МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет *компьютерных наук*

Кафедра *программирования и информационных технологий*

*Приложение по подбору авиабилетов*

*Курсовая работа*

09.03.02 *Информационные системы и технологии*

*Программная инженерия в информационных системах*

Допущен к защите

Зав. Кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_*С.Д.* Махортов*, к.ф.- м.н., доцент* \_\_.\_\_.20\_\_

Обучающиеся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*А.Е. Копылова, О.А. Транина 3 курс, д/о*

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*, ст. преподаватель*

Воронеж 2018

# Содержание

[Содержание 2](#_Toc8308442)

[Введение 4](#_Toc8308443)

[1. Постановка задачи 5](#_Toc8308444)

[2. Анализ 7](#_Toc8308445)

[2.1. Анализ предметной области 7](#_Toc8308446)

[2.2. Анализ задач 9](#_Toc8308447)

[2.2.1. Задача поиска авиабилетов 9](#_Toc8308448)

[2.2.2. Задача хранения данных пользователя, приложения и его настроек 9](#_Toc8308449)

[2.2.2.1. Подзадача хранения закладок 10](#_Toc8308450)

[2.2.2.2. Подзадача хранения истории поиска 10](#_Toc8308451)

[2.2.3. Задача отображения пользовательского интерфейса 11](#_Toc8308452)

[2.3. Анализ средств реализации 12](#_Toc8308453)

[2.4. Диаграммы 12](#_Toc8308454)

[3. Реализация 21](#_Toc8308455)

[3.1. Задача поиска авиабилетов 21](#_Toc8308456)

[3.2. Задача хранения данных пользователя, приложения и его настроек 21](#_Toc8308457)

[3.2.1. Подзадача хранения закладок 21](#_Toc8308458)

[3.2.2. Подзадача хранения истории поиска 22](#_Toc8308459)

[3.3. Задача отображения пользовательского интерфейса 24](#_Toc8308460)

[Заключение 24](#_Toc8308461)

[Скелет приложения 25](#_Toc8308462)

[Подсистема поиска авиабилетов 25](#_Toc8308463)

[Подсистема хранения истории поиска (класс SearchHistoryRepository) 31](#_Toc8308464)

[Подсистема хранения закладок 31](#_Toc8308465)

[Подсистема отображения пользовательских интерфейсов. 32](#_Toc8308466)

[Вид пользовательского интерфейса 48](#_Toc8308467)

[Список источников 54](#_Toc8308468)

[Приложения 54](#_Toc8308469)

# Введение

В современном мире мы наблюдаем проникновение информационных технологий практически во все сферы человеческой деятельности, не стала исключением и сфера предоставления услуг по поиску авиабилетов.

Учитывая темп жизни современного человека, время является важнейшим ресурсом. Не удивительно, что с распространением интернета приобрели высокую популярность различные сервисы онлайн-поиска для мобильных устройств. Ведь они позволяют существенно упростить и ускорить процесс получения информации о различных услугах. Очевидно, что использование программного приложения для предоставления информации о перелетах и ценах дает преимущества такие, как:

* экономия времени
* более удобная форма поиска с широким выбором фильтров
* более разнообразные предложения

При разработке проекта используется каскадная модель, которая включает в себя следующие этапы:

1. Определение требований
2. Проектирование
3. Конструирование (также «реализация» либо «кодирование»)
4. Воплощение
5. Тестирование и отладка (также «верификация»)
6. Инсталляция
7. Поддержка

Также в процессе разработки будет использоваться парадигма объектно-ориентированного программирования.

# Постановка задачи

Целью курсового проекта является создание приложения, работающего на операционной системе Android.

Основную функциональность разрабатываемого приложения отражает диаграмма прецедентов, изображенная на рисунке 1.

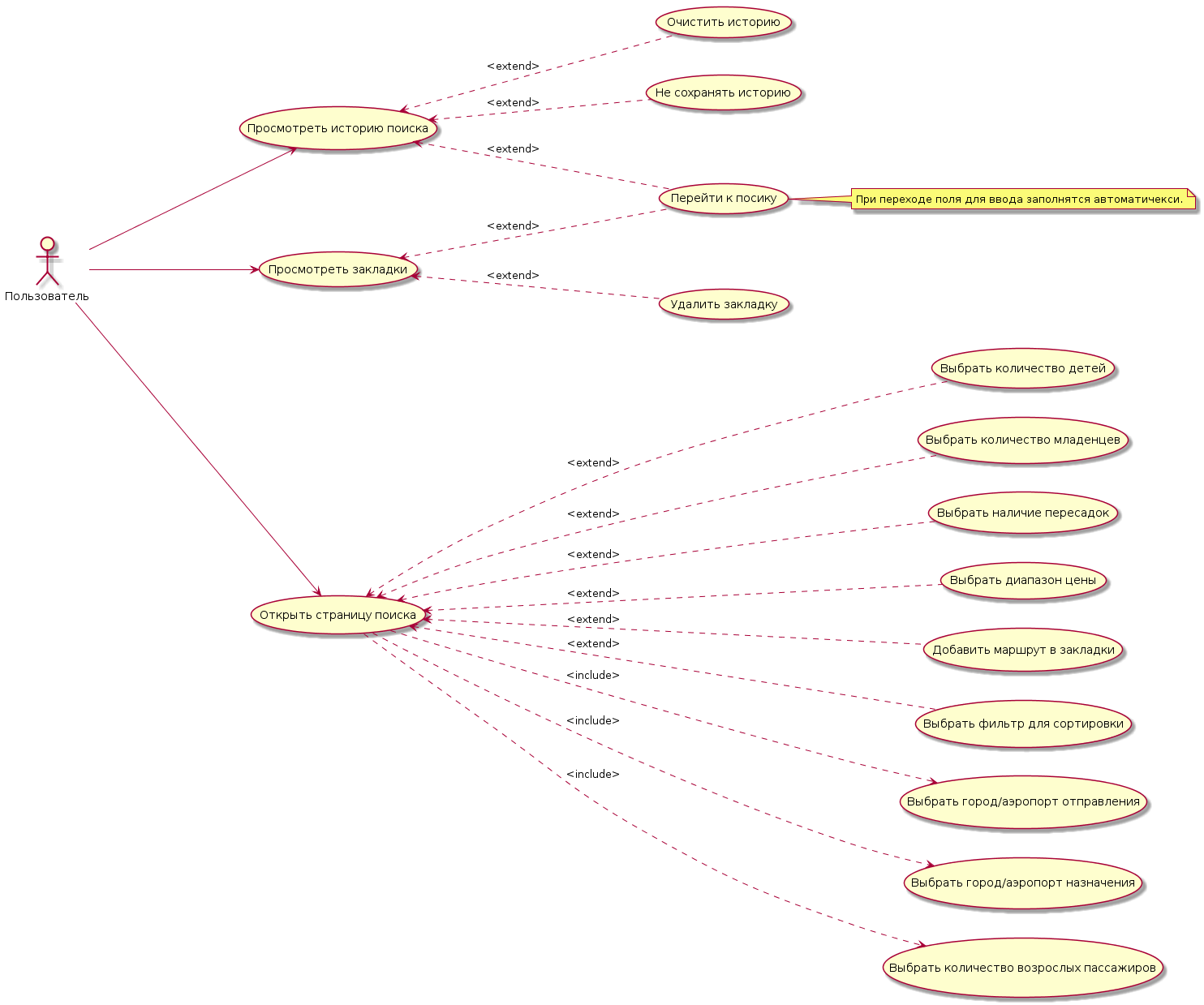


Рисунок 1. Диаграмма прецедентов.

