

Министерство образования и молодежной политики
Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Специальность 09.02.07: «Информационные системы и
программирование»

Разработка «автоматизированной системы для кафе»

Пояснительная записка

к курсовому проекту

КП-ПР-41-21-2026-ПЗ

Разработал:

Студент гр. ПР-41

_____ / М.К. Шмелев

Руководитель

_____ / Д.О. Гариев

2026

Министерство образования и молодежной политики
Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Специальность 09.02.07: «Информационные системы и
программирование»

Разработка «автоматизированной системы для кафе»

Курсовой проект

КП-ПР-41-21-2026-ПЗ

2026

Оглавление

Введение.....	4
1 Постановка задачи	5
1.1 Описание предметной области	5
1.2 Требования к программному продукту	7
2.Проектирование.....	10
2.1 Проектирование логической структуры	10
2.2 Проектирование физической структуры	14
3 Разработка и тестирование.....	23
Заключение	31
Список использованной литературы.....	32
Приложения	33
Приложение А. Исходный код.	34
Приложение Б. UI сервера.....	35
Приложение В. Интерфейс программы	36
Приложение Г. База данных.....	37

Инв. № подп	Подп. и дата	Инв. № дубл	Взам. инв. №	Подп. и дата
Лит	Изм.	№ докум.	Подп.	Дата
Разраб.	Шмелев М.К.			
Проф.	Гареев Д.О.			
Н. контр.	Гареев Д.О.			
Утв.	Гареев Д.О.			

КП-ПР-41-21-2026-ПЗ

Разработка
автоматизированной системы
для кафе

Лит	Лист	Листов
	3	35

Введение

Информационные технологии сегодня являются ключевым фактором в нашей жизни и на каждом предприятии. Сфера услуг общественного питания также нуждается в автоматизации рутинных процессов, связанных с заказами. Компании ищут лучшие автоматизированные системы из-за высокой конкуренции за сотрудников и клиентов.

Актуальность темы подтверждается тем, что каждый день создаются новые организации, которым требуются такие системы. На рынке много предложений: есть как дорогие решения, так и попроще для менее популярных заведений. Всё зависит от требований заказчика.

Целью курсового проекта является разработка автоматизированной системы, состоящей из серверной части и клиентского приложения, для автоматизации обработки заказов в кафе.

Задачи, которые необходимо выполнить для достижения поставленной цели:

1. Выполнить анализ предметной области.
2. Провести обзор существующих аналогов и показать преимущества собственной разработки.
3. Реализовать базу данных и логическую структуру приложения.
4. Реализовать серверную часть приложения.
5. Реализовать клиентскую часть приложения.
6. Разработать пользовательский интерфейс.
7. Осуществить тестирование разработанных модулей.

Инф. № подл	Подл. и дата	Инф. № подл.	Взам. инф. №	Подл. и дата
-------------	--------------	--------------	--------------	--------------

Ли	Изм.	№ докум.	Подп.	Дата	Лист 4
КП-ПР-41-21-2026-П3					

1 Постановка задачи

1.1 Описание предметной области

Предметной областью является управление предприятием общественного питания.

В настоящее время, если в заведении нет автоматизированной системы, процесс выглядит так: официант принимает заказ, записывает его в блокнот, затем передает данные кассиру и повару.

Этот подход очень неудобен, если в зале более пятидесяти столов и все они заняты, а официантов всего один или два человека, они физически не успеют качественно обслужить всех гостей. При высокой нагрузке смысл ручного метода пропадает.

Отсутствие контроля провоцирует целый ряд ошибок – блюда часто выносятся не на те столы, что вызывает раздражение гостей и конфликт с персоналом.

Отдельная проблема - нарушение курсов подачи. Официант может вынести десерт раньше основного блюда, что испортит впечатление от вкуса и от вечера. Кроме того, важна одновременная подача: если гости заказали два блюда, недопустимо принести одно блюдо, заставив второго гостя ждать и смотреть, как первый ест. Диаграмма, изложенная на *рисунке 1.* отображает данных подход.

Также существует риск кражи: сотрудники могут обманывать руководство, присваивая разницу между чеком и полученными деньгами. В разрабатываемой системе происходит учет денежных средств, что обеспечивает отчетность. Из этого можно сделать вывод: в XXI веке ни одно предприятие общественного питания не сможет выдержать конкуренцию, и работать без автоматизированной системы.

Инф. № подл	Подл. и дата	Инф. № дубл.	Взам. инф. №	Подл. и дата
-------------	--------------	--------------	--------------	--------------

Ли	Изм.	№ докум.	Подп.	Дата	Лист 5
					КП-ПР-41-21-2026-ПЗ

Диаграмма показывает протекающий процесс актеров без автоматизированной системы, гость заказывает позицию, оплачивает. Официант записывает в блокнот заказ, передает данные на кассу. Касса передает официанту чек на оплату. Повар готовит заказ и отдает официанту, демонстрация на рисунке 1.

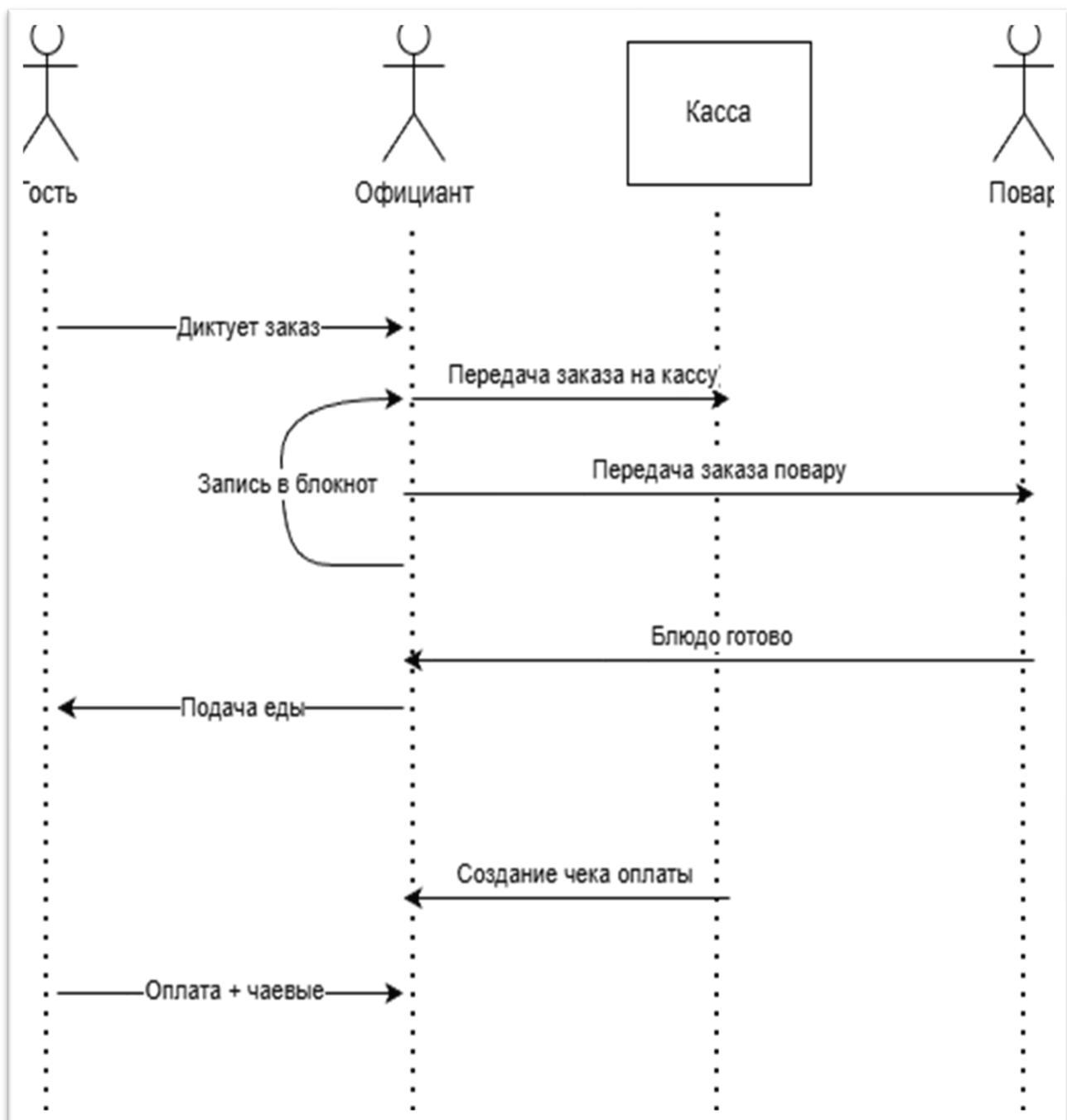


Рисунок 1 – Диаграмма последовательности.

Инд. № подл	Подл. и дата	Инд. № дубл.	Взам. инф. №	Подл. и дата

1.2 Требования к программному продукту

Проектируемая система должна выполнять следующие задачи:

1. Авторизация и доступ к функционалу по ролям: Система должна авторизировать пользователя (Администратор, официант, повар).
2. Управление меню: Администратор имеет возможность редактировать, добавлять, удалять, позиций меню, смена стоимости, категории и описания.
3. Приём заказов: Официант через приложение создает заказ, привязывает его к столу и добавляет блюда и так же может менять заказ.
4. Статусы заказов: Администратор или же официант могут его изменять, но если заказ попал в систему автоматический: то будет иметь статус выполняется иначе он будет иметь статус завершен.
5. Вывод всех позиций: как для официанта, так и для повара вывод позиций, которые нужно отдавать.
6. Оплата: Автоматический расчет стоимости.

Сравнение программного обеспечения с его аналогами:

На **таблице 1** - сравнение лидеров рынка с разрабатываемой системой проведено сравнение аналогов.

1. **ПКО** – предоставление комплексной автоматизированной системы, решение для всех типов заведений. Охватывает все аспекты бизнеса – от кассы до склада. Недостаток в том, что имеет высокую стоимость для малого бизнеса от 10 – 32 тыс. руб.

