МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра IT-инжиниринг

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой
PhD, доцент
_____ Т.С. Картбаев
«____» _____ 2019 г.

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка автоматизированной системы резервирования авиабилетов

Специальность: 5В070400 – «Вычислительная техника и программное обеспечение»

Выполнил: Исамбергенов С. М. Группа: ВТ-15-2
Научный руководитель: д.т.н., профессор Казиев Г. З.

Консультанты:
по экономической части: к.э.н., профессор _____ Ж.Г. Аренбаева
«13» мая 2019 г.

по безопасности
жизнедеятельности: д.т.н., ст. преп. _____ Ш.Ш. Бекбасаров
«14» мая 2019 г.

по применению
вычислительной техники: ст. преп. _____ М.Н. Майкотов
«14» мая 2019 г.

Нормоконтролер: ст. преп. _____ А.А. Айткazина
«15» мая 2019 г.

Рецензент: PhD, доцент _____ Ж.С. Есенгалиева
«____» _____ 2019 г.

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт систем управления и информационных технологий

Кафедра IT-инжиниринг

Специальность 5В070400 – «Вычислительная техника и программное обеспечение»

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Исамбергенову Султану Маликовичу

Тема проекта: Разработка автоматизированной системы резервирования авиабилетов

Утверждена приказом по университету № 124 от «26» октября 2018 г.

Срок сдачи законченного проекта «22» мая 2019 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): Руководство системы менеджмента качества на предприятии; международные стандарты ИСО-9001, данные преддипломной практики.

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта:

- аналитическая часть;
- проектная часть;
- экспериментальная часть;
- экономическая часть;
- безопасность жизнедеятельности;
- приложение А. Техническое задание;
- приложение Б. Листинг программы;
- приложение В. Акт внедрения.

Перечень графического материала (с точным указанием обязательных чертежей): представлены 9 таблиц, 55 иллюстраций.

Основная рекомендуемая литература:

1 Никсон, Робин Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript и CSS / Робин Никсон. - М.: "Издательство "Питер", 2013. - 560 с.

2 Титтел, Эд HTML, XHTML и CSS для чайников / Эд Титтел, Джефф Ноубл. - М.: Диалектика, 2016. - 400 с.

3 Макфарланд, Дэвид JavaScript и jQuery. Исчерпывающее руководство Дэвид Макфарланд. - М.: Эксмо, 2012. - 688 с.

4 Кингсли-Хью, К.Э. JavaScript 1.5: учебный курс / К.Э. Кингсли-Хью. - М.: СПб: Питер, 2013. - 272 с.

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Аренбаева Ж.Г.	04.03.2019 – 13.05.2019	
Безопасность жизнедеятельности	Бекбасаров Ш.Ш.	12.03.2019 – 14.05.2019	
Программное обеспечение	Майкотов М.Н.	18.04.2019 – 14.05.2019	
Нормоконтролер	Айтказина А. А..	04.04 – 15.05.19	

ГРАФИК
подготовки дипломной проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Аналитическая часть	05.11.2018 – 21.12.2018	
Проектная часть	07.01.2019 – 31.01.2019	
Экспериментальная часть	04.02.2019 – 12.04.2019	

Дата выдачи задания «29» октября 2018 г.

Заведующий кафедрой _____ Т.С. Картбаев

Научный руководитель проекта _____ Г.З. Казиев

Задание принял к
исполнению студент _____ С. М. Исамбергенов

Аңдатпа

Дипломдық жобаның тақырыбы: «Авиабилеттерді брондаудың автоматтандырылған жүйесін әзірлеу».
Жоба әуе билеттерін броньдау процесін автоматтандыратын веб-қосымшаны іске асыруға арналған.

Осы мақсатқа жету үшін HTML5, CSS, JavaScript, PHP, Bootstrap, XAMPP, jQuery, XHR, MySQL сияқты жаңа технологияларды оқып, қолдануы қажет.

Дипломдық жұмыс кіріспеден, 5 тараудан және қорытындыдан тұрады.

Кіріспе таңдап алынған тақырыптың өзектілігін көрсетеді, даму мақсаттары мен міндеттері белгіленеді, дамудың көлемін анықтайды.

Бірінші тарау технологияларды талдайды және таңдайды, ұқсас бағдарламалық өнімдермен салыстырғанда артықшылықтар мен кемшіліктерді талқылайды.

Екінші тарау қолданбаның дизайнын сипаттайды.

Үшінші тарауда тапсырманы іс жүзінде шешу кезеңдері сипатталады.

Төртінші тарау - әзірленген жобаның экономикалық бөлігін негіздеу.

Бесінші тарау ағымдағы жобаның шеңберінде еңбек жағдайларын жақсарту жөніндегі шаралары қарастырылады.

Диплом жобасында жұмыс істеу барысында жаңа технологиялар зерттелді және мақсаттарға қол жеткізілді.

Аннотация

Тема дипломного проекта: «Разработка автоматизированной системы резервирования авиабилетов».

Проект посвящен реализации веб-приложения, позволяющей автоматизировать процесс резервирования авиабилетов.

Для достижения поставленной цели необходимо изучить и использовать такие новейшие технологии, как: HTML5, CSS, JavaScript, PHP, Bootstrap, XAMPP, jQuery, XHR, MySQL.

В дипломный проект входят введение, 5 глав и итоговое заключение.

Во введении раскрывается актуальность выбранной темы, ставится цель разработки и задачи, определяется область применения разработки.

В первой главе производится анализ и выбор технологий, рассматриваются преимущества и недостатки по сравнению с аналогичными программными продуктами.

Во второй главе описывается проектирование приложения.

Третья глава поэтапно описывает практическое решение поставленной задачи.

В четвертой главе производится обоснование экономической части разрабатываемого проекта.

В пятой главе рассматриваются мероприятия по улучшению условий труда в рамках реализуемого проекта.

В процессе работы над дипломным проектом были изучены новейшие технологии и достигнуты поставленные цели.

Annotation

Theme of the graduation project: “Development of automated ticket reservation system”.

The project is dedicated to the implementation of a web application that automates the process of reserving air tickets.

To achieve this goal, it is necessary to study and use the latest technologies such as: HTML5, CSS, JavaScript, PHP, Bootstrap, XAMPP, jQuery, XHR, MySQL.

The graduation project includes an introduction, 5 chapters and a final conclusion.

The introduction reveals the relevance of the chosen topic, sets the development goal and objectives, determines the scope of the development.

The first chapter analyzes and selects technologies, discusses the advantages and disadvantages compared to similar software products.

The second chapter describes the design of the application.

The third chapter describes in stages the practical solution of the task.

The fourth chapter is the substantiation of the economic part of the developed project.

The fifth chapter deals with measures to improve working conditions in the framework of the ongoing project.

In the process of working on the graduation project, the newest technologies were studied and the goals were achieved.

Содержание

Введение

Системы бронирования авиабилетов были впервые введены в конце 1950-х годов как относительно простые автономные системы для контроля инвентаризации рейсов, поддержания расписаний рейсов, назначения мест и загрузки самолетов.

Современная система бронирования авиабилетов - это комплексный набор продуктов, обеспечивающий систему, которая помогает выполнять различные задачи по управлению авиакомпаниями и удовлетворять потребности клиентов с момента первоначального бронирования до завершения полета. Один из самых распространенных способов передвижения - путешествие по воздуху. Клиенты, имеющие желание путешествовать по воздуху в настоящее время, имеют широкий выбор авиакомпаний и выбор времени. В настоящее время конкуренция между авиакомпаниями настолько сильная, что существует множество скидок и множетсов разных услуг, предоставляемых клиентам, которые дадут преимущество этой конкретной авиакомпании. За последние четыре года Всемирная паутина стала чрезвычайно популярной, и в настоящее время большинство авиакомпаний предусматривают онлайн-бронирование своих рейсов.

Интернет стал основным ресурсом для людей, ищущих возможность бронирования через Интернет с помощью турагентств и без каких-либо проблем. Данный проект следует цели проверить все доступные авиарейсы, состоящие в базе данных авиакомпании Air-Sultan и вернуть результат, который может помочь клиентам в составлении плана их поездок.

Целью этого проекта является создание системы бронирования авиабилетов, где путешественник может запросить всю информацию о рейсе в соответствии с датой поездки. Они могут получать информацию о времени, стоимости и т. д. Одновременно и в одном месте. Система отображает все доступные авиакомпании, расписание и цены. Эта система поможет авиакомпании лучше обслуживать своих клиентов, удовлетворяя их потребности. Сайт будет использовать базу данных для хранения этой информации, а также самой последней информации о ценах и доступности для авиакомпании.

Проект системы бронирования авиабилетов - это реализация общего веб-сайта AirlineTicketing, такого как Aviasales который помогает клиентам искать наличие и цены различных авиабилетов, доступные при бронировании. Этот проект также охватывает различные функции, такие как онлайн-регистрация пользователей, изменение информации о веб-сайте управляющим персоналом или администратором веб-сайта, путем добавления, удаления или изменения информации о рейсах.

Ставится задача разработки многопользовательской системы, предназначенной для поиска и заказа билетов на авиарейсы. Каждому пользователю должна предоставляться возможность найти интересующие его рейсы, получить информацию о времени отбытия и прибытия, авиакомпании,

обслуживающей данный рейс, а также сделать заказ определенного количества билетов на выбранный рейс. Кроме того, система должна предоставлять администратору системы WEB-интерфейс для её сопровождения.

1 Описание программного продукта

Система бронирования авиабилетов - это одна из модификаций, которые были выполнены в системе обслуживания пассажиров и является интерфейсом глобальной системы распределения для осуществления бронирования для нужного авиаагентства из любого места. Система резервирования авиабилетов облегчает жизнь пассажирам, поскольку им не нужно стоять в очередях, чтобы зарегистрировать свои места, и они могут легко забронировать рейс из единой системы. С другой стороны, это также поспособствовало снятию дополнительной нагрузки с авиакомпаний, поскольку большинство пассажиров и туристических агентств начали использовать эту услугу. С помощью этой системы клиенты могут просматривать доступность всех рейсов с различными рекомендациями на конкретную дату, а также они могут забронировать место, отменить бронирование или изменить его. Единственная проблема с системой заключается в том, что она не позволяет пассажирам изменять конкретную часть своего бронирования. Помимо деталей рейса, он также отображает информацию о том, сколько осталось мест на тот или иной рейс. Пользователи следуют одним и тем же шагам, то есть проходят регистрацию, ищут рейс и бронируют его. Сначала разрабатывается программа, а затем интегрируется на веб-сайт.

С одной стороны, это помогает клиентам, а с другой - облегчает жизнь компаниям, обслуживающим авиалинии, сохраняя все записи о пассажирах, и, если по каким-либо причинам произошли какие-либо изменения в рейсе, то незамедлительно все пассажиры информируются. Эта система также используется компаниями для отслеживания пользовательских предпочтений постоянных клиентов, чтобы у них появлялась возможность предоставлять более качественные услуги и предлагать их клиентам. Это также помогает максимизировать получение доходов авиакомпаниями различными способами.

Таможенные пассажиры могут использовать эту систему для получения информации о специальных предложениях и скидках, предоставляемых ими. Для некоторых путешественников, которые путешествуют по частям или вынуждены путешествовать по разным направлениям один за другим, эта система также помогает им выбрать наилучшую возможную комбинацию для них.

Пользователи могут создавать учетные записи в системе и не обязываются заполнять необходимую информацию каждый раз, когда они бронируют, что экономит достаточное количество времени. С точки зрения пользователя интерфейс системы очень прост, и для его использования не требуются технические знания. Но чтобы использовать эту систему бронирования, клиенты должны сначала создать учетную запись и заполнить всю необходимую информацию. Если клиент уже забронировал билет, он

может также запланировать свое бронирование на любой другой желаемый рейс.

Основная цель этого программного обеспечения состоит в том, чтобы уменьшить количество ручных ошибок, связанных с процессом бронирования авиабилетов, и сделать так, чтобы клиентам было удобно бронировать рейсы, когда они этого требуют, чтобы они могли использовать это программное обеспечение для бронирования, изменения бронирования или отмены определенного предварительный заказ. Название программного обеспечения - «Air-Sultan».

Программное обеспечение предоставляет опции для просмотра различных рейсов, доступных с разными временами на определенную дату, и предоставляет клиентам возможность зарезервировать авиабилет, изменить или отменить конкретное бронирование, но оно не предоставляет клиентам детали стоимости билета и не позволяет клиенту изменить определенную часть своего бронирования, и клиент может изменить все детали.

1.1 Назначение программного продукта

Предлагаемый программный продукт предназначен для автоматизации резервирования авиабилетов. Структура сайта предполагает собой удобный с точки зрения пользователя интерфейс. Само приложение включает в себя три части:

Регистрация и вход в систему. Данный пункт позволяет только зашедшему пользователю зарегистрировать свои данные в базе данных, тем самым освобождая себя от надобности каждый раз вводить данные, что позволяет пользователю существенно сэкономить некоторое количество времени и облегчить дальнейшее пользование сайта.

При регистрации пользователю будет необходимо ввести данные, которые потребуются для бронирования билета, а именно: ФИО, дата рождения, e-mail и номер мобильного телефона. Вся информация, введенная пользователем при регистрации, необходима для получения доступа к аккаунту пользователя, а также его корректных идентификационных данных.

Часто возникают случаи, когда пользователь забывает свои логин и пароль, в этом случае он обращается к администратору путем соответствующей кнопки в панели входа.

На данном сайте также присутствует панель входа для администратора, имеющего привилегии, позволяющие добавлять, обновлять или удалять рейсы.

Вход в систему осуществляется путем корректного ввода логина и пароля. Все указанные данные при регистрации указываются в таблице, которое появляется после бронирования.

После регистрации и входа пользователя в систему под своими идентификационными данными, пользователь сможет приступить к использованию сайта.

Бронирование авиабилетов. Для достижения цели автоматизации этой системы каждому пользователю предоставляется возможность зарезервировать билет с любого города Казахстана на доступное время, но при условии, что еще остались свободные места в данном рейсе.

Бронируя рейс с одного города на другой в определенный день или время может оказаться такой случай, когда найдется либо несколько совпадений, либо ни одного. Чаще всего последний момент случается с теми пунктами назначения, спрос на которые не особо велик, и соответственно не нуждается в частой перевозке клиентов.

Панель для бронирования такой же, какой и в случае с регистрацией, с точки зрения пользователя, не затруднителен и комфортабелен для пользования.

Эффективность системы бронирования зависит от того, как организованы данные. В существующей системе большая часть данных вводится вручную, и это может быть очень проблематичным. Во избежание данной проблемы, решается, что наиболее логичным выводом в данной ситуации является применение дропдаунов, позволяющие выбирать пользователям данные из списка, нежели вписывать данные самому, тем самым предотвращая ненужные ошибки.

После введения данных пользователем, а это: пункт отбытия, пункт прибытия, дата отбытия и прибытия, и выбора класса, пользователь переходит на следующую страницу, где выводится список доступных рейсов на выбранную дату с одного пункта на другой. Затем, выбрав рейс, забронировав себе место, пользователь добавляет его в корзину, что позволяет пользователю, у которого будет время, передумать, или выбрать наиболее эффективный рейс. Пользователь не сможет забронировать себе место только в том случае, когда все места в самолете будут заняты. Таким образом, происходит бронирование в одну сторону, и без пересадок.

Бронирование типа туда и обратно предполагает рейс из пункта А в пункт Б и обратно. Сначала находятся все рейсы из пункта А в определенную дату, и сопоставляются со всеми рейсами, доступными обратно из пункта Б в пункт А в ту дату, которая уканаза на момент возвращения.

Также имеется панель с поиском всех доступных рейсов на каждый день, с одного пункта на любой другой. Обычно эта панель имеет слишком много данных, и носит чаще ознакомительный характер, потому что обычному пользователю сложно скоординироваться в огромной куче данных. Все детали, указанные на этой странице, полностью соответствуют тому, что указано на двух остальных, другими словами, фактически идентичен.

Корзина. Данный компонент является элементом интерфейса пользовательс, который предназначен для временного хранения данных о рейсах. Данные, находящиеся в корзине, имеют свойство, позволяющее показывать состояние рейса, его детали. Данные сохраняются до тех пор, пока клиент не решит либо отменить рейс, либо подтвердить его, другими словами, оплатить.

История. Данный компонент представляет пользователю увидеть историю его покупок, со всеми возможными данными, и не обладает рядом функционалов;

Добавление и изменение рейса администратором. Данный элемент панели администратора позволяет добавить определенный рейс с одного пункта назначения на другой. Данные, требующиеся для добавления администратором: название рейса в формате AA000, пункт и время отбытия, пункт и время прибытия, при чем, стоит заметить, пункты отбытия и прибытия не могут иметь одинаковое значение, также указывается модель самолета, на котором будет совершаться данный рейс, класс рейса, их вместимость и цена.

Модели самолета выбираются из базы данных. Существует два типа класса рейсов: эконом и бизнес. Эконом класс предполагает собой более бюджетный вариант со стандартным набором услуг, не имеющая никаких привилегий, и, тем самым, предиктивно имея цену намного ниже чем в бизнес классе.

Для изменения данных рейса, администратор должен нажать на соответствующий элемент страницы. Для нахождения деталей рейса используется поиск в базе данных по полю названия рейса, начав поиск, выводятся все данные, принадлежащие этому рейсу, и администратор может подкорректировать ранее внесенные ошибочные значения.

Удаление рейса применим только в том случае, когда администратор вводит некорректные значения и предназначен во избежание дальнейших не случившихся проблем.

1.2 Механизм работы резервирования билетов

Пользователи могут легко приобрести электронный билет, зайдя на сайт продажи билетов, выполнив поиск и выбрав пункт назначения, введя такие данные, как имя, способ путешествия, информацию о багаже и датах, и осуществить платеж с помощью банковских карт, банковского перевода или через онлайн платежные компании. Затем электронный билет отправляется по электронной почте или отправляется по телефону на телефон клиента. Если раньше туристические агенты и авиакомпании помогали клиентам совершать покупки билетов, то сегодня, благодаря усовершенствованной интернет-системе, бронировать авиабилеты становится все проще и проще. После того, как клиент совершил покупку, электронная запись и данные билета сохраняются в базе данных авиакомпании. База данных интегрирована с системой обслуживания пассажиров, которая затем подключается к аэропортам, авиакомпаниям, туристическим агентствам для обмена информацией в режиме реального времени.

Причина, по которой все больше и больше людей покупают билеты на рейсы, заключается в множестве преимуществ и удобств, которые может предложить система онлайн-бронирования авиабилетов. В отличие от

традиционного способа покупки билетов через офис турагентов, онлайн-бронирование сегодня предоставляет доступ к сотням рейсов, их ценам и другим услугам одним нажатием кнопки. Это может быть очень важно для клиента, чтобы найти наиболее выгодную сделку и наиболее удобное соединение. Онлайн система бронирования авиабилетов дает больше контроля клиентам в плане планирования отпуска и помогает им принимать обоснованные решения.

Система поиска рейсов настолько проста, что любой, у кого есть компьютер и доступ к Интернету, может легко найти, сравнить цены и приобрести любое рейс, которое пожелает.

Многие авиакомпании в наши дни также предоставляют другие услуги, такие как трансфер из аэропорта, услуги размещения, прокат автомобилей и парковка в аэропорту. Клиенты также могут воспользоваться частыми скидками, акциями и другими предложениями авиакомпаний, подписавшись на их электронную почту или просто просмотрев их веб-сайт. Клиенты также могут в полной мере знать о различных политиках компании, таких как отмена, правила провоза багажа и другие, также может безпрецедентно использовать эти услуги. Онлайн бронирование авиабилетов не только обеспечивает удобное и быстрое обслуживание клиентов с помощью компьютера из любой точки мира, но также часто предоставляет услуги по более доступным ценам.

Большинство авиакомпаний также предоставляют услуги онлайн-регистрации, после чего клиент может легко распечатать посадочный талон дома и сэкономить немало времени и хлопот в аэропорту.

Интерфейс данного приложения спроектирован таким образом, чтобы любому пользователю, даже тому, который видит это приложение впервые, можно было легко сориентироваться в структуре системы и ее возможностях.

Разумеется, для этого ему будет необходимо пройтись хотя бы раз по всем вкладкам приложения, попробовать весь его функционал. Но с большой вероятностью для пользователя не останется «белых» пятен в приложении, которые будут смущать его в ходе использования.

Система онлайн-бронирования авиабилетов также имеет свои ограничения. Потенциальным клиентам в Азии и других частях мира, где скорость и доступность соединения недостаточны, может быть сложно получить доступ к веб-сайту авиакомпаний и забронировать билеты. В большей части менее развитой части мира, помимо подключения к Интернету, ограниченные знания и доступ к технологиям также могут ограничивать возможности использования этой услуги. В результате многие люди в развивающихся странах все еще используют традиционный метод покупки билетов в офисах турагентов. Сбои машины, такие как потерянное соединение или не отвечающая программа, иногда могут также привести к исчезновению всего маршрута полета. Кроме того, клиент может быть не в состоянии получить правильную или достаточную информацию только с веб-сайта,

поскольку это не делается лицом к лицу с человеком, который знает все о связях, предложениях и политике компании.

Также существует проблема безопасности: иногда возможно, что кто-то может украсть информацию о вашей кредитной карте, но разработчики с каждым днем улучшают безопасность системы с постоянным развитием технологий.

1.3 Обзор существующих аналогогичных программ

В настоящее время разрабатывается большое количество программ, где каждый имеет определенное преимущество либо недостаток перед другой программой. Все программные продукты, занимающие определенную нишу в сфере предоставляемого функционала, имеют как сходные компоненты, так и что-то своё, что-то уникальное.

Именно поэтому у одних программных продуктов может быть очень большая аудитория благодарных пользователей, а другие продукты, несмотря на их схожесть, могут так и остаться неудачной попыткой решения поставленных задач.

В процессе проектирования любой системы в приоритете должен быть пользователь, который в конечном счете и будет использовать программный продукт.

Далее будет рассматриваться перечень программных продуктов, которые схожи по выполняемым функциям с тем, что разрабатывался. Будут отмечены плюсы и минусы аналогичного программного обеспечения относительно разработанного приложения.

Такой анализ позволит назвать сильные стороны приложения на нынешнем этапе его реализации, а также поможет выяснить, какие функции также пользуются популярностью и в процессе развития могут быть добавлены для использования конечным пользователем.

1.3.1 Aviasales

Регистрацию на этом сайте можно пройти разными сервисами, такими, как VK, Facebook, Twitter, Одноклассники. Это один из этапов упрощения пользования сайтом для пользователься, ведь не приходится вводить все данные, потому что они указаны в Вашем аккаунте социальной сети, хоть и может не содержать всех необходимых данных. Подтверждая соглашение, мы не можем знать на сто процентов, что происходит с нашими данными после регистрации и в каких целях они могут быть использованы. Данный пункт, возможно, будет относиться ко всем программам данного типа, с целью узнать геолокацию пользователя, и это, скорее всего, относится больше к преимуществу, нежели к недостаткам.

Во-вторых, данное приложение используется не для того, чтобы продавать сами авиабилеты, а находить их по всем доступным авиа-

агентствам, позволяя пользователю найти либо наиболее дешевый вариант, либо самый быстрый. Сайт предоставляет общение с чат-ботом, который отвечает на частозадаваемые вопросы, что существенно облегчает пользователю процесс бронирования при возникновении какого-либо рода ошибок или несостыковок.

И, в-третьих, здесь есть возможность оформить такси из аэропорта и забронировать номер в отеле. Данный сервис значительно упрощает перевозку клиента с одной точки на другой.

Этот сайт является хранимым и выполняемым в памяти компьютера или телефона, что создает дополнительную нагрузку на аппаратные части устройства пользования.

Из положительных моментов: приятный интерфейс для пользователя, наличие десктопных и мобильных версий для смартфонов и планшетов (вне зависимости от используемой платформы).

В общем, данный сайт, в виду своей простоты и уникальности, имеет большой спрос среди клиентов.