Пользователь обладает следующими возможностями:

* подбор авиабилетов на рейсы в один конец с возможностью добавления информации о количестве пассажиров;
* подбор авиабилетов на рейсы в оба конца с возможностью добавления информации о количестве пассажиров;
* добавление маршрута в закладки;
* просмотр закладок;
* удаление закладки;
* просмотр истории поиска;
* удаление элемента истории поиска;
* очистка истории поиска;
* возможность не сохранять историю поиска**.**

Система должна соответствовать следующим требованиям:

1. Использование нескольких источников данных (API).
2. Подбор не только прямых рейсов, но и рейсов с пересадками.
3. Реализовать все возможности пользователя, описанные выше.

Завершенный проект представляет собой полностью функционирующее Android –приложение, соответствующее требованиям, описанным выше.

# Анализ

## Анализ предметной области

Для поиска авиабилетов минимально необходимой информацией являются следующие данные:

* пункт отправления;
* пункт назначения;
* количество взрослых пассажиров;
* количество детей;
* количество младенцев;
* дата отправления;
* тип салона (бизнес-класс или эконом).

Однако часто билеты покупают с обратной дорогой или, как это еще называют, в обе стороны. В этом случае к уже перечисленным данным добавляются данные об обратном билете, в частности дата отправления из пункта назначения.

Также при поиске рейсов из пункта назначения в пункт прибытия необходимо учитывать не только прямые рейсы, но и рейсы с пересадками, то есть такие, у которых есть промежуточные точки остановок в пути. Однако не для всех пассажиров такие рейсы являются приемлемыми, так как они существенно снижают комфорт путешествия, добавляя к времени в дороге время на саму пересадку. Для таких случаев следует предоставить возможность пользователю приложения убрать такие рейсы из итоговых результатов поиска.

По причине того, что результатом поиска, как правило, является длинный список разнообразных вариантов рейсов в пункт назначения, пользователю следует предоставить возможность сортировки результатов по следующим критериям:

* по убыванию цены;
* по возрастанию цены;
* по времени в дороге (от самого долгого путешествия к самому короткому);
* по времени в дороге (от самого короткого путешествия к самому долгому);
* по количеству пересадок (от большего к меньшему);
* по количеству пересадок (от меньшего к большему).

В области поиска авиабилетов существует большое количество мобильных приложений, работающих на операционной системе Android. Рассмотрим наиболее крупные из них:

* Tutu.ru

Туту.ру (tutu.ru) – российский онлайн сервис путешествий, предоставляющий услуги по поиску и покупки билетов на все виды транспорта, а именно ж/д и авиа перевозки и автобусы. Также предоставляет необходимую информацию путешественникам, например, телефоны авиакомпаний и вокзалов, проезды в аэропорты. Минусом данного приложения является невозможность добавить маршрут или билет в закладки.

* Аэрофлот

Аэрофлот – официальное мобильное приложение ПАО Аэрофлот, предоставляющее возможность поиска, бронирование и покупки авиабилетов. Также предлагает некоторые дополнительные сервисы, такие как регистрация на рейс, онлайн табло и статус рейса, расписание регулярных рейсов. Однако данное приложение предоставляет информацию только о билетах авиакомпании «Аэрофлот».

* Tickets.ru

Tickets.ru – российский онлайн сервис путешествий, предоставляющий услуги по поиску и покупки билетов регулярных авиакомпаний с наиболее актуальными ценами. Недостатком данного приложения является отсутствие возможности сохранить информацию о маршруте и билете для ускорения дальнейшего использования.

## Анализ задач

### Задача поиска авиабилетов

Рассмотрим задачу поиска авиабилетов. Данная задача включает в себя следующие этапы:

1. Проверка данных, введенных пользователем. Если данные введены не корректно, то пользователю предлагают ввести их повторно.
2. Формирование и отправление запроса с данными на сторонний ресурс (API сервер).
3. Обработка ответа, если таковой получен. Если ответ не получен или произошла какая-либо ошибка, то пользователю выводится сообщение об этом и предлагается ввести данные для поиска повторно.
4. Отображение итоговых результатов поиска пользователю.

### Задача хранения данных пользователя, приложения и его настроек

Рассмотрим задачу хранения данных пользователя, приложения и его настроек. Данную задачу можно подразделить на две подзадачи, а именно:

* хранение закладок,
* хранение истории поиска.

#### Подзадача хранения закладок

Рассмотрим подзадачу хранения закладок. Закладка содержит в себе следующую информацию:

* пункт отправления;
* пункт назначения;
* количество взрослых пассажиров;
* количество детей;
* количество младенцев;
* наличие пересадок;
* тип путешествия (в один конец или с обратной дорогой);
* класс обслуживания (эконом или бизнес);

Для выполнения данной подзадачи следует реализовать следующие возможности пользователя:

1. Сохранение выбранного маршрута в закладки.
2. Просмотр всех закладок.
3. Удаление закладки.
4. Переход с закладки к форме поиска с заполненными полями, соответствующими информации закладки.

#### Подзадача хранения истории поиска

Рассмотрим подзадачу хранения истории поиска. Отдельным элементом истории поиска является совокупность информации, введенной пользователем для поиска рейсов. Это значит, что в данном элементе содержится информация о следующем:

* пункт отправления,
* пункт назначения,
* количество взрослых пассажиров,
* количество детей,
* количество младенцев,
* дата отправления.
* дата отправления из пункта назначения (если билеты в оба конца)
* наличие пересадок
* тип путешествия (в один конец или с обратной дорогой);
* класс обслуживания (эконом или бизнес);

Также данная подзадача включает в себя задачи по реализации возможности пользователя:

* просматривать историю поиска;
* удалять отдельный элемент истории поиска;
* очистить всю историю поиска;
* не сохранять историю поиска;
* переход с элемента истории поиска к форме поиска с заполненными полями, соответствующими информации закладки.

### Задача отображения пользовательского интерфейса

Рассмотрим задачу отображения пользовательского интерфейса. Данная задача включает в себя разработку и оформление следующих интерфейсов и страниц:

* форма для ввода данных для поиска;
* страница, содержащая результаты поиска с возможностями добавить маршрут в закладки и отсортировать результаты;
* страница, содержащая информацию о конкретной единице результата поиска;
* страница, содержащая список закладок пользователя с возможностью удаления закладки и перехода по клику на закладку к форме ввода данных с передачей данных закладки;
* страница, содержащая историю поиска с возможностью ее очищения перехода по клику на элемент истории поиска к форме ввода данных с передачей данных элемента истории поиска (здесь же находиться элемент пользовательского интерфейса, позволяющий не сохранять историю поиска).

