Министерство науки и высшего образования РФФГАОУ ВПО
Национальный исследовательский технологический университет
«МИСиС»

Институт Информационных технологий и компьютерных наук (ИТКН)
Кафедра Автоматизированных систем управления (АСУ)
09.03.01 Информатика и вычислительная техника (ИВТ)

КУРСОВАЯ РАБОТА

по дисциплине «Разработка клиент-серверных приложений»

Тема: «Авиакомпания, страны, пассажиры»

Студент: Жиделева С.А.
Группа: БИВТ-20-4
Проверил: Рзазаде У.А.

Москва, 2022

Введение

В наше время один из самых быстрых и удобных способов перемещения по планете являются авиаперелёты. Чтобы сделать путешествие ещё более быстрым и комфортным одних двигателей самолёта недостаточно, поэтому авиакомпании стараются создать максимально понятное и лёгкое в обращении сайт для совершения покупок авиабилетов. Такие сайты позволяют клиентам в несколько кликов находить удобные по времени и датам рейсы, смотреть на наличие пересадок, отмечать для себя цены и условия перелёта: максимально разрешённый вес багажа и ручной клади, пускаются ли животные и т. д. Многие пользователи не задумываются, что работают не с просто красиво оформленным сайтов, но целой системой, названной клиент-серверным приложением, соединённой с базой данных для более обширного анализа имеющихся данных и, следовательно, ёмкого и точного ответа на заброс пользователя.

Так как на удобство покупки авиабилета влияет не только цена и репутация авиакомпании, но и такой человеческий фактор как «приятность глазу». Этот фактор обязывает разработчиков уделять внимание не только работе сайта, но и его оформлению. Кто-то имеет большие временные ресурсы для разработки красивого и приятного глазу веб-страницы, однако кто-то прибегает к использованию шаблонов и конструкторов для создания сайта.

В этой курсовой работе при построении frontend-составляющей страницы были использованы шаблоны и элементы предоставленные Bootstrap, однако также имеются собственноручно написанные части.

1. Постановка задачи

На курсе «Разработка клиент-серверных приложений» была поставлена задача разработать собственное клиент-серверное приложение со всеми frontend и backend составляющими.

Содержание frontend-части курсовой:

- Приветственное сообщение, кратко описывающее назначение приложения;
- Меню, с возможностью перехода в каталог и на главную страницу из любой части сайта;
- Каталог стран, куда продаются авиабилеты;
- Карточки туров, содержащие описание, стоимость и наполнение тура;
- Footer сайта.

Содержание backend-части курсовой:

- Локальный сервер;
- СУБД PostgreSQL.Содержание сервера:
    - Контроллеры моделей для обработки GPPD – запросов;
    - Три и более моделей;
- База данных с таблицами, соответствующими созданным моделями;
- CRUD для созданных таблиц.

2. Проектирование приложения

Для разработки использовалась среда «Rider» от JetBrains. Web API – шаблон, использующийся для реализации приложения с Entity Framework. Для возможности работы с базой данных PostgreSQL установился специальный «Postgres Driver».

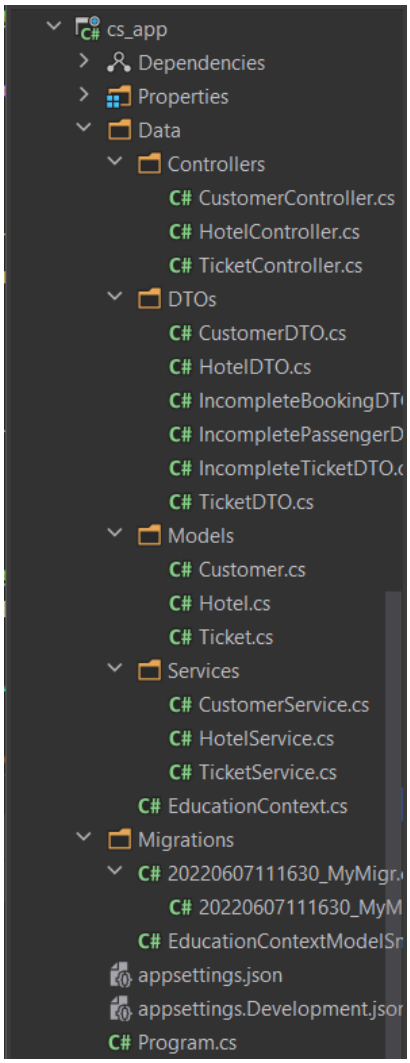

Рисунок 1. Составляющие сервера

Папка «**Controllers**» содержит в себе контроллеры, через которые сервер получает запросы, отправленные клиентами. Именно они реализуют технологию API – Get, Post, Put, Delete (GPPD)

