МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 01 01 «Программное обеспечение информационных

технологий»

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

Web-приложение «Авиакомпания»

Выполнил студент Гречуха Максим Юрьевич

(Ф.И.О.)

Руководитель проекта преп.-стаж. Качаловский А.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты преп.-стаж. Качаловский А.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер преп.-стаж. Качаловский А.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2019

**Содержание**

[Введение 3](#_Toc10389938)

[1 Постановка задачи 4](#_Toc10389939)

[1.1 Алгоритмы решения 4](#_Toc10389940)

[1.2 Обзор прототипа 4](#_Toc10389941)

[2 Разработка архитектуры проекта 5](#_Toc10389942)

[2.1 Обобщенная структура 5](#_Toc10389943)

[2.2 Взаимосвязь всех компонентов 5](#_Toc10389944)

[3 Разработка функциональной модели и модели данных ПС 6](#_Toc10389945)

[3.1 Модель базы 6](#_Toc10389946)

[3.2 Выполняемые функции 6](#_Toc10389947)

[4 Руководство пользователя 9](#_Toc10389948)

[5 Тестирование 15](#_Toc10389949)

[Заключение 16](#_Toc10389950)

[Список используемой литературы 17](#_Toc10389951)

[ПРИЛОЖЕНИЕ A 18](#_Toc10389952)

# Введение

В современном информационном обществе покупки через Интернет становятся популярнее с каждым днем. Причина в том, что можно заказать желаемый товар или купить билет на любой вид транспорта в любую точку мира всего за пару кликов, находясь дома. Этот метод покупок очень удобен для жителей городов, удаленных от аэропортов. Стремительный рост количества интернет-пользователей, социальные сети, развитие систем электронных платежей за последние 20 лет позволили Интернет-коммерции стать очень популярным способом покупок.

На данный момент интернет-магазины авиабилетов представляют собой новую, но уже устоявшуюся систему реализации данных услуг. По сути, эта система является более эффективной по сравнению со стандартной «физической» кассой в аэропорту. Пара кликов ­– билет уже у вас в корзине, без всяких очередей. Интернет-магазины подобного рода направленны на миллионную аудиторию. Они позволяют улучшить качество обслуживания клиентов, так как все действия по покупке товаров производятся в автоматическом режиме.

Пояснительная записка курсового проекта состоит из 23 страниц, 16 рисунков, 1 приложение, 9 источников литературы.

Основная цель курсового проекта: создание сайта авиакомпании.

Пояснительная записка состоит из введения, 5 глав и заключения.

В первом разделе рассматривается постановка задачи.

Во втором разделе предоставлена информация об архитектуре самого приложения и базы данных.

В третьем разделе описываются разработка функциональной модели и модели данных ПС.

Четвертый раздел содержит руководство пользователя для разработанного клиентского приложения.

Пятый раздел содержит информацию о тестировании.

В заключении описывается результат работы над курсовым проектом, выполненные цели.

1. **Постановка задачи**

Постановка задач - один из самых важных этапов в работе программиста. Наверное, это даже самый важный пункт после планирования. Во многом от того, насколько правильно будут поставлены задачи, зависит то, насколько будут достигнуты поставленные цели.

## **Алгоритмы решения**

При реализации курсового проекта использовались технологии ASP.Net CORE для реализации back-end, для front-end части HTML — стандартизированный язык разметки документов. Большинство веб-страниц содержат описание разметки на языке HTML со вставками языка Razor. Язык HTML интерпретируется браузерами, полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

## **Обзор прототипа**

В качестве прототипа приложения был выбран сайт <https://ibe.belavia.by> (рисунок 1.1). Этот сайт предназначен для бронирования и покупки авиабилетов. У него есть возможность выбора рейса из одного города в другой на необходимое число, чтобы пользователь мог сократить для себя время поиска нужной перелета. А также оформления бронирования онлайн.

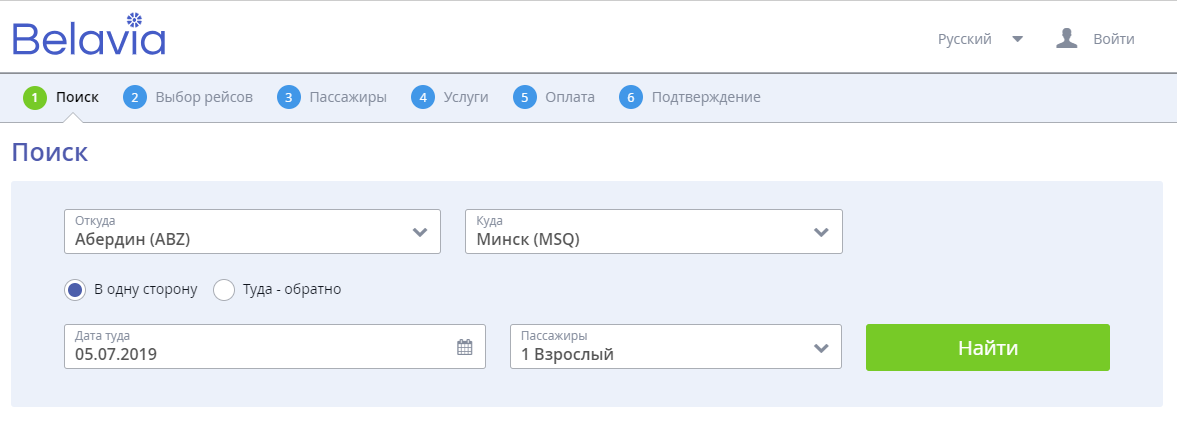
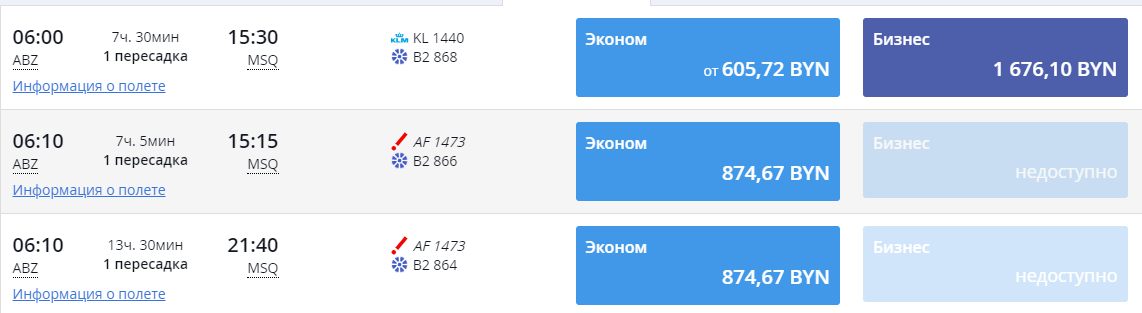
 

Рисунок 1.1 – Сайт <https://ibe.belavia.by>

Сайт предоставляет удобный интерфейс для взаимодействия с пользователем. Есть возможность регистрации аккаунтов, для того чтобы пользователь мог хранить в одном месте все свои бронирования и билеты.