1.3.2 Scyscanner

Скайсканер – это сайт, с более узким набором функционала, но не делающая его менее уникальным. В нем существует очень много полезных и важных сервисов. Стоит отметить, что данное приложение имеет удобный интерфейс, и для поиска необходимого рейса нужно лишь ввести пункты отбытия и прибытия, дату, указать, на какое количество людей осуществляется, собственно, бронирование. Рейс имеет как эконом, так и бизнес класс, также имеет разную ценовую категорию, в зависимости от его социального положения.

Данный сервис отличается тем, что он ни хуже, и ни лучше, но является очень удобным во всех его аспектах. Данный сайт также предоставляет возможность забронировать номер, но кроме того, имеется услуга проката автомобился на определенный срок за определенную цену, так как этот прием очень популярен во многих странах. Тут нет такого большого количества рекламы, администраторы выступают в роли помощников и предоставляют благие условия, отвечая на возникшие вопросы

Такое приложение является самым удачным вариантом, если клиенту необходимо сразу после высадки поехать по делам, но заказывать такси было бы не так экономично.

1.3.3 AnywayAnyday

Данный сервис отличается своей уникальностью. Он предполагает бронирование не только на воздушные перелеты, но и на железнодоржные перевозки. В список сервисов сайта, отличающихся от услуг других аналогичных сайтов, делающих их уникальными, являются следующие

сервисы: построение личного маршрута, трансферы, возможность указывать количество пассажиров, требующих бронирование, услуга обратного перелета.

Построение личного маршрута удобно тем, что позволяет пользователю уточнять свой маршрут, и для достижения целевого маршрута придется указать несколько пунктов назначения. Услуга бронирования ЖД билетов работает по такой же схеме: указываются пункты отбытия и прибытия, дата, и количество пассажиров, на которые нужно оформить заказ.

В наличии сервисов сайта также находится чат, позволяющий непосредственно общаться с менеджером сайта. В наше время сложно представить крупные сайты, без наличия в нем обратной связи, что существенно сужает круг поиска вопросов.

Вывод для данного сайта: удобный интерфейс, наличине уникальных услуг, обратная связь с менеджером, наличие трансфера и возможности брони ЖД билета.

1.4 Выбор средств и технологий

Данный подраздел повествует о том, какие технологии были использованы при разработке дипломного проекта, их основные достоинства и недостатки. Также поясняется, почему были выбраны именно эти технологии, и за какую часть в приложении они несут ответственность.

1.4.1 PHP

PHP: Hypertext Preprocessor (или просто PHP) - это язык программирования общего назначения, изначально разработанный для веб-разработки. Референсная реализация PHP теперь производится компанией The PHP Group. Изначально PHP расшифровывался как Personal Home Page, но теперь он обозначает рекурсивный PHP: Hypertext Preprocessor.

Код PHP может выполняться с интерфейсом командной строки (CLI), встроенным в код HTML, или его можно использовать в сочетании с различными системами веб-шаблонов, системами управления веб-контентом и веб-платформами. PHP-код обычно обрабатывается интерпретатором PHP, реализованным в виде модуля на веб-сервере или в виде исполняемого файла Common Gateway Interface (CGI).

Веб-сервер объединяет результаты интерпретированного и исполняемого кода PHP, которые могут быть любыми типами данных, включая изображения, с созданной веб-страницей. PHP может использоваться для многих задач программирования вне веб-контекста, таких как автономные графические приложения и управление роботизированным дроном.

1.4.2 HTML

HTML (HyperText Markup Language) – язык гипертекстовой разметки, предназначенный для размещения объектов web-страницы в определенном порядке. Этот язык в первую очередь позволяет определять тип помещаемого объекта, например, ссылка, рисунок, медиа-файл, какой-либо скриптовый файл, либо самый обычный текст. Это все осуществляется при помощи набора тегов. Также этот язык является связующим звеном между страницей браузера и дополнительными технологиями, которые будут использоваться.

Ни одна современная web-страница не обходится без использования этой разметки. Для разработки предложенного проекта был использован стандарт последней версии – HTML5, который отличается от своих предшественников более строгими правилами, является продуктом, который сочетает в себе свойства и синтаксические нормы стандартов HTML и XHTML, а также направлен на поддержку большего числа мультимедийных технологий.

1.4.3 CSS

CSS (Cascading Style Sheets) – каскадные таблицы стилей – формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

Главная задача CSS – это корректное визуальное представление страницы перед пользователем согласно заданным стилевым правилам.

CSS работает с большим множеством элементов страницы: шрифты, цвета фона и символов, поля, таблицы, строки, высота и ширина элементов, отступы, изрбражения, их корректным отображением, с позиционированием элементов и многое другое.

При помощи CSS можно изменить любой компонент страницы и заставить его выглядеть так, как оно требуется для дизайна страницы.

Основная цель CSS – это разгрузить файл HTML, поскольку раньше логическая разметка и описание внешнего вида страницы хранилось в одном файле и значительно затрудняло поддержку сайта и его дальнейшую разработку.

CSS является неотъемлемой частью современного программирования, так как от визуального содержания страницы зависит его удобство в использовании и популярность среди пользователей. Чем лучше сделан сайт с визуальной точки зрения, тем он более конкурентоспособен на поисковых платформах.

1.4.4 JavaScript

JavaScript – это полноценный динамический язык программирования, который позволяет, применяя его к HTML-документу, обеспечивать интерактивность создаваемых web-страниц. Этот язык невероятно многофункционален. С помощью него можно как создавать какие-либо

галереи просмотра изображений, изменяющиеся макеты, так и более сложные элементы: 2D и 3D графика, полномасштабные приложения с базами данных, а также игры различной сложности.

JavaScript сам по себе довольно компактный, но очень гибкий. Разработчиками написано большое количество инструментов поверх основного языка JavaScript, которые дают доступ к огромному количеству дополнительных функций, стоит лишь приложить немного усилий. К ним относятся:

- программные интерфейсы приложения (API), встроенные в браузеры, обеспечивающие различные функциональные возможности, такие как динамическое создание HTML и установку CSS стилей, захват и манипуляция видеопотоком, работа с веб-камерой пользователя или генерация 3D графики и аудио-сэмплов;
- сторонние API позволяют разработчикам внедрять функциональность в свои сайты от других разработчиков, таких как Twitter или Facebook;
- также мы можем применить к нашему HTML сторонние фреймворки и библиотеки, что позволит нам ускорить создание сайтов и приложений.

1.4.5 jQuery

jQuery - это библиотека JavaScript, предназначенная для упрощения обхода и манипулирования деревом HTML DOM, а также обработки событий, CSS-анимаций и Ajax. Это бесплатное программное обеспечение с открытым исходным кодом, использующее разрешающую лицензию MIT. Веб-анализ (с 2017 года) показывает, что это самая распространенная библиотека JavaScript с большим отрывом.

Синтаксис jQuery разработан для упрощения навигации по документу, выбора элементов DOM, создания анимации, обработки событий и разработки Ajax-приложений. jQuery также предоставляет разработчикам возможность создавать плагины поверх библиотеки JavaScript.

Это позволяет разработчикам создавать абстракции для низкоуровневого взаимодействия и анимации, расширенных эффектов и высокоуровневых тематических виджетов. Модульный подход к библиотеке jQuery позволяет создавать мощные динамические веб-страницы и веб-приложения.

Набор базовых функций jQuery - выбор элементов DOM, обход и манипулирование - активирован его механизмом выбора («Sizzle» из v1.3), создал новый «стиль программирования», алгоритмы слияния и структуры данных DOM. Этот стиль повлиял на архитектуру других JavaScript-фреймворков, таких как YUI v3 и Dojo, что впоследствии стимулировало создание стандартного API селекторов.

1.4.6 Bootstrap

Bootstrap - это бесплатный фреймворк с открытым исходным кодом для создания веб-сайтов и веб-приложений. Это самый популярный фреймворк HTML, CSS и JS для разработки адаптивных проектов в Интернете.

Поскольку веб все больше и больше развивается в сторону адаптивного дизайна, веб-разработчикам может быть непросто идти в ногу со временем. Bootstrap упрощает данное дело. Bootstrap позволяет создавать адаптивные веб-сайты без необходимости вносить «отзывчивый» бит. Bootstrap позаботится об этом.

Одним из основных преимуществ таких сред разработки, как Bootstrap, является то, что они могут помочь ускорить время разработки, сохраняя при этом качество и согласованность всего сайта. Вам больше не нужно переделывать каждый элемент. И вам не нужно тратить часы на то, чтобы все выглядело и работало правильно на браузерах, платформах и устройствах. Используя Bootstrap, вся тяжелая работа для вас.

Учитывая, что Bootstrap является самой популярным фреймворком разработки веб-интерфейсов, этот набор навыков может быть полезен для изучения. Добавление Bootstrap может помочь вам во многих отношениях - от ускроения создания веб-сайтов, до воплощения в реальность вашей мечты.

Кроме того, Bootstrap имеет собственный набор стилей, которые легко переопределить. Нет ограничений в "Bootstrap design". Вы можете свободно использовать любые компоненты Bootstrap по своему усмотрению, при этом добавляя свои собственные. Существуют тысячи сайтов, построенных на Bootstrap, где каждый из них имеет свой уникальный дизайн.

1.4.7 Sublime text

Sublime Text – проприетарный текстовый редактор. В возможности данного редактора входят: быстрая навигация, командная палитра, API плагинов на Python, одновременное редактирование, высокая степень настраиваемости.

Sublime Text поддерживает большое количество языков и имеет возможность подсветки синтаксиса для C, C++, C#, CSS, D, Dylan, Erlang, HTML, Groovy, Haskell, Java, JavaScript, LaTeX, Lisp, Lua, Markdown, MATLAB, OCaml, Perl, PHP, Python, R, Ruby, SQL, TCL и XML.

Редактор содержит различные визуальные темы, с возможностью загрузки дополнительных. Пользователи видят весь свой код в правой части экрана в виде миникарты, при клике на которую можно осуществлять навигацию. Имеется несколько режимов экрана.

Функция автодополнения позволяет пользователю при наборе кода предлагать различные варианты для завершения записи.

Редактор также автоматически завершает созданные пользователем переменные. Также, в функционал Sublime Text входят немаловажные особенности: выделение столбцов и множественная правка, подсветка

синтаксиса и высокая контрасность, поддержка систем сборки, заготовки, переход по файлам и т.д.

1.4.8 Django

Django - это бесплатный веб-фреймворк с открытым исходным кодом, написанный на Python. Фреймворк - это не что иное, как набор модулей, облегчающих разработку. Они сгруппированы вместе и позволяют создавать приложения или веб-сайты из существующего источника, а не с нуля.

Именно так веб-сайты, даже простые, разработанные одним человеком, могут по-прежнему включать расширенные функции, такие как поддержка аутентификации, панели управления и администрирования, контактные формы, поля для комментариев, поддержка загрузки файлов и многое другое. Другими словами, если бы вы создавали сайт с нуля, вам нужно было бы разработать эти компоненты самостоятельно. Используя вместо этого фреймворк, эти компоненты создаются автоматически, и вам просто нужно правильно настроить их в соответствии с вашим сайтом.

1.4.9 MySQL

MySQL - поддерживаемая Oracle система управления реляционными базами данных с открытым исходным кодом (RDBMS), основанная на языке структурированных запросов (SQL). MySQL работает практически на всех платформах, включая Linux, UNIX и Windows. Хотя он может использоваться в самых разных приложениях, MySQL чаще всего ассоциируется с веб-приложениями и онлайн-публикациями.

MySQL является важным компонентом корпоративного стека с открытым исходным кодом под названием LAMP. LAMP - это платформа для веб-разработки, использующая Linux в качестве операционной системы, Apache в качестве веб-сервера, MySQL в качестве системы управления реляционными базами данных и PHP в качестве объектно-ориентированного языка сценариев. (Иногда вместо PHP используется Perl или Python.)

Первоначально задуманная шведской компанией MySQL AB, MySQL была приобретена Sun Microsystems в 2008 году, а затем Oracle, когда она приобрела Sun в 2010 году. Разработчики могут использовать MySQL в рамках GNU General Public License (GPL), но предприятия должны получить коммерческую лицензию от Oracle.

Сегодня MySQL является СУБД, стоящей за многими ведущими веб-сайтами в мире и бесчисленным количеством корпоративных и ориентированных на потребителя веб-приложений, включая Facebook, Twitter и YouTube.

1.5 Постановка цели и задач

Достижение цели не является одномоментным мероприятием. Достижение цели представляет собой решение совокупности задач, связанных между собой по смыслу и направленных для достижения единой цели.

Каждая из задач направлена на изучение или разработку каких-либо вещей в проекте. Таким образом, получается, что задачи – своеобразные «кирпичики» при выполнении которых достигается определенная цель.

Ниже приведены цель дипломного проекта и задачи, которые необходимо реализовать для достижения поставленной цели.

Целью дипломного проекта является разработка программного продукта, который позволит создать веб-приложение, позволяющее автоматизировать процесс резервирования авиабилетов.

Основные задачи дипломного проекта:

– изучение предметной области;

– определение основных требований к программному продукту;

– выбор и обоснование технологий для разработки программного продукта;

– разработка пользовательского веб-интерфейса;

– проектирование базы данных для хранения данных пользователей;

– добавление функциональности на стороне сервера: регистрация, аутентификация, поиск рейсов, оплата;

– обоснование экономической целесообразности разрабатываемого продукта;

– предложение мероприятий по улучшению условий труда в рамках реализуемого проекта.

2 Проектирование программного продукта

Данная глава предполагает собой описание проектирования и разработки базы данных. Непосредственно, раскроются такие понятия, как структура ПО и функциональная структура приложения.

Важно отметить, почему мы выбираем именно этот тип базы данных и каким образом этот выбор сможет повлиять на дальнейшее развитие приложения. В связи с этим предстоит предусмотреть все возможные преимущества и недостатки выбранного типа базы данных, чтобы в дальнейшем учесть их при проектировании

Также описание изучения предметной области: какие таблицы и сущности должны присутствовать в базе данных для корректной работы приложения.

База данных должна содержать в себе четкие, упорядоченные таблицы, данные в которые разделены по группам и связаны между собой.

2.1 Структура программного обеспечения

Структура программного обеспечения включает в себя любое программное обеспечение, которое было использовано при создании определенного приложения или пакета приложений.

Все программное обеспечение, используемое в создании ПО, можно разделить на 3 группы: системное ПО, прикладное ПО и инструментальное ПО.

Системное ПО - программное обеспечение общего пользования. Сюда относятся операционные системы и их оболочки, различные специфические драйверы и утилиты, системы, которые предназначены для технического обслуживания машины разработчика;

Прикладное ПО: сюда относятся различные прикладные программы и пакеты этих программ. Например, это программы для работы с текстовой информацией, её оформлением, программы для просмотра веб-страниц или веб-браузеры и т. д.;

Инструментальное ПО также называют системами программирования. Это инструменты, которые разработчик приложения использует для просмотра исходного кода программы, его редактирования и дальнейшего запуска. В настоящее время такие системы чаще всего называют интегрированными средами разработки, поскольку они включают в себя большой перечень инструментов для работы с различными языками программирования и СУБД.

Помимо основных трех, существует еще одна группа ПО, которая редко упомянается, - базовое ПО. Базовое ПО – это ПО, которое встроено в аппаратное обеспечение компьютера. Наиболее подходящим к данной категории относят самую базовую систему ввода и вывода.

Исходя из схемы на рисунке 2.1.1, видно, что разработка приложения происходила под управлением операционной системы Microsoft Windows последней версии.

Для оформления всего отчета по проделанной работе был использовал пакет программ Microsoft Office и веб-приложение draw.io, которое использовалось для построения структурных и других схем.

Текстовым редактором для реализации данного програмного продукта был выбран Sublime Text последней версии. Приложение не имеет большого веса, и позволяет легко править данные. Уникальность Sublime Text состоит в том, что она поддерживает большое количество языков, имеет различные особенности, которые описывались ранее.

Далее приведена схема структуры разрабатываемого приложения:

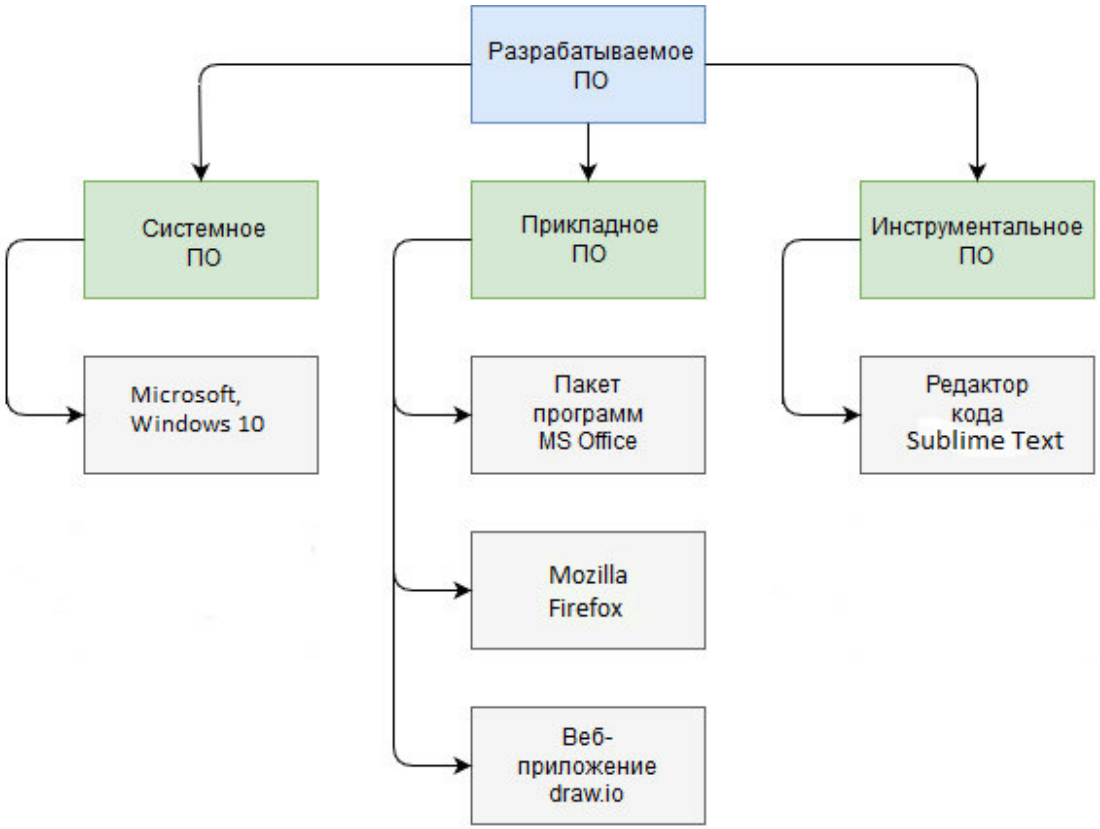


Рисунок 2.1.1 – Структура ПО

Такие продукты, как интегрированные среды разработки, позволяют сделать разработку программного обеспечения более простой и наглядной. Также они влияют на скорость разработки программного обеспечения и более простую реализацию каких-либо сложных моментов.

2.2 Функциональная структура веб-приложения

Функциональная структура веб-приложения предназначена для того, чтобы наглядно показать какие функции может предоставить приложение.

Глядя на такую структуру, сразу становится ясным стоит ли использовать это приложение и приведет ли оно к желаемому результату.

Приведенная далее структурная схема показывает, какие конкретно функции будут предоставляться разработанным программным продуктом. У любой структурной схемы веб-приложения корневым элементом схемы является либо главная страница, либо форма входа в аккаунт пользователя.

В данном случае главным корневым элементом схемы является форма входа, которая предлагает пользователю ввести свой логин и пароль для входа в систему. В том случае, если сотрудник является новым человеком для системы, то эта же форма предлагает ему перейти в форму регистрации. Таким образом, человек знает, куда направится в том случае, если вход в систему ему выполнить нужно, но он не имеет необходимых данных для этого.

Двойная стрелка на схеме говорит о том, что если человек из формы входа будет отправлен на регистрацию, то после завершения процесса регистрации он будет направлен назад к форме входа и сможет продолжить работу.

Регистрация предоставляется с целью уникальной идентификации каждого сотрудника системы. В Регистрации указываются все необходимые в дальнейшем данные пользователя. Таким образом, данные пользователя, которые он вводит в систему, защищены от посторонних глаз и большое количество пользователей может полноценно пользоваться данной системой.

Далее начинается процесс, который открывает пользователю к аккаунту. Этот процесс состоит из трех частей: идентификации, аутентификации и авторизации.

Процесс идентификации представляет собой сопоставление распознавания пользователя по его идентификатору. В данном случае идентификатором пользователя является его логин, который в принципе является уникальным и проверяется каждый раз системой на доступность. Если наличие пользователя подтверждается, то программа переходит к процессу аутентификации.

Процесс аутентификации представляет собой процедуру проверки подлинности пользователя. Другими словами, пользователю необходимо доказать, что он именно тот человек, которому принадлежит идентификатор. В данном приложении в качестве проверяемого значения был использован пароль. Программа сравнивает пароль, введенный пользователем и пароль, хранящийся в базе данных, и в случае совпадения осуществляет процедура авторизации.

Авторизация – это предоставление доступа к какому-либо ресурсу, в данном случае к аккаунту пользователя. Этот процесс осуществляется после идентификации и аутентификации, которые предшествуют его выполнению.

Далее пользователь, попадая в систему, может выбрать на какую страницу ему направиться и начать работать, то есть заполнить поля для бронирования, либо перейти в корзину или историю.

На главной странице находятся поля, необходимые для заполнения. Необходимо заполнить эти поля:

- пункт отбытия;
- время отбытия;
- пункт прибытия;
- время прибытия;
- дата вылета.

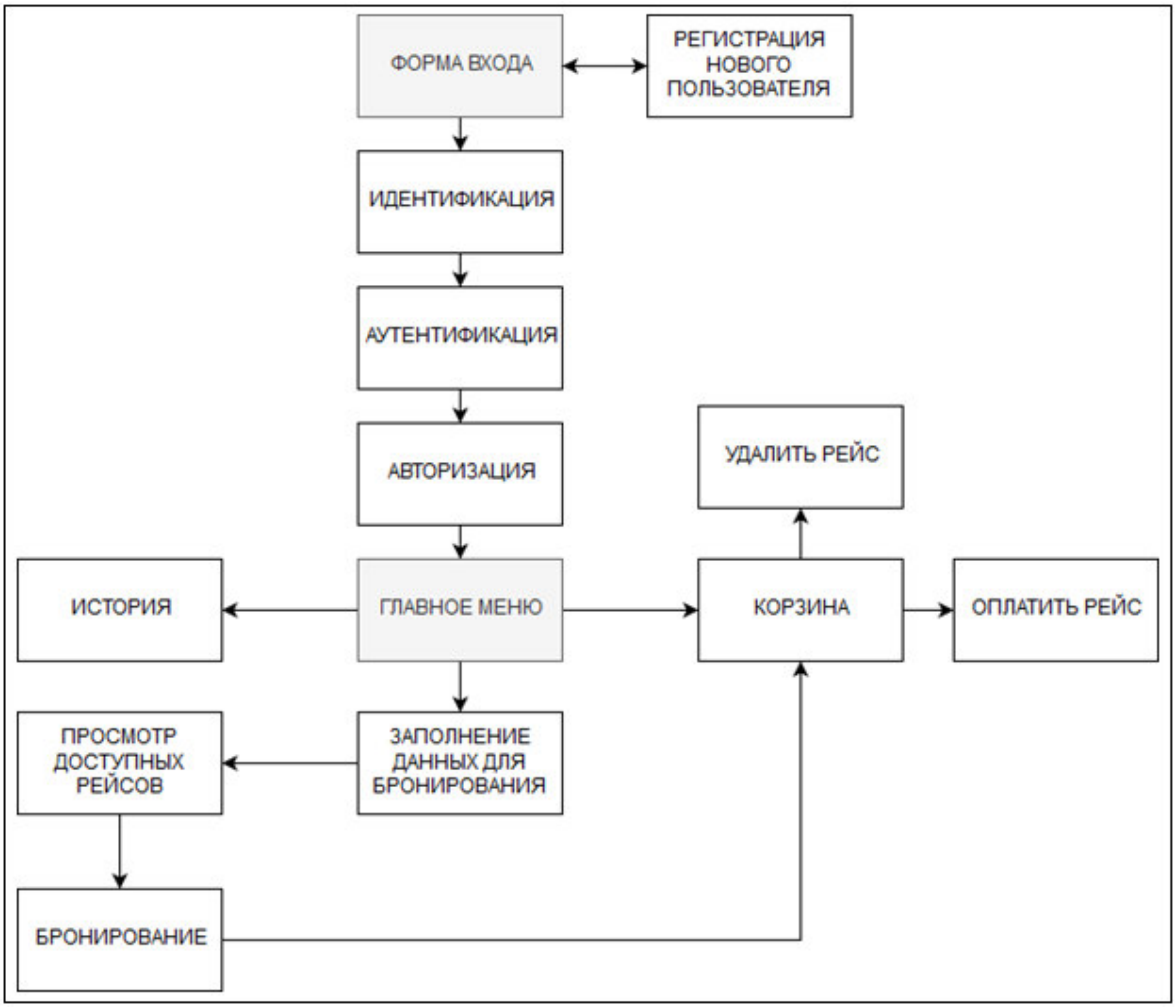


Рисунок 2.2.1 – Функциональная структура веб-приложения

После заполнения этих данных, пользователь переходит на следующую страницу, где выводится список всех доступных рейсов, соответствующих критериям поиска. После удовлетворения всех условий, пользователь добавляет этот рейс в свою корзину. Корзина преполагает собой временное хранение. После добавления рейса в корзину, пользователь оплачивает рейс и наслаждается скорой поездкой.

