МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01 Информационные системы и технологии

Направление специальности 1-40 01 02 03 Информационные системы

и технологии (издательско-полиграфический комплекс)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине Программирование в Internet

Тема: Web-приложение для заказа электронных билетов на самолет

Исполнитель

Студент 4 курса 2 группы Росляков Д.П.

(Ф.И.О.)

Руководитель работы ассистент Бернацкий П.В. (учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель ассистент Бернацкий П.В.

(подпись)

Минск 2022

**Содержание**

[Введение 3](#_Toc122465373)

[1 Постановка задачи и обзор аналогичных решений 4](#_Toc122465374)

[1.1 Постановка задачи 4](#_Toc122465375)

[1.2 Актуальность задачи 4](#_Toc122465376)

[1.3 Обзор Аналогов 4](#_Toc122465377)

[1.4 Вывод 9](#_Toc122465412)

[2 Проектирование программного средства 10](#_Toc122465413)

[2.1 Проектирование базы данных 10](#_Toc122465414)

[2.2 Проектирование UML диаграммы 11](#_Toc122465415)

[2.3 Проектирование основных алгоритмов 12](#_Toc122465416)

[2.4 Выбор средств реализации 14](#_Toc122465417)

[2.5 Вывод 15](#_Toc122465418)

[3 Разработка программного средства 16](#_Toc122465419)

[3.1 Разработка frontend части 16](#_Toc122465420)

[3.2 Разработка backend части 19](#_Toc122465421)

[3.3 Контейнеризация 22](#_Toc122465422)

[3.4 Вывод 25](#_Toc122465423)

[4 Тестирование 26](#_Toc122465424)

[4.1 Ручное тестирования frontend части 26](#_Toc122465425)

[4.2 Вывод 30](#_Toc122465426)

[5 Руководство пользователя 31](#_Toc122465427)

[5.1 Руководство пользователя 31](#_Toc122465428)

[5.2 Руководство администратора 32](#_Toc122465429)

[5.3 Вывод 34](#_Toc122465430)

[Заключение 35](#_Toc122465431)

[Список использованной литературы 36](#_Toc122465432)

[Приложение A. Листинг контроллера, отвечающего за обработку наличия прав администратора 37](#_Toc122465433)

[Приложение B. Листинг метода контроллера «TicketsController» 38](#_Toc122465434)

[Приложение C. Листинг кода, отвечающего за создание нового билета 39](#_Toc122465435)

**Введение**

Бум использования компьютерных технологий в последнее время все больше захватывает туристический рынок. Доказательство тому – стремительный рост новых агентств глобальных компьютерных систем бронирования Amadeus, Galileo, Sabre, Worldspan, Fidelio. Только за 2019 год эти компании практически удвоили число своих пользователей.

Объемы информации огромны и растут, с каждым годом, всё с большей скоростью. Поэтому в любой организации, как в большой или маленькой возникают проблемы с обработкой данных, это сильно сказывается на эффективности работы всего бизнеса и предприятия в целом. Основные идеи современной информационной технологии базируются на концепции, согласно которой данные должны быть организованы в базах данных (БД) с целью адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей.

Бизнес и предприятия диктующие современные требования к условиям качества работы, давно определили, что эффективная и прибыльное дело всецело зависит от уровня систем автоматизации компании.

В данном курсовом проекте целью является разработка приложения для заказа электронных билетов на самолет. К задачам следует отнести разработку REST приложения, которое инкапсулирует всю бизнес-логику данного приложения, а также создание веб-ресурса, позволяющего не только приобрести билет, но совершить его отмену. В качестве основных технологий были выбраны С# и EntityFramework для REST сервиса или backend части приложения, а для frontend части NodeJS, ReactJS, Reactstrap, React-bootstrap. Таким образом используется два стека для двух составных этой площадки. Преимуществом использования данного подхода к созданию в том, что оно имеет наибольший потенциал к горизонтальному и вертикальному расширению, так как основная логика изолирована в одном модуле, который представляет данные в максимально формальном виде, с которым могут работать множество технологий, следовательно есть возможность иметь неограниченное количество сервисом, которые будут представлять данные в любом виде будь то стилизованные HTML-документы и строго оформленная статистика.

# Постановка задачи и обзор аналогичных решений

Прежде чем приступать к проекту, нужно сначала выяснить, что он будет в себя включать и что будет сделано в итоге. Помочь в этом этапе помогут такие инструменты как, выделение основных подтем в выбранной области и ознакомление с аналогами, если таковые существуют.

# Постановка задачи

Целью данного проекта является создание доступного для каждого человека в зоне досягаемости ресурса, для заказа электронных билетов на самолет. Также следует отметить, что кто-то должны создаваться рейсы для людей, и для этой цели будет создана отдельная роль. Как следствие, над всей системой должен быть управляющей орган, которым является администрация данного приложения.

Таким образом к задачам следует отнести создание веб-приложения, позволяющего одним пользователям заказывать билет на самолет, а другим регистрировать рейсы.

Авиакомпания занимается перевозками пассажиров. Также она устанавливает маршруты полетов. Рейсы осуществляются по установленным маршрутам согласно расписанию. На каждый рейс существует определенное количество билетов.

Абсолютно каждая авиакомпания использует определенную систему дистрибуции. Наиболее развитые используют GDS (глобальные дистрибьюторские системы, которые формируются из основных международных компьютерных систем резервирования). В итоге сервисы продаж авиабилетов при поиске информации пользуются ресурсами глобальных дистрибьюторских систем. Однако доступ к GDS является не бесплатным, поэтому в роли дистрибутивной системы для разрабатываемого продукта будет выступать БД, созданная в MS SQL Server 2019.

# Актуальность задачи

Актуальность данной задачи обусловлена человеческой сущностью. Поскольку, человек – существо социальное, оно нуждается в общение, иначе могут последовать неприятные последствия в виде одиночества, депрессии, суицида. И, во избежание столь печального конца, будет разработано средство, которое, хоть и в малой степени, но способно повлияет на людей. Люди будут способны завести новые знакомства, а это значит и новые возможности, всем известен тот факт, что один человек может многое, а множество единомышленников может всё.

# Обзор Аналогов

Следует начать с обзора аналогов, так как разрабатываемый продукт должен взять лучшие стороны от всех, иначе он будет потерян среди других невзрачных середнячков выбранной ниши.

На данный момент существует огромное количество информационных систем, занимающихся продажами авиабилетов. Они могут представлять собой как самостоятельные приложения, так и онлайн сервисы, предоставляя пользователю доступ к веб-службам поставщиков. Как было сказано в введении данной работы в настоящее время в мире существуют несколько популярных GDS: Amadeus, Galileo, Sabre, Worldspan, Fidelio. Рассмотренные средства, в настоящее время существующие на рынке, приведены в таблице 1.1.

Таблица 1.1 – Сравниваемые программные средства

|  |  |  |  |
| --- | --- | --- | --- |
| Информационная система | Разработчик | Системные требования | Адрес в интернете |
| *Galileo* | Travelport | ОС: Windows XP, и выше  Процессор: Pentium Pll - 266  ОЗУ: 128 Мб. | https://www.travelport.com/ |
| *Sabre* | [American Airlines](https://en.wikipedia.org/wiki/American_Airlines) | два мейнфрема IBM 7090 | https://www.sabre.com/ |
| *Worldspan* | Travelport | ОС: Windows XP, и выше  Процессор: Pentium G4560  ОЗУ: 1024 Мб. | https://www.travelopro.com/worldspan-gds.php |

* + 1. **Galileo**

Galileo – одна из самых востребованных систем продаж авиабилетов в мире [1]. Обеспечивает пользователя возможностью бронировать не только авиаперевозки, но и сопутствующие туристические услуги. Также, как и Amadeus данная система предназначена для работы на терминалах с заранее обученными пользователями. Главная страница представлена на рисунке 1.1.

Поддерживает только два вида подключения:

* + стационарное подключение;
  + подключение через интернет.

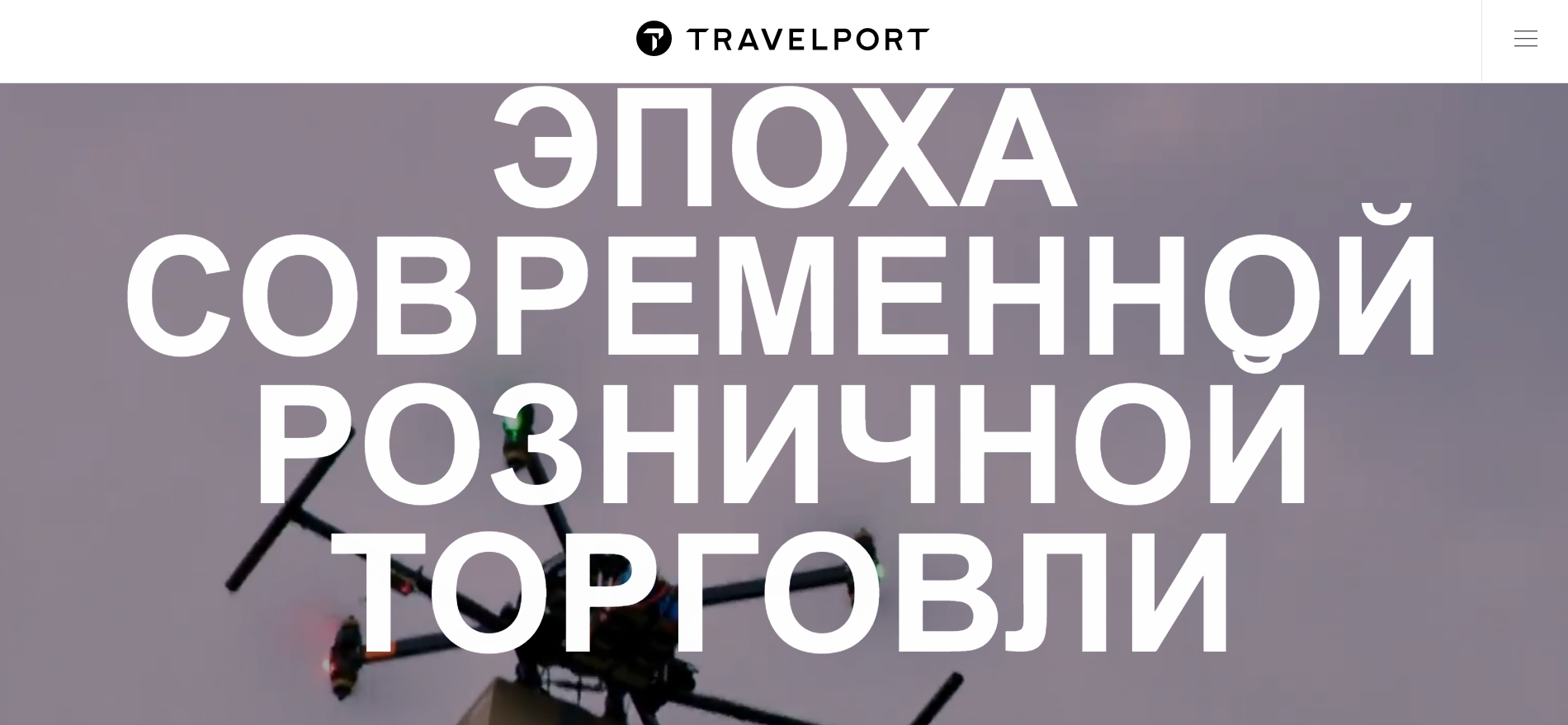


Рисунок 1.1 – Главная страница сервиса travelport.com

Стационарное подключение осуществляется посредством линий SITA и стоит дороже подключения через интернет, но такая версия предусматривает бесплатное ПО, состоящее из шести программ, расширяющих графический интерфейс системы и предоставляющих дополнительные функции по бронированию. Подключение через интернет стоит дороже, однако поддерживает многопользовательский режим. Система Galileo не поддерживает русский язык, как и Amadeus, но существует программа автоматизации деятельности туристических агентств Galileo Office, созданная специально для российского рынка услуг.

* + 1. **Sabre**

Sabre – сайт авиакомпании Sabre, которая является крупнейшим американским бюджетным перевозчиком [2]. В отличие от большинства других сайтов, осуществляющих продажу авиабилетов, не использует ни одну из GDS. Вся информация о рейсах, ценах и билетах предоставляется непосредственно самой авиакомпанией в виде БД, а диалог с потенциальным покупателем ведется при помощи сайта, оперирующего к данной БД. Sabre.com позволяет осуществлять поиск авиабилетов по различным критериям, а также бронирование номеров в гостиницах и заказ автотранспорта. Также при авторизации пользователя становится доступен личный кабинет, позволяющий просматривать статистику по осуществленным перелетам и предлагаемые авиакомпанией рейсы, подобранные индивидуально. Главная страница Sabre представлена на рисунке 1.2.