1. **Разработка архитектуры проекта**

Архитектура — это базовая организация системы, воплощенная в ее компонентах, их отношениях между собой и с окружением, а также принципы, определяющие проектирование и развитие системы.

## **Обобщенная структура**

Курсовой проект построен на многоуровневой N-layer архитектуре. Данная архитектура основана на следующих принципах:

1. Проектирование чётко устанавливает разграничение функций между уровнями;
2. Нижние уровни независимы от верхних уровней;
3. Верхние уровни вызывают функции нижних уровней, но при этом взаимодействуют только соседние уровни иерархии.

Для данного курсового проекта были разработаны следующие уровни:

1. Уровень «Модель» – описание таблиц в базе данных;
2. Уровень «Представление» – отвечает за отображение данных модели пользователю, реагируя на изменения модели;
3. Уровень «Контроллер» – описание классов, методы которых предназначены для создания ответа пользователю по его запросу.

## **Взаимосвязь всех компонентов**

В решении курсового проекта была использована n-layer архитектура (рисунок 2.2). Эта архитектура выбрана так как она идеально подходит для приложений с длительным жизненным циклом и сложной бизнес логикой.

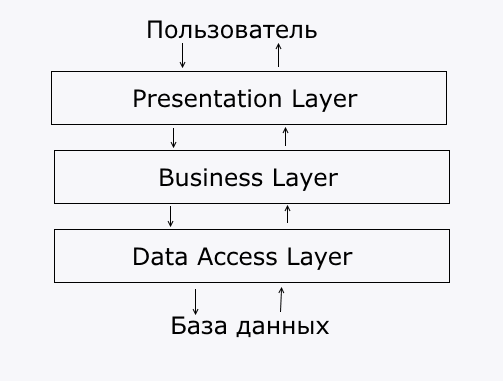


Рисунок 2.2 – Общая схема N-layer архитектуры

1. **Разработка функциональной модели и модели данных ПС**

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь.

## **Модель базы**

Для базы данных Книжного магазина было разработано 5 таблиц (рисунок 3.1). База данных была разработана в СУБД MS SQL Server.

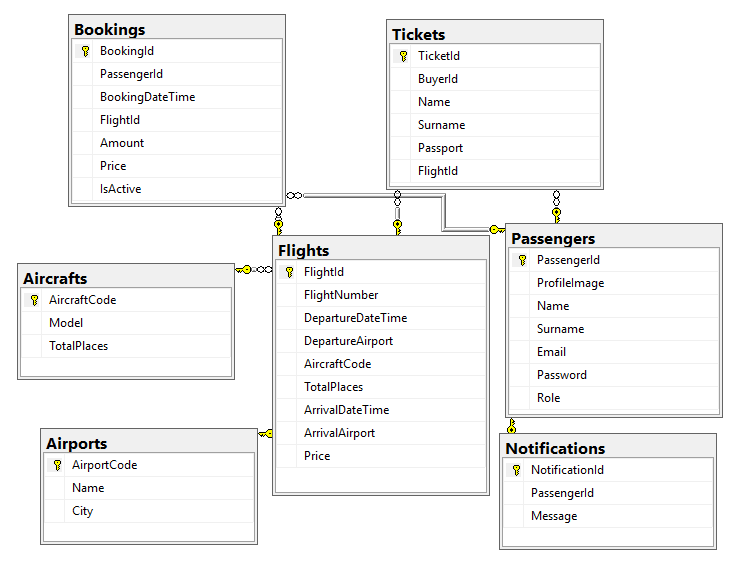


Рисунок 3.1 – Структура базы данных

## **Выполняемые функции**

Таблица Passengers содержит информацию о пользователях системы:

* PassengerId – первичный ключ, идентификатор пользователя;
* ProfileImage – фотография профиля;
* Name– имя пользователя;
* Surname – фамилия пользователя
* Email – электронная почта;
* Password – пароль пользователя;
* Role – роль пользователя в системе.

Таблица Notifications содержит информацию об уведомлениях:

* NotificationId – первичный ключ, идентификатор уведомления;
* PassengerId – идентификатор пользователя;
* Message – текст уведомления;

Таблица Tickets содержит информацию о билетах:

* TicketId – первичный ключ, идентификатор билета;
* BuyerId – идентификатор пользователя, купившего билет;
* Name – имя пассажира;
* Surname – фамилия пассажира;
* Passport – номер пасспорта пассажира;
* FlightId – идентификатор перелета.

Таблица Bookings предназначена для хранения информации о бронированиях:

* BookingId– первичный ключ, идентификатор бронирования;
* PassengerId – идентификатор пользователя, забронировавшего билет;
* BookingDateTime – дата оформления бронирования;
* FlightId – идентификатор перелета;
* Amount – количество забронированных мест;
* Price – цена одного билета;
* IsActive – текущий статус бронирования.

Таблица Aircrafts предназначена для хранения информации о самолетах:

* AircraftCode – первичный ключ, код самолета;
* Model – модель самолета;
* TotalPlaces – вместимость пассажиров.

Таблица Airports предназначена для хранения информации о аэропортах:

* AirportCode– первичный ключ, код аэропорта;
* Name – название аэропорта;
* City – город.

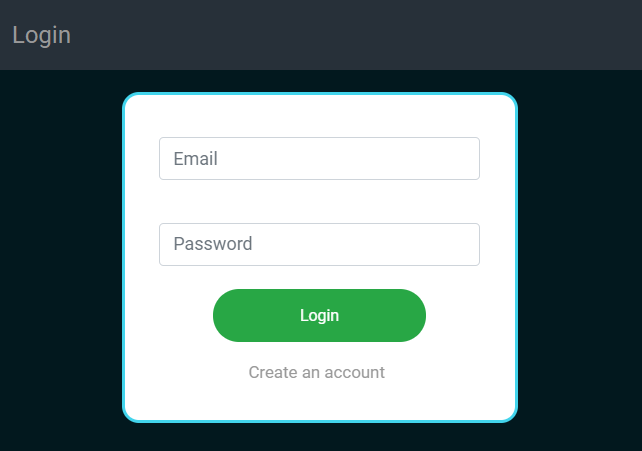
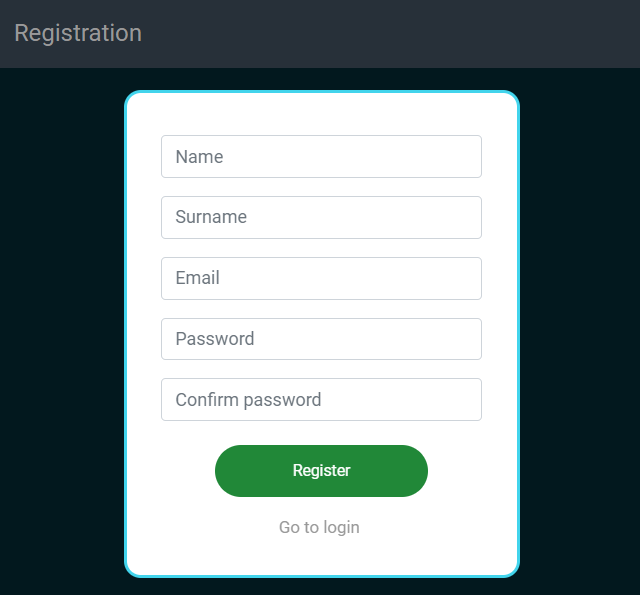
Таблица Flights предназначена для хранения информации о перелетах:

