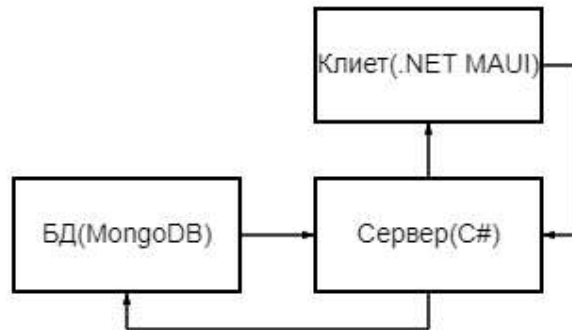


## Архитектура приложения:



## Функционал:

- Создание, редактирование и удаление заказов клиентов с различными атрибутами (имя, контакты, список позиций, общая стоимость заказа, дата заказа)
- Поиск и фильтрация клиентов по разным критериям в виде таблицы (по дате заказа, по контактным данным)
- Генерация статистики и отрисовка графиков по заказам клиентов (по доходности, по востребованности категории товаров, средний чек и общий доход за месяц)
- Авторизация в качестве кассира, бухгалтера, менеджера в программе (предварительная регистрация обязательна)
- Создание редактирование и удаление категорий (имя категорий)
- Создание редактирование и удаление позиций (имя позиций, стоимость позиции, принадлежность к категории)

## Используемые классы

- **Позиция:**
  - **Поля:**
    - Имя позиции (тип данных: string)
    - Стоимость позиции (тип данных: float)
    - Принадлежность к категории (тип данных: string)
    - ID позиции (тип данных: string)
  - **Тип класса:** Public
  - **Методы:**
    - Создание позиции
    - Редактирование позиции
    - Удаление позиций

- ID позиции (тип данных: string)
- Категория:
  - Поля:
    - Имя категории (тип данных: string)
    - ID позиции (тип данных: string)
  - Тип класса: Public
  - Методы:
    - Создание категорий
    - Редактирование категорий
    - Удаление категорий
- Заказ:
  - Поля:
    - Имя клиента (тип данных: string)
    - Контакты клиента (тип данных: string)
    - Список позиций заказа (тип данных: list)
    - Общая стоимость заказа (тип данных: float)
    - Дата заказа (тип данных: date)
    - ID позиции (тип данных: string)
  - Тип класса: Public
  - Методы:
    - Создание
    - Редактирование
    - Удаление заказов
- Пользователь:
  - Поля:
    - Логин (тип данных string)
    - Пароль (тип данных string)
    - Роль пользователя (кассир, бухгалтер или менеджер) (тип данных string)
    - ID позиции (тип данных: string)
  - Тип класса: Не указан
  - Методы:
    - Авторизация пользователя в программе.
    - Регистрация пользователя в программе.

- **DataStorage:**
  - **Поля:**
    - Объект подключения к базе данных (тип данных: object)
    - Объект базы данных по умолчанию (тип данных: object)
  - Тип класса: Public
  - Методы:
    - -Чтение данных из коллекции по заданному запросу
    - -Запись данных в коллекцию
    - -Обновление данных в коллекции по заданному запросу
    - -Удаление данных из коллекции по заданному запросу

### **Модели данных в БД:**

- Позиция
- Категория
- Заказ
- Пользователь

### **Система авторизации:**

Один из способов сделать авторизацию более безопасной - это хранить пароль в виде хеша, а не в открытом виде. Хеш - это результат применения специальной функции к паролю, который невозможно восстановить обратно. Таким образом, даже если злоумышленник получит доступ к базе данных с хешами паролей, он не сможет узнать исходные пароли.

Однако, простое хеширование паролей может быть недостаточно, так как существуют специальные словари, в которых хранятся хеши часто используемых паролей. Поэтому, для увеличения безопасности, рекомендуется использовать так называемую соль - случайное значение, которое добавляется к паролю перед хешированием. Соль делает хеш уникальным для каждого пароля, даже если они совпадают. Соль нужно хранить вместе с хешем пароля, чтобы можно было проверить введенный пароль при авторизации.

Для реализации хеширования паролей с солью на C# можно использовать следующий подход:

- Импортировать пространство имен `System`, `System.Security.Cryptography` и `System.Text` для работы с хеш-функциями, кодировками и строками.
- Создание метода `GenerateSalt`, который будет генерировать случайную соль заданной длины в виде массива байтов. Для этого можно использовать класс `RNGCryptoServiceProvider` из пространства имен `System.Security.Cryptography`, который предоставляет криптографически безопасный генератор случайных чисел.
- Создание метода `HashPassword`, который будет принимать пароль в виде строки и соль в виде массива байтов и возвращать хеш пароля в виде строки. Для этого вы можете использовать класс `SHA256Managed` из пространства имен `System.Security.Cryptography`, который предоставляет реализацию хеш-функции SHA-256. Вы также можете использовать класс `Encoding` из пространства имен `System.Text` для преобразования строк в массивы байтов и обратно.