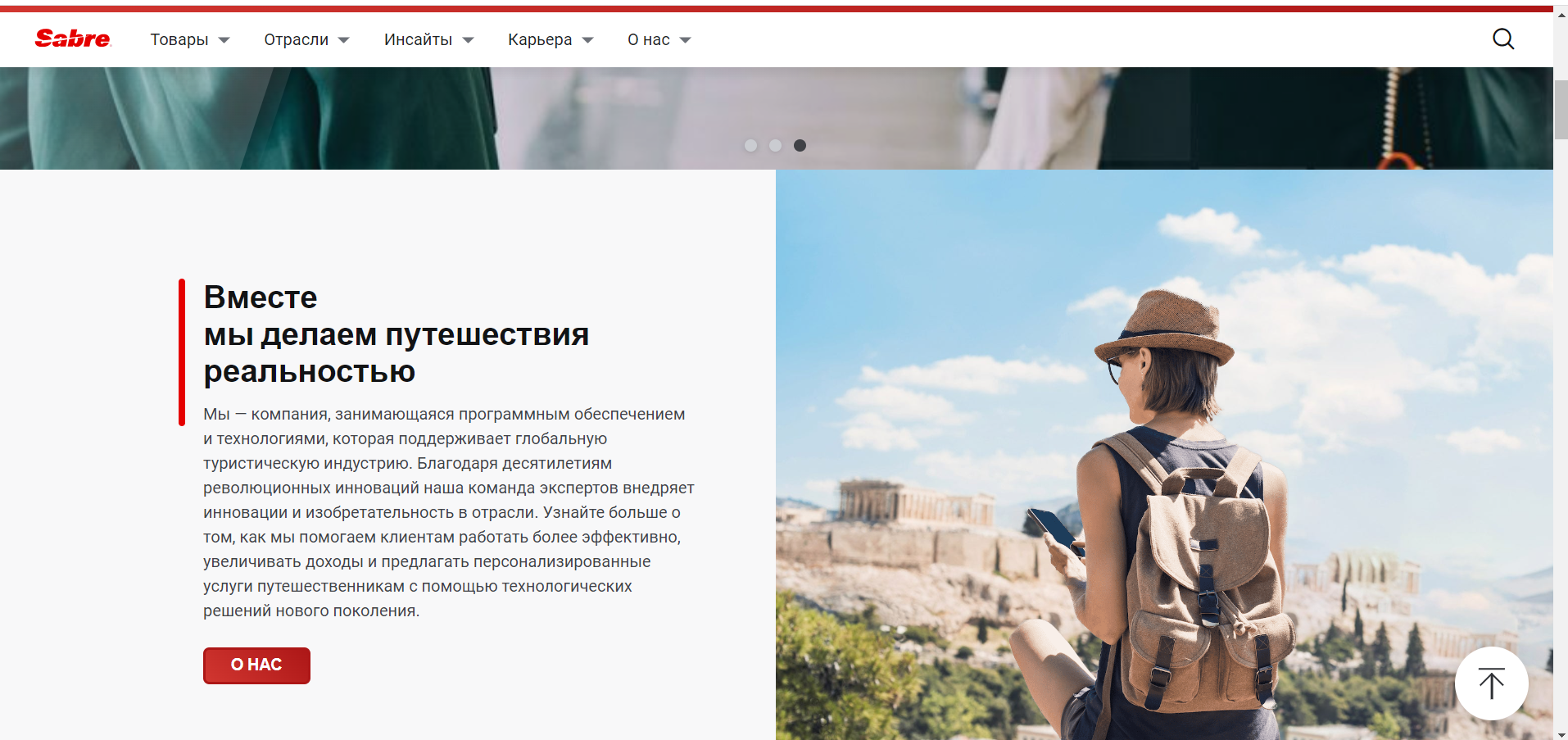


Рисунок 1.2 – Главная страница сервиса sabre.com

Подобная структура системы имеет как плюсы, так и минусы – данная информационная система обладает высоким быстродействием и удобным интерфейсом, понятным рядовому пользователю, но, в силу отсутствия подключения к GDS, количество осуществляемых рейсов, а значит и продаваемых билетов ограничено интересами авиакомпании.

* + 1. **Worldspan**

Worldspan – информационная система, позволяющая бронировать авиа и ЖД билеты, а также гостиничные номера [3]. Представляет собой мощную платформу, настроенную н использование GDS и оснащенную множеством таких средств как:

• автоматизация процессов. Представляет собой набор плагинов, осуществляющих оптимизацию трудовых затрат на обработку заказов;

• микшер результатов. Аналитическое средство для фильтрации поисковых результатов конкретного пользователя с целью предоставления наиболее подходящего варианта;

• личный кабинет. Область, защищенная информационной системой и предоставляющая отчеты по действиям авторизированного пользователя, таким как бронирование билета или аннулирование заказа. Личные кабинеты разных групп пользователей различаются функциональными возможностями;

• мидл-офис. Область, в которой реализуются все функции и компоненты Worldspan, такие как управление пользователями и их группами, настройки подключений, управление справочниками и т.п.;

• управление платежными методами. Worldspan позволяет подключать различные способы оплаты, от расчета при помощи банковской карты в режиме онлайн до расчета через терминалы обслуживания;

• настройка локализации. Информационная система поддерживает не только русский, но и английский и украинский языки. Архитектура Worldspan представляет собой множество модулей. Несмотря на то, что разработчики рассматриваемой системы сотрудничают с самыми популярными GDS, оплачивать доступ к данным глобальным дистрибутивным системам необходимо отдельно, что вкупе с затратами на саму Worldspan делает приобретение невыгодным для турагентств и совершенно невыгодным для рядового пользователя.

Главная страница Worldspan представлена на рисунке 1.3.

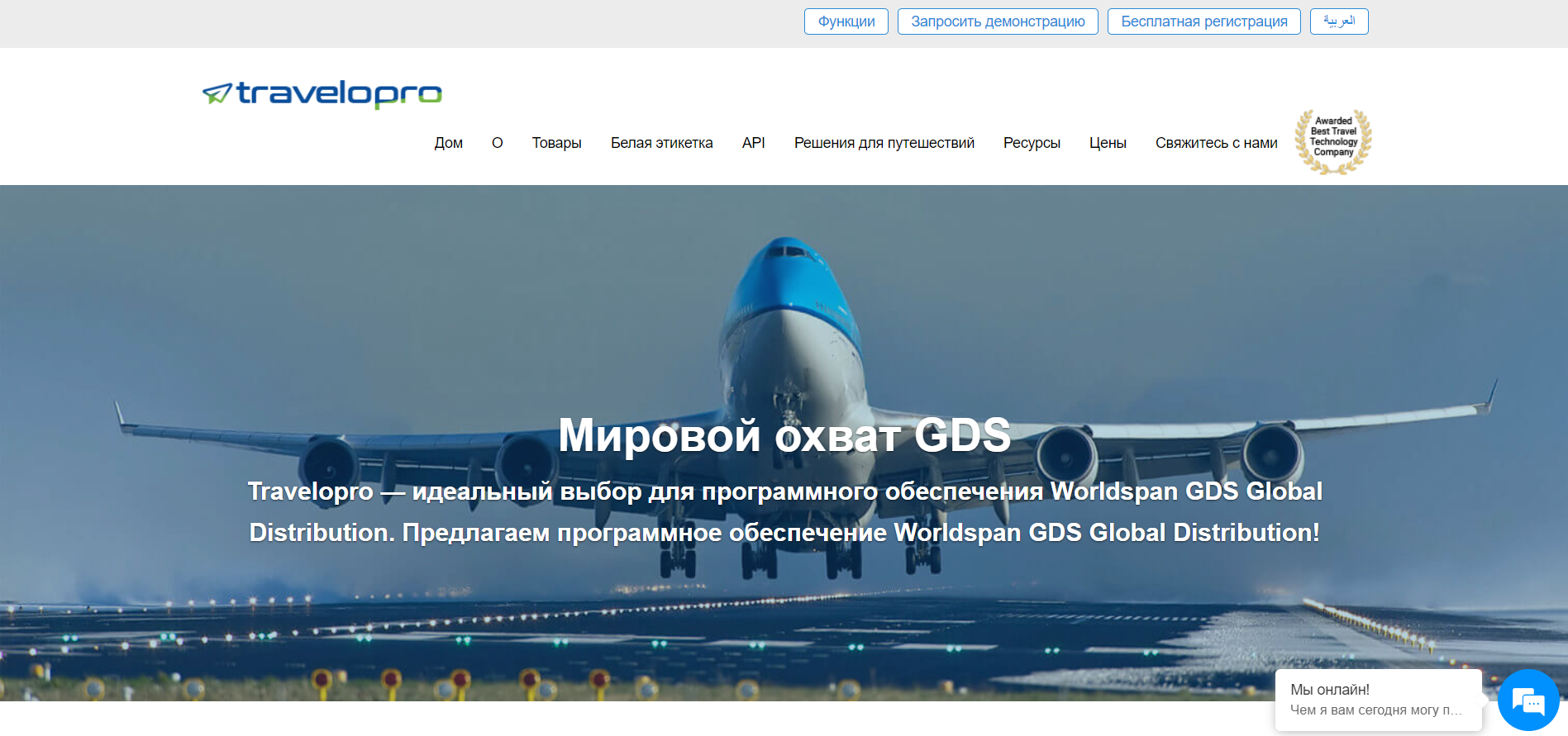


Рисунок 1.3 – Главная страница сервиса travelopro.com

Результаты сравнения программных средств представлены в таблице 1.2.

Таблица 1.2 – Результаты сравнения программных средств

|  |  |  |
| --- | --- | --- |
| Название системы | Достоинства | Недостатки |
| *Galileo* | Лучшие способы дистрибьюции и управления ресурсами в туристической индустрии, существует Windows версия системы | Недоступность для рядового пользователя, отсутствие русского языка |
| *Sabre* | Простой и понятный интерфейс, бесплатное использование | Отсутствие подключения к GDS |
| *Worldspan* | Удобный интерфейс, обширный функционал, модульность | Невыгодна для приобретения рядовым пользователем |

По результатам проведенного анализа можно сказать, что на данный момент существует множество мощных информационных систем продаж авиабилетов, обладающих обширными функциональными средствами. Однако большая часть таких средств является недоступной для рядового пользователя либо в силу сложности интерфейса, либо в силу своей высокой цены.

# Вывод

Основываясь на исследованиях аналогов, сделаем вывод, что функционально приложение должно поддерживать:

* поддерживать роли администратора и пользователя;
* обеспечивать возможность регистрации и авторизации;
* администратор обеспечивает возможность управления: регистрирование, редактирование, удаление маршрутов, рейсов, самолетов;
* позволять пользователю просматривать расписание;
* позволять пользователю осуществлять поиск и фильтрацию расписания;
* позволять пользователю управлять заказом: регистрация и отмена.

# Проектирование программного средства

После того как основной функционал был определен, можно начинать непосредственно проектирование приложения. Проектированию подлежит база данных приложения, а также UML диаграмма.

# Проектирование базы данных

Основными сущностями в данном проекте будут являться: пользователи, мероприятия, фотографии и категории. Роли будут являться предопределенными сущностями, поэтому их стоит сделать internal. Спроектированная база данных изображена на рисунке 2.1.