2.3 Проектирование базы данных

База данных – неотъемлимая часть в разработке приложения, в которой будет храниться информация, необходимая для корректной работы приложения и для его взаимодействия с пользователем.

Предполагается, что для базы данных всегда должен быть отведен достаточно большой объем памяти компьютера (или сервера). Поскольку одновременно база данных должна иметь доступ к огромнейшему количеству информации.

Проектирование базы данных — это всегда достаточно сложный процесс, который требует не только внимательности разработчика, но и проявление его умственных способностей. От того насколько точно будет продумана логика приложения, а затем и перенесена в таблицы базы данных, зависит насколько просто потом будет разработчику получать данные из этой таблицы.

При разработке предлагаемого приложения возникла необходимость создать свою базу данных, поскольку пользователю необходимо хранить данные, загружаемые в приложение и после его выхода из аккаунта, и отключения браузера. База данных предназначается для того, чтобы хранить в ней все данные пользователя, рейсов, аэропортов, моделей самолетов, вместимости самолета и наличия классов в каждом рейсе.

Сначала может показаться, что нужно разрабатывать сложную базу данных, с большим количеством таблиц и связей в ней. Но на самом деле при правильном подходе к построению таблиц и продумывании для чего нам нужны те или данные, выясняется, что данные можно хранить в более структурированном виде.

Именно этот структурированный вид и логическая продуманность базы позволяет ускорить процесс выполнения запросов в базу данных. А также иметь доступ к конкретному сегменту СУБД, а не искать нужную запись в хаотичном потоке информации, как это было бы, если бы данные хранились в обычном файле.

Также необходимо хранить в базе данных лишь те значения, которые действительно нужны для работы с приложением. Лишние значения, которые в дальнейшем в базе данных не используются, лишь нагружают ее и не дают в полной мере ощутить преимущества скорости работы современных СУБД. К тому же все СУБД обладают функцией, позволяющей нам в любой момент времени добавить нужные поля в любую из таблиц базы данных. Так что данное свойство позволяет разработчику не создавать поля в структуру таблицы впрок.

Также свойство транзакционности при пользовании базами данных позволяет избежать утечки данных из базы данных. Все запросы, отправляемые в базу данных, либо будут выполнены, либо нет, но данные при этом не исчезнут никуда, как это могло бы произойти, если бы для хранения данных приложения использовались обычные текстовые файлы.

В современном проектировании баз данных существует две основные технологии, согласно которым проектируется большинство актуальных на

сегодняшний день баз данных: SQL и NoSQL, реляционные и нереляционные базы данных.

Различия этих технологий заключаются в том, какой принцип используется в каждой технологии для проектирования базы данных, какие типы информации поддерживают, как хранят информацию и разница в функционале, предоставляемая базами данных.

Реляционные базы отличаются тем, что хранят в себе описание каких-либо конкретных объектов. Это могут сведения о человеке, сотруднике, описание содержимого товарной корзины в интернет-магазине и т. д. Все эти данные хранятся в таблицах, сгруппированных по типу объектов. Формат таблиц для хранения объектов обычно задается на этапе проектирования базы данных.

Нереляционные базы данных работают по иному принципу. То, что в реляционных базах данных разбито на взаимосвязи нескольких таблиц, в нереляционной базе данных те же самые значения могут храниться в виде целостной самостоятельной сущности.

Каждая из множества систем управления базами данных имеет собственное внутренне устройство, которое влияет на методы работы с ней. Например, нереляционные базы данных лучше поддаются процессу масштабирования, в то время как реляционные базы данных больше подходят для хранения уже структурированной информации.

Конечный выбор той или другой системы управления зависит от того, какие особенности имеет проект, для которого, собственно, и проектируется хранилище данных. На данных момент в большинстве крупных проектов одновременно используют оба типа систем управления базами данных.

Для приложения, разрабатываемого в рамках дипломного проекта, будет достаточно использования только реляционной базы данных, поскольку все данные, которыми оперируют компоненты приложения могут быть преобразованы в несколько взаимосвязанных таблиц.

В данном случае структура базы данных составляется на этапе проектирования базы данных и мало подвержена дальнейшим изменениям. Поскольку приложения предоставляет конкретный неизменяющийся со временем функционал, то нет необходимости вольно обращаться с потоком предоставляемой информации и все время подстраивать его под определенные моменты.

Система управления базами данных — это совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

СУБД — это комплекс программ, позволяющих создать базу данных и манипулировать данными (вставлять, обновлять, удалять и выбирать). Система обеспечивает безопасность, надёжность хранения и целостность данных, а также предоставляет средства для администрирования БД.

Для проектирования базы данных в дипломном проекте была выбрана СУБД MySQL. MySQL - это бесплатная база данных с открытым исходным

кодом, которая облегчает эффективное управление базами данных, подключая их к программному обеспечению. Это стабильное, надежное и мощное решение с расширенными функциями и преимуществами.

Безопасность данных. Будучи более безопасной и надежной системой управления базами данных, MySQL известен во всем мире и используется в популярных веб-приложениях, таких как WordPress, Drupal, Joomla, Facebook и Twitter. Безопасность данных и поддержка обработки транзакций, которые сопровождают самую последнюю версию MySQL, могут принести большую пользу любому бизнесу, особенно если это бизнес электронной коммерции, который включает частые денежные переводы.

Управляемое масштабирование. MySQL предлагает непревзойденную масштабируемость, чтобы упростить управление глубоко внедренными приложениями, используя меньшие по размеру следы даже в больших хранилищах, в которых хранятся терабайты данных. Главная особенность MySQL - гибкость по требованию. Это решение с открытым исходным кодом позволяет полностью настроить предприятия электронной коммерции с уникальными требованиями к серверу баз данных.

Высокая производительность. Отличительная особенность механизма хранения MySQL позволяет системным администраторам настраивать сервер базы данных MySQL для безупречной производительности. Являясь веб-сайтом электронной коммерции, который получает миллион запросов каждый день, или высокоскоростной системой транзакционной обработки, MySQL разработан для удовлетворения даже самых требовательных приложений, обеспечивая оптимальную скорость, полнотекстовый индекс и уникальные кэш-памяти для повышения производительности.

Круглосуточность. MySQL предоставляет полную гарантию бесперебойной работы 24X7 и предлагает широкий спектр решений высокой доступности, таких как специализированные кластерные серверы и конфигурации репликации master / slave.

Комплексная поддержка транзакций. MySQL занимает первое место в списке надежных механизмов транзакционных баз данных, доступных на рынке. Благодаря таким функциям, как полная автоматическая, согласованная, изолированная и надежная поддержка транзакций, поддержка нескольких версий транзакций и неограниченная блокировка на уровне строк, это решение для полного доступа целостность данных.

Это гарантирует мгновенную идентификацию тупика через принудительную ссылочную целостность сервера.

Полный контроль рабочего процесса. Поскольку среднее время загрузки и установки составляет менее 30 минут, MySQL означает удобство использования с самого первого дня. В качестве платформы используется Linux, Microsoft, Macintosh или UNIX, MySQL - это комплексное решение с функциями самостоятельного управления, которое автоматизирует все, начиная с расширения пространства. и конфигурация для проектирования данных и администрирования данных.

Решения, которые были разработаны для работы с данными в этой СУБД, гораздо позднее появились в аналогичных системах управления. К тому же данная система является системой с открытым исходным кодом, а это в первую очередь влияет на скорость ее развития, так как чем больше человек задействовано в разработке ПО, тем более неожиданные решения могут появиться в перечне функциональных возможностей системы.

Данная СУБД поддерживает большую часть официального стандарта SQL.

По мимо этого здесь реализован ряд важных современных функций для работы с базами данных: внешние ключи, сложные запросы, изменяемые представления, триггеры и т. п.

По истине уникальными функциями в работе с MySQL является возможность создавать собственные элементы для работы с данными. Например, можно создать собственную функцию, оператор, агрегатные функции, методы индексирования, и даже собственные типы данных.

MySQL имеет клиент-серверную архитектуру. Таким образом, есть главный серверный процесс, именуемый postgres. Он принимает запросы от клиента и оперирует с файлами базы данных. Этот серверный процесс может работать как локально на клиентской машине, так и находясь на сервере. Одновременно серверный процесс может принимать подключения сразу от нескольких клиентов и работать с ними. Он все время находится в ожидании подключения клиентов.

В качестве клиента используется специальное клиентское приложение, которое имеет возможность выполнять запросы в базе данных. Клиентские приложения очень разнообразны: различные текстовые утилиты, графические приложения, либо другой специализированный инструмент для работы с базой данных.

Использование подобной СУБД является мощным заделом к будущему развитию программного продукта. Гораздо проще сразу создать хранилище данных с использованием современного инструмента, который в будущем сможет удовлетворить все необходимые потребности, нежели переводить проект с одной СУБД на другую.

Перевод проекта на новую СУБД процесс сложный и дорогой. Может возникнуть проблема утечки информации или конфликта типов данных, что усложнит этот процесс.

К тому же СУБД MySQL активно поддерживается, дорабатывается, совершенствуется и обновляется по сегодняшний день многими разработчиками, так как находится в открытом доступе.

После выбора СУБД можно переходить к анализу предметной области и проектированию схемы базы данных.

Анализ предметной области представляет собой один из наиболее важных этапов в разработке приложения. Этот этап позволяет выявить все данные, которые потребуются для корректного взаимодействия пользователя

с приложением, а также учтут какие данные пользователь захочет хранить в базе данных.

Для того, чтобы не забыть добавить данные из какой-либо области приложения, необходимо поэтапно переходить к каждому функционалу приложения от начала взаимодействия с пользователем и анализировать, какие данные окажутся востребованными для корректной реализации данного функционала.

Первое взаимодействие с системой. Вход в приложение. Для того, чтобы пользователь мог осуществить вход в систему ему необходимо иметь собственный идентификатор и пароль. В качестве идентификатора может выступать какой-либо уникальный номер, имя пользователя, или, например, логин.

В данном случае мы будем использовать логин, потому что он уникален для каждого пользователя.

Пароль – какой-либо набор символов, придуманный пользователем. Условиями для создания пароля служат наличие как минимум: строчных и прописных букв, цифр и шести символов. Также стоит отметить, что пароль в базе данных хранится в зашифрованном виде.

В том случае, если пользователь не зарегистрирован, то от формы входа он направится к форме регистрации. Здесь уже будут использоваться два ранее рассмотренных свойства: логин, который служит идентификатором пользователя и пароль от аккаунта пользователя. Также здесь необходимо будет добавить поля имени и фамилии пользователя, дата рождения, e-mail и номер телефона.

В итоге подошел момент к созданию первой таблицы базы данных с названием «Пассажир» и в БД носит название passanger. Эта таблица будет хранить в себе все данные относительно каждого пользователя в полях, названия которых были перечислены выше: логин, пароль, дата рождения, ФИО, e-mail, номер телефона.

Запрос на создание таблицы ‘в базе данных выглядит следующим образом:

```
CREATE TABLE `passanger` (
  `username` varchar(30) NOT NULL,
  `firstname` varchar(45) DEFAULT NULL,
  `patronymic` varchar(45) DEFAULT NULL,
  `lastname` varchar(45) DEFAULT NULL,
  `email` varchar(45) DEFAULT NULL,
  `cellphone` varchar(15) DEFAULT NULL,
  `gender` varchar(10) DEFAULT NULL,
  `birthday` date DEFAULT NULL,
  `password` varchar(45) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Рисунок 2.3.1 – Запрос для создания таблицы Passanger

Для поля даты рождения используется тип date default null, для остальных полей varchar.

Результат запроса на все данные из таблицы, которая уже заполнена тестовыми значениями изображена на рисунке 2.3.2:

	username	firstname	middlename	lastname	email	cellphone	gender	birthday	password
1								<null>	
2	allore	Sultan	Malikovich	Issambergenov	sultan_97.28@mail.ru	87015275547	Male	1997-12-28	1937284600MCSultan
3	angelcramer@gmail.com	Angel	<null>	Cramer	angelcramer@gmail.com	84232919875	F	1985-07-11	123456789ABCabc
4	caballerouna	Una	<null>	Caballero	unacab@gmail.com	86015757957	<null>	<null>	asdadQGH111
5	enrique	Enrique		Iglesias	enrique-iglesias@gmail.com	88005553535	Male	1975-05-08	1937284600MCEnriqu
6	eugenemiller	Eugene	<null>	Miller	milereugene@gmail.com	83343379236	M	1990-05-06	1122334455ABCabc
7	fenwickjohn	John	<null>	Fenwick	fenwickjohn222@gmail.com	89138393108	M	1970-03-12	QWERTYqwerty123
8	isambergenov	Sultan		Issambergenov	sultan-isambergenov@gmail.com	87015275547	Male	1997-12-28	1937284600MCSultan
9	Michael	Michael		Jackson	michael-jackson@gmail.com	83123554224	Male	1990-01-01	12345Aa
10	roxannejohnson	Roxanne		Johnson	roxannejohnson@gmail.com	83205840924	F	1995-02-28	asfsdafsdf123AS
11	sultan	Sultan	Malikovich	Issambergenov	sultan_97.28@mail.ru	87015275547	Male	1997-12-28	1937284600MCSultan
12	s_isamberg	Sultan		Issambergenov	s_isamberg@gmail.com	87015275547	Male	1997-12-28	123456789Aa
13	UshakovSolomon318	Solomon	Vasilievich	Ushakov	ushakovsolomon318@mail.ru	89648013099	Male	1983-04-12	nr9NG9Rzjcbm
14	ZaharovaIrina394	Irina	Alexandrovna	Zaharova	zaharova11@list.ru	89787042222	Female	1978-01-23	MCJ7weeg1ndc

Рисунок 2.3.2 – Данные таблицы Passanger

После входа в систему пользователь может забронировать себе рейс, заполнив поля, необходимые для составления списка доступных рейсов.

Для того, чтобы сохранить данные в базе данных, имеются обязательные условия, без которых регистрация пользователя невозможна, а это: логин, пароль, имя и фамилия пользователя, дата рождения, электронный адрес и номер телефона.

К необязательным относятся указывание отчества и пола пользователя, не несущие особой смысловой нагрузки.

Поле, хранящее в себе логин, предназначено для того, чтобы бронирование оформлялось именно на этого пользователя, а не на кого-то другого.

Далее создаем таблицу с названиями аэропортов. Эта таблица будет хранить в себе все данные относительно каждого аэропорта, находящегося на территории Республики Казахстан.

```
CREATE TABLE `airport` (
  `code` varchar(10) NOT NULL,
  `name` varchar(50) NOT NULL,
  `city` varchar(20) NOT NULL,
  `region` varchar(20) NOT NULL,
  `country` varchar(20) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Рисунок 2.3.3 – Запрос для создания таблицы Airport

Таблица Аэропорт. В этой таблице хранятся такие данные, как: код аэропорта, название аэропорта, название города, область и страна, в которой находится аэропорт.

Данные этой таблицы будут использоваться при составлении списка рейсов администратором, указывая при этом идентификаторы в качестве кода аэропорта.

Результат запроса на все данные из таблицы, которая уже заполнена тестовыми значениями будет выглядеть следующим образом:

	code	name	city	region	country
1	AGZ	Международный аэропорт Аягуз	Аягуз	ВКО	Казахстан
2	AKX	Международный аэропорт Актобе	Актюбинск	Актюбинская	Казахстан
3	ALA	Международный аэропорт Алматы	Алма-Ата	Алматинская	Казахстан
4	ATX	Международный аэропорт Атбасар	Атбасар	Акмолинская	Казахстан
5	AYK	Международный аэропорт Аркалык	Аркалык	Костанайская	Казахстан
6	BXH	Международный аэропорт Балхаш	Балхаш	Карагандинская	Казахстан
7	BXJ	Международный аэропорт Боралдай	Боралдай	Алматинская	Казахстан
8	CIT	Международный аэропорт Шымкент	Шымкент	Туркестанская	Казахстан
9	DMB	Международный аэропорт Аулие-Ата	Тараз	Жамбылская	Казахстан
10	DZN	Международный аэропорт Жезказган	Жезказган	Карагандинская	Казахстан
11	EKB	Международный аэропорт Экибастуз	Экибастуз	Павлодарская	Казахстан
12	GUW	Международный аэропорт Атырау	Атырау	Атырауская	Казахстан
13	KGF	Международный аэропорт Сары-Арка	Караганда	Карагандинская	Казахстан
14	KOV	Международный аэропорт Кокшетау	Кокшетау	Акмолинская	Казахстан
15	KSN	Международный аэропорт Наримановка	Костанай	Костанайская	Казахстан
16	KZO	Международный аэропорт Кызылорда	Кызылорда	Кызылординская	Казахстан
17	PLX	Международный аэропорт Семей	Семей	ВКО	Казахстан
18	PPK	Международный аэропорт Петропавловск	Петропавлоск	СКО	Казахстан
19	PWQ	Международный аэропорт Павлодар	Павлодар	Павлодарская	Казахстан
20	SCO	Международный аэропорт Актау	Актау	Мангистауская	Казахстан
21	TDK	Международный аэропорт Талдыкорган	Талдыкорган	Алматинская	Казахстан
22	TSE	Международный аэропорт Нур-Султан	Нур-Султан	Акмолинская	Казахстан
23	UKK	Международный аэропорт Усть-Каменогорск	Усть-Каменогорск	ВКО	Казахстан
24	URA	Международный аэропорт Уральск	Уральск	ЗКО	Казахстан

Рисунок 2.3.4 – Данные таблицы Airport

Следующей таблицей является «Самолет». Данная таблица будет хранить в себе идентификационный номер, компанию, к которой принадлежит воздушное судно и модель самолета. На рисунке 2.3.5 показан запрос для создания этой таблицы.

```
CREATE TABLE `airplane` (
  `ID` varchar(10) NOT NULL,
  `type` varchar(10) NOT NULL,
  `company` varchar(20) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Рисунок 2.3.5 – Запрос для создания таблицы Airplane

Данная таблица носит лишь информационный характер, который также будет использоваться при создании списка рейсов.

Таблица «Перелеты». Данная таблица предназначена для составления рейсов и также доступна в панели администратора. Она содержит название рейса, пункт и время отбытия, пункт и время прибытия, указанные в таблице «Аэропорты» и идентификационный номер самолета из таблицы «Самолет». Связав эти таблицы, мы получим полноценный рейс со всеми данными. Но кроме этого, для каждого рейса существует два типа класса: эконом и бизнес класс.

Таблица «Класс» представляет собой информацию, содержащую название рейса из таблицы «Перелеты», название класса, максимальную вместимость и цену.

Запрос на создание таблицы в базе данных выглядит следующим образом:

```
CREATE TABLE `class` (
  `number` varchar(10) NOT NULL,
  `name` varchar(20) NOT NULL,
  `capacity` int(11) NOT NULL,
  `price` float NOT NULL,
  PRIMARY KEY (`number`,`name`),
  KEY `number_idx` (`number`),
  CONSTRAINT `number` FOREIGN KEY (`number`) REFERENCES `flight` (`number`)
      ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Рисунок 2.3.6 – Запрос для создания таблицы Class

Результат запроса на все данные из таблицы, которая уже заполнена тестовыми значениями выглядит следующим образом:

	number	name	capacity	price
1	AA100	Business	20	75000
2	AA100	Economy	150	35000
3	AA1000	Business	20	81000
4	AA1000	Economy	220	24000
5	AA101	Business	15	65000
6	AA101	Economy	150	28500
7	AA102	Business	20	83000
8	AA102	Economy	100	19000
9	AA103	Business	30	77000
10	AA103	Economy	200	23000
11	AA104	Business	5	44500
12	AA104	Economy	150	23000
13	AA105	Business	20	52000
14	AA105	Economy	200	30500
15	AA106	Economy	150	28000
16	AA107	Economy	150	30500
17	AA108	Economy	150	22500
18	AA109	Economy	150	21500
19	AA110	Economy	150	24500
20	AA111	Economy	150	25500

Рисунок 2.3.7 – Данные таблицы Class

Следующей и последней таблицей в БД будет таблица «Бронирование». Данная таблица соединяет все данные воедино, позволяя пользователю завершить процесс резервирования.

Соединяется данными идентификационного номера самолета, логина пользователя, типа класса и номера рейса. Эта таблица состоит из идентификационного номера, времени и даты, номера рейса, логина пользователя, на которое оформляется бронирование, типа класса и статуса, позволяющего проверить, оплачен данный билет или нет. Данная таблица предназначена для хранения всех заказов.

Запрос для создания таблицы приведен на рисунке 2.3.8.

```
CREATE TABLE `book` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `time` datetime NOT NULL,
  `date` date NOT NULL,
  `flightno` varchar(10) NOT NULL,
  `username` varchar(45) NOT NULL,
  `classtype` varchar(20) NOT NULL,
  `paid` int(1) DEFAULT '0',
  PRIMARY KEY (`ID`,`flightno`),
  KEY `username_idx` (`username`),
  KEY `classname_idx` (`classtype`),
  KEY `flightno_idx` (`flightno`,`classtype`),
  CONSTRAINT `flightno` FOREIGN KEY (`flightno`, `classtype`)
     REFERENCES `class` (`number`, `name`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `username` FOREIGN KEY (`username`)
     REFERENCES `passanger` (`username`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=68 DEFAULT CHARSET=latin1;
```

Рисунок 2.3.8 – Запрос для создания таблицы Book

Эта таблица состоит из идентификационного номера, времени и даты, номера рейса, логина пользователя, на которое оформляется бронирование, типа класса и статуса, позволяющего проверить, оплачен данный билет или нет. Данная таблица предназначена для хранения всех заказов.

Далее представлена структура базы данных, которая показывает связи между таблицами в базе данных, и обобщенный вид всех таблиц, включенных в базу данных.

Создание таких таблиц позволяет наглядно продемонстрировать не только саму структуру базы данных, но и описывает все данные, которые в дальнейшем должны использоваться в приложении.

Каждая таблица на схеме выделена в отдельный прямоугольник, в котором указывается название таблицы и все ее поля. Ключевые поля в таблицах выделяются специальным символом слева от названия поля.

Также схема демонстрирует связь таблиц по конкретным полям этих таблиц. Эта связь демонстрирует взаимоотношения данных таблиц баз данных.

