МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-изданий)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии и стандарты проектирования»

Тема программное средство «Бронирование столика»

Исполнитель

студент 2 курса группы 5 Тихомиров Тимофей Михайлович

(Ф.И.О.)

Руководитель работы ассистент Северинчик Н.А.

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

Курсовой проект защищён с оценкой

Председатель Пацей Н.В.

(подпись)

Минск 2021

Содержание

[ВВЕДЕНИЕ 3](#_Toc72975840)

[1. Аналитический обзор литературы 4](#_Toc72975841)

[2. Анализ требований к программному средству и разработка функциональных требований 6](#_Toc72975842)

[3. Проектирование программного средства 9](#_Toc72975843)

[3.1 Проектирование архитектуры приложения 9](#_Toc72975844)

[3.2 Проектирование базы данных 12](#_Toc72975845)

[3.3 Проектирование доступа к базе данных 15](#_Toc72975846)

[3.4 Проектирование логики сценариев использования 16](#_Toc72975847)

[4. Создание (реализация) программного средства 18](#_Toc72975848)

[4.1 Диаграмма классов 18](#_Toc72975849)

[4.2 Реализация архитектуры MVVM 19](#_Toc72975850)

[4.3 Реализация представления 20](#_Toc72975851)

[5. Тестирование, проверка работоспособности и анализ полученных данных 22](#_Toc72975852)

[6. Руководство по использованию 26](#_Toc72975853)

[ЗАКЛЮЧЕНИЕ 37](#_Toc72975854)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 38](#_Toc72975855)

[Приложение А 39](#_Toc72975856)

[Приложение Б 41](#_Toc72975857)

# ВВЕДЕНИЕ

Данный курсовой проект посвящен разработке программного средства «Бронирование столика», основной целью которого является возможность предварительного заказа столика и меню. Основными возможностями клиента являются выполнение регистрации, авторизации, просмотр данных о своих заказах в личном кабинете, просмотр списка ресторанов, добавление и отмена заказа.

В качестве интерфейса прикладного программирования был выбран обширный API-интерфейс — Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом.

Для работы с WPF использовался объектно-ориентированный язык программирования с С-подобным синтаксисом — С#, разработанный для создания приложений на платформе Microsoft .NET Framework.

Хранение данных осуществляется в Microsoft SQL Server

# Аналитический обзор литературы

Немаловажным этапом в разработке программного продукта является аналитический обзор прототипов и литературных источников.

На сегодняшний день можно встретить достаточно большое количество программных решений, разработанных для предварительного заказа в ресторане. Я рассмотрел несколько приложений для бронирования столика, которые по своей сути аналоги моего программного средства, однако функционал везде абсолютно разный. В качестве примеров программных средств я взял сайты opentable.co.uk и discoveri.by.

Opentable.co.uk — это сайт, для заказа столика в ресторане. Существует мобильное приложение, в котором пользователи могут оформить заказ, а также найти множество дополнительной информации о компании.