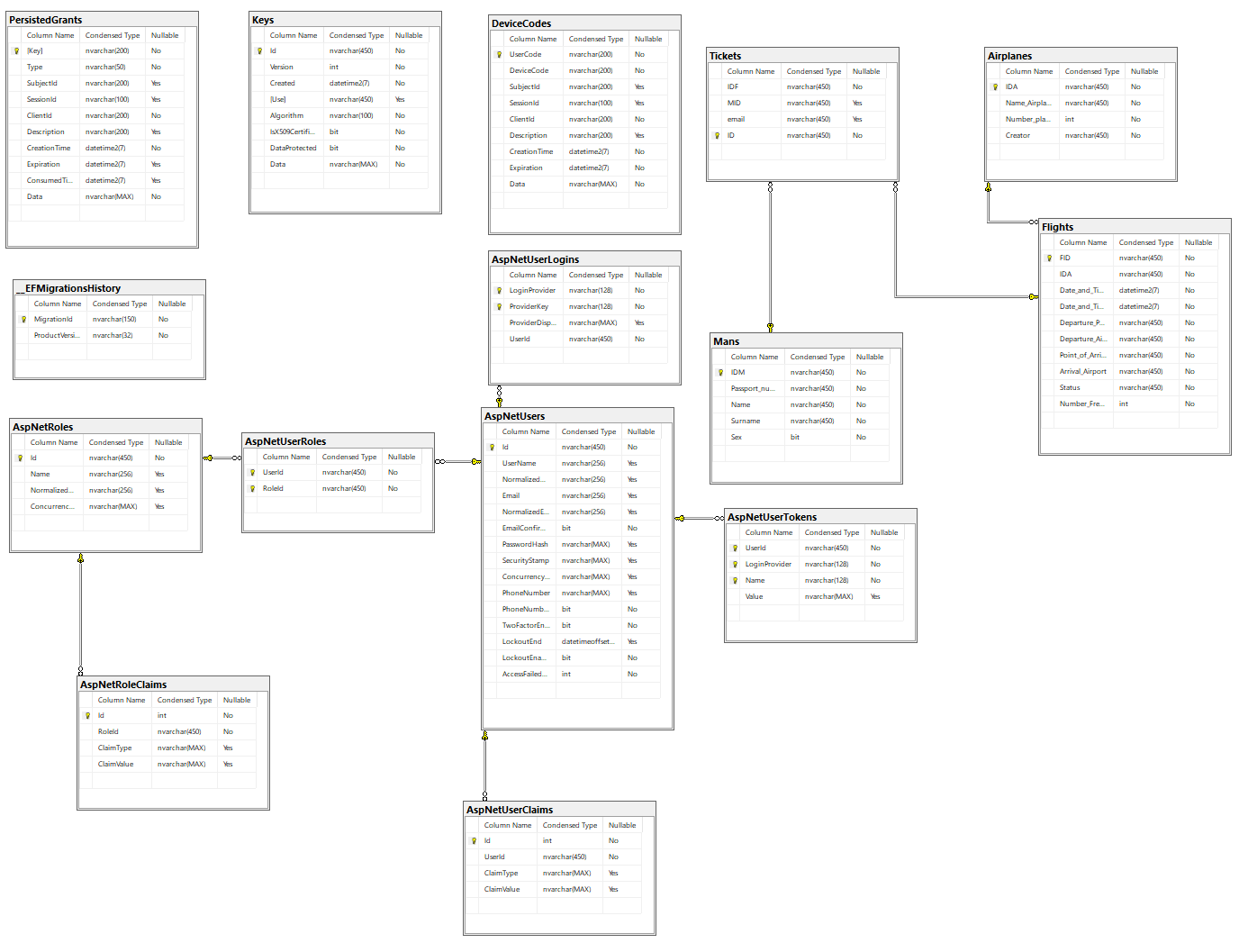


Рисунок 2.1 – Спроектированная база данных

Сущность, отвечающая за билеты, содержит в себе идентификатор рейса, имя мероприятия, идентификатор человека, место, идентификатор билета, почту.

Сущность, созданная для человека, содержит в себе идентификатор человека, имя и фамилию человека, пол, его личный номер.

СущностьAirplane, содержит в себе идентификатор самолета, наименование, количество мест, а также имя компании производителя.

Сущность для рейсов, содержит в себе идентификатор рейса, идентификатор самолета, количество свободных мест, статус рейса, время вылета, время прибытия, место прибытия, место вылета, наименование аэропорта прибытия и отбытия.

Основными сущностями в данном проекте будут являться: рейсы, билеты, самолеты и люди.

# Проектирование UML диаграммы

Ролями, предоставляемыми данной системой, являются администратор и пользователь. На основе предоставленного списка ролей необходимо построить UML диаграмму. UML диаграмма изображена на рисунке 2.2.

Таким образом видно, что неавторизованный пользователь имеет право авторизоваться или зарегистрироваться, в то время как уже авторизованный имеет больше возможностей, например поиск по билетам по разным критериям или просто просмотреть весь список билетов и отменять заказанный билет. Следующий ролью является администратор, который имеет право на добавление, просмотр, а также изменение существующих рейсов, билетов, людей, самолетов.

Данный этап является одним из самых значимых по время проектирования, так как именно тут определяется, кто и какими ролями будет обладать в данной системе. Обзор аналогов не отвечает на этот вопрос, просто дает некоторые подсказки, которые дают возможность воспользоваться, но не обязательно. Но данная сфера не обладает какими-либо серьезными связями между пользователями различных групп. Если все свести к минимуму, то можно заметить, что функционал достаточно тривиален, и не требует к себе особо серьезного внимания со стороны проектирования, тем не менее, это не делает этап менее значимых.

Каждая роль имеет своих пользователей, которые должны мочь выполнять только то, что предусмотрено для них системой, любое нарушение пределов данных и оговоренных в этом приложении ролей может привести к непредвиденным последствиям, что означает важность правильной реализации данный ролей. Реализация, таким образом, должна предусматривать, что каждый пользователь может совершать определенные действия, согласно его роли. Если пользователь пытается сделать что-то, что находиться вне его полномочий, система должна пресечь его действия, иначе они пагубно отразятся на системе.

Обычно при использовании правильных технологий, настройка прав идет достаточно просто и не требует каких-то больших временных затрат со стороны разработчика или тестировщика, но даже в таким серьезных технологиях могут встречаться изъяны, способные погубить целый проект, поэтому используя даже самые мощные и серьезные технологии нельзя относиться к этому пренебрежительно.

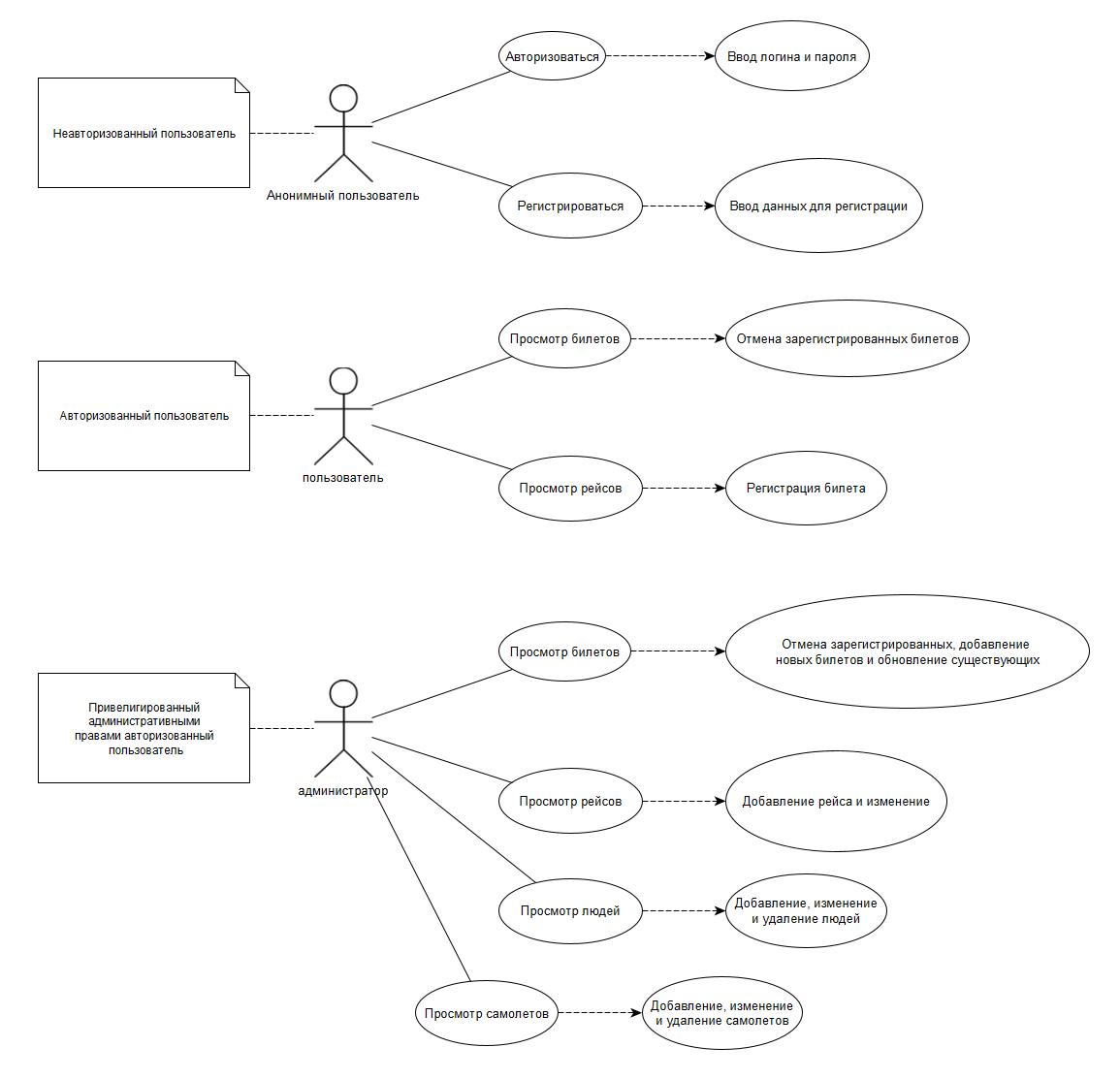


Рисунок 2.2 – Спроектированная UML диаграмма

Администратор, тот, кто является владельцем наибольших полномочий имеет право создавать категории, изменять их или же просто просматривать их список. Также он имеет право просматривать все мероприятия и, как результат, удалять их или изменять, в случае указания неверной информации организаторами.

# Проектирование основных алгоритмов

Поскольку администратор является самой привилегированной сущностью в данном приложении, то его созданию стоит отнестись с максимальной ответственностью. Будет мудрым решением создать код, выполняющий проверку на наличие прав администратора в системе. Блок-схема данного алгоритма изображена на рисунке 2.3.

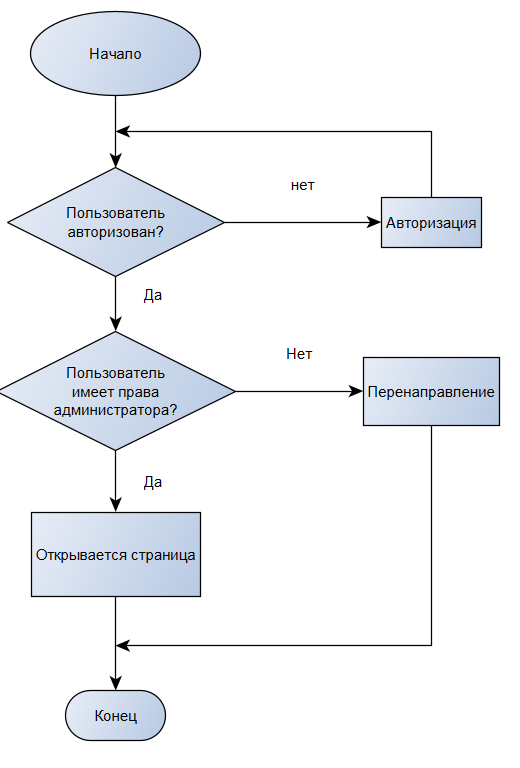


Рисунок 2.3 – Блок-схема алгоритма проверки прав администратора

Данный алгоритм гарантирует доступ к странице при наличии прав администратора в системе. А также не дает возможности получить доступ к страницам с закрытым доступом.

Второй алгоритм, который может быть описан это алгоритм сохранения данных на сервере. Блок-схема данного алгоритма изображен на рисунке 2.4.

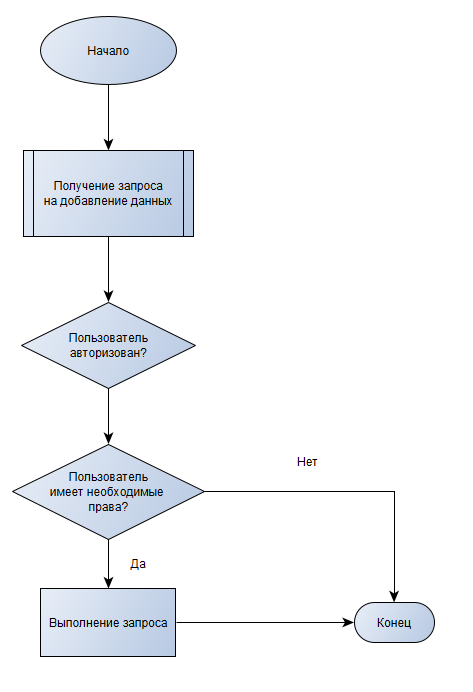


Рисунок 2.4 – Блок-схема алгоритма добавления данных

Таким образом были определены путь развития приложения, что должно быть реализовано и как должен быть рассредоточен доступ между ролями. Также определены процессы выполнения некоторых главных алгоритмов приложения.

