

МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет компьютерных наук  
Кафедра программирования и информационных технологий

*Система покупки электронных билетов*

*Курсовой проект по дисциплине «Технологии программирования»*

09.03.02 Информационные системы и технологии  
*Программная инженерия в информационных системах*

Обучающийся \_\_\_\_\_ *В.М. Беспалов, 3 курс, д/о*

Обучающийся \_\_\_\_\_ *Д.О. Ступак, 3 курс, д/о*

Обучающийся \_\_\_\_\_ *И.П. Ткаченко, 3 курс, д/о*

Руководитель \_\_\_\_\_ *В.С. Тарасов, ст. преподаватель*

Воронеж 2021

## Содержание

Содержание .....	1
Введение .....	4
1 Постановка задачи .....	8
2 Анализ предметной области .....	9
2.1 Анализ существующих решений .....	9
2.2 Продуктовые воронки .....	11
2.3 Анализ средств реализации .....	12
3 Диаграммы, описывающие работу системы .....	13
3.1 Диаграмма прецедентов .....	14
3.2 Диаграмма классов .....	15
3.3 Диаграмма объектов .....	16
3.4 Диаграмма состояний .....	16
3.5 Диаграмма последовательностей .....	18
3.6 Диаграмма взаимодействия .....	21
3.7 Диаграмма активности .....	24
3.8 Диаграмма развёртывания .....	24
3.9 Диаграмма IDEF-0 .....	25
4 Реализация .....	26
4.1 Реализация серверного приложения .....	26
4.1.1 Используемые технологии .....	26
4.1.2 Структура приложения .....	27
4.1.3 Работа приложения .....	27
4.1.4 Тестирование .....	28

4.2 Реализация клиентского приложения .....	29
4.2.1 Используемые технологии .....	29
4.2.2 Структура приложения.....	30
4.2.3 Работа клиентского приложения.....	31
4.2.4 Тестирование .....	37
4.3 Реализация telegram-бота .....	37
4.3.1 Используемые технологии .....	37
4.3.2 Работа приложения .....	37
5 Пользовательские воронки.....	38
Заключение .....	43
Список использованных источников .....	44

## Введение

Перед любой информационной системой стоит цель удовлетворить потребности людей. Потребность в транспорте и передвижении – входит в их число. Экономия времени, собственных сил и красивый вид из иллюминатора – это всё относится к авиаперелётам. Зачастую не существует иного способа, как преодоление расстояния воздушным путём. «Постоянное наращивание и модернизация объектов инфраструктуры аэропортового комплекса позволит к 2030 г. увеличить пропускную способность до 64 млн. пасс» [1]

Цифры из статистики говорят о том, что воздушный транспорт еще проходит этап становления, и в будущем он только увеличит своё влияние. Поэтому данная сфера актуальна для исследования и реализации программного продукта.

Электронный формат покупки билетов упрощает не только взаимодействие между людьми, но и даёт покупателю актуальную и структурированную информацию. Покупатель анализирует представленные данные и выбирает лучший для себя вариант.

Разработка мобильной версии системы для покупки электронных авиабилетов обусловлена следующими причинами:

- открытая статистика компании Top Annie (Платформа, анализирующая мобильные данные), говорит о том, что рынок мобильных приложений находит все больше отклика от общества. Начиная с 2019 года пользователи со всего мира провели в мобильных приложениях на 50% больше времени от 2018 года;
- по оценке Criteo, (Ведущая рекламная платформа для открытого интернета): через мобильные приложения пользователи за сеанс просматривают в 4 раза больше товарных позиций, чем через адаптивный сайт. Крупные компании не задумываясь обеспечивают свои продукты мобильными приложениями, упрощающими покупку их услуг;

- наличие аналитики поведения покупателей даёт разработчикам приложений точную информацию по улучшению собственного продукта;
- постоянный доступ к смартфону позволяет покупателю, не отвлекаясь от важных дел найти интересующую его услугу. Это просто, удобно и дает пользователям свободу действий.

На территории России – основным потребителем мобильных приложений являются пользователи платформы Android, а именно «73% населения предпочитает Android и только 26% пользуется гаджетами на IOS» [2]. Исходя из данных сведений был сделан выбор в пользу разработки приложения для платформы Android.

Цель:

Основными целями разработки данной системы являются:

- упрощение взаимодействия покупателя с продавцом в лице авиакомпании;
- предоставление актуальной информации о билетах;
- повышение уровня осведомленности пользователя о ценах на авиабилеты.

Задачи:

- разработать мобильное приложение для платформы Android;
- провести аналитический сбор данных об активности пользователей в приложении;
- создать рекомендательную систему, которая содержит рейсы, интересные пользователю.

Анализ источников:

Создание выбранной системы включает в себя не только изучение документации Фреймворков, но и информации, дающей представление об

авиаперевозках. С какими данными работает данная сфера, какие факторы влияют на выбор определенного билета и главное направление – каким образом преподнести пользователю информацию, чтобы он пришёл к ожидаемому результату. Исходя из этого, были изучены следующие источники:

- мобильные приложения по продаже авиабилетов, такие как Aviasales, Аэрофлот, Туту.ру, Авиабилет;
- сервисы, предоставляющие данные о полётах, к примеру: Skyscanner, Amadeus travel;
- сайты авиакомпаний с информацией, которые описывают этапы покупки билетов и данные, необходимые для покупки.

В ходе анализа мобильных приложений по продаже билетов, были выявлены основные виды фильтров, ускоряющие выбор покупателем билета. Также были выявлены небольшие изъяны, реализацию которых учитывала бы данная разрабатываемая система.

Проанализировав сервисы, предоставляющие данные, было уменьшено количество фильтров, доступных к разработке. Приведённая на сервисах информация не обеспечивала фильтры необходимыми сведениями.

Анализ Сайтов авиакомпаний дал представление о процессе взаимодействия покупателя с авиакомпанией, начиная от покупки билета и заканчивая регистрацией покупателя (уже пассажира) на рейс.

Проведя анализ изложенных выше источников, было разработано техническое задание, описывающее основной функционал, интерфейс и этапы разработки мобильного приложения, а также была предложена рекомендательная схема подбора билетов.

Методология:

В качестве методологии разработки данной системы была выбрана каскадная модель разработки продуктов. Разработка системы ведётся поэтапно: каждый этап имеет ограниченные сроки, включает в себя

последовательное выполнение процессов, распределение задач и ведение отчетности.

## 1 Постановка задачи

Составить техническое задание проекта, а именно:

- проанализировать уже существующие дизайны систем поиска и покупки товаров;
- составить сценарии работы приложения;
- создать дизайн конечного приложения.

Разработать серверное приложение, а именно:

- создать проект приложения;
- разработать схему базы данных для хранения информации;
- создать сервис поиска билетов для заполнения базы данных приложения актуальной информацией;
- создать сервис для взаимодействия с клиентским приложением, который будет предоставлять запрошенные данные по авиабилетам;
- создать тесты для проверки работоспособности приложения перед его окончательной сборкой и развёрткой.

Разработать клиентское приложение, а именно:

- создать проект приложения;
- создать окна мобильного приложения с разработанным дизайном;
- создать контроллеры окон приложения;
- создать сервис для соединения с серверным приложением;
- создать тесты для проверки работоспособности приложения перед его окончательной сборкой.



В дополнение к всем вышеперечисленным пунктам необходимо:

- создать систему CI/CD для автоматизации тестирования и обновления приложений;
- создать сервис для работы с Telegram для рекомендаций пользователям интересных предложений;

## **2 Анализ предметной области**

### **2.1 Анализ существующих решений**

Анализ существующих решений будем проводить на основе данных сервисов:

Таблица 1 - Примеры существующих решений

<a href="https://bit.ly/3hktFDB">https://bit.ly/3hktFDB</a>	Aviasales
<a href="https://bit.ly/3bagPUt">https://bit.ly/3bagPUt</a>	Yandex.Flights

#### **1. Aviasales**

Общее описание:

Aviasales является метапоисковым сервисом и не продаёт билеты. Найдя нужный билет, пользователь переходит на страницу авиакассы или авиакомпании, чтобы совершить оплату. Оплата происходит на странице авиакомпании или авиакассы либо прямо в интерфейсе поисковика. В интерфейсе поисковика, то есть в формате экспресс-бронирования (assisted booking), продаются билеты «Уральских авиалиний» и S7 Airlines.

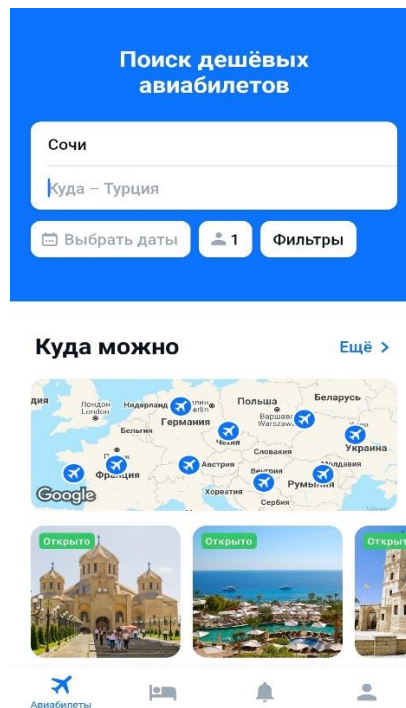


Рисунок 1 - Страница поиска Aviasales

#### Достоинства:

- понятный дизайн;
- агрегация информация от многих перевозчиков;
- большое количество страниц;
- возможность забронировать отель.

#### Недостатки:

- дизайн перегружен анимациями.

## 2. Yandex.Flights

#### Общее описание:

Сервис для подбора авиабилетов от Яндекса. Также как и Aviasales самостоятельно не продает билеты. При покупке билетов переходит на страницу авиакомпании для покупки билетов.

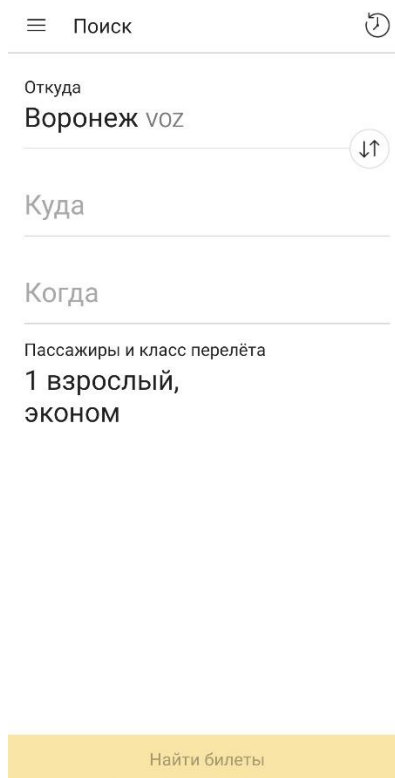


Рисунок 2 - Страница поиска Yandex.Flights

Достоинства:

— Простой интерфейс поиска.