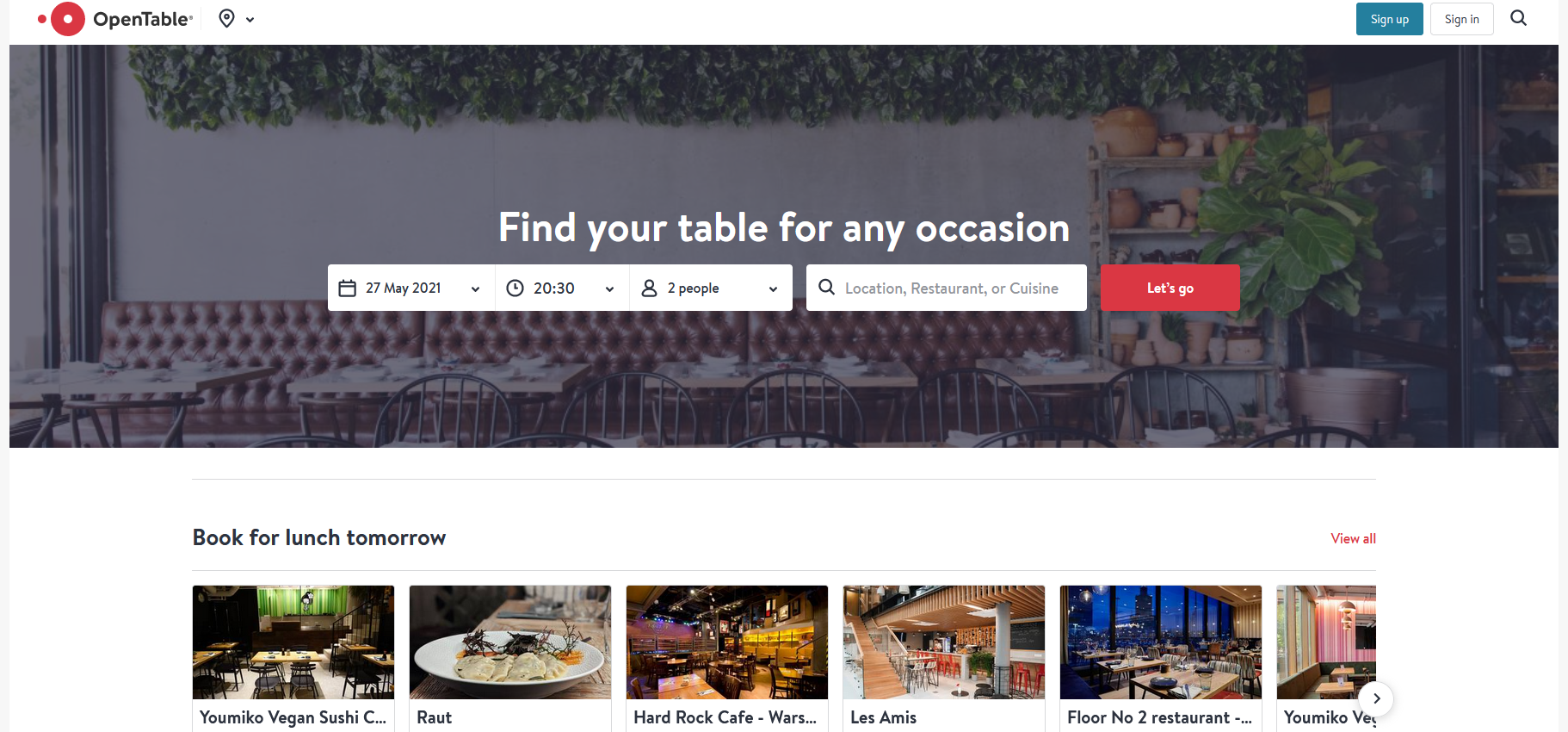


Рисунок 1.1 – главное окно

Discoveri.by — сайт белорусского ресторана пиццерии «Дискавери». Я разобрал возможности приложения от данной компании для заказа столика и проанализировал какие функции данного приложения я мог бы использовать сам при проектировании моего приложения.