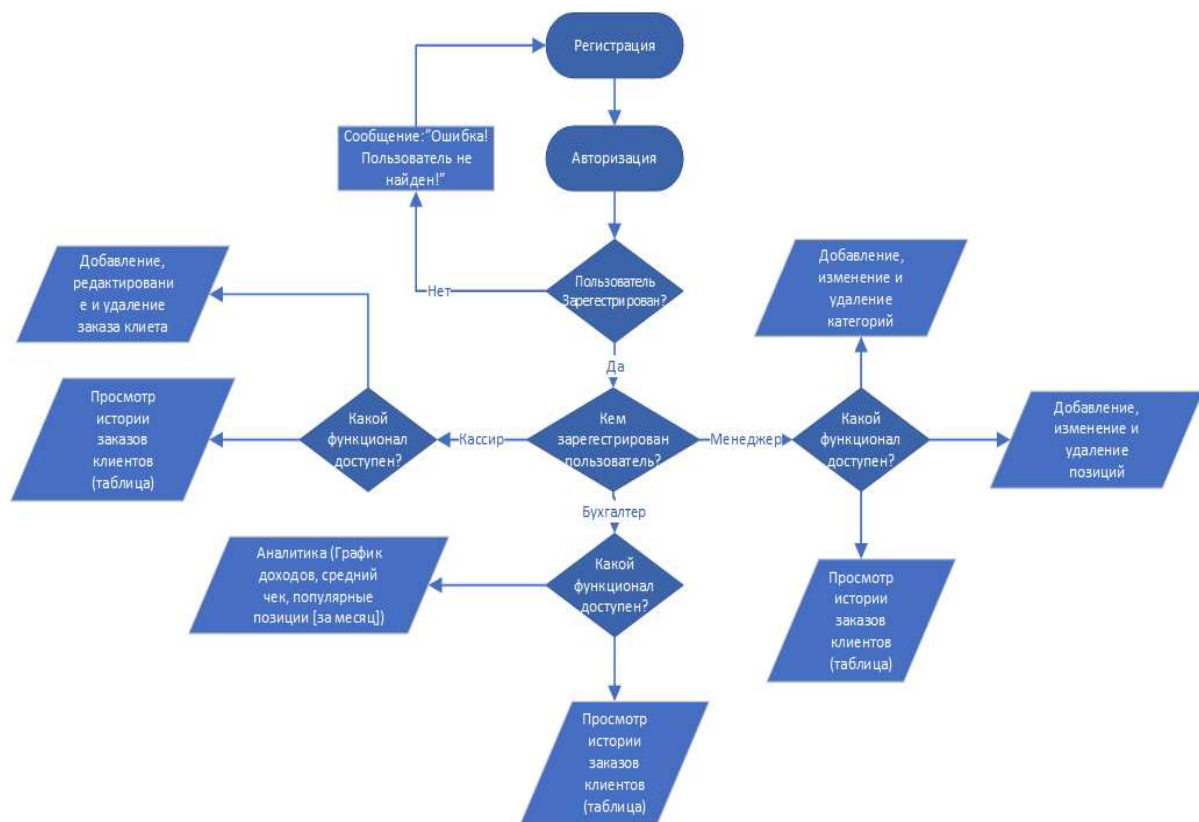
- Создание метода `ValidatePassword`, который будет принимать введенный пароль в виде строки, хеш пароля в виде строки и соль в виде массива байтов и возвращать `true`, если пароль верный, и `false`, если нет. Для этого можно использовать метод `HashPassword` для получения хеша введенного пароля с той же солью и сравнить его с хранящимся хешем пароля.

### Примерный код авторизации

```
// Метод для генерации случайной соли заданной длины
public static byte[] GenerateSalt(int length)
{
    // Создаем объект криптографически безопасного генератора случайных чисел
    using (var rng = new RNGCryptoServiceProvider())
    {
        // Создаем массив байтов заданной длины
        var salt = new byte[length];
        // Заполняем массив случайными байтами
        rng.GetBytes(salt);
        // Возвращаем массив байтов
        return salt;
    }
}

// Метод для хеширования пароля с солью
public static string HashPassword(string password, byte[] salt)
{
    // Создаем объект хеш-функции SHA-256
    using (var sha = new SHA256Managed())
    {
        // Преобразуем пароль в массив байтов
        var passwordBytes = Encoding.UTF8.GetBytes(password);
        // Соединяем массивы байтов пароля и соли
        var passwordAndSaltBytes = new byte[passwordBytes.Length + salt.Length];
        Array.Copy(passwordBytes, 0, passwordAndSaltBytes, 0, passwordBytes.Length);
        Array.Copy(salt, 0, passwordAndSaltBytes, passwordBytes.Length, salt.Length);
        // Вычисляем хеш от соединенных массивов байтов
        var hashBytes = sha.ComputeHash(passwordAndSaltBytes);
        // Преобразуем хеш в строку в шестнадцатеричном формате
        var hashString = BitConverter.ToString(hashBytes).Replace("-", "");
    }
}
```