Папка «**DTOs**» включает в себя DTO-файлы, предназначенные для обмена данными внутри приложения (без связи с БД).

Папка «**Models**» содержит модели, являющиеся представлениями записей базы данных.

Папка «**Services**» содержит классы, связывающие БД и контроллеры.

Папка «**Migrations**» отвечает за миграцию, помогающая сравнивать состояния экземпляров сервисов и находить несоответствия.

2.1. Описание архитектуры базы данных

В ходе выполнения технического задания данной курсовой работы, была использована СУБД PostgreSQL.

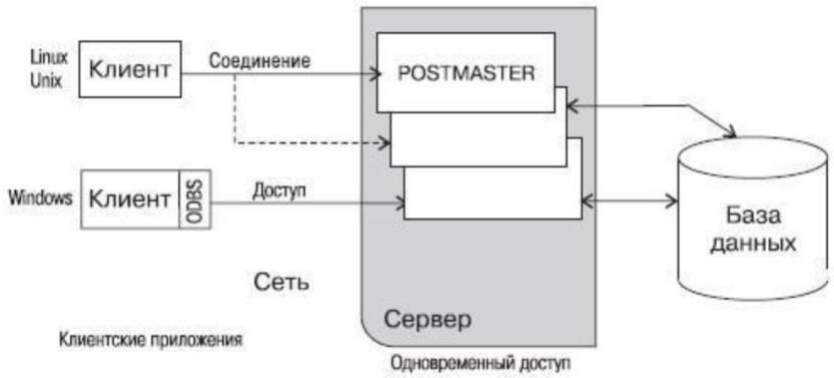

Рисунок 2. Структура приложения с использованием СУБД на базе PostgreSQL

Причины выбора «PostgreSQL» для реализации БД:

- Полная SQL-совместимость.
- Объектно-ориентированность: PostgreSQL — не только реляционная, но и объектно-ориентированная СУБД.
- Доступ из приложений к этим базы осуществляется с помощью механизма базы данных. Клиентские программы не могут получить доступ к этим независимо, даже если они работают на том же компьютере, на котором осуществляется серверный процесс.
- Разделение контрагентов и сервера в PostgreSQL помогает сформировать распределенную систему.