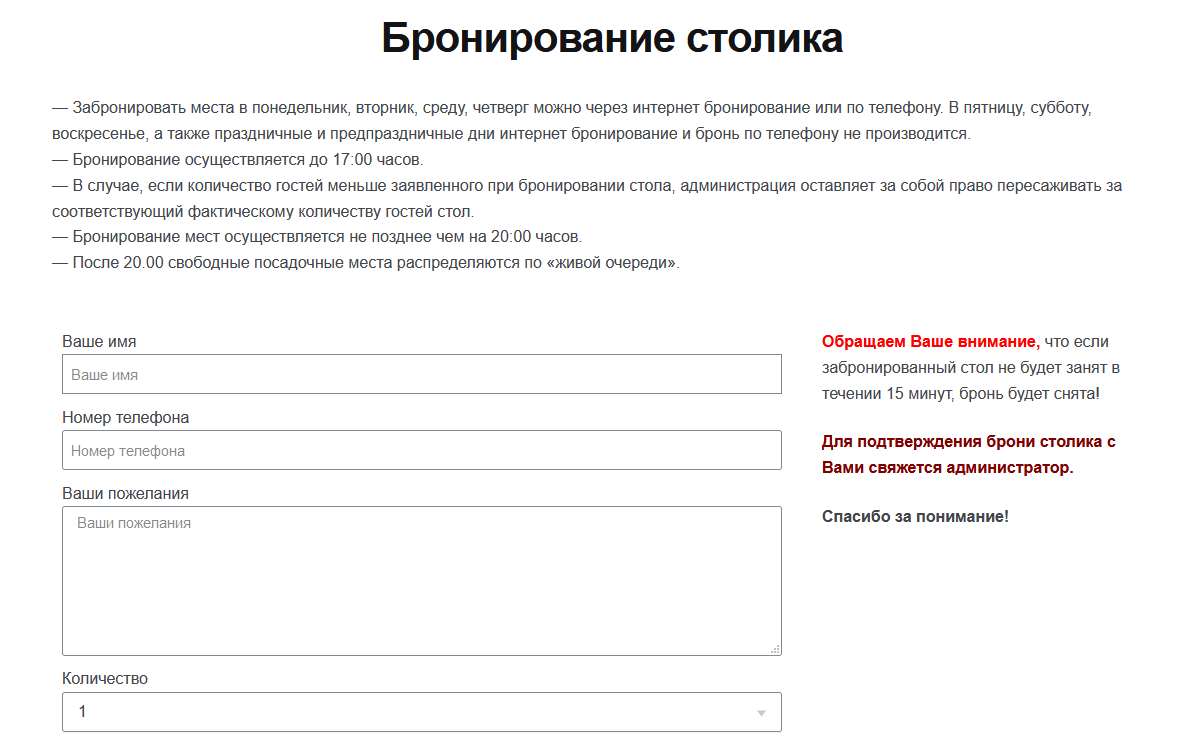


Рисунок 1.2 – Форма для оформления заказа

# Анализ требований к программному средству и разработка функциональных требований

Задачей курсового проекта является разработка приложения, предназначенного для бронирования столика в ресторане, а также получение информации о статусе заказа. В функции администратора входит поддержка работы с базой данных, добавление новых ресторанов и подтверждение заказа с оповещением клиента о выполнении заказа по электронной почте.

Программный продукт должен быть реализовать на объектно-ориентированном языке программирования C# с использованием технологии WPF. Windows Presentation Foundation (WPF) — система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является DirectX, в отличие от Window Forms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX.

Также существует урезанная версия CLR, называющаяся WPF/E, она же известна как Silverlight.

База данных была разработана в Microsoft SQL Server.

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире.

SQL Server характеризуется такими особенностями как:

* Производительность. SQL Server работает очень быстро.
* Надежность и безопасность. SQL Server предоставляет шифрование данных.
* Простота. С данной СУБД относительно легко работать и вести администрирование.

Для организации баз данных MS SQL Server использует реляционную модель, которая предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

Основной используемый язык запросов —Transact-SQL — реализован на структурированном языке запросов (SQL) с расширениями.

Для осуществления связи между базой данных и приложением на C# необходим посредник, которым будет являться технология Entity Framework. Она основана на платформе .NET Framework и предназначена для работы с данными.

Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Эта технология предоставляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

* Database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных;
* Model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере;
* Code first: разработчик создает класс модели данных, которые будут храниться в базе данных, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

В курсовом проектировании для разработки был выбран третий подход.

Приложение необходимо реализовать с использованием шаблона проектирования MVVM и с организацией взаимодействия с базой данных. В качестве клиентской части необходимо реализовать оконное приложение с использованием стандартных и иных библиотек. Приложение должно иметь возможность запускаться без использования интегрированных средств разработки.

Для разработанного продукта необходимо предусмотреть следующие задачи:

* Главное поле для отображения данных;
* Интуитивно понятный интерфейс;
* Создание пользовательских записей и их просмотр.

По итогу работы над продуктом, его необходимо протестировать и отладить, проанализировать его возможности и, по необходимости, внести дополнения или улучшения проекта, с возможностью дальнейшего расширения проекта.

Программное средство предоставляет пользователю следующие функциональные возможности:

* Выполнение регистрации и авторизации.
* Создание заказов.
* Просмотр заказов.
* Добавление заказов в корзину.
* Удаление заказов из корзины.
* Изменение личной информации.
* Просмотр различной информации о магазине

К функциям администратора относятся:

* Добавление товаров.
* Поддержка работы с базой данных.

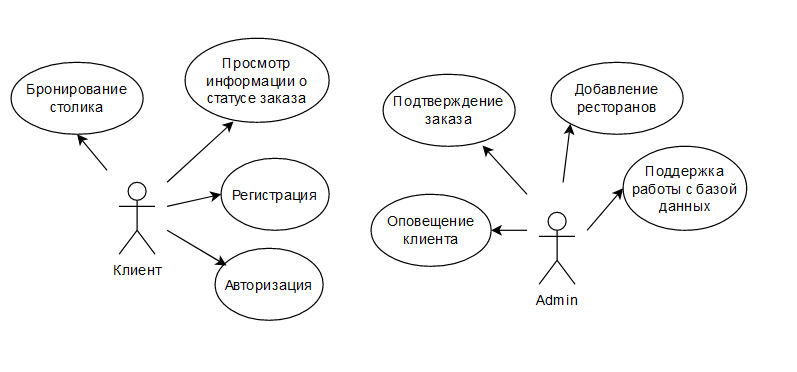


Рисунок 2.1 – «диаграмма возможностей»

# Проектирование программного средства

Проектирование программного средства — процесс создания проекта программного обеспечения. Целью проектировaния является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Исходные условия задачи уже были сформулированы во втором разделе данной пояснительной записки. Этап проектирования подразумевает их анализ.