# Выбор средств реализации

Следующий логическим этапом следует выбор средств реализации проекта. Было решено, что данный проект будет написан на двух языках программирования таких как С# и JavaScript, используемые для backend и frontend соответственно. Для backend части будет выбран ASP .Net Core. А для frontend был сделан выбор в пользу ReactJS. Для работы с базой данных был выбран EntityFramework, этот модуль унифицирует доступ к любой базе данных, что делает данное приложение абсолютно гибким в плане этого вопроса. В случае данного проекта MS SQL Server 2019 был выбран в качестве базы данных.

К инструментам следует отнести Visual Studio 2022, VS Code, Docker Visual Studio 2022 – это IDE для разработки приложений под C#, является самым продвинутым в данном инструменте. Для разработки frontend части приложения будет использован VS Code, который, в свою очередь, является универсальным инструментов и позволяет работать с любым стеком технологий, в данной случае он используется для JavaScript, ReactJS и их зависимостей. Docker был включен в данный список так как в данной работе присутствует необходимость создания контейнера с backend частью данного приложения, то есть необходимость контейнеризации данного приложения.

Таким образом разработчик получает полноценный инструментарий, максимально ускоряющий процесс разработки. Следовательно, создание MVP потребует цену в несколько недель и минимальных усилий разработчика, что является одним из критериев качества выбранных технологий. К другим также стоит отнести цены данных продуктов для коммерческой разработки, эргономичность использования, покрытия множества сфер разработки, например, включение всего функционала.

# Вывод

Таким образом в конце данной главы была спроектирована база данных, описана UML-диаграмма, некоторые основные алгоритмы, а также определен основной стек технологий, используемый в данном проекте.

Каждый выбор, осуществленный в данной главе, является осознанным и обоснованным. Продумывание всех подобных моментов в начале, исключает неопределенность в будущем. Таким образом, заранее известен список всех используемых технологий, сферы деятельности, которые они покрывают и некоторые основные алгоритмы, вокруг и на основе которых потом будут появляться новые механизмы взаимодействия между модулями системы. Таким образом, создавая администратора вначале, владелец сервиса освобождается от первичной инициализации базы данный администраторской информацией, необходимой для входа, что значительно упрощает весь процесс администрации.

# Разработка программного средства

Приложение имеет две составляющие, это frontend часть и backend часть. Frontend отвечает за представления, с которыми может работать пользователи системы, будь то администраторы или обычные посетители.

Backend в свою очередь предоставляет внешний API с который взаимодействует frontend и представляет полученную информацию с backend пользователю в human-friendly формате. Сам же backend сервер является REST сервисом, который оперирует с json и предоставляет REST API.

# Разработка frontend части

Разработка frontend части будет осуществлена на ReactJS с помощью редактора кода VS Code. Дополнение к ReactJS будет также использован Reactstrap, React-bootstrap, jquery. Выбор пал на данные инструменты в связи с их популярностью, и, как следствие, наличием огромного количества информации о всевозможных проблемах и их решениях, которые могут возникнуть во время разработки. Также данные инструменты позволяют довольно быстро создать MVP, что позволит, в случае необходимости, войти на рынок.

Архитектура проекта изображена на рисунке 3.1.

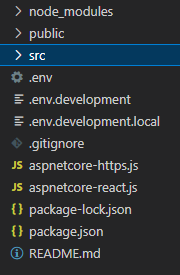


Рисунок 3.1 – Архитектура frontend проекта

Следует отметить назначение всех каталогов в проекте, дабы прояснить их суть и необходимость их наличия. Каталог node\_modules содержит в себе все зависимости, которые будут использованы в данном NodeJS проекте, без них проект не будет работать. Директорий public содержит в себе public ресурсы, которые доступны любому пользователю в независимости от их прав. В каталоге src размещен весь код frontend, который отвечают за всю логику взаимодействия с backend путем использования его внешнего REST API.

Теперь следует более детально рассмотреть каталог src. Содержимое каталога src изображено на рисунке 3.2.

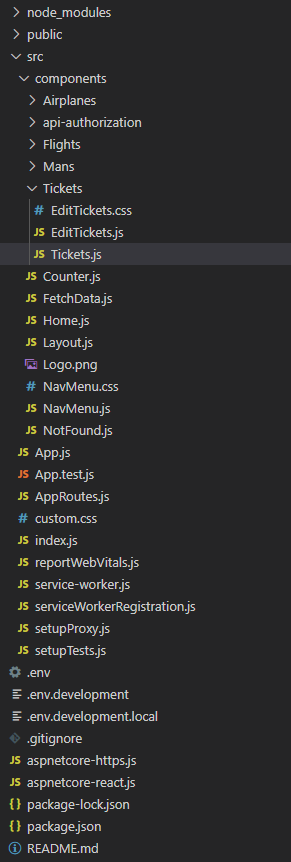


Рисунок 3.2 – Содержимое каталога src

Внутри каталога src содержаться параметры системы, которые определяют, как и каким образом пользователь обладающими определенным набором правам будет взаимодействовать с компонентами на frontend-е. В нем есть файл AppRoutes.js определяющий к каким страницам могут иметь доступ не авторизованные пользователи. Также в файле setupProxy.js указываются параметры прокси-сервера. Однако при запуске frontend-а первым откроется файл index.js.

В каталоге components располагается весь код, отвечающий за взаимодействия пользователей с системой, например, такие как управление рейсами или билетами. Данный каталог, содержит весь функционал доступный как администратору, так и обычному пользователю. Содержание каталога components находится на рисунке 3.3.

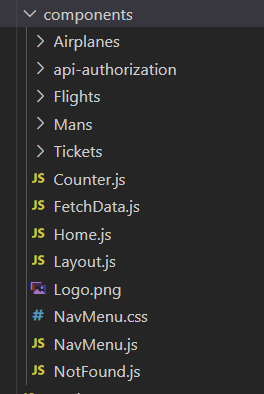


Рисунок 3.3 – Содержимое каталога components

Папки Airplane, api-authorization, Flights, Mans, Tickets содержат в себе код, который отвечает за взаимодействие пользователя со страницей браузера, обработку данных, отправку этих данным на backend посредством использования запросов.

Внутри каталога Airplanes можно наблюдать три файла. Файл Airplanes.js демонстрирует все содержимое имеющее непосредственное отношение к самолетам, которые указываются в рейсах, а именно характеристики летающего средства. К нему прилагается файл для редактирования данного содержимого, наблюдаемого на странице, в файле, указанном выше. В нем внесенные изменения отправляются на backend-е для последующего изменения, добавления или удаления данных. На рисунке 3.4 демонстрируется содержимое папки Airplane.

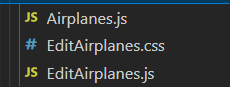


Рисунок 3.4 – Содержимое каталога Airplanes

Каталог api-authorization содержит код для авторизации пользователя, включающий как описание процесса авторизации, так и процесса выхода с системы.

Каталог Flights содержит код, необходимый для работы с рейсами. Включает в себя функционал добавления, удаление и редактирования рейсов администратором. Содержимое каталога Flights изображено на рисунке 3.5

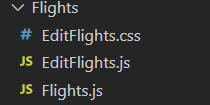


Рисунок 3.5 – Содержимое каталога Flights

Код в файле EditFlights отвечает за создание, редактирование или удаление информации, связанной с рейсами доступ к ней, имеют пользователи, обладающие правами администратора и Flights для просмотра содержимого.

Директорий Mans содержит в себе описание и функционал, которые отвечают за отображение данных о людях раннее приобретавших или приобретающих билеты на рейсы, данная страница обеспечивается доступ при наличии определенных прав. Содержимое каталога Mans изображено на рисунке 3.6.

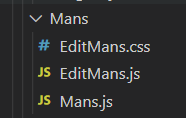


Рисунок 3.6 – Содержимое каталога Mans

Каталог Tickets содержит в себе реализацию, которая позволяет отображать данные о заказанных билетах раннее приобретавшихся на рейсы, данные страницы доступны при наличии как обычных, так и администраторских прав.

# Разработка backend части

Разработка будет вестись на С# с использованием ASP .Net Core, с помощью которых в итоге будет создано REST приложение, предоставляющее свой API для frontend, с которым, уже в свою очередь, будет работать пользователь. Разработка будет вестись в Visual Studio 2022.

Вспомогательными средствами будет EntityFramework, который позволяют быстро и удобно работать с репозиториями, относящимся к базам данных. EntityFramework, который поможет создать code-first базу данных, то есть сначала есть возможность написания кода, а потом генерация базы данных на основе написанного кода. ASP .Net Core позволяет создавать web-приложения.

Прежде чем начать разрабатывать необходимо проинициализировать проект. Правильная инициализация проекта значительно упростить работу в будущем, во время разработки программного обеспечения. Инициализация проекта выполняется на рисунке 3.7.

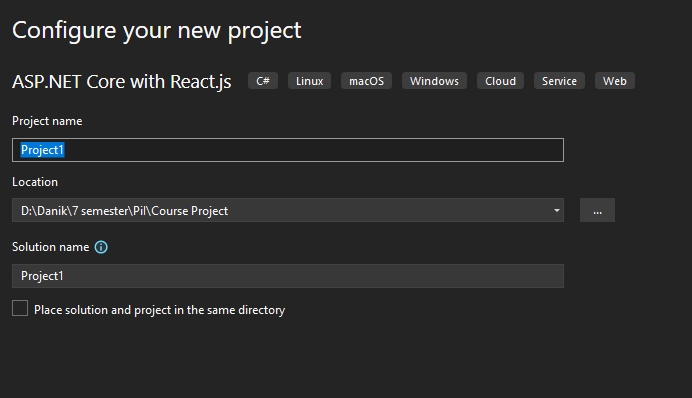


Рисунок 3.7 – Инициализация проекта ASP .Net Core with React.js

После инициализации проекта можно приступать к написаю кода.

Прежде всего стоит определить некоторые понятия, которые будут встречаться по ходу разработки. EntityFramework предоставляет множество различный инструментов, классов для ускорения ведения разработки.

Контроллерами называются классы, которые обрабатывают HTTP запросы, согласно указанным URL, методам и переданными заголовками, параметрами или телом запроса.

Сервисами же называют классы, объекты которых содержат в себе методы работы с бизнес-логикой, таким образом они инкапсулируют в себе все, что каким-либо образом связано с этой областью программы.

Репозитории – классы, предоставляющие доступ к данным, сохраненных в базе данных посредством использования EntityFramework, который, в свою очередь, позволяет с помощью DbSet объявить необходимые сущности, описать их свойства и на основе этого описания создать таблицы.

Структура проекта изображена на рисунке 3.8.

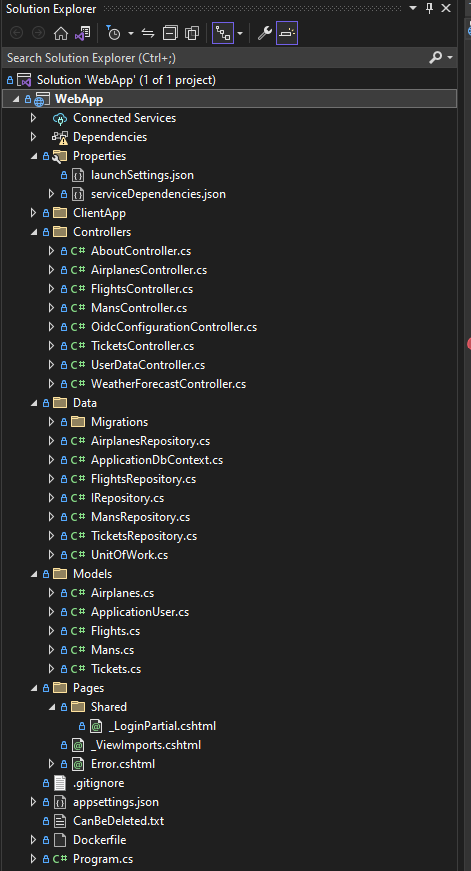


Рисунок 3.8 – Структура проекта

Структура проекта содержит в себе множество каталогов, в которых, согласно смысловой нагрузки, которые несут их названия, содержаться классы проекта.

Каталог Controllers несет в себе множество классов, отвечающие за обработку всевозможных http запросов.

Каталог Data хранит в себе реализацию бизнес-логики, когда запросы приходят с fronted-а, то их функциональная составляющая реализована в данном месте.