Недостатки:

— Низкая скорость работы;

— Большое количество рекламы.

## 2.2 Продуктовые воронки

Рассмотрим количество шагов, которые необходимо пройти до основных функций приложения – покупка билета. При входе, пользователю необходимо ввести пункты назначения, а также дату. Далее, ему предлагается список рейсов на выбор. При нажатии на рейс, открывается окно с выбором рейса, где возможно приобрести билет. Таким образом, всего за четыре экрана

ВОЗМОЖНО

приобрести

билет.

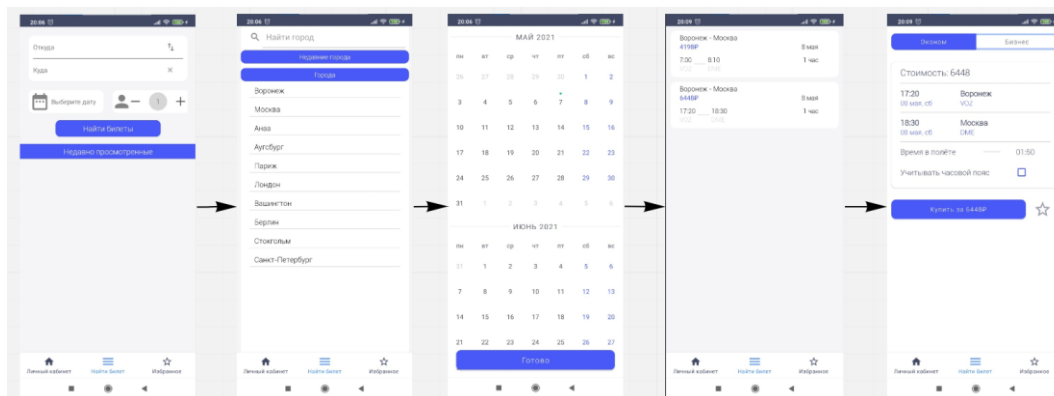


Рисунок 3 - Продуктовая воронка

## 2.3 Анализ средств реализации

В качестве средств реализации системы поиска авиабилетов были выбраны следующие технологии:

1. PostgreSQL была выбрана в качестве СУБД. Она является бесплатной наиболее популярной в данный момент и регулярной обновляемой системой управления с регулярно обновляемой базой данных.
2. В качестве IDE была выбрана среда AndroidStudio из-за того, что именно она рекомендуется для разработки Android-приложений под большинство версий данной системы. Она позволяет удобно проектировать приложения и проверять работу кода.
3. IntelliJ IDEA была выбрана в качестве среды разработки для API. Она выбрана из-за удобства использования в процессе работы. В процессе написания кода программистом она занимается построением синтаксического дерева, определением особенностей размещенных ссылок, анализом возможных путей исполнения операторов и передачи данных. Основываясь на полученных результатах, программа обращает внимание специалиста на существующие ошибки и самостоятельно устраняет их, предоставляет варианты автоматического

дополнения кода. Благодаря указанным особенностям она избавляет пользователя от повседневной рутины и позволяет ему сконцентрироваться на более важных задачах.

4. В качестве языка разработки используются Java, так как данный язык является широко распространенным в современном мире вследствие своей универсальности. Именно на данном языке написан гибкий и удобный фреймворк Spring, при помощи которого реализован API приложения.
5. В качестве сервера для работы с API используется Heroku из-за простоты использования и стабильности работы. Также Heroku позволяет использовать Docker контейнеры для запуска своего backend-приложения.

### **3 Диаграммы, описывающие работу системы**

Ввиду сложности разработки системы, были разработаны UML диаграммы, которые, ввиду универсальности, должны помочь проверяющему понять принципы работы системы.

### 3.1 Диаграмма прецедентов



Рисунок 4 - Диаграмма прецедентов

Каждый пользователь при взаимодействии с приложением имеет ряд возможностей, который наглядно представлен на рис. 4:

Авторизованный пользователь может:

- Искать рейсы;
- Просматривать детальную информацию о рейсе;
- Добавлять рейс в избранные;
- Удалять рейс из избранных;
- Покупать билеты;
- Просматривать историю платежей;
- Выходить из аккаунта.

Неавторизованный пользователь может:

- Авторизоваться;
- Искать рейсы;
- Просматривать детальную информацию о рейсе.

Администратор может:

- Просматривать статистику о использовании приложения другими пользователями.

### 3.2 Диаграмма классов

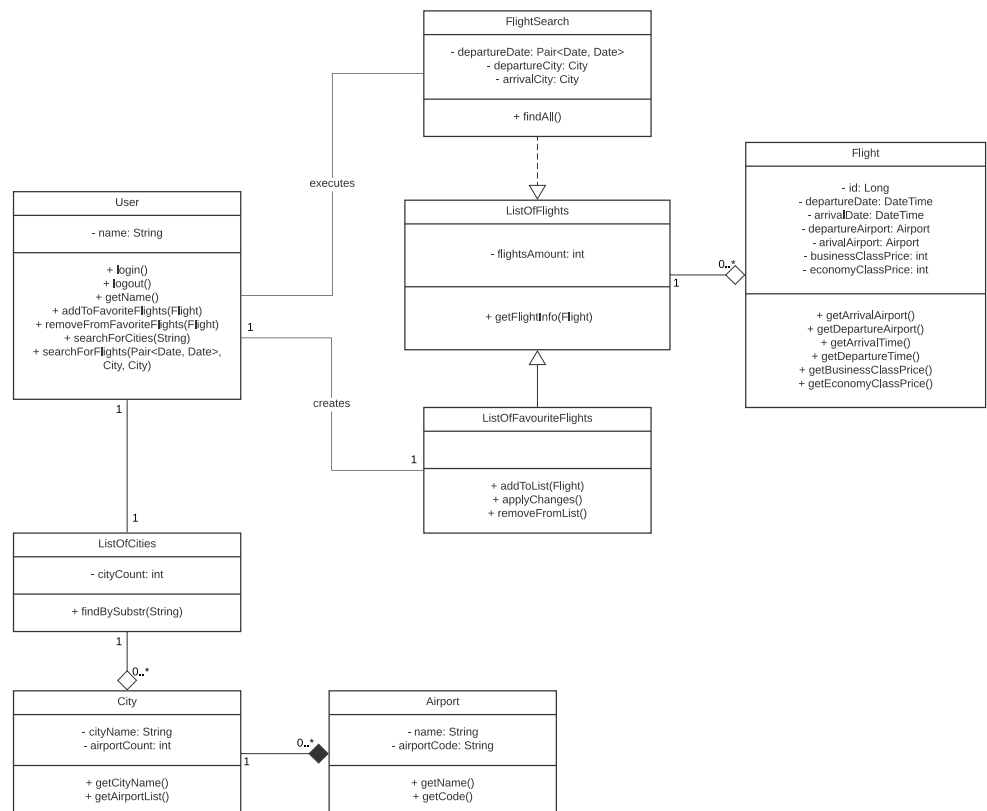


Рисунок 5 - Диаграмма классов

На рисунке 5 изображена диаграмма классов, отображающая основные компоненты системы и их взаимодействие.

### 3.3 Диаграмма объектов

Диаграмма объектов, изображенная на рисунке 6, отображает экземпляры классов и взаимодействие между ними.

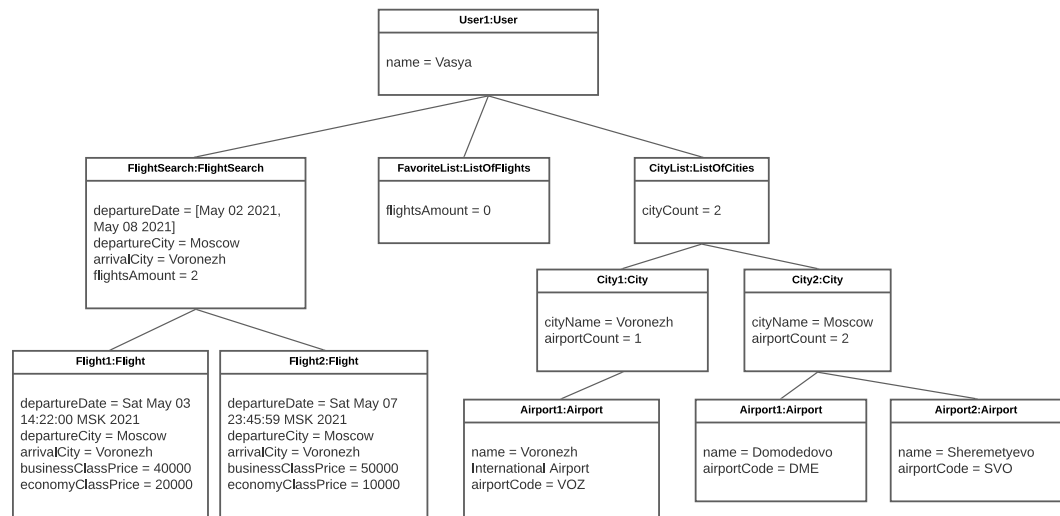


Рисунок 6 - Диаграмма объектов

### 3.4 Диаграмма состояний

Диаграмма состояний описывает состояния основных объектов системы.

На диаграмме, представленной на рисунке 6 представлено изменение состояний объекта пользователь. Начальное состояние представлено неавторизованным пользователем. Авторизация пользователя происходит с участием дополнительного сервиса. После успешной авторизации пользователь переходит в свое конечное состояние – авторизованный пользователь. В случае неуспешного входа будет предложена попытка повторного входа.



Рисунок 7 - Авторизация пользователя



Следующая диаграмма, отображенная на рисунке 8, описывает изменение состояний объекта рейс.

Начальное состояние включает в себя экран для поиска рейсов. После выбора фильтров для рейса начальное состояние переходит в состояние «Рейс с фильтрами». Далее следует поиск рейсов. В случае если рейсы не найдены, то состояние рейса не изменится, иначе состояние перейдет в состояние «Найденные рейсы». Далее представлена альтернатива – оформление покупки билета и добавление в избранное. Оба процесса требуют авторизации пользователя. При оформлении покупки - помимо выбора рейса, требуется информация о пассажирах. В этом случае рейс меняет состояние на «Билет на рейс с данными о пассажирах», а после оплаты – «Купленные билеты на рейс». В случае добавления рейса в избранное – состояние изменяется на избранный рейс. Сущность приходит в свое конечное состояние.