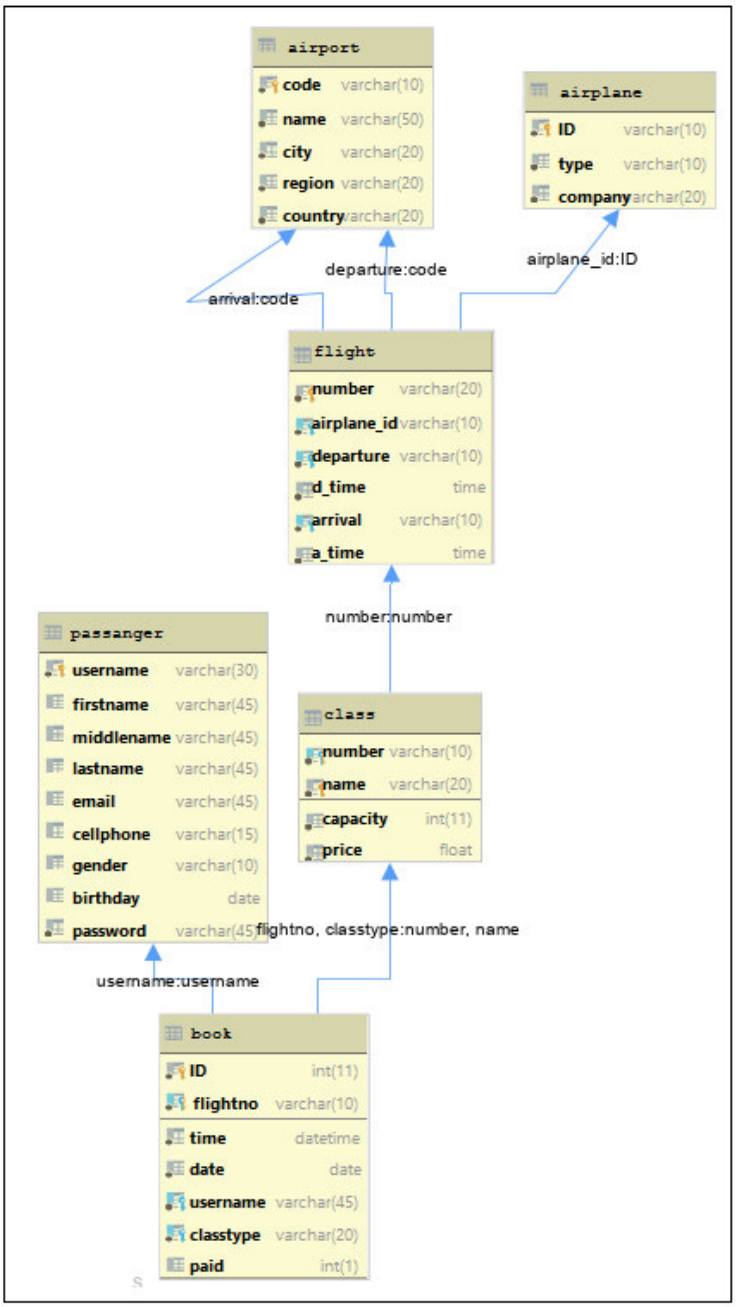


Рисунок 2.3.7 – Структура базы данных

2.4 Инструменты для работы с базой данных

В условиях разрабатываемого проекта непосредственная работа с базой данных происходила под руководством плагина Database Support для работы с базами данных в PyCharm от компании-разработчика JetBrains и редакторе SublimeText последней версии.

Плагин позволяет работать с базами данных напрямую из интегрированной среды разработки. Это одно из самых полезных свойств среды разработки, которое позволяет проводить разработку и отладку продукта с использованием только одного приложения.

Также стоит отметить, что данный плагин является универсальным средством для работы с базами данных. Интерфейс и внешний вид данного приложения абсолютно не зависит от типа базы данных, которую подключает разработчик. В случае первичного использования СУБД на данном компьютере плагин сам предложит уставить все необходимые для работы плагины, что очень упрощает задачу разработчику и сокращает время на поиск необходимой версии драйвера.

3 Разработка программного продукта

Клиент-серверное приложение представляет собой приложение, которое имеет две части: клиент и сервер. Такие приложения в основном отображаются и взаимодействуют с пользователем через веб-браузер.

К клиентской части относится та, которая визуально взаимодействует с пользователем. На этой стороне работают такие языки разметки, стилей и программирования как HTML, CSS и JavaScript.

Серверная часть приложения не имеет собственного визуального представления и взаимодействует с пользователем через веб-браузер. Название этой части вытекает из того, что все действия выполняются на сервере — специальном компьютере, который может быть расположен как за тысячи километров от браузера, так и в непосредственной близости, вплоть до одной машины.

На сервере обычно располагается база данных и оперируют такие языки как Java, PHP, C# и т. д. Данное приложение разрабатывается на языке программирования PHP, JS.

Для начала работы необходимо создать соответствующую директорию с проектом.

```
C:\Users\Sultan>mkdir C:\diploma\airline
```

Рисунок 3.1 – Создание директории проекта

После создания директории проекта, у нас в папке есть определенные модули, такие как static, scripts, fonts, img.

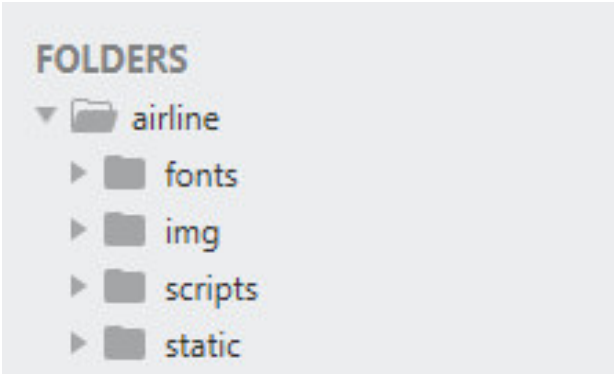


Рисунок 3.2 – Структура проекта airline

В папке static находятся все стили, применяемые к данному сайту. Каждый компонент отвечает за дизайн страниц администратора, пользователя.

Также различные стили применяются к полям ввода пароля, корзины и поиска.

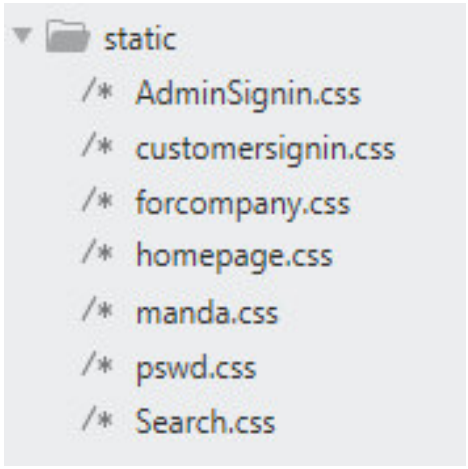


Рисунок 3.3 – Содержание модуля static

Стили используются для того, чтобы придать сайту более красочный вид, для определения верстки, дизайна и вариаций макета.

Модуль scripts, отвечающий некоторую часть функционала сайта:

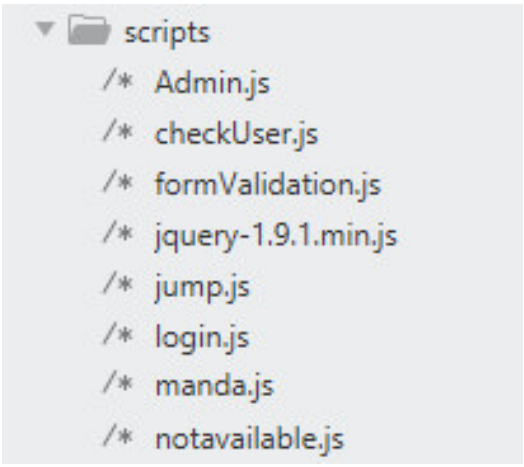


Рисунок 3.4 – Содержание модуля scripts

Данный раздел выполняет различные функции, которые будут описаны далее. В модулях fonts и img находятся шрифты и изображения, применяемые на сайте, соответственно.

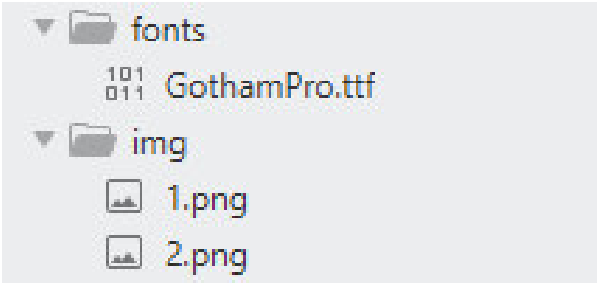


Рисунок 3.5 – Модули fonts и img

Теперь можно переходить к непосредственной разработке проекта. Рассмотрим элементы модуля scripts и какие функции они выполняют, ведь от них зависит основная работа, происходящая на сайте.

3.1 Элемент Admin.js

Admin.js позволяет менеджеру или администратору сайта добавлять, изменять или удалять рейсы. Далее разберем каждую функцию и его реализацию.

– добавление рейса;

```
$("#ad").click(function(){
var flightno = document.getElementById( elementId: "flightno2").value;
var airplaneid = document.getElementById( elementId: "airplaneid2").value;
var departure = document.getElementById( elementId: "departure2").value;
var dtime = document.getElementById( elementId: "dtime2").value;
var arrival = document.getElementById( elementId: "arrival2").value;
var atime = document.getElementById( elementId: "atime2").value;
var ec = document.getElementById( elementId: "ecapacity2").value;
var ep = document.getElementById( elementId: "eprice2").value;
var bc = document.getElementById( elementId: "bcapacity2").value;
var bp = document.getElementById( elementId: "bprice2").value;
xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
          if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
           var content = xmlhttp.responseText;
           if(content != "0")
           {
              alert(content);
            }
```

Рисунок 3.1.1 – Функция добавления рейса в Admin.js

Для добавления рейса, менеджеру необходимо заполнить поля значениями, описанными в функции: flightno, airplaneid, departure, dtime, arrival, atime, ecapacity, eprice, bcapacity, bprice. После заполнения отмеченных полей, необходимо добавить данные в базу данных. Для этого создается запрос, ссылающийся на Adminadd.php, в котором содержится отрывок SQL-кода:

```
$sql = "INSERT INTO flight VALUES( '$flightno', '$airplaneid', '$departure', '$dtime', '$arrival
if(! mysqli_query($conn, $sql))
{
    echo "Errormessage: ".mysqli_error($conn)."\n";
}
$sql = "INSERT INTO class VALUES( '$flightno', 'Economy', '$ec', '$ep')";
if(! mysqli_query($conn, $sql))
{
    echo "Errormessage: ".mysqli_error($conn)."\n";
}
$sql = "INSERT INTO class VALUES('$flightno', 'Business', '$bc', '$bp')";
if(! mysqli_query($conn, $sql))
{
    echo "Errormessage: ".mysqli_error($conn)."\n";
}
echo 0;
```

Рисунок 3.1.2 – SQL-запрос для добавления рейса в БД

```
xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
          if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
           var content = xmlhttp.responseText;
           if(content != "0")
           {
              alert(content);
            }
             else
             {
             alert("Рейс добавлен");
             $(':input','#addf')
    .not(':button, :submit, :reset, :hidden')
    .val( e: '');
             }
        }
    }
    xmlhttp.open( method: "GET", url: "Adminadd.php?flightno="+flightno+"&airplaneid="+
        airplaneid+"&departure="+departure+"&dtime="+dtime+"&arrival="+arrival+"&atime="+
        atime+"&ec="+ec+"&ep="+ep+"&bc="+bc+"&bp="+bp, async: true);
    xmlhttp.send();
```

Рисунок 3.1.3 – XHP-запрос на добавление рейса

Выполнив данные манипуляции, менеджер сможет добавить рейс в БД. Страница для добавления рейсов администратором показана на рисунке 3.1.4:

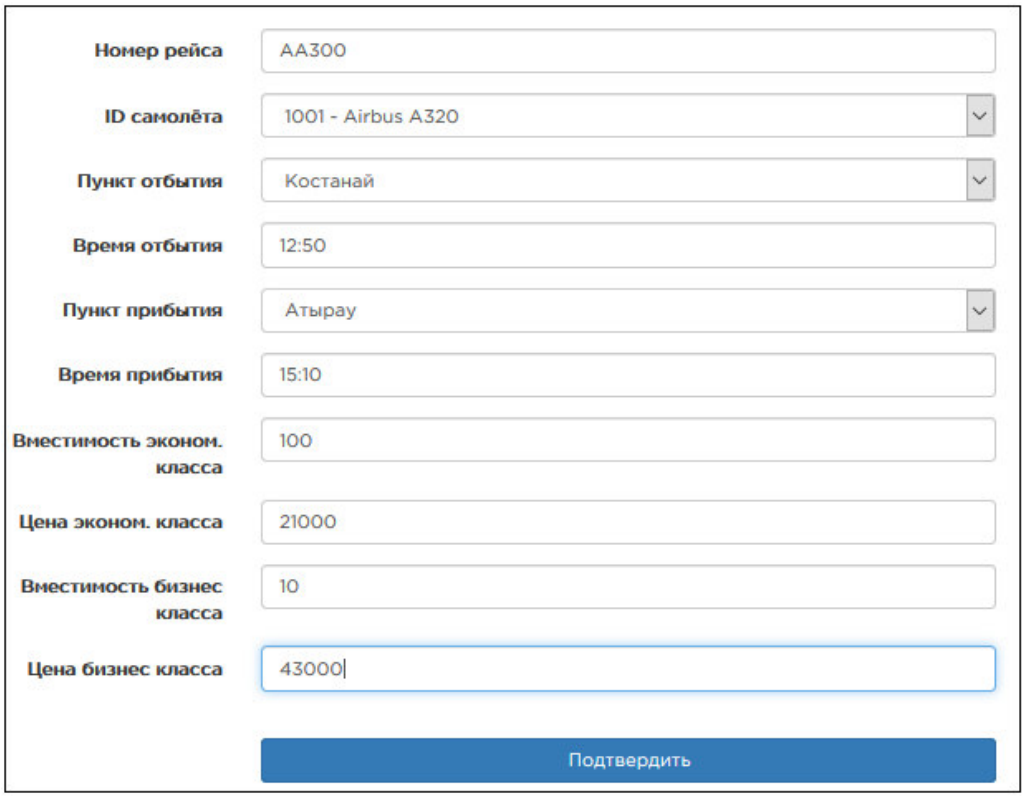


Рисунок 3.1.4 – Страница добавления рейса

После заполнения данных появляется соответствующее всплывающее окно, сообщающее нам о том, что рейс добавлен. С целью уточнить, добавлен данный рейс в таблицу или нет, выполним запрос SELECT *FROM FLIGHTS WHERE NUMBER = ‘AA300’;

	number	airplane_id	departure	d_time	arrival	a_time
1	AA300	1002	KSN	12:50:00	GUW	15:10:00

Рисунок 3.1.5 – Вывод данных таблицы Flight

Как можно заметить, рейс под названием AA215 из Павлодара в Костанай был успешно добавлен в нашу таблицу.

– обновление информации о рейсе;

```
//Обновление рейса
$("#up").click(function(){
var flightno = document.getElementById( elementId: "flightno1").value;
var airplaneid = document.getElementById( elementId: "airplaneid1").value;
var departure = document.getElementById( elementId: "departure1").value;
var dtime = document.getElementById( elementId: "dtime1").value;
var arrival = document.getElementById( elementId: "arrival1").value;
var atime = document.getElementById( elementId: "atime1").value;
var ec = document.getElementById( elementId: "ecapacity1").value;
var ep = document.getElementById( elementId: "eprice1").value;
var bc = document.getElementById( elementId: "bcapacity1").value;
var bp = document.getElementById( elementId: "bprice1").value;
```

Рисунок 3.1.6 – Функция обновления рейса в Admin.js

Для данной функции применяется соответствующий SQL-запрос, позволяющий обновить данные:

```
$sql = "UPDATE flight SET  airplane_id = '$airplaneid',
departure = '$departure', d_time = '$dtime', arrival = '$arrival',
a_time = '$atime' WHERE number = '$flightno'";
if(! mysqli_query($conn, $sql))
{
    echo "\nErrormessage: ".mysqli_error($conn)."\n";
}
$sql = "UPDATE class SET capacity = '$ec', price = '$ep'
WHERE number = '$flightno' AND name = 'Economy'";
if(! mysqli_query($conn, $sql))
{
    echo "\nErrormessage: ".mysqli_error($conn)."\n";
}
$sql = "UPDATE class SET capacity = '$bc', price = '$bp' WHERE number = '$flightno'
AND name = 'Business'";
if(! mysqli_query($conn, $sql))
{
    echo "Errormessage: ".mysqli_error($conn)."\n";
}
echo 0;
```

Рисунок 3.1.7 - SQL-запрос для обновления информации рейса

```
xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
          if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
           var content = xmlhttp.responseText;
           if(content != "0")
           {
              alert(content);
            }
             else
             {
             alert("Информация обновлена");
             $(':input','#result')
    .not(':button, :submit, :reset, :hidden')
    .val( e: '');
             }
        }
    }
    xmlhttp.open( method: "GET", url: "Adminupdate.php?flightno="+flightno+"&airplaneid="+
        airplaneid+"&departure="+departure+"&dtime="+dtime+"&arrival="+arrival+
        "&atime="+atime+"&ec="+ec+"&ep="+ep+"&bc="+bc+"&bp="+bp, async: true);
    xmlhttp.send();
```

Рисунок 3.1.8 - XHP-запрос на изменение информации о рейсе

Для обновления, изменения информации о рейсе, менеджеру необходимо лишь заполнить поле с названием рейса.

Рисунок 3.1.9 – Поле с номером рейса

После заполнения поля с номером рейса, происходит поиск данного рейса в базе данных, после чего остальные поля заполнятся автоматически.

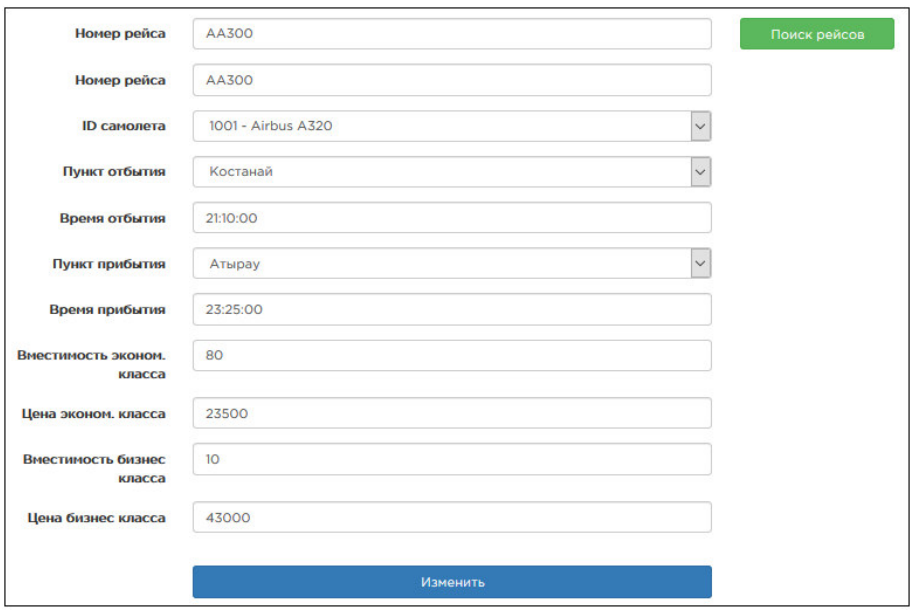


Рисунок 3.1.10 – Вывод автозаполненных данных

Менеджеру предоставляется возможность изменить информацию о рейсе согласно новоприобритенным данным. После изменения рейса, появляется соответствующее всплывающее окно, подтверждающее изменение рейса. Для того, чтобы проверить, что информация обновилась, выполним запрос на вывод данных таблицы Flight: SELECT *FROM FLIGHT WHERE NUMBER = ‘AA300’:

	number	airplane_id	departure	d_time	arrival	a_time
1	AA300	1002	KSN	21:10:00	GUW	23:25:00

Рисунок 3.1.11 – Вывод измененного рейса

Данные, указанные в этой таблице, отличаются от тех, что были ранее, что подтверждает изменение данных.

В случае, при поиске, если такого рейса в базе данных не существует, то появляется всплывающее окно:

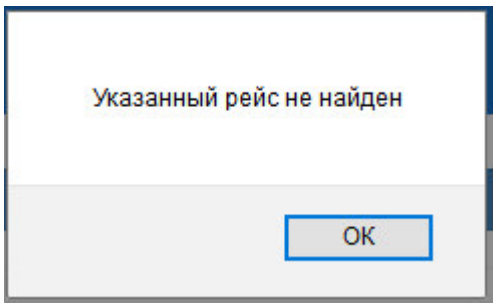


Рисунок 3.1.12 – Сообщение об ошибке

– удаление рейса;

```
$("#de").click(function(){
    var flightno = document.getElementById( elementId: "flightno1").value;
    xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
            if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
             var content = xmlhttp.responseText;

                 alert(content);
        }
    }
```

Рисунок 3.1.13 – Функция удаления рейса

Удаление рейса необходимо из-за дальнейшей ненадобности, либо ранее ошибочного введения в БД.

Запрос, позволяющий удалять рейсы, находится в Admindelete.php и выглядит следующим образом:

```
$sql = "DELETE FROM flight WHERE number = '$flightno'";
if(! mysqli_query($conn, $sql))
{

    echo "Errormessage: ".mysqli_error($conn)."\n";
}
else
{
    echo "Удалено";
}
```

Рисунок 3.1.14 - SQL-запрос для удаления рейса

Проверив тестовый рейс AA300 можно обнаружить, что данного рейса уже не существует и процесс удаления рейса прошел успешно.

После удаления рейса появляется всплывающее окно, сообщающее об успешном удалении рейса.

3.2 Элемент checkUser.js

Данный элемент позволяет проверять логин пользователя при регистрации на уникальность

```
$(document).ready(function() {
var checking_html='Идет проверка...';
$('#check_username_availability').click(function() {
$('#username_availability_result').html(checking_html);
check_availability();
});
});
function check_availability() {
var username=$('#username').val();
// use ajax to run the check
$.post("checkUser.php", { username : username },
function(result) {
if(result==1) {
$('#username_availability_result').html(username + ' уже существует');
} else {
$('#username_availability_result').html(username + ' доступен для пользования');
}
});
}
```

Рисунок 3.2.1 – Реализация функции checkUser.js

Далее приводится запрос, с помощью которого через существующую базу данных проверяется наличие логина пользователя в системе. Процедура выполняется нажатием на соответствующую кнопки “Проверить доступ”. Далее, появляются оповещения о статусе логина.

SQL-запрос, выполняемый в этой функции, выглядит следующим образом:

```
$result=mysqli_query($conn,"SELECT username FROM passanger WHERE username = '$username'");

    while($row = mysqli_fetch_array($result)) {
      if( $row['username']) {
          echo 1;
      }
      else {
          echo 0;
      }
    }
```

Рисунок 3.2.2 – SQL-запрос проверки логина на уникальность

Случай, когда логин свободен:

Рисунок 3.2.3 – Вывод доступности логина

В случае, если логин уже занят другим пользователем, появляется соответствующее сообщение:

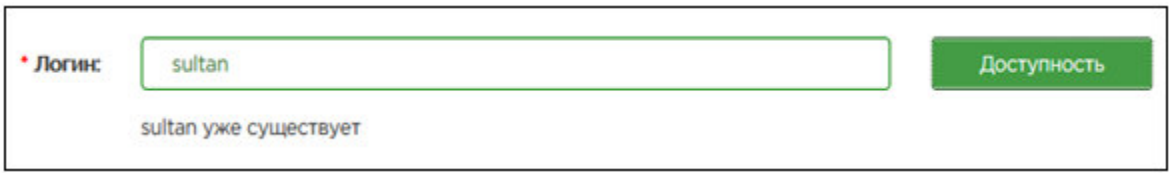


Рисунок 3.2.4 – Вывод занятости логина

Данная форма очень удобна, так как позволяет проверить доступность логина до завершения процесса регистрации и упрощает работу пользователя.

3.3 Элемент formValidation.js

Форма валидации необходима при регистрации чтобы учитывались необходимые паттерны и вводились корректные данные, ограничивался ввод неподходящих символов. Например, в поле, где нужно вводить только цифры, ввод букв будет автоматически игнорироваться без сообщения о какой-либо ошибке.

При введении неправильных данных, поле с вводом обозначается красным, что сообщает об его некорректности.

Поля, обозначенные символом '*', обязательны для заполнения и регистрация нового пользователя без них не состоится.

Регистрация нового пользователя в системе выглядит следующим образом:

На рисунке 3.3.1 показана форма регистрации нового пользователя.