## **3.1 Проектирование архитектуры приложения**

Архитектура программного обеспечения — совокупность вaжнейших решений об организации программной системы. Архитектура включает:

* выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
* соединение выбрaнных элементов структуры и поведения во всё более крупные системы;
* архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

Для удовлетворения проектируемой системы различным атрибутам качества применяются различные архитектурные шаблоны (паттерны). В разрабатываемом приложении используется архитектурный шаблон Model-View-ViewModel (MVVM).

Шаблон MVVM имеет три основных слоя: модель, которая представляет бизнес-логику приложения, представление пользовательского интерфейса, и представление-модель, в котором содержится вся логика построения графического интерфейса и ссылка на модель, поэтому он выступает в качестве модели для представления.

На рисунке 3.1 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.

Рисунок 3.1 – Структура шаблона MVVM

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Так как пользовательский интерфейс и качество его реализации играет далеко не последнее место в конечном результате, разработка эффективного интерфейса, приятного и удобного для конечного пользователя, является важной задачей. Поэтому для хорошего проектирования View необходимо понять, как пользователь будет взаимодействовать с приложением. Для этого была составлена схема на рисунке 3.2, на которой представлен принцип работы приложения с точки зрения пользователя.



Рисунок 3.2 – Принцип работы приложения

На диаграмме 3.2 видно, что пользователь при запуске приложения будет попадать на главную страницу, откуда он сможет зарегистрироваться, авторизоваться, просмотреть список продуктов, расположение магазинов, узнать информацию о поддержке, открыть и просмотреть корзину, а также узнать информацию о личных данных. Некоторые возможности доступны только после авторизации.

После входа в приложение пользователю откроется окно приложения с 3 вкладками меню:

* Home;
* Login;
* Register;

На вкладке «Home» пользователь может выбрать ресторан и перейти к оформлению заказа.

На вкладке «Login» авторизация в систему.

На вкладке «Register» пользователь может завести новый аккаунт, чтобы в дальнейшем можно было оформлять заказы.

При авторизации пользователя с ролью «Администратор» появляется еще один соответствующий пункт на странице личного аккаунта. Для администратора появляется доступ к добавлению ресторанов и меню.

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться Представление, чтобы влиять на Модель.

Model или Модель представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения.

Для разработки приложения в качестве сервиса для хранения данных была выбрана СУБД Microsoft SQL Server. На диаграмме размещение на рисунке 3.3 показано графическое представление инфраструктуры приложения.

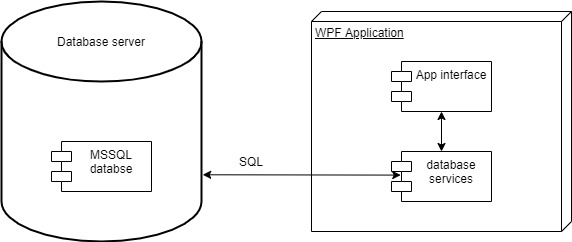


Рисунок 3.3 – Диаграмма размещения

## **Проектирование базы данных**

В приложении В описана структура проекта по пакетам классов.

Проектирование баз данных — процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования базы данных:

* обеспечение хранения в БД всей необходимой информации;
* обеспечение возможности получения данных по всем необходимым запросам;
* сокращение избыточности и дублирования данных;
* обеспечение целостности базы данных.

Проектирование базы данных проводится в два этапа: концептуальное (инфологическое) и логическое (даталогическое) проектирование.

Концептуальное (инфологическое) проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня aбстракции. В результате этого этапа создаётся ER-модель. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных.

Основными понятиями ER-модели являются: сущность, связь и атрибут.

Сущность – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна.

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация обычно является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

Атрибут сущности это любая детaль, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности.

Были определены необходимые связи. Например, между сущностями список покупок и заказ установлена связь один-к-одному.

Для каждой сущности были выделены свои атрибуты. Например, для сущности заказ были выделены такие атрибуты, как идентификатор заказа, идентификатор пользователя, по которому можно идентифицировать пользователя, который сделал заказ, дата покупки, общая стоимость заказа, количество товара в заказе.