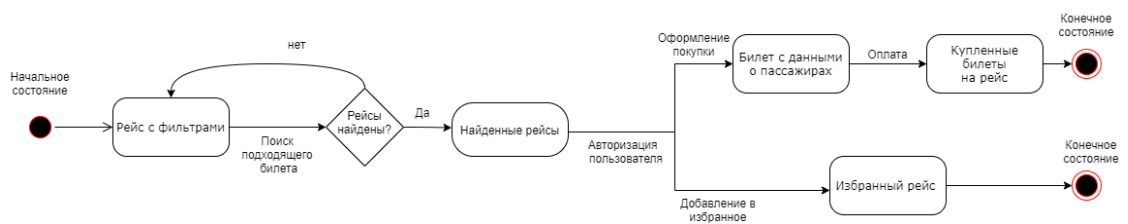


Рисунок 8 - Диаграмма состояний рейса

Диаграмма, отображенная на рисунке 9, описывает изменение состояний для сущности администратор.

Начальное состояние представлено пользователем сайта. После прохождения авторизации начальное состояние переходит в состояние авторизованный администратор, которому доступна функция: получение статистики. Сущность перешла в свое конечное состояние.



Рисунок 9 - Диаграмма состояний администратора

### 3.5 Диаграмма последовательностей

Диаграмма последовательностей включает в себя временное описание процессов, протекающих в системе. По горизонтали указаны объекты, участвующие в процессе, по вертикали – промежутки активности объектов.

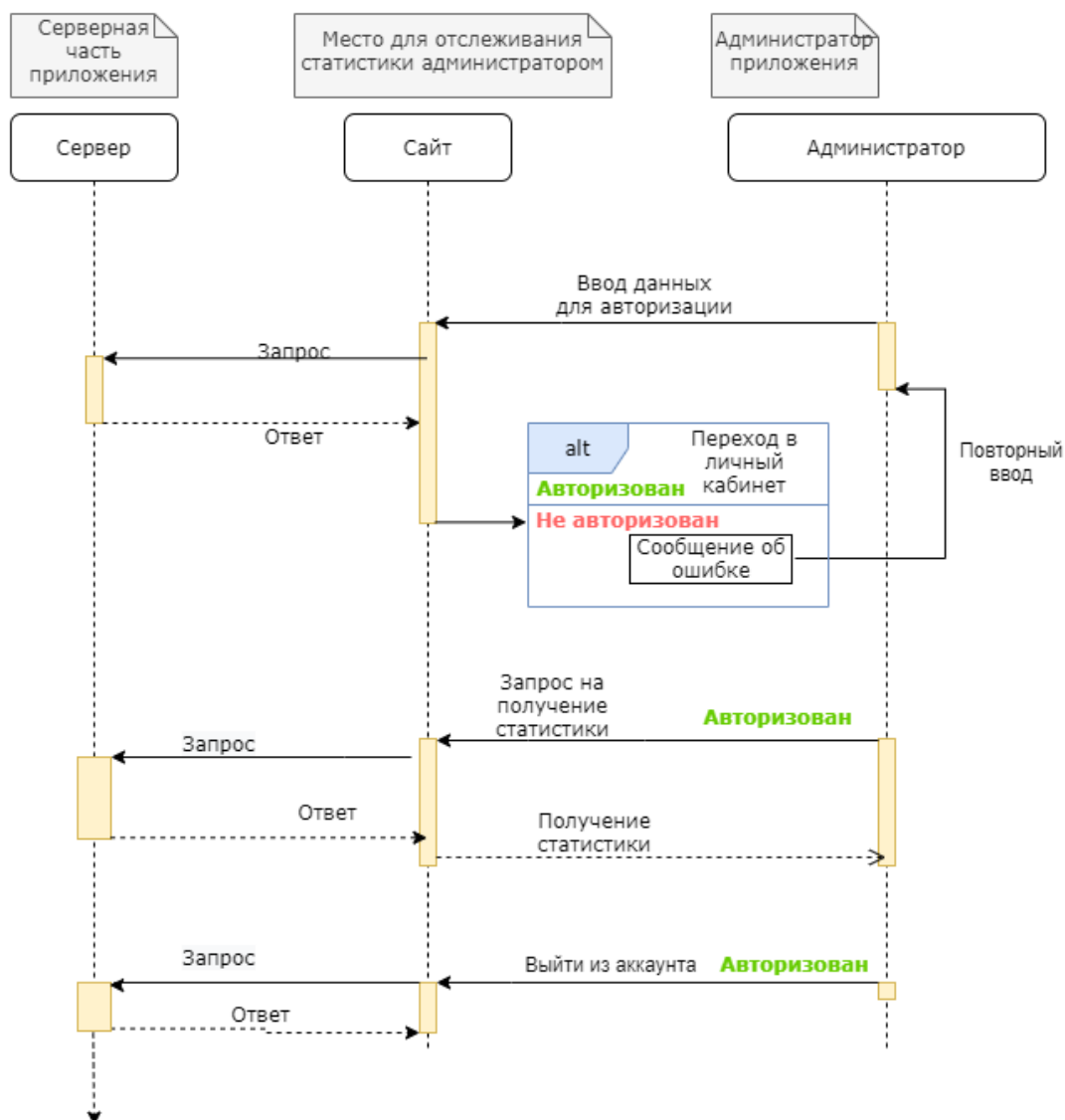


Рисунок 10 - Последовательности действий администратора

На диаграмме, отображенной на рисунке 10, представлено взаимодействие объектов – сервер, сайт, администратор. Для возможности выполнения действий – администратору необходимо пройти авторизацию. Авторизация проходит с участием серверной части приложения. После получения ответа сервера происходит или повторный ввод данных или открывается доступ к личному кабинету. После успешной авторизации

администратору доступна функция получения статистики о пользователях.  
При условии, что администратор авторизован – он может выйти из аккаунта.

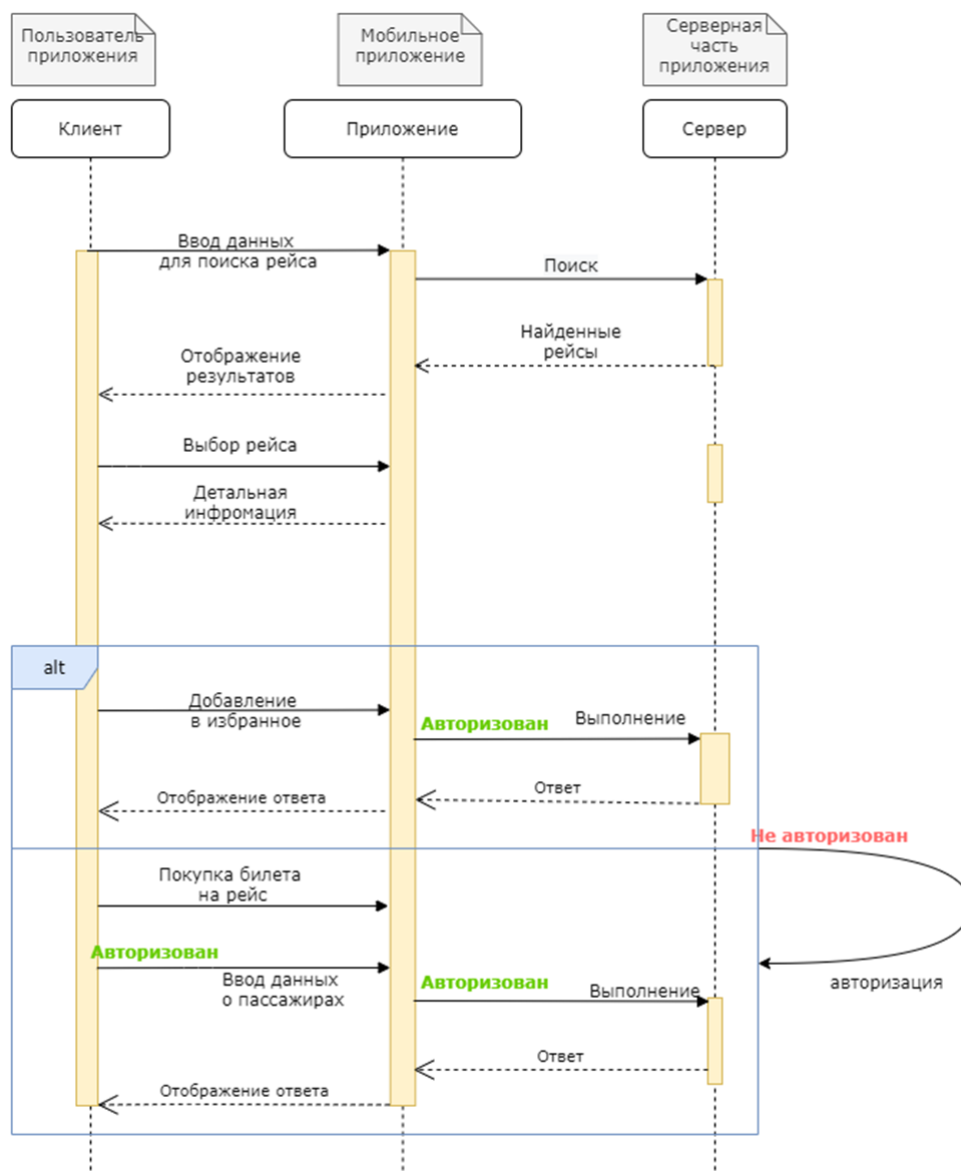


Рисунок 11 - Диаграмма последовательностей действий пользователя, сервера и приложения при работе с рейсами

Для сущности пользователь представлены операции поиск рейсов, добавление в избранное и покупка билетов.

Для поиска рейсов, пользователю необходимо ввести данные для фильтра. После запроса к серверу на экране приложения будут отображены найденные рейсы с указанными фильтрами. Теперь пользователю доступен детальный просмотр рейса и альтернативный выбор следующих действий: добавление в избранное или покупка билетов на рейс.

Для добавления в избранное пользователь должен быть авторизован. Авторизация проходит на стороне клиента с участием дополнительного сервиса. В случае, если пользователь уже авторизован, происходит запрос на сервер с выполнением операции добавления в избранное. После ответа сервера происходит отображение его ответа на экране приложения.

При покупке билетов на рейс также необходима авторизация, после прохождения авторизации необходимо заполнить сведения о пассажирах. За этим следует запрос на сервер. После ответа сервера происходит отображение его содержимого на экран приложения.