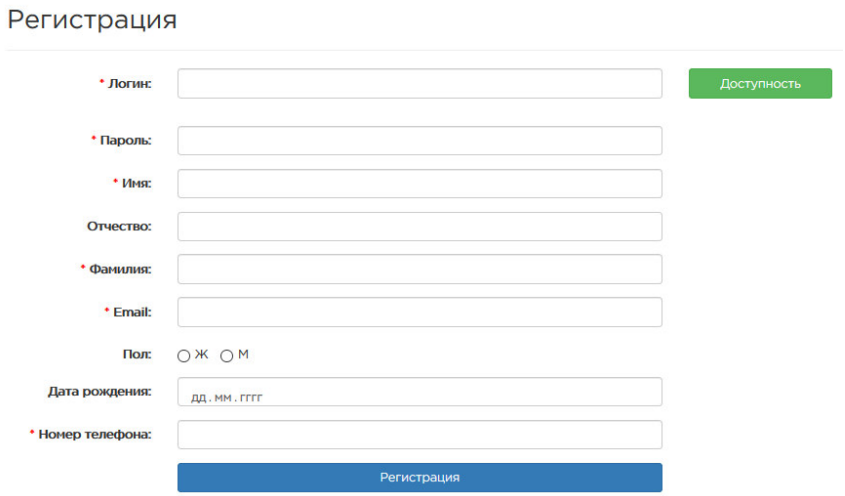


Рисунок 3.3.1 – Форма регистрации нового пользователя

Для каждого поля существуют свои условия валидации.

```
function validateForm() {
    var x = document.forms["myForm"]["username"].value;
    var valid = 1;
    if(x == null || x == "") {
        alert("Логин должен быть заполнен");
        valid = 0;
    }
```

Рисунок 3.3.2 – Валидация логина

Валидация логина является обязательной для заполнения, и имеет лишь одно требование: логин, который собирается быть зарегистрирован, не должен существовать в базе данных. Эта функция была реализована в checkUser.js.

```
var y = document.forms["myForm"]["pwd1"].value;
if(y == null || y == "") {
    alert("Пароль должен быть заполнен");
    valid = 0;
}

var regex = /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?!.*[^a-zA-Z0-9])(.{6,})$/;
if(!regex.test(y)){
        alert("Пароль не соответствует требованиям");
        valid = 0;
}
```

Рисунок 3.3.3 – Валидация и требования к паролю

Стоит подметить, что переменная regex это регулярное выражение, которое содержит необходимый набор символов, необходимый для корректного ввода пароля. Пароль должен содержать в себе как минимум: прописную и строчную буквы, число, и не менее шести символов.

К обязательным формам валидации также относятся: Имя, Фамилия, номер телефона, электронный адрес.

После завершения этапа регистрации, пользователь переходит обратно на главную страницу, где уже может забронировать себе билет с одной точки на другую.

Главная страница выглядит следующим образом:

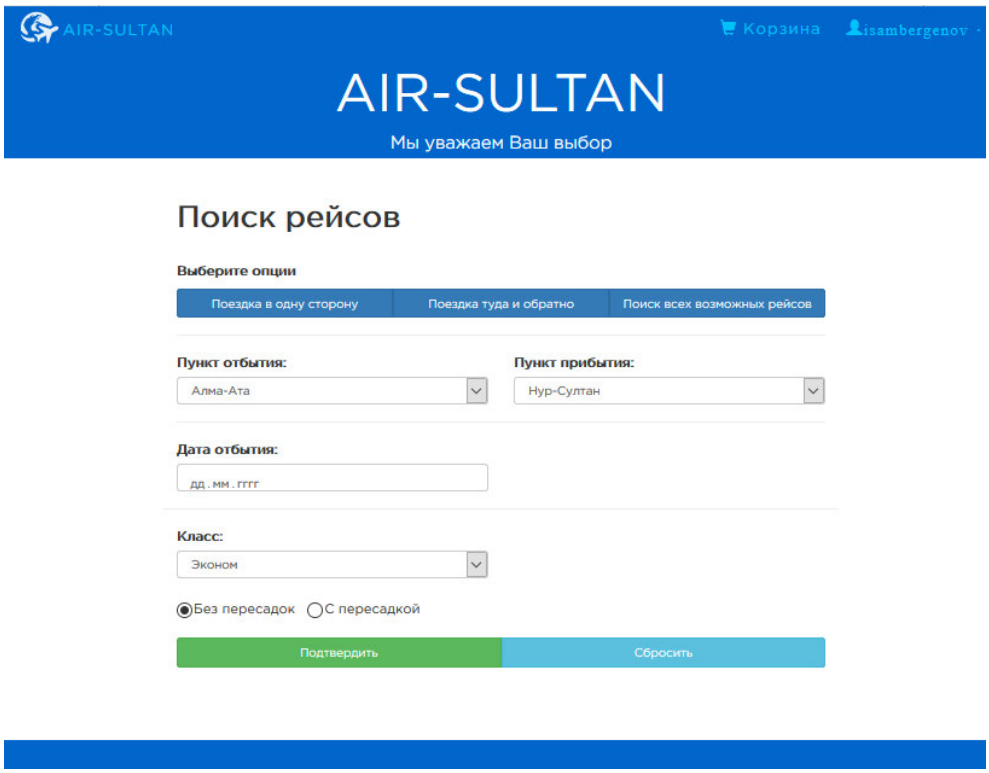


Рисунок 3.3.4 – Главная страница

На левом верхнем углу расположен наименование компании Air-Sultan.

На правом верхнем углу, для удобства пользователя размещены Корзина, и имя пользователя. Корзина предназначена для временного хранения информации о рейсах.

Ниже расположены основные поля для заполнения. Имеется два вида поиска рейсов: в одну сторону, и рейс туда и обратно.

3.4 Элемент login

Данный элемент позволяет пользователю зайти в свой ранее созданный аккаунт.

На рисунке 3.4.1 изображена форма входа.

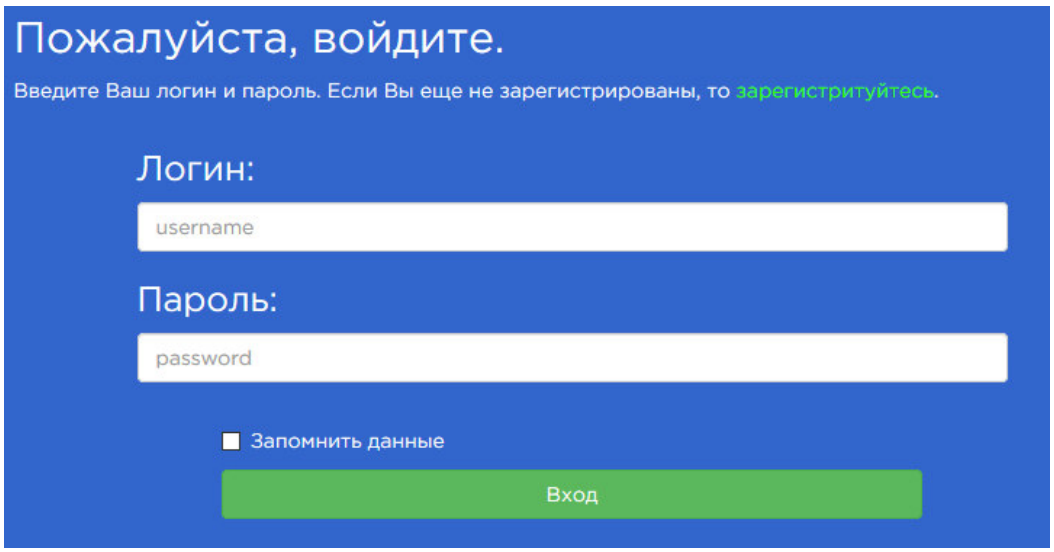


Рисунок 3.4.1 – Форма входа

При входе на сайт проверяются введенные логин и пароль пользователя. Для пользователей, не имеющих аккаунт, дается соответствующая ссылка.

```
$username=$_POST['username'];
$pwd=$_POST['pwd'];
$res=mysqli_query($conn,"SELECT * FROM passanger WHERE username='$username'");
$row=mysqli_fetch_array($res);
if($row['password']==$pwd)
{
    $_SESSION['user']=$row['username'];
    header("Location: homepage.html");
}
```

Рисунок 3.4.2 – Проверка на корректность введенных данных

3.5 Элементы поиска

В системе существуют три вида поиска рейсов. Первый вид поиска рейсов подразумевает собой поездку в одну сторону двумя различными способами: с пересадкой или без.

Ко второму виду поиска относится рейс типа туда и обратно и последний вид поиска рейса — это поиск всех возможных рейсов на определенную дату.

В первом случае, происходит поиск рейса из точки А в точку Б в определенную дату и время, затем создается второй запрос на поиск из точки Б в точку А. Дата возвращения также указываются пользователем. Тип класса выбирается из двух возможных в дроп-даун списке: Эконом-класс или Бизнес-класс.

На рисунке 3.5.1 проиллюстрирован пример заполнения полей, который необходим при поиске подходящих рейсов.

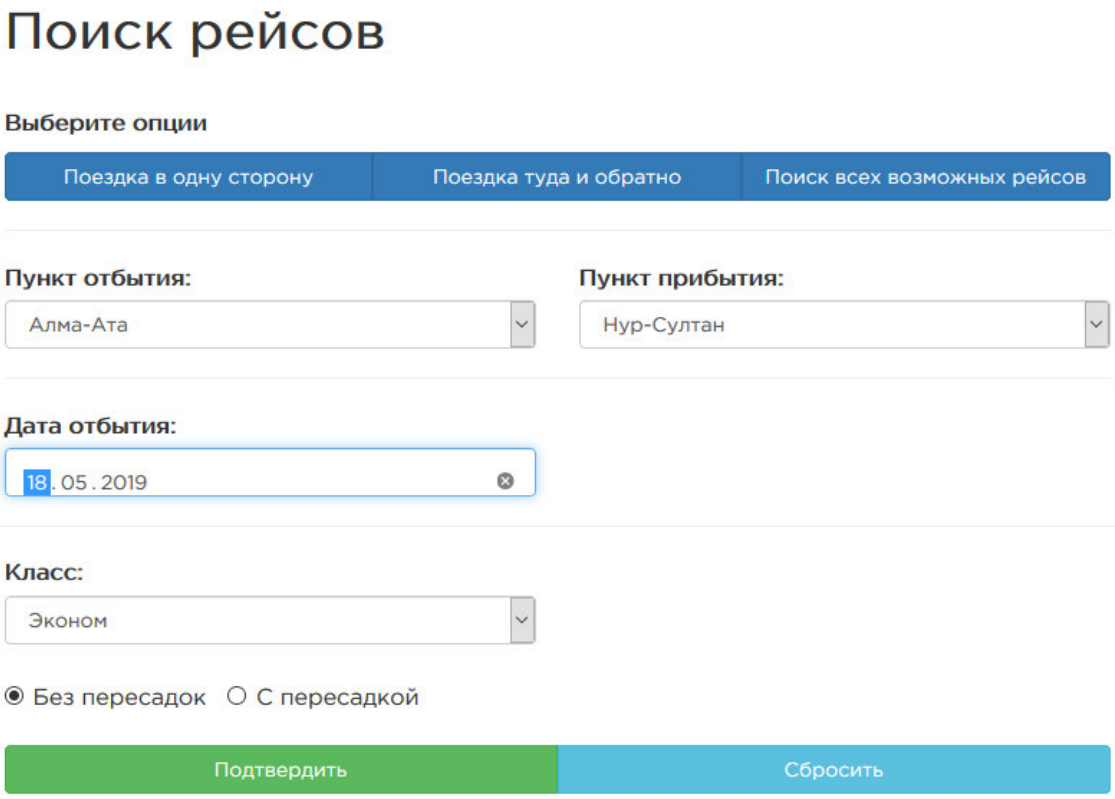


Рисунок 3.5.1 – Заполнение полей для поиска рейсов в одну сторону

В данном случае, находится рейс, находящийся в базе данных из одного пункта во второй, при его наличии. Функция, отвечающая за поиск рейсов в одну сторону, выглядит следующим образом:

```
if($stop=="nonstop"){
  $sql1 = "SELECT FL.number AS FLnumber, company, type, departure, d_time, arrival, a_time,
  C.name AS classname, capacity, price, COUNT(*)
          FROM flight FL,  class C, airplane AP , airport A
          WHERE (FL.number = C.number) AND (FL.airplane_id = AP.ID) AND C.name = '$class' AND
         ((((city LIKE '%$from%') AND (code = departure)) OR  ((city LIKE '%$to%')
        AND (code = arrival)))
        OR (((departure LIKE '%$from%') AND (arrival LIKE '%$to%'))) )
          GROUP BY FL.number
          HAVING COUNT(*)>1
          ORDER BY FL.number";
  $result = mysqli_query($con,$sql1);
```

Рисунок 3.5.2 – Функция поиска рейсов в одну сторону

После заполнения полей необходимыми данными, далее выводится таблица, которая содержит список доступных рейсов, соответствующих критериям поиска пользователя.

На рисунке 3.5.3 указываются необходимые детали и возможность добавить рейс в корзину.

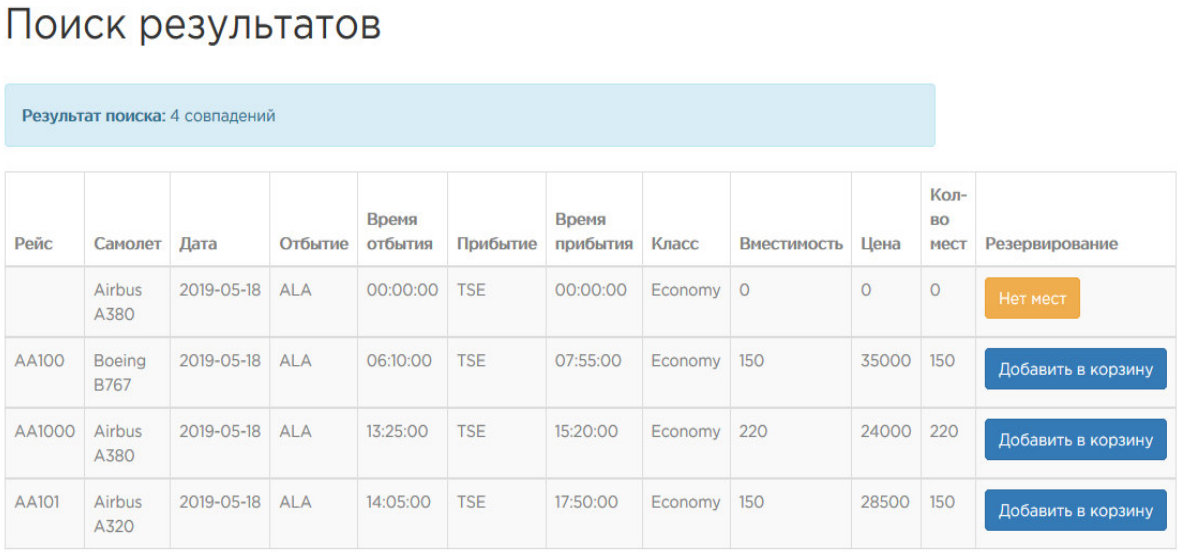


Рисунок 3.5.3 – Список рейсов в одну сторону

К данным, указываемым в таблице, относятся название рейса, модель самолета, дата, пункт и время отбытия и прибытия, цена, вместимость, тип класса и количество оставшихся мест. На рисунке 3.5.3 видно, что на один из рейсов мест не осталось, и соответственно появляется сообщение, оповещающее об отсутствии свободных мест. Для этого была реализована функция, позволяющая расчитывать количество оставшихся мест.

```
$seatreserved = "SELECT flightno, classtype, COUNT(*)
            FROM book B
            WHERE B.date = '".$departdate."' AND B.flightno = '".$row['FLnumber']."'
            AND B.classtype ='".$row['classname']."' AND paid=1
            GROUP BY flightno, classtype";
$reserved = mysqli_query($con, $seatreserved);
$reservedNumber = mysqli_fetch_array($reserved);

$capacity = mysqli_query($con, "SELECT capacity FROM class C
WHERE C.number='".$row['FLnumber']."' AND C.name= '".$row['classname']."'");
$capacityNumber = mysqli_fetch_array($capacity);


if(mysqli_num_rows($reserved)>0){
    $availableNumber = $capacityNumber['capacity'] - $reservedNumber['COUNT(*)'];
}else{
    $availableNumber = $capacityNumber['capacity'];
}
```

Рисунок 3.5.4 – Расчет оставшихся мест

После просмотра доступных рейсов, пользователь выбирает нужный рейс и добавляет его в корзину для дальнейшей обработки либо удаляет его из списка корзины.

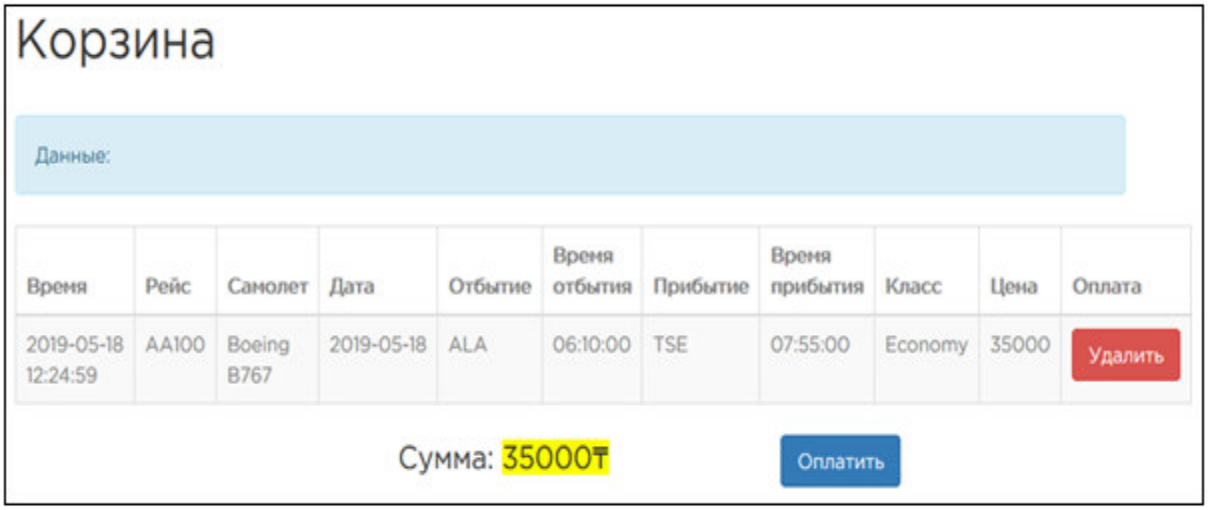


Рисунок 3.5.5 – Добавление рейса в корзину

Ниже приведен SQL-запрос, добавляющий детали рейса в корзину к определенному пользователю. При отсутствии элементов в корзине, появляется сообщение “Пусто”.

```
if(!isset($_SESSION['user'])){
    header("Location: customersignin.html");
}else{
    $user = $_SESSION['user'];


    if($type =="all" || $type =="onewaynonstop" ){

    $flightno = $_POST["flightno"];
    $class = $_POST["classtype"];
    $price = $_POST["price"];
    $date = $_POST["date"];

    $sql = "INSERT INTO book (time, date, flightno, username, classtype, paid)
            VALUES ('$time', '$date', '$flightno', '$user', '$class', '0')";;


    $result = mysqli_query($con,$sql);
    header("Location: cartshow.php");
    }
```

Рисунок 3.5.6 – Запрос добавления рейса в корзину

После добавления рейса в корзину, его можно либо удалить его из списка, либо же оплатить. Реализация удаления рейса показана ниже:

```php
<?php
session_start();
include_once 'dbconnect2.php';
if(!isset($_SESSION['user'])){
    header("Location: customersignin.html");
}else{
    $user = $_SESSION['user'];
    $bookid = $_POST["bookid"];

    $delete = "DELETE FROM book WHERE ID = '$bookid'";
    if(mysqli_query($con,$delete)){
        header("Location: cartshow.php");
    }else{
        echo "Error";
    }
}
?>
```

Рисунок 3.5.7 – Удаление рейса из корзины

В случае, если пользователю необходимо оплатить данный рейс, он переходит на следующую страницу и получает сообщение об успешной оплате и данный рейс записывается в историю бронирований пользователя.

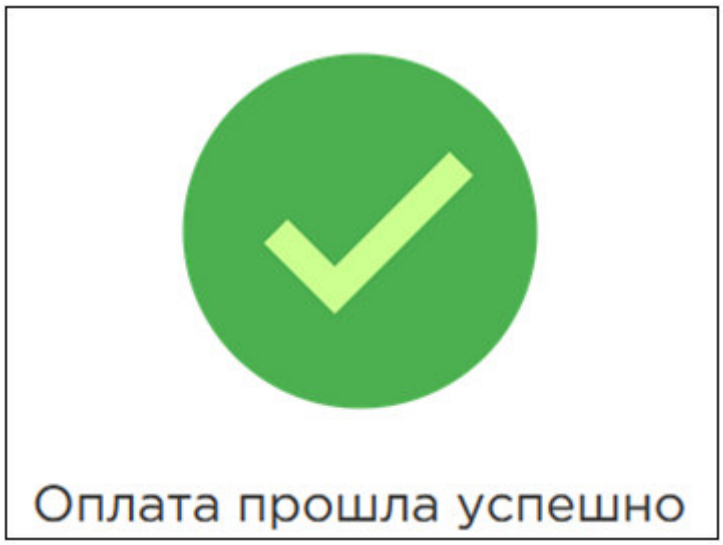


Рисунок 3.5.8 – Сообщение об успешной оплате

В истории бронирований сохраняются все рейсы, совершенные данным пользователем. Панель истории находится наверху около логина пользователя

История носит только информационный характер и не подлежит никаким модификациям.

Меню истории бронирований в навгиционной панели изображена на рисунке 3.5.9:

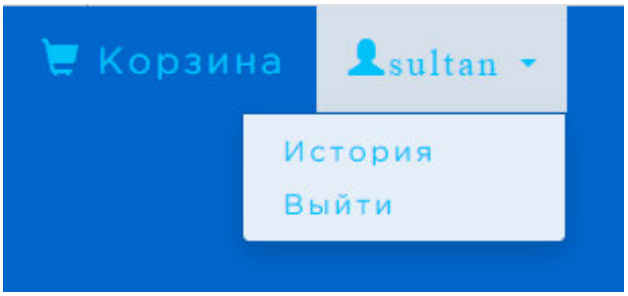
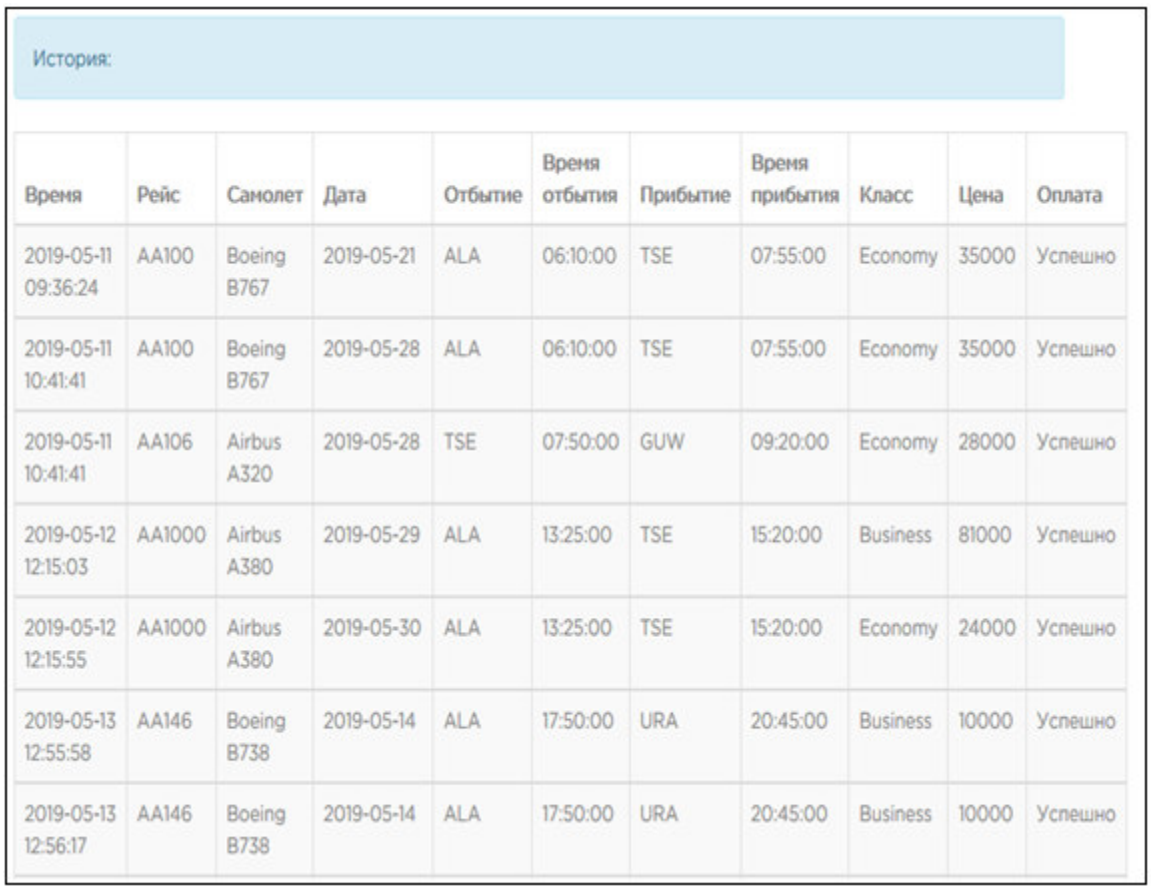


Рисунок 3.5.9 – Расположение Истории бронирований

В истории сохраняются все детали рейса, совершенные данным пользователем, также отмечается статус оплаты, и присваивается значение “Успешно”.