– FlightId– первичный ключ, идентификатор перелета;

* FlightNumber– номер перелета;
* DepartureDateTime– дата и время вылета;
* DepartureAirport– аэропорт отправления;
* AircraftCode– самолет;
* TotalPlaces– количество мест;
* ArrivalDateTime – дата и время прибытия;
* ArrivalAirport – аэропорт прибытия;
* Price – цена перелета.

1. **Руководство пользователя**

На сайте присутствует возможность войти в свой аккаунт и зарегистрироваться (рисунок 4.1)

При загрузке сайта отображается начальная страница (рисунок 4.2)

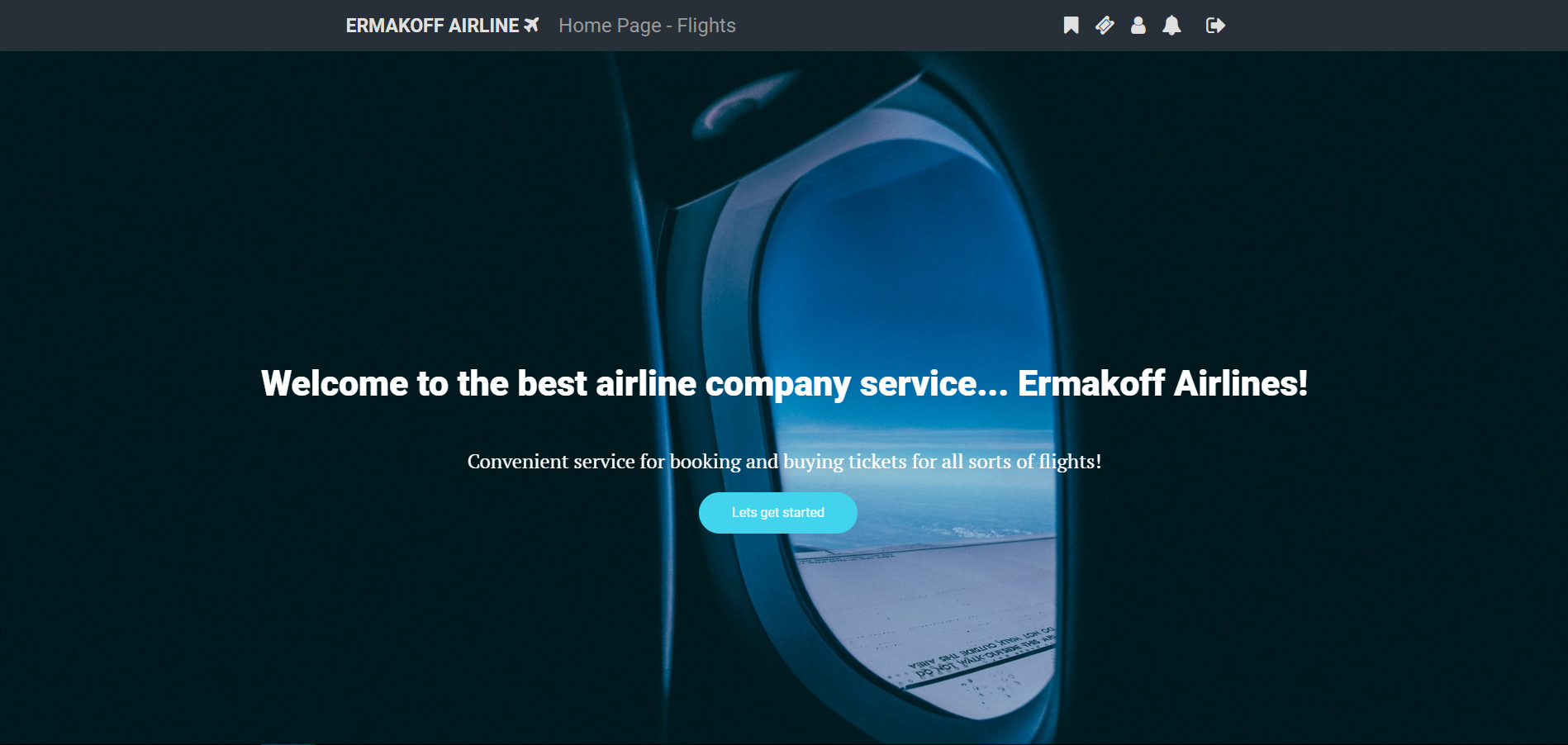
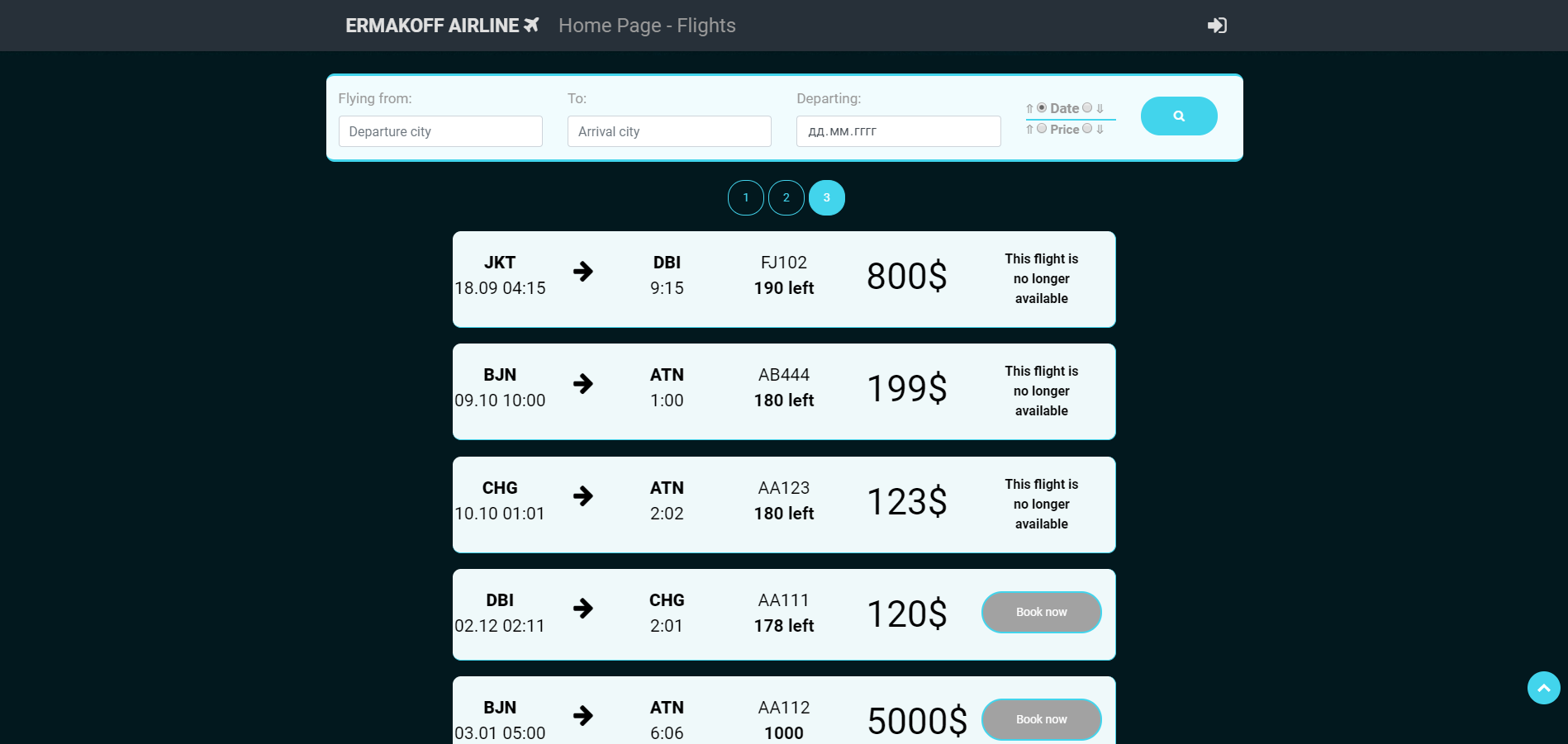