Логическое (даталогическое) проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели дaнных логическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Логическая модель базы данных представлена на рисунке 3.4

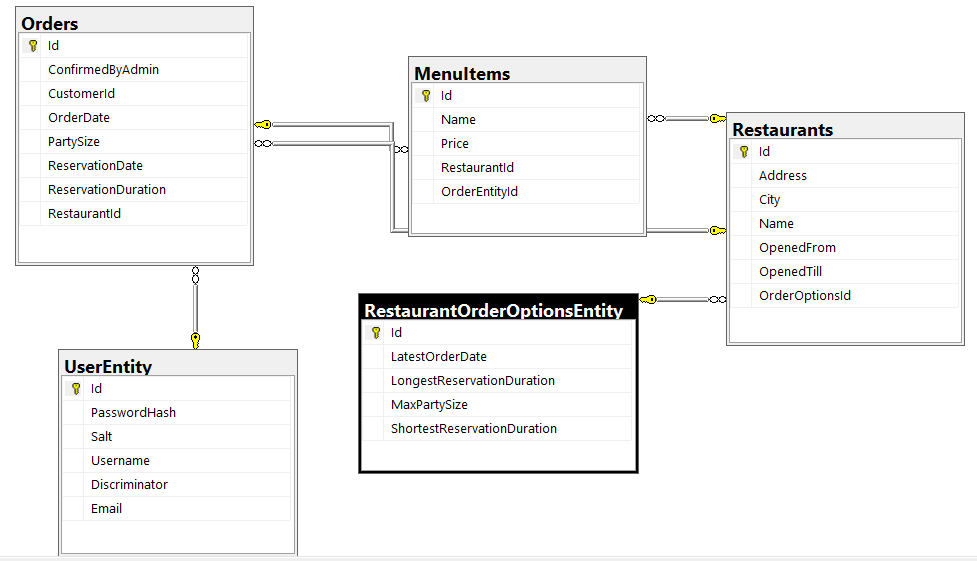


Рисунок 3.4 - Логическая модель базы данных

Всего в базе данных содержится 5 таблиц. В таблице UserEntity хранятся все пользователи, которые зарегистрировались в приложении, каждый из которых идентифицируется своим идентификатором. В таблице Restaurants содержится список ресторанов. Таблица MenuItems содержит список товаров, которые являются частью заказа. Таблица Orders хранит в себе информацию о каждом из заказов, совершенных пользователями

Таблица UserEntity содержит 6 столбцов:

* Id;
* Username;
* Password\_Hash;
* Disctriminator;
* Salt;
* Email

В столбце Id хранится уникальный идентификатор пользователя, который выдается автоматически при регистрации. В столбце Username хранится соответственно имя пользователя, в столбце Email – адрес по которому пользователю будет отправляться уведомление, Login и Password\_Hash хранят основные данные, которые пользователь вводит при регистрации, а в дальнейшем при авторизации в аккаунт. Discriminator определяет тип пользователя, который зашел в аккаунт, поскольку в приложении предусмотрены 2 роли: администратор и обычный пользователь.

Таблица Order содержит 8 столбцов:

* Order\_Id;
* Customer\_Id;
* Order\_Date;
* Total\_Price;
* ConfirmedByAdmin
* Party Size;
* ReservationDate;
* ReservationDuration;

В столбце Order\_Id содержится идентификатор заказа, при создании заметка ему автоматически присваивается свой идентификатор. В столбце User\_Id находится уникальный идентификатор пользователя, который выполнил заказ. Order\_Date − время, в которое был зарегистрирован заказ. Total\_Price − содержит полную стоимость заказа. ReservationDate − день заказа, ReservationDuration.

MenuItems содержит 5 столбцов:

* Id;
* Name;
* Price.
* RestaurantId
* OrderEntityId

В столбце Order\_List\_Id содержится идентификатор записи в листе покупок. Product\_Id − уникальный идентификатор продукта из листа покупок. Order\_Id − идентификатор заказа, по которому определяется к какому заказу относится данная запись.

Таблица Restaurants содержит 7 столбцов:

* Id;
* Address;
* City;
* Name;
* OpenedFrom;
* OpenedTill;
* OrderOptionsId;

В данной таблице хранится информация о каждом из продуктов, которые продаются в этом магазине. Столбец Product\_Id хранит уникальный идентификатор продукта, столбец Category отражает категорию, к которой относится товар, Столбцы Model, Type, Connection\_Type, Weight, Backlight, Height, Width, dpi, Microphone, Gaming\_mode хранят описание и техническую информацию о каждом из товаров. Столбец Price хранит информацию о стоимости товара. В столбце Number\_on\_warehouse записывается информация о количестве данного товара на складе. В столбцах Picture\_Main, Picture1, Picture2, MenuPicture хранятся картинки данного товара в байтовом представлении, которые используются для отображения внешнего вида товара на вкладках приложения. Столбцы Description и ShortDescription содержат полное и краткое описание товара соответственно.

