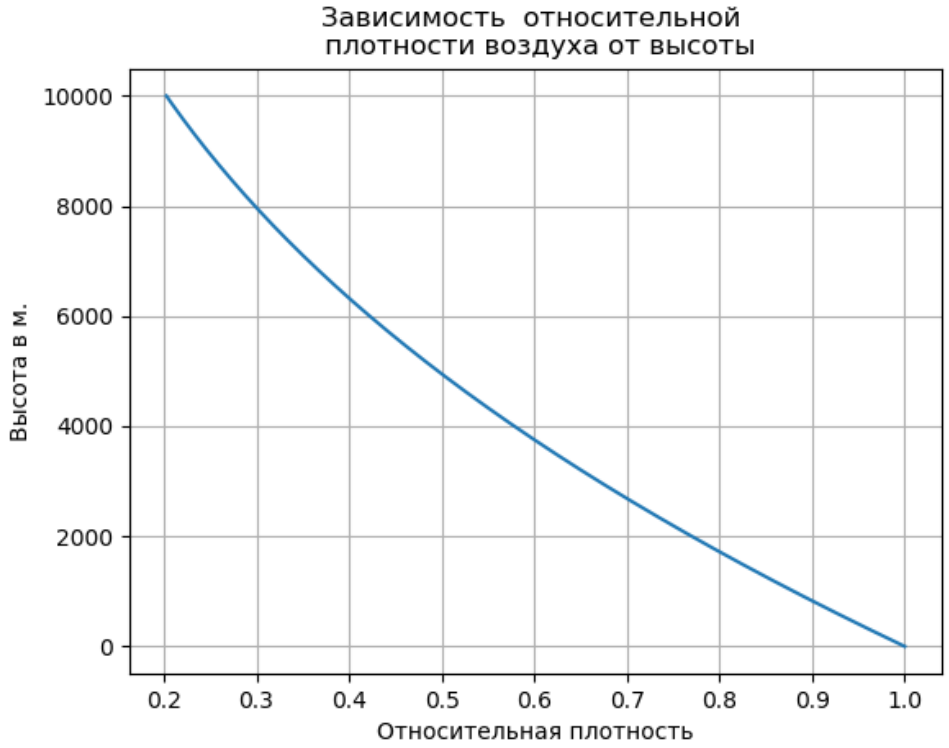

Рисунок 2.2 Зависимость относительной плотности воздуха от высоты

За величину миделя принимают квадрат роста человека – это самое оптимальное решение для расчётов.

На парашютиста помимо силы притяжения и силы тяжести действует ещё и сила сопротивления воздуха, которая в свою очередь направлена в обратную сторону от силы притяжения и тяжести:

Формула 2.1 Расчёт силы сопротивления воздуха

$$F_c = k * V^2$$

Формула 2.2 Вес, рост парашютиста

$$k = \rho * \frac{C * F}{2}$$

Значение *C* обычно берётся из специальных таблиц и в моих расчётах будет примерно унифицироваться в зависимости от выбранных параметров.

Далее (Формуле 2.3) можно составить дифференциальную систему уравнений для определения траектории движения парашютиста:

$$\begin{cases} m\frac{dV_\zeta}{dt} = -mg + F_C * sin\theta, \\ m\frac{dV_\eta}{dt} = -F_C * cos\theta \end{cases} \quad (2.3)$$

Затем можно поделить на m обе части системы уравнений и обозначить k/m = r (Формулв 2.4):

$$\begin{cases} m\frac{dV_\zeta}{dt} = -g + r * sin\theta * V^2, \\ m\frac{dV_\eta}{dt} = -r * cos\theta * V^2 \end{cases} \quad (2.4)$$

Далее авторы стать предлагают решить данную систему дифференциальных уравнений для того, чтобы продемонстрировать наглядность движения парашютиста на графике, но в рамках моей разработки нет необходимости подобного решения и все необходимые формулы выведены [11][12]. Далее, в пункте 3.4.4 я продемонстрирую свои выведенные приближённые формулы, которые опираются на данное исследование статьи. Но не стоит забывать, что любой расчёт, каким бы точным он не был, не может учитывать случайных факторов, которые могут появиться в момент прыжка, например, другие парашютисты или неожиданный порыв ветра, всё это может влиять на конечные характеристики прыжка с разной степенью критичности. Опираясь на текущие формулы, мне удалось получить финальные значения скоростей и времени. Которые максимально близки к реальным параметрам, которые удавалось фиксировать при выполнении прыжков.

В целом, при расчётах также желательно учитывать обтекаемость одежды и парашютной системы, расстояние между ногами и руками, поэтому в своих формулах я отталкиваюсь от средней площади сопротивления тела парашютиста. В случае более детального рассмотрения модели прыжка появляется возможность создания матрицы состояний на момент времени.

2.4.Разработка структуры базы данных

База данных на базе Oracle Database и при использовании среды SQLDeveloper будет содержать данные о зарегистрированных клиентах, а также данные о информации о клиенте, которую клиент лично должен будет внести в приложение. Т.к. для каждого клиента есть его уникальный аккаунт с паролем, никнеймом и почтой, то и вся остальная информация привязана именно к этому аккаунту.

Ниже, на рисунке 2.3, будет представлена общая схема, сделанная в программе ERWin с демонстрацией идентифицирующей связи 1 к 1:

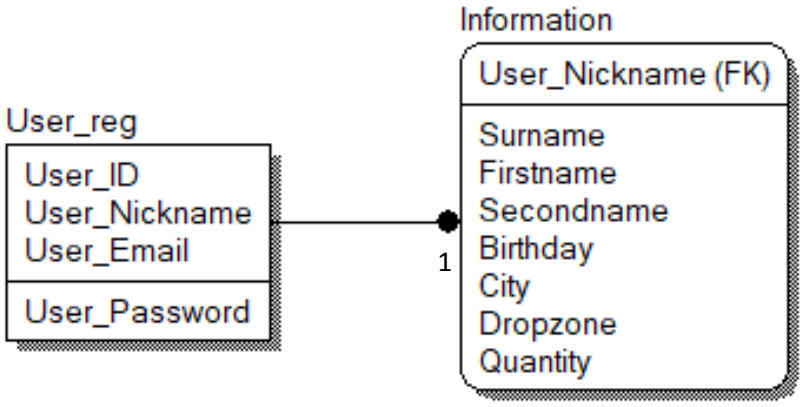

Рисунок 2.3 Общая схема-концепт БД

Однако, привести к подобному виду связи можно далеко не в каждой СУБД, т.к. по факту игнорируется первичный ключ таблицы-источника, что обычно несвойственно при создании реляционной модели. Но в рамках некоторого эксперимента было принято к использованию именно такое решение при наличии идентифицирующей связи между сущностями. Стоит отметить то, как заполняется первичный ключ таблицы «Информация». После регистрации пользователя, никнейм, который пользователь выбрал себе и который является уникальным значением, как и email пользователя, данные заносятся в ключевую таблицу, после чего отрабатывает триггер в описанный в рамках СУБД с параметром «AFTER INSERT ON», что позволяет заполнять поле никнейма пользователя в таблице «Информация» сразу после его попадания в базу. Также для правильного функционирования фреймворка

hibernate создаётся последовательность для идентификаторов. Ниже, в таблице 2.1 представлен общий скрипт базы данных с триггером.

Таблица 2.1 Скрипт описания базы данных

```
CREATE TABLE User_reg (
   User_id NUMBER NOT NULL,
   User_Email VARCHAR2(50) NOT NULL,
   User_Password VARCHAR2(50) NOT NULL,
   User_Nickname VARCHAR2(50) NOT NULL,

   CONSTRAINT pk_user PRIMARY KEY (User_id),
   CONSTRAINT uk1_mail UNIQUE (User_Email),
   CONSTRAINT uk2_nick UNIQUE (User_Nickname)
);

CREATE SEQUENCE "HIBERNATE_SEQUENCE" MINVALUE 1 MAXVALUE
9999999999999999999999999999
INCREMENT BY 1 START WITH 3 CACHE 20 NOORDER NOCYCLE;

CREATE OR REPLACE TRIGGER User_reg
AFTER INSERT ON User_reg
FOR EACH ROW
BEGIN
   insert into Information(User_Nickname)
      values (: new. User_Nickname);
END;

CREATE TABLE Information (
   User_Nickname VARCHAR2(50) NOT NULL,
   SurName VARCHAR2(50) NULL,
   FirstName VARCHAR2(50) NULL,
   SecondName VARCHAR2(50) NULL,
   Birthday DATE NULL,
   City VARCHAR2(50) NULL,
   Dropzone VARCHAR2(50) NULL,
   Quantity VARCHAR2(10) NULL,

   CONSTRAINT pk_info PRIMARY KEY (User_Nickname),
   CONSTRAINT fk_info FOREIGN KEY (User_Nickname)
      REFERENCES User_reg (User_Nickname)
);
```

Далее в таблицах 2.2 и 2.3 представлен маппинг двух таблиц (подробное описание полей таблиц с типом данных):

Таблица 2.2 Описание таблицы «Регистрация»

Атрибут	Тип данных	Уникальность	Заполняемость	Описание
User_id	NUMBER	PRIMARI KEY	NOT NULL	Уникальный идентификатор сущности
User_Email	VARCHAR2(50)	UNIQUE	NOT NULL	Адрес электронной почты пользователя
User_Nickname	VARCHAR2(50)	UNIQUE	NOT NULL	Псевдоним, придуманный пользователем
User_Password	VARCHAR2(50)	default	NOT NULL	Пароль пользователя

Таблица 2.3 Описание таблицы «Информация»

Атрибут	Тип данных	Уникальность	Заполняемость	Описание
User_Nickname	VARCHAR2(50)	PRIMARI KEY (foreign key)	NOT NULL	Псевдоним пользователя
FirstName	VARCHAR2(50)	default	NULL	Имя пользователя
SurName	VARCHAR2(50)	default	NULL	Фамилия пользователя
SecondName	VARCHAR2(50)	default	NULL	Отчество пользователя
Birthday	DATE	default	NULL	Дата рождения пользователя
City	VARCHAR2(50)	default	NULL	Родной город пользователя
Dropzone	VARCHAR2(50)	default	NULL	Название любимого аэродрома

Quantity	VARCHAR2(10)	default	NULL	Количество прыжков с парашютом

Хочется отметить, что можно было подойти к классической реализации модели базы данных и создать в таблице «Информация» дополнительное уникальное поле-идентификатор в качестве первичного ключа и воспользоваться неидентифицирующей связью, но было принято альтернативное нетривиальное решение.

