МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет компьютерных наук

Кафедра информационных технологий в управлении предприятием

Курсовая работа по курсу

«Технологии программирования»

**«Система для подбора авиабилетов»**

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_В.А. Парфенов, 3 курс, д/о

Обучающийся\_\_\_\_\_\_\_\_\_\_\_\_ А.И Некрут, 3 курс, д/о

Руководитель \_\_\_\_\_\_\_\_\_\_\_ Х.А Полещук

Воронеж 2019

**Оглавление**

[1 Введение 4](#_Toc10764603)

[2 Постановка задачи 4](#_Toc10764604)

[3 Анализ 5](#_Toc10764605)

[3.1 Анализ предметной области 5](#_Toc10764606)

[3.2 Рассмотрение существующих сервисов аналогов 5](#_Toc10764607)

[3.3 Анализ процессов 7](#_Toc10764608)

[3.3.1 Процесс поиска авиабилетов 8](#_Toc10764609)

[3.3.2 Процесс авторизации 9](#_Toc10764610)

[3.3.3 Процесс регистрации 10](#_Toc10764611)

[3.4 Анализ подзадач 12](#_Toc10764612)

[3.4.1 Подзадача “Закладки” 12](#_Toc10764613)

[3.4.2 Подзадача “История” 12](#_Toc10764614)

[3.4.3 Подзадача хранения данных 12](#_Toc10764615)

[3.5 Графическое описание работы веб-сервиса 14](#_Toc10764616)

[3.5.1 Диаграмма состояний 14](#_Toc10764617)

[3.5.2 Диаграмма активности 15](#_Toc10764618)

[3.6 Анализ средств реализации 17](#_Toc10764619)

[3.6.1 Клиентская часть 17](#_Toc10764620)

[3.6.2 Серверный уровень 17](#_Toc10764621)

[3.6.3 Интеграция сторонних источников: 18](#_Toc10764622)

[4 Реализация 19](#_Toc10764623)

[4.1 Диаграмма развертывания 19](#_Toc10764624)

[4.2 Архитектура приложения 20](#_Toc10764625)

[4.3 База данных 21](#_Toc10764626)

[4.4 Нереализованные функции 21](#_Toc10764627)

[5 Интерфейс 21](#_Toc10764628)

[6 План тестирования 27](#_Toc10764629)

[7 Заключение 28](#_Toc10764630)

[8 Список источников 29](#_Toc10764631)

# Введение

По статистике гражданской авиации России на сегодняшний день авиатранспортом в личных целях пользуются только около 8% россиян. Это очень небольшая цифра, если учитывать тот факт, что население становится мобильнее. Люди все чаще отправляются в командировки используя авиатранспорт, увеличивается количество предпринимателей, которым необходимо к короткий срок переместиться из точки А в точку Б. Также многие люди навещают далеких друзей и родных. С ростом числа пользователей услугами авиатранспорта возрастет и спрос на программные средства, задействованные в этой области.

Всем этим людям необходимо получать правильную и актуальную информацию об авиабилетах. Сегодня инструментом получения таких данных являются не только аэропорты и горячие линии, но и веб-сервисы и приложения, так как почти у каждого человека есть постоянный доступ в сеть интернет.

Сервисы предоставляют различную информацию для пользователей, начиная с времени отправления и прибытия транспорта и заканчивая возможностью бронирования билетов. Люди, пользующиеся их услугами (далее – пользователи), желают быстро и без усилий получать актуальную и проверенную информацию.

В данной работе рассматривается проблема создания веб-сервиса, предоставляющего пользователям возможность получать информацию об авиабилетах – их ценах, продолжительности полета, а также об авиамаршрутах с пересадками.

# Постановка задачи

Целью работы является создание веб-сервиса предоставляющего информацию об авиабилетах. Для достижения поставленной цели необходимо решить следующие задачи:

* анализ задачи: сбор материалов для анализа, анализ данных, рассмотрение существующих сервисов и их функциональности;
* определение требований к сервису, исходя из анализа задачи;
* выбор технологий для разработки сервиса (для клиентской и серверной частей), обусловленный требованиями к приложению и будущими особенностями;
* разработка сервиса: построение архитектуры приложения, проектирование структуры базы данных, интеграция сторонних источников данных и т.д.;
* тестирование сервиса и устранение ошибок.

Требования к системе:

1. Все функции, представляемые приложением отображены на use-case диаграмме.
2. Требования, указанные в техническом задании
3. Сервис отображать маршруты с пересадками

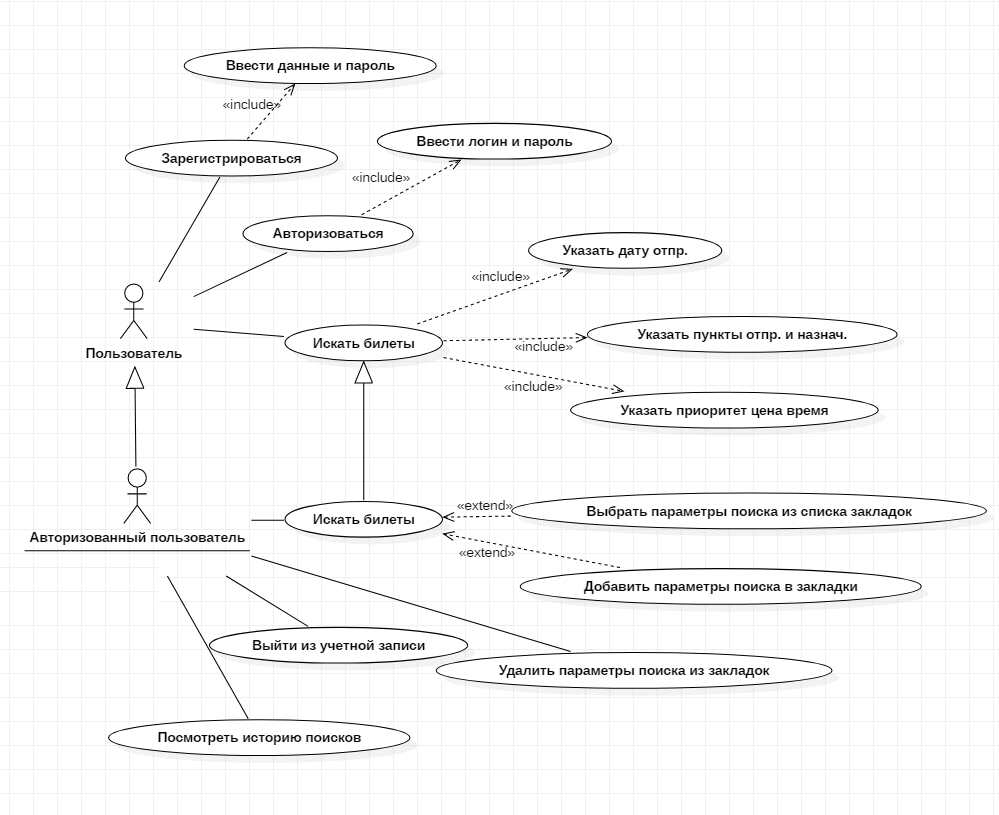


Рисунок 1 use-case диаграмма

Варианты использования:

1. Регистрация – незарегистрированный пользователь заполняет свои данные для внесения в систему.

2. Авторизация – пользователь вводит логин и пароль и, если такая пара значений имеется в базе данных, получает доступы к расширенным возможностям сайта.

3. Выход из учетной записи.