Рисунок 4.2 – Стартовое окно

После нажатия на кнопку «Let’s get started» анимация переместит пользователя к блоку поиска перелетов с сортировкой по дате и по цене, под которым отображаются нужные рейсы со всей необходимой информацией и кнопки пагинации (рисунок 4.3).

 Рисунок 4.3 – Поиск и доступные перелеты с пагинацией

При нажатии на кнопку «Book now» на нужной рейсе, пользователь попадает на страницу непосредственного бронирования, где он может выбрать необходимое ему число билетов, но не больше 5 (рисунок 4.4).

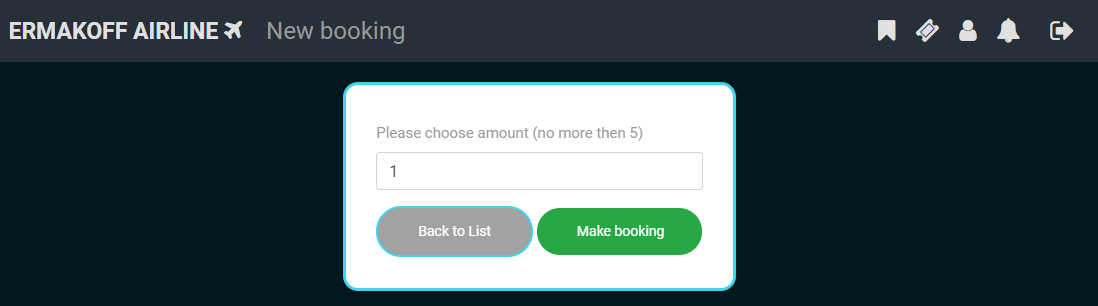
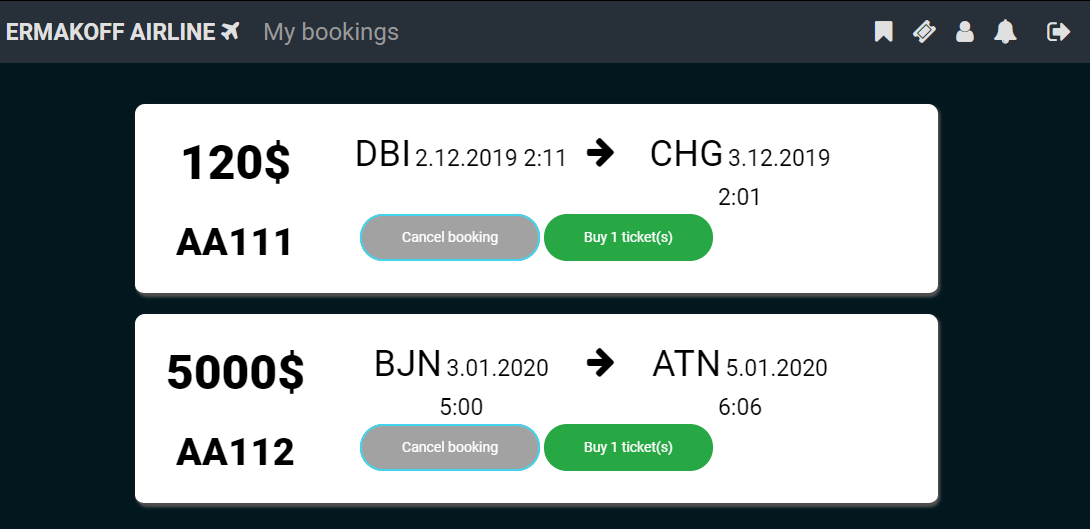


Рисунок 4.4 – Бронирование билета

После нажатия на кнопку «Make booking» создается бронирование и пользователь автоматически перенаправляется на страницу своих бронирований, где так же есть пагинация (рисунок 4.5).