2. **R-Keeper** – для сложных и очень крупных сетей. Архитектура и зависимость к оборудованию, плюсом нужен обученный человек чтобы постоянно следить за системой. Огромное количество разных модулей продаж. Дорогая система – для покупки полной версии приложения как от 60 – 150 тыс. руб.

Разрабатываемая система – Менее нагруженная, бесплатная, не требует мощного ПО, можно обойтись без pos – терминалов. Запутаться в функциях не получится, есть базовые и готовые решения для обработки заказов. Это экономит время обучение персонала, сложность минимальная. Низкие системные требования

Инф. № подл	Подл. и дата	Инф. № дубл.	Взам. инф. №	Подл. и дата

Ли	Изм.	№ докум.	Подп.	Дата	Лист 7
					KП-ПР-41-21-2026-ПЗ

– этот продукт можно развернуть на любом офисном пк. И заказчик получает полный контроль над сервером, что в будущем играет большую роль в расширение и масштабировании.

Таблица 1 - сравнение лидеров рынка с разрабатываемой системой

Продукты:	ПКО	R-Keeper	Разрабатываемая система
аудитория	Все типы заведений	крупные ресторанные сети.	Малый и средний бизнес.
Стоимость внедрения	Средняя варьируется от 10.000 – 32.000 тыс.руб. еще ежемесячная подписка	Высокая (60 000 – 150 000 руб. за полную версию).	Бесплатная (затраты только на разработку и внедрение).
Оборудование	Требуются POS-терминалы и совместимое оборудование.	Высокие требования к ПК и POS - Терминалам	Низкие требования. Работает на любом офисном ПК,
Сложность обучение	Охватывает все аспекты (от кассы до склада), требует обучения.	Высокая сложность. Очень много модулей ведения учета.	Минимальная сложность. Быстрое обучение персонала.
Обслуживание	Требует техподдержки.	Требуется обученный специалист.	Простое. Не требует обученного специалиста.
Функциональность	Комплексная (Касса, Склад, Кухня, Финансы, Персонал).	Мощная система разными вариантами продаж.	Готовое решения для обработки заказов, ничего лишнего.

Чтобы все задачи были реализованы правильно к проекту предъявляются следующее. *Функциональные требования:* операции над пользователем, аутентификация пользователей. Система требует ввод логина и пароля.

Инф. № подп	Подп. и дата

Регистрация пользователя: требует ФИО, логин, пароль. Выбор роли, но только через права администратора. Так же вывод авторизованных пользователей.

Операции над заказом: создание заказа, система требует выбрать позицию из списка меню, и привязывает заказ к определенному столу. Редактирование заказа: возможность удаление и добавления позиций в заказ, количество порций. Автоматический расчет стоимости: система мгновенно рассчитывает стоимость заказа при любом изменении. Отправка повару: передача данных на кухню, вывод позиций для готовки. Закрытие: когда гости готовы оплатить официант закрывает заказ и предлагает оплатить счет, смена статуса заказа “Оплачено”, “Готов” смена зависит от действий с заказом.

Управление меню: Просмотр и редактирование позиций: возможность добавления и удаления. Управление статусом: показывает в стопе-листе ли блюдо. Ценообразование: изменение стоимости блюд.

Кухня: подробный вывод заказов и детализация позиций (номер стола, название блюда, количество), контроль готовности.

Инф. № подл	Подл. и дата	Инф. № дубл.	Взам. инф. №	Подл. и дата
Ли	Изм.	№ докум.	Подп.	Дата

Лист 9

КП-ПР-41-21-2026-ПЗ

2.Проектирование

Проектирование важная часть в разработке программного обеспечения (ПО). Начнем с того какие требования для проекта автоматизированной системы для кафе:

Стек технологий сервер:

1. Бекэнд: ASP.NET Core API, .NET 9. C#, Слоистая архитектура.
2. Представление (UI): WPF (приложение), .NET 9.
3. База данных: SQL Server, и объектно-реляционный преобразователь (ORM) Entity Framework.
4. Безопасность: JWT (JSON Web Token), BCrypt – Шифрование пароля.
5. Документация API: Swagger.

Стек технологий клиент:

1. Бекэнд: .NET 9, C#.
2. Представление (UI): WPF (приложение), .NET 9
3. Паттерн: Code-Behind (Event-driven)"
4. Взаимодействие с API: HttpClient, Text.Json.
5. Дизайн: MaterialDesignInXamlToolkit

2.1 Проектирование логической структуры

Логическая структура построена на основе модели баз данных. Главная задача была сделать правильную структуру, следуя современным практикам, чтобы обеспечить модульность и возможность масштабирования. Для понимания представлена ER Диаграмма базы данных, изображенная на рисунке 2. Которая служит основой для проектирования базовых классов сущностей. Она показывает связи сущностей, ключи.

Инф. № подп	Подп. и дата	Инф. № дубл.	Взам. инф. №	Подп. и дата	Подп. и дата

Ли	Изм.	№ докум.	Подп.	Дата	Лист 10
КП-ПР-41-21-2026-ПЗ					

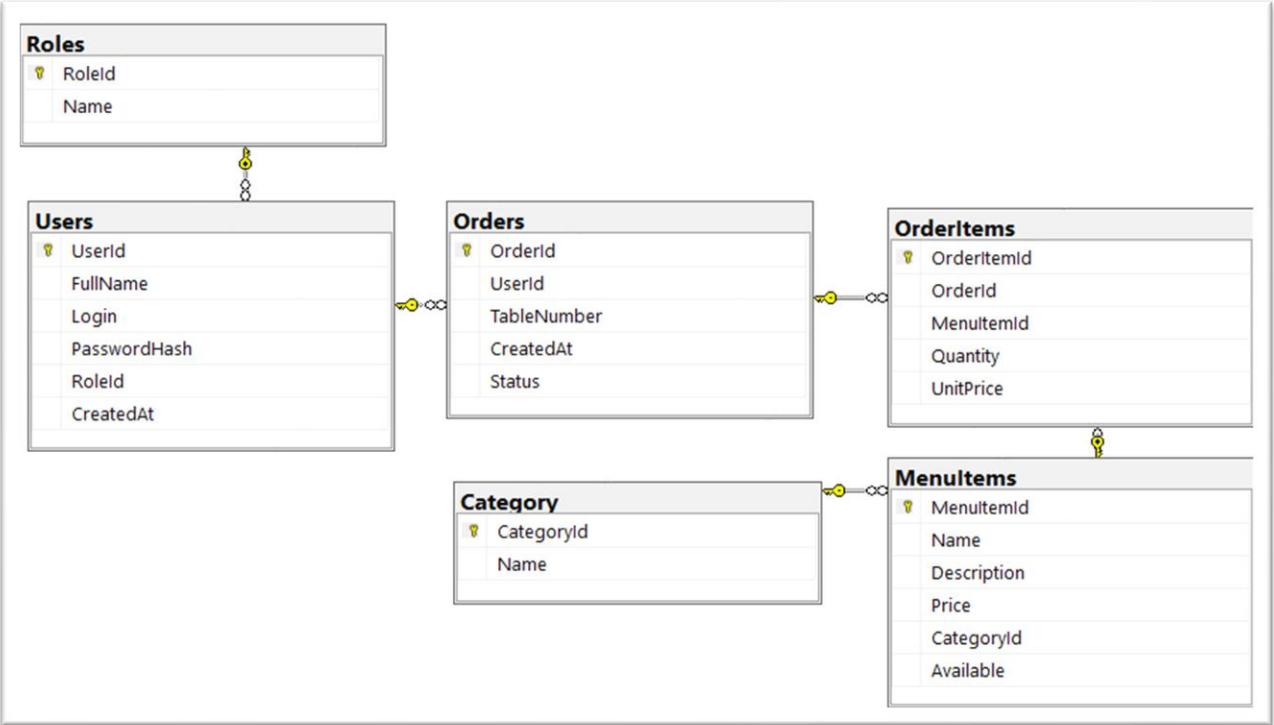


Рисунок 2 - ER Диаграмма

Основными сущностями являются: User, Order, MenuItem, OrderItem, вот что они из себя представляют:

User – ключ таблицы UserId. Остальные поля: Login, PasswordHash, FullName. Имеет связь многие к одному с сущностью Role.

Order (Заказ) – главная сущность, отражающая обслуживания клиента. Ключ OrderId. Остальные поля: TableNumber, Status, CreatedAt, UserId. Связи: Связана с User (официант, открывший заказ) и содержит список OrderItems.

MenuItem - справочник блюд и напитков. Ключ MenuItemId. Остальные поля: Name, Description, Price, Available, CategoryId. Связи: относится к одной Category.

OrderItems – позиция заказа. Ключ OrderItemId. Остальные поля: OrderId, MenuItemId, Quantity, UnitPrice.

Category – категории блюд. Ключ CategoryId, остальное Name.

UML – Диаграммы: Диаграмма последовательности показывает, как протекает бизнес-процесс во время его работы, как работают сотрудники, система, представлено на рисунке 3. На диаграмме изображена последовательность действий при выборе автоматизированной системы для обработки заказов. Клиент

Инф. № подп	Подп. и дата	Инф. № дубл.	Взам. инф. №	Подп. и дата	Подп. и дата

Ли	Изм.	№ докум.	Подп.	Дата

заказывает у официанта позиции, корректирует заказы, производит оплату. Официант обрабатывает заказ и вбивает его в систему, Программа WPF это связь между персоналом и сервером, то, что требует персонал то и отправит приложение на сервер. Сервер API здесь основная логика обработки заказов. Повару предоставлен вывод заказов.