2.5.Разработка структуры серверной части приложения

Т.к. приложение является клиент-серверным, то его неотъемлемой и обязательной частью будет являться именно серверная часть. Она обеспечивает полноценное функционирование как регистрации с авторизацией в приложении, так и доступ к сервисам API.

Основная работа серверной части заключается всё-таки в авторизации и регистрации пользователя. В качестве сервера используется локальный сервер на базе Tomcat – это контейнер сервлетов. Вся обратная связь проходит путём работы HTTP методов GET и POST запросов в соответствии с работой клиент-серверной архитектурой. В серверной части приложения обрабатываются коды, которые позволяют понять положительный результат запроса или получена ошибка.

Ниже на рисунке 2.4 представлен общий алгоритм работы серверной части приложения, данный алгоритм во многом повторяет алгоритм авторизации и регистрации пользователя:

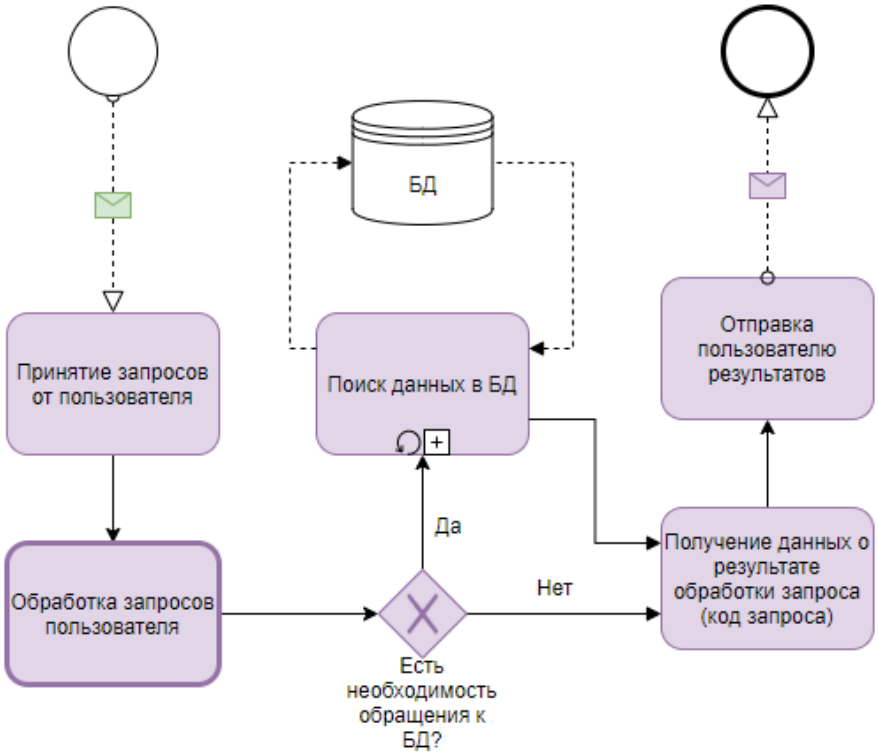

Рисунок 2.4 Общая структура работы серверной части приложения

В настоящий момент не предусматривается функция отправки письма на email пользователя после прохождения регистрации.

2.6.Разработка структуры клиентской части приложения

Клиентская часть приложения содержит в себе значительный объём данных. Доступ ко всем элементам приложения происходит только после прохождения первого этапа – это авторизация (в том случае если пользователь уже зарегистрирован в приложении) или регистрация.

После проведения манипуляций с входом в приложение, которые схемой обозначены ранее в пункте 2.5, пользователь получает доступ на главную страницу приложения откуда может переходить по разным активностям приложения, которые реализованы в нём в соответствии с техническим заданием и определёнными в первой части пояснительной записки функционалом приложения.

Пользователь имеет возможность переходить с любой активности в любую без повторной авторизации. В случае, если у пользователя пропадёт интернет соединение с сервером, но авторизации выполнена, те активности,

которые способны функционировать в офлайн режиме, продолжат свою работу, но пропадёт доступ к погоде и карте.

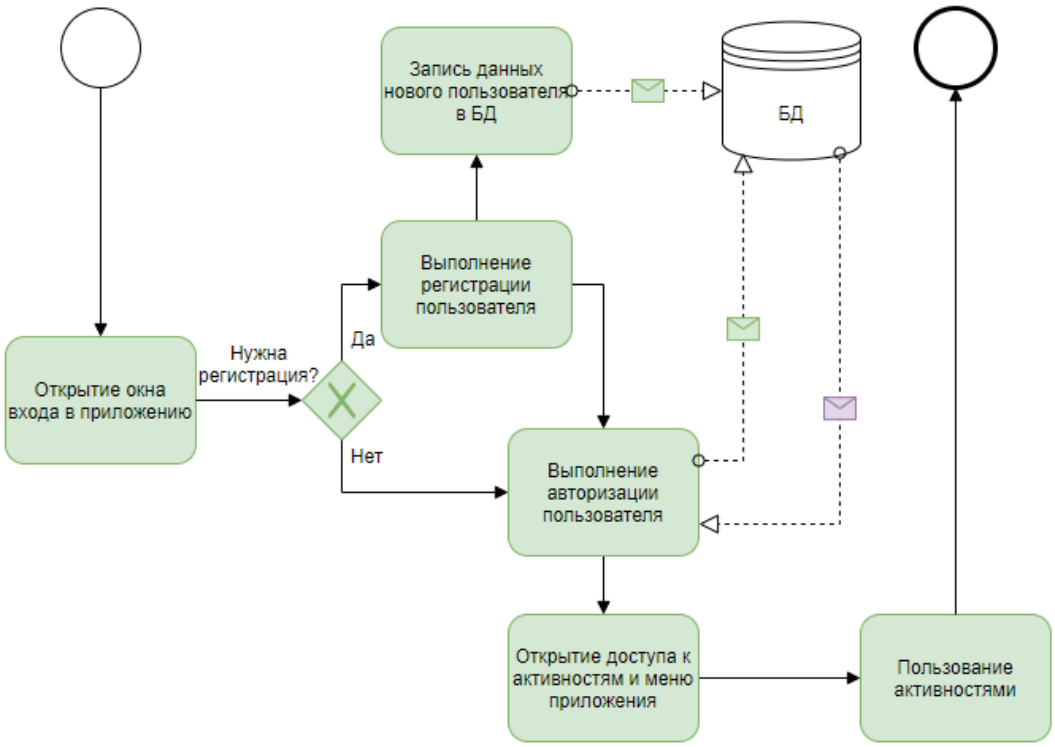

Рисунок 2.5 Общая структура работы клиентской части приложения

У каждой активности есть своё дополнительное меню, которое может содержать дополнительный функционал. Также стоит заметить, что пользователь не сможет в настоящий момент поменять свой никнейм после того, как в серверной части произойдёт его фиксация. Связь клиентской части с базой данных происходит только через сервер и отправку на него соответствующих HTTP запросов (обозначено в пункте 2.5 на рисунке 2.4).

2.7.Разработка интерфейса пользователя

Как отмечалось ранее, в пункте 2.1, интерфейс пользователя должен содержать элементы и атрибутику парашютного спорта, но при этом, должен быть интуитивно понятным любому пользователю. Активности не должны быть нагружены лишней информацией, ненужными всплывающими окнами и в целом не должны быть загромождены меню и кнопками.

Чтобы избежать подобного загромождения, я принял решение дробления приложения на 3 основные части: вход в приложение, главное

меню, иные активности. Ниже, на рисунке 2.5 представлена общая структура интерфейса приложения:

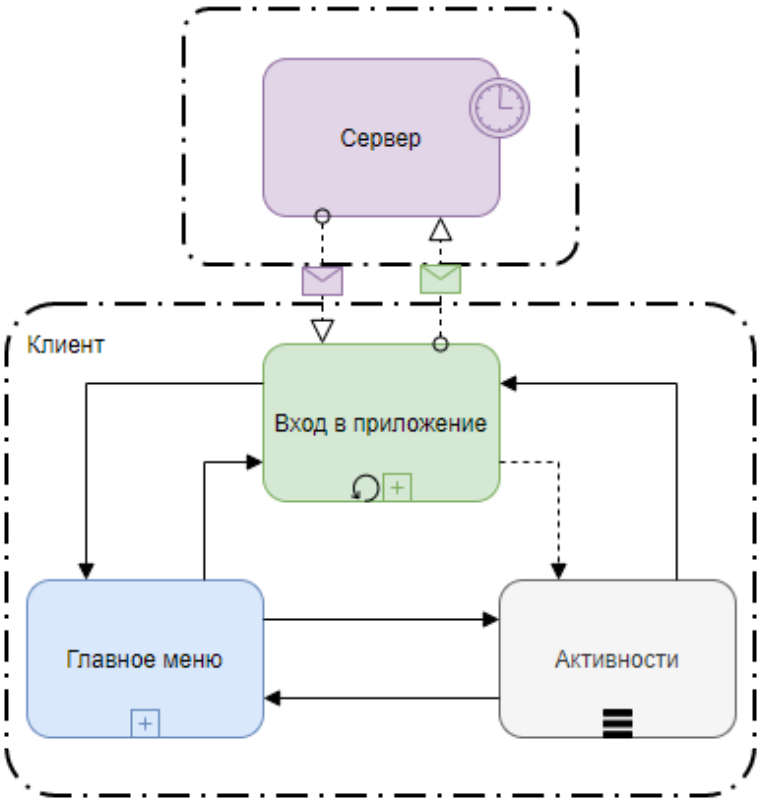

Рисунок 2.6 Общая структура интерфейса приложения

Далее, в пункте 2.7.1 по пункт 2.7.8 будут представлены общие алгоритмы работы всех присутствующих в мобильном приложении активностей. Отдельно хочется отметить, что уже на этапе разработки интерфейсов, было принято решение о том, что для некоторых активностей общая вкладка с «меню» будет незначительно отличатся, однако, пользователь должен иметь возможность перейти на активность авторизации из любой другой активности.