 Рисунок 4.5 – Активные бронирования пользователя

На каждом бронировании есть две кнопки отмены бронирования и покупки билета, при нажатии на первую бронирование пропадет из списка, а при на вторую пользователь будет перенаправлен на страницу оформления, для ввода информации о пассажире и платежных данных (рисунок 4.6)

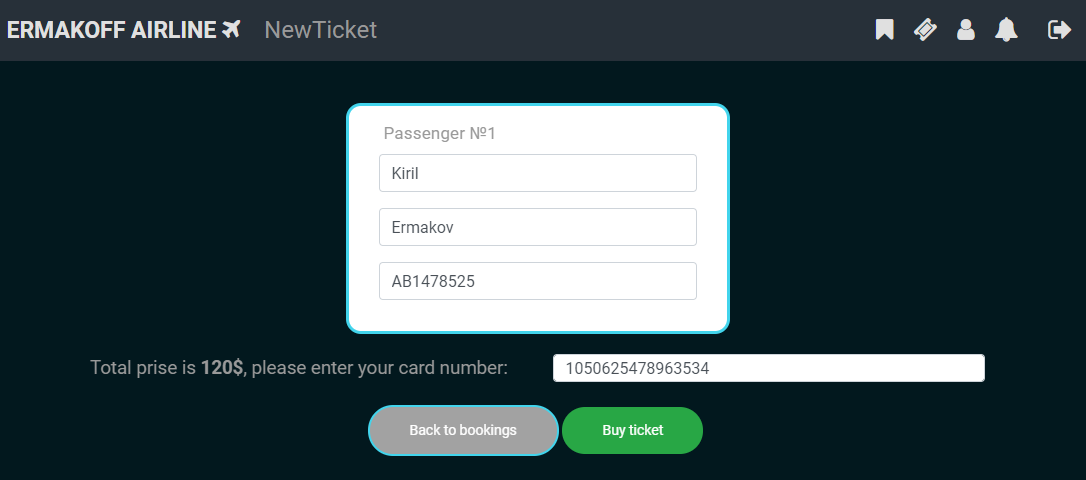


Рисунок 4.6 – Оформление билета

После заполнения всех полей и нажатия на кнопку «Buy ticket» мы попадаем на страницу всех билетов, где можем вернуть любой билет, после чего он снова отобразится на странице бронирований (рисунок 4.7).

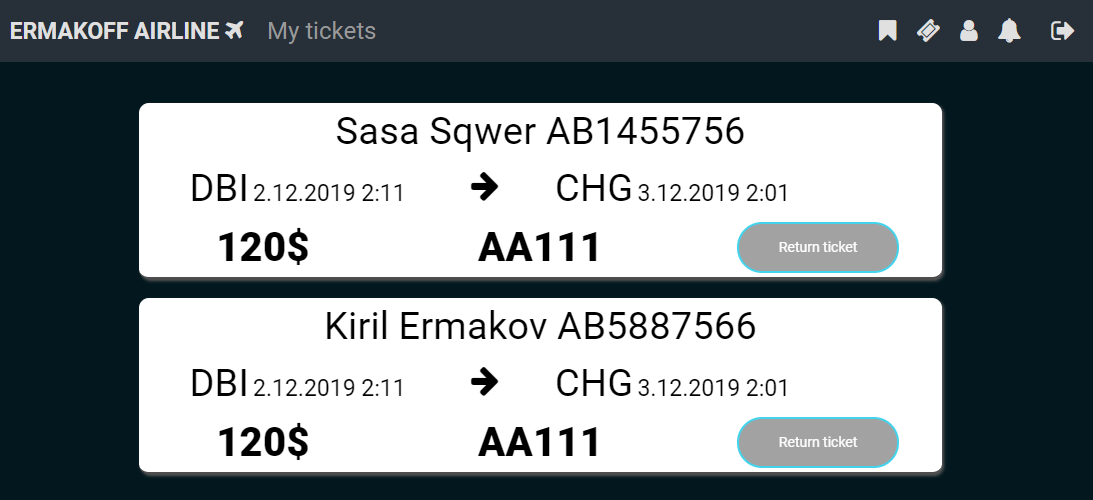


Рисунок 4.7 – Все купленные билеты

На сайте имеется страница профиля где пользователь может изменить свое имя и фамилию или обновить пароль, а также изменить свою фотографию профиля (рисунок 4.8).

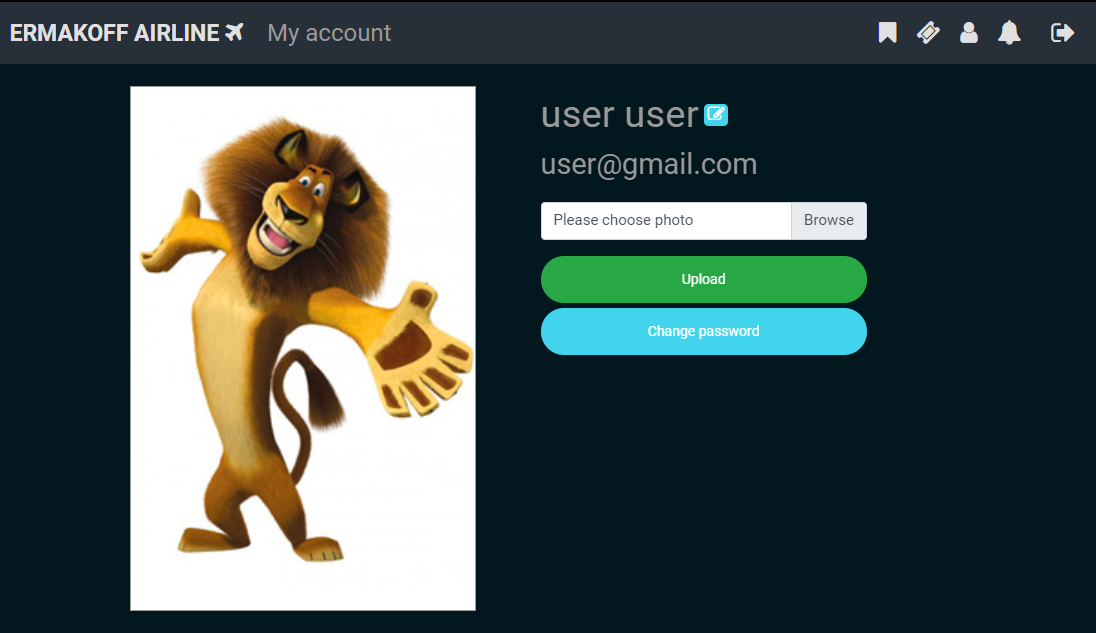


Рисунок 4.8 – Страница профиля

Нажав на колокольчик, пользователь может увидеть свои уведомления, которые можно удалить нажав на крестик в углу (рисунок 4.9).