2.1.1 Вербальная модель

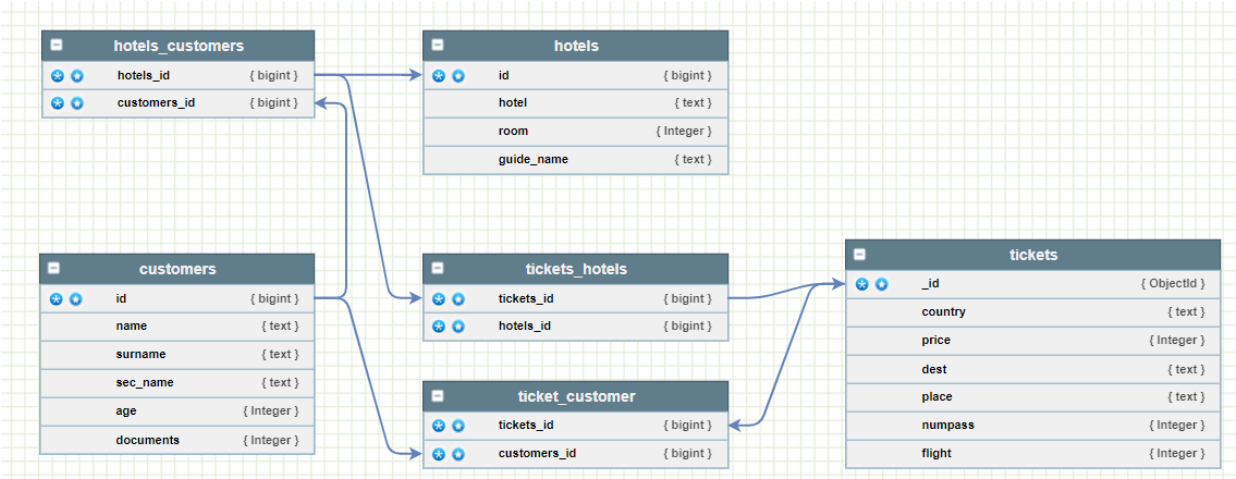

Рисунок 3. Схема базы данных

База данных содержит 3 таблицы, которые представляют основные сущности базы данных: билеты, отели, клиенты – и также 3 таблицы для связи через идентификаторы. База данных предназначена для работы турагентства, которое продает путёвки в различные страны. Каждая таблица имеет уникальный первичный ключ для быстрого доступа к данным, а с помощью внешних ключей таблицы связаны между собой.

2.1.2 Реляционная модель

При разработке базы данных были созданы следующие сущности со свойственными им атрибутами:

1. Таблица «**tickets**» содержит всю информацию о рейсе
    a. Id – идентификатор билета
    b. Country – название страны
    c. Price – цена за перелёт
    d. Dest – город прибытья
    e. Numpass – номер посадочного талона
    f. Flight – номер рейса
2. Таблица «**hotels**» включает в себя полную информацию о бронировании:
    a. Id – идентификатор отеля

b. Hotel – название отеля
c. Room – номер комнаты в отеле
d. Guide_name – имя гида

3. Таблица «**customers**» содержит основную информацию о пассажире
   a. Id – идентификатор клиента
   b. Name – имя клиента
   c. Surname – фамилия клиента
   d. Sec_name – отчество клиента
   e. Age – возраст клиента
   f. Documents – документы (паспортные данные)

2.1.3 Реализация базы данных

Для реализации базы данных использовалась утилита pgAdmin4.

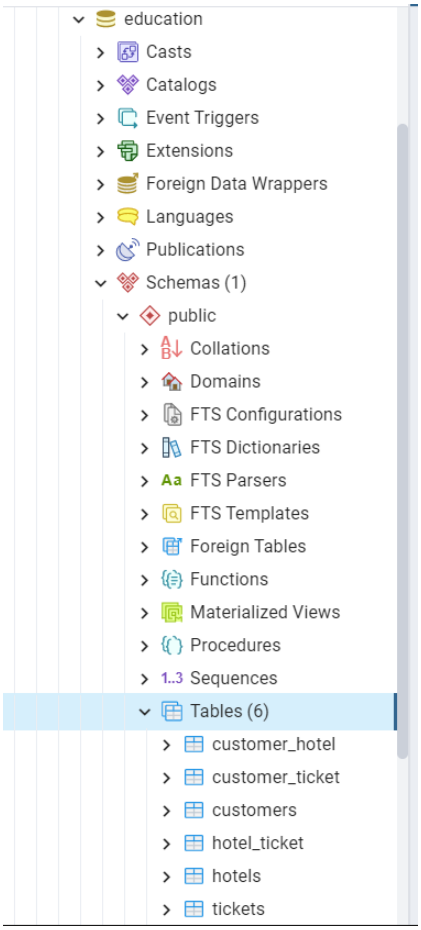

Рисунок 4. Структура БД

В качестве заготовки использовалась база данных «education», которая позднее была видоизменена и настроена под нужды приложения, связанного с авиаперелётами.

2.2 Описание архитектуры приложения

Архитектура «Клиент–Сервер» подразумевает разделение процессов предоставления услуг и отправки запросов на них на различных компьютерах в сети, каждый из которых осуществляют собственные задачи вне зависимости от прочих.