## **3.3** **Проектирование доступа к базе данных**

Для доступа к базе данных используется Entity Framework Core. Этот подход предоставляет ряд существенных преимуществ: нам не нужно беспокоиться о коде доступа к данным и не нужно знать деталей работы СУБД SQL Server и синтаксиса языка запросов T-SQL, вместо этого мы работаем с таблицами базы данных как с классами C#, с полями этих таблиц - как со свойствами классов, а синтаксис SQL-запросов заменен на  [LINQ](https://professorweb.ru/my/LINQ/base/level1/info_linq.php). Entity Framework берет на себя обязанности по преобразованию кода C# в SQL-инструкции.

* Как говорилось ранее, в своей работе я использовал подход Code-First, с помощью которого создавался набор классов, которые отражают модель конкретной базы данных;

# Создание (реализация) программного средства

Следующим этапом разработки приложения является непосредственная реализация программного решения в соответствии с уже сформированными требованиями и шаблонами.

4.1 Диаграмма классов

Структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов, методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

На данной диаграмме классов изображены классы программы, методы, поля и свойства классов. Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру (поля, методы) и типы отношений (наследование, реализация интерфейсов). Диаграмма классов проекта «Бронирование столика» представлен на рисунке 4.1.

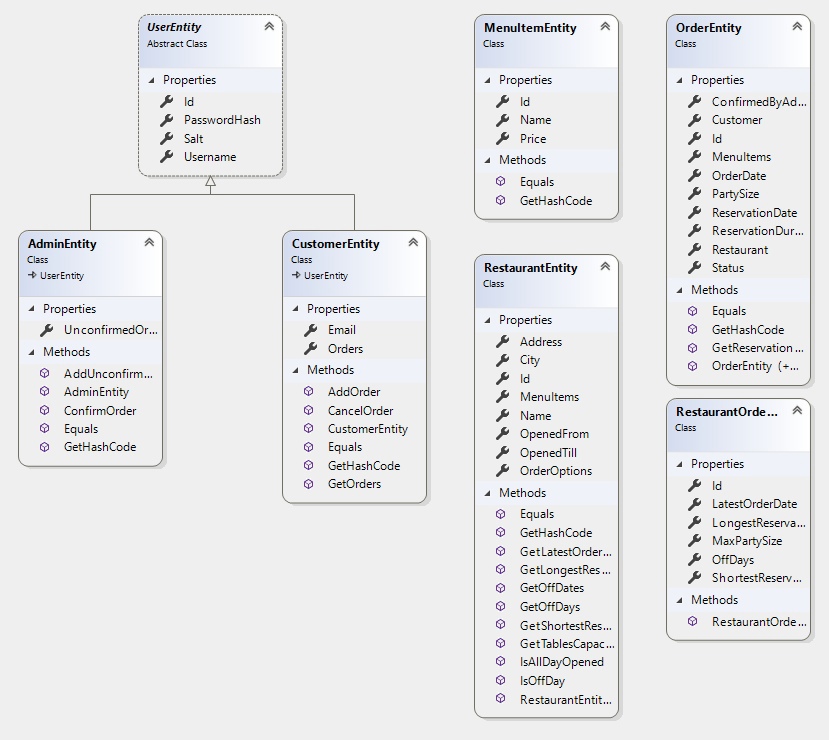


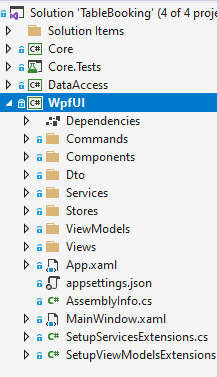
Рисунок 4.1 − Диаграмма классов

4.2 Реализация архитектуры MVVM

Для реализации паттерна MVVM файлы программы были распределены по соответствующим директориям и реализовали соответствующие функции. Разделение проекта на логические модули представлено на рисунке 4.3.

**4.2 Реализация архитектуры MVVM**

Для реализации паттерна MVVM файлы программы были распределены по соответствующим директориям и реализовали соответствующие функции. Разделение проекта на логические модули представлено на рисунке 4.2.