4. Поиск билетов – пользователь вводит информация о желаемых билетах, авторизованный пользователь может искать по сохраненному шаблону(“Закладки”).

5. Добавление параметров поиска в закладки – авторизованный пользователь сохраняет шаблон поиска в закладки.

6. Удалить параметры поиска из закладок.

7. Просмотреть историю поисков – авторизованный пользователь переходит на страницу, содержащую историю запросов.

Система должна соответствовать следующим требованиям:

1. Проект должен иметь возможность добавлять новый функционал с минимальным изменением существующего кода.
2. Реализовывать все функции, описанные выше.

# Анализ

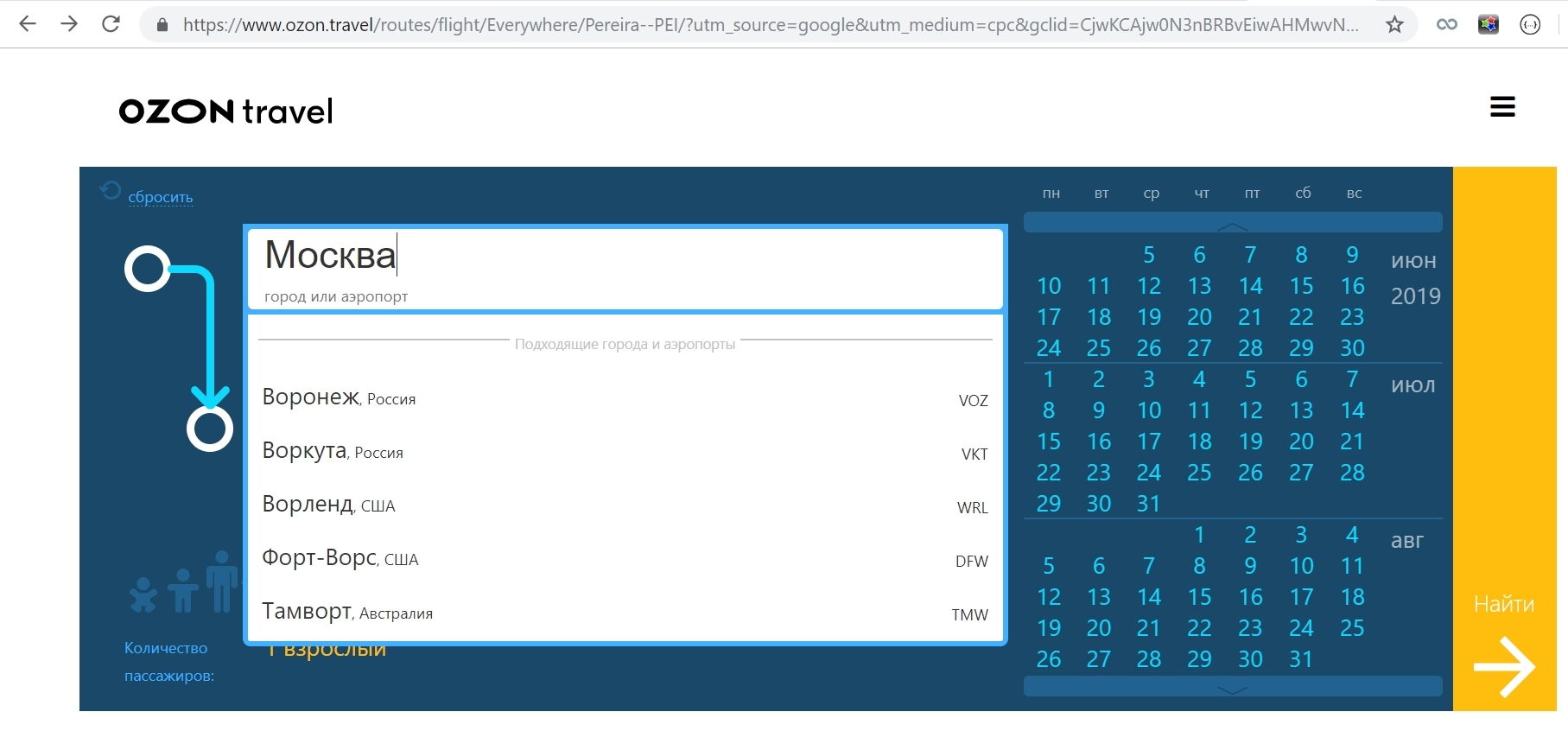
## Анализ предметной области

## Рассмотрение существующих сервисов аналогов

В сети уже существует множество веб-сервисов, предоставляющих возможности по получению информации об авиабилетах. Рассмотрим несколько самых популярных сайтов.

1. OZON.travel — интернет-сервис для бронирования авиа и железнодорожных билетов, а также организации деловых поездок. Позволяет приобрести авиабилеты всех российских и более 500 международных авиакомпаний.

Рисунок 2 пользовательский интерфейс сайта ozon.travel.ru



При первом обращении к сервису выдал ошибочное автозаполнение поля, очевидно, имеются ошибки в скриптах веб-страницы. Сервис имеет загроможденный календарем интерфейс и не предоставляет функций добавления параметров поиска в закладки и просмотра истории

1. Aviasales.ru — российский метапоисковик авиабилетов, принадлежащий компании Go Travel Un Limited. Ежемесячная аудитория сервиса оценивается в около 7 миллионов человек.