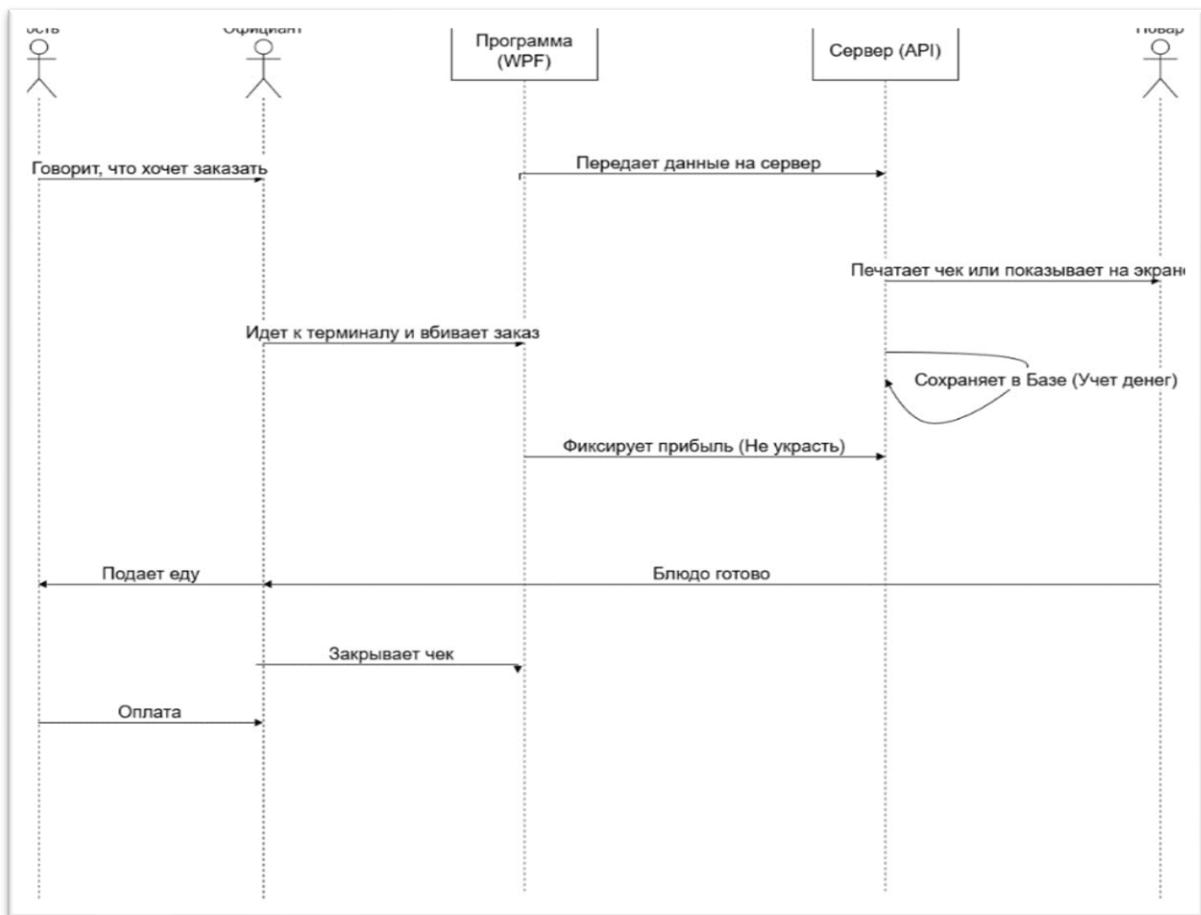


Рисунок 3 - Диаграмма последовательности с использованием системы

Процесс взаимодействия можно описать таким: Официант принимает заказ у клиента на стойке с ПОС-терминалом или же блокнотом, подходит к стойке, заводит заказ в систему в клиент приложении, дальше данные валидируются и отправляются по API на обработку на сервер с ними происходит прописанная логика, различные проверки и сервер возвращает ответ, дальше заказ переходит на кухню к поварам, когда они приготовят официант отмечает в клиент приложении что заказ готов, заказ уходит на сервер и отмечается как готов и официант его приносит. Дополнительно система обеспечивает синхронизацию статусов заказов между клиентским приложением и серверной частью в реальном времени.

Инд. № подп	Подп. и дата	Инд. № дубл.	Взам. инф. №	Подп. и дата
-------------	--------------	--------------	--------------	--------------

На рисунке 4 изображена диаграмма вариантов использования (прецедентов).

Диаграмма показывает актеров, которые оперируют с автоматизированной системой. Каждый актер делает свое дело, официант и повар вместе могут смотреть товары, менеджер управляет меню, учетными записями. Это диаграмма показывает процесс взаимодействия прецедентов.

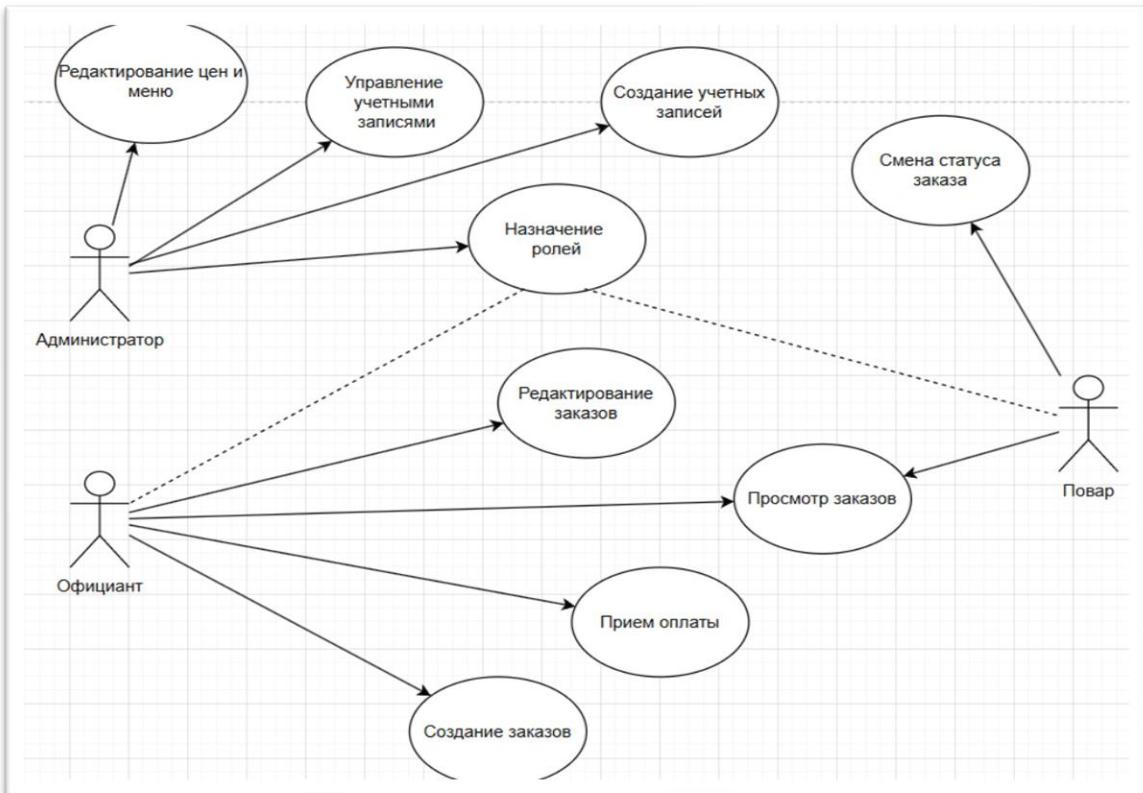


Рисунок 4 - Диаграмма прецедентов

2.2 Проектирование физической структуры

Физическая структура продукта реализована в соответствии с паттерном архитектуры N-Layer (Многослойная архитектура) на платформе .NET 9. Проект будет организован в виде решения CafeSolution. Файловая структура проекта выглядит следующим образом.

1. Controllers: (OrdersController.cs, MenuController.cs, UserController.cs) – слой представления, отвечает за прием HTTP запросов от клиента (WPF), валидация данных на входе, отвечает и принимает данные в формате Json.

Инф. № подп	Подп. и дата	Инф. № дубл.	Взам. инф. №	Подп. и дата

2. Services: (OrderService.cs, UserService.cs) – Слой бизнес логики. Здесь происходят вычисления заказов, управление пользователями, управление транзакциями.

3. Repositories - Отдельный слой, который работает с базой данных не знает о сервисе и контроллере. Предоставляет базовые методы CRUD с моделями.

4. Models - Модели, которые на прямую связаны с таблицами базы данных.

DTOs - Объекты для передачи данных между клиентом и сервером. Разделены по папкам (MenuItems, OrderItems, Orders, Roles, Users. Data (CafeDbContext) – Контекст базы данных, отвечает за соединение с СУБД.

В качестве выбранной базы данных будет использоваться MS SQL и будет управляется через SQL SERVER. Взаимодействие с бд будет использована библиотеке EF-Core (Entity Framework). Каждая модель из папки Models = таблица базы данных с названием, полями, и типами данных. Миграции бд в папке Migrations.

Чтобы понимать, как работает система, а именно паттерн, ниже представлена схема на рисунке 5. На изображение файловое древо проекта показано какой паттерн проектирования использован и в какой последовательности взаимодействует каждый модуль паттерна.

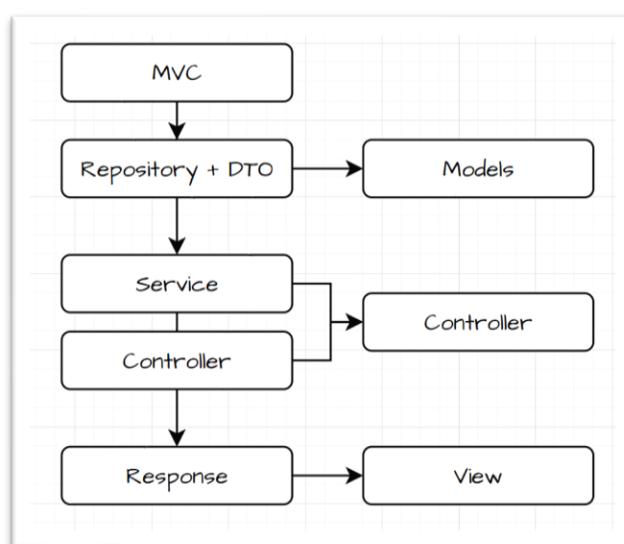


Рисунок 5 - Файловое древо проекта

Инд. № подп	Подп. и дата

Для документирования API – конечных точек для клиента будет использоваться Swagger. У HTTP есть свои методы: GET – получить, POST – создать, PUT – обновить, DELETE – удалить, PATCH – частичное обновление. Ниже представлено документирование конечных точек API в swagger.