Каталог Models содержит сущности БД необходимые для взаимодействия.

Все http запросы обрабатываются соответствующими контроллерами, в качестве примера в приложении A приведен метод контроллера, обрабатывающий любые пришедшие запросы на регистрацию.

Здесь присутствуют необходимые аннотации для REST-контроллеров. Сами же методы, обрабатывающие запросы, находятся внутри класса. Метод IsAdminAsync непосредственно отвечает за проверку принадлежности пользователя к администраторам и в качестве параметров принимает тело запроса в котором указано имя пользователя.

В самом начале происходит проверка принятой информации, смотрится указано ли имя пользователя, после этого совершается обращение к сущности БД и путем использования встроенных методов выясняется причастность данного пользователя к группе администраторов, и в соответствии с ответом возвращается результат.

Далее рассмотрению подлежит метод-обработчик запроса контроллера, который отвечает за получение информации о билетах. Код в приложении B.

Код непосредственно создания нового билета приведен немногим ниже в приложении С.

По абсолютно такому же принципу написаны все остальные методы-обработчики и их контроллеры.

# Контейнеризация

Далее следует развернуть docker-контейнер с backend сервером. Для этого в Visual Studio Installer найти и установить нагрузку, указанную на рисунке 3.9.

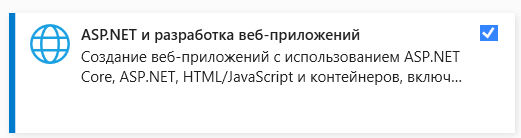


Рисунок 3.9 – Скачивание нагрузки для docker контейнера

После этого появится в проекте появится возможность добавить Docker поддержку. Изображение на рисунке 3.10.

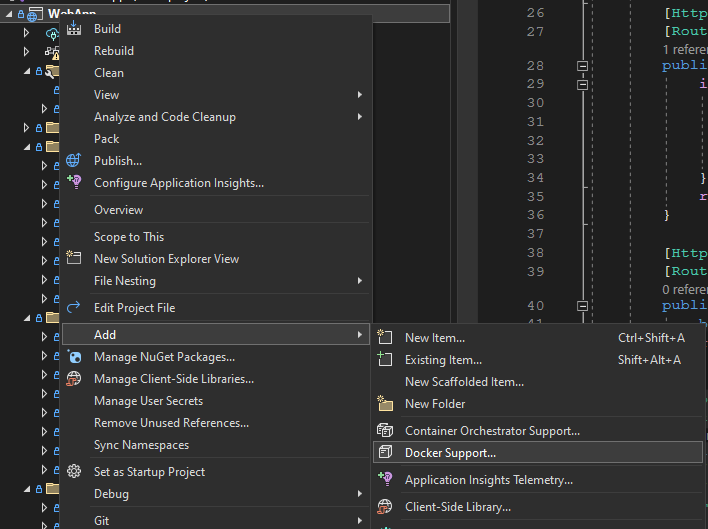


Рисунок 3.10 – Добавление Docker поддержки

После добавления поддержки появится окно, указанное на рисунке 3.11 в котором нужно выбрать тип контейнера, который будет использоваться.

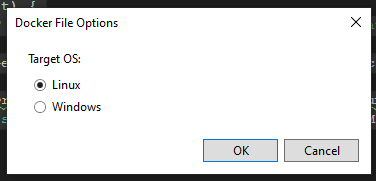


Рисунок 3.11 – Выбор типа контейнера

Дальше в проекте появляется файл в нем должно быть содержимое такого плана. Изображение на рисунке 3.12.

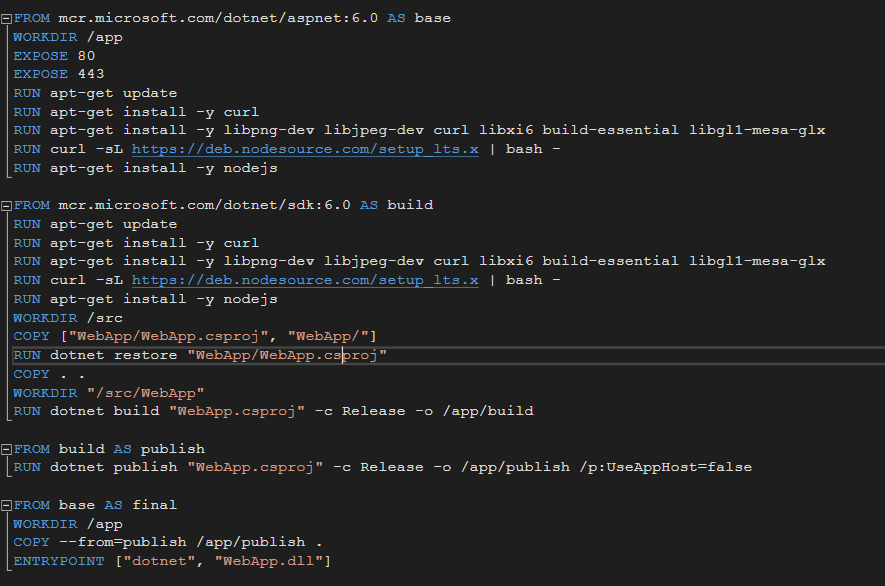


Рисунок 3.12 –Содержимое контейнера

В результате проделанной работы собираем проект в Docker. После этого появится образ в docker-desktop. Image данного проекта изображен на рисунке 3.13.

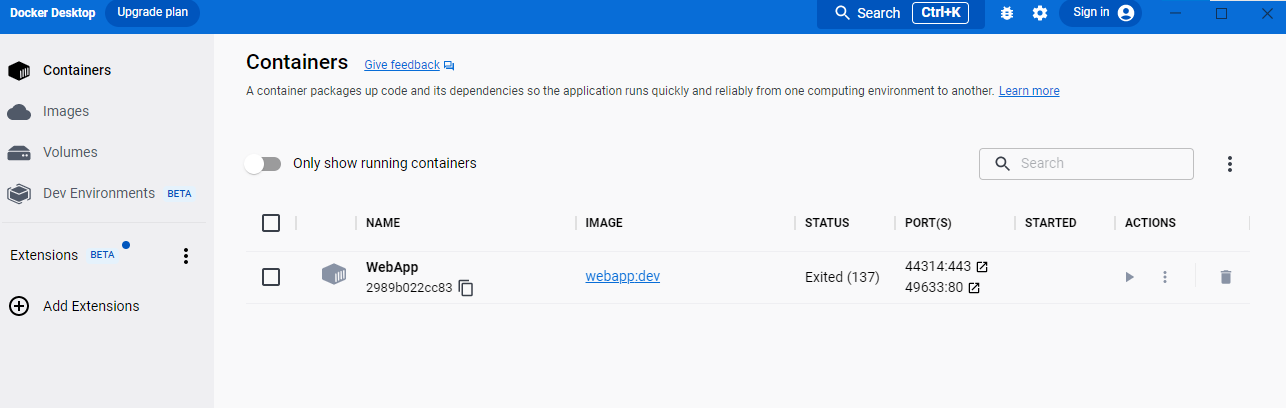


Рисунок 3.13 – Image backend

После его запуска появится возможность обратиться к данному web-приложению.

# Вывод

По результату проведенной работы в данном разделе, были разработаны две части данного приложения, а именно frontend и backend части, каждая из которых является отдельный сервером. Между ними, как архитектурными единицами, существует одно яркое отличие, в то время как backend является полностью независимым от других сервисом сервером, frontend сервер же является полностью зависимым от backend части приложения, так как сам по себе он не несет никакой смысловой нагрузки и не способен обрабатывать никакую бизнес-логику. И в конце всего процесса был создан docker-контейнер.

После разработки стоит приступить к следующей не менее важной части разработки любого приложения, к тестированию этого самого приложения. Так как, прежде чем оно увидит свет, оно должно быть протестировано во всех основных местах, или, как план минимум, проведены smoke-тесты, что подразумевает тестирование основных аспектов приложения самими разработчиками, дабы облегчить жизнь QA отделу.

# Тестирование

Тестирование является не менее важной компонентой любого успешного продукта, так как он исключает попадание нерелевантного функционала в общее пользование, что может испортить впечатление клиентов от пользования этим продуктом. Тестирование – процесс выявление ошибок работы или несоответствующий требованиям функционал приложения, которые может повлечь за собой ошибки, которые могут повлечь потерю времени и денег клиентов, что испортит и отобьет любое желание далее пользоваться разработанной платформой.

Во время разработки данного приложения было решено, провести мануальное тестирование для frontend и unit-тестирование для backend части, таким образом, приложение будет протестировано двумя способами и может в будущем быть разбито на две группы QA Automation и QA Manual.

# Ручное тестирования frontend части

Тестирование frontend части включает в себе проверку правильности осуществления перехода между страницами, которые совершают пользователи системы, также проверку правильности отправления запросов, их обработки frontend сервером, а также правильности отображение представлений.

Рассмотрим сценарий заказа билета. Для начала необходимо зарегистрироваться. При заполнении формы неправильными данными должно быть получено оповещение об ошибки. Заполнение формы неправильными данными и уведомление об ошибке изображено на рисунке 4.1.

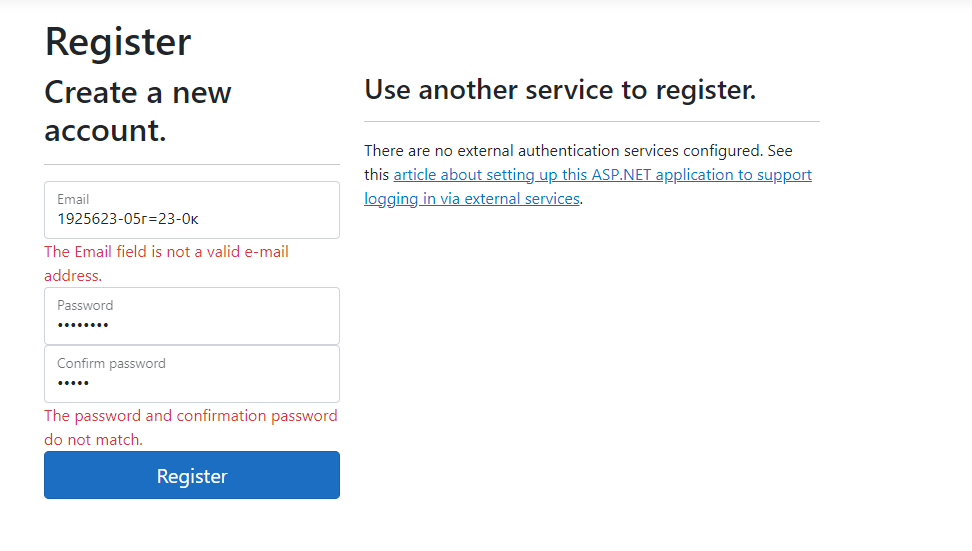


Рисунок 4.1 – Форма заполнения неверными данными

Теперь заполним формы правильными данными и завершим регистрацию. После успешного завершения регистрации должна быть изображена форма авторизации. Успешная регистрация и переход изображена на рисунке 4.2.

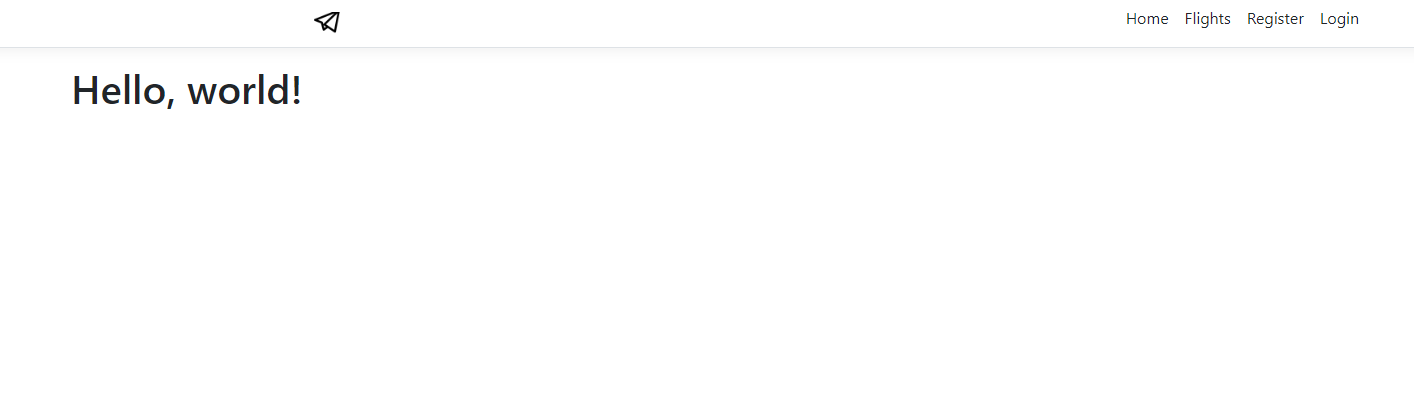


Рисунок 4.2 – Успешная регистрация и окно авторизации

После регистрации ввести данные. При неверном введение данным, получим оповещение об ошибке. Оповещение об ошибке изображено на рисунке 4.3.