2.7.1. Регистрация и авторизация

В данной активности весь алгоритм работы интерфейса сводится к предоставлению пользователю возможностей выбора действия по авторизации в приложении или по регистрации в нём. Соответственно авторизоваться могут только зарегистрированные пользователи. Сам

интерфейс состоит из 4-х активных кнопок, ползунка для возможности демонстрации введённого пароля и 3-х текстовых окон для ввода текста.

Процессы, которые происходят после взаимодействия с теми или иными функциями описаны в пункте 3.4.1, а ниже на рисунке 2.7 приведён общий алгоритм работы интерфейса без конкретики по сетевой составляющей. Также на активности имеются скрытые функционалы – всплывающие окна при необходимости вывести пользователю информационное сообщение.

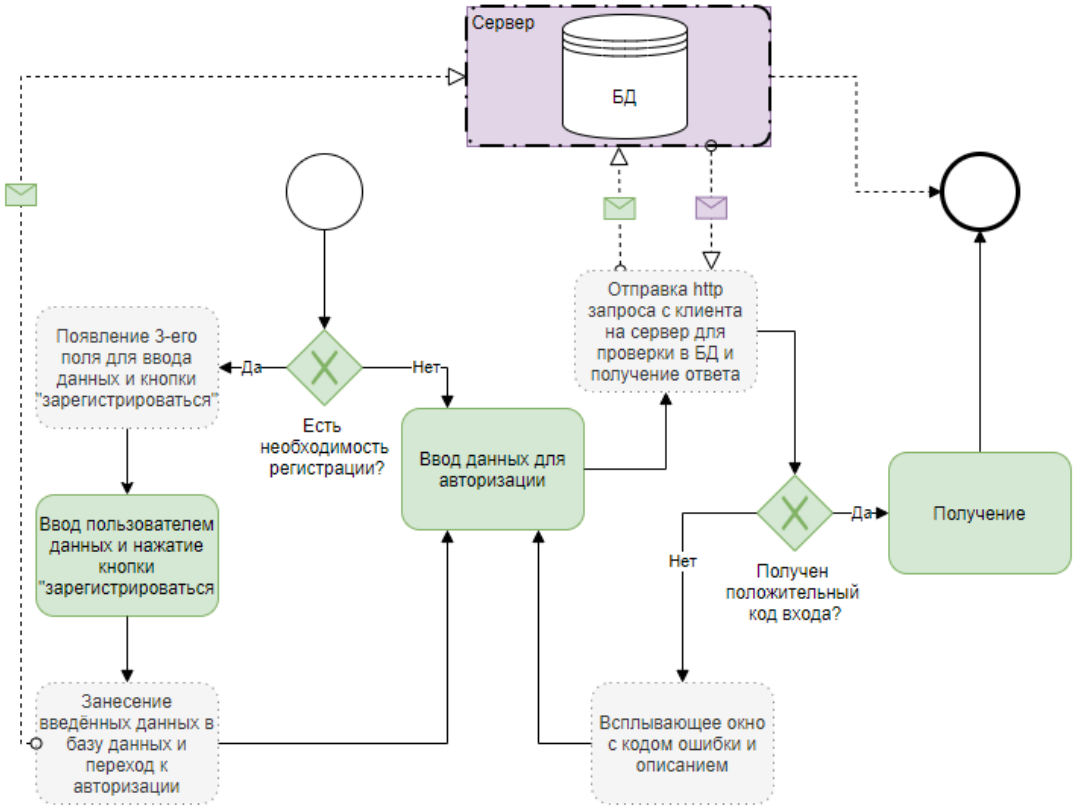

Рисунок 2.7 Алгоритм работы интерфейса «регистрация и авторизация»

2.7.2. Главная страница

На главной странице приложения находится активность с набором кнопок, позволяющих выбрать следующую активность для перехода. Также в верхней части находится меню для перехода к предыдущей активности.

Пользователь может попасть в главное меню, также, как и во все остальные активности, идущие после регистрации/авторизации, только после авторизации в приложении и получении от сервера удовлетворяющего кода авторизации (описано в пункте 3.4.1).

На фоне активности находится анимационное изображение или статическая картинка.

На рисунке 2.8 изображен алгоритм работы интерфейса для главной страницы:

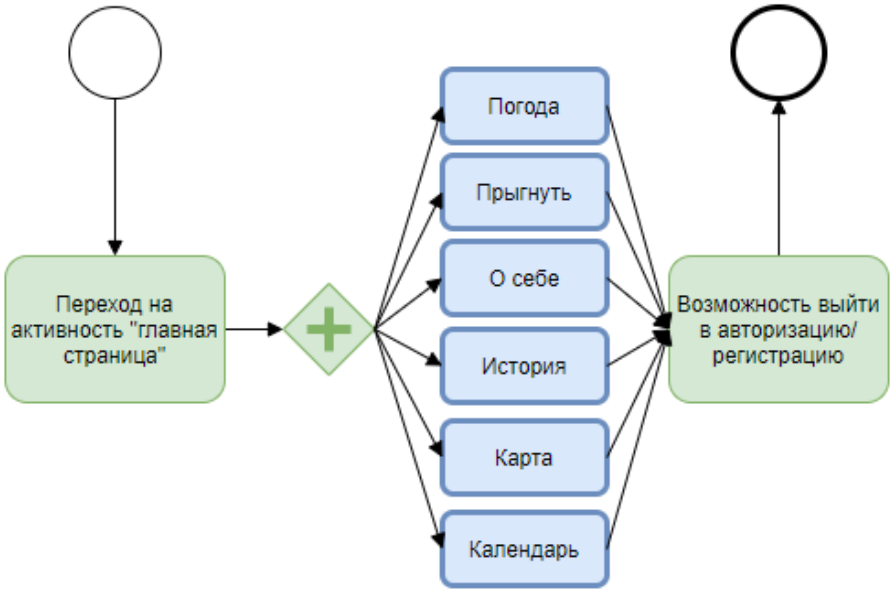

Рисунок 2.8 Алгоритм работы интерфейса «главная страница»

2.7.3. Погода

Активность «погода» - является активностью, в которую можно попасть только через главное меню приложения. Представляет из себя сервис API с демонстрацией прогноза погоды в выбранном регионе и местности.

Отправляется автоматический запрос на ресурс предоставления данных о погоде и по ключам API возвращается ответ в визуализированной форме для клиента.

Внутри активности есть дополнительное меню с возможностью выйти в главное меню или выйти на этап «авторизации и регистрации».

На рисунке 2.9 представлен алгоритм работы интерфейса «погода»:

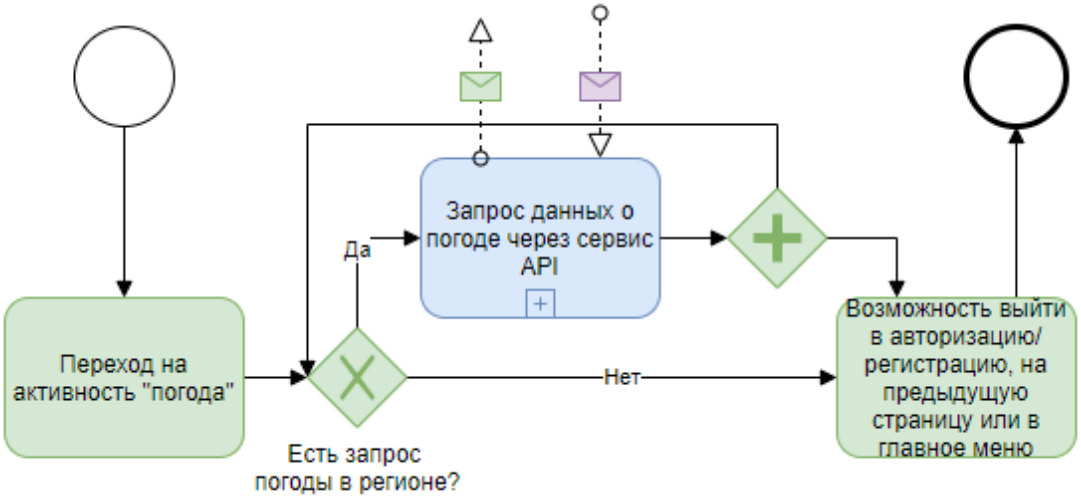

Рисунок 2.9 Алгоритм работы интерфейса «погода»

2.7.4. Математическая модель прыжка

Активность «математической модели прыжка», которая в приложении называется кратко «прыгнуть» является самой нагруженной по математическим вычислениям и имеет внутри себя 5 полей для ввода чисел, которые являются параметрами пользователя. Также есть 1 поле с выборкой заранее предустановленных значений, которые определяют коэффициент для расчёта. Имеется 2 поля для вывода результатов расчётов.

В активности имеется классическое меню для перехода «назад», «на главною» и «выйти».