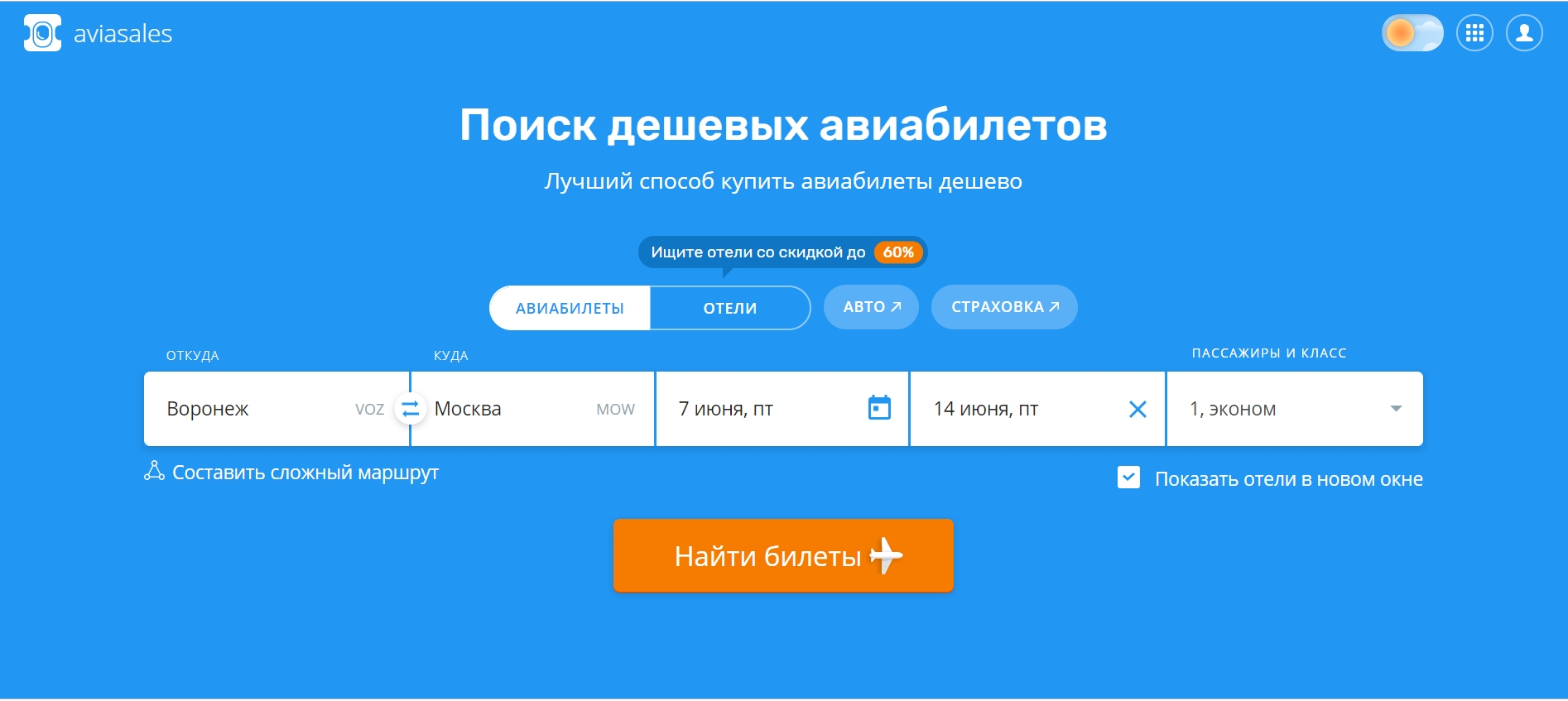


Рисунок 3 пользовательский интерфейс сайта aviasales.ru

После регистрации на сайте, появляется возможность просматривать историю поисков, однако функция добавления запроса в закладки для дальнейшего быстрого использования шаблона поиска отсутствует. Кроме того, после нажатия на кнопку “Найти билеты”, первые результаты отображаются лишь через 10 секунд ожидания.

1. Avia.tutu.ru – российский онлайн сервис путешествий, предоставляющий услуги по поиску и покупки билетов авиабилетов. Также предоставляет необходимую информацию путешественникам.

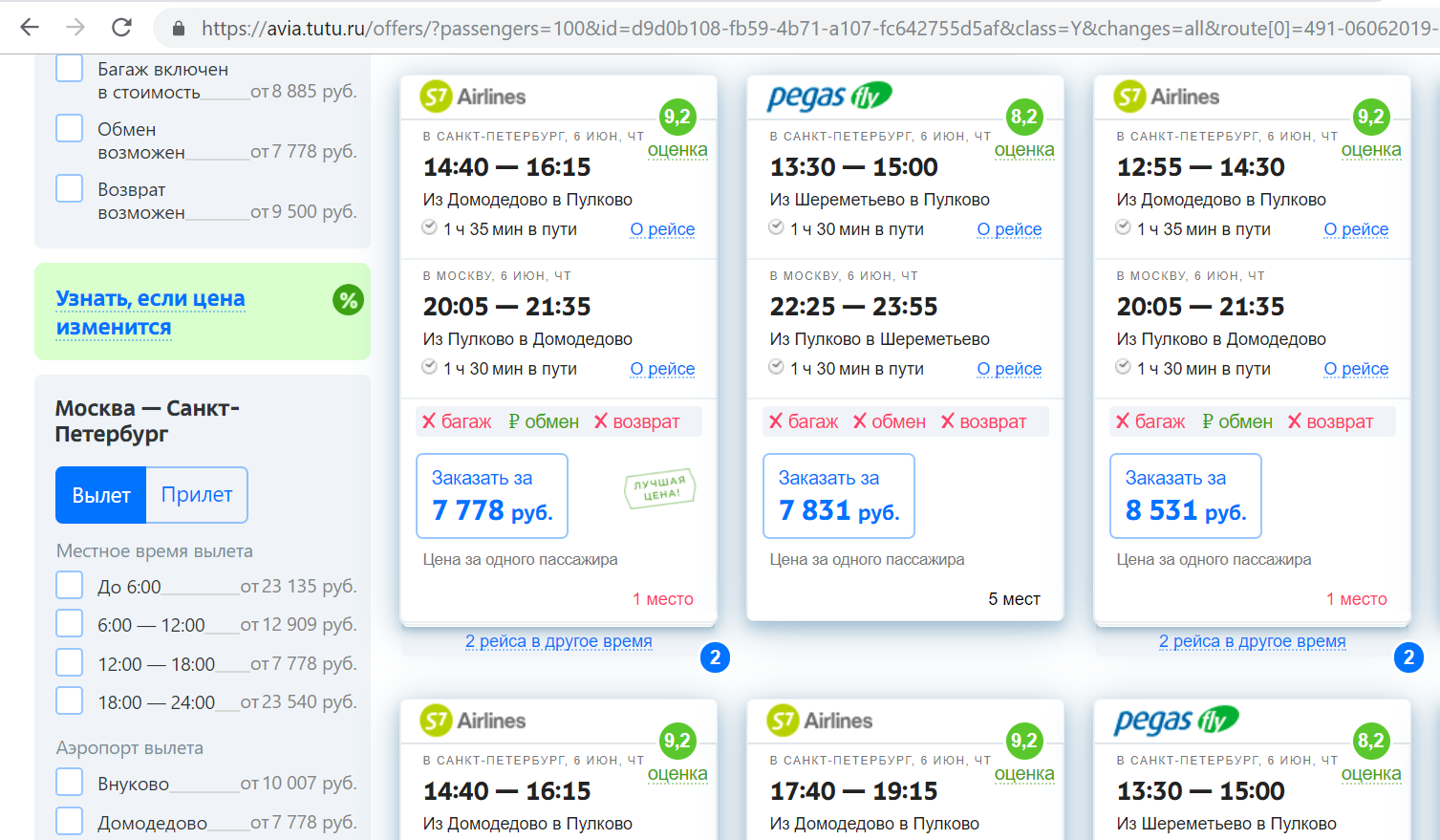


Рисунок 4 пользовательский интерфейс сайта avia.tutu.ru c выведенной информацией о билетах

Имеет загроможденный вывод, затрудняющий быстрое восприятие информации. Не имеет функций добавления поиска в закладки или просмотра истории. После нескольких поисковых запросов было подсчитано, что среднее время ожидания результата составляет 9,5 секунд, это, безусловно, также можно отнести к минусам.

Как выяснилось, ни один из аналогов не предоставляет пользователю возможность добавить параметры поиска в закладки. Некоторые достаточно долго обрабатывают поисковой запрос или имеют визуальные избытки в интерфейсе.

Разрабатываемое приложение должно предоставлять пользователям функцию “добавление в закладки”, отсутствующую в данных сервисах, а также обладать удобным интерфейсом и хорошей скоростью работы.

## Анализ процессов

Для описания процессов, протекающих в веб-приложении, было решено расписать их пошагово и построить диаграммы последовательности и коммуникации при помощи языка графического описания для объектного моделирования UML.