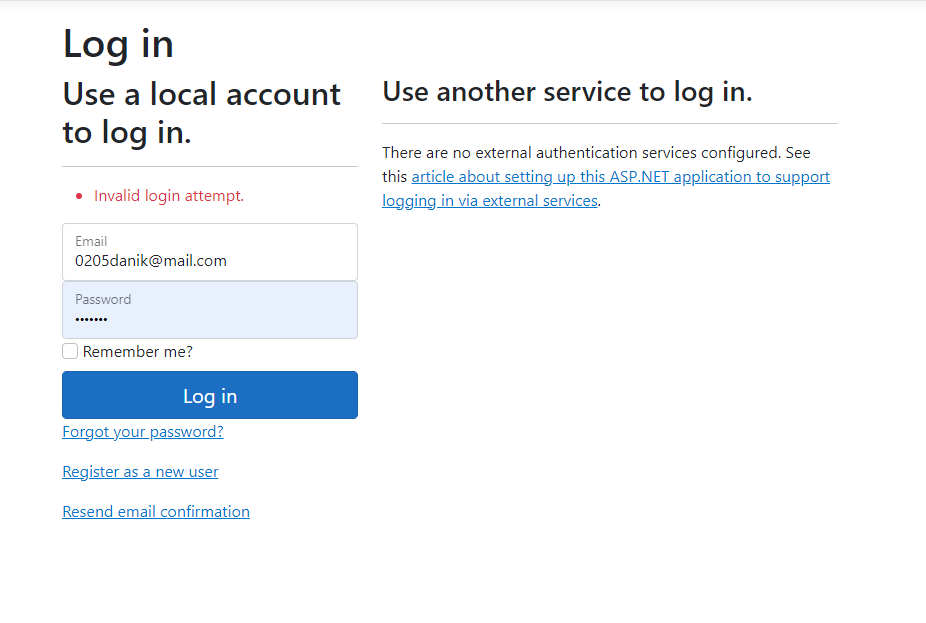


Рисунок 4.3 – Неправильный ввод данных для входа и оповещение об ошибке

Ввод правильных данных изображен на рисунке 4.4.

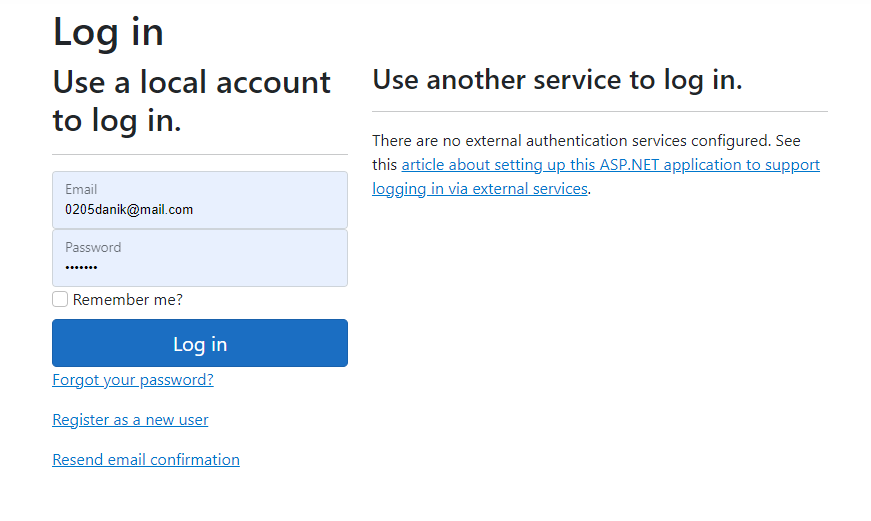


Рисунок 4.4 – Правильный ввод данных при авторизации

После ввода правильных данных и авторизации должен быть изображен личный кабинет пользователя. Переход на главную страницу после верного ввода данных изображен на рисунке изображен на рисунке 4.5.

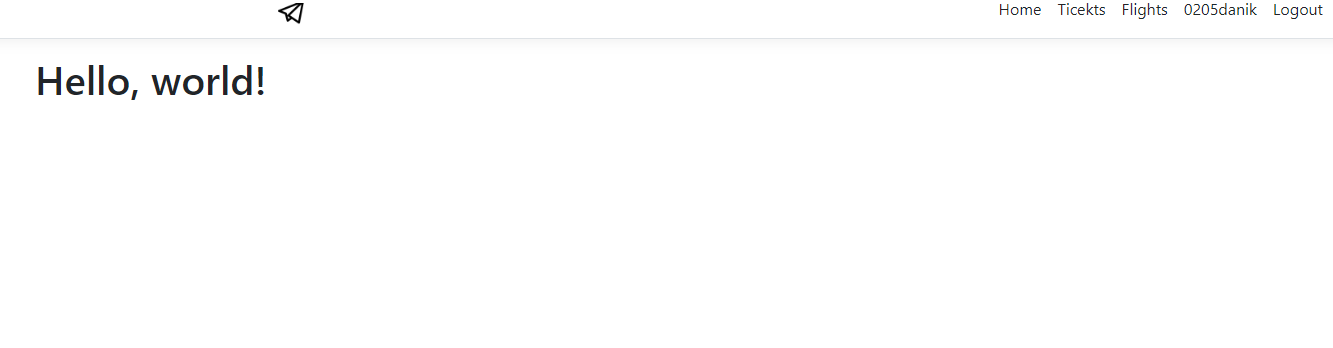


Рисунок 4.5 – Главная страница

Как видим, отображается логин пользователя, что свидетельствует о правильной авторизации, далее откроем вкладку присоединения к событиям. Вкладка с рейсами, в которых данный пользователь сможет, в случае желания, приобрести изображены на рисунке 4.6.

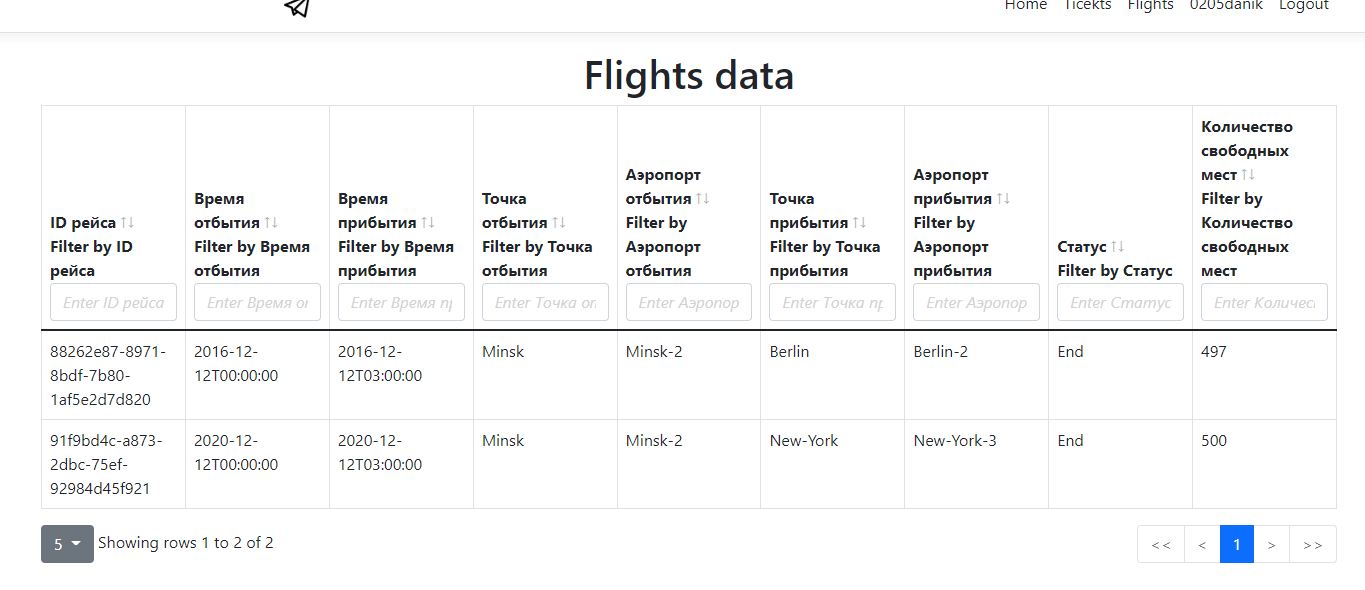


Рисунок 4.6 – Страница рейсов

Для того, чтобы приобрести билет следует нажать на «Flights», на который пользователь желает приобрести билет, после появиться сообщение изображено на рисунке 4.7.

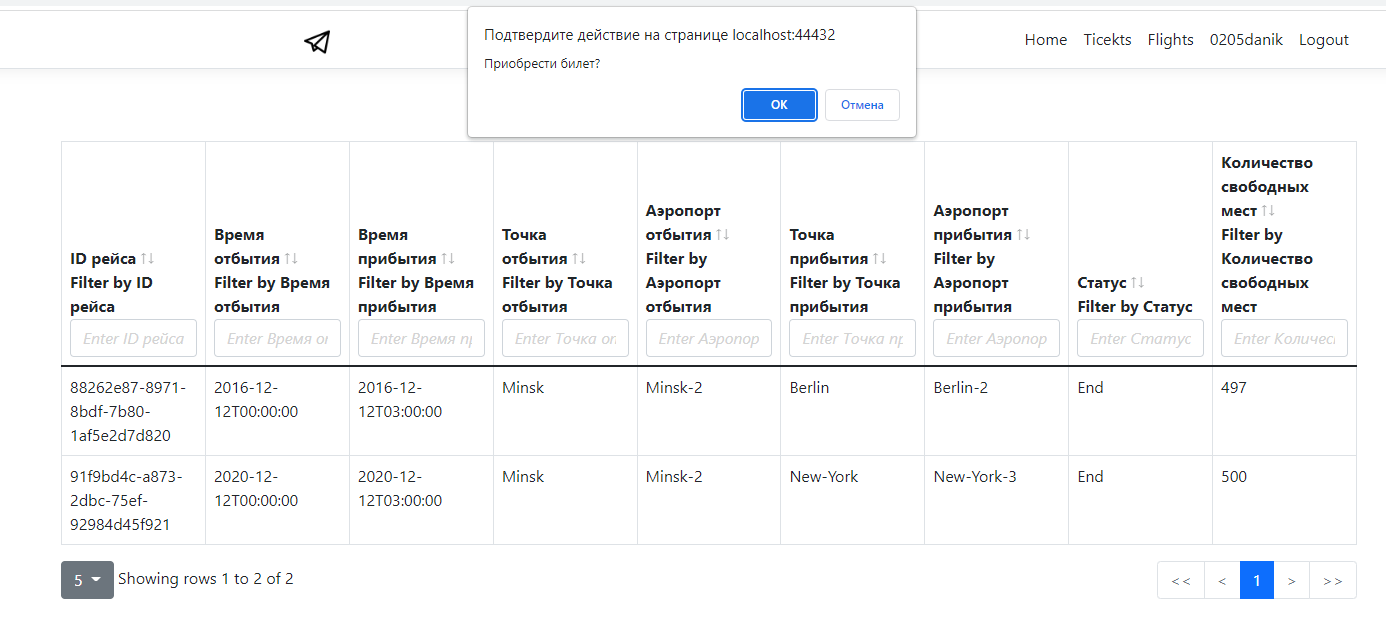


Рисунок 4.7 – Приобретение билета

Далее необходимо нажать «Ок» для приобретения билета. Изображение на рисунке 4.8.