Ниже, на рисунке 2.10 представлен алгоритм работы интерфейса «математическая модель прыжка»:

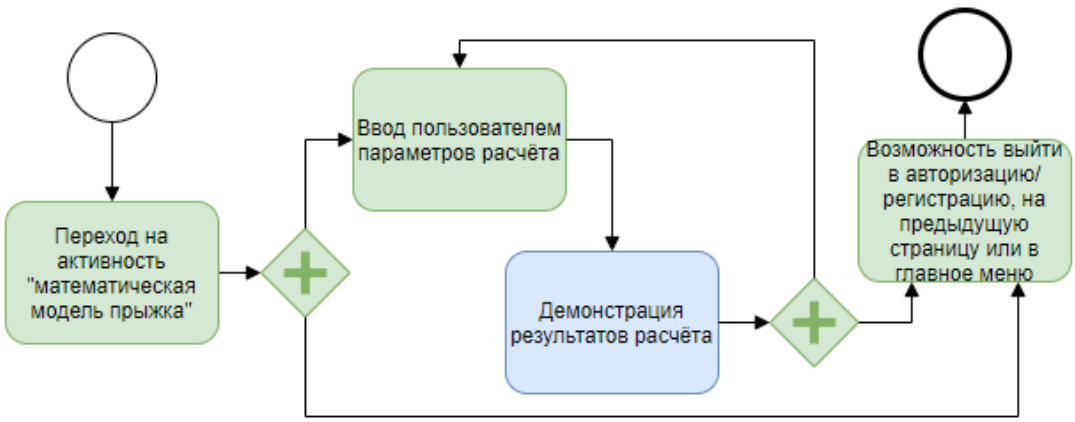

Рисунок 2.10 Алгоритм работы интерфейса «математическая модель прыжка»

2.7.5. Информация о пользователе

В активности «информация о пользователе», которая в рамках приложения называется более кратко – «о себе», находится 7 активных окон для ввода текста и 1 неактивное окно, в котором находится неизменяемый никнейм пользователя, который придумывается при регистрации и является уникальным идентификатором. Все остальные окна для ввода текста заполняются самим пользователем, вот их состав, исключая поле «никнейм»: «Имя», «Фамилия», «Отчество», «Дата рождения», «Родной город», «Любимая дропзона», «Количество прыжков». Каждое название поля, в случае, если оно не заполнено прописывается полупрозрачным шрифтом на поле ввода текста для наглядности.

Все данные, которые уже были сохранены клиентом подтягиваются из базы данных при выполнении авторизации и отображаются сразу при заходе на активность «о себе».

Помимо 8 полей с текстом в активности есть 3 кнопки и возможность входа в дополнительное меню. В случае, если не нажата кнопка «редактирования» все поля для ввода текста заблокированы и защищены от случайного нажатия.

В активности имеется классическое меню для перехода «назад», «на главною» и «выйти».

Ниже представлен рисунок 2.11 с алгоритмом работы данного интерфейса:

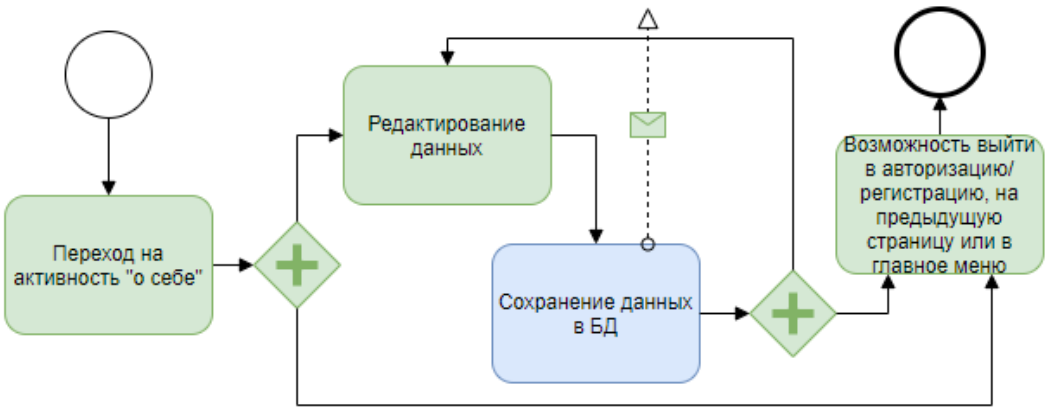

Рисунок 2.11 Алгоритм работы интерфейса «информация о пользователе»

2.7.6. История парашютного спорта

«История парашютного спорта» - данная активность является шаблонной для работы с длинными текстовыми данными. Имеется отдельное верхнее меню, не похожее на остальные. Динамическая шапка, которая увеличивается при приближении к верхней части активности. Также есть ползунок и возможность увеличения масштаба текста, если это необходимо для пользователя.

На рисунке 2.12 представлен алгоритм работы интерфейса «история парашютного спорта»:

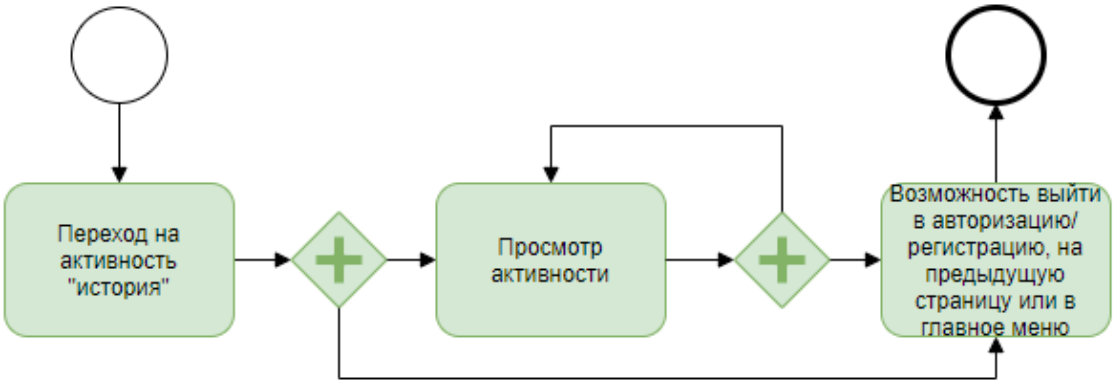

Рисунок 2.12 Алгоритм работы интерфейса «история парашютного спорта»

2.7.7. Карта с аэродромами

Активность «карта с аэродромами» (в приложении кратко «карта») является второй активность с предоставлением сервиса по API ключам. Представляет из себя карту с отмеченными аэродромами и возможностью ставить дополнительные маркеры.

Часть функционала уже была встроена в API, поэтому была доделана некоторая связь с активностью погоды. При выборе какой-либо дропзоны появляется возможность просмотра погоды в регионы этой дропзоны. При нажатии на данную всплывающую кнопку пользователь переходит на активность погоды (из которой по кнопки из меню может вернуться назад на

карту или вернуться туда через главное меню) где автоматически подставляется в запрос региона выбранный на карте.

В активности имеется классическое меню для перехода «назад», «на главною» и «выйти».

На рисунке 2.13 представлен алгоритм работы интерфейса:

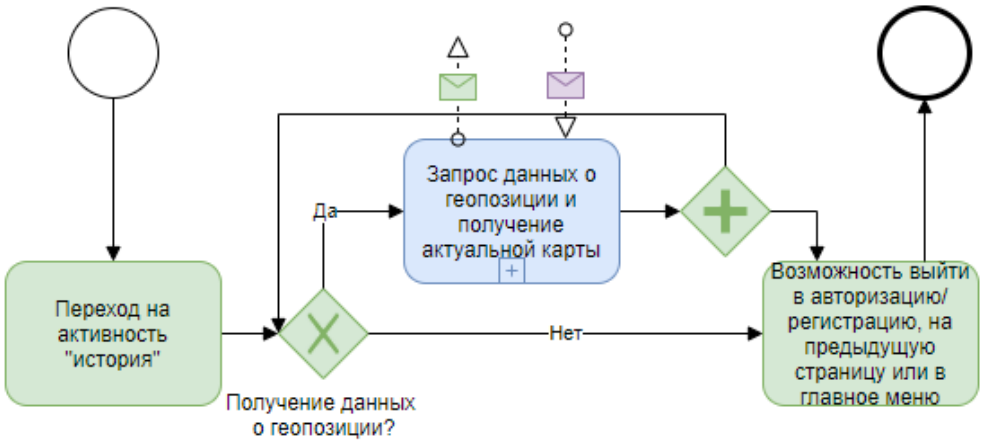

Рисунок 2.13 Алгоритм работы интерфейса «карта с аэродромами»

2.7.8. Календарь прыжков

Интерфейс «календарь прыжков» является по факту копией встроенного в смартфоны на системе Android календаря, но при этом никак не влияет на его наполненность. Сессия с календарём сохраняется в офлайне и не требует дополнительной связи с БД.

В активности имеется классическое меню для перехода «назад», «на главною» и «выйти».

На рисунке 2.14 изображён алгоритм работы интерфейса:

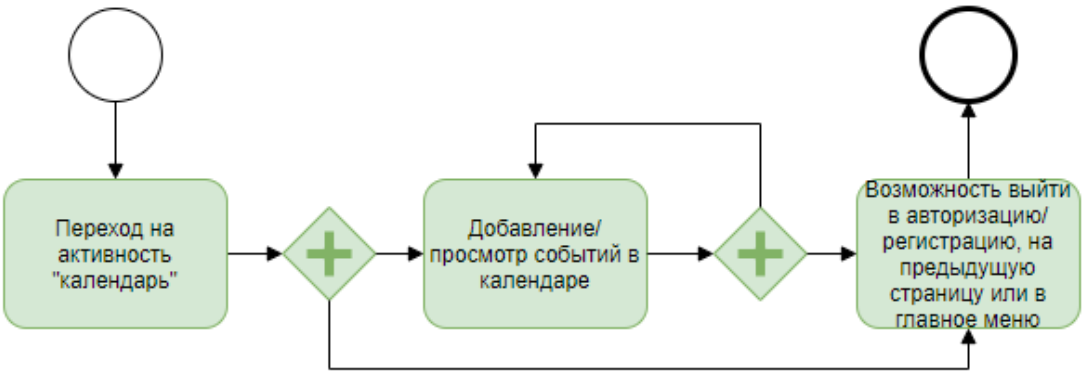

Рисунок 2.14 Алгоритм работы интерфейса «календарь прыжков»

2.8.Разработка плана тестирования приложения

Приложение содержит в себе набор алгоритмов, каждый из которых может тестироваться абсолютно разными тест-кейсами. Т.к. приложение разделено на две части, а именно: на клиентскую и серверную части, то предлагаю рассмотреть подход общего глобального тестирования каждой части отдельно. В кейсы тестирования серверной и клиентской части будут заложены работы с вложенными алгоритмами. Все итерации тестирования представлены в виде схемы с описанием каждого блока. Это унифицированная схема для универсального тестирования. Разработана в рамках проектируемого мобильного приложения, но в случае схожести архитектур и применяемых технологических решений может быть переиспользована для других мобильных клиент-серверных приложений с незначительными доработками по возможным техническим дополнениям и альтернативным бизнес кейсам. На рисунках 2.15 и 2.16 представлены процессы ручного тестирования клиентской и серверной частей мобильного приложения.