Menu – здесь конечные точки категории menu, а именно получение (GetMenu), добавление (Add), обновление (Update), удаление (DeleteItem), получение по id (MenuItemId). Права администратора требуются: Add, Update, DeleteItem, MenuItemId. все это представлено на рисунке 6.

The screenshot shows the Swagger UI interface for the 'Menu' API. On the left, there is a vertical sidebar with columns labeled: 'Инф. № подл', 'Подл. и дата', 'Инф. № дубл', 'Взам. инф. №', and 'Подл. и данные'. The main area displays the following endpoints:

- GET /api/Menu/GetMenu**: Status bar is blue.
- POST /api/Menu/Add**: Status bar is green.
- PUT /api/Menu/{id} (Update)**: Status bar is orange.
- DELETE /api/Menu/{id} (DeleteItem)**: Status bar is red.
- GET /api/Menu/{id} (MenuItemId)**: Status bar is blue.

Each endpoint row includes a lock icon and a dropdown arrow on the right side.

Рисунок 6 - Конечные точки Menu

Для запросов по адресу Menu используется DTO: MenuItemResponseDto – возвращает ответ после запроса к конечной точке, сам класс выбирает что возвращать в ответ вместо полного готового объекта. CreateMenuItemDto – когда идет обращение к конечной точке то данный класс реализует макет данных который должен прийти. UpdateMenuItemDto – когда выбран объект, который нужно обновить используется этот класс он не позволяет обновлять все поля, а только которые в нем прописаны.

Orders - здесь конечные точки категории orders, а именно создание заказа, получение всех заказов, получение по id, добавить позиции в заказ, удалить заказ, получить заказ привязанный к user, получить статус заказа, обновить статус заказа, и удалить позицию из заказа, все это представлено на рисунке 7.

Инф. № подл	Подл. и дата	Инф. № дубл	Взам. инф. №	Подл. и данные

Orders	
POST	/api/Orders/CreateOrder
GET	/api/Orders/GetAll
GET	/api/Orders/{id}/GetOrderById
POST	/api/Orders/{id}/AddItemsToOrder
DELETE	/api/Orders/{id}/DeleteOrder
GET	/api/Orders/{userId}/userOrder
GET	/api/Orders/{status}/status
PATCH	/api/Orders/{id}/statusUpdate
DELETE	/api/Orders/{id}/deleteItem

Рисунок 7 - Конечные точки

Для запросов по адресу Orders используется DTO OrderResponseDto – возвращает полный ответ сервера после создания или запроса заказа, содержит статус, дату создания и вложенный список блюд, скрывая служебную логику. CreateOrderDto – реализует макет данных при создании нового заказа, требует указать сотрудника, номер стола и список позиций. OrderItemDto – используется как вложенный объект, описывает конкретное блюдо и его количество в составе заказа.

User - здесь конечные точки для управления над пользователем: войти в приложение (Login), получить пользователей (User), регистрация пользователя (Register), удалить пользователя (DeleteUser), получить пользователя по Id (GetUserId), и обновить данные пользователя (UpdateUser). Данные конечные точки используются для управления учетными записями пользователей системы, а именно изменять, удалять, сохранять, добавлять. Права администратора требуются: (api/User), DeleteUser, GetUserId, изображено на рисунке 8.

Инф. № подп	Подп. и дата	Инф. № дубл.	Взам. инф. №	Подп. и дата

Ли	Изм.	№ докум.	Подп.	Дата

User	
POST	/api/User/Login
GET	/api/User
POST	/api/User/Register
DELETE	/api/User/DeleteUser
GET	/api/User/{id}/GetUserId
PATCH	/api/User/{id}/UpdateUser

Рисунок 8 – Конечные точки User

Клиент будет использовать программу WPF (Windows Presentation Foundation) – это фреймворк для создания, современных десктопных приложений для Windows. Его работа будет заключаться в следующем: UI – главное в этом приложении, если будет удобен дизайн то и будет проще делать заказы а значит больше прибыли это все показывает как важно делать информативную и понятную разметку xaml для WPF, создание заказов в клиенте и их отправка на сервер по конечной точке API, а также принятие ответов от сервера. Его суть заключается быть почтальоном среди сервера и клиента. Для безопасности я буду использовать JWT токен и BCrypt для шифрования пароля и вот почему: Пароли очень важная атрибуция объекта пользователя если бд взломают или сделают SQL инъекцию, то узнают пароли и испортят данные.

Сравнение выбора средств: BCrypt.Net-Next – безопасное хранение паролей C# (и .NET), это библиотека используется только для паролей она не шифрует отдельные данные, она медленная по сравнению с аналогами, но проста в использовании для системы, использует криптографическую хеш-функцию.

JWT - это безопасность, не нужно будет много раз входить в приложение если что то случится например перебои с интернетом, чтобы повар и официант не смог менять цены и позиции меню, в токен вшита роль входящего сотрудника. В сравнении с cookie выбранный JWT имеет несколько преимуществ: сервер не

Инф. № подп	Подп. и дата
Инф. № докл.	Взам. инф. №
Подп. и дата	Подп. и дата
Лист	

Ли	Изм.	№ докум.	Подп.	Дата

хранит состояние сессии что снижает загрузку, проверка токена происходит на сервере, не надо будет обращаться к бд очень часто. Токен использует криптографическую подпись.

.NET 9 и asp.net core, потому что они идеально подходят к созданию API, фреймворк очень удобен тем, что выполняет много автоматических задач, например он маршрутизирует запросы, какой контроллер обрабатывает HTTP запрос (GET, POST, PUT, DELETE). Преобразует объекты в форматы JSON для отправки клиенту и наоборот десериализует. Управление жизненным циклом с помощью middleware – конвейером. И предоставляет готовые ресурсы для создания точек.

MS SQL – реляционная БД, хорошо работает с .NET, не сложная, можно управлять через интерфейс в программе SQL Server, через визуал проще взаимодействовать с данными и управлять ими, так же возможно подключить к Visual Studio и управлять ее данными через IDE. Паттерн я выбрал чтобы каждый слой не знал о друг друге, система не маленькая и так проще будет тестировать каждый блок и чинить в случае поломки.

Аналоги: Windows Presentation Foundation указаны в списке ниже.

1. WPF – Мощный фреймворк для десктопных приложений, работает с .NET на чем и работает автоматизированная система, оптимизирована под API. По сравнению с аналогами он гарантированно хороший выбор.

2. UWP - Universal Windows Platform довольно устаревшая система, на замену ему пришел WinUI 3, он менее гибкий по сравнению с WPF или WinUI, также не совместим с версией < windows 10.

3. Avalonia UI – мощный фреймворк для создания десктопных приложений на windows похож на WPF но он является сторонним решением (не от Microsoft) и ориентирован в первую очередь на кроссплатформенность (Linux, macOS).

Таким образом, выбранный стек технологий (ASP.NET Core Web API + MS SQL Server + WPF) полностью соответствует техническим требованиям проекта, обеспечивает необходимый уровень безопасности (JWT) и позволяет реализовать автоматизированную систему для предприятия общественного питания.

Инф. № подп	Подп. и дата
Инф. № дубл.	Взам. инф. №
Подп. и дата	Подп. и дата

Ли	Изм.	№ докум.	Подп.	Дата	Лист 18
КП-ПР-41-21-2026-ПЗ					

Макеты приложения:

- страница входа в приложение показана на рисунке 9, тут обычный вход в приложение с логином и паролем.
- страница меню указана на рисунке 10, здесь мы видим меню, заказы, категории.
- страница администратора указан на рисунке 11, Он управляет профилями пользователей.
- страница заказа указана на рисунке 12, редактирование, удаление, добавление заказа, полное управление, деление на разные счета.

Все страницы будут реализованы в приложении WPF автоматизированной системы. Представленный интерфейс является примерным дизайном каждого экрана приложения. Будет обеспечено синхронное отображение данных, поступающих с серверной части системы. Приложение будет использоваться как основной инструмент для управления автоматизированной системой кафе.

1. Вход в систему (Авторизация, ввод логина и пароля пользователя), на рисунке 9.

The image shows a login screen with a light gray background featuring a faint grid pattern. In the upper right corner, there is俄文 text "Войдите в свою учетную запись". Below this, there are two input fields: the top one is labeled "Login" and the bottom one is labeled "Password". At the bottom center of the screen is a large purple rectangular button with the word "вход" written in white. On the far left edge of the screen, there is a vertical scroll bar.