## Анализ средств реализации

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

1. Android SDK – средство разработки мобильных приложений для операционной системы Android. Чертой, отличающей от других средств разработки, является наличие широких функциональных возможностей, позволяющих запускать тестирование и отладку исходных кодов, оценивать работу приложения в режиме совместимости с различными версиями ОС Android.
2. В качестве СУБД была выбрана SQLite, так как закладки было решено хранить во внутренней базе данных устройства, работающего на операционной системе Android.

Для реализации отдельных подзадач были выбраны следующие библиотеки и средства:

* Протокол прикладного уровня HTTPS и библиотека Retrofit 2 Android для взаимодействия с внешними API серверами.
* Библиотека Glide для асинхронной загрузки изображений из сети интернет.
* Библиотека Room для взаимодействие с базой данных.

## Диаграммы

Для описания работы системы были составлены следующие диаграммы:

* Диаграмма активности изображена на рисунке 2.
* Диаграмма коммуникаций изображена на рисунке 3.
* Диаграмма последовательностей изображена на рисунке 4.
* Диаграмма состояний изображена на рисунке 5.
* Диаграммы классов изображены на рисунках 6-11.
* Диаграмма объектов изображена на рисунке 12.
* Диаграмма развертывания изображена на рисунке 13.

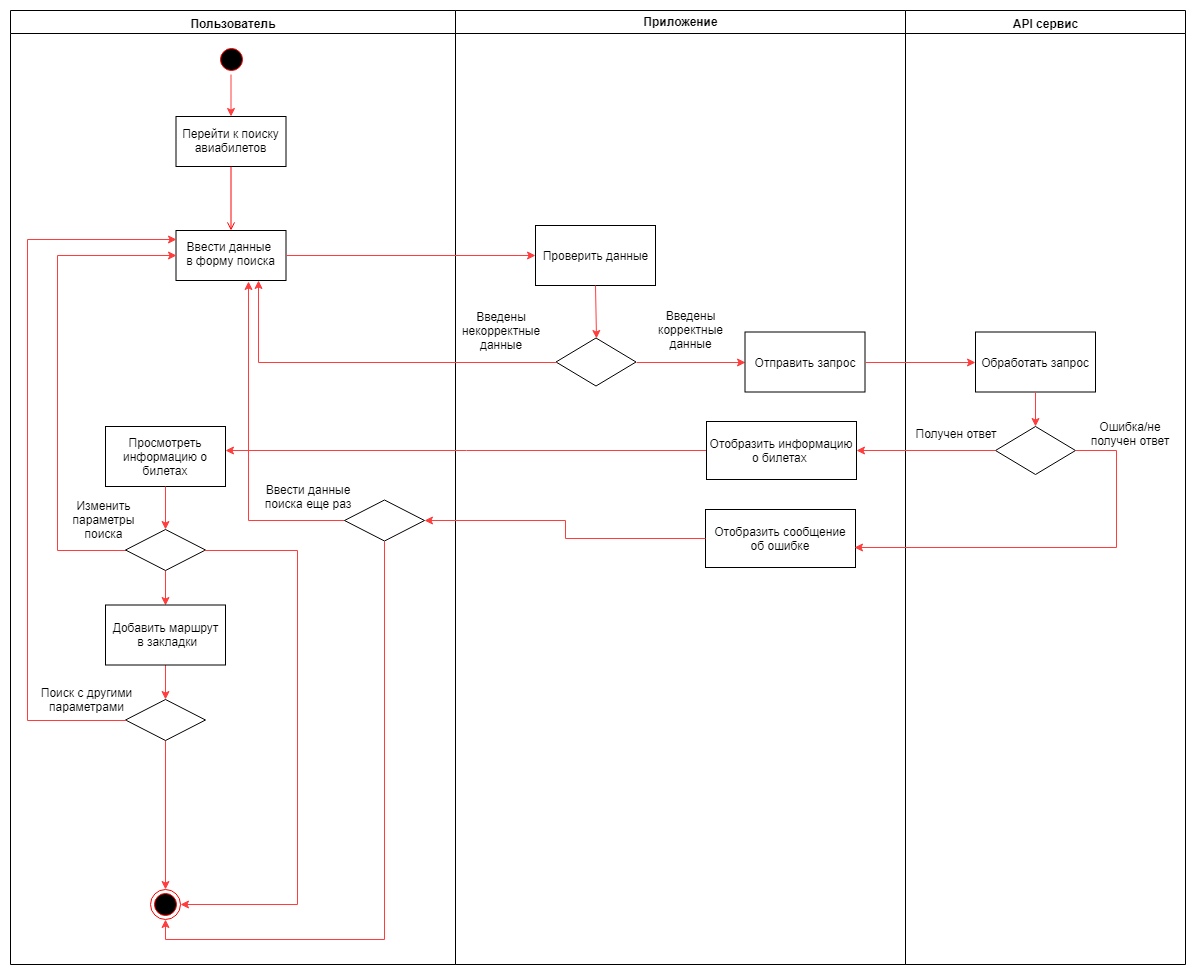


Рисунок 2. Диаграмма активности.

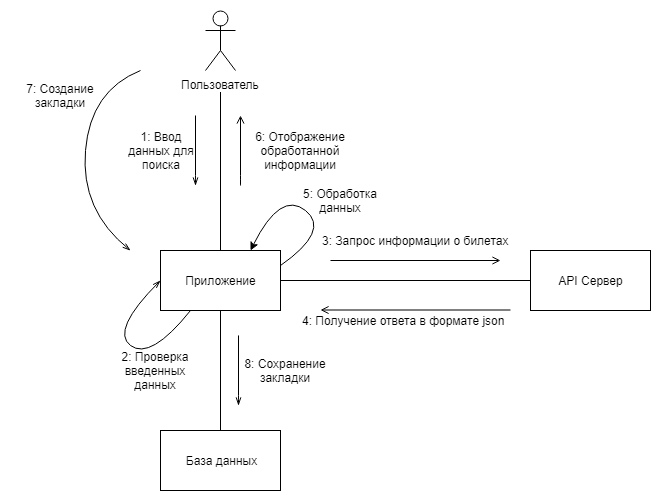


Рисунок 3. Диаграмма коммуникаций.