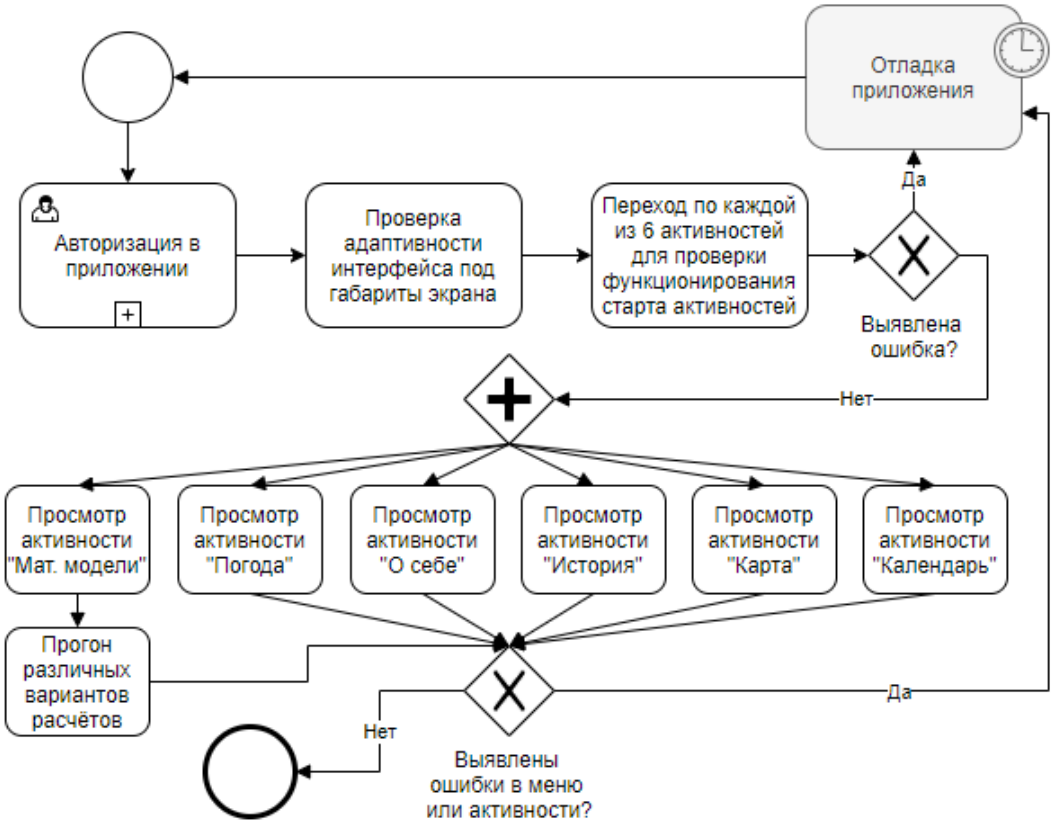

Рисунок 2.15 Общий процесс тестирования и отладки клиентской части приложения

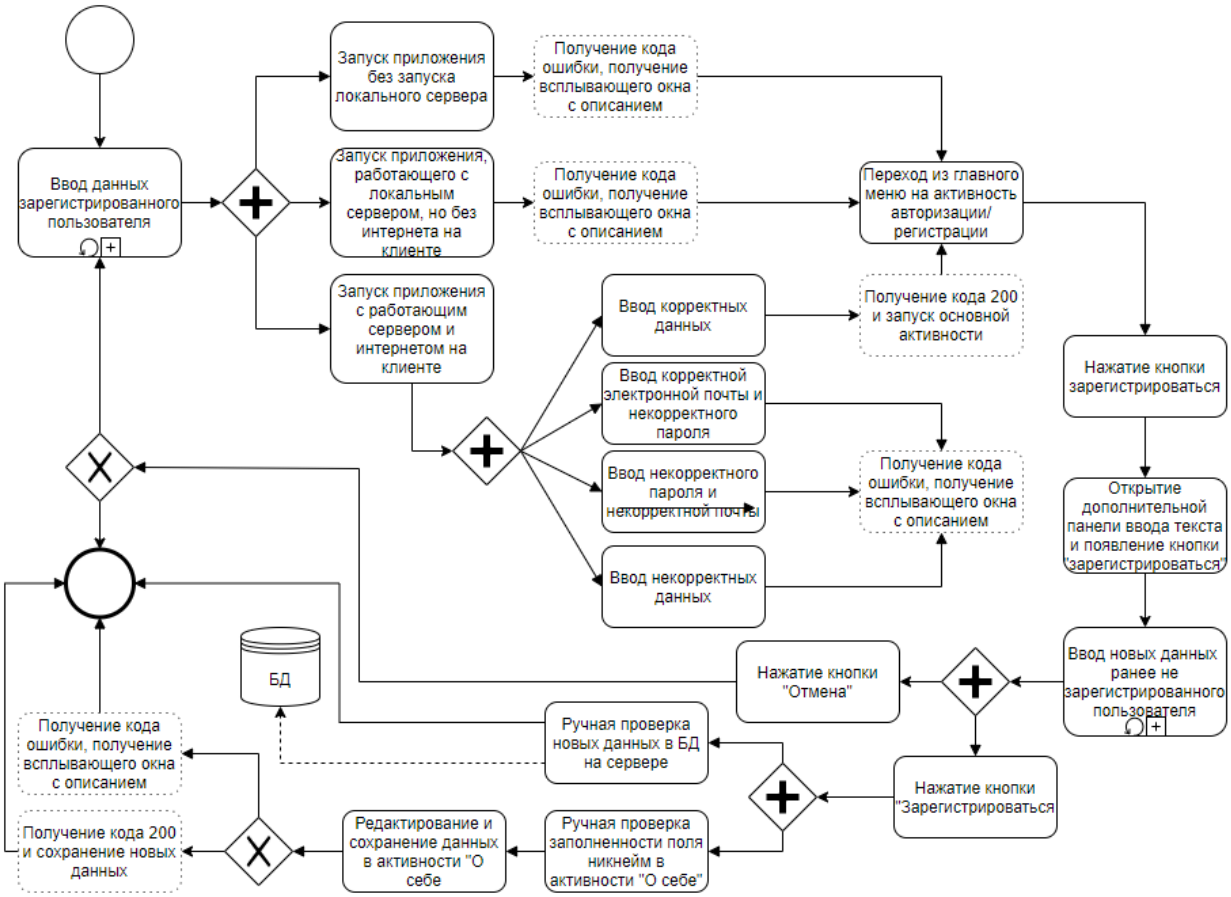

Рисунок 2.16 Схема процесса тестирования и отладки серверной части приложения

3. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

В экспериментальной части представлен результат проделанной работы по разработке мобильного приложения для планирования выполнения прыжков с парашютом.

3.1.Структура приложения

Разработанная структура клиент-серверного приложения на базе языка программирования Java с использованием API и различных фреймворков, приведена на рисунке 3.1.

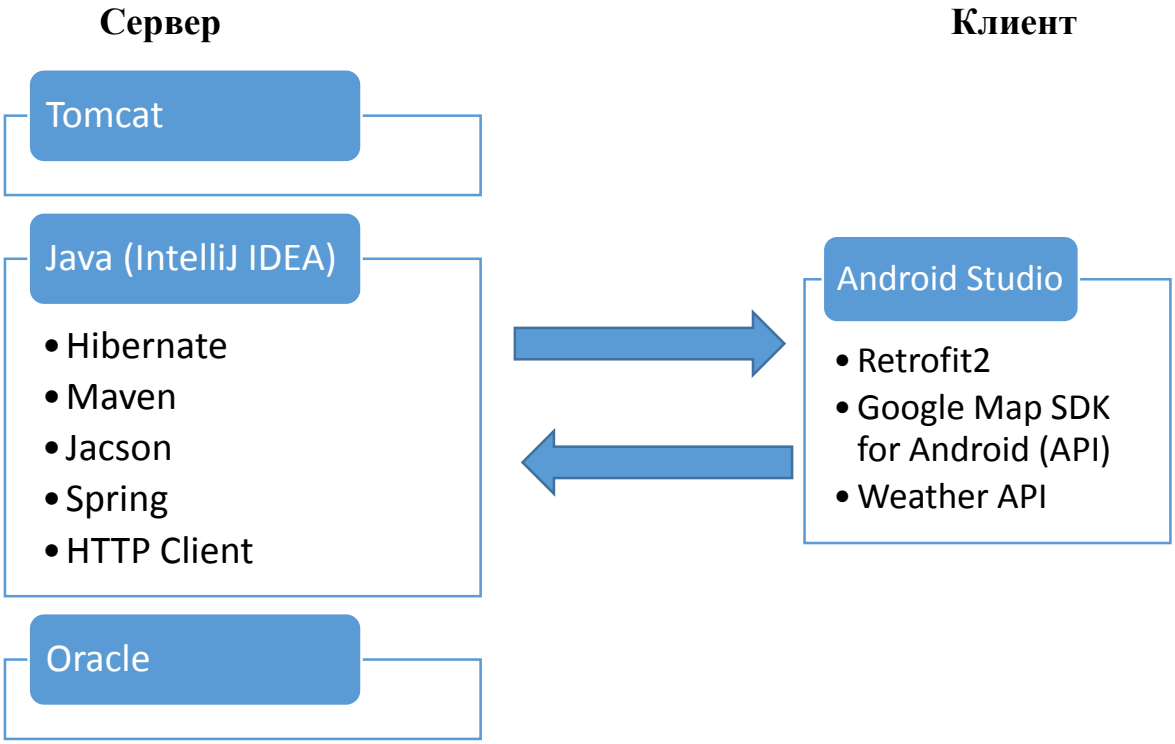

Рисунок 3.1 Конечная структура проекта

Данная структура несколько отличается от общего вида, предоставленного на рисунке 2.1, но не нарушает его. В качестве контейнера для развёртывания приложения используется Tomcat, а уже база данных Oracle находится в рамках данного контейнера. В качестве языка, на котором пишется серверная часть приложения используется Java, а ниже в блоке представлены фреймворки, задействованные для реализации всего задуманного функционала. Клиентская часть представлена ОС Android, описание которой производилось в Android Studio с использованием перечисленных сервисов API.