Рисунок 9 - Страница входа

Инф. № подп	Подп. и дата	Инф. № дубл.	Взам. инф. №	Подп. и дата
Ли	Изм.	№ докум.	Подп.	Дата

Лист
19

КП-ПР-41-21-2026-ПЗ

2. Главное меню (Для Официанта, повара, администратора) на рисунке 10

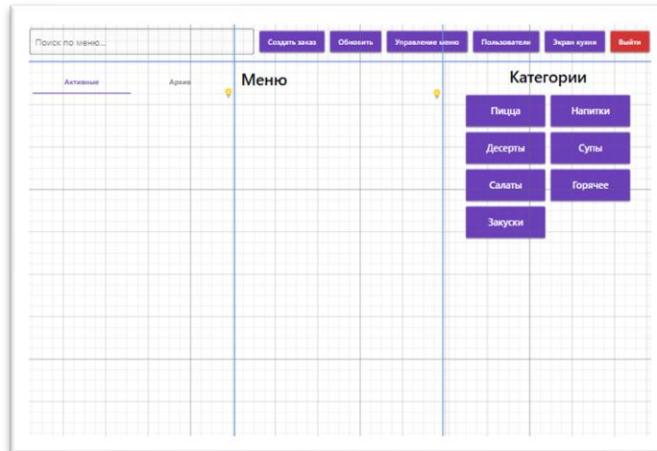


Рисунок 10 - Страница меню

3. Управление пользователями (Только для администратора), на рисунке 11

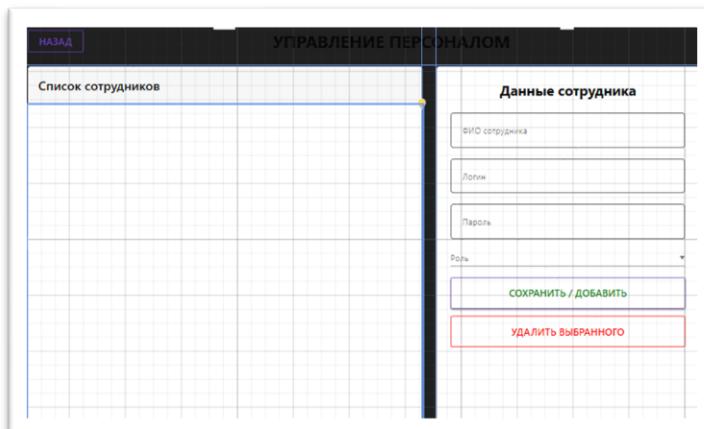


Рисунок 11 - Изменение пользователей

4. Состав заказа, разделение на счета (Повар, Администратор, Официант) продемонстрировано на рисунке 12.

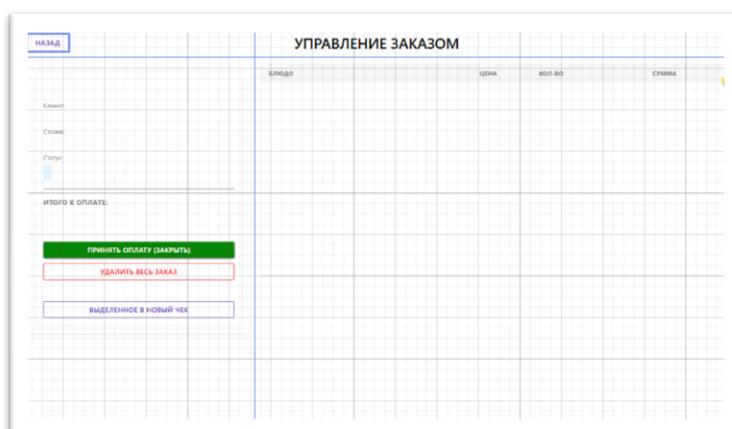


Рисунок 12 - Страница заказа

Инф. № подп	Подп. и дата	Инф. № дубл.	Взам. инф. №

Ли Изм. № докум. Подп. Дата

КП-ПР-41-21-2026-П3

Лист
20

3 Разработка и тестирование

Разработка программного продукта велась с использованием многоуровневой архитектуры (N-Layer), что позволяет разделить ответственность между компонентами системы. Реализация серверной части (ASP.NET Core Web API) построена на основе трех слоев, а именно:

1. Web Layer (Controllers) – принимает HTTP-запросы.
2. Service Layer (Services) — содержит бизнес-логику.
3. Data Access Layer (Repositories) — взаимодействует с базой данных через Entity Framework Core.

На рисунке 13 показано то, как в файле *Program.cs* настроены внедрение зависимостей (DI - Dependency Injection). Что связывает интерфейсы и их реализацию.

```
builder.Services.AddScoped<IMenuItemRepository, MenuItemRepository>();
builder.Services.AddScoped<IUserRepository, UserRepository>();
builder.Services.AddScoped<IOrderRepository, OrderRepository>();
builder.Services.AddScoped<IOrderItemRepository, OrderItemRepository>();

builder.Services.AddScoped<IUserService, UserService>();
builder.Services.AddScoped<IOrderService, OrderService>();
builder.Services.AddScoped<ITokenService, TokenService>();
```

Рисунок 13 - DI

Модуль аутентификации и безопасности. Безопасность реализована с использованием JWT-токенов. Сервис *TokenService* генерирует токен, содержащий ID, Login, и его роль (все это Claims). Реализация на рисунке 14

```
var claims = new List<Claim>
{
    new Claim(ClaimTypes.NameIdentifier, user.UserId.ToString()),
    new Claim(ClaimTypes.Name, user.Login),
    new Claim(ClaimTypes.Role, user.RoleId.ToString())
};
```

Рисунок 14 Данные пользователя в токене

Инф. № подп	Подп. и дата	Инф. № дубл.	Взам. инф. №	Подп. и дата

Одна из главных функций системы является обработка заказов. Реализация этого модуля начинается с контроллера **OrdersController**, который предоставляет API-методы для создания и управления заказами указан на рисунке 15 . Метод создания заказа в **OrderService** показывает основную логику: валидацию блюд, расчет стоимости и сохранение в БД.

```

public async Task<OrderResponseDto> CreateOrderAsync(CreateOrderDto orderDto)
{
    //Создаем новый заказ
    var order = new Order
    {
        UserId = orderDto.UserId,
        TableNumber = orderDto.TableNumber,
        Status = orderDto.Status,
        CreatedAt = DateTime.UtcNow
    };
    //Добавляем блюда в заказ
    foreach (var item in orderDto.Items)
    {
        var menuItem = await _menuItemRepository.GetMenuItemByIdAsync(item.MenuItemId);
        if(menuItem == null || menuItem.Available == false)
        {
            throw new Exception($"Блюдо {item.MenuItemId} не найдено. или же в стоп листе");
        }
        var newOrderItem = new OrderItem
        {
            UnitPrice = menuItem.Price,
            MenuItemId = item.MenuItemId,
            Quantity = item.Quantity,
            MenuItem = menuItem
        };
        order.OrderItems.Add(newOrderItem);
    }
    await _orderRepository.CreateOrderAsync(order);
    // Формируем ответ
    var orderResponse = new OrderResponseDto
    {
        OrderId = order.OrderId,
        UserId = order.UserId,
        TableNumber = order.TableNumber,
        CreatedAt = order.CreatedAt,
        Status = order.Status,
        TotalAmount = order.OrderItems.Sum(oi => oi.UnitPrice * oi.Quantity),

        Items = order.OrderItems.Select(oi => new OrderItemDto
        {
            OrderItemId = oi.OrderItemId,
            OrderId = oi.OrderId,
            MenuItemId = oi.MenuItemId,
            Quantity = oi.Quantity,
            UnitPrice = oi.UnitPrice,
            MenuItemName = oi.MenuItem?.Name ?? "Неизвестно"
        }).ToList()
    };
    return orderResponse;
}

```

Рисунок 15 - Создание заказа

Инф. № подп	Подп. и дата

Ли	Изм.	№ докум.	Подп.	Дата

Серверная часть приложения развертывается в Docker-контейнере. Для обеспечения переносимости, изоляции, упрощения процесса развертывания использовалась контейнеризация Docker. Это гарантирует одинаковое поведение приложения как на локальной машине разработчика, так и на сервере. Конфигурация контейнера описана в Dockerfile, представленном на рисунке 16.

```
# Образ для запуска сервера
FROM mcr.microsoft.com/dotnet/aspnet:9.0 AS base
USER $APP_UID
WORKDIR /app
EXPOSE 8080
EXPOSE 8081

# Образ для сборки приложения
FROM mcr.microsoft.com/dotnet/sdk:9.0 AS build
ARG BUILD_CONFIGURATION=Release
WORKDIR /src
COPY ["CafeAPI/CafeAPI.csproj", "CafeAPI/"]
RUN dotnet restore "./CafeAPI/CafeAPI.csproj"
COPY .
WORKDIR "/src/CafeAPI"
RUN dotnet build "./CafeAPI.csproj" -c $BUILD_CONFIGURATION -o /app/build

# Сборка и публикация приложения
FROM build AS publish
ARG BUILD_CONFIGURATION=Release
RUN dotnet publish "./CafeAPI.csproj" -c $BUILD_CONFIGURATION -o /app/publish /p:UseAppHost=false

# Финальный этап – создание исполняемого образа
FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "CafeAPI.dll"]
```

Рисунок 16 – Dockerfile

В программе Docker идет демонстрация запущенного контейнера, в котором находится база данных (cafedb) и сервер api (cafeapi) продемонстрировано на рисунке 17.

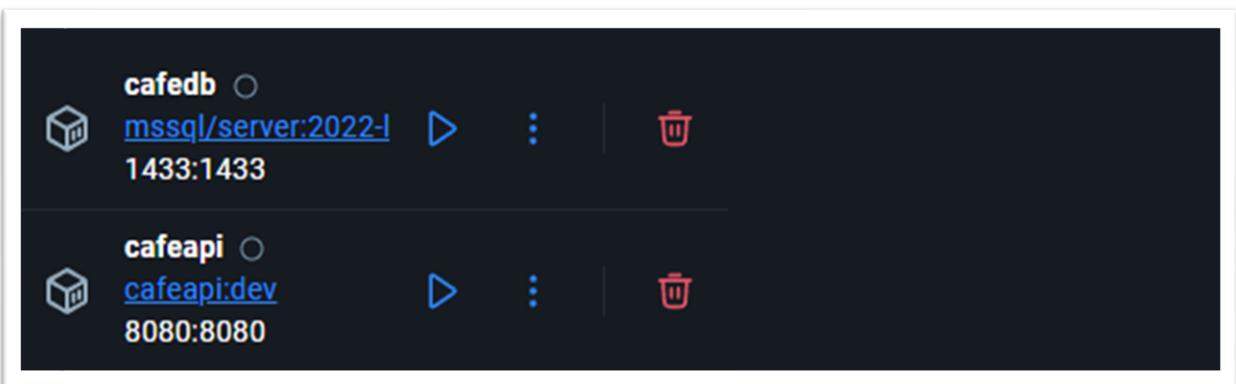


Рисунок 17 – Контейнер

Инд. № подп	Подп. и дата	Инф. № подп.	Взам. инф. №	Подп. и дата

Отправка данных в клиентском приложении реализовано через: класс ***ApiService*** взаимодействует с сервером для управления данными через экземпляр ***_httpClient***. Ниже представлен пример метода авторизации на рисунке 18.

```
public async Task<LoginResponseDto?> LoginAsync(string login, string password)
{
    // Создаем DTO для передачи данных
    var loginDto = new LoginUserDto
    {
        Login = login,
        Password = password
    };
    // Выполняем POST-запрос
    try
    {
        // Добавляем логирование запроса
        LogRequest("api/User/Login");

        // Отправляем запрос на сервер
        var response = await _httpClient.PostAsJsonAsync("api/User/Login", loginDto );
        if (response.IsSuccessStatusCode) // Проверяем успешность ответа
        {
            // Читаем и возвращаем данные из ответа
            return await response.Content.ReadFromJsonAsync<LoginResponseDto>();
        }
        else return null;
    }
    //возможные исключения
    catch (Exception ex)
    {
        Debug.WriteLine(ex);
        return null;
    }
}
```