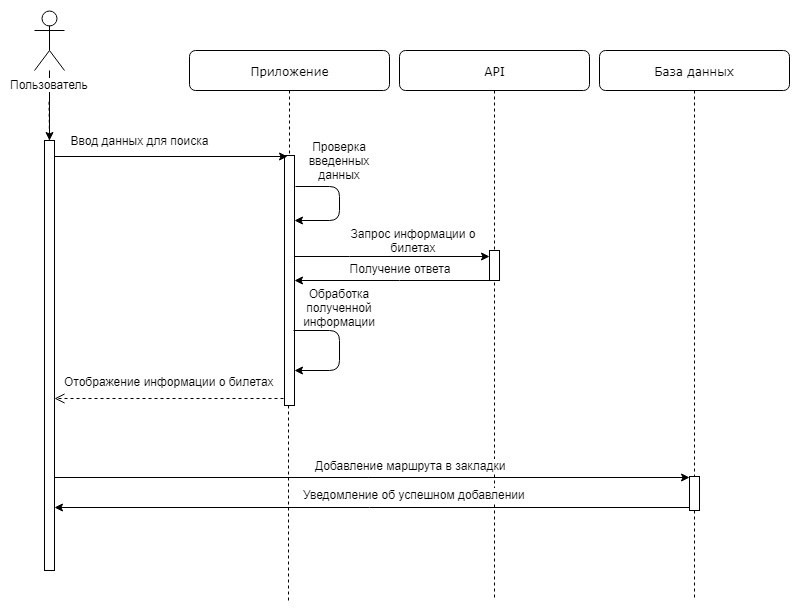


Рисунок 4. Диаграмма поседовательностей.(запихнуть в интерфейс)

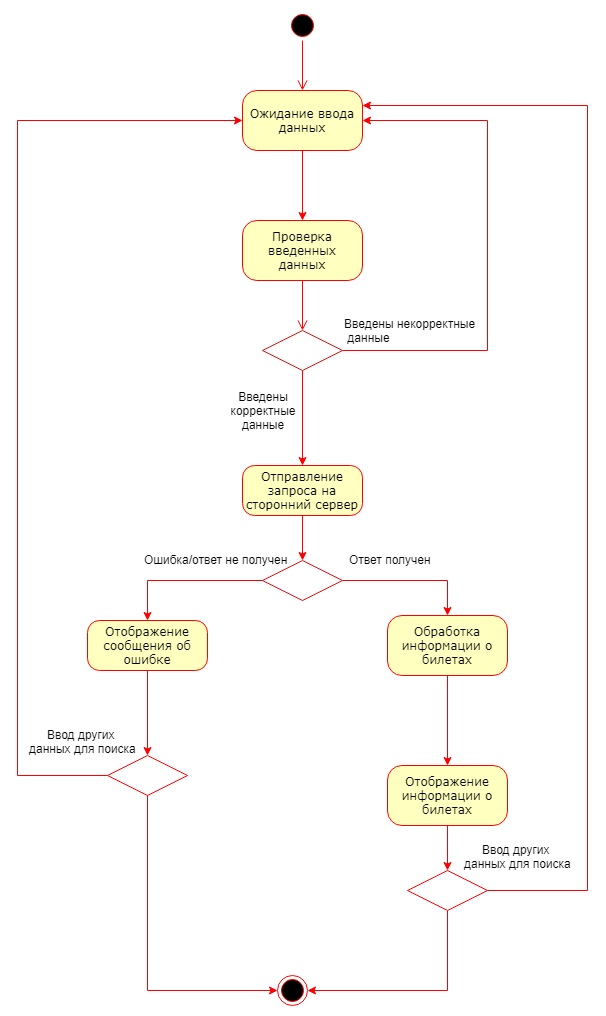


Рисунок 5. Диаграмма состояний.

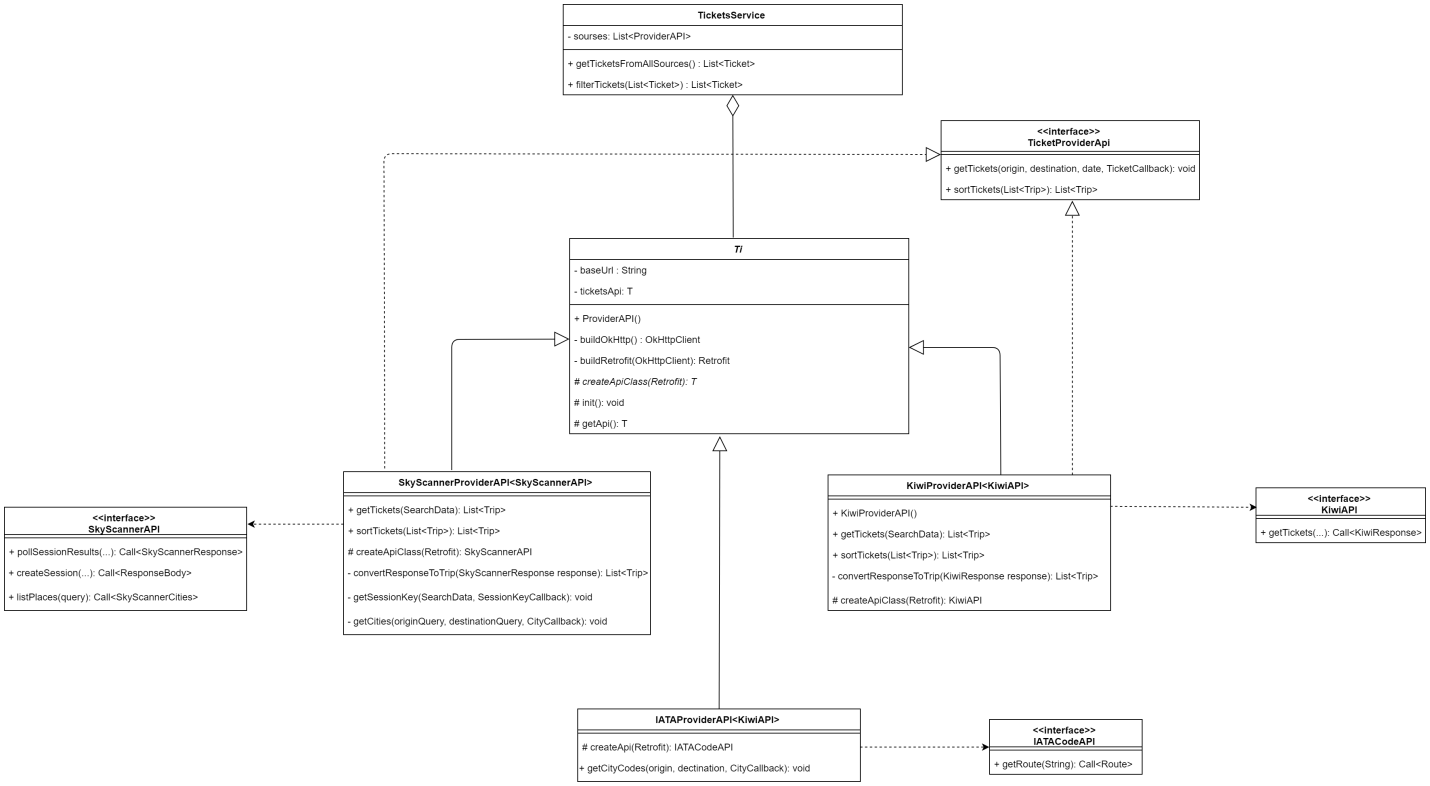


Рисунок 6. Диаграмма классов модуль поиска авиабилетов.