3.2.Реализация серверной части приложения

О стеке технологий, участвующих в реализации серверной части, я уже писал в пункте 2.2 и пункте 2.4. Здесь я хочу вкратце напомнить о применяемых технологиях и программных средах для реализации серверной части мобильного приложения для планирования выполнения прыжков с парашютом.

- Tomcat - контейнер сервлетов для развёртывания приложения;
- Среда разработки Java (IntelliJ IDEA) - интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java;
- Hibernate – фреймворк для связи РСУБД и ООП;
- Maven – фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM (англ. Project Object Model), являющемся подмножеством XML;
- Jackson –сериализация/десериализация JSON;
- HTTP Client – фреймворк для работы с http запросами
- Oracle - реляционная СУБД. Хранит данные о регистрации клиента и информацию о нём.

В рамках написания программного кода были применены как стандартные решения для реализации связи с сервером, так и уникальные в рамках конкретного приложения. И уникальных можно отметить обработчик некоторых сетевых ошибок, а также работа с предоставлением данных пользователя, которые были им сохранены ранее в личном кабинете (в информации о пользователе). Из типовых решений можно отметить описание связи клиент-сервер по средству получения сетевых кодов клиентом от сервера и сервером от клиента. Данная реализация является хоть и типовой, но единственной верной в рамках текущей реализации приложения и функционала в нём.

3.3.Реализация базы данных в Oracle SQLDeveloper

База данных состоит на данный момент из 2-х таблиц (схема представлена на рисунке 3.2). В первой - содержится информация о регистрации пользователей, которая соответственно используется и в качестве подтверждения авторизации. Там имеются такие поля как id (идентифицирующий номер пользователя) – автоматически генерируется при регистрации нового пользователя, email (электронная почта), nickname (псевдоним пользователя) – при помощи триггера в БД после добавления в таблицу «Регистрация» автоматически добавляется в таблицу «Информация» и password (пароль). Стоит отметить, что в данный момент пароль никак не шифруется – это явный недостаток, который в дальнейшем будет исправляться, чтобы не было возможности получить пароль пользователя через БД. Во второй таблице содержится поле псевдонима из первой (в качестве основного ключа) и различная информация о самом пользователе. Вся информация из второй таблицы не является обязательной к заполнению, и любой пользователь сознательно или неосознанно может оставить все поля кроме nickname (заполняется автоматически при регистрации) пустыми или заполнить лишь некоторые из них.

Подобная структура БД позволяет дополнять и нагружать её дополнительными ресурсами без изменения общей логики работы с регистрацией и авторизацией, т.е. для дополнения приложений не придётся менять логику именно серверной части и уже реализованной клиентской. Это значительно упрощает возможности для расширения функционала приложения. Несмотря на свою ёмкость, представленная схема БД не является перегруженной связями и информацией.

Более подробно состав и принципы реализации базы данных в моём мобильном приложении описаны ранее в пункте 2.4.

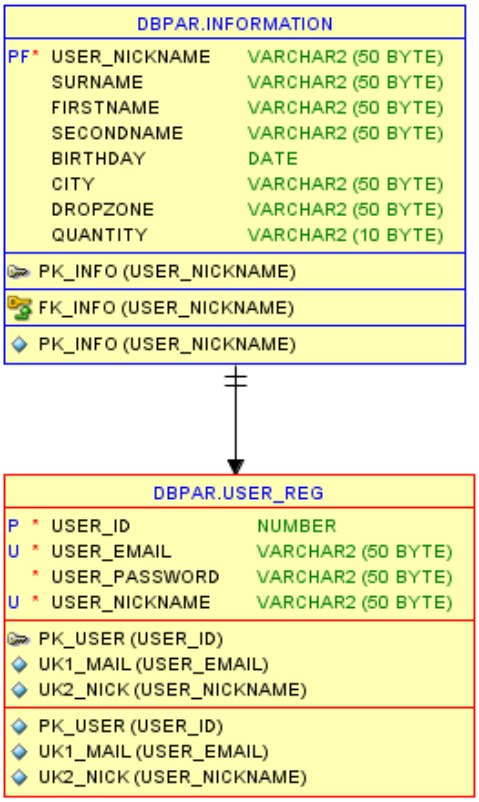

Рисунок 3.2 Схема базы данных приложения

3.4.Реализация клиентской части приложения (интерфейсы и алгоритмы)

Среда разработки Android Studio. Для создания активностей и интерфейса на них используются стандартные библиотеки и элементная база Android Studio. Подключённая библиотека Retrofit2 используется для работы с HTTP запросами при регистрации, авторизации и сохранении данных о пользователе.

Добавлены через API карта и погода. В настоящий момент реализовано в двух раздельных активностях.

3.4.1. Регистрация и авторизация

Авторизация – это первая активность, которую видит пользователь при входе в приложение (Рисунок 3.3).

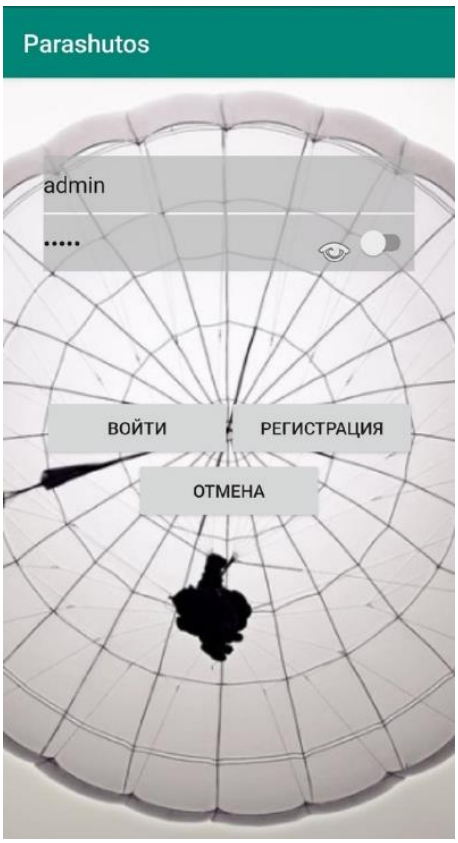

Рисунок 3.3 - Интерфейс авторизации

Пользователь имеет в доступе 2 поля для ввода текста, кнопку «показать введённый пароль» справа от поля ввода пароля и 3 активных кнопки «Войти», «Регистрация» и «Отмена». В случае, если пользователь незарегистрирован при нажатии на кнопку «Регистрация» для пользователя откроется возможность зарегистрироваться. Рассмотрим краткий алгоритм работы авторизации:

1) Ввод данных в активные поля
2) Формируется HTTP запрос со стороны клиента для отправки на сервер с целью подтверждения наличия данного пользователя в системе
3) Сервер проверяет сходство полученных данных с данными в БД
    a. Если такого клиента в базе нет, выводится сообщение «Такой пользователь не зарегистрирован» (Рисунок 3.4)
    b. Если email пользователя найден, но пароль не совпадает, то выводится ироничное сообщение «Неверный пароль, а функции поменять нет, ХА, вспоминай» - хочу сразу обратить внимание,

что такой функции пока действительно нет, но в дальнейшем это необходимый функционал. (Рисунок 3.5)

c. Если логин (email) и пароль совпали с имеющимися в БД, то сервер возвращает положительный статус и пользователь, успешно авторизовавшись, входит на главную страницу приложения. Также из БД подцепляются данные из таблицы «Информация» и в соответствующей активности заполняются все поля.

Рисунок 3.4 Нет пользователя

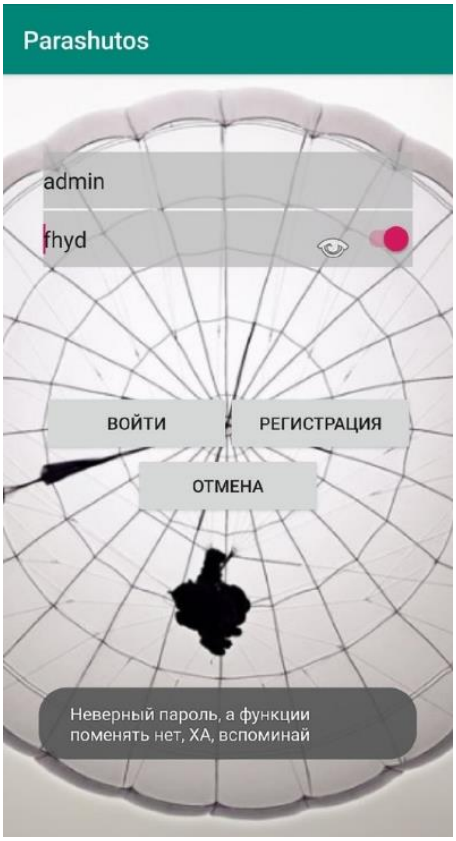

Рисунок 3.5 Неверный пароль

При нажатии кнопки «Регистрация» становится неактивной кнопка «Войти», появляется дополнительное поля для ввода псевдонима (никнейма) пользователя появляется также новая кнопка «Зарегистрироваться», а кнопка «Регистрация» меняет своё название на «Назад к входу» (при её нажатии восстанавливается функционал авторизации). Функционалом кнопки «Зарегистрироваться» является сохранение данных о новом пользователе в БД. После нажатия этой кнопки, также возвращается принцип работы авторизации, но теперь не нужно вводить заново данные, т.к. они сохранены в

активных полях, поэтому при нажатии кнопки вход пользователь перейдёт на главную страницу. Ещё раз опишу по пунктам алгоритм регистрации:

1) Нажатие на кнопку «Регистрация» («Вход неактивна», новое активное поле (никнейм), название «Регистрация» меняется на «Назад к входу», новая кнопка «Зарегистрироваться») (Рисунок 3.6);
2) Пользователь заполняет 3 активных поля (почта, никнейм и пароль);
3) Пользователь выбирает дальнейшее действие
    a. Кнопка «Зарегистрироваться»
        i. Данные отправляются на сервер по HTTP;
        ii. Сервер сохраняет нового пользователя в БД и заполняет поле никнейм в таблице «Информация», сохраняя все остальные поля пустыми (если не возникло ошибок на стороне сервера, все высвечиваются в нижней части экрана)
        iii. От сервера приходит положительный статус регистрации
        iv. Пользователь может выполнить авторизацию
    b. Кнопка «Отмена» или «Назад к входу»
        i. Позволяет пользователю вернуться к авторизации

Рисунок 3.6 Регистрация нового пользователя

3.4.2. Главная страница

На основной активности главной страницы (Рисунок 3.7) пользователю даётся возможность выбрать одну из четырёх активностей: «WEATHER» (заглушка), «ПРЫГНУТЬ», «О СЕБЕ» и «ИСТОРИЯ». Каких-то особых алгоритмов в работе главной страницы нет, т.к. её основной задачей является переход по активностям.