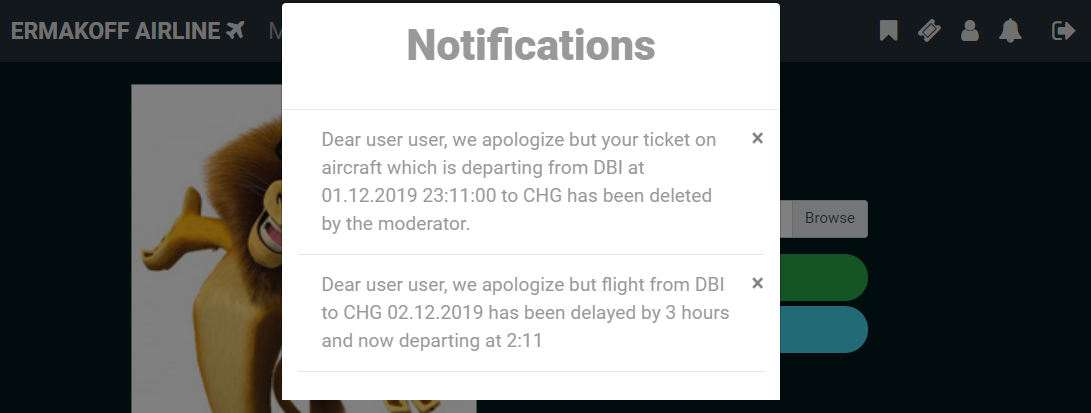


Рисунок 4.9 – Уведомления

На этом заканчивается часть обычного пользователя. Далее рассмотрим часть модератора или администратора, когда администратор заходит на сайт у него появляется особая вкладка под названием панель администратора, с помощью которой модератор может управлять объектами системы, например перелетами (рисунок 4.10).

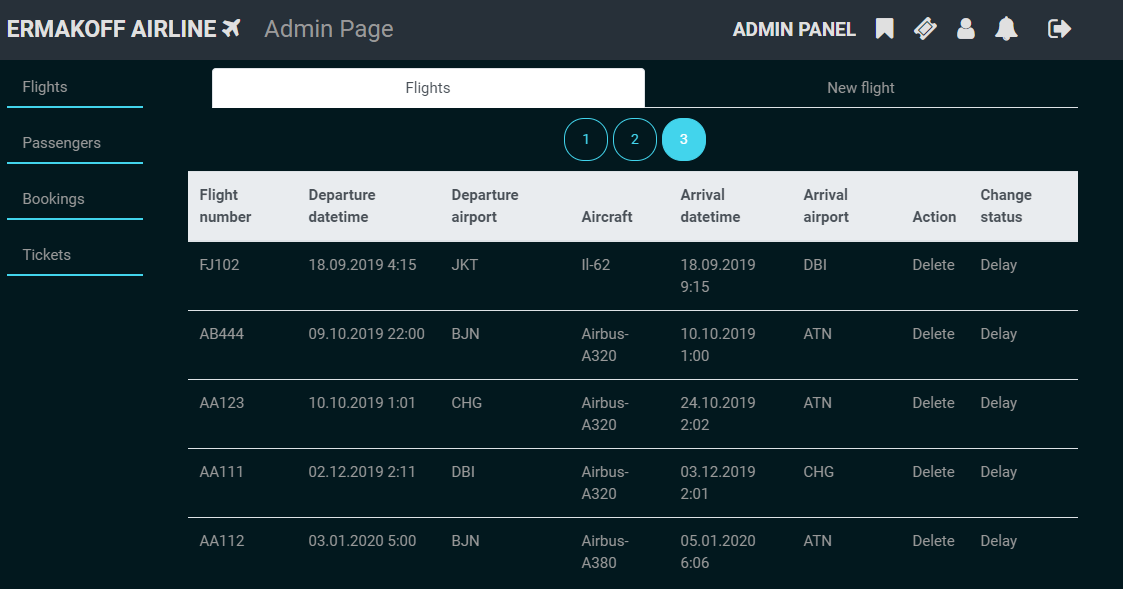


Рисунок 4.10 – Панель модератора, перелеты

Во вкладке «New flight» администратор может добавлять новые перелеты (рисунок 4.11).

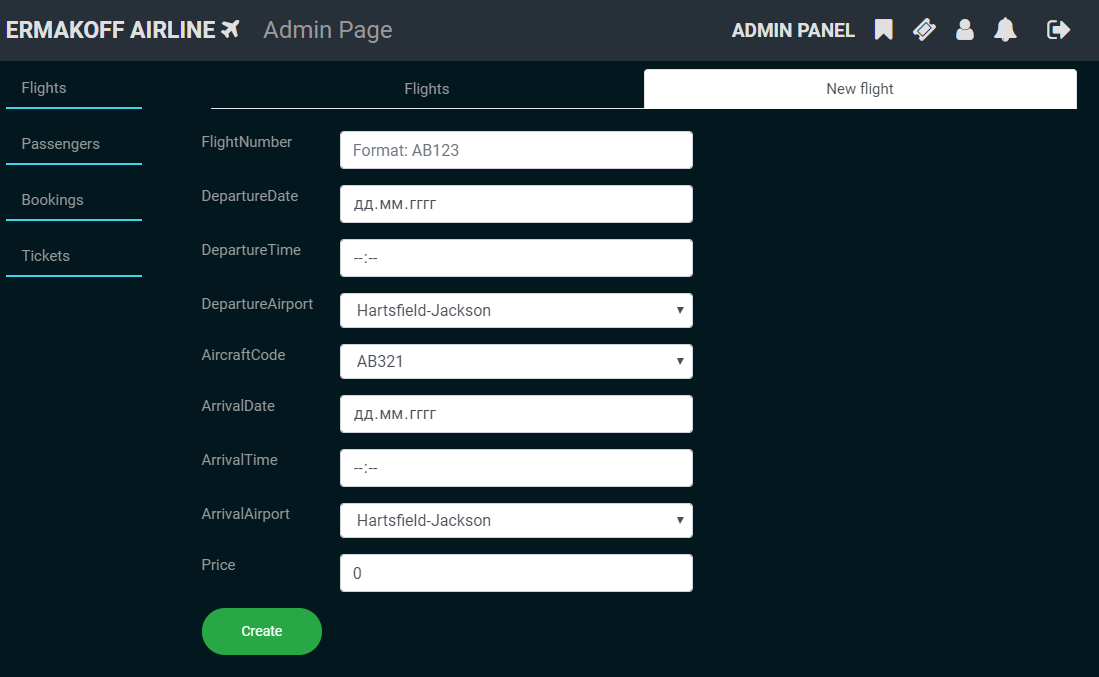
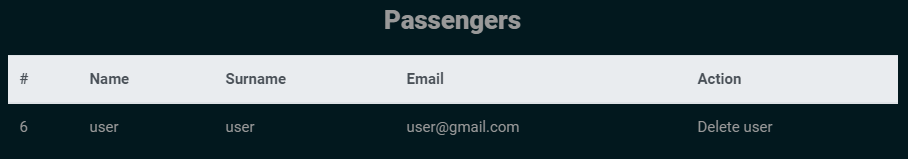
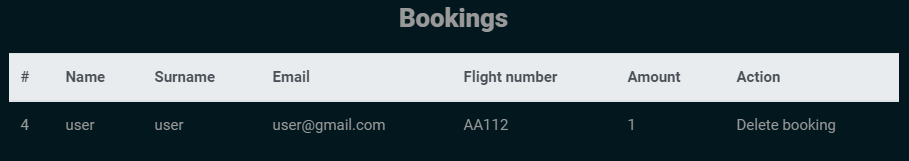


Рисунок 4.11 – Панель модератора, добавление перелета

Так же в панели администратора имеются вкладки для просмотра и удаления всех пользователей, бронирований и билетов (рисунок 4.12).