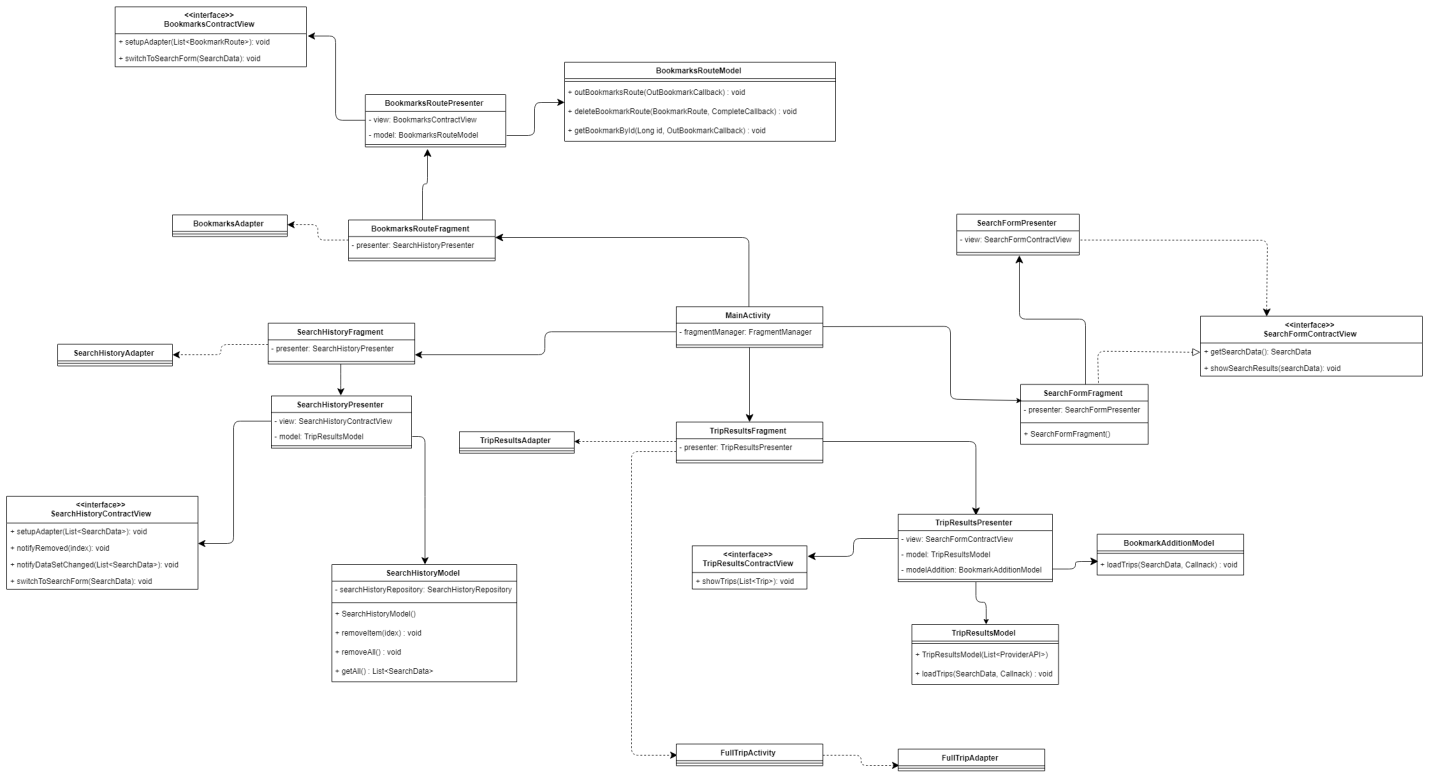


Рисунок 8. Диаграмма классов пользовательского интерфейса.

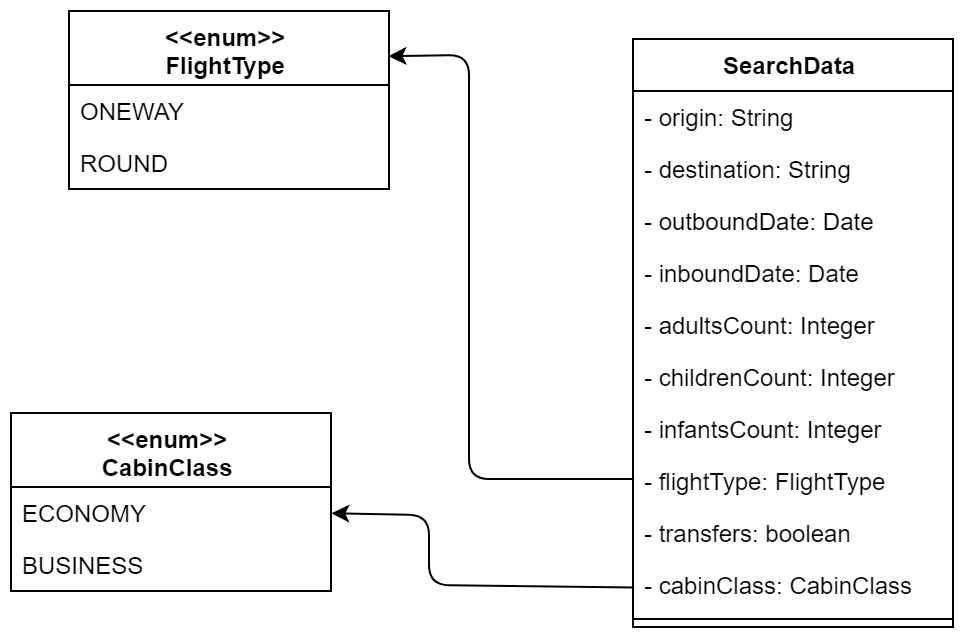


Рисунок 9. Диаграмма классов чтения из пользовательского интерфейса.