История:

Время	Рейс	Самолет	Дата	Отбытие	Время отбытия	Прибытие	Время прибытия	Класс	Цена	Оплата
2019-05-11 09:36:24	AA100	Boeing B767	2019-05-21	ALA	06:10:00	TSE	07:55:00	Economy	35000	Успешно
2019-05-11 10:41:41	AA100	Boeing B767	2019-05-28	ALA	06:10:00	TSE	07:55:00	Economy	35000	Успешно
2019-05-11 10:41:41	AA106	Airbus A320	2019-05-28	TSE	07:50:00	GUW	09:20:00	Economy	28000	Успешно
2019-05-12 12:15:03	AA1000	Airbus A380	2019-05-29	ALA	13:25:00	TSE	15:20:00	Business	81000	Успешно
2019-05-12 12:15:55	AA1000	Airbus A380	2019-05-30	ALA	13:25:00	TSE	15:20:00	Economy	24000	Успешно
2019-05-13 12:55:58	AA146	Boeing B738	2019-05-14	ALA	17:50:00	URA	20:45:00	Business	10000	Успешно
2019-05-13 12:56:17	AA146	Boeing B738	2019-05-14	ALA	17:50:00	URA	20:45:00	Business	10000	Успешно

Рисунок 3.5.10 – История бронирований пользователя

Для бронирований типа туда и обратно, используется другая форма заполнения.

Данный тип поездки подразумевает рейс из пункта А в пункт Б в дату А, и обратно из пункта Б в пункт А в дату Б. Система сопоставляет все

доступные рейсы, допустим, из Алма-Аты в Нур-Султан, находит их на указанную дату, и, соответственно, из Нур-Султана в Алма-Ату на дату возвращения.

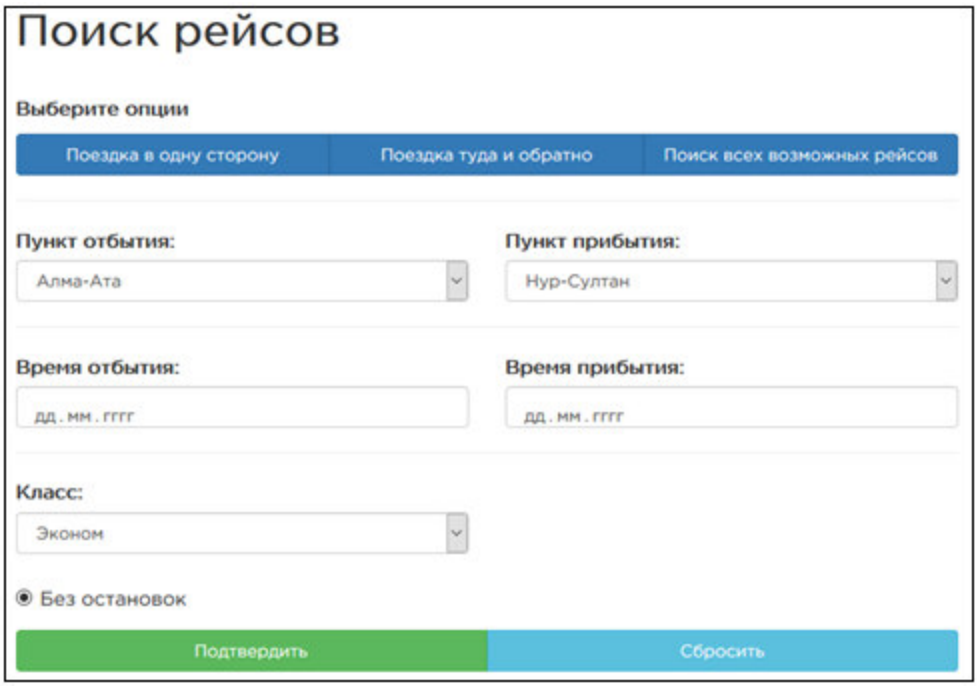


Рисунок 3.5.11 – Форма заполнения рейса “туда и обратно”

Ниже приведены запросы, позволяющие добавить детали рейса типа “туда и обратно” в корзину:

```
$flightno = $_POST["flightno"];
$class = $_POST["classtype"];
$price = $_POST["price"];
$date = $_POST["date"];


$flightno2 = $_POST["flightno2"];
$class2 = $_POST["classtype2"];
$price2 = $_POST["price2"];


$sql = "INSERT INTO book (time, date, flightno, username, classtype, paid)
        VALUES ('$time', '$date', '$flightno', '$user', '$class', '0')";;

$result = mysqli_query($con,$sql);

$sql2 = "INSERT INTO book (time, date, flightno, username, classtype, paid)
        VALUES ('$time', '$date', '$flightno2', '$user', '$class2', '0')";;

$result2 = mysqli_query($con,$sql2);
```

Рисунок 3.5.12 – Реализация поездки “туда и обратно”

Затем расчитываются свободные места для обоих рейсов:

```
$seatreserved = "SELECT flightno, classtype, COUNT(*)
            FROM book B
            WHERE B.date = '".$row['date']."' AND B.flightno = '".$row['FLnumber']."'
            AND B.classtype ='".$row['classname']."' AND paid=1
            GROUP BY flightno, classtype";
$reserved = mysqli_query($con, $seatreserved);
$reservedNumber = mysqli_fetch_array($reserved);

$capacity = mysqli_query($con, "SELECT capacity FROM class C
WHERE C.number='".$row['FLnumber']."' AND C.name= '".$row['classname']."'");
$capacityNumber = mysqli_fetch_array($capacity);
```

Рисунок 3.5.13 – Расчет свободных мест

Вывод в виде таблицы списка рейсов, реализация:

```
echo "<table class='table table-bordered table-striped table-hover'>
      <thead>
      <tr>
        <th>Время</th>
        <th>Рейс</th>
        <th>Самолет</th>
        <th>Дата</th>
        <th>Отбытие</th>
        <th>Время отбытия</th>
        <th>Прибытие</th>
        <th>Время прибытия</th>
        <th>Класс</th>
        <th>Цена</th>
        <th>Оплата</th>
while($row = mysqli_fetch_array($result)) {
    echo "<tbody><tr>";
    echo "<td>" . $row['time'] . "</td>";
    echo "<td>" . $row['FLnumber'] . "</td>";
    echo "<td>" . $row['company']." ".$row['type']. "</td>";
    echo "<td>" . $row['date'] . "</td>";
    echo "<td>" . $row['departure'] . "</td>";
    echo "<td>" . $row['d_time'] . "</td>";
    echo "<td>" . $row['arrival'] . "</td>";
```

Рисунок 3.5.14 – Реализация списка в виде таблицы

При отсутствии мест в одном из двух рейсов, появляется надлежащее сообщение, оповещающаее о том, что мест не осталось.

Поиск результатов

Совпадений: 8 результатов

Рейс	Самолет	Дата	Отбытие	Время отбытия	Прибытие	Время прибытия	Класс	Вместимость	Цена	Кол-во мест	Резервирование
	Airbus A380	2019-05-19	ALA	00:00:00	TSE	00:00:00	Economy	0	0	0	
AA102	Airbus A380	2019-05-28	TSE	09:55:00	ALA	11:35:00	Economy	100	19000	100	Недоступно
	Airbus A380	2019-05-19	ALA	00:00:00	TSE	00:00:00	Economy	0	0	0	
AA103	Embraer E190	2019-05-28	TSE	17:45:00	ALA	19:35:00	Economy	200	23000	200	Недоступно
AA100	Boeing B767	2019-05-19	ALA	06:10:00	TSE	07:55:00	Economy	150	35000	150	
AA102	Airbus A380	2019-05-28	TSE	09:55:00	ALA	11:35:00	Economy	100	19000	100	Добавить в корзину

Рисунок 3.5.15 – Вывод списка рейсов туда и обратно

Далее, выбранный рейс добавляется в корзину и оплачивается, то есть процедуры повторяются так же, как и при бронировании перелета типа в одну сторону.

После завершения сеанса, желательно выйти со своего аккаунта в целях безопасности.

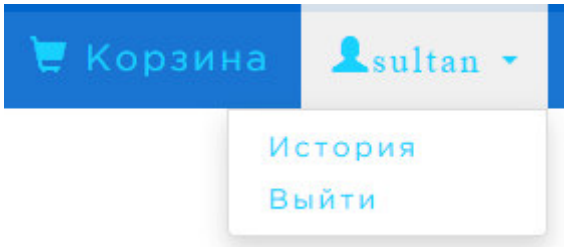


Рисунок 3.5.16 – Меню для выхода из аккаунта

Сессия завершается с помощью команды unset($_SESSION['user']);

4 Технико-экономическое обоснование

4.1 Описание работы и обоснование необходимости

Целью данного раздела является расчет затрат на создание программного продукта и расчет себестоимости прикладной программы.

Для определения себестоимости также необходимо учесть:

– трудоемкость разработки программного продукта;
– материальные затраты;
– затраты на оплату труда;
– социальный налог;
– амортизация основных фондов.

4.2 Расчет трудоемкости разработки программного продукта

Таблица 4.1 – Распределение работ по этапам и видам, оценка их трудоемкости

Этап разработки	Вид работы	Трудоемкость разработки, чел.×ч.
1	Формулировка задач	55
2	Составление алгоритма	120
3	Составление блок-схемы алгоритма	40
4	Разработка клиентской части проекта	150
5	Реализация администраторской части проекта	200
6	Тестирование программы	150
7	Оформление документации	60
ИТОГО		775

4.3 Расчет затрат на разработку программного продукта

Нахождение затрат на разработку ПП выполняется путем создания определенной сметы, которая включает следующие статьи:

– расход на материалы и оборудования;
– расход на заработную плату работникам;
– вычеты на социальные отчисления; (или просто "социальные отчисления")
– амортизационные затраты основных фондов.

Таблица 4.2 – Затраты на материальные ресурсы

Наименование материального ресурса	Единица измерения	Количество	Цена за единицу, тг	Сумма, тг
Ноутбук Acer Aspire 5 A515-52G-532C	шт.	1	115000	115000
Веб-сайт Air-Sultan	шт.	1	90000	90000
ИТОГО				205000

Общая сумма затрат на материальные ресурсы ($З_М$) определяется по формуле:

$$З_M = \sum_{i=1}^{n} P_i \times Ц_i , \quad (4.1)$$

где P_i – расход i-го вида материального ресурса, натуральные единицы;

$Ц_i$ – цена за единицу i-го вида материального ресурса, тг;

i – вид материального ресурса;

n – количество видов материальных ресурсов.

Если для разработки ПП используется электрооборудование, то необходимо рассчитать затраты на электроэнергию по форме, приведенной в таблице 4.3.

Общая сумма затрат на электроэнергию ($З_Э$) рассчитывается по формуле:

$$З_Э = \sum_{i=1}^{n} M_i \times K_i \times T_i \times Ц , \quad (4.2)$$

где M_i – паспортная мощность i-го электрооборудования, кВт;

K_i – коэффициент использования мощности i-го электрооборудования (принимается K_i=0,9);

T_i – время работы i-го оборудования за весь период разработки ПП ч;

Ц – цена электроэнергии, тг/кВт×ч;

i – вид электрооборудования;

n – количество электрооборудования.

Затраты на электроэнергию находятся исходя из продолжительности периода разработки ПО, количества кВт/ч, затраченных на проектирование

ПО и тарифа за 1 кВт/ч. Тариф по городу Алматы для юридических лиц в 2019 году составляет 18,32 тенге за 1 кВт/ч с учетом НДС (согласно данным представленным на официальном сайте ТОО «АлматыЭнергоСбыт»).

$$З_э = 0,9 \cdot 0,9 \cdot 775 \cdot 18,32 = 11500,38 \text{ тг}$$
$$З_э = 0,3 \cdot 0,7 \cdot 775 \cdot 18,32 = 2981,58 \text{ тг}$$

Таблица 4.3 – Затраты на электроэнергию

Наименование оборудования	Паспортная мощность, кВт	Коэффициент использования мощности	Время работы оборудования для разработки ПП, ч	Цена электро энергии, тг/кВт·ч	Сумма, тг
Ноутбук Acer Aspire 5	0,9	0,9	775	18,32	11500,38
Освещение	0,3	0,7	775	18,32	2981,58
ИТОГО					14481,96

4.4 Расчет расходов на оплату труда

В разработке программного продукта заняты 2 сотрудника: инженер-разработчик и программист. Средняя заработная плата инженер-разработчика в 2019 году составляет 200 000 тг, а программиста 180 000 тг (для города Алматы).

Рабочие часы сотрудника за месяц определяются по формуле:

$$Ч_м = N_м \cdot Ч_{рд}, \quad (4.3)$$

где $Ч_м$ — рабочие часы сотрудника за месяц;
$N_м$ — количество рабочих дней за месяц;
$Ч_{рд}$ — количество рабочих часов в день.

$$Ч_м = 21 \times 8 = 168 \text{ ч.}$$

Часовая ставка работника может быть рассчитана по формуле:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i} \quad (4.4)$$

Инженер-разработчик:

$$ЧС_i = \frac{200000}{168} = 1190,48 \ тг$$

Программист:

$$ЧС_i = \frac{180000}{168} = 892,86 \ тг,$$

где $ЗП_i$ – месячная заработная плата i-го работника, тг;

$ФРВ_i$ – месячный фонд рабочего времени i-го работника, час.

Для определения трудоемкости разработки программного продукта используются данные из таблицы 4.1.

Трудоемкость разработки программного продукта инженера разработчика равна 575 чел.×ч (описание задачи, разработка алгоритма, разработка блок-схемы алгоритма, реализация клиентской части проекта, тестирование программы, документация).

$$T_2 = 55 + 120 + 40 + 150 + 150 + 60 = 575 \text{ чел.×ч.}$$

Трудоемкость разработки программного продукта программиста равна 390 чел.×ч. (разработка блок-схемы алгоритма, разработка администраторской части проекта, тестирование программы).

$$T_2 = 40 + 200 + 150 = 390 \text{ чел.×ч.}$$

Общая сумма затрат на оплату труда ($З_{ТР}$) определяется по формуле:

$$З_{ТР} = \sum_{i=1}^{n} ЧС_i \times T_i \ , \qquad (4.5)$$

где $ЧС_i$ – часовая ставка i-го работника, тг;

T_i – трудоемкость разработки ПП, чел.×ч;

i – категория работника;

n – количество работников, занятых разработкой ПП.

Инженер-разработчик:

$$З_{тр} = 1190,48 \times 575 = 684526 \text{ тг.}$$

Программист:

$$З_{тр} = 892,86 \times 390 = 348215,4 \text{ тг.}$$

Общая сумма:

$$З_{тр} = 684526 + 348215,4 = 1032741,4 \text{ тг.}$$

Таблица 4.4 – Затраты на оплату труда

Квалификация	Трудоемкость разработки ПП, чел.×ч	Часовая ставка, тг/ч	Сумма, тг
Инженер-разработчик	575	1190,48	684 526,00
Программист	390	892,86	348 215,40
ИТОГО			1032741,4

Дополнительная заработная плата:

$$З_{доп} = З_{тр} \times 10\% \quad (4.6)$$

$$З_{доп} = 1032741,4 \times 0,1 = 103274,14 \text{ тг.}$$

Фонд заработной платы:

$$Ф_{зп} = З_{тр.о} + З_{доп} \quad (4.7)$$

$$Ф_{зп} = 1032741,4 + 103274,14 = 1136015,54 \text{ тг.}$$

Расчет социального налога:

$$Н_с = (Ф_{зп} – ОПВ) \times 11\%, \quad (4.8)$$

где ОПВ – обязательные пенсионные взносы – 10% от $Ф_{зп}$.

$$Н_с = (1136015,54 – (1136015,54 \times 0,1)) \times 0,11 = 112465,54 \text{ тг.}$$

Расчет амортизационных основных фондов
Общая сумма амортизационных отчислений определяется по формуле:

$$З_{АМ} = \sum_{i=1}^{n} \frac{Ф_i \times Н_{Ai} \times Т_{НИРi}}{100 \times Т_{Эфi}}, \quad (4.9)$$

где $Ф_i$ – стоимость i-го ОФ, тг;

$Н_{Ai}$ – годовая норма амортизации i-го ОФ, %;

$Т_{НИРi}$ – время работы i-го ОФ за весь период разработки ПП, ч;

$Т_{Эфi}$ – эффективный фонд времени работы i-го ОФ за год, ч/год (по информации egov.kz за 2019 год принимается $Т_{Эфi}$=1968);

i – вид ОФ;

n – количество ОФ.

Расчет годовой нормы амортизации ОФ:

$$H_{Ai} = \frac{100}{4} = 25$$

$$H_{Ai} = \frac{100}{T_{Ni}}, \qquad (4.10)$$

где $Т_{Ni}$ – возможный срок использования i-го ОФ, год;

Для определения времени работы ПО для разработки программного продукта используются данные из таблицы 4.1.

Время работы ПО Open Server Panel и разработанного веб-сайта Air-Sultan для разработки программног продукта составляет 470 часов (реализация клиентской части проекта, реализация администраторской части проекта, тестирование программы).

$$Т_i = 150 + 200 + 150 = 500 \text{ ч.}$$

Оборудование:

$$З_{АМ} = \frac{115000 \cdot 25 \cdot 775}{100 \cdot 1968} = 11321{,}77 \ тг$$

Веб-сайт Air-Sultan:

$$З_{АМ} = \frac{90000 \cdot 25 \cdot 500}{100 \cdot 1968} = 5716{,}46 \ тг$$

Таблица 4.5 – Амортизация основных фондов (ОФ)

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Эффективный фонд времени работы оборудования и ПО, ч/год	Время работы оборудования и ПО для разработки ПП, ч	Сумма, тг
Ноутбук Acer Aspire 5	115000	25	1968	775	11321,77
Веб-сайт Air-Sultan	90000	25	1968	500	5716,46
ИТОГО					17038,23

Таблица 4.6 – Смета затрат на разработку ПП

Статьи затрат	Сумма, тг	%
1. Расходы на материалы 2. Расходы на электроэнергию 3. Затраты на оплату труда 4. Социальные отчисления 5. Амортизация затраты основных фондов	205 000,00 14 481,96 1 032 741,40 112 465,54 17 038,23	15 1 75 8 1
ИТОГО	1 381 727,13	100

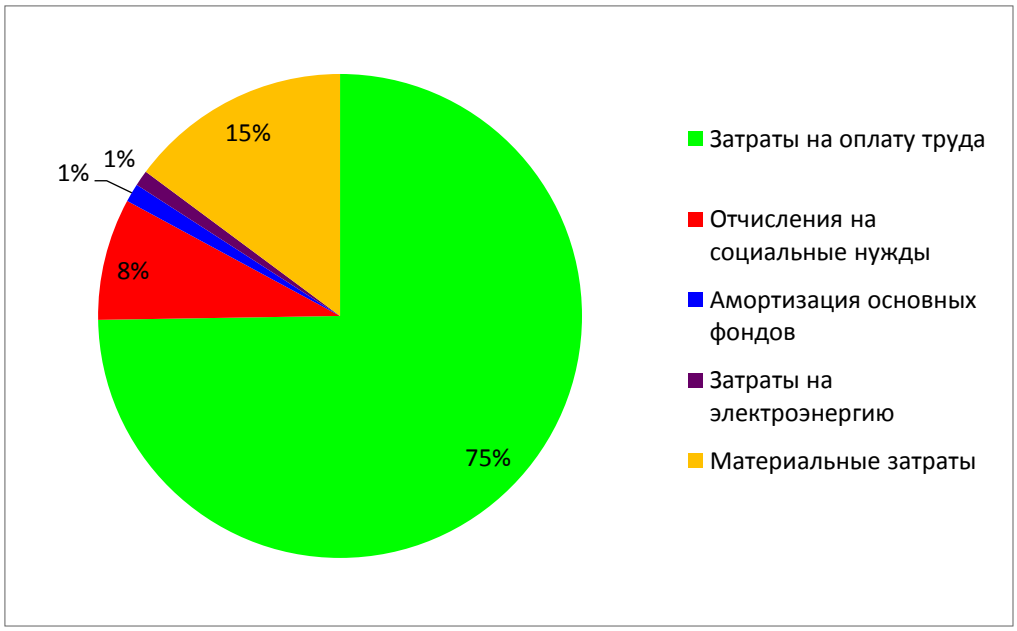


Рисунок 4.1 – Смета затрат на разработку программного продукта

4.5 Определение возможной (договорной) цены программного продукта

Договорная цена ($Ц_Д$) для прикладных ПП рассчитывается по формуле:

$$Ц_Д = З_{НИР} \cdot \left(1 + \frac{Р}{100}\right), \quad (4.11)$$

где $З_{НИР}$ – затраты на разработку ПП (из таблицы 4.6), тг;
Р – средний уровень рентабельности ПП – 25%.

$Ц_Д = 1381727,13 \cdot (1 + 0,25) = 1381727,13 + 345431,78 = 1727158,91$ тг

Далее определяется цена реализации с учетом налога на добавленную стоимость (НДС), ставка НДС устанавливается законодательно Налоговым Кодексом РК. На 2019 год ставка НДС установлена в размере 12%.

Цена реализации с учетом НДС рассчитывается по формуле:

$$Ц_Р = Ц_Д \cdot (1 + НДС) \quad (4.12)$$

$Ц_Р = 1727158,91 \cdot (1 + 0,12) = 1727158,91 + 207259,07 = 1934417,98$ тг

4.6 Вывод по технико-экономической части

Цена реализации веб-сайта резервирования авиабилетов с учетом всех возможных затрат составляет 1 381 727,13 тенге. Основную часть затрат составляют затраты на оплату труда (75%).

Таким образом, цена реализации с учетом НДС равна 1 934 417,98 тг, себестоимость – 1 381 727,13 тг и прибыль – 345 431,78 тг.

5 Охрана труда и безопасность жизнедеятельности

Темой дипломного проекта было выбрано WEB-приложение, состоящее из двух частей, первая часть которого является этапом разработки технического задания, а ко второму этапу относится программная разработка. При разработке были соблюдены все правила техники безопасности.

В связи с тем, что работа является сидячей, а именно разработка сайта, то основными пунктами были: освещение, вентиляция, а также работа с электрическими приборами и правила их эксплуатации.

Помещение представляет собой комнату размерами 12х4х8. К списку оборудований относятся 4 ПК. В помещении есть 2 окна по 0.5 $м^2$ и лампа мощностью 60 Вт. Данные компьютеры не создают большого шумового давления в пределах нормативов. В связи с тем, что вентиляция проведена не самым лучшим образом, я решил, что в данном разделе Безопасности Жизнедеятельности я буду рассчитывать вентиляцию.