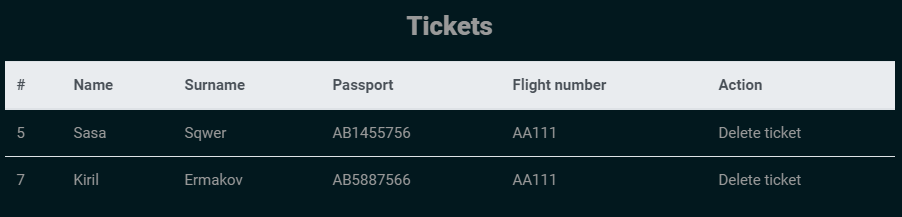


Рисунок 4.12 – Панель модератора: пассажиры, бронирования и билеты

1. **Тестирование**

На основную логику проекта были разработаны и успешно выполнены интеграционные тесты. Интеграционные тесты дают уверенность, что ваша программа работает как задумано. Такие тесты можно запускать многократно. Успешное выполнение тестов покажет разработчику, что его изменения не сломали ничего, что ломать не планировалось.

Провалившийся тест позволит обнаружить, что в коде сделаны изменения, которые меняют или ломают его поведение. Исследование ошибки, которую выдает провалившийся тест, и сравнение ожидаемого результата с полученным даст возможность понять, где возникла ошибка, будь она в коде или в требованиях. На рисунке 5.1. представлены результаты выполненных тестов.

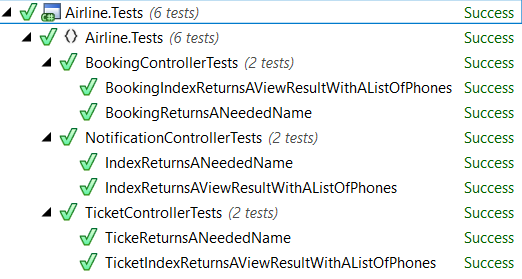


Рисунок 5.1 – Результаты выполнения тестов

Также в проекте реализована валидация на случай неочевидного или неправильного поведения пользователя.

В данном курсовом проекте валидируются все поля ввода, например поля для ввода адреса электронной почты, имени и фамилии, паспортных данных, номера банковской карты, на которых в случае некорректного ввода, может произойти ошибка или непредвиденное поведение программы. Валидация приложения – это один из основных гарантов надёжности приложения и всякий программист должен предусмотреть и предотвратить непредвиденное поведение пользователя.

# Заключение

Разработка приложения – трудоемкое занятие, выполнение которого требует всестороннего знания той предметной области, к которой относится тема разрабатываемого приложения.

В процессе выполнения курсового проекта были разработаны структура и алгоритм работы приложения «Авиакомпания». Программное обеспечение написано на языке c# платформе Asp.net Core с использованием среды разработки Visual Studio и базы данных Microsoft SQL Server. Был проведен комплекс тестирования в составе программы «Авиакомпания», в результате которого были устранены некоторые недостатки. Все поставленные задачи были выполнены. Приложение предоставляет приятный, понятный и удобный интерфейс для обслуживающего администратора магазина, а также простых посетителей.

В результате выполнения курсового проекта можно сделать вывод, что сегодня внедрение информационных систем может способствовать получению более рациональных вариантов решения управленческих задач за счет внедрения интеллектуальных систем. Эффективность от использования данной информационной системы, достигается за счет замены обычных касс с огромными очередями на удобный интерфейс онлайн сервиса, что приводит к более рациональной организации переработки информации на компьютере и снижению объемов человеческого труда. Освобождение сотрудников от рутинной работы, за счет ее автоматизации, и повышения скорости обработки пользовательских запросов.

Для удалённого доступа пользователей приложение было опубликовано на Azure. Ссылка перехода: <https://mixerairlinewebsite.azurewebsites.net>

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# Список используемой литературы

1. Цыганенко, Н. П. Курс лекций по предмету «Программирование серверных кроссплатформенных приложений» - 2019.
2. N-layer архитектура [Электронный ресурс] / Github – Режим доступа: <https://accetone.github.io/cwp/10/#/> - Дата доступа: 16.04.2019.
3. ASP.NET Core документация [Электронный ресурс] – режим доступа https://docs.microsoft.com/en-us/aspnet/core/?view=as.. Дата доступа 15.03.2019
4. ASP.NET Core tutorials [Электронный ресурс] – режим доступа https://www.tutorialspoint.com/asp.net\_core/ Дата доступа 01.04.2019
5. Дизайн API [Электронный ресурс] – режим доступа https://docs.microsoft.com/en-us/azure/architecture/b.. Дата доступа 05.04.2019
6. Архитектуры WEB приложений [Электронный ресурс] – режим доступа https://docs.microsoft.com/en-us/dotnet/standard/mode.. Дата доступа 12.04.2019
7. Entity Framework Core [Электронный ресурс] – режим доступа https://docs.microsoft.com/en-us/ef/core/ Дата доступа 01.04.2019
8. Deploy To Azure [Электронный ресурс] – режим доступа https://docs.microsoft.com/en-us/aspnet/core/tutorial.. Дата доступа 25.03.2019
9. ASP.NET Core MVC [Электронный ресурс] – режим доступа https://docs.microsoft.com/en-us/aspnet/core/mvc/over.. Дата доступа 20.03.2019