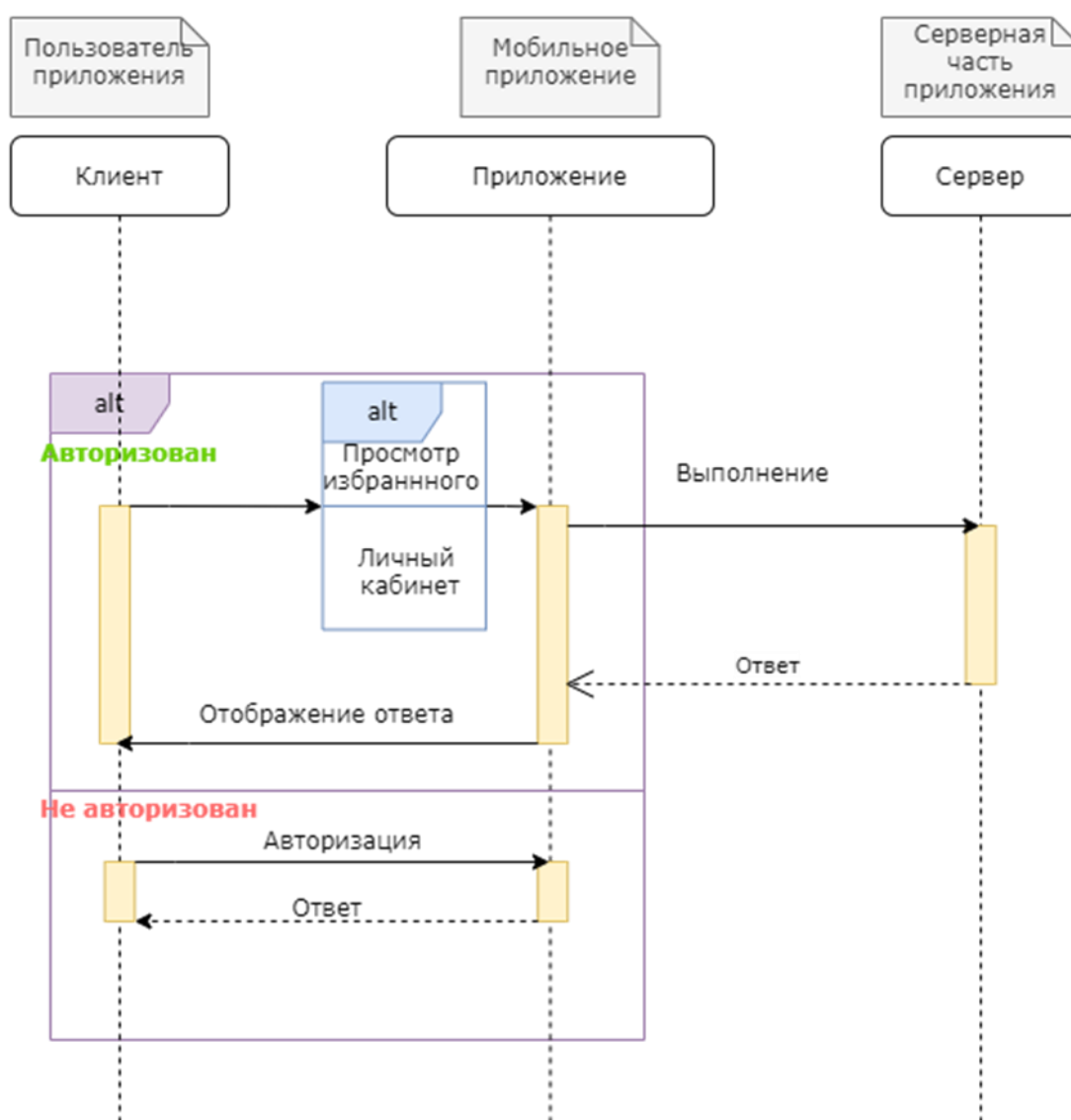


Рисунок 12 - Последовательности при работе с авторизованным или неавторизованным клиентами

В продолжение функций, доступных пользователю: операции просмотра избранного и просмотра личного кабинета. На диаграмме представлен альтернативный блок: авторизованный – неавторизованный пользователь. Если пользователь авторизован – ему доступны функции просмотра, избранного с дальнейшим удалением рейсов из него и просмотр личного кабинета. Обе функции делают запросы к серверу и отображают на экран приложения информацию, содержащуюся в ответе. В случае, если пользователь неавторизованный - ему будет предложена авторизация с участием вспомогательного сервиса.

### 3.6 Диаграмма взаимодействия

Диаграмма взаимодействия описывает взаимодействия объектов системы без учета временной составляющей, лишь последовательность действий, описывающих каждую операцию.

На рисунке 13 представлены объекты пользователь, приложение, сервер. Пользователю предоставлена операция по поиску рейса с заданными фильтрами. Приложению отправляется запрос на сервер, сервер обрабатывает запрос и отправляет ответ, который будет предоставлен пользователю для дальнейшего изучения.



Рисунок 13 - Взаимодействия при поиске рейсов с заданными фильтрами

Диаграмма, изображенная на рисунке 14 описывает авторизацию пользователя с участием приложения. Приложение использует сторонний сервис для авторизации. После попытки авторизации будет получен ответ, прошел ли пользователь авторизацию.

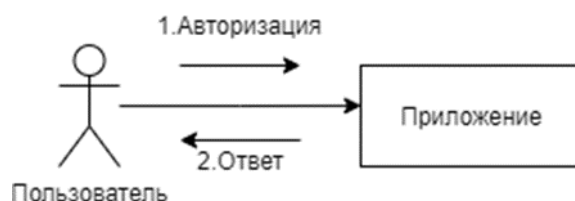


Рисунок 14 - Взаимодействия при авторизации

На рисунке 15 основное действие выполняется авторизованным пользователем. Пользователь производит покупку билетов. После внесения данных о пассажирах будет предложена оплата билетов. В случае покупки – приложение отправит соответствующий запрос на сервер и получит ответ, который приложение отобразит пользователю.



Рисунок 15 - Взаимодействия при покупке билетов

На рисунке 16 основное действие выполняется авторизованным пользователем. Пользователь производит добавление рейса в избранное. После добавления – приложение отправит соответствующий запрос на сервер и получит ответ, который приложение отобразит пользователю.



Рисунок 16 - Взаимодействия при добавлении в избранное

На диаграмме, отображенной на рисунке 17, основное действие выполняется авторизованным пользователем. Пользователю предложены операции по просмотру избранных рейсов и личного кабинета. После выбора

нужной операции приложение отправит соответствующий запрос на сервер и получит ответ, который приложение отобразит пользователю.



Рисунок 17 - Взаимодействия при просмотре личного кабинета

Далее представлены диаграммы для объекта администратор.

На рисунке 18 представлена авторизация администратора. Приложение отправляет соответствующий запрос на сервер. Сервер присылает ответ приложению, а приложение сообщает администратору об успешности или неуспешности авторизации.



Рисунок 18 - Взаимодействия администратора при авторизации

На диаграмме, изображенной на рисунке 19, представлен выбор операций, доступных авторизованному администратору. Операции: просмотр статистики и выход из аккаунта. Приложение отправляет соответствующий запрос на сервер. Сервер присылает ответ приложению, а приложение отображает данные, присланные сервером.



Рисунок 19 - Взаимодействия при просмотре статистики

### 3.7 Диаграмма активности

Данная диаграмма, отображенная на рисунке 20, показывает, какие действия будут выполняться в системе.

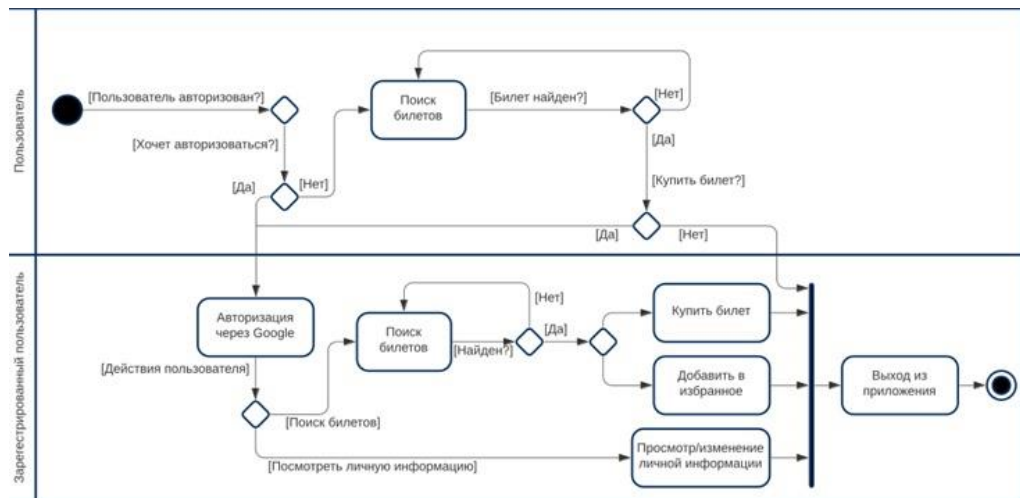


Рисунок 20 - Диаграмма активности

На данной диаграмме представлено два типа пользователей: неавторизованные и авторизованные.

Неавторизованным пользователям доступны только активности поиска билетов и авторизации.

Авторизованные пользователи могут искать билеты, покупать их или добавлять в избранное и просматривать свою личную информацию (список избранных билетов, историю покупок).

### 3.8 Диаграмма развёртывания

Данная диаграмма показывает, где и какие части приложения расположены физически и как они связаны.



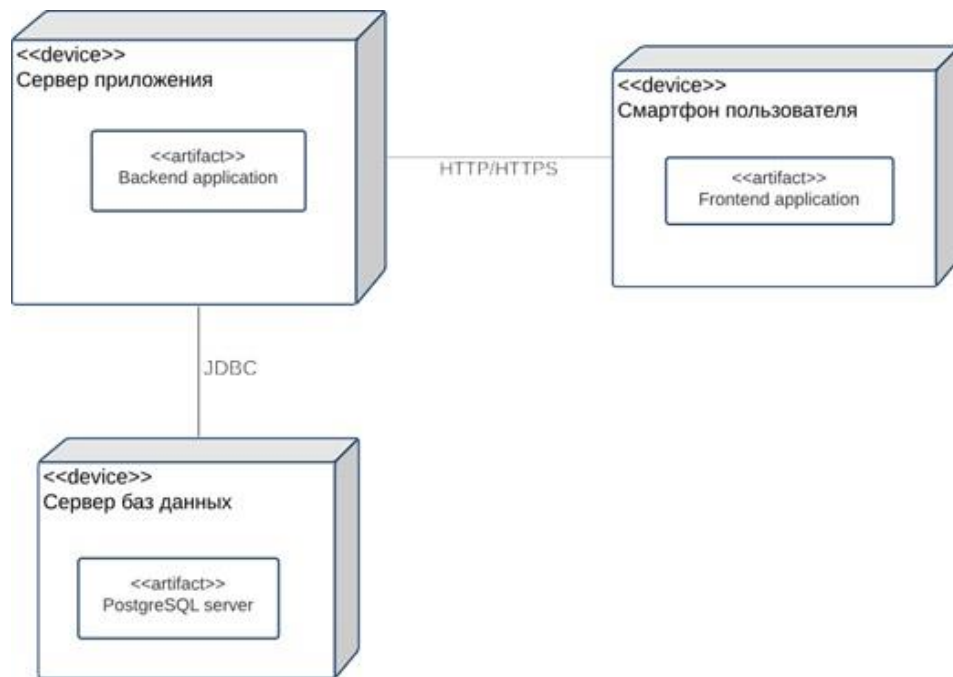


Рисунок 21 - Диаграмма развёртывания

На данной диаграмме видно, что приложение состоит из трёх частей, расположенных на трёх девайсах: серверная часть приложения, база данных и клиентская часть приложения.