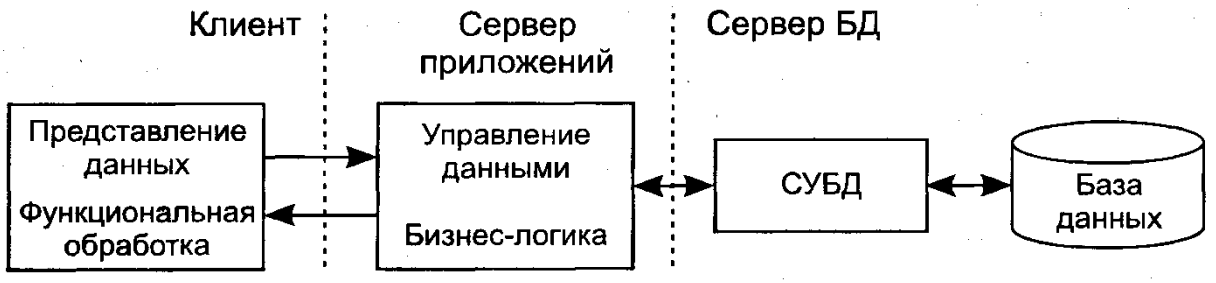

Рисунок 5. Архитектура «клиент-сервер»

В архитектуре «Клиент-Сервер» несколько компьютеров-клиентов отправляют запросы и получают услуги от централизованной служебной машины – сервера, которая также может называться хост-системой (host – от англ. «хозяин»).

Клиентская машина предоставляет пользователю так называемый «дружественный интерфейс» (user-friendly interface), дабы облегчить его взаимодействие с сервером[4].

2.3 Описание web-интерфейса

Интерфейс написан с использованием технологий HTML, CSS, а также, Bootstrap.

Веб-интерфейс — веб-страница или совокупность веб-страниц, предоставляющая пользовательский интерфейс для взаимодействия с сервисом или устройством посредством протокола HTTP и веб-браузера.

HTML – язык разметки веб-страниц для их просмотра в браузере.

CSS – язык изменения внешнего вида веб-документа.

Bootstrap – инструмент для создания сайтов, включающий в себя HTML и CSS заготовки для оформления веб-документов.

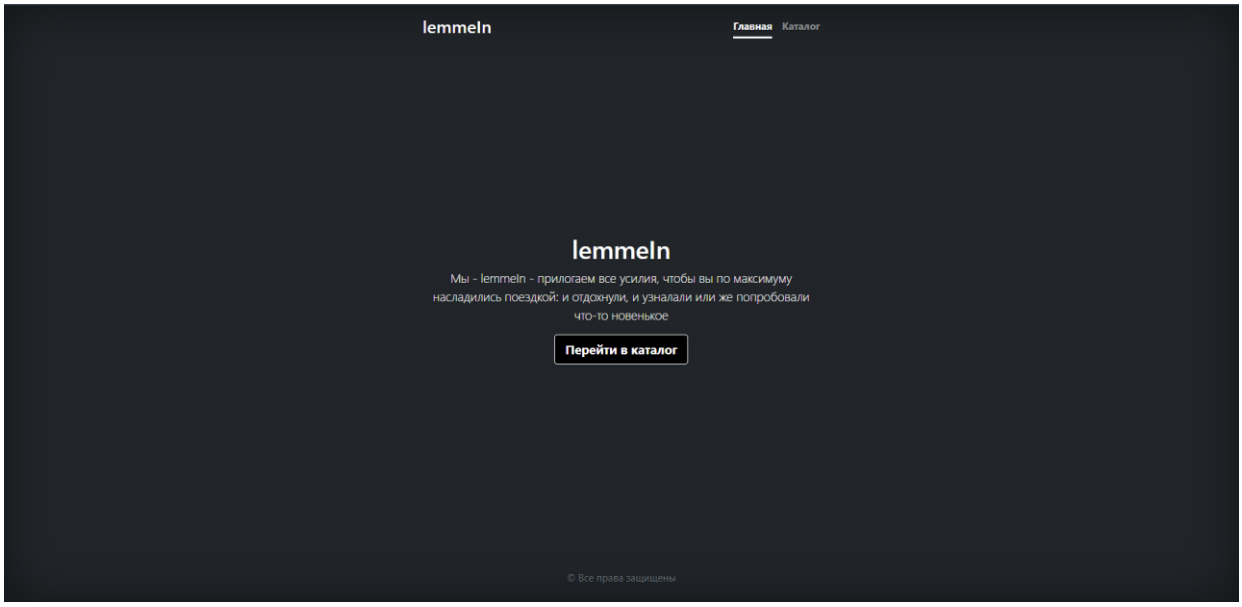

Рисунок 6. Главная страница

На главной странице расположено меню, выполняющее навигационную функцию. Помимо этого, имеется заголовок – название сайта, и приветственное сообщение. Внизу расположен футер сайта со значком копирайта.