### Процесс поиска авиабилетов

Поиск авиабилетов состоит из следующий шагов:

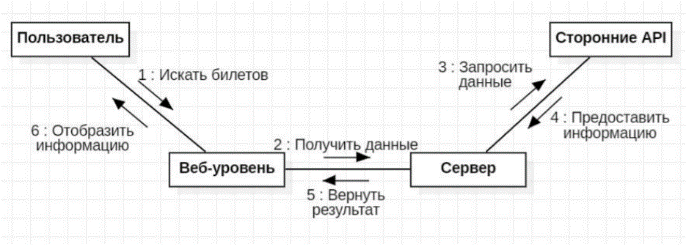
1. Пользователь выбирает пункты отправления и назначения, используя выпадающие списки.
2. Пользователь выбирает дату отправления.
3. Пользователь указывает приоритет поиска – цена/время полета (первые три шага не обязательно должны быть расположены в таком порядке)
4. Пользователь нажимает на кнопку “Подобрать”
5. Происходит проверка правильности введенных данных
6. Если данные введены корректно, отправляется запрос на сервер, содержащий параметры поиска
7. Сервер подготавливает и отправляет запрос к сторонним API
8. Сервер обрабатывает полученные данные и отправляет клиенту ответ, в теле которого содержится информация о билетах

Рисунок 5 диаграмма коммуникаций “Поиск авиабилетов”

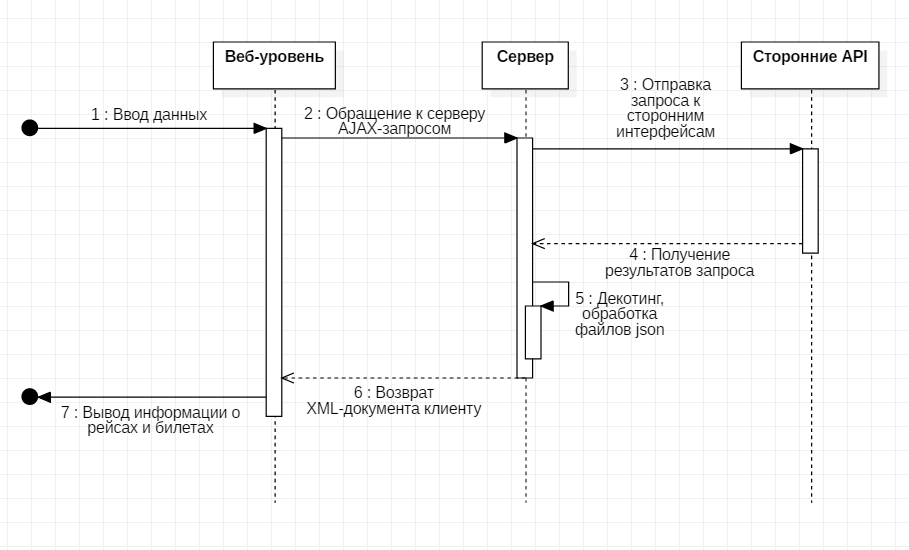


Рисунок 6 диаграмма последовательности “Поиск авиабилетов”

### Процесс авторизации

Процесс авторизации состоит из следующих шагов:

1. Пользователь вводит данные на странице авторизации
2. Данные передаются на сервер
3. Сервер отправляет к СУБД сообщение, содержащее SELECT запрос
4. Если сервер в ответе от СУБД получает данные, соответствующие указанным пользователям, происходит подтверждение регистрации, в противном случае отказ
5. Сервер отправляет клиенту результат авторизации
6. Результат отображается на странице

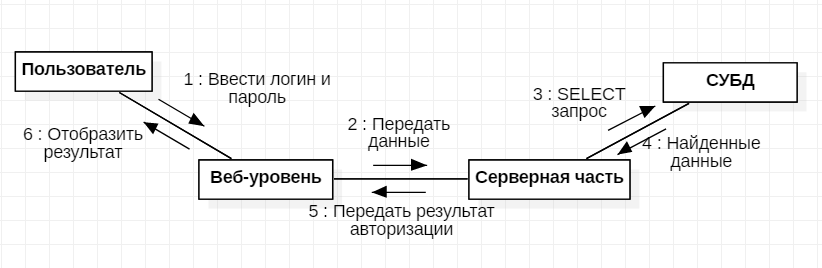
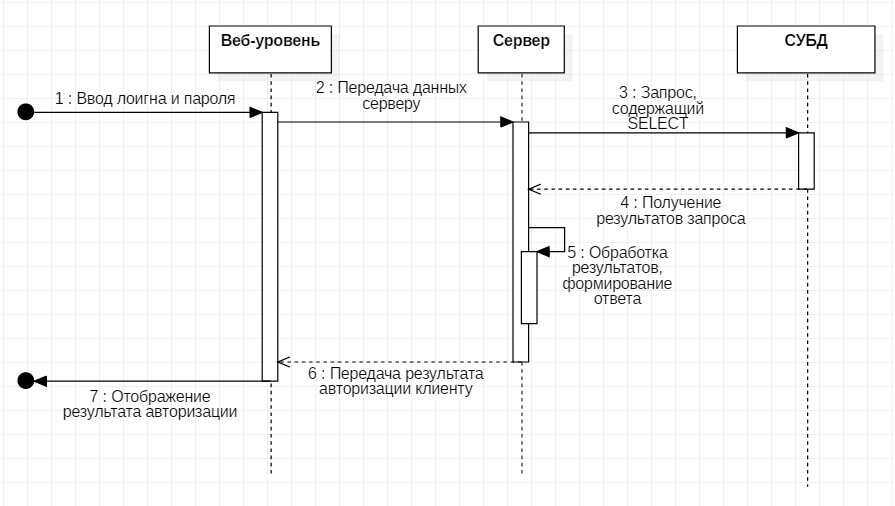


Рисунок 8 диаграмма последовательности “Авторизация”

Рисунок 7 диаграмма коммуникаций “Авторизация”

### Процесс регистрации

Процесс регистрации включает следующие шаги:

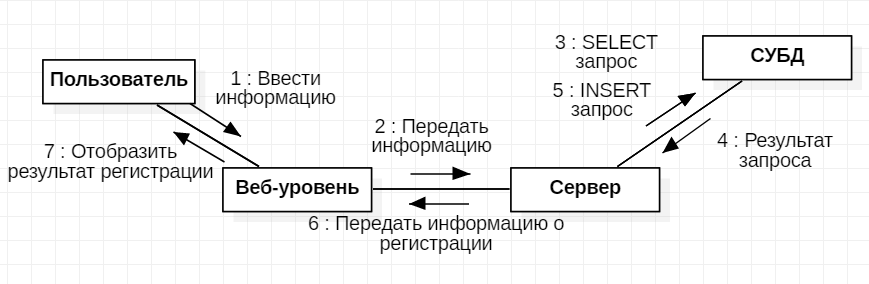
1. Пользователь вводит данные на странице регистрации
2. Данные передаются на сервер
3. Сервер отправляет к СУБД сообщение, содержащее SELECT запрос
4. Если ответ от СУБД, полученный сервером, не содержит данных, соответствующих введенным пользователем, сервер отправляет сообщение к СУБД, содержащее INSERT запрос
5. Сервер отправляет результат авторизации клиенту
6. Клиент отображает результат

Рисунок 9 диаграмма коммуникаций “Регистрация”