Рисунок 18 - Авторизация

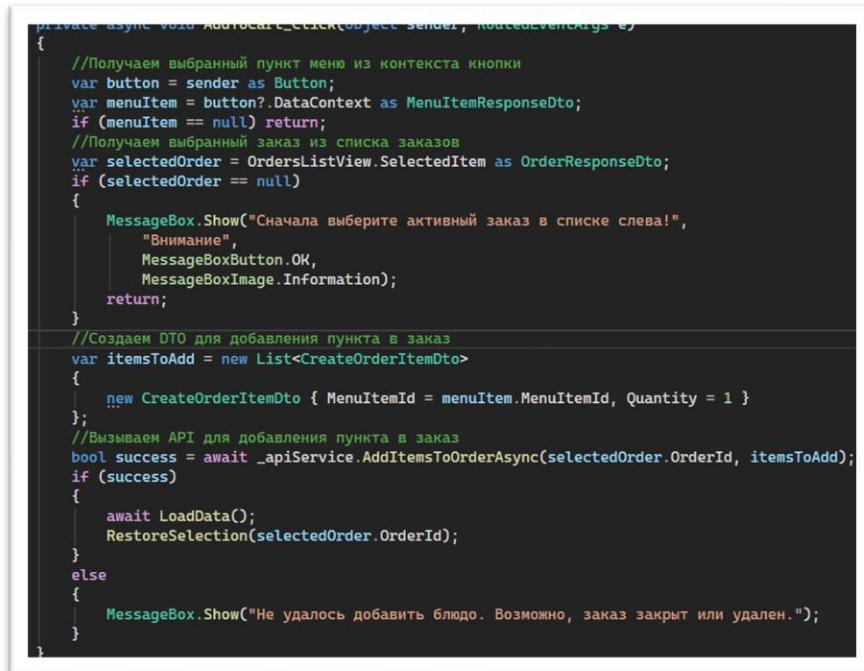
Метод обновления профиля. Явно показано как берет ссылку на эндпоинт, сериализует объект и отправляет на сервер. Рисунок 19

```
public async Task<bool> UpdateUserAsync(int userId, CreateUserDto userDto, int roleID)
{
    try
    {
        string url = $"api/User/{userId}/UpdateUser?Role={roleID}";
        var response = await _httpClient.PatchAsJsonAsync(url, userDto);
        return response.IsSuccessStatusCode;
    }
    catch (Exception ex)
    {
        Debug.WriteLine($"ОШИБКА ОБНОВЛЕНИЯ: {ex.Message}");
        return false;
    }
}
```

Рисунок 19 - Обновление профиля

Инф. № подл	Подл. и дата

Кроме авторизации и обновления профиля, важной частью клиентского приложения является добавления позиций в заказ (MainPage). В файле *MainPage.xaml.cs* реализована логика добавлений позиции меню в заказ. Демонстрация на рисунке 20.



```
private async void AddMenuItem_Click(object sender, RoutedEventArgs e)
{
    //Получаем выбранный пункт меню из контекста кнопки
    var button = sender as Button;
    var menuItem = button?.DataContext as MenuItemResponseDto;
    if (menuItem == null) return;
    //Получаем выбранный заказ из списка заказов
    var selectedOrder = OrdersListView.SelectedItem as OrderResponseDto;
    if (selectedOrder == null)
    {
        MessageBox.Show("Сначала выберите активный заказ в списке слева!",
            "Внимание",
            MessageBoxButton.OK,
            MessageBoxImage.Information);
        return;
    }
    //Создаем DTO для добавления пункта в заказ
    var itemsToAdd = new List<CreateOrderItemDto>
    {
        new CreateOrderItemDto { MenuItemId = menuItem.MenuItemId, Quantity = 1 }
    };
    //Вызываем API для добавления пункта в заказ
    bool success = await _apiService.AddItemsToOrderAsync(selectedOrder.OrderId, itemsToAdd);
    if (success)
    {
        await LoadData();
        RestoreSelection(selectedOrder.OrderId);
    }
    else
    {
        MessageBox.Show("Не удалось добавить блюдо. Возможно, заказ закрыт или удален.");
    }
}
```

Рисунок 20 - Карточки товара

Тестирование программного продукта проводилось с целью проверки корректности работы реализованных модулей, устойчивости системы к ошибочным действиям пользователя, а также соответствия функциональных возможностей заявленным требованиям.

Методика тестирования осуществлялась по принципу *черного ящика*, при котором проверялась работа системы без анализа внутренней реализации кода. Тестирование выполнялось параллельно с процессом разработки и включало ручную проверку серверной и клиентской частей приложения. Для комплексной оценки работоспособности были разработаны тестовые сценарии, охватывающие основные функции автоматизированной системы. В ходе тестирования были реализованы и проверены механизмы валидации пользовательских действий.

Проверка на то выбран ли заказ слева чтобы добавлять в него позиции меню, программа сразу предупреждает и не дает сделать действие, потому что что id

Инф. № подл	Подл. и дата	Инф. № дубл.	Взам. инф. №	Подл. и дата
-------------	--------------	--------------	--------------	--------------

Ли	Изм.	№ докум.	Подп.	Дата
----	------	----------	-------	------

заказа пуст, нету того места куда добавлять заказы, так же и на сервере есть проверки в контроллерах на то, чтобы данные приходили полные, а не битые или полупустые. Демонстрация на рисунке 21.

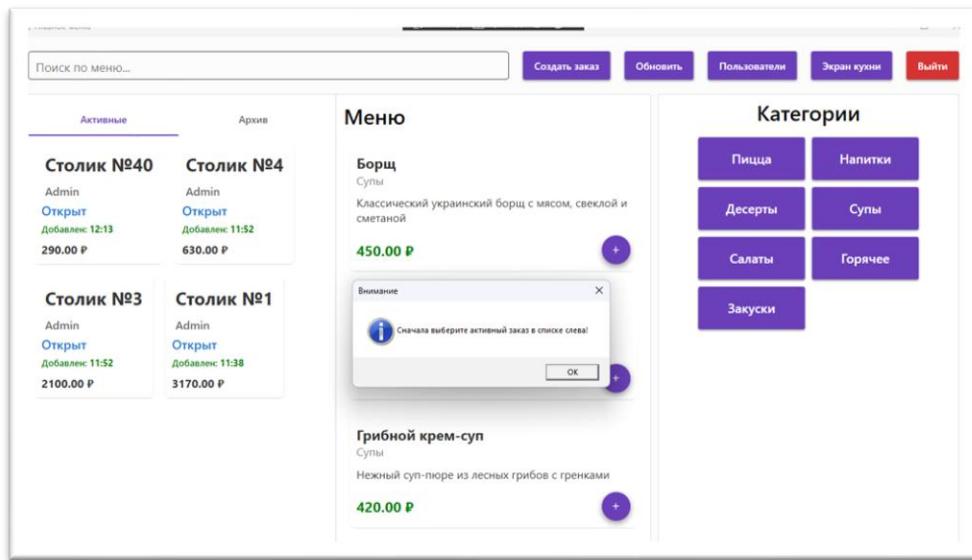


Рисунок 21 – Проверка добавления заказа

Проверка на занятый стол аналогична тоже важна мы не можем закрепить на один стол два заказа. Демонстрация на рисунке 22.

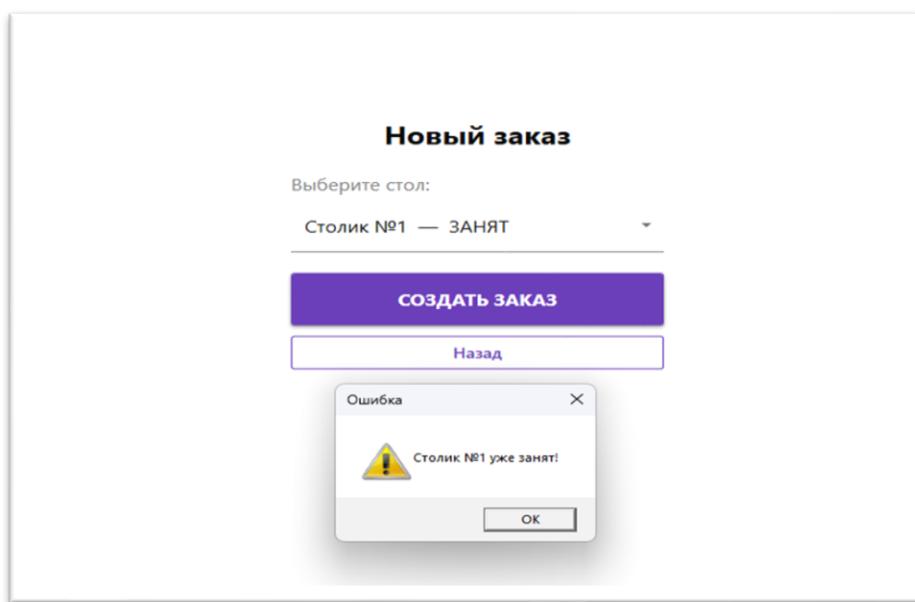


Рисунок 22 – Проверка стола.