## Концепт-дизайн:

The image shows a concept design for a login page. At the top, there is a dark gray header bar. On the left side of the header, the word "Newborn" is written in white. On the right side, the words "Вход" (Login) and "Регистрация" (Registration) are written in white. Below the header, centered on the page, is a white rectangular box with a thin gray border. Inside this box, the text "Войти в систему" (Login to the system) is at the top. Below it are two input fields: the first is labeled "Email:" and the second is labeled "Пароль:" (Password:). At the bottom of the box is a green button with the white text "войти" (login).

*Рисунок 5 – Страница входа*

The image shows a concept design for a registration page. It has the same dark gray header bar as the login page, with "Newborn" on the left and "Вход" (Login) and "Регистрация" (Registration) on the right. Below the header, centered on the page, is a white rectangular box with a thin gray border. Inside this box, the text "Создать аккаунт" (Create account) is at the top. Below it are two input fields: the first is labeled "Email:" and the second is labeled "Пароль:" (Password:). At the bottom of the box is a green button with the white text "создать" (create).

*Рисунок 6 – Страница регистрации*





Рисунок 7 – Главная страница



Рисунок 8 – Страница аналитики

Newborn	История заказов				
Обзор					
Аналитика					
История					
Добавить заказ					
Ассортимент					
Выйти					
	№	Дата	Время	Сумма	
	1	21.12.2000	14:21	12 211 руб.	
	1	21.12.2000	14:21	12 211 руб.	
	1	21.12.2000	14:21	12 211 руб.	
	1	21.12.2000	14:21	12 211 руб.	
	1	21.12.2000	14:21	12 211 руб.	
	1	21.12.2000	14:21	12 211 руб.	
	1	21.12.2000	14:21	12 211 руб.	
	1	21.12.2000	14:21	12 211 руб.	
	1	21.12.2000	14:21	12 211 руб.	
	1	21.12.2000	14:21	12 211 руб.	
	1	21.12.2000	14:21	12 211 руб.	

Рисунок 9 – Страница истории заказов




Newborn	Категории > Добавить категорию		
Обзор	<div>Имя:</div> <div>   </div>		
Аналитика	<div>Позиции:</div> <div> <div>Бамбуковый торт 20 руб.</div> <div>Добавить позицию</div> </div>		
История			
Добавить заказ			
Ассортимент			
Выйти			