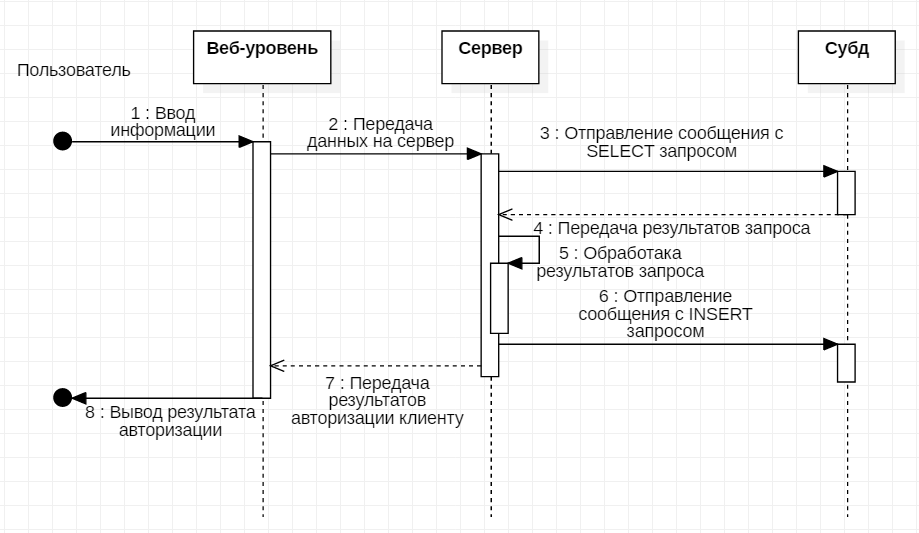


Рисунок 10 диаграмма последовательности “Регистрация”

## Анализ подзадач

### Подзадача “Закладки”

Данная подзадача рассматривает реализацию следующих функций, предоставляемых авторизованному пользователю:

1. Добавление указанных параметров поиска в закладки, для дальнейшего возможного использования в качестве шаблона поиска
2. Удаление закладки
3. Осуществление поиска авиабилетов, согласно шаблону выбранной закладки.

### Подзадача “История”

Подзадача рассматривает реализацию функции просмотра истории авторизованным пользователем.

### Подзадача хранения данных

После рассмотрения подзадач “Закладки” и “История” становится очевидной необходимость создания базы данных, которая должна содержать в себе следующую информацию

Для пользователя:

* Персональный идентификатор
* Пароль
* Имя

Для закладок:

* Пункт отправления
* Пункт назначения
* Дата добавления закладки
* Приоритет поиска

Для истории:

* Дата поиска
* Пункт отправления
* Пункт назначения
* Дата рейса

Для визуализации подзадачи хранения данных воспользуемся ER-диаграммой.

Модель «Сущность - Отношение» используется для проектирования баз данных с целью удаления всякой избыточности данных.

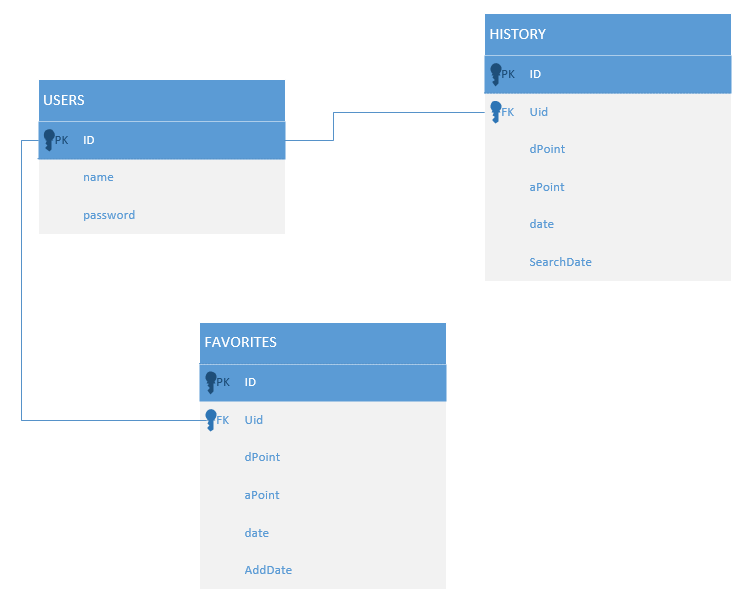


Рисунок 11 ER-диаграмма базы данных

## Графическое описание работы веб-сервиса

Для удобства рассмотрения работы сервиса было решено использовать диаграммы состояний и активности нотаций графической нотации UML.

### Диаграмма состояний

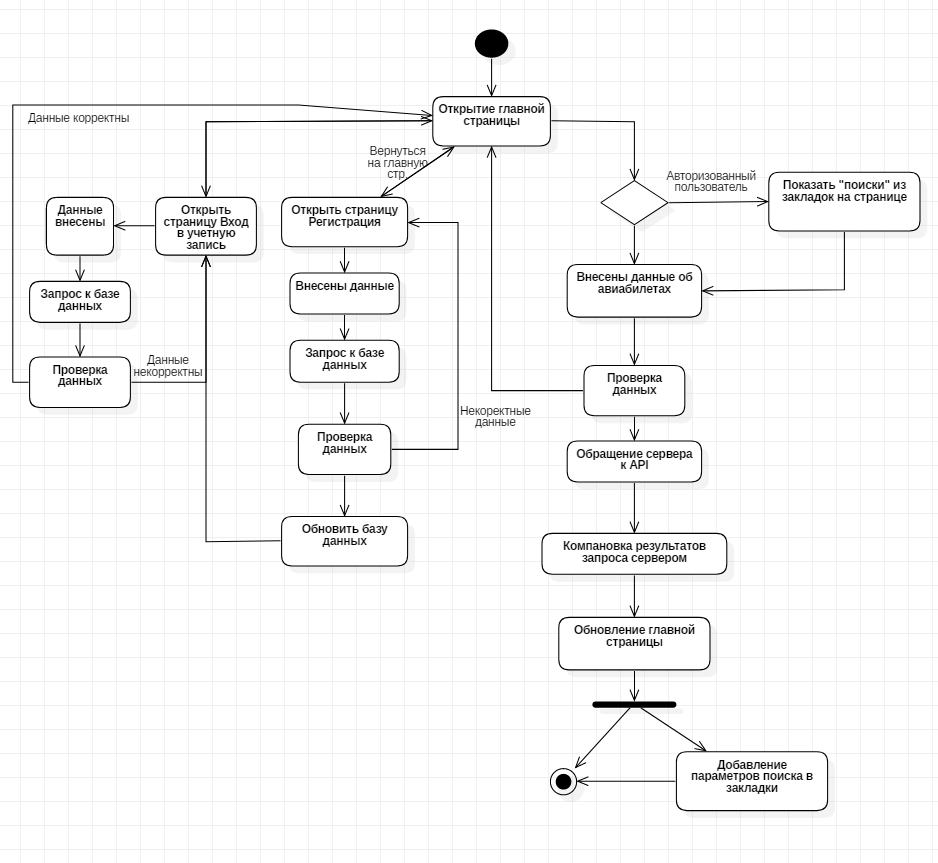
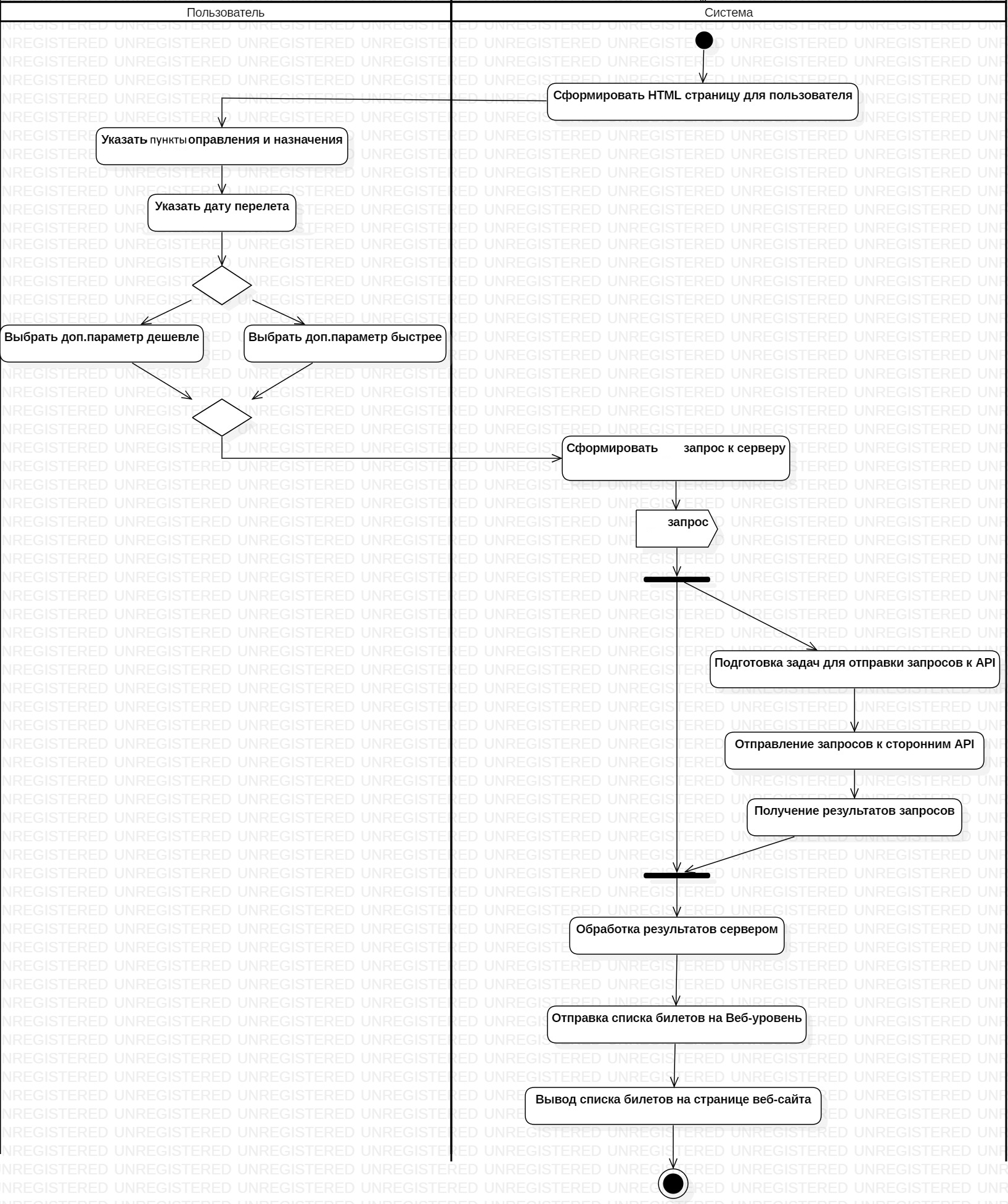
Данная диаграмма описывает все состояния, в которых находится система при выполнении основных пользовательских сценариев (авторизация, регистрация, поиск билетов, добавление поиска в закладки, использование шаблона поиска из закладок), за исключением сценария просмотра истории.