Также на главной странице присутствует меню в правой верхней части экрана (Рисунок 3.7). Данное меню позволяет выйти к активности «Регистрации/авторизации», если это необходимо пользователю. И соответственно пользователь выходит из своего аккаунта и ему необходимо заново авторизоваться.

Рисунок 3.7 Главная страница

3.4.3. Погода

На данный момент не реализовано находится в приложении в качестве макета будущей активности. В дальнейшем вижу данную активность как сводка прогнозов погоды по каждому аэродрому, который отмечен на

активности с картой, запрос на обновление данных о погоде будет отправляться сервером N-раз в сутки и далее визуально обновляться у пользователей.

На данной активности присутствует меню с двумя вариантами, а именно «Выйти» - данный пункт перенаправляет на активность «Регистрации/авторизации» и пункт «На главную», который позволяет вернуться на «Главную страницу».

Рисунок 3.8 Аактивность «Погода»

3.4.4. Математическая модель прыжка

В активности «Прыгнуть» пользователь наблюдает большое количество активных полей, в которые он может занести различные характеристики, как своих габаритов, так и же и полётных параметров прыжка. Всего на активности находится 6 активных полей (Рисунок 3.9), в которых прописаны подсказки о назначении этого поля и физических величинах, в которых необходимо указывать значения. Кнопка для вычисления данных. 2 поля для вывода вычисленных данных.

Рисунок 3.9 Общий вид активности с расчётом математической модели прыжка

Поля:

1. «Вес парашютиста» - вес самого спортсмена в килограммах;
2. «Масса экипировки» - подразумевается парашютная система, каска и иные принадлежности, которые будут на спортсмене;
3. «Рост парашютиста» - рост спортсмена в метрах;
4. «Высота прыжка» - соответственно высота, с которой парашютист будет отделяться от самолёта. Делится на 4 категории внутри приложения, от которых зависит условно обозначенная критическая высота раскрытия парашюта, от которой в дальнейшем по формулам высчитывается критическое время открытия (timeCritOpen) и высота, на которой будет достигнута максимальная скорость (hightMaxSpeed):
    - 1 категория – от 200 до 1000 метров: значение критической высоты открытия 200 м, набора максимальной скорости на этих высотах не будет;

- 2 категория - от 1000 до 2000 метров: значение критической высоты открытия 300 м, примерное время набора максимальной скорости 12 с;
- 3 категория – от 2000 до 4000 метров: значение критической высоты открытия 350 м, примерное время набора максимальной скорости 12,5 с;
- 4 категория – выше 4000 метров: значение критической высоты открытия 400 м, примерное время набора максимальной скорости 14 с;

5. «Время полёта» - здесь имеется ввиду, время, которое парашютист собирается находиться в свободном падении;
6. «Spinner» с выборкой значений. Значения в поле spinner несут крайне важное смысловое и математическое значение, т.к. от положения парашютиста в воздухе зависит его скорость полёта, ускорение и время, затраченное на весь полёт. Есть возможность выбрать из 3-х положений, каждое из положений имеет свой определённый коэффициент сопротивления воздуха:
    a. «С максимальным ускорением» - самый маленький коэффициент = 0.0975;
    b. «Равноускоренное падение» - среднее принятое значение = 0.195;
    c. «Плашмя, руки разведены» - большое значение сопротивления = 0.2125.

На активности есть 2 поля для вывода ответов. В первое поле выводится значение максимальной скорости в м/с (maxSpeed), а во второе критическое время открытия (timeCritOpen) и высота, на которой будет достигнута максимальная скорость (hightMaxSpeed) с сопутствующими поясняющими фразами.

Все математические вычисления происходят после заполнения всех полей корректными данными и по нажатию на кнопку с парашютистом. Ниже,

на рисунках 3.10 – 3.13, будут представлены различные вариации работы данной активности.

Рисунок 3.10 Пустые поля для ввода

Рисунок 3.11 Макс. ускорение, корректное время

Рисунок 3.12 Падение плашмя, корректное время

Рисунок 3.13 Равноускоренное, некорректное время

Формулы, использованные для расчётов (переменные в формулах взяты из кода приложения и не являются общепринятыми):

Формула 3.1 Максимальная скорость парашютиста

$$maxSpeed = Math.sqrt((2 * G)/(p * Cx * St)), \quad (3.1)$$

G – сила тяжести, Cx – коэффициент сопротивления вохдуха, p – плотность воздуха, St – условная площадь тела парашютиста.

Формула 3.2 Критическое время раскрытия

$$ctimeCritOpen = Math.sqrt(2 * (Hightmax - hu)/a) \quad (3.2)$$

Hightmax – высота прыжка, hu – установленное значение критической высоты открытия, a – ускорение парашютиста с учётом сопротивления.

Формула 3.3 Ускорение парашютиста с учётом сопротивления

$$a = g - g * (p * Cx * St) \quad (3.3)$$

g – ускорение свободного падения.

Формула 3.4 Высота набора максимальной скорости

$$hightMaxSpeed = Hightmax - a * Math.pow(timeHMS, 2)/2 \quad (3.4)$$

timeHMS – установленное (для каждого промежутка высот) среднее время набора максимальной скорости

На данной активности присутствует меню с двумя вариантами, а именно «Выйти» - данный пункт перенаправляет на активность «Регистрации/авторизации» и пункт «На главную», который позволяет вернуться на «Главную страницу».

3.4.5. Информация о пользователе

В этой активности пользователь может редактировать информацию о себе. Как описывалось в пункте 3.4.1 поля уже заполнены данными о пользователе. В случае, если пользователь ни разу их не заполнял, в любом случае в верхнем поле, которое всегда является неактивным к заполнению, будет выведен никнейм пользователя, который он написал при регистрации. При входе в активность все 8 текстовых полей неактивны, при нажатии на кнопку «Редактировать», активность появляется у всех полей, кроме первого (никнейма). Поля: «Имя», «Фамилия», «Отчество», «Дата рождения», «Родной город», «Любимая дропзона», «Количество прыжков» (Рисунок 3.14).

При заполнении поля «Дата рождения» выполняется проверка на корректность даты, необходимо, чтобы она соответствовала виду

«дд.мм.гггг», если дата некорректна, выведется ошибка об этом и подсказка, какого вида должна быть дата.

При заполнении всех необходимых полей, пользователь, для того, чтобы данные записались в БД, должен нажать кнопку «Сохранить». Алгоритм её работы:

1) Данные готовятся для отправки по HTTP на сервер с одновременной проверкой даты на корректность (если поле пустое, то пустым оно и останется);
2) Данные отправляются на сервер для обновления таблицы «Информация», ожидается статус от сервера
    a. Если статус положительный, данные сохраняются в таблицу;
    b. Если сервер возвращает ошибку, она выводится на экран, сохранения данных не происходит.

На активности присутствует также кнопка «Отмена», которая работает как выход на «Главную страницу».

На активности присутствует меню с двумя вариантами, а именно «Выйти» - данный пункт перенаправляет на активность «Регистрации/авторизации» и пункт «На главную», который позволяет вернуться на «Главную страницу».

Сама активность в приложении называется более кратко - “О себе”, это сделано, чтобы не сильно загружать верхнюю панель меню в приложении.

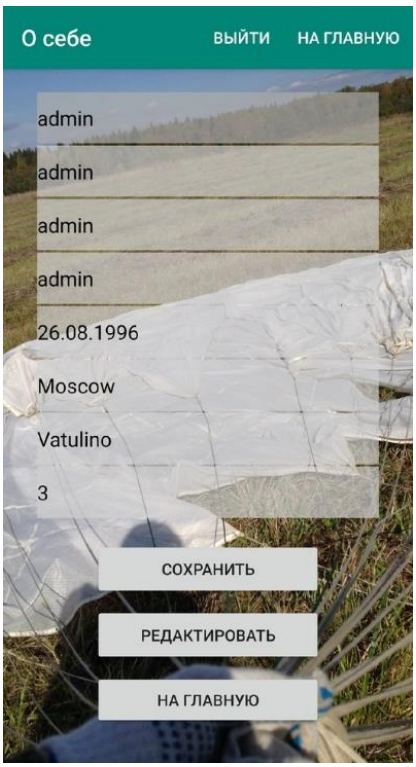

Рисунок 3.14 - Активность «О себе» с заполненными полями

3.4.6. История парашютного спорта

Данная активность основывается на встроенном конструкторе в Android Studio Scrolling_activity. Был незначительно изменён её дизайн и соответственно текст, выведенный на экран пользователя является историей создания парашютного спорта.

Данная активность в дальнейшем может дорабатываться и расширяться другими источниками информации. Также, как я писал ранее можно будет реализовать интересную функцию по выбору более популярных медиа файлов, которые выкладывают пользователи и сохранять эти файлы (с разрешения пользователей) в активности истории, что будет некоторым наглядным пособием для начинающих спортсменов. На рисунке 3.15 представлен вид активности «История»:

Рисунок 3.15 - Активность «История»

На данной активности присутствует меню с двумя вариантами, а именно «Выйти» - данный пункт перенаправляет на активность «Регистрации/авторизации» и пункт «На главную», который позволяет вернуться на «Главную страницу».