Рисунок 10 – Страница добавления позиций

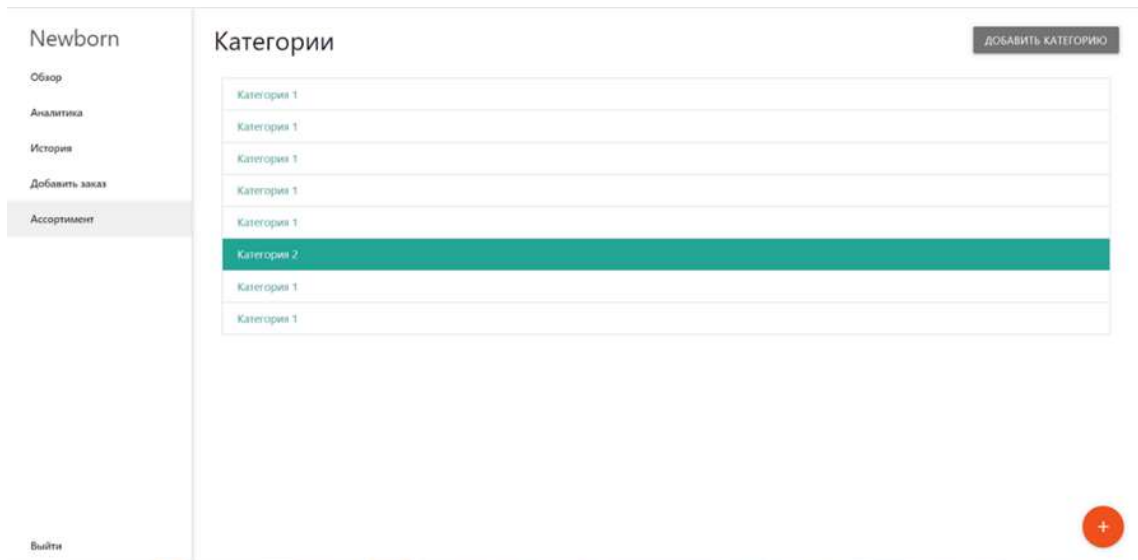


Рисунок 11 – Страница добавления категорий

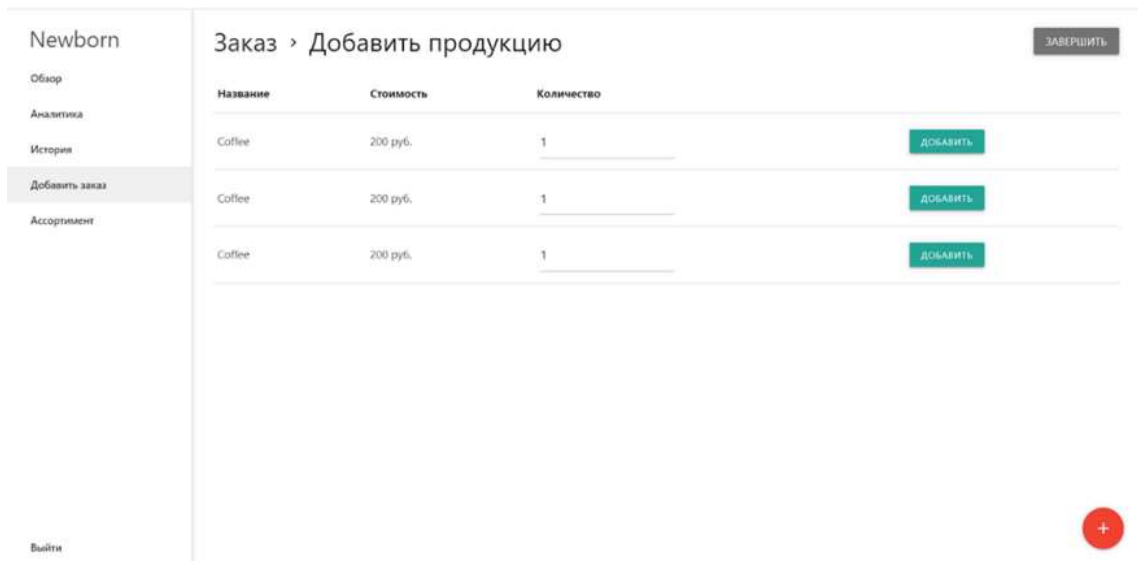


Рисунок 12 – Страница формирования заказа

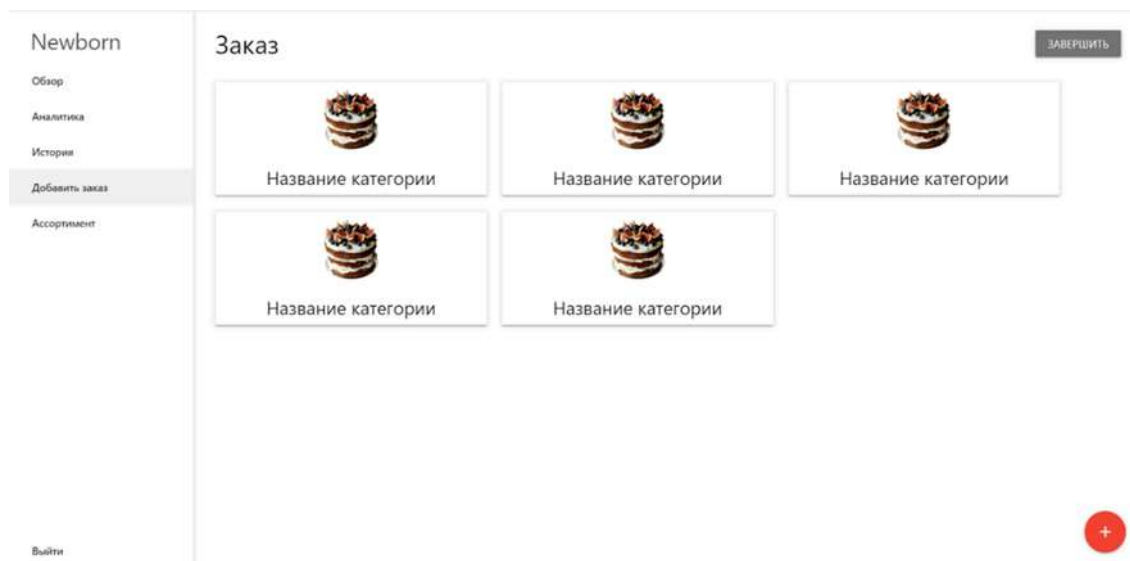


Рисунок 13 – Страница вида заказа