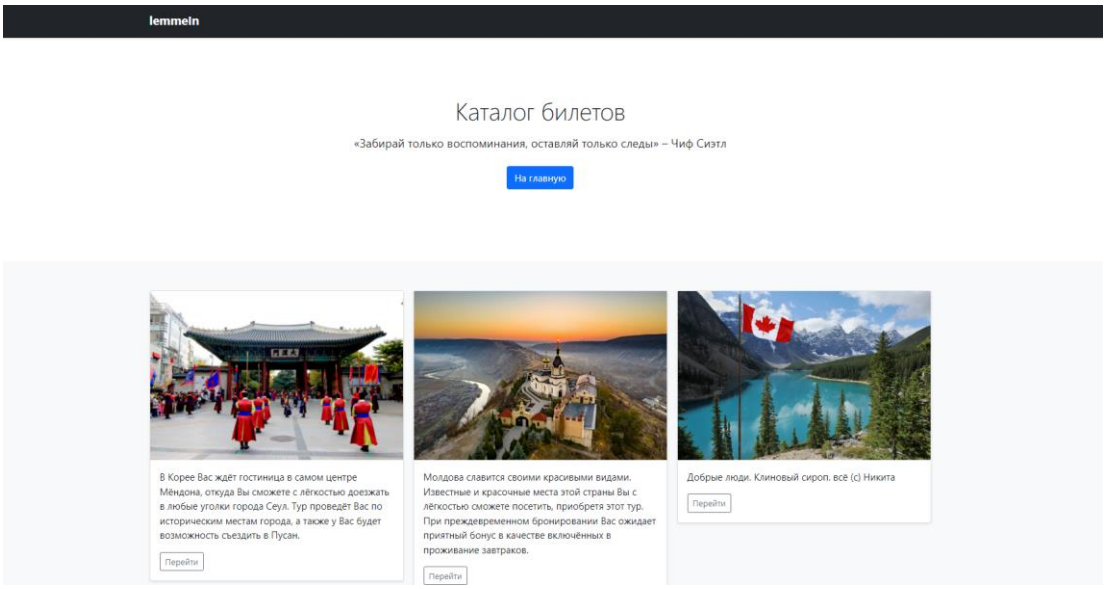

Рисунок 7. Страница каталога

На странице каталога можно обратить внимание на галерею путёвок, выполненную при помощи Bootstrap. Каждая путёвка имеет краткое описание и стоимость, а также, фотографию той страны, куда можно отправиться и кнопку для перехода на страницу выбранной путёвки.