Серверная часть связывается с базой данных посредством драйвера JDBC и интернет-соединения.

Клиентская часть приложения связывается с серверной посредством HTTP/HTTPS запросов.

### 3.9 Диаграмма IDEF-0

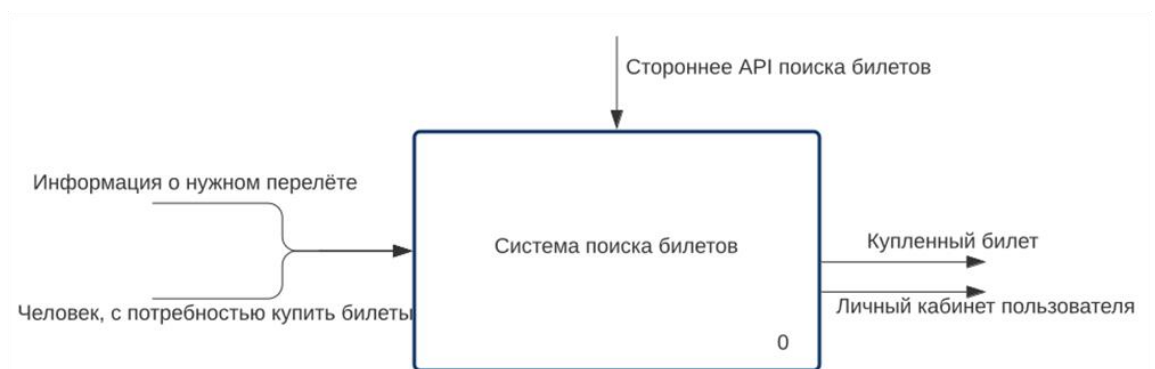


Рисунок 22 - Диаграмма IDEF-0

## **4 Реализация**

Сервис поиска билетов реализует REST архитектуру и делится на три приложения: основная серверная часть, отвечающая за поиск и покупку билетов, клиентская часть, которая отвечает за поиск и отображение информации пользователю, и telegram-бот, который отвечает за рассылку сообщений пользователям, которые на неё подписаны.

### **4.1 Реализация серверного приложения**

#### **4.1.1 Используемые технологии**

Для реализации backend приложения были использованы следующие технологии:

- Java – строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems. Данный язык программирования имеет множество фреймворков, которые можно использовать как основной при реализации;
- Gradle – система автоматической сборки проектов, построенная на принципах Apache Ant и Apache Maven. Позволяет автоматизировать сборку приложения, а также собирать проекты, модули которых написаны на разных языках программирования;
- Spring Framework – универсальный фреймворк с открытым исходным кодом для Java-платформы. На данный момент один из самых популярных фреймворков для создания веб-приложений на Java. Используется в качестве основного фреймворк приложения;
- Gson - библиотека от Google для сериализации классов приложения в json объекты и десериализации json объектов в классы.

### 4.1.2 Структура приложения

Приложение разделено на несколько модулей:

- db (DataBase) – модуль, отвечающий за работу с базой данных;
- services – модуль, реализующий бизнес-логику приложения, а также интеграцию SkyScanner сервиса;
- main – модуль, содержащий точку запуска приложения и контроллеры HTTP-запросов к приложению (controllers).

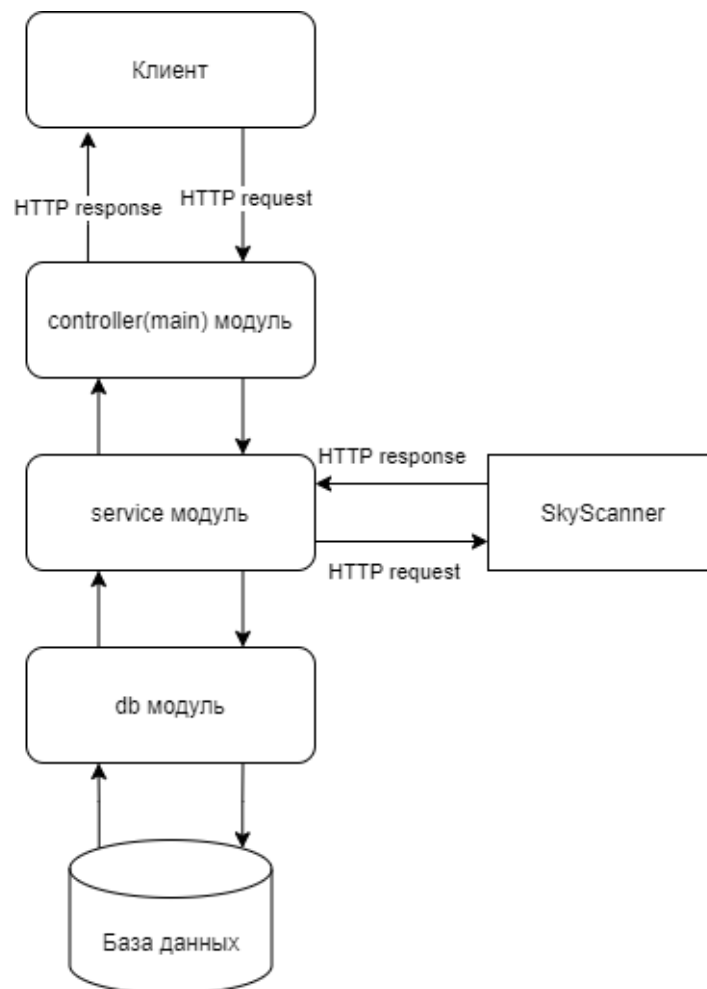


Рисунок 23 - Схема работы приложения

### 4.1.3 Работа приложения

Общий алгоритм приложения состоит в том, что сервер принимает HTTP запросы на точки входа и вызывает методы бизнес-логики, возвращая результаты выполнения методов в виде HTTP ответа.

Для более подробного описания работы приложения рассмотрим запрос поиска перелётов.

Приложению приходит HTTP запрос, содержащий json объект с информацией об искомом перелёте.

```
{
  .... "originPlace":{
  ....   .... "PlaceId":"String",
  ....   .... "PlaceName":"String",
  ....   .... "CityId":"String",
  ....   .... "CountryName":"String"
  .... },
  .... "destinationPlace":{
  ....   .... "PlaceId":"String",
  ....   .... "PlaceName":"String",
  ....   .... "CityId":"String",
  ....   .... "CountryName":"String"
  .... },
  .... "outboundDate":"dd-mm-yyyy",
  .... "inboundDate":"dd-mm-yyyy"
}
```

Рисунок 24 - Общий вид HTTP запроса для поиска билетов

Информация, содержащаяся в json объекте, передаётся в DTO класс Flight и контроллер вызывает метод search из класса FlightService, который находится в модуле services.

Затем в классе FlightService выполняется запрос к SkyScanner для поиска доступных авиаперелётов.

В результате выполнения данного запроса приложение получает json объект, из которого нужные пользователю данные передаются в DTO класс Flight и возвращаются в результате вызова метода в модуль main.

В этом модуле, на основе данных в DTO классе, генерируется json объект, который затем отправляется в виде HTTP ответа на клиентскую сторону.

#### 4.1.4 Тестирование

Серверная часть приложения была написана с использованием нескольких модулей («контроллеры», «сервисы», «база данных»), что позволяет тестировать каждый модуль по-отдельности, выявляя неточности. В

ходе реализации использовалось интеграционное тестирование, включающее в себя проверку работы сразу нескольких модулей, а именно модулей «сервисы» и «база данных».

Для написания тестов была произведена настройка контекста тестов. В конфигурационный файл были добавлены аннотации по работе с контекстом приложения, импорт данных из файла “application.properties” и указание адреса для сканирования компонентов.

Использовались юнит-тесты – это тесты, применяемые к отдельным методам. С их помощью были протестированы классы, описывающие бизнес-логику приложения – взаимодействие поступающих моделей данных, дополнительной логики и работы с базой данных. Таким образом были протестированы методы для добавления в рейсы, недавние рейсы, города, недавние города, для добавления и удаления из избранного, сверка добавленных элементов и их реальное количество. Создавались объекты исходных данных, вызывался соответствующий метод и сверялись ожидаемые и полученные результаты. В методы поступали уже провалидированные модели, поэтому проверять объекты на заполненность полей не считалось необходимым. В результате каждый метод, был обёрнут необходимой логикой и протестирован.

## **4.2 Реализация клиентского приложения**

### **4.2.1 Используемые технологии**

Для реализации клиентского приложения были использованы следующие технологии:

- Java – строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems. Данный язык программирования имеет множество фреймворков, которые можно использовать как основной при реализации;

- Android SDK – набор компонентов для создания и сборки Android приложений;
- AndroidX (Android Extension Library) – библиотека поддержки Android. Содержит вспомогательные библиотеки, которые не поставляются вместе с операционной системой Android.

Также были использованы дополнительные библиотеки из открытых источников кода.

#### **4.2.2 Структура приложения**

Приложение разделено на несколько модулей:

- UI – модуль, отвечающий за работу модулей интерфейса приложения и их логику;
- API – модуль, реализующий соединение клиентского приложения с сервером, а также классов DTO;
- MainActivity – модуль, содержащий точку запуска приложения и реализации точек взаимодействия с приложением(endpoints);
- Additions – модуль, содержащий в себе изменённые реализации сторонних библиотек;
- Utils – модуль, содержащий в себе утилитарные функции, которые могут использоваться в различных частях приложения для обработки данных.

Также в корневой директории проекта находятся класс экрана запуска приложения (SplashScreenActivity), который запускает всё приложение, а также класс, ответственный за работу всего приложения (MainActivity), который отвечает за функционирование основных функций приложения

Помимо прочего, для простоты изменения внешнего вида приложения, описание графики элементов содержится в отдельных xml файлах. Эти файлы хранятся в модуле для хранения постоянных ресурсов приложения(res) и

содержат описания внешнего вида элементов приложения, которые считываются при запуске и сборке приложения.