3.4.7. Карта с аэродромами

Карта с аэродромами реализовано в стандартном конструкторе Android Studio для работы с картами, при помощи API ключа. Пользователь по зажатию экрана может добавлять новые маркеры и удалять предыдущие. При нажатии на название маркера происходит пеход на активность погоды, где автоматически показывается погода в местном регионе.

Ниже, на рисунке 3.16 продемонстрирован вид активности в приложении:

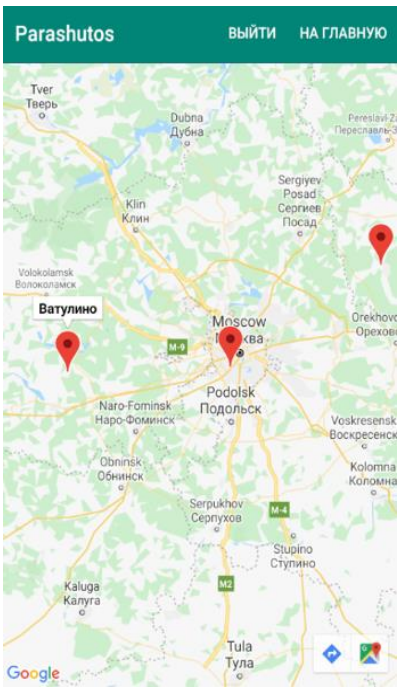

Рисунок 3.16 Активность «Карта дропзон»

3.4.8. Календарь прыжков

Календарь прыжков реализована при помощи стандартных шаблонных функционалов в Android Studio, но при этом никак не задействует встроенный в смартфон «Календарь». Вид данной активности в приложении представлен на рисунке 3.17:

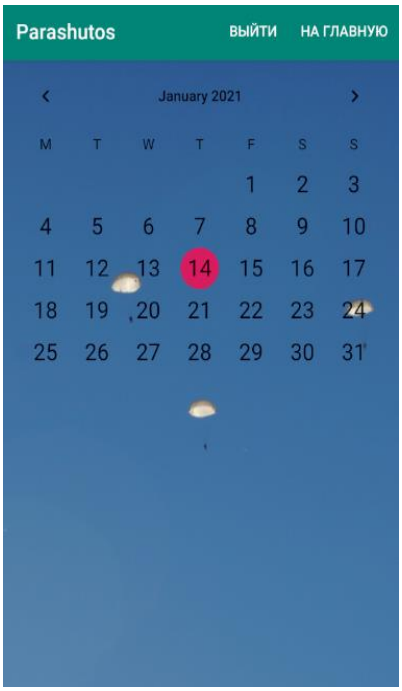

Рисунок 3.17 Активность «Календарь»

3.5. Тестирование и отладка приложения

Проводилось общее тестирование работоспособности приложения. Выполнялась проверка различных вариаций отказа, связанного с отсутствием сети интернет на смартфоне и при отсутствии работы локального сервера. Все данные проверки показали на смартфоне необходимые всплывающие информационные окна о выявленной проблеме, следовательно, обработчик ошибок и исключений работает в штатном режиме.

Было полностью протестирована интерфейсная составляющая мобильного приложения, как на визуальные отклонения, так и на функционирование всех элементов, включая работоспособность абсолютно всех кнопок в меню и на активностях. Также проводилась проверка восстановления сессии в приложении после экстренного завершения работы самого приложения или смартфона.

Все выявленные недочёта с развёртыванием интерфейса на различных моделях смартфонов были исправлены в ходе отладки приложения, которая проходила на встроенном в Android Studio эмуляторе смартфонов. Было создано около 5 различных машин с разными прописанными характеристиками и под различные версии операционной системы Android. На всех виртуальных машинах приложением в итоговом тестировании была продемонстрирована хорошая масштабируемость под разрешение экрана и функционирование всего встроенного функционала в полном объёме.

Подводя итоги тестирования и отладки мобильного приложения для планирования выполнения прыжков с парашютом можно сделать вывод, что приложение протестирована на 100% и имеет оптимизированный и отлаженный программный код. Помимо отлаженного программного кода в приложении в штатном режиме и с полноценной скоростью и откликом работают все функции, соответствующие первичным требованиям к функционалу данного мобильного приложения.

3.6.Руководство пользователя

Т.к. приложение имеет «дружелюбный» (интуитивно-понятный) интерфейс, руководство пользователя можно значительно сократить и поместить в небольшой список действий, которые доступны пользователю при использовании мобильного приложения, а также добавить некоторые особенности функционирования.

При открытии приложения пользователь имеет возможность выполнять следующие действия:

- Регистрация или авторизация;
- После авторизации возможность выбора одной из активности для перехода;
- Пользователь может просмотреть сводку походы в выбранном регионе;
- Пользователь имеет возможность просматривать локации аэродромов;
- Пользователь имеет возможность переходить при выборе локации аэродрома в активность погоды и просматривать погоду на выбранном аэродроме автоматически;
- Пользователь имеет возможность рассчитывать математические показатели прыжка в отдельной активности;
- В приложении есть возможность просмотра календаря и делать отметки о сделанных и планируемых прыжках;
- Пользователь имеет возможность заполнять и редактировать информацию о себе;
- Пользователь имеет возможность просматривать историческую сводку о парашютах и парашютном спорте;
- Каждый пользователь, входя в приложение, открывает свою собственную отдельную активную сетевую сессию.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы разработано мобильное приложение для планирования выполнения прыжков с парашютом на базе платформы Android.

В ходе выполнения работы проведён обзор основных конкурирующих приложений в тематике парашютного спорта с целью выявления требований к функционалу мобильного приложения.

Проведён обзор современных технологий для разработки мобильных приложений и иных вспомогательных систем, таких как API и фреймворки. Выбрана основная операционная система для мобильного приложения – ОС Android. Приложение построено на клиент-серверной архитектуре. В результате обзора технологий был выбран объектно-ориентированный язык программирования Java в совокупности с фреймворками Spring, Maven, Hibernate, Jackson, HTTP Client и с API библиотеками «Google Map SDK for Android» и «Weather API». В ходе обзора технологий, была выбрана система управления базами данных Oracle Database в среде SQLDeveloper для реализации реляционной БД в рамках мобильного приложения для хранения и записи данных о регистрации и личных данных пользователей.

Разработан и протестирован клиентский интерфейс приложения, созданный в Android Studio. При проведении тестирования был разработан общий план проведения ручного тестирования. Была проведена отладка написанного программного кода на выбранном языке программирования.

Разработано общее краткое руководство для пользователя. В общей сложности применено большое количество технологий, которые совмещены друг с другом в одном приложении и реализовано полноценное сетевое взаимодействие.

Итогом проделанной работы стало отлаженное и протестированная клиент-серверное мобильное приложение с реализацией всего запланированного функционала. В настоящий момент приложение функционирует на локальном сервере, т.к. для вывода подобного приложения

в общий доступ для скачивания необходимо пройти большой набор бюрократических условностей. В дополнение хочется отметить, что обозначены границы расширения функционала представленного мобильного приложения и обозначены места интеграции данного функционала (речь идёт про работу с датчиками телефона и работу с медиа данными пользователей). В дальнейших планах лежит доработка приложения до обозначенного функционала и выход его на общую платформу.

Работа выполнена в полном объёме в соответствии с ТЗ и удовлетворяет всем предъявляемым требованиям к функционалу.

Было придумано краткое название плиложения и название его иконки на смартфоне – «Parashutos».

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Java Documentation – [Электронный ресурс]: https://docs.oracle.com/en/java/ (Дата обращения: 18.09.2020).
2. Машнин Т.С. Технология Web-сервисов платформы Java. СПб.: БХВ-Петербург, 2012. – 560 с. (Дата обращения: 17.09.2020)
3. Documentation for app developers – [Электронный ресурс]: https://developer.android.com/docs (Дата обращения: 18.09.2020).
4. Б. Брила – Oracle Database 11g. – М.: ЛитРес, 2019. – 864 с. (Дата обращения: 05.10.2020).
5. Database Documentation – [Электронный ресурс]: https://docs.oracle.com/en/database/index.html (Дата обращения: 18.09.2020).
6. Учебник по Python – [Электронный ресурс]: https://docs.python.org/3/tutorial/index.html (Дата обращения: 19.10.2020).
7. Документация по PostgreSQL – [Электронный ресурс]: https://www.postgresql.org/ (Дата обращения: 19.10.2020).
8. БФАС, Парашютный спорт, История – [Электронный ресурс]: http://bfas.by/parachuting/o-sporte2/istoriya (Дата обращения: 01.10.2020).
9. Boogie – Skydiving logbook – [Электронный ресурс]: https://boogie.io/.
10. Информация о приложениях – [Электронный ресурс]: https://play.google.com/store/apps (Дата обращения: 21.10.2020).
11. Ключкова И.Ю., Мамонов С.С. – Моделирование движения парашютиста при раскрытом парашюте: Вестник РГРТУ. 2018. № 66. Часть 1. (Дата обращения: 05.09.2020).
12. Усачёв Ю.В., Курашин В.Н. – Математическая модель движений парашютиста: Вестник РГУ имени С.А.Есенина, 2010-№1(26) Статья 14. (Дата обращения: 05.09.2020).
13. Vijini Mallawaarachchi 10 Common Software Architectural Patterns in a nutshell: Towards Data Science, 2017. (Дата обращения: 25.09.2020).
14. Сбор статистики – [Электронный ресурс]: https://wordstat.yandex.ru. (Дата обращения: 05.09.2020).

ПРИЛОЖЕНИЕ

В качестве основного приложения к пояснительной записке идёт ссылка на проект на платформе GitHub, которая является крупнейшим хостингом для хранения любых IT проектов, а также обеспечивает контроль за версиями:

https://github.com/Romsnipz/Parashutos.git