Следующая проверка это удаление блюд из архива. Важность такой проверки конечная выручка, то есть взять и посчитать сумму всех заказов. Продемонстрировано на рисунке 23.

Инф. № подп	Подп. и дата	Инф. № дубл.	Взам. инф. №	Подп. и дата	Лист
Ли	Изм.	№ докум.	Подп.	Дата	26
					KП-ПР-41-21-2026-ПЗ

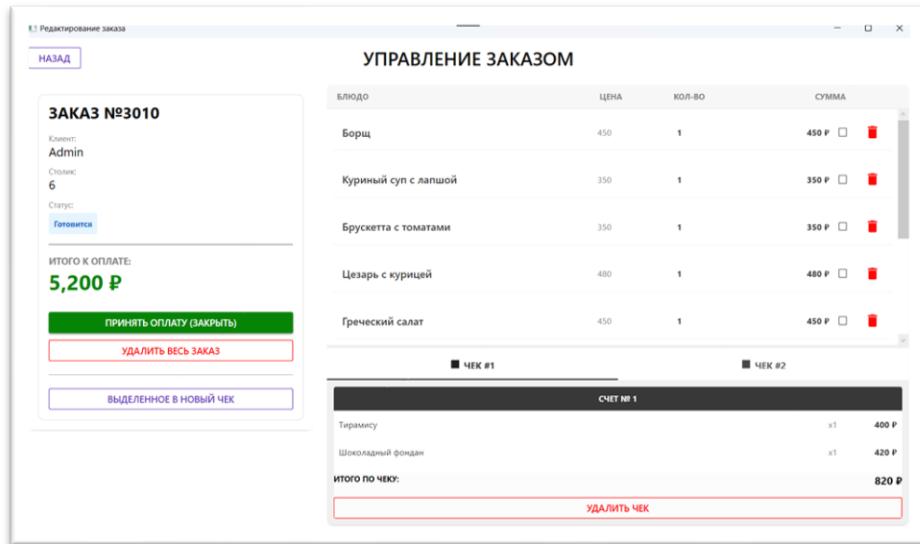


Рисунок 23 - Удаление из архива

Проверка на статус заказа, если заказ благополучно готов или оплачен, то заказ уходит в архив и он не будет отображаться на странице кухни. Продемонстрировано на рисунке 24.

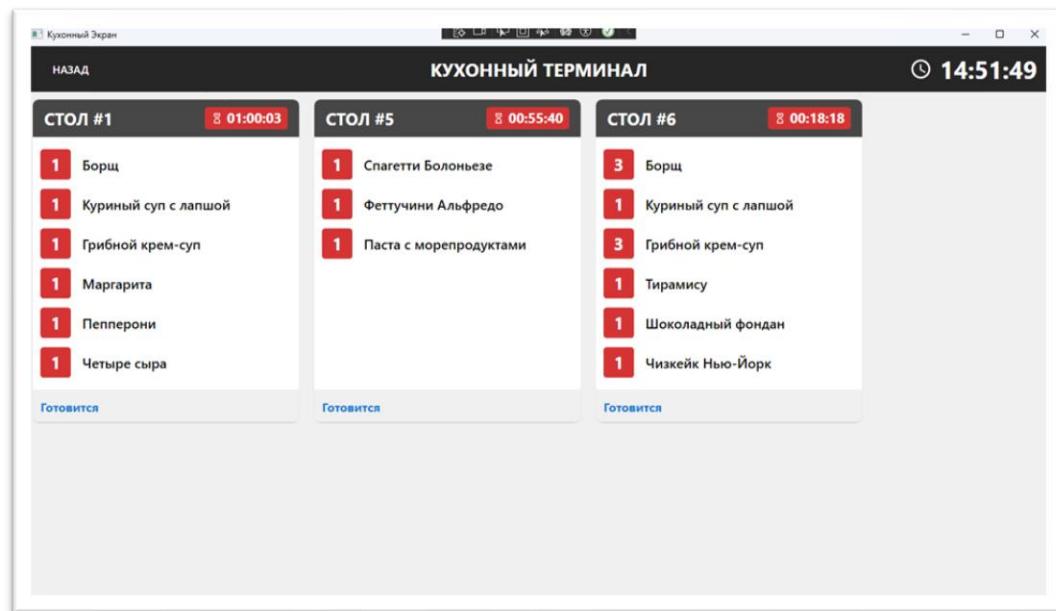


Рисунок 24 - Статус заказа кухня

Тестирование API-сервера проводилось вручную. Каждый эндпоинт проверялся путём отправки HTTP-запросов с различными входными данными и анализа ответов сервера.

Для этого использовался инструмент *Swagger*, который автоматически формирует описание доступных эндпоинтов и подставляет шаблоны JSON-

Инф. № подл	Подл. и дата	Инф. № дубл.	Взам. инф. №	Подл. и дата
-------------	--------------	--------------	--------------	--------------

Ли	Изм.	№ докум.	Подп.	Дата
----	------	----------	-------	------

запросов для их заполнения. Это упрощает процесс тестирования, так как позволяет быстро отправлять запросы и сразу получать коды ответов сервера.

Клиентское приложение на WPF и тестировщик анализируют HTTP-коды ответов и соответствующим образом реагируют на результат выполнения запроса. Основные коды ответов сервера приведены в **Таблица 2 - Таблица ответов сервера с расшифровкой кодов**.

Таблица 2 - Таблица ответов сервера с расшифровкой кодов.

Код	Название	Описание
200	OK	Запрос успешно выполнен
201	Created	Ресурс успешно создан
400	Bad Request	Ошибка валидации входных данных
401	Unauthorized	Отсутствует или некорректен токен авторизации
403	Forbidden	Недостаточно прав для выполнения операции
404	Not Found	Запрошенный ресурс не найден
500	Internal Server Error	Внутренняя ошибка сервера

Таким образом, использование стандартных HTTP-кодов позволяет упростить обработку результатов запросов как на стороне сервера, так и на стороне клиентского приложения. Это упрощает реализацию логики обработки ошибок, повышает читаемость кода и делает систему более устойчивой к некорректным входным данным и ошибкам выполнения.

Инф. № подп	Подп. и дата	Инф. № дубл.	Взам. инф. №	Подп. и дата

Ли	Изм.	№ докум.	Подп.	Дата

Заключение

В ходе выполнения курсового проекта была выполнена цель разработки автоматизированной системы для кафе. Главной целью работы являлось создание автоматизированной системы, для оптимизации процессов обслуживания клиентов.

Проведен анализ предметной области. На основе анализа была спроектирована архитектура будущей системы. Выбрана клиент-серверная архитектура с использованием REST API, обеспечивающая гибкость и масштабируемость решения.

Реализована база данных и логическая структура. Спроектирована и создана реляционная база данных на MS SQL Server, включающая таблицы для хранения данных о пользователях, ролях, категориях меню и заказах. Взаимодействие приложения с базой данных организовано посредством ORM Entity Framework Core, что упростило манипуляцию данными.

Реализована серверная часть приложения. Разработан API на платформе ASP.NET Core Web API. Внедрена система аутентификации и авторизации на основе JWT-токенов, обеспечивающая разграничение прав доступа. Созданы контроллеры для управления меню, обработки заказов и администрирования пользователей.

Разработаны клиентская часть и пользовательский интерфейс. Создано приложение на технологии WPF с понятным графическим интерфейсом. Реализована сложная бизнес-логика, включая проверку занятости столиков в реальном времени для предотвращения дублирования заказов.

Дополнительно была обеспечена переносимость системы: для упрощения развертывания серверной части применена контейнеризация Docker.

Таким образом, в результате проделанной работы цель и все поставленные задачи были выполнены в полном объеме.

Инф. № подл	Подл. и дата	Инф. № дубл.	Взам. инф. №	Подл. и дата
-------------	--------------	--------------	--------------	--------------

Ли	Изм.	№ докум.	Подп.	Дата	KП-ПР-41-21-2026-П3	Лист 29
----	------	----------	-------	------	---------------------	------------

Список использованной литературы

1. Макдональд М. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов [Текст] / М. Макдональд. - Москва: Вильямс, 2013. - 1024 с.
2. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения [Текст] / Р. Мартин. - Санкт-Петербург: Питер, 2021. - 352 с.
3. Прайс М. C# 9 и .NET 5. Разработка и оптимизация [Текст] / М. Прайс. - Санкт-Петербург: Питер, 2022. - 848 с.
4. Троелсен Э. Язык программирования C# 9 и платформа .NET 5 [Текст] / Э. Троелсен, Ф. Джепикс. - Москва: Вильямс, 2022. - 1328 с.
5. Фримен А. ASP.NET Core для профессионалов [Текст] / А. Фримен. - Москва: Вильямс, 2019. - 912 с.
6. Документация по службе «Управление API» [Электронный ресурс]. - Режим доступа: <https://learn.microsoft.com/ru-ru/azure/api-management/> (дата обращения 20.01.2026)
7. Документация по ASP.NET Core [Электронный ресурс]. - Режим доступа: <https://learn.microsoft.com/ru-ru/aspnet/core/> (дата обращения 20.01.2026)
8. Документация по Entity Framework Core [Электронный ресурс]. - Режим доступа: <https://learn.microsoft.com/ru-ru/ef/core/> (дата обращения 20.01.2026)
9. Руководство по языку C# [Электронный ресурс]. - Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/csharp/> (дата обращения 20.01.2026)

Инф. № подп	Подп. и дата	Инф. № дубл.	Взам. инф. №	Подп. и дата

Ли	Изм.	№ докум.	Подп.	Дата

КП-ПР-41-21-2026-ПЗ

Лист
30

Приложения

Инф. № подп	Подп. и дата	Инф. № дубл.	Взам. инф. №	Подп. и дата

Ли	Изм.	№ докум.	Подп.	Дата

КП-ПР-41-21-2026-ПЗ

Лист
31

Приложение А. Исходный код.