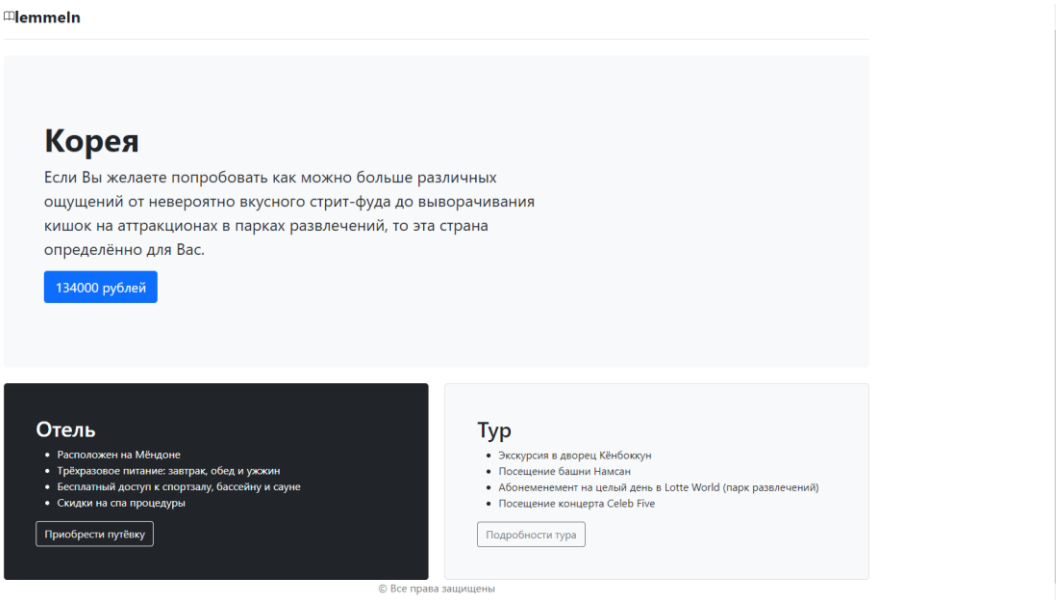

Рисунок 8. Страница путёвки

Перейдя на страницу билета, можно прочитать условия проживания в отеле по данной путёвки, а также этапы проведения самого тура.

2.4 Проектирование web-сервера

Для реализации сервера были использованы объектно-ориентированный язык программирования C# и фреймворк Entity Framework.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис очень близок к C++ и Java. Язык имеет статическую типизацию,
поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и приемы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Entity Framework Core — это современный модуль сопоставления "объект — база данных" для .NET. Он поддерживает запросы LINQ, отслеживание изменений, обновления и миграции схемы. EF Core работает с многими базами данных, включая базы данных SQL (локальные и в Azure), SQLite, MySQL, PostgreSQL и Azure Cosmos DB.

Веб-сервер — это компьютер, на котором хранятся файлы сайтов (HTML-документы, CSS-стили, JavaScript-файлы, различный контент), и который доставляет их на веб-браузер на устройстве конечного пользователя. Также под веб-сервером понимается ПО, с помощью которого контролируется доступ веб-пользователей к размещенным на сервере файлам. Такое ПО называется HTTP-сервером и работает с URL-адресами и HTTP-протоколами.

Веб-серверы для публикации сайтов делятся на статические и динамические.

Статические веб-серверы (стоки) — это «железо» с установленным на нем ПО для HTTP, которое направляет размещенные файлы в браузер в неизменном виде.

В динамических веб-серверах на статические веб-сервера устанавливается дополнительное программное обеспечение, чаще всего сервера приложения и базы данных. В таких серверах исходные файлы изменяются перед отправкой по HTTP.

2.5 Реализация необходимых классов сервера

Контроллер — это точка обращения на сервер, содержащая набор методов GPPD (Get, Post, Put, Delete), вызывающихся при соответствующем запросе.

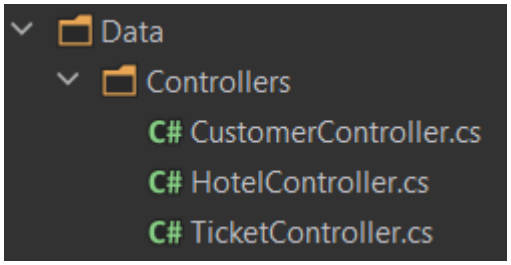

Рисунок 10. Контроллеры

DTO — это не привязанные к базе данных модели, которые используются для обмена данными исключительно внутри сервера.

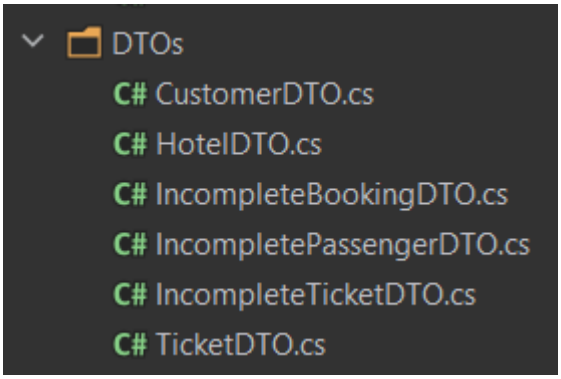

Рисунок 11. DTO

Модели — это сущности и записи базы данных, реализованные в виде классов.