Рисунок 12 диаграмма состояний

### Диаграмма активности

Приведенная диаграммы активности являются расширением диаграммы состояний, для случая поиска билетов. Диаграмма разделена на две части – действия пользователя на клиенте и действия системы, которые включают в себя работу скриптов, модулей работы со сторонними API и модулей обработки результатов запросов к таким источникам.

Рисунок 13 диаграмма активности “Поиск билетов”



## Анализ средств реализации

При создании и тестировании сайта использовался локальный веб-сервер Denwer с установленной СУБД MySQL 5.1, а также HTTP-сервером Apache 2.2.22

Для упрощения работы с базой данных было использовано веб-приложение PhpMyAdmin версии 3.5.1

### Клиентская часть

При выполнении клиентской части были использованы:

1. HTML — стандартизированный язык разметки документов
2. CSS — формальный язык описания внешнего вида документа, написанного с использованием языка разметки.
3. JavaScript — мультипарадигменный язык программирования. В приложении был выдержан в функциональном стиле, т.к. основные скрипты написаны в одном файле и не инкапсулированы. Для создания приложения был использован плагин ZebraDatePicker, позволяющий упростить выбор даты поездки. Также была широко использована библиотека jQuery, в частности: для редактирования выпадающего списка городов посредством AJAX-запросов сторонним api; для общения с сервером и динамической выдачи информации об авиабилетах; для обеспечения работоспособности вышеупомянутого плагина.

### Серверный уровень

Языком программирования для серверного уровня стал PHP, ниже приведены причины такого выбора:

1. PHP — гибкий и универсальный язык. Он совместим со всеми основными платформами: Windows, Unix, Linux и MacOS; поддерживает большинство серверов (включая Apache, Microsoft IIS, Netscape, iPlanet, Caudium, Xitami и Tornado) и более 20 баз данных (среди которых — MySQL, MongoDB, PostgreSQL и др.). Это свойства языка php делает приложение, написанное на нем, большой степени переносимым.
2. Код на PHP прозрачен и отлично документируется. А значит, требуется меньше времени для того, чтобы любой программист мог понять.
3. Объектно-ориентированность. Разработка может происходить в несколько итераций (Например, в нашем случае: модуль общения с API, модуль декодинга json и сортировки данных, модуль общения с бд и т.д.). Таким образом, выполняется требование масштабируемости, указанное в техническом задании.

### Интеграция сторонних источников:

Одним из важнейших этапов разработки приложения стала интеграция сторонних сервисов. Нужно было сделать приложение гибким, с возможностью расширения и улучшения качества поиска.

Выбор сторонних источников данных основывался на следующих требованиях – они должны быть полными, актуальными и достоверными. Кроме того, эти сервисы должны предоставлять бесплатный доступ к своим API. Были выбраны следующие источники:

* Яндекс Расписания – сервис для поиска рейсов и маршрутов, предоставляющий свой API
* travelpayouts.com - API автокомплита для стран, городов и аэропортов для полей пункта отправления и пункта прибытия

# Реализация

## Диаграмма развертывания

Для наглядности и удобства дальнейшего описания архитектуры и функциональности веб-сервиса ниже приведена диаграмма развертывания, которая демонстрирует физическое развертывание артефактов на узлах.

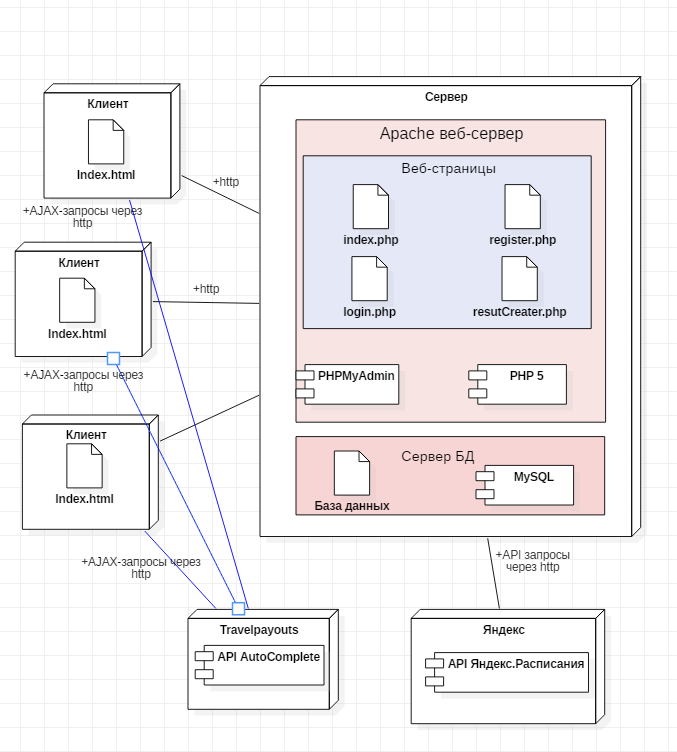


Рисунок 14 диаграмма развертывания

## Архитектура приложения

Веб-сервис придерживается архитектурного стиля REST, благодаря этому клиентская часть приложения может быть отделена от серверной, к примеру, два этих модуля могут быть развёрнуты на двух разных серверах. Все операции сервера с сервером проходят в режиме stateless – сервер не хранит никакой информации о сервере.