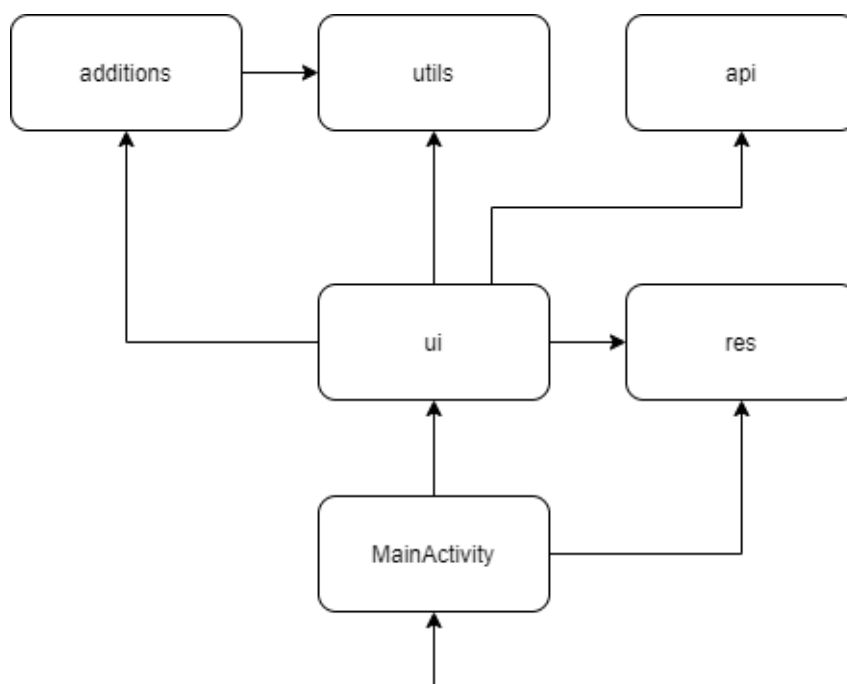


Рисунок 25 - Общая схема взаимодействия модулей клиентского приложения

#### 4.2.3 Работа клиентского приложения

Основной принцип работы приложения заключается в реагировании на действия пользователя и их дальнейшей обработке. Так пользователь может переключаться между вкладками приложения, выбирать вводить данные для поиска и совершать другие действия, предусмотренные приложением. Также приложение совершает HTTP запросы к серверу, если ему необходимо получить данные, необходимые пользователю, к примеру, информацию о билетах, которые ищет пользователь, или информацию для его личного кабинета. Так как данному приложению необходимо обращаться к серверу посредством сети Интернет, то необходимо обрабатывать ошибки соединения. В таком случае на экран выводится сообщение о невозможности подключения к серверу.

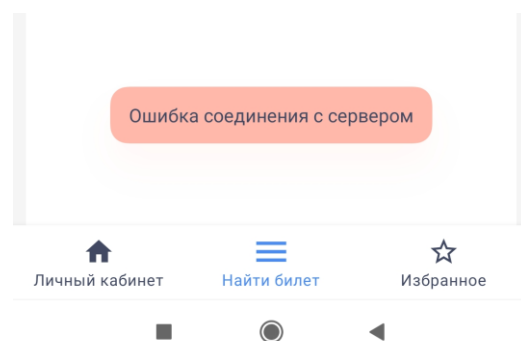


Рисунок 26 - Сообщение об ошибке соединения

Также действия пользователя запоминаются приложением для сбора статистики использования сервиса.

Чтобы более подробно разобраться в работе клиентского приложения, разберём сценарий поиска билетов.

Для этого пользователь переходит на вкладку поиска и заполняет поля необходимой информацией.

Данные с сервера загружаются постепенно. Это сделано для того, чтобы как можно быстрее отображать краткую информацию о рейсах пользователю.



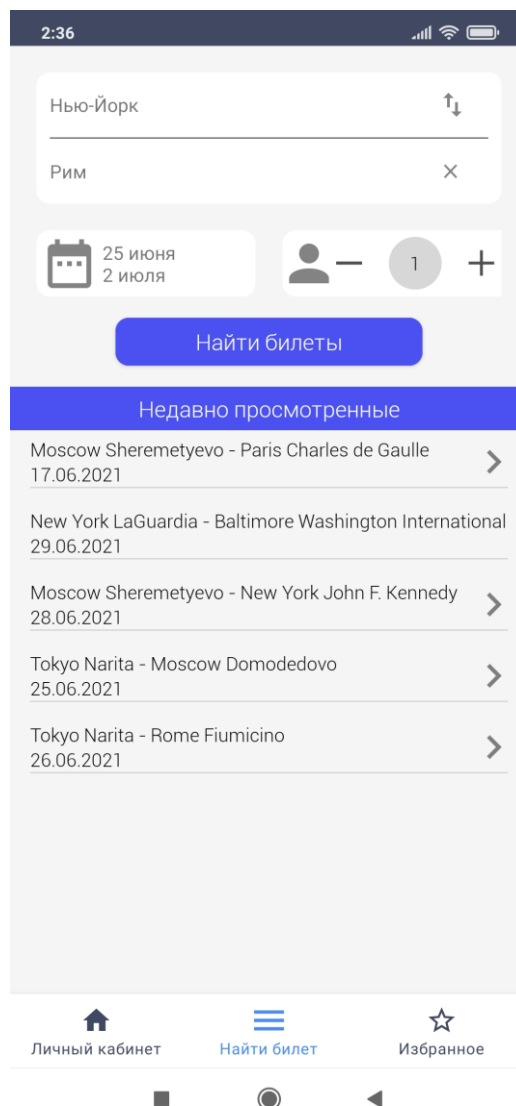


Рисунок 27 - Экран поиска билетов

После ввода данных пользователь нажимает кнопку «Найти билеты».

Приложение получает информацию о нажатии и считывает внесённые пользователем данные. Если какие-то данные введены неверно, то на экране появляется соответствующая информация.

Если все данные введены правильно, то приложение создаёт внутреннюю задачу для выполнения запроса к серверу и передаёт ей введённые данные. Если при выполнении задачи возникает какая-либо ошибка, на экране появится соответствующее уведомление.

После получения информации от сервера, приложение передаёт данные на экран устройства.

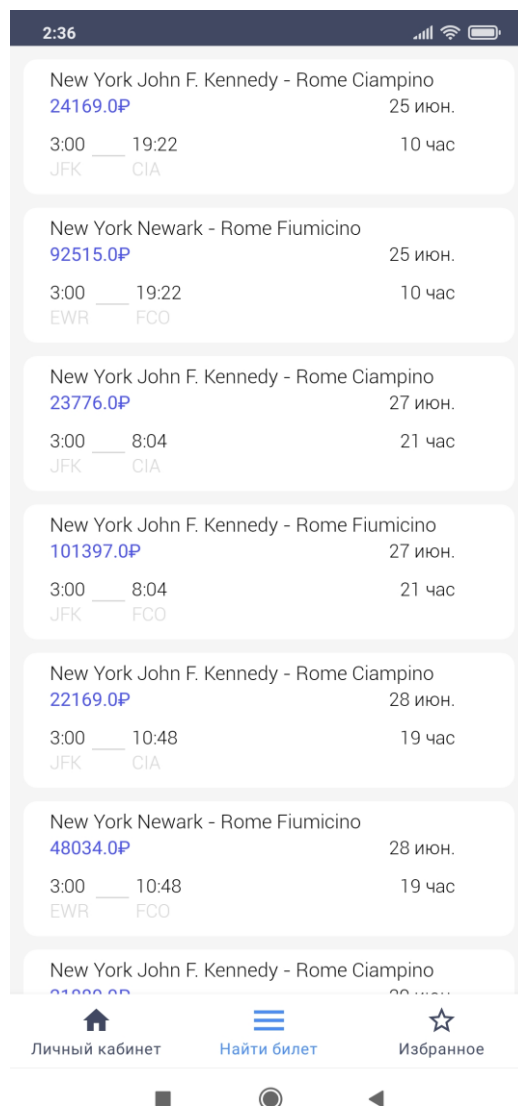


Рисунок 28 - Список найденных рейсов

Далее пользователь может приобрести билет, если зарегистрирован в приложении. Если пользователь является неавторизованным, то приложение перенаправит его на вкладку входа и регистрации, которые реализованы через OAuth сервис авторизации компании Google.

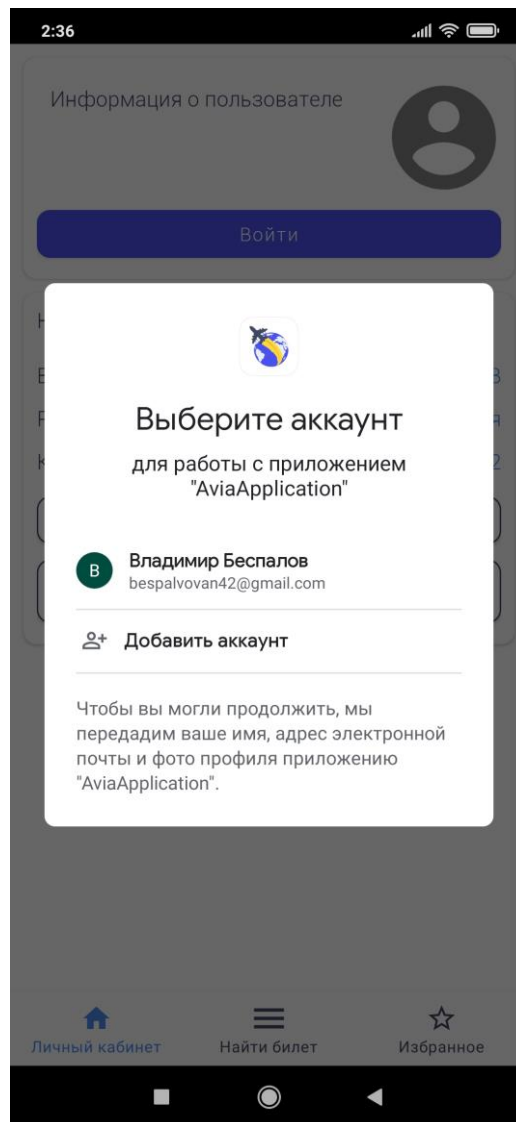


Рисунок 29 - Окно авторизации Google

В итоге, в результате использования вкладки «Найти билет» или вкладки «Избранное» приложение должно переводить пользователя на экран «Информация о полёте», на котором содержится полная информация о выбранном рейсе.