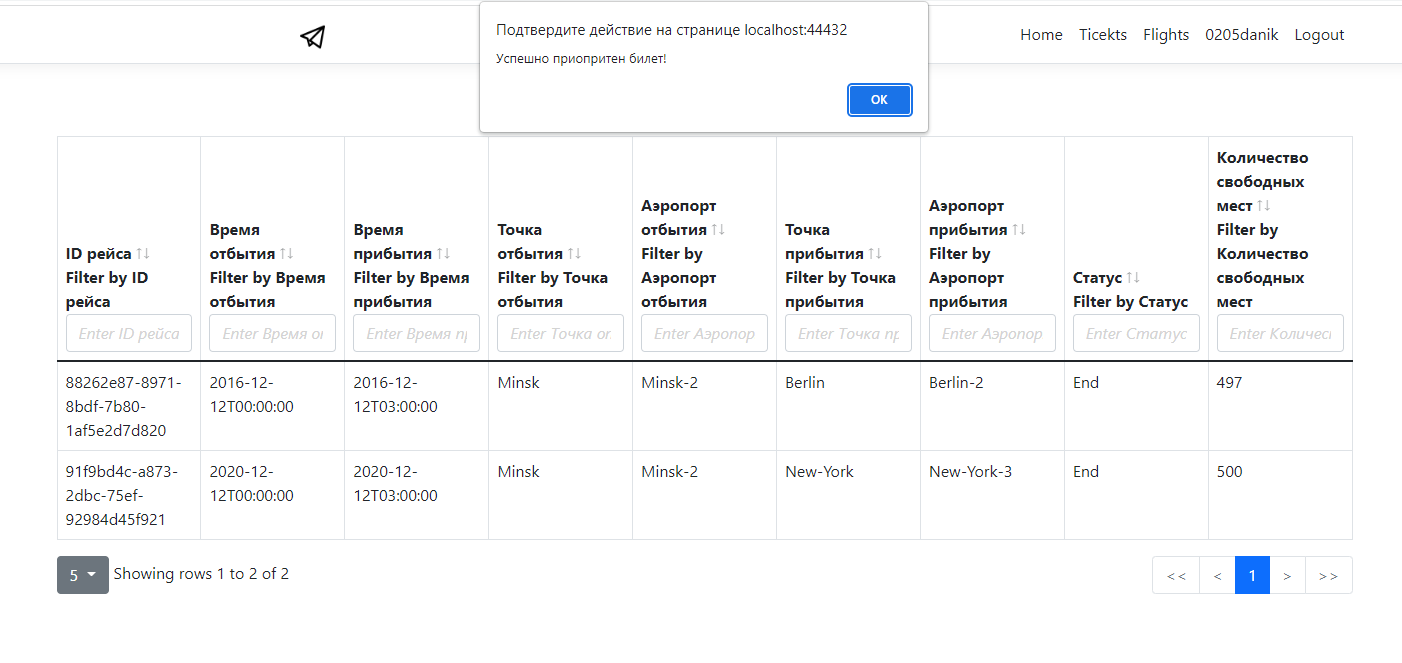


Рисунок 4.8 – После приобретения билета

Далее следует нажать на «Ок», и перейти на вкладку «Tickets». В ней можно увидеть билет, приобретенный на выбранный рейс изображение на рисунке 4.9.

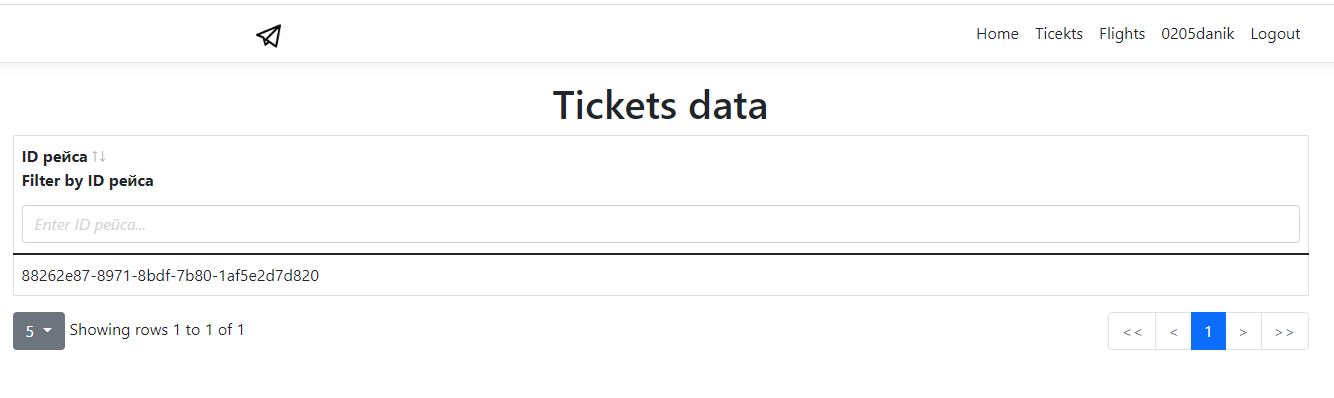


Рисунок 4.7 – Билеты пользователя

Как результат данного ручного теста, следует сделать вывод, что функционал приобретения билета пользователем реализован правильно, обработка ошибок осуществлена, верно, таким образом тест успешен.

# Вывод

После проведения ручного и unit тестирований, можно сделать вывод, что покрываемый функционал работает правильно и никаких ошибок в ходе исполнения замечено не было. В свою очередь это оправдывает время, затраченное на грамотное проектирование проекта, а также на его реализацию.

# Руководство пользователя

После того как было выполнено тестирование, становится ясно, что приложение работает корректно. Следующим этапом будет написание руководство пользователя. В данной системе существует три категории пользователей такие как пользователь, организатор и администратор. Каждого из них следует рассмотреть раздельно. Следует отметить, что каждая роль, включает в себе все права лежащей ниже роли в иерархии привилегий.

# Руководство пользователя

Любой посетитель в самом начале имеет право регистрироваться или авторизоваться. Для того, чтобы зарегистрироваться следует осуществить переход с главной страницы на вкладку «Register». Главная страница данного приложения изображена на рисунке 5.1.

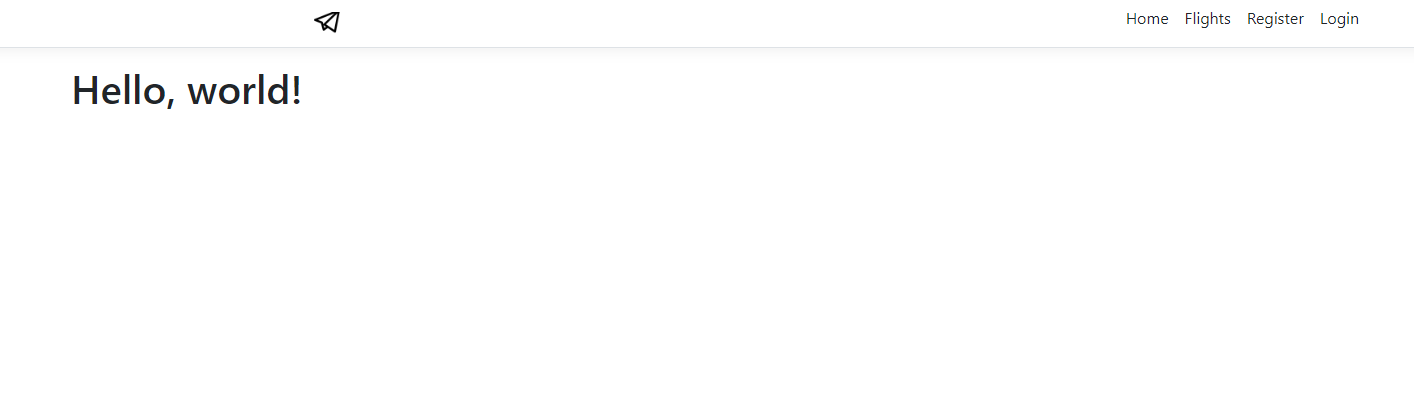


Рисунок 5.1 – Главная страница приложения

Переход на страницу с формой «Register» изображен на рисунке 5.2.

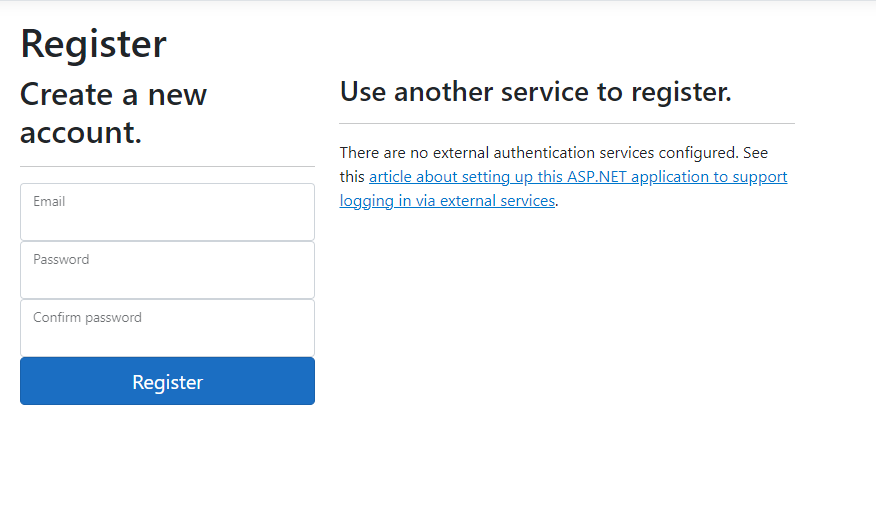


Рисунок 5.2 – Форма регистрация

После регистрации должен осуществиться переход на главную страницу. Переход на главную страницу после регистрации осуществлен на рисунке 5.1.

После осуществляется авторизация. Нажимаем на «Login» происходит автоматический вход и перевод на главная страницу.

Теперь пользователь может заказать билет. Для этого необходимо перейти на вкладку «Flights» и нажать на рейс, для приобретения билета. Рисунок 4.8.

Чтобы отменить ранее приобретенный билет на рейс необходимо перейти на вкладку «Tickets» и нажать на необходимый билет. При нажатии на билет появится окно с подтверждением совершаемого действия. Рисунок 5.3.

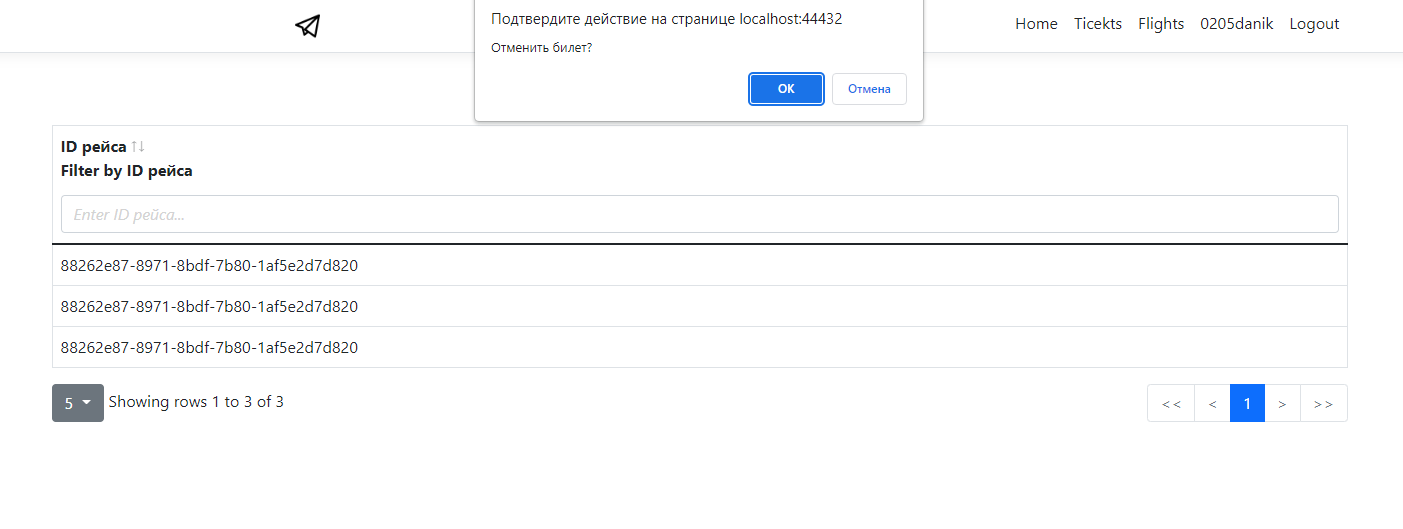


Рисунок 5.3 – Отменить заказанный билет

Также можно посмотреть параметры своего аккаунта и при необходимости изменить их. Для этого необходимо перейти на вкладку с наименованием логина вашего аккаунта.

# Руководство администратора

При входе в систему появляется возможность проследовать в разные разделы такие как управление рейсом, людьми(пассажирами), билетами, самолетами. Данное меню выбора изображено на рисунке 5.4.