  
Рисунок 4.2 – Логические модули проекта

В директории Services содержатся вспомогательные классы, предназначенные для навигации (рисунок 4.3). Так, например, для навигации внутри основного фрейма был создан интерфейс INavigationService, в котором есть вспомогательный метод для навигации внутри данного окна.

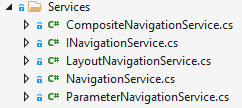


Рисунок 4.3 – Директорий Services

Директория Entities содержит все классы сущностей программы, описание которых представлено в разделе 4.1.

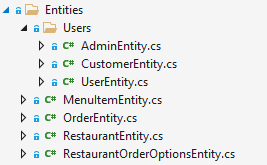


Рисунок 4.4 – Директорий Entities

В приложении используется паттерн Command который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. В WPF команды представлены интерфейсом ICommand. В приложении он представлен в виде DelegateCommand, а также ViewModelBase (Рисунок 4.5). Классы реализует два метода:

* CanExecute: определяет, может ли команда выполняться
* Execute: собственно, выполняет логику команды

А также реализует событие CanExecuteChanged вызывается при изменении условий, указывающий, может ли команда выполняться.

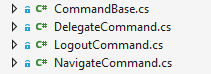


Рисунок 4.5 − Классы, реализующие ICommand

В директории Components расположены пользовательские элементы управления (Рисунок 4.6).

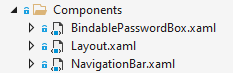


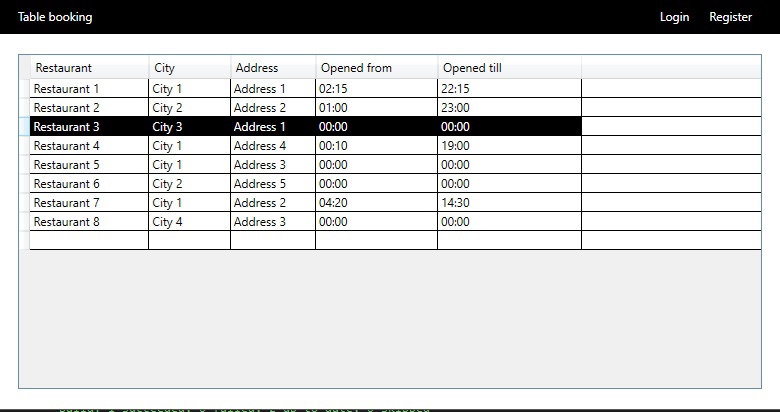
Рисунок 4.6 − Директорий Components

Тестирование, проверка работоспособности и анализ полученных данных

# Руководство по использованию

После запуска приложения в первую очередь открывается главное окно, на котором показана краткая информация о магазине.

Рисунок 6.1 – Главное окно



После того, пользователь может войти в аккаунт(Рисунок 6.2), зарегистрироваться(Рисунок 6.3), выбрав соответствующие пункты в меню справа. Также он может просмотреть каталог товаров по категориям, выбрав ее в меню справа во вкладке Products(Рисунок 6.4). Пользователь может узнать о верификации продуктов и политике возврата выбрав пункт Support в меню справа(Рисунок 6.5). Есть возможность узнать о расположении магазинов в разных странах выбрав пункт меню Shops(Рисунок 6.6). Получить контактную информацию можно выбрав пункт меню Contacts(Рисунок 6.7). Узнать о сотрудничестве и получить краткую информацию о магазине можно во вкладке Info(Рисунок 6.8).