5.1 Расчет аспирационной системы для производственного помещения

Таблица 5.1 – Исходные данные

Город	Алматы
Параметры помещения (Д*Ш*В), м	12 * 8 * 4
Данные по оборудованию	
Количество компьютеров, шт.	4
Мощность $P_{об}$, кВт/ч	0.5
КПД η	0.95
Данные по источнику света	
Мощность $N_{осв}$, $Вт/м^2$	60
Вид источника освещения	люминесцентные лампы
Количество сотрудников	2 мужчины, 2 женщины
Окна	
Количество, шт.	2
Площадь 1 окна, $м^2$	0.5
Расположение	СВ
Вид	остекление в один-х метал. переплет
Расчетное время суток, ч.	12-13
Температура в помещении	
Летом, ^oC	23
Зимой, ^oC	21
Вид положения работы	сидячее

5.1.1 Расчет тепловых нагрузок в помещении

В помещениях различного назначения действуют в основном тепловые нагрузки, возникающие снаружи помещения (наружные); тепловые нагрузки, возникающие внутри здания (внутренние).

5.1.2 Расчет тепловых нагрузок снаружи помещения

Наружные тепловые нагрузки подставлены следящими составляющими:

– теплопоступления или теплопотери в результате разности температур снаружи и внутри здания через стены, потолки, полы, окна и двери;

– разность температур снаружи здания и внутри него летом является положительной, в результате чего имеет место приток тепла снаружи во внутрь помещения; и наоборот – зимой эта разность является отрицательной и направления потока тепла меняется;

– теплопоступления от солнечного излучения через застекленные площади; данная нагрузка проявляется в форме ощущаемого тепла;

– теплопоступления от инфильтрации.

Следует отметить, что наружные тепловые нагрузки могут обладать различными свойствами, т.е. могут быть положительными в зависимости от времени года и времени суток.

В зависимости от времени года и времени суток наружные тепловые нагрузки могут быть положительными.

Теплопоступления и теплопотери в результате разности температур определяется по формуле (5.1):

$$Q_{\text{огр}} = V_{\text{пом}} * X_o * \left(t_{H\text{расч}} - t_{\text{Врасч}}\right) \quad (5.1)$$

где $V_{\text{пом}}$ – объем помещения (5.2), м^3

$$V_{\text{пом}} = 12 * 8 * 4 = 384 \text{ м}^3; \quad (5.2)$$

где X_o – удельная тепловая характеристика, Вт/м^3 ^{o}С, X_o = 0.42;

$t_{\text{Нрасч}}$ – наружная температура. Для холодного периода - средняя температура самого холодного месяца в 13 часов, для теплого периода - средняя температура самого жаркого месяца в 13 часов.

$t_{\text{Врасч}}$ – внутренняя температура, выбирается с учетом комфортных условий или технологических требований, предъявляемых к производственным процессам.

Для теплого времени года (5.3):

$t_{\text{Нрасч}}$ = 29,4 ^{o}С

$t_{\text{Врасч}}$ = 26 ^{o}С

$$Q_{\text{огр}} = 384 * 0.42 * 3{,}4 = 548 \text{ Вт} \quad (5.3)$$

Для холодного времени года (5.4):

$t_{Нрасч}$ = -9 °С

$t_{Врасч}$ = 19 °С

$$Q_{огр} = 384 * 0.42 * |-28| = 4515{,}84 \text{ Вт} \quad (5.4)$$

Избыточная теплота солнечного излучения в зависимости от типа стекла почти до 90% поглощается средой помещения, остальная часть отражается. максимальная тепловая нагрузка достигается при максимальном уровне излучения, которое имеет прямую и рассеянную составляющие. Интенсивность излучения зависит от ширины местности, времени года и времени суток.

Теплопоступление от солнечного излучения через остекление определяется по формуле (5.5):

$$Q_P = (q^I F_O^I + q^{II} F_O^{II}) * \beta_{с.з} \quad (5.5)$$

где q^I, q^{II} - тепловые потоки от прямой и рассеянной солнечной радиации, Вт/м^2;

F_O^I, F_O^{II}- площади светового проема, облучаемые и необлучаемые прямой солнечной радиации, м^3;

$\beta_{с.з}$ – коэффициент теплопропускания.

$\beta_{с.з} = 0.15$;

При отсутствии наружных затеняющих козырьков, ребер и т.д. для периода облучения остекления солнцем, когда его лучи проникают через окно в помещение $F_O^I = F_O;\ F_O^{II} = 0$, (5.6):

$$Q_P = q^I F_O * \beta_{с.з} = \left(q_{вп} + q_{вр}\right) * K_1^C * K_2 * n * S_O \quad (5.6)$$

где $q_{вп}$, $q_{вр}$- тепловые потоки от прямой рассеянной радиации, Вт/м^2.

Для широты в 44°СШ до полудня в 11-12 ч. При расположении З: $q_{вп}$, $q_{вр}$

$q_{вп} = 73$ Вт/м^2

$q_{вр} = 77$ Вт/м^2

F_O– площадь светового проема (5.7) (n – число окон, S_O – площадь одного окна);

$$F_O = n * S_o = 2 * 0.5 = 1 \text{ м}^2 \quad (5.7)$$

где K_1 – коэффициент затемнения остекления переплетами (K_1^C – для облученных проемов).

$K_1^C = 0.72$
K_2 – коэфициент загрязнения остекления.
$K_2 = 0.9$

Тогда (5.8):

$$Q_P = (73 + 77) * 0.72 * 0.9 * 0.15 * 1 = 14{,}6 \text{ Вт} \quad (5.8)$$

Поступление солнечного излучения равно (5.9):

$$Q_P = 290 \text{ Вт} \quad (5.9)$$

5.1.3 Расчет тепловых нагрузок внутри помещения

Внутренние тепловые нагрузки в жилых, офисных или относящийся к сфере обслуживание помещения слагаются в основном из тепла:

– выделяемого людьми;

– выделяемого лампами и осветительными, электробытовыми приборами;

– выделяемого компьютерами, печатающими устройствами и фотокопировальными машинами и пр. (в офисных и других помещениях).

В производственных и технологических помещениях различного назначения дополнительными источниками тепловыделение могут быть: нагретое производственное оборудование; горячие материалы, в том числе жидкости различного рода полуфабрикаты; продукты сгорания и химических реакций.

Теплопоступления от людей зависит от интенсивности выполняемой работы и параметров окружающего воздуха. Тепло, выделяемое человеком, складывается из ощутимого (явного), то есть передаваемого в воздух помещения путём конвекции и лучеиспусканий, и скрытого тепла, затрачиваемого на испарение влаги с поверхности кожи и из легких.

Летом при 24 один мужчина выделяет явного тепла 61 Вт, а общего – 102 Вт (при работе стоя, легком движении). Женщина выделяет 85% ль нормы тепловыделений взрослого мужчины. Выделение явного тепла составит (5.10):

$$Q_л^я = 61 * 2 + 61 * 2 * 0{,}85 = 225{,}7 \text{ Вт} \quad (5.10)$$

Выделение общего тепла (5.11):

$$Q_л^о = 102 * 2 + 102 * 2 * 0{,}85 = 377{,}4 \text{ Вт} \quad (5.11)$$

Зимой при 20 ^{o}C один мужчина выделяет явного тепла 82 Вт, а общего – 103 Вт. Женщина выделяет 85% нормы тепловыделений взрослого мужчины. Тогда выделение явного тепла в помещении составит (5.12):

$$Q_{\text{л}}^{\text{я}} = 82 * 2 + 82 * 2 * 0{,}85 = 303{,}4 \text{ Вт} \quad (5.12)$$

Выделение общего тепла (5.13):

$$Q_{\text{л}}^{\text{о}} = 103 * 2 + 103 * 2 * 0{,}85 = 381{,}1 \text{ Вт} \quad (5.13)$$

Теплопоступление от осветительных приборов, оргтехники и оборудования рассчитывается следующим образом. Теплопоступление от ламп определяется по формуле (5.15):

$$Q_{\text{осв}} = \eta * N_{\text{осв}} * F_{\text{пол}} \quad (5.15)$$

где η - коэффициент перехода электрической энергии в тепловую (для люминесцентных ламп $\eta = 0.5 – 0.6$);

$N_{\text{осв}}$ – установленная мощность ламп ($N_{\text{осв}} = 60$ Вт/м^2);

$F_{\text{пол}}$ – площадь пола ($F_{\text{пол}} = 12 * 8 = 96$ м^2);

Тогда (5.16):

$$Q_{\text{осв}} = 0.5 * 60 * 96 = 2880 \text{ Вт} \quad (5.16)$$

Тепло, выделяемое производственным оборудованием, определяется по формуле (5.17):

$$Q_{\text{об}} = N_{\text{уст}} * K \quad (5.17)$$

где $N_{\text{уст}}$ – паспортная мощность оборудования ($N_{\text{уст}} = 1.8$ кВт/ч);

K - единиц оборудования ($K = 4$);

Тогда (5.18):

$$Q_{\text{об}} = 1.8 * 4 * 0.95 = 10.3 \text{ кВт} \quad (5.18)$$

Теплопритоки, возникающие за счет находящейся оргтехники – это 30% мощности оборудования (5.19):

$$Q_{\text{об}} = 1.8 * 4 * 0.3 = 3.24 \text{ кВт} \quad (5.19)$$

5.2 Расчет теплового баланса помещения

На основании выполненных расчетов поставим баланс теплопоступлений в помещении:

Летом (5.20):

$$Q_{\text{изб}} = 43 + 226 + 2880 + 10300 + 3240 + 550 = 172504 \text{ Дж} \quad (5.20)$$

Зимой (5.21):

$$Q_{\text{изб}} = 43 + 303 + 2880 + 10300 + 3240 + 4515 = 21249 \text{ Дж} \quad (5.21)$$

Т.к. тепловой баланс для зимы больше летнего теплового баланса, то рассчитаем тепло-напряжённость воздуха по формуле (5.22, 5.23):

$$Q_H = \frac{Q_{\text{изб}} * 860}{V_{\text{пом}}} \quad (5.22)$$

$$Q_H = \frac{21249 * 860}{384} = 47{,}588 \text{ кал/м}^3 \approx 47{,}58 \text{ ккал/м}^3 \quad (5.23)$$

При $Q_H > 20$ ккал/м^3, $\Delta t = 8$ °C

Определение количества воздуха, необходимое для поступления в помещение (5.24) и расчет (5.25):

$$L = \frac{Q_{\text{изб}} * 860}{C * \Delta t * \gamma} \quad (5.24)$$

где $C = 0.24$ ккал/(кг°C) – теплоемкость воздуха,

$\gamma = 1.206$кг/м^3 – удельная масса приточного воздуха.

$$L = \frac{21249 * 860}{0.24 * 10^4 * 8 * 1.206} = 789.22 \frac{\text{м}^3}{\text{час}} \quad (5.25)$$

Определение кратности воздухообмена (5.26, 5.27):

$$n = \frac{L}{V_{\text{пом}}} \quad (5.26)$$

$$n = \frac{789{,}22}{384} = 2.055 \text{ час}^{-1} \quad (5.27)$$

5.3 Выбор кондиционера

Исходя из полученных данных, берется кондиционер типа BALLU BCFB/OUT-48HN1 в количестве 1 штук. Характеристики приведены в таблицах 5.3.1 и 5.3.2.

Управление работой настенного кондиционера производится дистанционного пульта, который позволяет задать режим работы кондиционера: обогрев, охлаждение, осушку, вентиляцию, ночной режим, энергосберегающий режим; задать требуемую температуру, которую должен поддерживать автоматический; выбрать режим работы вентилятора: настроить таймер, который включить или выключить кондиционер в заданное время; автоматически регулировать положение направляющих штор и изменить таким образом направление воздушного потока.

Таблица 5.3.1 – Характеристики выбранного кондиционера.

Электропитание	Расход воздуха внутреннего блока	Расход воздуха внешнего блока	Производительность по теплу	Производительность по холоду	Мощность компрессора	Электронагреватель
В	$м^3/ч$	$м^3/ч$	кВт	кВт	кВт	кВт
220	606	786	3.47	3.3	2.1	3

Таблица 5.3.2 – Размеры выбранного кондиционера.

Размеры внутреннего блока (В*Ш*Г)	Вес внутреннего блока	Хладагент	Уровень шума внутреннего блока
мм	кг	-	ДБ
288*800*206	9	R410A	27/29/41

5.4 Приведение схемы расположения кондиционера в помещении

Во внешнем блоке находятся компрессор, конденсатор и вентилятор. Внешний блок можно установить на стене здания, на крыше или на чердаке, в подсобном помещении или на балконе, то есть в таком месте, где горячий конденсатор может продуваться атмосферным воздухом более низкой температуры.

Внутренний блок устанавливается непосредственно в кондиционируемом помещении и предназначен для охлаждения или нагревания воздуха, фильтрации его и создания необходимой подвижности воздуха в помещении. Внутренние блоки поддерживают заданную температуру, обеспечивают равномерное распределение воздуха в помещении и работают практически бесшумно (уровень шума 35-38 дБ).

Управление работой настенного кондиционера производится с дистанционного пульта, который позволяет задать режим работы кондиционера: обогрев, охлаждение, осушку, вентиляцию, ночной режим; задать требуемую температуру, которую должен поддерживать автоматически; выбрать режим работы вентилятора: настроить таймер,

который включит или выключит кондиционер в заданное время; автоматически регулировать положение направляющих шторок и изменить таким образом направление воздушного потока.

Так как количества воздуха, необходимое для поступления в помещение равно 789,22 м3/час, то будет использован один кондиционер BALLU BCFB/OUT- 48HN1, который выдает необходимый нам расход воздуха.

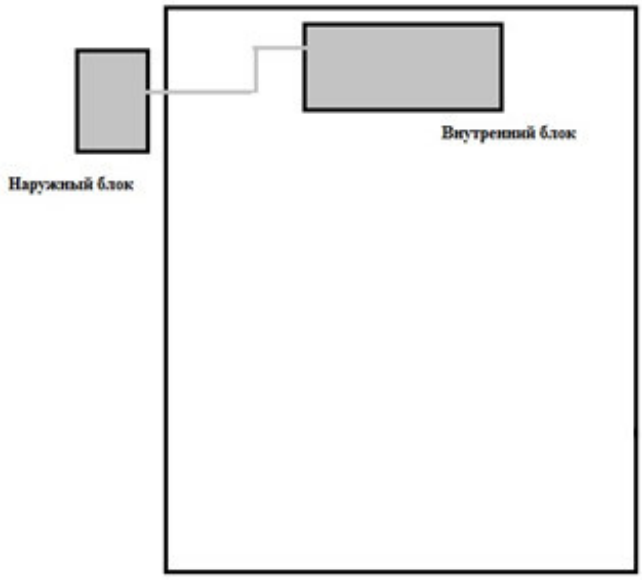


Рисунок 5.4 - Схема расположение кондиционера в производственном помещении

5.5 Вывод по части безопасности жизнедеятельности

В этой части дипломного проекта мы расчитали возмонжные тепловые нагрузки как и внутренние так и внешние для выбранного нами помещения. Расчитали необходимое количество подаваемого воздуха, после того как узнали количество воздуха выбрали подходящий кондиционер по его характеристикам и указали место расположения его блоков внешнего вентиляционного и внутреннего в помещении.

Тем самым можно сделать вывод, что безопасность жизнедеятельности учавствует во всех аспектах жизнедеятельности человека в любых профессиях и быте. На примере разработки моего проекта были соблюдены все нормы вентиляции, установлен кондционер, а так же все пункты с овещением. Так же установлено, что тепловые нагрузки зависят от размерности помещения и от количества людей находящимся в нем, а так же от таких внешних тепловых нагрузок как солнечные лучи и количество окон в помещении через которые они проходят. Большое влияние оказали погодные условия в городе проведения разработки.

Заключение

В ходе выполнения дипломного проекта была разработана система, которая обеспечивает пользователей автоматизированной системой бронирования авиабилетов.

При выполнении дипломного проекта были выполнены следующие задачи:

– проведено изучение предметной области на примере виртуальной IT-компании;

– определены основные требования к программному продукту;

– был проведен анализ существующих аналогов данной системы, ползволяющие глубже изучить структуру программы;

– определены основные требования к системе учета и регистрации;

– спроектирована базы данных, программная часть и аппаратная часть;

– проведен анализ существующих технологий для решения поставленных задач;

– проведен анализ и выбор компонентов для сборки разработанного проекта;

– разработка системы контроля и учета;

– настройка локального сервера;

– произведено экономическое обоснование целесообразности разрабатываемого продукта, где были сделано следующее заключение:

цена реализации окупает все затраты, потрачены на разработку. Прибыль от реализации проекта составила 345 431,78 тг;

также были исследованы условия труда в помещении, где велась разработка, и были предложены мероприятия по улучшению качества воздуха в помещении.

Список литературы

1 Ховард Майкл, Лебланк Дэвид, Виега Джон Как написать безопасный код на C++, Java, Perl, PHP, ASP.NET; ДМК Пресс - М., 2014. - 288 с. «HTML5 + CSS3. Основы современного веб-дизайна» / Кириченко А. В., Хрусталев А. А. – СПб: «Наука и Техника», 2018. – 352 с.

2 Дронов В. PHP, MySQL и Dreamweaver. Разработка интерактивных Web-сайтов; БХВ-Петербург - М., 2007. - 480 с. «JavaScript. Подробное руководство, 6-е издание» / Д. Флэнаган. – СПб: Символ-Плюс, 2012. – 1080 с.

3 Лоусон, Б. Изучаем HTML5. Библиотека специалиста / Б. Лоусон, Р. Шарп. - М.: Книга по Требованию, 2012. - 304 с.

4 Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р. Никсон. - Москва: РГГУ, 2016. - 688 с. «JavaScript на примерах» / А. П. Никольский. – СПб.: «Наука и техника», 2017. – 272 с.

5 Пилгрим, Марк Погружение в HTML5 / Марк Пилгрим. - М.: БХВ-Петербург, 2011. - 304 с.

6 Хуан, Диего Гоше HTML5. Для профессионалов/Хуан Диего Гоше. - М.: "Издательство "Питер", 2011. - 496 с.

7 Дакетт, Джон Основы веб-программирования с использованием HTML, XHTML и CSS / Джон Дакетт. - М.: Эксмо, 2012. - 768 с.

8 Мейер, Э. CSS — каскадные таблицы стилей. Подробное руководство / Э. Мейер. - М.: СПб: Символ-Плюс; Издание 2-е, 2016. - 576 с. «Spring 4 для профессионалов» / Крис Шеффер, Кларенс Хо, Роб Харроп; – Москва: Вильямс, 2015. – 752 с.

9 Робсон, Э. Изучаем HTML, XHTML и CSS / Э. Робсон. - М.: Питер, 2017. - 958 с. «Java. Методы программирования» / И. Н. Блинов, В. С. Романчик. – Минск: издательство «Четыре четверти», 2013. – 896 с.

10 Аткинсон MySQL. Библиотека профессионала / Аткинсон, Леон. - М.: Вильямс, 2014. - 624 с.

11 Конверс PHP 5 и MySQL. Библия пользователя / Конверс, др. Т. и. - М.: Вильямс, 2016. - 426 с.

12 Кингсли-Хью, К.Э. JavaScript 1.5: учебный курс / К.Э. Кингсли-Хью. - М.: СПб: Питер, 2013. - 272 с.

13 Дронов, В. JavaScript в Web-дизайне / В. Дронов. - М.: СПб: БХВ, 2014. - 880 с.

14 Аткинсон, Леон MySQL. Библиотека профессионала; М.: Вильямс, 2013. – 624 с.

15 Грофф, Джеймс; Вайнберг, Пол SQL: полное руководство; Киев: BHV, 2012. – 608 с.

16 Яргер, Р.Дж.; Риз, Дж.; Кинг, Т. MySQL и mSQL: Базы данных для небольших предприятий и Интернета; СПб: Символ-Плюс, 2011. – 560 с.

17 Авиабилеты – IT системы бронирования; — 2013. URL: habr.com/

18 Пауэлл, Томас; Шнайдер, Фриц Полный справочник по JavaScript; М.: Вильямс; Издание 2-е, 2007. - 960 с.

19 Яргер, Р.Дж.; Риз, Дж.; Кинг, Т. MySQL и mSQL: Базы данных для небольших предприятий и Интернета; СПб: Символ-Плюс, 2013. - 560 с.

20 Крокфорд Д. JavaScript. Сильные стороны; Питер - М., 2016. **-** 262 с.

21 Чаффер Д. Изучаем jQuery 1.3. Эффективная веб-разработка на JavaScript; Символ-плюс - М., 2015. - 391 с.

Приложение А
(обязательное)

Техническое задание

Наименование программного продукта (ПП): «Air-Sultan».

Цель разработки ПО: создание веб-приложения со структурированием базы данных по данной тематике, функцией для реализации навигации поисковых запросов и доступом к Интернету для осуществления связи и обновления данных.

Идеология программного обеспечения: идеологией программного обеспечения является оценивание в выборе образовательного фонда в определённой области у пользователей.

Используемая технология создания ПО: среда Sublime Text, PyCharm.

Выбор модели ПО: структурная модель – позволяет оформлять отдельные блоки программы (повторяющиеся и неповторяющиеся) в процедуры и функции, которые затем могут использоваться в других частях программы. Изменения в коде функций и процедур не влекут за собой изменение других частей кода программы.

Описание: модуль авторизация – главный модуль, с помощью которого непосредственно будут совершаться различные функции. Включение этого модуля происходит после регистрации пользователя. Модуль регистрация пользователя – модуль, в котором пользователь заполнит свои личные данные для регистрации перед совершением рейсов. После выбора его данные и результат отправленной заявки будет занесён в базу. Модуль заполнения данных брони – немаловажная чать ПП, используемая при регистрировании данных брони для определенного пользователя. Модуль корзины – временное хранение данных о рейсах пользователя. Модуль истории бронирований – данный модуль сохраняет все совершенные рейсы пользователя. База пользователей и их результатов – база, содержащая данные обо всех пользователях, аэропортах, рейсах, типах класса.

Выбор архитектуры построения ПП: Model-View-Controller.

Выбор метода разработки структуры ПП: нисходящий метод.

Выбор языка программирования: PHP, Javascript.

Предполагаемая аудитория (тип, возраст и т.д.): люди совершеннолетнего возраста

Общий объем ПП, Мб: 15 Мб.