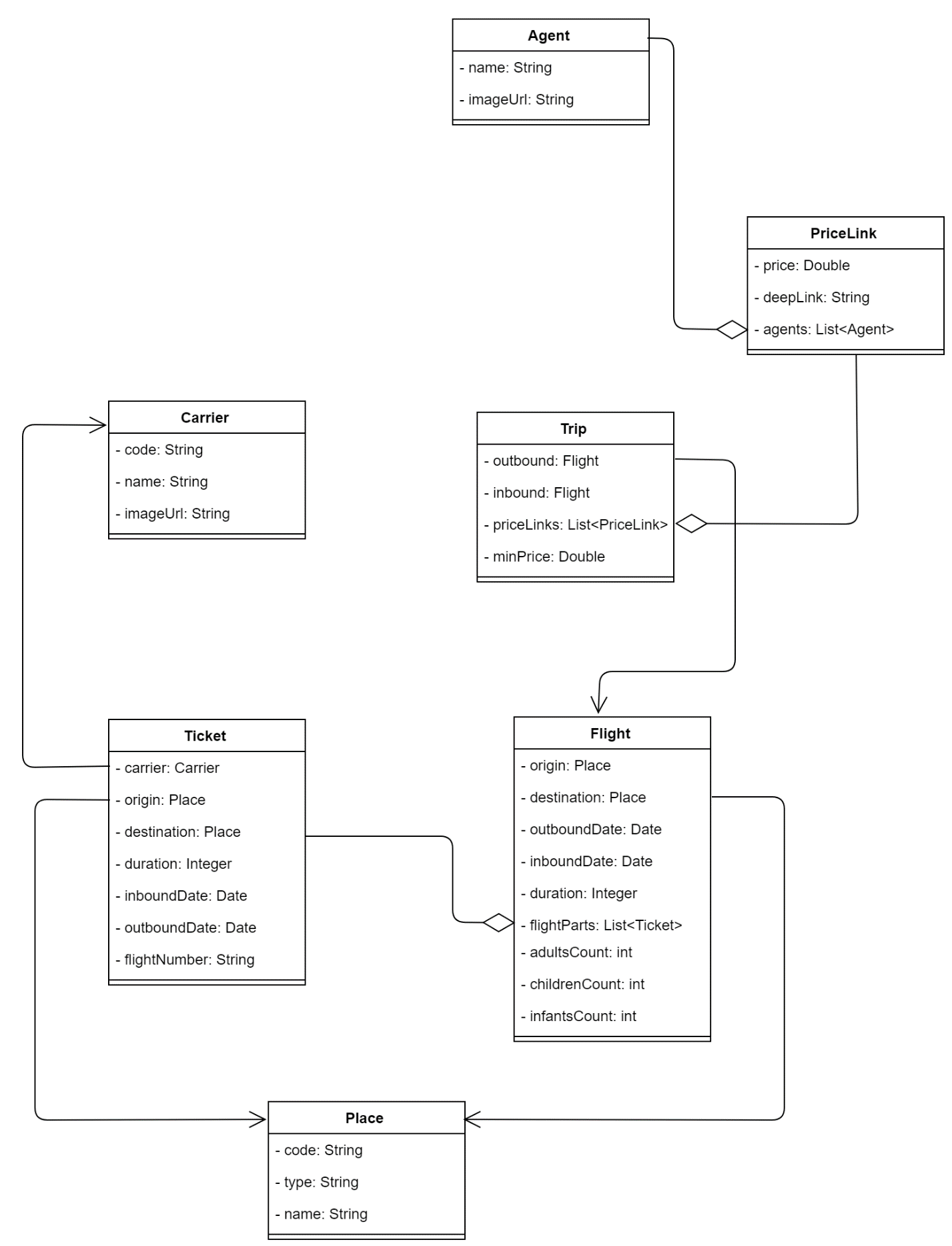


Рисунок 10. Диаграмма классов полученных билетов.

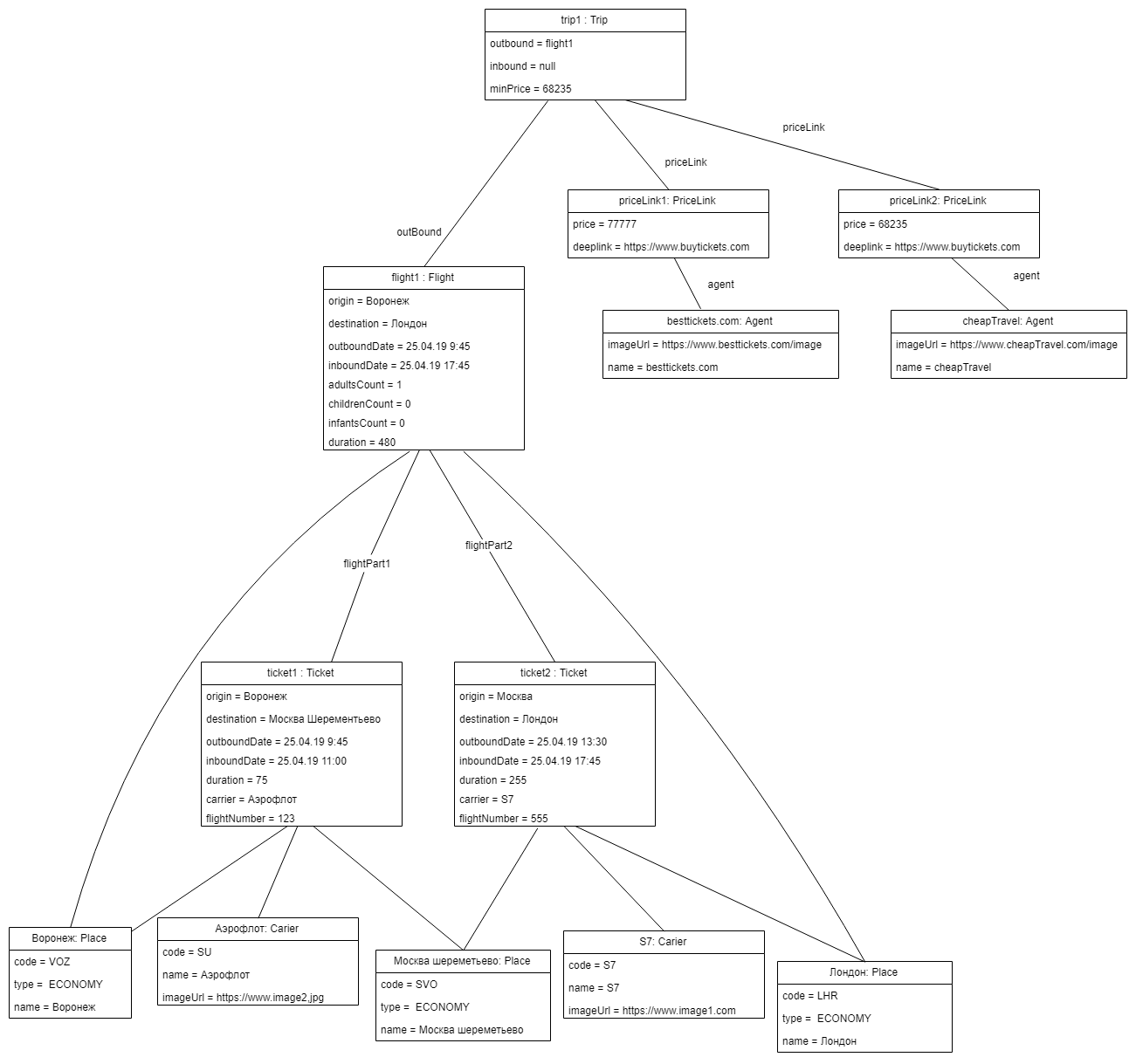


Рисунок 12. Диаграмма объектов полученных билетов.