# ПРИЛОЖЕНИЕ A

Листинг 1. Бизнес-логика приложения

|  |
| --- |
| [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Login(LoginViewModel model)  {  if (!ModelState.IsValid)  return View(model);  var redirectUrl = HttpContext.Session.GetString("RedirectUrl");  var logmodel = \_passengerService.Login(model);  if (logmodel.Message != null)  return View(logmodel);  await Authenticate(model.Email);  if (redirectUrl != null)  {  HttpContext.Session.Remove("RedirectUrl");  return Redirect(Convert.ToString(redirectUrl));  }  return RedirectToAction("Index", "Home");  }  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Registration(RegistrationViewModel model)  {  if (!ModelState.IsValid)  return View(model);  var redirectUrl = HttpContext.Session.GetString("RedirectUrl");  var regmodel = \_passengerService.Registration(model);  if (regmodel.Message != null)  return View(regmodel);  await Authenticate(model.Email);  if (redirectUrl != null)  {  HttpContext.Session.Remove("RedirectUrl");  return Redirect(Convert.ToString(redirectUrl));  }  ViewBag.Title = "My account";  return RedirectToAction("Index", "Home");  }  private async Task Authenticate(string userName)  {  var claims = new List<Claim>  {  new Claim(ClaimsIdentity.DefaultNameClaimType, userName),  new Claim(ClaimsIdentity.DefaultRoleClaimType, \_passengerService.GetRole(userName))  };  ClaimsIdentity id = new ClaimsIdentity(claims, "ApplicationCookie", ClaimsIdentity.DefaultNameClaimType,  ClaimsIdentity.DefaultRoleClaimType);  await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(id));  }  public async Task<IActionResult> Logout()  {  await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);  return RedirectToAction("Index", "Home");  }  [HttpGet]  [Authorize]  public ActionResult ChangePassword()  {  var editUser = new ChangePasswordViewModel() { Id = \_passengerService.GetByEmail(User.Identity.Name).PassengerId };  return View(editUser);  }  [HttpPost]  [Authorize]  public ActionResult ChangePassword(ChangePasswordViewModel model)  {  if (!ModelState.IsValid)  return View(model);  \_passengerService.ChangePassword(model);  return RedirectToAction("MyAccount");  }  [HttpPost]  public ActionResult Upload(PassengerViewModel model, IFormFile photo)  {  if (photo == null)  ModelState.AddModelError(string.Empty, "Please choose file");  else  {  byte[] imageData = null;  using (var binaryReader = new BinaryReader(photo.OpenReadStream()))  {  imageData = binaryReader.ReadBytes((int)photo.Length);  }  model.ProfileImage = imageData;  \_passengerService.UploadProfileImage(\_passengerService.GetByEmail(User.Identity.Name).PassengerId, imageData);  }  return RedirectToAction("MyAccount");  }  [HttpGet]  [Authorize]  public ActionResult EditInfo()  {  var passenger = \_passengerService.GetById(\_passengerService.GetByEmail(User.Identity.Name).PassengerId);  var editUser = new EditUserInfoViewModel()  {  Id = passenger.PassengerId,  Name = passenger.Name,  Surname = passenger.Surname  };  return View(editUser);  }  [HttpPost]  [Authorize]  public ActionResult EditInfo(EditUserInfoViewModel model)  {  if (!ModelState.IsValid)  return View(model);  \_passengerService.EditInfo(model);  return RedirectToAction("MyAccount");  }  [HttpGet]  [Authorize]  public ActionResult DeleteNotification(int notificationId)  {  \_notificationService.DeleteNotification(notificationId);  return Ok();  }  [HttpPost]  public ActionResult NewBooking(NewBookingViewModel model)  {  var passengerId = \_passengerService.GetByEmail(User.Identity.Name).PassengerId;  var booking = \_bookingService.NewBooking(passengerId, model.FlightId, model.Amount);  if (booking.Message == null)  {  return RedirectToAction("Index");  }  else  model.Message = booking.Message;  return View(model);  }  public ActionResult CancelBooking(int bookingId)  {  \_bookingService.IsThereAccess(\_passengerService.GetByEmail(User.Identity.Name).PassengerId, bookingId);  \_bookingService.CancelBooking(bookingId);  return RedirectToAction("Index");  }  [HttpGet]  public ActionResult Index(bool? backToList)  {  var model = new FlightPaginationModel();  var str = HttpContext.Session.GetString("LastSearch");  if (backToList != null && str != null)  model = JsonConvert.DeserializeObject<FlightPaginationModel>(str);  else  {  var flightList = new PagedList<FlightViewModel>();  flightList.PageNumber = 0;  flightList.PageSize = Amount;  model.FlightList = flightList;  model = \_flightService.GetNeededFlightsOffset(model);  HttpContext.Session.SetString("LastSearch", JsonConvert.SerializeObject(model));  }  return View(model);  }  [HttpPost]  public ActionResult GetFlightsOffset(FlightPaginationModel model)  {  if (!ModelState.IsValid)  {  return View(model);  }  var flightList = new PagedList<FlightViewModel>();  flightList.PageNumber = model.PageNumber;  flightList.PageSize = Amount;  model.FlightList = flightList;  model = \_flightService.GetNeededFlightsOffset(model);  if (model.FlightList.RowsCount == 0)  {  model.Date = null;  model = \_flightService.GetNeededFlightsOffset(model);  model.FlightList.Message = "There are no flights between these cities that day";  }  HttpContext.Session.SetString("LastSearch", JsonConvert.SerializeObject(model));  return PartialView("FlightPartialView", model);  }  public ActionResult Index()  {  var ticketList = new PagedList<TicketViewModel>();  ticketList.PageNumber = 0;  ticketList.PageSize = Amount;  var model = new PaginationModel();  model.UserId = \_passengerService.GetByEmail(User.Identity.Name).PassengerId;  model.TicketList = ticketList;  model = \_ticketService.GetTicketsOffset(model);  return View(model);  }  public PartialViewResult GetTicketsOffset(PaginationRequest number)  {  var ticketList = new PagedList<TicketViewModel>();  ticketList.PageNumber = number.PageNumber;  ticketList.PageSize = Amount;  var model = new PaginationModel();  model.UserId = \_passengerService.GetByEmail(User.Identity.Name).PassengerId;  model.TicketList = ticketList;  model = \_ticketService.GetTicketsOffset(model);  return PartialView("~/Views/Ticket/TicketPartialView.cshtml", model);  }  [HttpGet]  public ActionResult NewTicket(int bookingId)  {  \_bookingService.IsThereAccess(\_passengerService.GetByEmail(User.Identity.Name).PassengerId, bookingId);  var booking = \_bookingService.GetById(bookingId);  var model = new NewTicketViewModel  {  FlightId = booking.FlightId,  Amount = booking.Amount,  TotalPrice = booking.Amount \* Convert.ToInt32(booking.Price)  };  for (int i = 0; i < model.Amount; i++)  model.Passengers.Add(new PassengerInfo());  return View(model);  }  [HttpPost]  public ActionResult NewTicket(NewTicketViewModel model)  {  var passengerId = \_passengerService.GetByEmail(User.Identity.Name).PassengerId;  foreach (var p in model.Passengers)  \_ticketService.NewTicket(passengerId, model.FlightId, p.Name, p.Surname, p.Passport);  return RedirectToAction("Index");  }  [HttpGet]  public ActionResult Delete(int ticketId, string passport)  {  \_ticketService.IsThereAccess(\_passengerService.GetByEmail(User.Identity.Name).PassengerId, ticketId);  \_ticketService.DeleteTicket(ticketId, passport);  return RedirectToAction("Index");  } |