На данном экране находится кнопка «Добавить в избранное» в виде звездочки, которая добавляет в список избранных рейсов пользователя содержимое текущего экрана. Кнопки «Эконом»-«Бизнес» изменяют информацию о цене билета в соответствии с данными, которые были приняты от сервера.

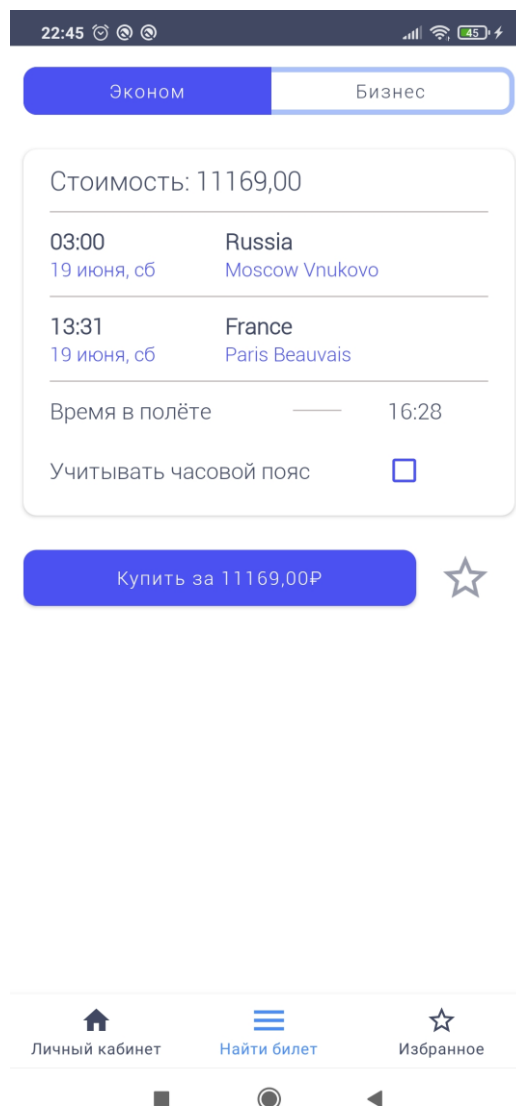


Рисунок 30 - Экран «Информация о рейсе»

При нажатии на кнопку «Купить за» должен осуществляться переход на экран покупки билета, где ему нужно было бы ввести необходимые данные, но данная функция не была реализована, так как необходимо заключение договора с организациями-перевозчиками, что невозможно сделать ввиду специфики курсового проекта. Вместо этого билет просто добавляется в список купленных билетов пользователя в том случае, если пользователь авторизован.

Также пользователь может добавить данный билет в «Избранное», чтобы отслеживать цены на данный перелёт. Данное отслеживание также реализовано в системе рассылки telegram-бота.

#### **4.2.4 Тестирование**

Для тестирования клиентской части приложения использовалась библиотека Junit версии 5.7.1. С помощью данной библиотеки были протестированы функции, ответственные за работу с API backend части приложения. Была проведена проверка корректности обработки ответа от сервера в различных случаях, а также корректность обработки случая, при котором клиентская часть сервиса не получает ответа от серверной.

### **4.3 Реализация telegram-бота**

#### **4.3.1 Используемые технологии**

Для реализации telegram-бота были выбраны следующие технологии:

- Telegram – кроссплатформенный мессенджер с функциями VoIP, позволяющий обмениваться текстовыми, голосовыми и видеосообщениями. Также позволяет создавать ботов с помощью TelegramAPI;
- Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью. Данный язык программирования хорошо подходит для написания сценарных программ. Также имеет большое количество библиотек, предназначенных для работы с TelegramAPI;
- Aiogram – библиотека на языке Python для взаимодействия с TelegramAPI, построенная на принципах асинхронных запросов. Обладает большим функционалом, позволяя создавать telegram-ботов разной сложности.

#### **4.3.2 Работа приложения**

Telegram-бот представляет собой небольшой отдельный клиент-сервер, который выполняет запросы к Telegram и выполняет свои скрипты при получении ответов. В данном случае бот сохраняет необходимую

информацию о пользователях, перешедших по пригласительной ссылке. Вся информация изначально содержится в этой ссылке, и представляет собой email пользователя, указанный при регистрации в приложении.

В отдельном потоке бот проверяет, не появилось ли более дешёвых билетов на направления, которые пользователи, подписанные на рассылку, отметили избранными. Если на отмеченные перелёты появились более дешёвые варианты, бот отправляет соответствующее сообщение пользователю с информацией о перелётах и новых ценах.

Также бот имеет панель администрации, в которую можно попасть введя команду «/admin» и пароль. На данной панели можно поменять промежуток проверки новых билетов, а также посмотреть, сколько пользователей подписано на рассылку.

Поскольку бот создан на языке Python с библиотекой aiogram, то его можно легко модернизировать и изменять или добавлять функционал.

## **5 Пользовательские воронки**

Для сбора данных, необходимых о популярности приложения были использованы событийные воронки, реализованные при помощи платформы «AppMetrica» от компании «Яндекс». Суть событийных воронок заключается в том, что они показывают как общее, так и относительное число пользователей, которые совершили некоторые, заранее определенные действия, в необходимом порядке. Такие действия называются событиями, а «AppMetrica» позволяет просматривать полный их список в промежутки времени, определенные пользователем:

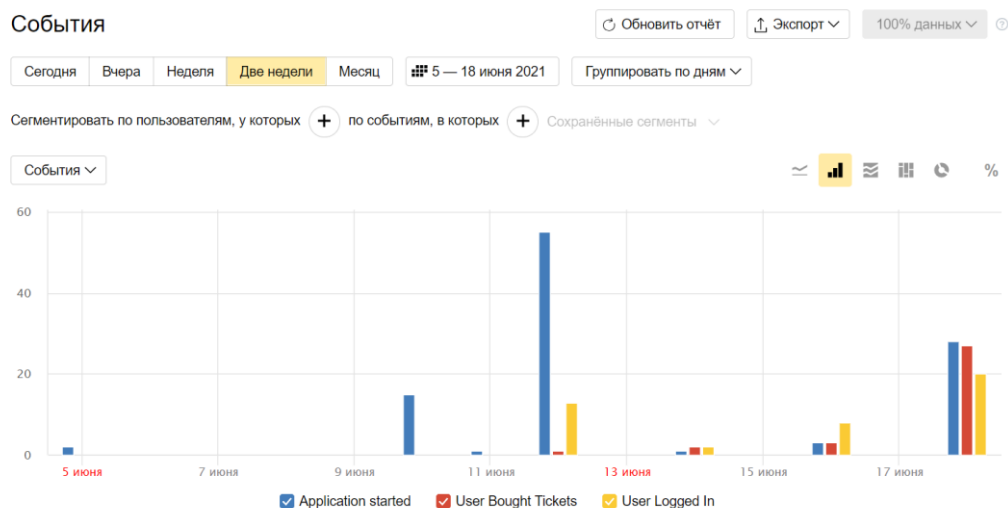


Рисунок 31 - Список событий за последний месяц

На данном рисунке отображается общее число событий, но «AppMetrica» позволяет показывать общее число событий, совершенных уникальными пользователями. Также данная платформа позволяет создавать пользовательские событийные воронки, которые будут отображать, какое число пользователей смогло дойти до определенного шага. Главной целью данного приложения является продажа билетов, следовательно, необходимо реализовать воронку, отображающую процент пользователей, которые смогли дойти до финального шага: покупки билета.

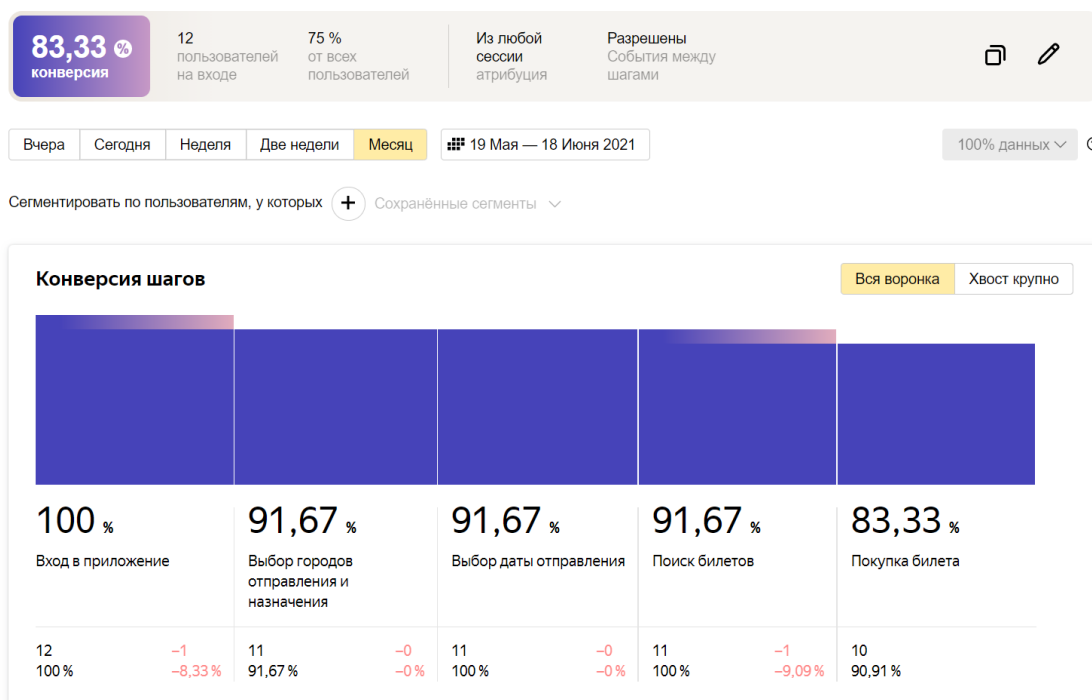


Рисунок 32 - Воронка «Поиск и покупка билета»

Также нам необходимо знать, насколько удобным является меню «Избранное», поэтому мы реализовали воронку, в которой пользователь добавляет рейс в избранное, а позже покупает билет на данный рейс.