Приложение Б
(обязательное)

Листинг программы

Admin.js

```
$(document).ready(function() {
	$("#update").hide();
	$("#add").show();
	$("#delete").hide();
	//Добавление рейса
	$("#ad").click(function(){
	var flightno = document.getElementById("flightno2").value;
	var airplaneid = document.getElementById("airplaneid2").value;
	var departure = document.getElementById("departure2").value;
	var dtime = document.getElementById("dtime2").value;
	var arrival = document.getElementById("arrival2").value;
	var atime = document.getElementById("atime2").value;
	var ec = document.getElementById("ecapacity2").value;
	var ep = document.getElementById("eprice2").value;
	var bc = document.getElementById("bcapacity2").value;
	var bp = document.getElementById("bprice2").value;
	xmlhttp = new XMLHttpRequest();
	xmlhttp.onreadystatechange = function() {
	  if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
	   var content = xmlhttp.responseText;
	   if(content != "0")
	   {
	            alert(content);

	           }
	           else
	           {
	           alert("Рейс добавлен");
	           $(':input','#addf')
	          .not(':button, :submit, :reset, :hidden')
	         .val('');
	           }
	}
      }
xmlhttp.open("GET","Adminadd.php?flightno="+flightno+"&airplaneid="+airplaneid+"&departu
re="+departure+"&dtime="+dtime+"&arrival="+arrival+"&atime="+atime+"&ec="+ec+"&ep="+
ep+"&bc="+bc+"&bp="+bp,true);
      xmlhttp.send();

	});

	$("#up").click(function(){
	var flightno = document.getElementById("flightno1").value;
	var airplaneid = document.getElementById("airplaneid1").value;
```

```
var departure = document.getElementById("departure1").value;
        var dtime = document.getElementById("dtime1").value;
        var arrival = document.getElementById("arrival1").value;
        var atime = document.getElementById("atime1").value;
        var ec = document.getElementById("ecapacity1").value;
        var ep = document.getElementById("eprice1").value;
        var bc = document.getElementById("bcapacity1").value;
        var bp = document.getElementById("bprice1").value;
        xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
          if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
           var content = xmlhttp.responseText;

           if(content != "0")
           {
                    alert(content);

                   }
                   else
                   {
                   alert("Информация обновлена");
                   $(':input','#result')
                  .not(':button, :submit, :reset, :hidden')
                 .val('');
                   }
        }
      }


xmlhttp.open("GET","Adminupdate.php?flightno="+flightno+"&airplaneid="+airplaneid+"&depa
rture="+departure+"&dtime="+dtime+"&arrival="+arrival+"&atime="+atime+"&ec="+ec+"&ep
="+ep+"&bc="+bc+"&bp="+bp,true);
      xmlhttp.send();

        });


        //Поиск рейсов
        $("#search").click(function(){
                var url = "Adminsearch.php";

                $.getJSON(url,          {flightno:          document.getElementById("number").value},
function(data){
                        if(data.flights == "")
                                alert("Указанный рейс не найден")
else{

                        $.each(data.flights, function(i, flight){
```

```
                document.getElementById("flightno1").value          = flight.number;
                document.getElementById("airplaneid1").value = flight.airplane_id;
                document.getElementById("departure1").value = flight.departure;
                document.getElementById("dtime1").value = flight.d_time;
                document.getElementById("arrival1").value = flight.arrival;
                document.getElementById("atime1").value = flight.a_time;

                        });
                        $.each(data.classes, function(i, class_info){
                                if(class_info.name == "Economy")
                                {
                        document.getElementById("ecapacity1").value = class_info.capacity;
                        document.getElementById("eprice1").value = class_info.price;
                                }
                                else
                                {
                        document.getElementById("bcapacity1").value = class_info.capacity;
                        document.getElementById("bprice1").value = class_info.price;
                                }
                        });
                }
                });
        });
        $("#de").click(function(){
                var flightno = document.getElementById("flightno1").value;
                xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
          if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
           var content = xmlhttp.responseText;

                        alert(content);
        }
      }
      xmlhttp.open("GET","Admindelete.php?flightno="+flightno,true);
      xmlhttp.send();
      $(':input','#result')
          .not(':button, :submit, :reset, :hidden')
        .val('');
        });
        $("#ad1").hover(function(){
                xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {

    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
            var content = xmlhttp.responseText;
                document.getElementById("detail").innerHTML = content;
                        $("#detail").show();

        }
```

```
    }
        .val('');
        $("#number").val('');
                $("#update").show();
                $("#add").hide();
                $("#de").hide();
                $("#up").show();;
        });
        $("#d").click(function(){
                $("#number").val('');
                $(':input','#result')
         .not(':button, :submit, :reset, :hidden')
        .val('');
                $("#update").show();
                $("#add").hide();
                $("#de").show();
                $("#up").hide();
        });
```

Adminadd.php

```
<?php
$flightno = $_GET['flightno'];
$airplaneid = $_GET['airplaneid'];
$departure = $_GET['departure'];
$dtime = $_GET['dtime'];
$arrival = $_GET['arrival'];
$atime = $_GET['atime'];
$ec = $_GET['ec'];
$ep = $_GET['ep'];
$bc = $_GET['bc'];
$bp = $_GET['bp'];
include_once 'dbconnect.php';
$sql = "INSERT INTO flight VALUES( '$flightno', '$airplaneid', '$departure', '$dtime', '$arrival',
'$atime' )";
if(! mysqli_query($conn, $sql))
{
        echo "Errormessage: ".mysqli_error($conn)."\n";
}
$sql = "INSERT INTO class VALUES( '$flightno', 'Economy', '$ec', '$ep')";
if(! mysqli_query($conn, $sql))
{
        echo "Errormessage: ".mysqli_error($conn)."\n";

}
$sql = "INSERT INTO class VALUES('$flightno', 'Business', '$bc', '$bp')";
if(! mysqli_query($conn, $sql))
{
        echo "Errormessage: ".mysqli_error($conn)."\n";
}
```

```
echo 0;

mysqli_close($conn);

?>
```

Admindelete.php

```
<?php
$flightno = $_GET['flightno'];
include_once 'dbconnect.php';
$sql = "DELETE FROM flight WHERE number = '$flightno'";
if(! mysqli_query($conn, $sql))
{
	echo "Errormessage: ".mysqli_error($conn)."\n";
}
else
{
	echo "Удалено";
}
mysqli_close($conn);
?>
```

Adminsearch.php

```
<?php
$flightno = $_GET['flightno'];
include_once 'dbconnect.php';
$var = array();
$sql = "SELECT * FROM flight WHERE flight.number = '$flightno'";
if(! ($result = mysqli_query($conn, $sql)))
{
	echo "Errormessage: ".mysqli_error($conn)."\n";
}
else
{
	while($row = mysqli_fetch_object($result))
	{
		$var[] = $row;
	}
	echo '{"flights":'.json_encode($var).' , ';
}
$var2 = array();
$sql = "SELECT * FROM class WHERE number = '$flightno'";
if(! ($result = mysqli_query($conn, $sql)))
{
	echo "Errormessage: ".mysqli_error($conn)."\n";
}
else
{
	while($row = mysqli_fetch_object($result))
```

```
        {
                $var2[] = $row;
        echo '"classes":'.json_encode($var2).'}';
        }
}
mysqli_close($conn);
?>
```

AdminUpdate.php

```
<?php
$flightno = $_GET['flightno'];
$airplaneid = $_GET['airplaneid'];
$departure = $_GET['departure'];
$dtime = $_GET['dtime'];
$arrival = $_GET['arrival'];
$atime = $_GET['atime'];
$ec = $_GET['ec'];
$ep = $_GET['ep'];
$bc = $_GET['bc'];
$bp = $_GET['bp'];
include_once 'dbconnect.php';
$sql = "UPDATE flight SET  airplane_id = '$airplaneid', departure = '$departure', d_time =
'$dtime', arrival = '$arrival', a_time = '$atime' WHERE number = '$flightno'";
if(! mysqli_query($conn, $sql))
{
        echo "\nErrormessage: ".mysqli_error($conn)."\n";
}
$sql = "UPDATE class SET capacity = '$ec', price = '$ep' WHERE number = '$flightno' AND
name = 'Economy'";
if(! mysqli_query($conn, $sql))
{
        echo "\nErrormessage: ".mysqli_error($conn)."\n";
}
$sql = "UPDATE class SET capacity = '$bc', price = '$bp' WHERE number = '$flightno' AND
name = 'Business'";
if(! mysqli_query($conn, $sql))
{
        echo "Errormessage: ".mysqli_error($conn)."\n";
}
echo 0;

mysqli_close($conn);
?>
```

Airline2.sql

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";
```

```
CREATE TABLE `airplane` (
  `type` varchar(10) NOT NULL,
  `company` varchar(20) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
INSERT INTO `airplane` (`ID`, `type`, `company`) VALUES
('1001', 'A320', 'Airbus'),
('1002', 'A380', 'Airbus'),
('1003', 'B738', 'Boeing'),
('1004', 'B747', 'Boeing'),
('1005', 'B757', 'Boeing'),
('1006', 'B767', 'Boeing'),
('1007', 'E190', 'Embraer'),
('1008', 'E190-2', 'Embraer');
CREATE TABLE `airport` (
  `code` varchar(10) NOT NULL,
  `name` varchar(50) NOT NULL,
  `city` varchar(20) NOT NULL,
  `region` varchar(20) NOT NULL,
  `country` varchar(20) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `airport` (`code`, `name`, `city`, `region`, `country`) VALUES
('AGZ', 'Международный аэропорт Аягуз', 'Аягуз', 'ВКО', 'Казахстан'),
('AKX', 'Международный аэропорт Актобе', 'Актюбинск', 'Актюбинская', 'Казахстан'),
('ALA', 'Международный аэропорт Алматы', 'Алма-Ата', 'Алматинская', 'Казахстан'),
('ATX', 'Международный аэропорт Атбасар', 'Атбасар', 'Акмолинская', 'Казахстан'),
('AYK', 'Международный аэропорт Аркалык', 'Аркалык', 'Костанайская', 'Казахстан'),
('BXH', 'Международный аэропорт Балхаш', 'Балхаш', 'Карагандинская', 'Казахстан'),
('BXJ', 'Международный аэропорт Боралдай', 'Боралдай', 'Алматинская', 'Казахстан'),
('CIT', 'Международный аэропорт Шымкент', 'Шымкент', 'Туркестанская', 'Казахстан'),
('DMB', 'Международный аэропорт Аулие-Ата', 'Тараз', 'Жамбылская', 'Казахстан'),
('DZN', 'Международный аэропорт Жезказган', 'Жезказган', 'Карагандинская', 'Казахстан'),
('EKB', 'Международный аэропорт Экибастуз', 'Экибастуз', 'Павлодарская', 'Казахстан'),
('GUW', 'Международный аэропорт Атырау', 'Атырау', 'Атырауская', 'Казахстан'),
('KGF', 'Международный аэропорт Сары-Арка', 'Караганда', 'Карагандинская', 'Казахстан'),
('KOV', 'Международный аэропорт Кокшетау', 'Кокшетау', 'Акмолинская', 'Казахстан'),
('KSN', 'Международный аэропорт Наримановка', 'Костанай', 'Костанайская', 'Казахстан'),
('KZO', 'Международный аэропорт Кызылорда', 'Кызылорда', 'Кызылординская',
'Казахстан'),
('PLX', 'Международный аэропорт Семей', 'Семей', 'ВКО', 'Казахстан'),
('PPK', 'Международный аэропорт Петропавловск', 'Петропавлоск', 'СКО', 'Казахстан'),
('PWQ', 'Международный аэропорт Павлодар', 'Павлодар', 'Павлодарская', 'Казахстан'),
('SCO', 'Международный аэропорт Актау', 'Актау', 'Мангистауская', 'Казахстан'),
('TDK', 'Международный аэропорт Талдыкорган', 'Талдыкорган', 'Алматинская',
'Казахстан'),
('TSE', 'Международный аэропорт Нур-Султан', 'Нур-Султан', 'Акмолинская', 'Казахстан'),
('UKK', 'Международный аэропорт Усть-Каменогорск', 'Усть-Каменогорск', 'ВКО',
'Казахстан'),
('URA', 'Международный аэропорт Уральск', 'Уральск', 'ЗКО', 'Казахстан');
```

```
CREATE TABLE `book` (
  `ID` int(11) NOT NULL,
  `time` datetime NOT NULL,
  `date` date NOT NULL,
  `flightno` varchar(10) NOT NULL,
  `username` varchar(45) NOT NULL,
  `classtype` varchar(20) NOT NULL,
  `paid` int(1) DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `book` (`ID`, `time`, `date`, `flightno`, `username`, `classtype`, `paid`) VALUES
(68, '2019-05-11 09:36:24', '2019-05-21', 'AA100', 'sultan', 'Economy', 1),
(69, '2019-05-11 10:41:41', '2019-05-28', 'AA100', 'sultan', 'Economy', 1),
(70, '2019-05-11 10:41:41', '2019-05-28', 'AA106', 'sultan', 'Economy', 1),

CREATE TABLE `class` (
  `number` varchar(10) NOT NULL,
  `name` varchar(20) NOT NULL,
  `capacity` int(11) NOT NULL,
  `price` float NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `class` (`number`, `name`, `capacity`, `price`) VALUES
('', 'Business', 0, 0),
('', 'Economy', 0, 0),
('AA100', 'Business', 20, 75000),
('AA100', 'Economy', 150, 35000),
('AA1000', 'Business', 20, 81000),
('AA1000', 'Economy', 220, 24000),

CREATE TABLE `flight` (
  `number` varchar(20) NOT NULL,
  `airplane_id` varchar(10) NOT NULL,
  `departure` varchar(10) NOT NULL,
d_time` time NOT NULL,
  `arrival` varchar(10) NOT NULL,
  `a_time` time NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `flight` (`number`, `airplane_id`, `departure`, `d_time`, `arrival`, `a_time`)
VALUES
('', '1002', 'ALA', '00:00:00', 'TSE', '00:00:00'),
('AA100', '1006', 'ALA', '06:10:00', 'TSE', '07:55:00'),
('AA1000', '1002', 'ALA', '13:25:00', 'TSE', '15:20:00'),
CREATE TABLE `passanger` (
  `username` varchar(30) NOT NULL,
`firstname` varchar(45) DEFAULT NULL,
middlename` varchar(45) DEFAULT NULL
  `lastname` varchar(45) DEFAULT NULL,
```

```
  `email` varchar(45) DEFAULT NULL,
  `cellphone` varchar(15) DEFAULT NULL,
  `gender` varchar(10) DEFAULT NULL,
  `birthday` date DEFAULT NULL,
  `password` varchar(45) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
INSERT INTO `passanger` (`username`, `firstname`, `middlename`, `lastname`, `email`,
`cellphone`, `gender`, `birthday`, `password`) VALUES
 ('isambergenov', 'Sultan', '', 'Issambergenov', 'sultan-isambergenov@gmail.com', '87015275547',
'Male', '1997-12-28', '1937284600MCSultan");

ALTER TABLE `airplane`
  ADD PRIMARY KEY (`ID`);

ALTER TABLE `airport`
  ADD PRIMARY KEY (`code`);

ALTER TABLE `book`
  ADD PRIMARY KEY (`ID`,`flightno`),
  ADD KEY `username_idx` (`username`),
  ADD KEY `classname_idx` (`classtype`),
  ADD KEY `flightno_idx` (`flightno`,`classtype`);
ALTER TABLE `class`
  ADD PRIMARY KEY (`number`,`name`),
  ADD KEY `number_idx` (`number`);
ALTER TABLE `flight`
  ADD PRIMARY KEY (`number`),
  ADD KEY `code_idx` (`departure`,`arrival`),
  ADD KEY `airplaneid_idx` (`airplane_id`),
  ADD KEY `arrival_idx` (`arrival`);

ALTER TABLE `passanger`
 ADD PRIMARY KEY (`username`);
ALTER TABLE `book`
  MODIFY `ID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=89;


ALTER TABLE `book`
ADD CONSTRAINT `flightno` FOREIGN KEY (`flightno`,`classtype`) REFERENCES `class`
(`number`, `name`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `username` FOREIGN KEY (`username`) REFERENCES `passanger`
(`username`) ON DELETE NO ACTION ON UPDATE NO ACTION;

--
ALTER TABLE `class`
  ADD CONSTRAINT `number` FOREIGN KEY (`number`) REFERENCES `flight` (`number`)
ON DELETE CASCADE ON UPDATE CASCADE;
ALTER TABLE `flight`
```

```
  ADD CONSTRAINT `airplaneid` FOREIGN KEY (`airplane_id`) REFERENCES `airplane`
(`ID`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `arrival` FOREIGN KEY (`arrival`) REFERENCES `airport` (`code`) ON
DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `departure` FOREIGN KEY (`departure`) REFERENCES `airport`
(`code`) ON DELETE CASCADE ON UPDATE CASCADE;
COMMIT;
```

checkUser.js

```
$(document).ready(function() {
var checking_html='Идет проверка...';
$('#check_username_availability').click(function() {
$('#username_availability_result').html(checking_html);
check_availability();
});
});
function check_availability() {
var username=$('#username').val();
// use ajax to run the check
$.post("checkUser.php", { username : username },
function(result) {
if(result==1) {
$('#username_availability_result').html(username + ' уже существует');
} else {
$('#username_availability_result').html(username + ' доступен для пользования');
}
});
}
```

Login.php

```
include_once 'dbconnect2.php';
   $sql = mysqli_query($con,"UPDATE book
         SET paid = '1'
         WHERE username = '$user'");
mysqli_close($con);
?>
   </div>
  </div>
</div>
<footer class="container-fluid text-center">
      <a href="#signUpPage" title="To Top">
         <span class="glyphicon glyphicon-chevron-up"></span>
</a>
      <p>Air-Sultan</p>
</footer>
```

SearchResultRoundtrip.php

```
<div class="container-fluid text-center">
  <div class="row content">
```

```
    <div class="col-sm-2 sidenav">
    </div>
    <div class="col-sm-8 text-left">
      <h1>Поиск результатов</h1>
<?php
include_once 'dbconnect2.php';
if (!$con) {
  die('Could not connect: ' . mysqli_error($con));
}
function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
$from = test_input($_POST["from"]);
$to = test_input($_POST["to"]);
$depart = $_POST["depart"];
$return = $_POST["return"];
$class = $_POST["class"];
global $sql,$sql2, $availableNumber;
    //search by code only, non-stop
    $sql = "SELECT FL.number AS FLnumber, company, type, departure, d_time, arrival, a_time,
C.name AS classname, capacity, price
          FROM flight FL,  class C, airplane AP , airport A
          WHERE (FL.number = C.number) AND (FL.airplane_id = AP.ID) AND C.name =
'$class' AND
          ((departure = '$from') AND (arrival = '$to'))
          GROUP BY FL.number
          ORDER BY FL.number";
    $result = mysqli_query($con,$sql);
    $rowcount = mysqli_num_rows($result);
    $sql2 = "SELECT FL.number AS FLnumber, company, type, departure, d_time, arrival,
a_time, C.name AS classname, capacity, price
          FROM flight FL,  class C, airplane AP , airport A
          WHERE (FL.number = C.number) AND (FL.airplane_id = AP.ID) AND C.name =
'$class' AND
          ((departure = '$to') AND (arrival = '$from'))
          GROUP BY FL.number
          ORDER BY FL.number";
    $result2 = mysqli_query($con,$sql2);
     $rowcount2 = mysqli_num_rows($result2);

    $rowtotal = $rowcount*$rowcount2;
    if($rowtotal == 0){
       echo "<div class='alert alert-info'><strong>Совпадений: </strong>".$rowtotal."
результатов</div>";
    }
          $seatreserved4 = "SELECT flightno, classtype, COUNT(*)
```

```
                FROM book B
                WHERE B.date = '".$return."' AND B.flightno = '".$row2['FLnumber']."'AND
B.classtype ='".$row2['classname']."' AND paid=1
                GROUP BY flightno, classtype";
        $reserved4 = mysqli_query($con, $seatreserved4);
        $reservedNumber4 = mysqli_fetch_array($reserved4);
```

Register.php

```
<?php
session_start();
if(isset($_SESSION['user'])!="")
{
 header("Location: homepage.html");
}
include_once 'dbconnect.php';

// if(isset($_POST['btn-signup']))
// {
 $username = $_POST['username'];
 $firstname =$_POST['firstname'];
  $lastname =$_POST['lastname'];
   $tel =$_POST['tel'];
 $email = $_POST['email'];
 $pwd1 = $_POST['pwd1'];

 if(isset($_POST['middlename'])){
        $middlename = $_POST['middlename'];
 }else{
        $middlename = "";
 }

  if(isset($_POST['gender'])){
        $gender = $_POST['gender'];

 }else{
        $gender = "";
 }

  if(isset($_POST['birthday'])){
        $birthday = $_POST['birthday'];
 }else{
        $birthday = "";
 }
 if(mysqli_query($conn,"INSERT INTO
passanger(username,email,password,firstname,lastname,cellphone,middlename,gender,birthday)
VALUES('$username','$email','$pwd1','$firstname','$lastname','$tel','$middlename','$gender','$bir
thday')"))
 {
```

```
 $_SESSION['user']=$username;

      header("Location: homepage.html");
       }
 else
 {
  ?>
      <script>alert('Ошибка регистрации');</script>
      <?php
 }
// }
mysqli_close($conn);
?>
```

Homepage.html

```
<div id="oneway">
                <form role="form" action="SearchResultOneway.php" method="post">
                  <div class="row">
                  <div class="col-sm-6">
                    <label for="from">Пункт отбытия:</label>
                                <select class="form-control" id="from" name="from" required>
                                 <option value="ALA">Алма-Ата</option>
                                 <option value="TSE">Нур-Султан</option>
                                 <option value="SCO">Актау</option>
                                 <option value="AKX">Актюбинск</option>
                                 <option value="AYK">Аркалык</option>
                                 <option value="ATX">Атбасар</option>
                                 <option value="GUW">Атырау</option>
                                 <option value="AGZ">Аягуз</option>
                                 <option value="BXH">Балхаш</option>
                                 <option value="BXJ">Боралдай</option>
                                 <option value="DZN">Жезказган</option>
                                 <option value="KGF">Караганда</option>
                                 <option value="KOV">Кокшетау</option>
                                 <option value="KSN">Костанай</option>
                                 <option value="PWQ">Павлодар</option>
                                 <option value="PPK">Петропавловск</option>
                                 <option value="PLX">Семей</option>
                                 <option value="TDK">Талдыкорган</option>
                                 <option value="DMB">Тараз</option>
                                 <option value="URA">Уральск</option>
                                 <option value="UKK">Усть-каменогорск</option>
                                 <option value="CIT">Шымкент</option>
                                 <option value="EKB">Экибастуз</option>
                          </select>

                  </div>
                  <div class="col-sm-6">
                    <label for="to">Пункт прибытия:</label>
```

```
                <select class="form-control" id="to" name="to" required>
                        <option value="TSE">Нур-Султан</option>
                        <option value="ALA">Алма-Ата</option>
                        <option value="SCO">Актау</option>
                        <option value="AKX">Актюбинск</option>
                        <option value="AYK">Аркалык</option>
                        <option value="ATX">Атбасар</option>
                        <option value="GUW">Атырау</option>
                        <option value="AGZ">Аягуз</option>
                        <option value="BXH">Балхаш</option>
                        <option value="BXJ">Боралдай</option>
                        <option value="DZN">Жезказган</option>
                        <option value="KGF">Караганда</option>
                        <option value="KOV">Кокшетау</option>
                        <option value="KSN">Костанай</option>
                        <option value="PWQ">Павлодар</option>
                        <option value="PPK">Петропавловск</option>
                        <option value="PLX">Семей</option>
                        <option value="TDK">Талдыкорган</option>
                        <option value="DMB">Тараз</option>
                        <option value="URA">Уральск</option>
                        <option value="UKK">Усть-каменогорск</option>
                        <option value="CIT">Шымкент</option>
                        <option value="EKB">Экибастуз</option>
                    </select>
</div>
                </div>
                <br>
                <div class="row">
                <div class="col-sm-6">
                  <label class="radio-inline">
                    <input type="radio" name="stop" value="nonstop" checked>Без пересадок
                  </label>
                  <label class="radio-inline">
                    <input type="radio" name="stop" value="1stop">С пересадкой
                  </label>
                </div>
                </div>
                <br>
                <div class="btn-group btn-group-justified">
                        <div class="btn-group">
                                <button type="submit" class="btn btn-
success">Подтвердить</button>
                        </div>
                        <div class="btn-group">
                                <button type="reset"  class="btn btn-info"
value="Reset">Сбросить</button>
                        </div>
                </div>
```

```
                </form>

        </div>

if(!isset($_SESSION['user'])){
   header("Location: customersignin.html");
}else{
   $user = $_SESSION['user'];


        if($type =="all" || $type =="onewaynonstop" ){

        $flightno = $_POST["flightno"];
        $class = $_POST["classtype"];
        $price = $_POST["price"];
        $date = $_POST["date"];

        $sql = "INSERT INTO book (time, date, flightno, username, classtype, paid)
                        VALUES ('$time', '$date', '$flightno', '$user', '$class', '0')";;
```

Приложение В
(обязательно)

Акты внедрения

Акт внедрения веб-приложения «Air-Sultan»

Настоящий Акт свидетельствует, что веб-приложение «Air-Sultan», разработанное Исамбергеновым Султаном, внедрено в ИП «Маханов».

Процесс внедрения проходил с 2 по 6 мая 2019 г.

Заявленные характеристики системы предполагали наличие следующих основных возможностей:

- регистрация и авторизация пользователей;
- функция поиска рейсов различных типов;
- реализация корзины, истории броней;
- реализация панели менеджера.

В ходе опытной эксплуатации веб-приложения подтверждено, что оно обладает всеми заявленными возможностями.

На момент подписания настоящего Акта система установлена и эксплуатируется сотрудниками данной компании. Программой пользуются 10 человек.

Генеральный директор