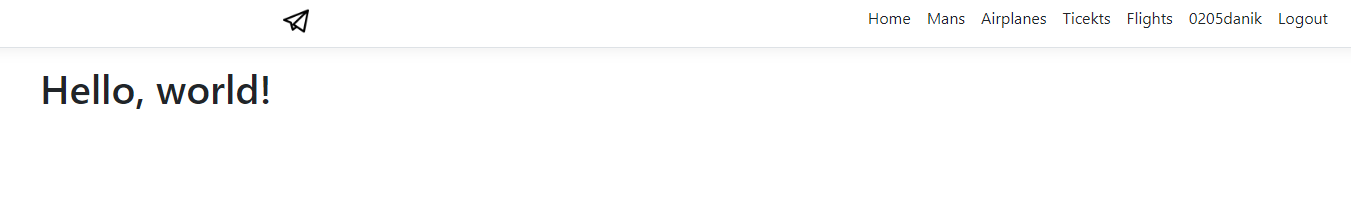


Рисунок 5.3 – Главное меню администратора

Для редактирования информации о людях(пассажирах) необходимо перейти во вкладку «Mans»(рисунок 5.4) и нажать на человека для внесения изменений данных о нем(рисунок 5.5).

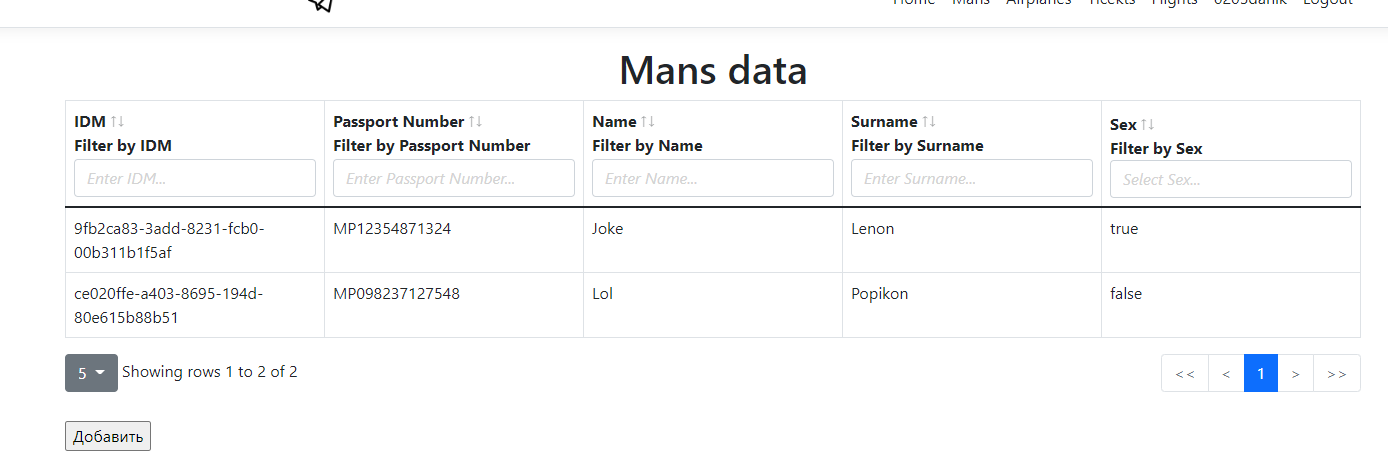


Рисунок 5.4 – Страница данных о людях(пассажирах)

Если необходимо добавление нового человека(пассажира) необходимо нажать кнопку «Добавить» на рисунке 5.4, если удалить существующего, то кнопку «Удалить» рисунок 5.5.

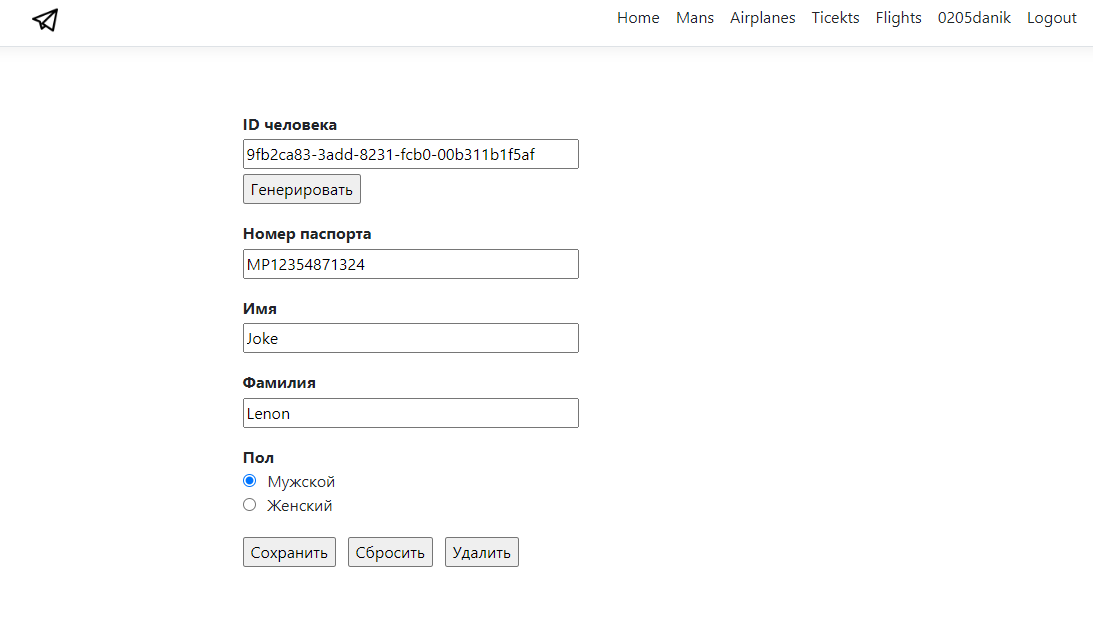


Рисунок 5.5 – Страница редактирования данных о людях(пассажирах)

При добавлении открывается страница на рисунке 5.6.

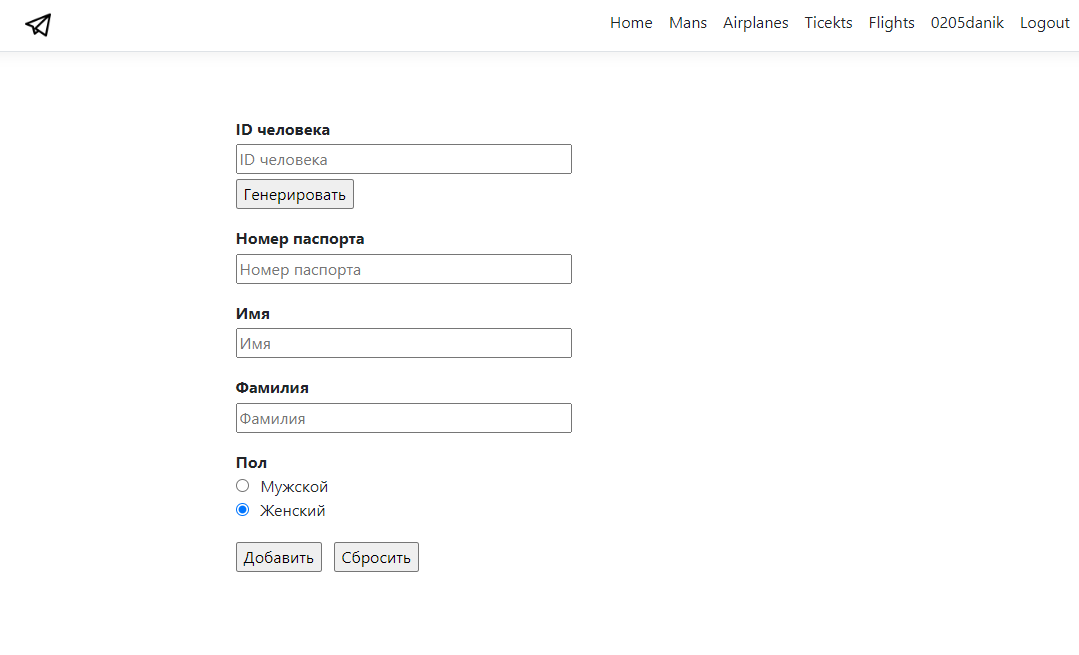


Рисунок 5.6 – Страница добавления данных о новом человеке(пассажире)

Операции добавления, обновления и удаления для всех остальных элементов главного меню идентичны тем, что были описаны выше для людей(пассажирам).

# Вывод

Подводя итог руководства пользователя, можно отметить, что весь необходимый функционал для администрирования был предоставлен в таком же полном объеме, как и функционал для посетителя или организатора, пользователи этих ролей могут влиять друг на друга путем предоставляемых системой средств.

**Заключение**

Подводя итог, следует сказать, что на пути разработки данного продукта был получен бесценный опыт в разработки веб приложения, а именно как frontend, так и backend частей. Были использованы такие технологии как ASP .Net Core, NodeJS, Reactstrap, ReactJS, React bootstrap и другие. Основными языками программирования были выбрано C# и JavaScript.

В результате создания frontend части получился отдельный сервер, отвечающий за представление пользователям информации, которым тот, в свою очередь, запрашивает у теневой части приложения, а именно backend сервера, который и владеет всей информацией о мероприятиях, клиентах, категориях и ролях.

Backend сервер был разработан как REST сервис, и является независимым решением, что позволяет масштабировать данную систему как вертикально, так и горизонтально, что позволяет разработчикам не привязываться к определенным технологиям, и быть мобильными, то есть менять стек технологий при желании. Данный сервис может участвовать в любых бизнес-моделях, как B2C, так и B2B.

Использование выбранного стека технологий целесообразно только в настоящее время, неизвестно, что будет далее, но, как показывает практика, выбранные технологии существуют достаточно долго и только набирают обороты, что может в некоторой степени, гарантировать актуальность данного стека и его востребованность на рынке. Популярность отдельных технологий в случае необходимости позволит достаточно быстро найти новых разработчиков, которые, несомненно, ускорят развитие данного продукта, что влечет за собой увеличение прибыли, роста и связей.

**Список использованной литературы**

1. Travelport [Электронный ресурс]. – Режим доступа: https://www.travelport.com/ – Дата доступа: 06.12.2021
2. Sabre [Электронный ресурс]. – Режим доступа: https://www.sabre.com/. – Дата доступа: 06.12.2021
3. Travelopro [Электронный ресурс]. – Режим доступа: https://www.travelopro.com/worldspan-gds.php. – Дата доступа: 06.12.2021

Приложение A. Листинг контроллера, отвечающего за обработку наличия прав администратора

[ApiController]

[Route("api/[controller]/")]

public class UserDataController : Controller {

private readonly UserManager<ApplicationUser> userManager;

private readonly SignInManager<ApplicationUser> signInManager;

public UserDataController(UserManager<ApplicationUser> userManager,

SignInManager<ApplicationUser> signInManager) {

this.userManager = userManager;

this.signInManager = signInManager;

}

[HttpGet]

[Route("canaccess")]

public async Task<bool> IsAdminAsync(string username) {

if (username != null || username != "") {

ApplicationUser user = await userManager.FindByEmailAsync(username);

bool IsAdmin = await userManager.IsInRoleAsync(user, "Administrator"); // Get the roles for the user

return IsAdmin;

}

return false;

}

}

Приложение B. Листинг метода контроллера «TicketsController»

[HttpGet]

[Route("get")]

public async Task<IEnumerable<Tickets>?> Get(string username) {

bool IsAdmin = await IsAdminAsync(username);

return IsAdmin ? unitOfWork.Ticket.GetList() : unitOfWork.Ticket.GetList().Where(x => x.email == username);

}

Приложение C. Листинг кода, отвечающего за создание нового билета

[HttpPost]

[Route("create")]

public bool Create([FromBody] Tickets ticket) {

//dbf.GetList().Where(x => x.FID == IDF && x.Number\_Free\_places > 0).FirstOrDefault().Number\_Free\_places -= 1;

//dbf - Repository Flights

Flights? isExist = unitOfWork.Flights.GetList().FirstOrDefault(x => x.FID == ticket.IDF&& x.Number\_Free\_places > 0);

if (isExist != null) {

unitOfWork.Flights.GetList().FirstOrDefault(x => x.FID == ticket.IDF && x.Number\_Free\_places > 0).Number\_Free\_places -= 1;

unitOfWork.Ticket.Create(new Tickets() { ID = ticket.ID, IDF = ticket.IDF, MID = ticket.MID == "null" ? null : ticket.MID, email = ticket.email });

unitOfWork.Save();

return true;

}

return false;

}