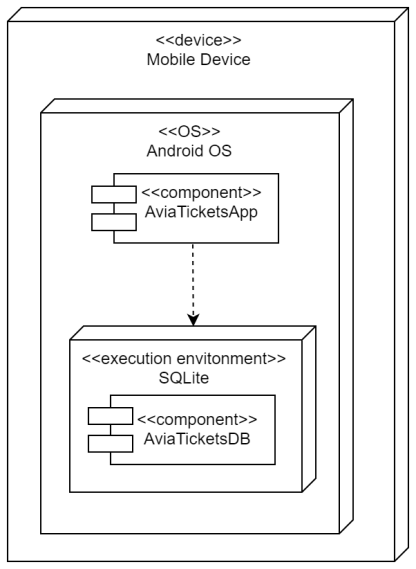


Рисунок 13. Диаграмма развертывания.

# Реализация

В данной главе описана реализация всех подсистем приложения по поиску авиабилетов, задачи которых описаны в главе «Анализ» в разделе «Анализ задач».

## Задача поиска авиабилетов

## Задача хранения данных пользователя, приложения и его настроек

### Подзадача хранения закладок

Для хранения закладок пользователя выбрана база данных SQLite. Все закладки хранятся в таблице, которая имеет структуру, изображенную на рисунке \*.

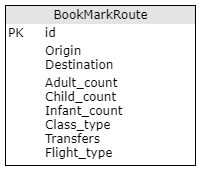


Рисунок \*. Таблица закладок.

Для связи с базой данных используется библиотека Room, для чего была использована организация классов, изображенная рисунке \*.

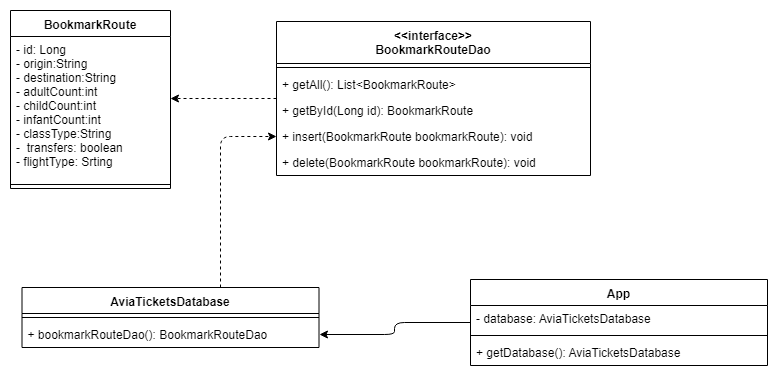


Рисунок \*. Диаграмма классов общения с базой данных.

Класс BookmarkRoute необходим для проекции строки базы данных.

Класс AviaTicketsDatabase является наследником класс RoomDatabase и необходим для работы с BookmarkRouteDao.

Интерфейс BookmarkRouteDao необходим для манипулирования данными, а именно:

* метод getAll() возвращает все закладки, содержащиеся в базе данных на момент вызова;
* метод getById(Long id) возвращает закладку, id которой равен полученному;
* метод insert (BookmarkRoute bookmarkRoute) добавляет в базу данных полученную закладку;
* метод delete (BookmarkRoute bookmarkRoute) удаляет из базы данных полученную закладку.

Класс App необходим для реализации паттерна одиночка, которая обеспечивает существование только одного экземпляра класса AviaTicketsDatabase на все приложение.

### Подзадача хранения истории поиска

Для хранения истории поиска и пользовательских настроек, позволяющих не сохранять историю поиска, было использовано постоянное хранилище данных Android SharedPreferences.

Для реализации данной подзадачи создан класс SearchHistoryRepository, изображенные на рисунке \*. В данном классе находятся строковые константы:

* PREFS\_NAME – имя, по которому можно обратиться к части памяти, в которой хранится история поиска;
* SEARCH\_DATA\_KEY – ???

Также данный класс содержит методы, обеспечивающие функциональность, реализуемую в данной подзадаче:

* метод getAllSearchData() – возвращает все элементы, хранящиеся на момент вызова в списке элементов истории поиска;
* метод addSearchData(SerachData) – добавляет в список элементов истории поиска полученный элемент;
* метод RemoveSearchData(index) – удаляет из списка элементов истории поиска элемент по его индексу;
* метод RemoveAll() – очищает историю поиска;
* метод getSharedPreferences() – возвращает все элементы истории поиска, хранящиеся на момент вызова в SharedPreferences
* метод putToSharedPrefeernces() – добавляет в SharedPreferences, список элементов истории поиска;
* метод … - сохраняет настройки приложения (хранить историю или нет).

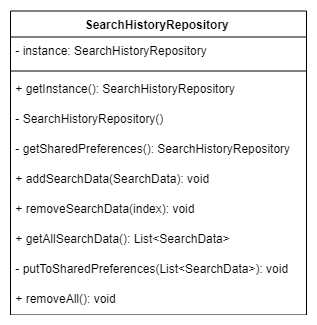


Рисунок \*. Диаграмма классов хранения истории.

## Задача отображения пользовательского интерфейса

Для отображения всех страниц, представляющих собой пользовательский интерфейс, была использована архитектура MVP, позволяющая отделить отображение интерфейса и логику работы с ним.

Полностью диаграмму классов, реализующую данную задачу, можно увидеть в Приложении %.

### Форма для ввода данных для поиска

### Результаты поиска

### История поиска

Для отображения страницы с историей поиска и обеспечения ее работы была использована архитектура, изображенная в виде диаграммы классов на рисунке \*.

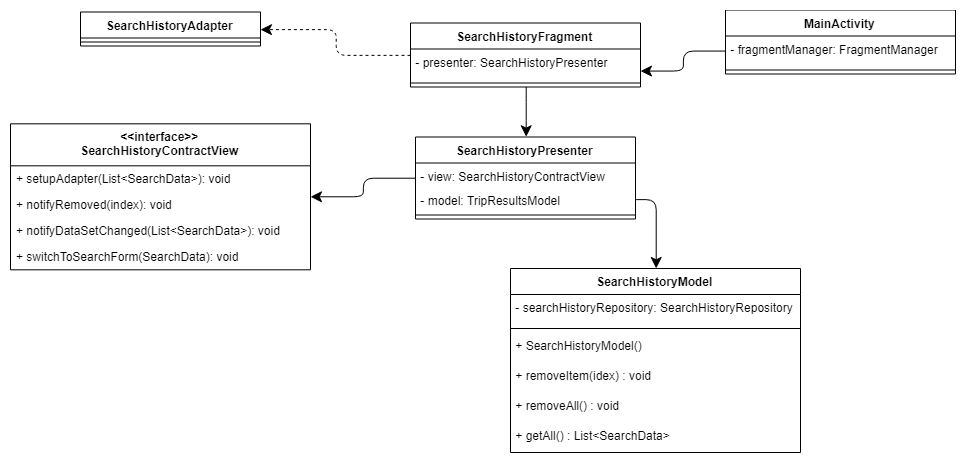


Рисунок \* - Диаграмма классов отображения истории поиска

Для введения архитектуры MVP введены классы SearchHistoryModel, SearchHistoryPresenter, SearchHistoryFragment, а также интерфейс SearchHistoryContractView, который обеспечивает связь между двумя последними.