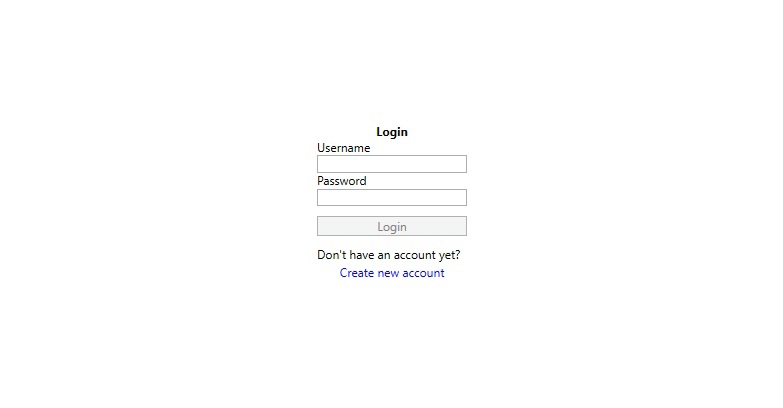


Рисунок 6.2 – Окно авторизации

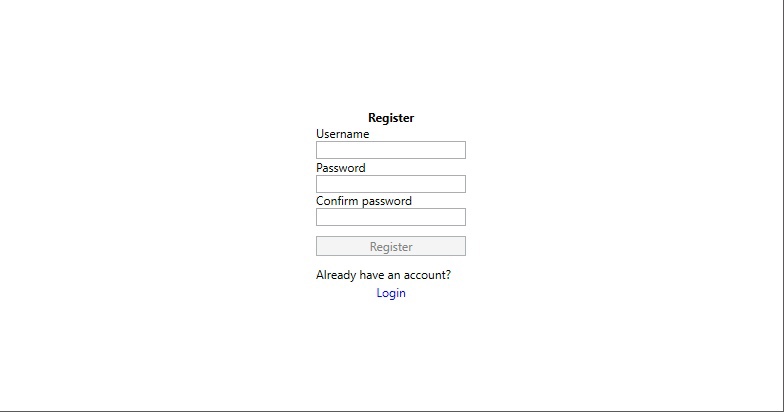


Рисунок 6.3 – Окно регистрации

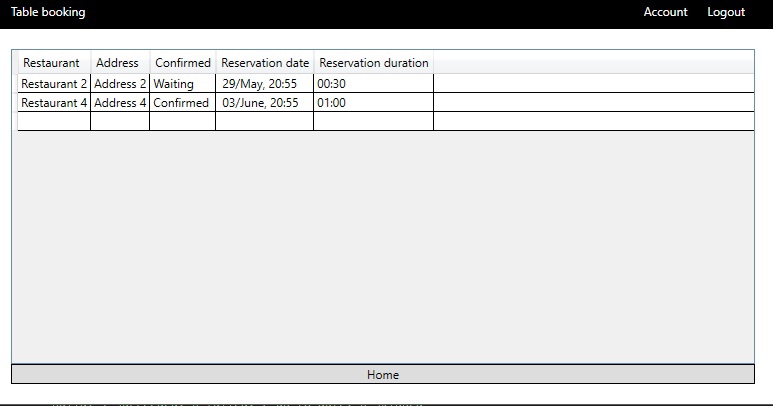


Рисунок 6.4 – Вкладка Account

# ЗАКЛЮЧЕНИЕ

В данном курсовом проекте было разработано программное средство «Бронирование столика» при помощи языка программирования C#, API-интерфейса Windows Presentation Foundation, технологии Entity Framework и базы данных Microsoft SQL Server. При выполнении курсового проекта использовались принципы и приемы ООП.

Разработанное программное средство предоставляет пользователю следующие функциональные возможности:

* Выполнение регистрации и авторизации.
* Создание заказов.
* Просмотр заказов.
* Добавление товаров в корзину.
* Удаление товаров из корзины.
* Изменение личной информации.
* Просмотр различной информации о магазине

Администратор может всё то же и плюс ко всему:

* Добавление товаров.
* Поддержка работы с базой данных.

Проектирование осуществлялось по паттерну проектирования MVVM.

Для удобной работы с данными, был реализован паттерн Repository и UnitOfWork.

Приложение хорошо работает с базой данных. В ходе тестирования, не было выявлено нарушений в работе приложения с базой данных.

Приложение было успешно протестировано, что означает пригодность этого приложения для своей цели.

Для хранения исходного кода и удобства контроля версий проекта использовался крупнейший веб-сервис для совместной разработки – GitHub. Ссылка на проект: (https://github.com/hassty/TableBooking/tree/master).

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

[1]. Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2021. – 175 с.

[2]. Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с. [3]. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа: http://msdn.microsoft.com/library/

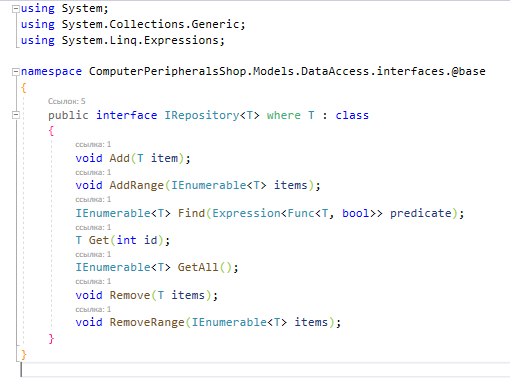
[4]. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com

[5]. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: https://professorweb.ru

[6]. StackOverflow [Электронный ресурс] / Режим доступа: <https://stackoverflow.com>

[7]. Хабр [Электронный ресурс] / Режим доступа: <https://habr.com>.

Приложение А



Приложение Б