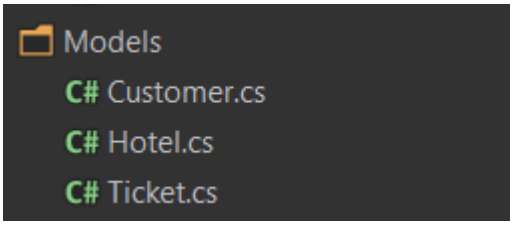

Рисунок 12. Модели

Сервисы — это классы, взаимодействующие с базой данных иреализовывающие связь между ней и контроллерами.

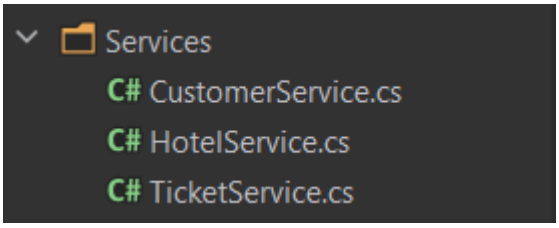

Рисунок 13. Сервисы

3 Внешний вид «Swagger»

Swagger — это фреймворк для спецификации RESTful API. Он дает возможность не только интерактивно просматриватьспецификацию, но и отправлять запросы (Swagger

UI).

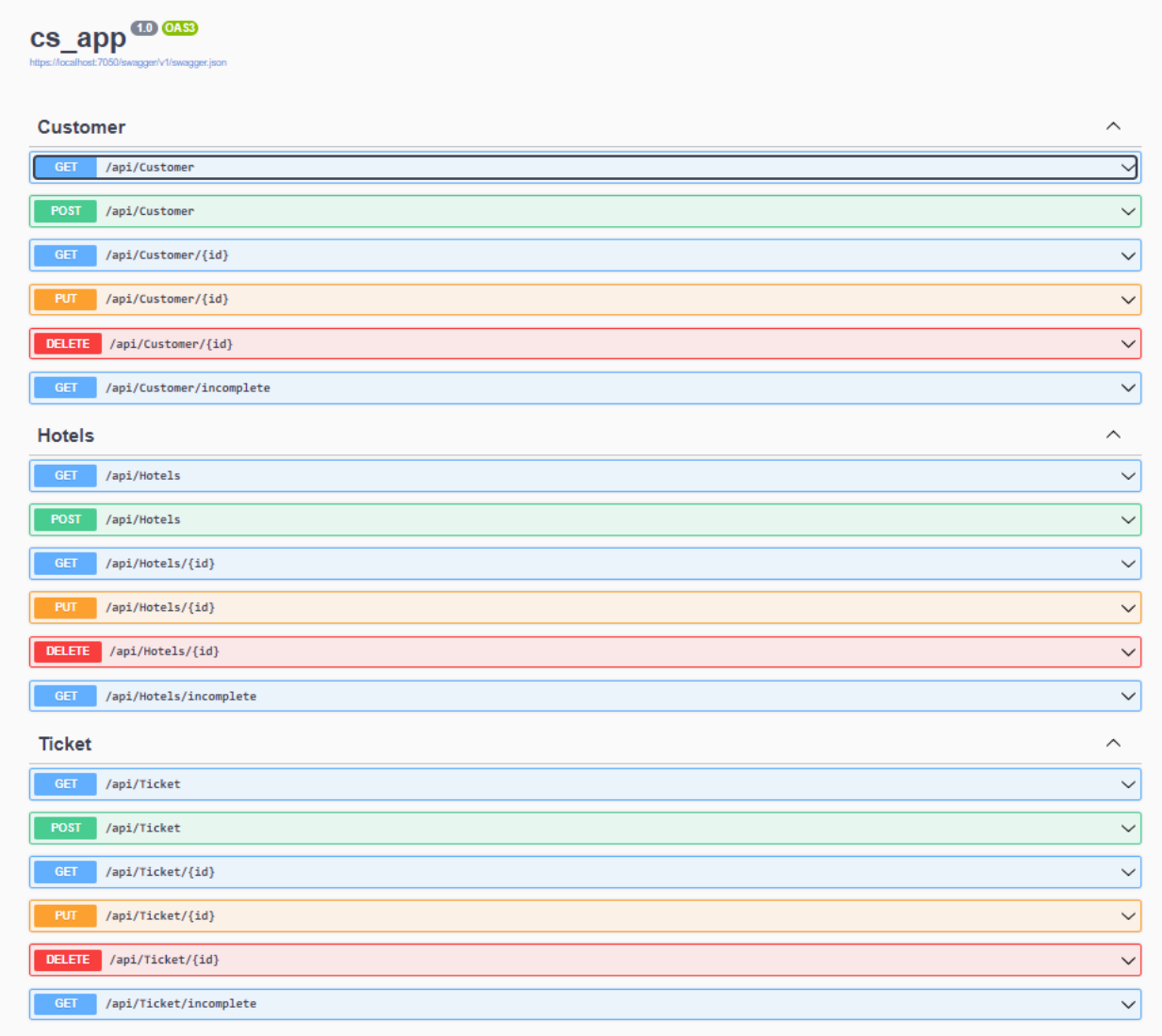

Рисунок 15. Swagger

Также в нём возможно при помощи Swagger Codegen сгенерировать клиента или сервер поспецификации API Swagger,.

При изучении интерфейса, в глаза бросаются разноцветные кнопки. Используя их, можно для каждой из сущностей создать запрос при помощи методов GPPD (Get, Post, Put, Delete).

4 Внешний вид «Blazor»

Blazor – это экспериментальный веб UI – фреймворк на базе C#, Razor и HTML, который работает непосредственно в браузере посредством WebAssembly. Создание Blazor ставило перед собой задачу упростить процесс построения простых и качественных одностраничных приложений, которые могут быть запущены в любом браузере.

Рисунок 16. Blazor

В случае с авиаперелётами создаётся информация о рейсе, а после этого подкрепляются данные о пассажирах, вылетающих именно этим рейсом. В качестве примера был создан «нулевой пассажир», дабы показать пример работы приложения.

Заключение

В процессе выполнения курсовой работы был создан сайт для турагентства, продающего билеты в разные страны. Для хранения данных была использована СУБД PostgreSQL, как наиболее развитая открытая СУБД в мире. Для разработки серверной части веб-приложения использовался фреймворк «Spring Framework», как один из наиболее актуальных для данной цели, в разработке клиентского приложения использован фреймворк «Bootstrap», а также язык разметки HTML и стили CSS.