Репозиторий проекта:

<https://github.com/Bikmacs/CafeSystem>

Рисунок 25 - QR-Code репозитория.

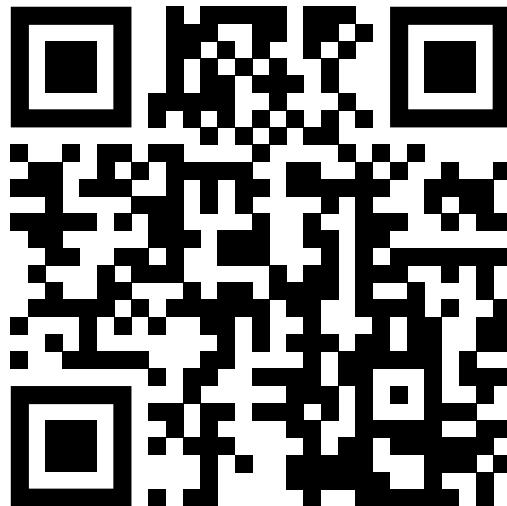


Рисунок 25 - QR-Code

Инф. № подп	Подп. и дата	Инф. № дубл.	Взам. инф. №	Подп. и дата

Ли	Изм.	№ докум.	Подп.	Дата

КП-ПР-41-21-2026-ПЗ

Лист
32

Приложение Б. UI сервера

Для документирования и тестирования серверной части (Backend) используется инструмент Swagger UI. Он предоставляет графический интерфейс для отправки HTTP-запросов к API и просмотра схем данных.

Полная спецификация разработана в формате OpenAPI 3.0 и опубликована с использованием инструментов Swagger UI на платформе GitHubPages. Это позволяет разработчикам фронтенда и тестировщикам взаимодействовать с сервером удаленно, без необходимости разворачивать локальное окружение. Ссылка: <https://bikmacs.github.io/cafeSwagger/> QR-Code ведущий на сайт с описание эндпоинтов. На рисунке 26.

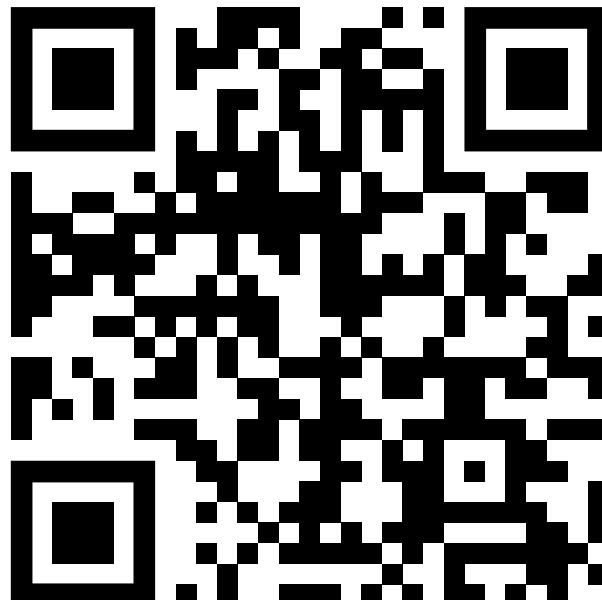


Рисунок 26 - Qrcode

Инф. № подп	Подп. и дата	Инф. № дубл.	Взам. инф. №	Подп. и дата

Ли	Изм.	№ докум.	Подп.	Дата

КП-ПР-41-21-2026-ПЗ

Лист
33

Приложение В. Интерфейс программы

Карта приложения. Схема навигации по приложению, показывает какие страницы есть в данном приложении. Рисунок 27.

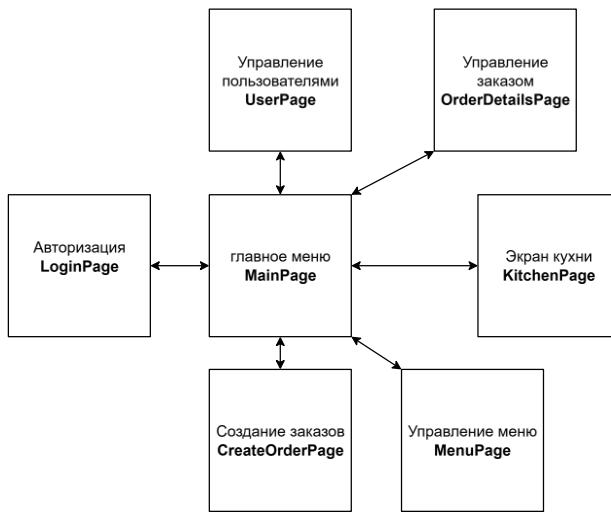


Рисунок 27 - Карта приложения

Схема приложения клиента показывает связи и переходы данных. Объясняет, как действуют элементы программы, какая последовательность действий. Рисунок 28.

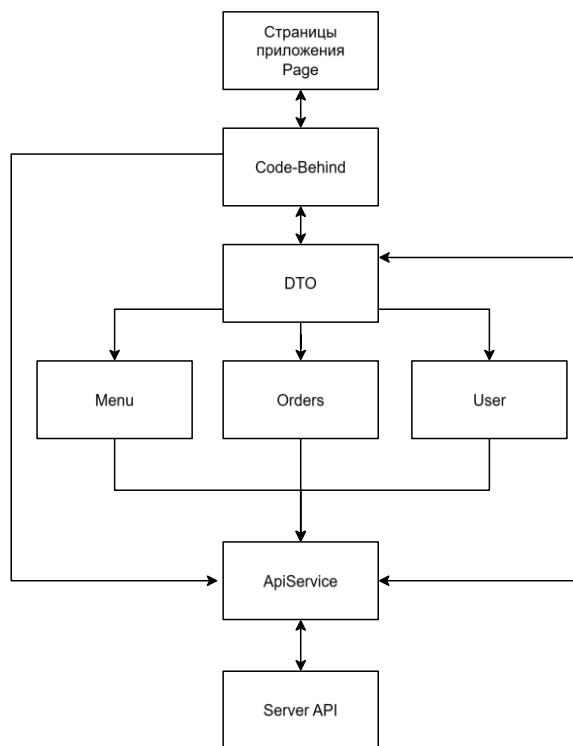


Рисунок 28 – Схема работы

Инф. № подп	Подп. и дата	Инф. № дубл.	Взам. инф.	Взам. № дубл.	Подп. и дата	Лист
Ли	Изм.	№ докум.	Подп.	Дата		34

Приложение Г. База данных.

Диаграмма базы данных кафе с типом данных и null значением полей, и указанием связей, продемонстрирована на рисунке 29.

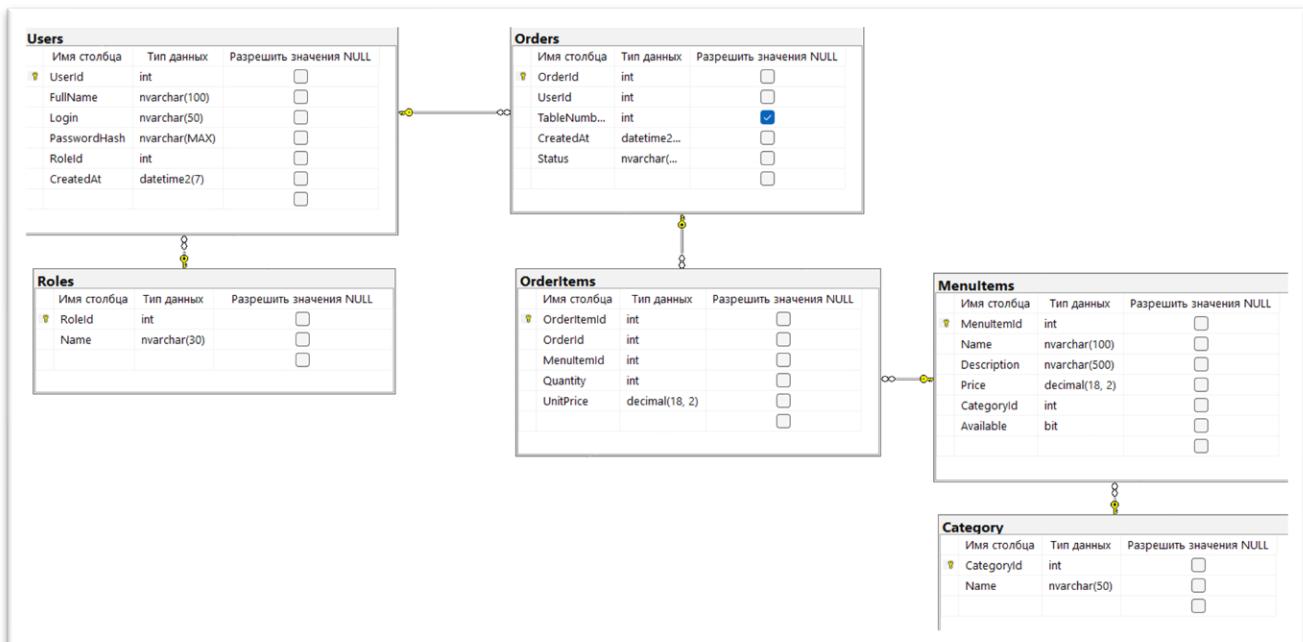


Рисунок 29 - Диаграмма баз данных

Зарезервированные данные , используемые для автоматического заполнения базы данных при инициализации, определены в контексте базы данных сервера в классе *CafeDbContext* и доступны по ссылке: <https://github.com/Bikmacs/CafeSystem/blob/main/CafeSolution/CafeAPI/Data/CafeDbContext.cs>. QR Code. Рисунок 30.



Рисунок 30 - Qrcode

Инф. № подп	Подп. и дата	Инф. № дубл.	Взам. инф. №	Подп. и дата	Лист