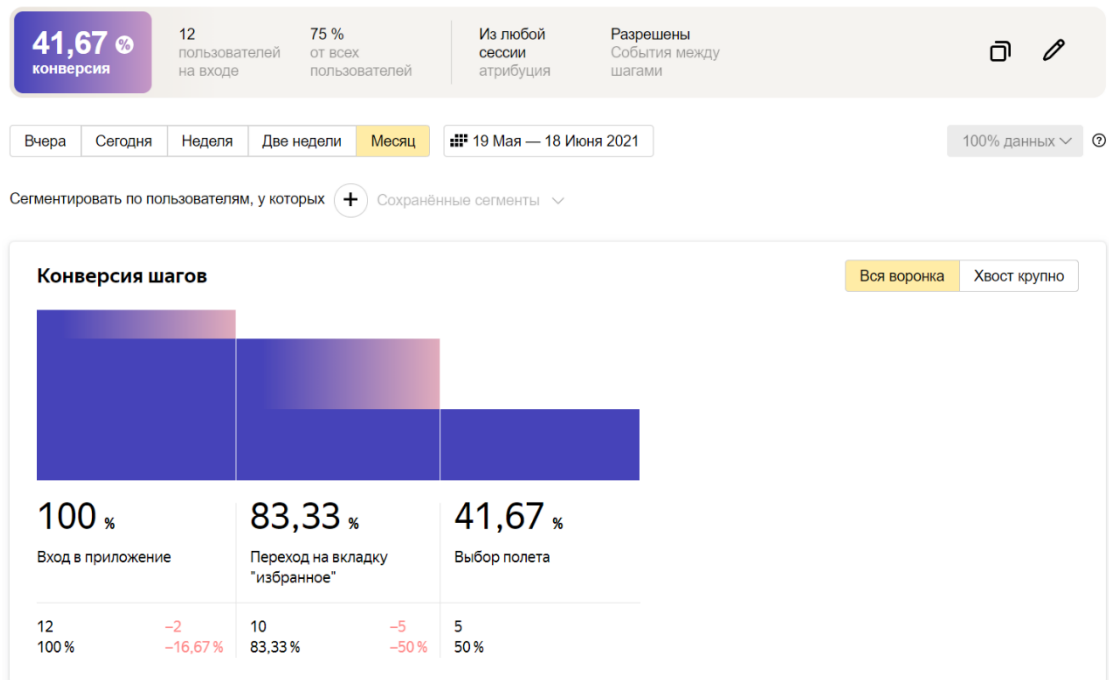


Рисунок 33 - Воронка «Покупка билета через избранное»

Реализация покупки в целом предусматривает создание меню «История платежей», для этого также была создана отдельная воронка (включает в себя авторизацию пользователей)



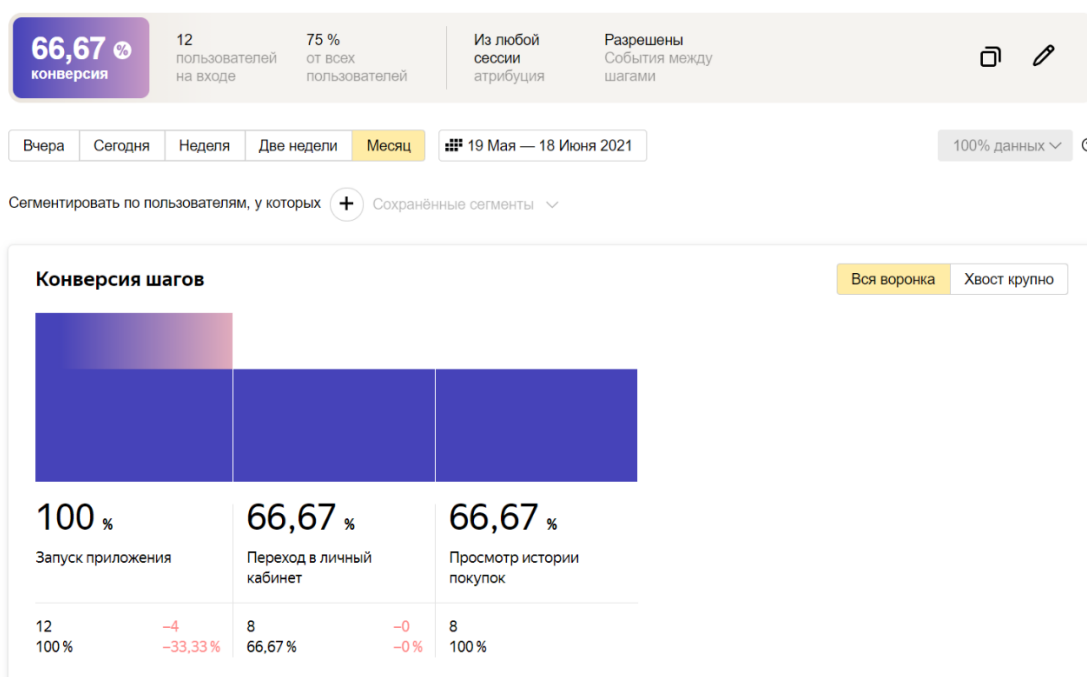


Рисунок 34 - Воронка «Доступ к истории покупок»

Механика событий платформы «AppMetrica» позволяет отслеживать общее число событий за определенный срок. К примеру, мы можем узнать общее число билетов, купленных за последние две недели:

<input type="checkbox"/> События	События	Пользователи	Событий на пользователя	% от всех пользователей
<input type="checkbox"/> Всего	33 100 %	5 100 %	4,13	62,5 %
<input checked="" type="checkbox"/> User Bought Tickets	33 100 %	5 100 %	4,13	62,5 %

Рисунок 35 - Билеты, купленные за последние две недели

Для того, чтобы узнать общее число зарегистрированных пользователей применяется статистика платформы «Google Cloud Platform», которая показывает количество новых пользователей за выбранный промежуток времени, а также среднее время взаимодействия с приложением.

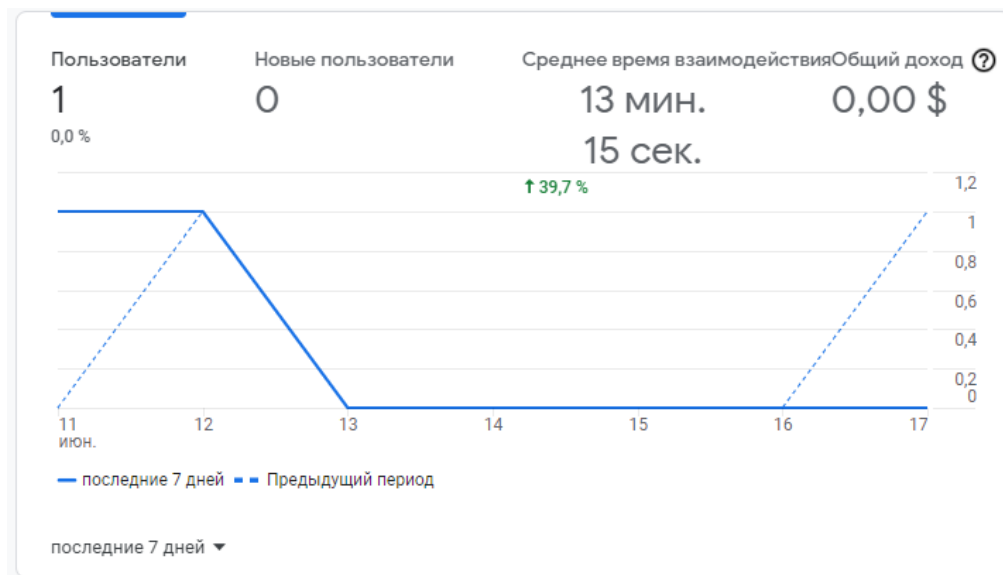


Рисунок 36 - Google Cloud Console новые пользователи

Также данная платформа позволяет просмотреть страны, в которых проживают зарегистрированные пользователи.

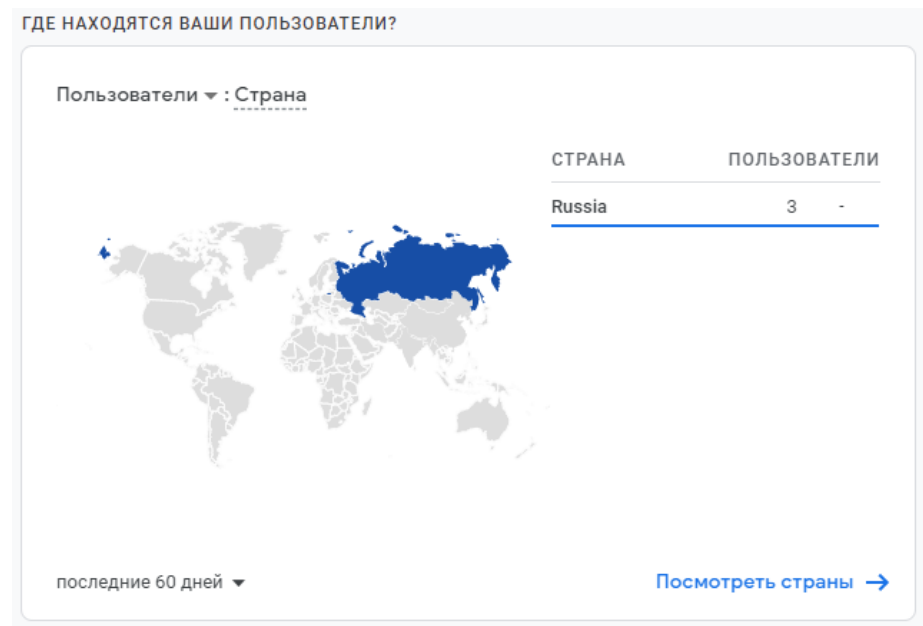


Рисунок 37 - Расположение пользователей приложения по странам

## **Заключение**

В ходе выполнения данного проекта, была проведена исследовательская работа в области сервисов поиска и бронирования авиабилетов. На основе собранных данных было создан сервис поиска и бронирования билетов состоящий из трёх приложений: сервера, мобильного приложения и telegram-бота.

В итоге, сервер отвечает за обработку информации и бизнес-логику сервиса, мобильное приложение отвечает за взаимодействие пользователя с сервисом, а telegram-бот отвечает за систему рассылки.

Данный сервис был создан в соответствии поставленным задачам, а также имеет потенциал в модернизации в виде добавления нового функционала, такого как бронирование отеля в пункте назначения. Но, к сожалению, данные функции не добавлены в текущую версию приложения, так как необходимо заключать договора с коммерческими организациями, отвечающими за отели. Также покупка билетов не была реализована до конца ввиду невозможности получения доступа к бронированию билетов напрямую у перевозчиков.

Протестирована функциональность сервиса с точки зрения удобства использования API, а также проведены тесты как серверной, так и клиентской части приложения.

### **Список использованных источников**

1. Филина, В.Н. Стратегии развития рынка авиационных транспортных услуг / В.Н. Филина; Проблемы прогнозирования, 2019
2. Bloomberg URL: <https://kanobu.ru/news/kto-populyarnее-vmire-ios-ili-android-novaya-detalnaya-infografika-otvechaet-naetot-izvechnyj-vopro-417479/> (дата обращения: 23.03.2021).