Данное приложение помогает желающим купить авиабилеты подыскать путёвки на выгодных для них условиях и с бонусами от авиакомпании. Проект возможно развивать и дальше: добавить систему логинов и паролей, чтобы у клиентов были собственные личные кабинеты, что позволит отслеживать предпочтения пользователя с целью предложения им особых персональных бонусов в путёвках; добавить систему «Милей», суть которой набрать пассажиром определённое количество миль во время поездок, что приведёт к большой скидке на следующий полёт или же клиент сам сможет выбрать большой бонус; реализовать возможность выбора места в самолёте и т.д.

Во время разработки клиент-серверного приложения использовалась среда разработки «Rider» и язык программирования C#. Web-интерфейс был выполнен при помощи HTML, CSS, Bootstrap. Для тестирования функционала использовался Web UI Framework «Blazor». Реализация и интеграция СУБД происходили с помощью PostgreSQL и pgAdmin4, а также модуля Entity Framework.

В ходе проекта цели были достигнуты, и поставленная задача была выполнена.

Список литературы

1. Фримен Адам «ASP.NET Core MVC 2 с примерами на C# дляпрофессионалов»; пер. с англ. Ю.Н.Артеменко. – 6-е изд.: с. 991
2. Brian Mulloy «Web API Design: Crafting Interfaces that Developers Love»; с. 38
3. HTMLBook – http://htmlbook.ru
4. Архитектура клиент-сервер – https://habr.com/ru/post/495698/