Ниже приведет список с кратким описанием методов, предоставляемых REST-сервисом

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Тип метода | Параметры | Описание |
| /resultCreater/get-tickets/ | GET | departurePoint, arrivalPoint, fdate, dIsAirport, aIsAirport, preference  все параметры кроме preference обязательны | Поиск по входным данным возвращает список билетов, соответствующих параметрам |
| /story/putSearch/ | POST | departureName, arrivalName, ID, searchDate, fdate,  все параметры обязательны | Сохранить в БД поисковой запрос авторизованного пользователя |
| /story/Search/ | GET | ID | Возвращение истории поисковых запросов пользователя |

## База данных

Модуль приложения, отвечающий за историю поисков билетов обращается к базе данных MySQL, схема которой изображена на рисунке 15

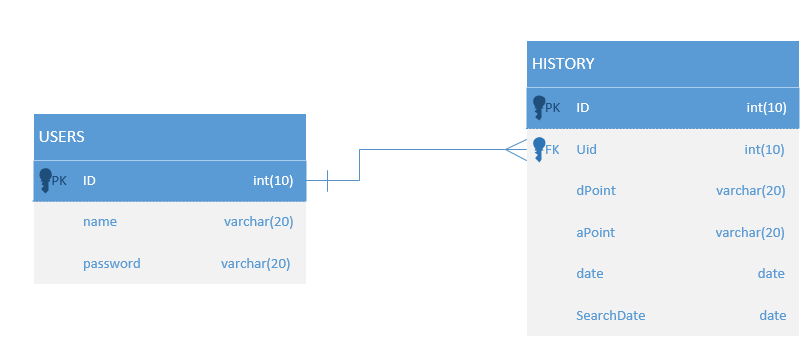


Рисунок 15 физическая модель базы данных

## Нереализованные функции

Приложение предоставляет пользователям все функции, указанные в техническом задании, за исключением закладок.

# Интерфейс

Пользовательский интерфейс содержит две основные страницу и 6 дополнительных. Все страницы выполнены в одной цветовой гамме. В верхней части страницы располагается навигационная панель, которая предоставляет пользователю перемещаться по сайту. В нижней части подпись с указанием разработчиков сайта.

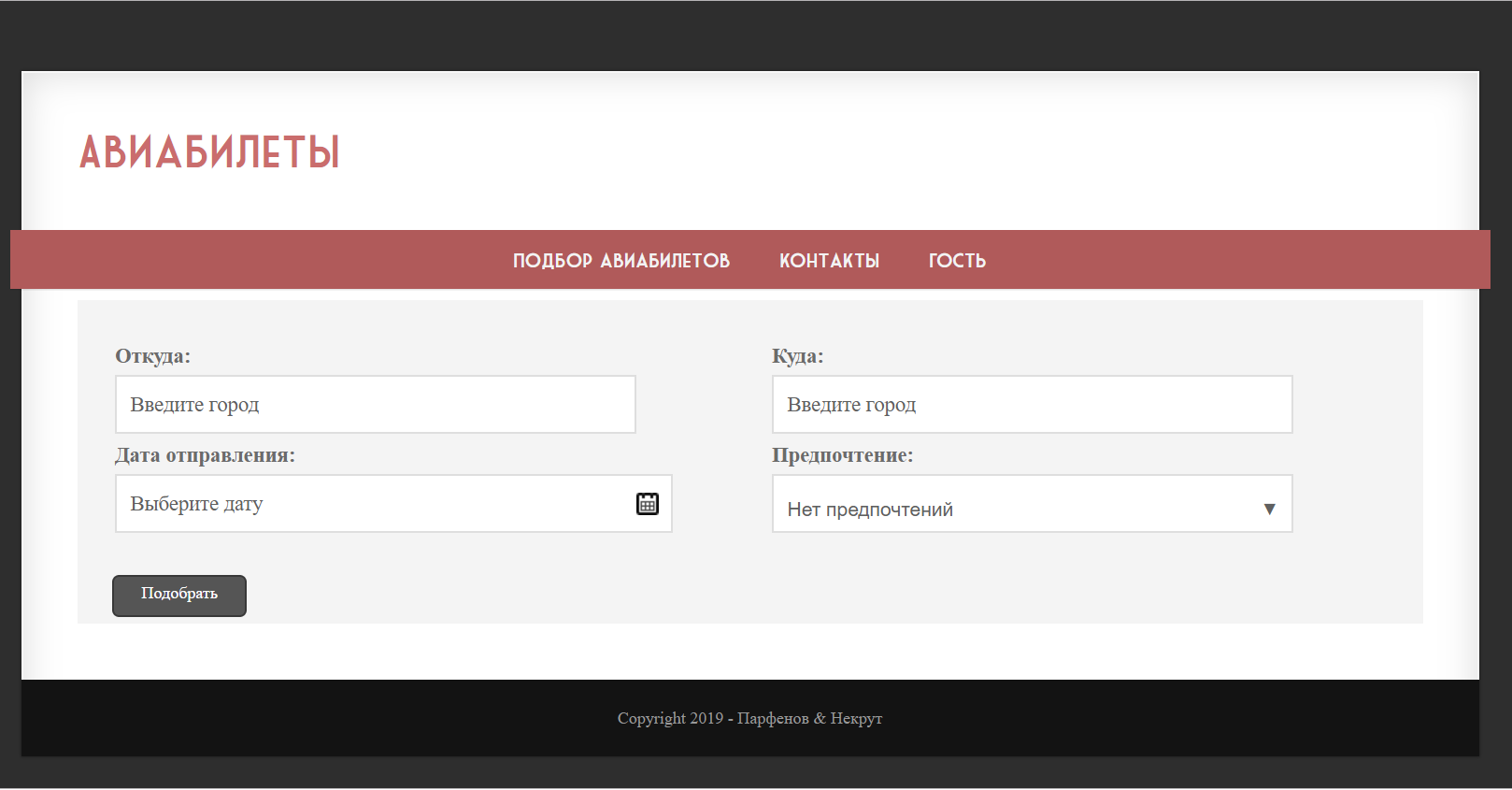


Рисунок 16 главная страница сайта

На главной странице поиска (рисунок 16) отображены:

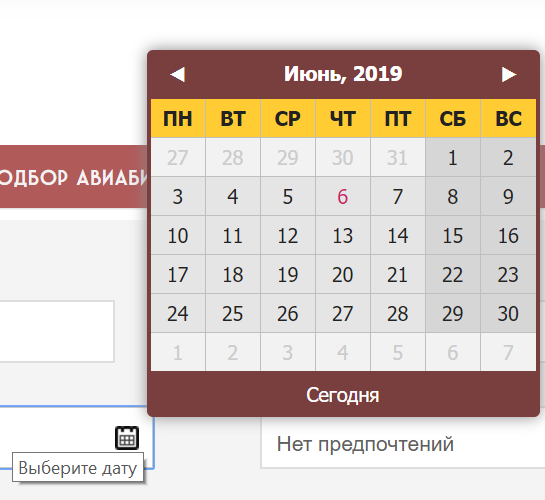
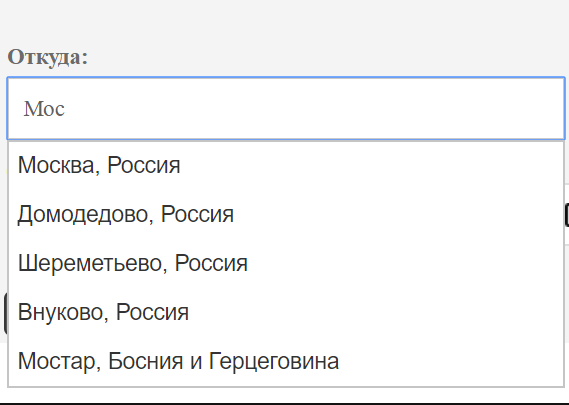
1. Поля для ввода точки отправления с функцией автозаполнения (рисунок 17 слева)
2. Поле выбора даты с плагином выбора по календарю (рисунок 17 справа)
3. Поле выбора предпочтения поиска
4. Кнопка осуществления поиска билетов

Рисунок 17 поля ввода пунктов отправления и даты

После ввода данных и нажатия на кнопку подобрать в нижней части страницы появляется информация о рейсах (рисунок 18):

1. Дата, время и аэропорт, из которого вылетает самолет
2. Время в пути
3. Дата, время и аэропорт, в который самолет приземляется
4. Цена билета
5. Информация о пересадках (рисунок 19), если они присутствуют (при отсутствии прочерк)



Рисунок 18 вывод информации о рейсах



Рисунок 19 вывод информации о рейсах (демонстрация пересадок)

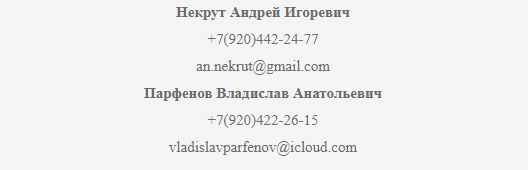
Страница «Контакты» (рисунок 20) содержит информацию о разработчиках, и способы связи с ними (номер телефона и e-mail). 

Рисунок 20 страница «Контакты»

Страница «История» (рисунок 21) показывает все предыдущие поисковые запросы конкретного авторизованного пользователя.

Выводится следующая информация:

1. Номер поиска
2. Дата поиска
3. Выбранные города отправления и прибытия
4. Выбранная дата перелета

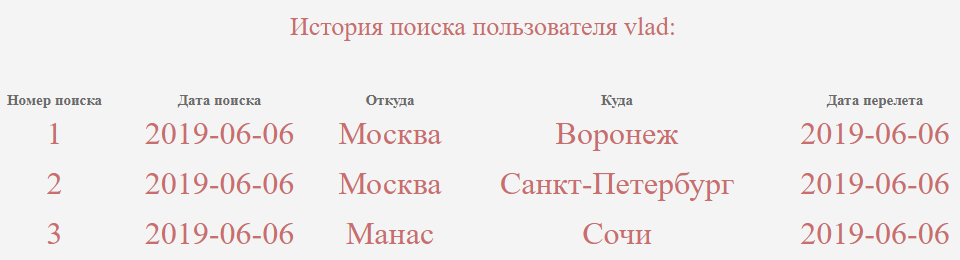


Рисунок 21 история поиска пользователя

Страница регистрации (рисунок 22) содержит два поля ввода: для логина и для пароля.

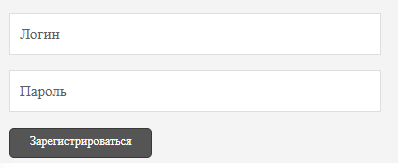


Рисунок 22 форма регистрации

Если указать уже зарегистрированный логин, то будет ошибка (рисунок 23).



Рисунок 23 обработка ошибки при регистрации

Страница авторизации (рисунок 24) так же содержит два поля ввода: для логина и для пароля.

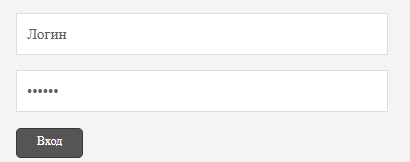


Рисунок 24 форма авторизации

Если ввести неправильный логин (рисунок 25 слева) или пароль (рисунок 25 справа), то появятся следующие сообщения ошибок:



Рисунок 23 обработка ошибок при авторизации

# План тестирования

Для того чтобы обеспечить соответствие системы требованиям, были выбраны следующие виды тестирования системы (указанные виды могут пересекаться между собой):

1. Smoke Testing – поверхностное тестирование всех модулей приложения на предмет их работоспособности. В нашей системе будет проверено соединения с Яндекс.API при запросе. В результате должно выводиться расписание рейсов. И так же будет проверено соединение с базой данных MySQL.
2. Sanity Testing – тестирование работы конкретных функций системы для доказательства того, что система работает согласно требованиям в Техническом Задании. В рамках данного теста будет проверяться возможность подбора авиабилетов с разными критериями.
3. Usability Testing – тестирование того, насколько легко конечный пользователь сможет освоить систему. Здесь проверятся наличие сообщений-подсказок для пользователя при неправильных действиях с его стороны.

Данные тесты будут проводиться по мере готовности соответствующих компонент, позволяющих произвести перечисленные виды тестирования.

В ходе тестирования была выявлена полная работоспособность системы, все критерии выхода из стадии тестирования были выполнены.

В результате тестирования в приложении был обнаружен один незначительный недочет, несильно влияющий на работу сайта: если ввести город собственноручно, а не выбрать предложенный из выпадающего списка, то подбор авиабилетов не будет осуществлен.

Но так как этот недочет не оказывает существенного влияния, можно сделать вывод, что система полностью соответствует поставленным требованиям и техническому заданию.

# Заключение

В ходе разработки веб-приложения была проанализирована предметная область, составлено техническое задание, рассмотрены аналоги сайта, выявлены их преимущества и недостатки. Итогом проделанной работы является полностью функционирующее приложение по подбору авиабилетов, обладающее следующими возможностями:

* Авторизация и регистрация пользователей
* Поиск рейсов по заданным критериям (быстрее или дешевле) и отображение пересадок при их наличии
* Возможность просмотра истории поисков пользователей

После завершения разработки сайта было проведено тестирование, в результате которого была установлена полная работоспособность приложения.

В итоге был создан определенно полезный веб-сайт, экономящий самый важный ресурс – время.

# Список источников

1. **Самоучитель по PHP**

<http://www.php-s.ru/self-teacher/>

1. **Robin Nixon** - *Learning PHP, MySQL, & JavaScript 4th Edition*
2. **Мэт Зандстра** - *PHP. Объекты, шаблоны и методики программирования*
3. **Документация по использованию API автокомплита для стран, городов и аэропортов**

https://www.clck.ru/GT2RM

1. **Документация по использованию Yandex.API** <https://tech.yandex.ru/rasp/doc/reference/schedule-point-point-docpage/>