Класс SearchHistoryModel содержит в себе поля и методы, осуществляющие вызов методов работы с SharedPreferences, а именно:

* поле SearchHistoryRepository searchHistoryRepository ???
* метод removeItem(int index) удаляет элемент истории поиска по его индексу;
* метод removeAll() удаляет все элементов истории поиска, другими словами очищаее историю поиска;
* метод List<SearchData> getAll() возвращает весь список истории поиска.

История поиска отображается в виде списка элементов, реализованного в виде RecyclerView, для чего необходим класс SearchHistoryAdapter, имеющий следующие поля и методы:

* поле LayoutInflater inflater; ???
* поле List<SearchData> searchDataList; ???
* поле SearchHistoryPresenter presenter;???
* поле SearchHistoryViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i)???
* метод onBindViewHolder(@NonNull SearchHistoryViewHolder
* searchHistoryViewHolder, int index)???
* метод getItemCount()???
* метод setPresenter(SearchHistoryPresenter presenter) ???
* метод setSearchDataList(List<SearchData> searchDataList)???

Класс SearchHistoryPresenter необходим для ???, содержит методы и поля:

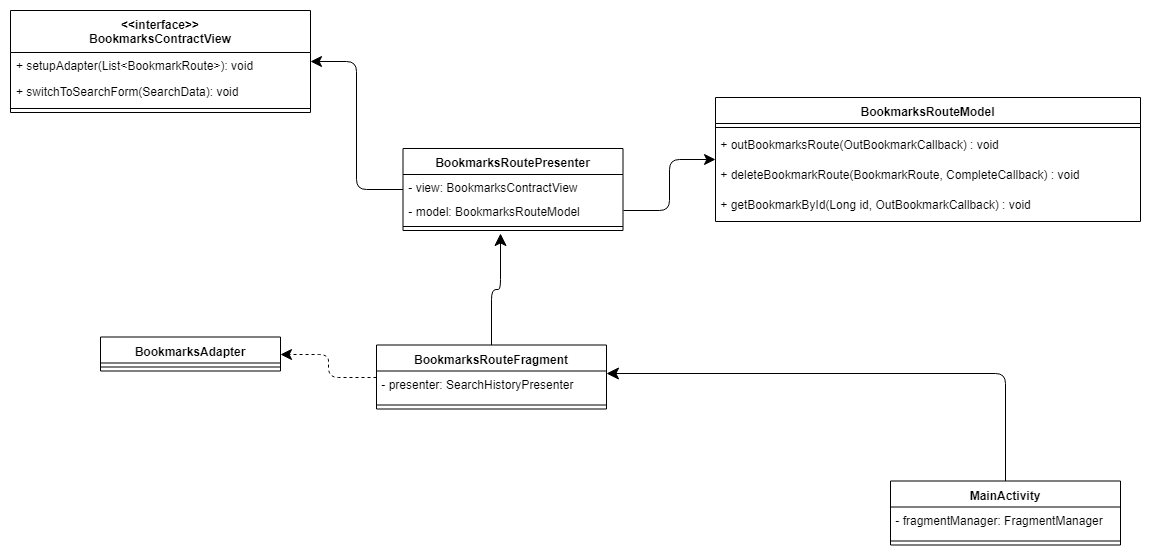
* поле SearchHistoryContractView view;???
* поле SearchHistoryModel model; ???
* метод attachView(SearchHistoryContractView searchHistoryContractView) ???
* метод viewIsReady() ???
* метод removeItem(int index) удаляет элемент истории поиска по индексу;
* метод itemChosen(SearchData searchData)
* метод clearHistory() очищает историю поиска.

Класс SearchHistoryFragment необходим для отображения пользовательского интерфейса, здесь находятся поля и методы:

* поле RecyclerView recyclerView;
* поле Button clearButton;
* поле SearchHistoryPresenter presenter;
* поле SearchHistoryAdapter adapter;
* поле TextView noSearchHistoryTextView;
* метод View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
* метод setupAdapter(List<SearchData> searchDataList)
* метод notifyRemoved(int index)
* метод notifyDataSetChanged(List<SearchData> searchDataList)
* метод switchToSearchForm(SearchData searchData)
* метод showEmptyMessage()
* метод hideEmptyMessage()

### Закладки

Для отображения страницы с закладками и обеспечения ее работы была использована архитектура, изображенная в виде диаграммы классов на рисунке \*.



Рисунок\* - Диаграмма классов отображения страницы с закладками

В классе BookmarksRouteModel содержаться методы, осуществляющие работу с базой данных, а именно:

* + - outBookmarksRoute(OutBookmarkCallback callback) – метод, отвечающий за вывод всех закладок, содержащихся в базе данных на момент вызова;
    - deleteBookmarkRoute(BookmarkRoute value, CompleteCallback callback) – метод, отвечающий за удаление закладки.

Здесь OutBookmarkCallback и CompleteCallback – это интерфейсы, обеспечивающие взаимосвязь с асинхронными запросами.

По причине того, что взаимодействие с базой данных – это тяжеловесный процесс, который может сильно нагружать приложение и замедлять отображение пользовательского интерфейса, вводятся классы BookmarksRouteOut, BookmarksRouteDelete и BookmarksRouteGetById, которые являются наследниками базового класса AsyncTask. Таким образом, запросы к базе данных осуществляются в отдельном асинхронном потоке.

Закладки отображаются в виде списка элементов, реализованного в виде RecyclerView, для чего необходим класс BookMarkAdapter, имеющий поля и методы, аналогичные полям и методам класса SearchHistoryAdapter, описанном в пункте 3.3.4.

С целью введения архитектуры MVP введены классы BookmarksRouteFragment, BookmarksRoutePresenter и интерфейс BookmarksContractView, обеспечивающий связь BookmarksRouteFragment и BookmarksRoutePresenter.

Класс BookmarksRouteFragment необходим для отображения пользовательского интерфейса, здесь находятся методы:

* + - метод onCreate(Bundle savedInstanceState)
    - метод View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
    - метод setAdapter(List<BookmarkRoute> bookmarkRoutes) назанчает адаптер для RecyclerView;
    - метод switchToSearchForm(SearchData searchData) осуществляет переход к форме поиска с заполнением полей информацией, сохраненной в закладке.

Для вызова методов связи с базой данных использован класс BookmarksRoutePresenter, содержащий в себе следующие поля и методы:

* статичное поле List<BookmarkRoute> bookmarkRoutes – список закладок;
* метод attachView(BookmarksRouteFragment activity)
* метод detachView()
* метод viewIsReady()
* метод loadBookmarks() выводит все закладки;
* метод delete(int index) удаляет закладку по ее индексу в списке всех закладок;
* метод itemChosen(SearchData searchData)

# Интерфейс

# , по клику на который можно перейти на форму поиска, при этом поля формы будут заполнены информацией, хранящейся в закладке. Данный список

# Заключение

# Список источников

# Приложения