**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc165050057)

[ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ И ОБЩАЯ ПОСТАНОВКА ЗАДАЧ 5](#_Toc165050058)

[1.1 История и развитие служб доставки еды 5](#_Toc165050059)

[1.2 Организационная структура службы доставки 8](#_Toc165050060)

[1.3 Функции, выполняемые менеджером заказов 12](#_Toc165050061)

[1.4 Постановка задачи 15](#_Toc165050062)

[ГЛАВА 2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ 17](#_Toc165050063)

[2.1 Требования к программе 17](#_Toc165050064)

[2.2 Руководство пользователя 18](#_Toc165050065)

[2.3 Руководство программиста 27](#_Toc165050066)

[ЗАКЛЮЧЕНИЕ 31](#_Toc165050067)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 32](#_Toc165050068)

[ПРИЛОЖЕНИЕ 33](#_Toc165050069)

# ВВЕДЕНИЕ

В современном мире автоматизация играет ключевую роль в оптимизации бизнес-процессов. Компьютерные технологии позволяют эффективно управлять операциями, сокращать временные затраты и минимизировать ошибки, связанные с человеческим фактором. Особенно актуальна автоматизация в сфере услуг, где скорость и точность обработки заказов напрямую влияют на удовлетворенность клиентов.

Разработка программной системы для управления заказами в службе доставки еды является важной задачей, так как позволяет упорядочить процессы приема, обработки и выполнения заказов, а также улучшить взаимодействие между клиентами, курьерами и ресторанами. Такая система помогает вести учет заказов, контролировать сроки доставки, управлять финансовыми операциями и анализировать эффективность работы службы.

Целью данной курсовой работы является создание программной системы для управления заказами в службе доставки еды.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучить и описать предметную область.
2. Определить ключевые бизнес-процессы службы доставки.
3. Разработать систему, обеспечивающую: удобное оформление и отслеживание заказов; учет клиентов, курьеров и партнерских ресторанов; автоматизацию финансовых расчетов и формирование отчетности; повышение скорости и качества обслуживания клиентов; защиту персональных данных пользователей.

Объектом исследования является процесс управления заказами в службе доставки еды.

Предметом исследования выступает разработка программной системы для автоматизации данного процесса.

Методы исследования, использованные в работе:

1. анализ (при изучении предметной области);
2. синтез (при проектировании системы);
3. формализация (при моделировании структуры базы данных и интерфейсов).

Разрабатываемая система позволит оптимизировать обработку заказов, сократить время их выполнения, упростить взаимодействие между участниками процесса и повысить общую эффективность работы службы доставки.

# ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ И ОБЩАЯ ПОСТАНОВКА ЗАДАЧ

## 1.1 История и развитие служб доставки еды

## Службы доставки еды представляют собой важный сегмент современной экономики, претерпевший значительную эволюцию за последние десятилетия. Истоки этой услуги можно проследить в глубокой древности, когда первые формы доставки пищи начали появляться в развитых цивилизациях своего времени.

## В Древнем Риме существовала развитая система термополий - закусочных, где горожане могли приобрести готовую пищу. Как отмечает историк Стивен Эллис в своей работе "Римская торговая революция", эти заведения нередко предлагали услуги по доставке еды состоятельным гражданам. Аналогичные сервисы существовали и в других древних цивилизациях. В Китае эпохи династии Сун (X-XIII века), как описывает Жак Герне в своем исследовании повседневной жизни того периода, существовала развитая сеть уличных торговцев, которые доставляли готовые блюда в дома зажиточных горожан.

## Современный этап развития служб доставки еды начался в конце XIX - начале XX века. В США в 1880-х годах появились первые службы доставки обедов для рабочих, а в 1920-х годах началась эра доставки пиццы. Как отмечают Джекл и Скалл в своей работе о фастфуде, именно в этот период сформировалась бизнес-модель, которая легла в основу современных служб доставки. В 1960-х годах с развитием телефонной связи и автомобильной инфраструктуры сервисы доставки еды стали массовым явлением в развитых странах. Значительный прорыв в индустрии проишел в 1990-х годах с распространением интернета. Первые онлайн-заказы еды появилсь еще до широкого распространения всемирной паутины – в 1979 году в Великобритании была запущена система Videotex, позволяющая делать заказы через телевизор. Однако революция началась в 2000-х годах с появлением специализированных онлайн-платформ.

## Важными вехами в развитии индустрии доставки еды стали такие события, как основание в 2004 году Grubhub — одной из первых крупных платформ онлайн-заказов в США, запуск российского сервиса Delivery Club в 2009 году, а также появление в 2014 году Uber Eats, созданного как ответвление компании Uber. Современный этап развития отрасли, охватывающий 2020-е годы, отличается рядом ключевых тенденций. Одной из них является консолидация рынка: крупные компании активно поглощают более мелких конкурентов. Так, в 2020 году Just Eat приобрел Grubhub за 7,3 миллиарда долларов, что позволило создать одного из крупнейших игроков на глобальном рынке. Существенную роль играет технологическая трансформация, включающая внедрение искусственного интеллекта для оптимизации маршрутов доставки, прогнозирования спроса и персонализации клиентских предложений. Согласно исследованию Chen и соавторов (2022), использование AI-технологий позволяет сократить среднее время доставки на 15–20 процентов. Также наблюдается расширение спектра предлагаемых услуг: помимо доставки готовых блюд, компании предоставляют возможность заказа продуктов, полуфабрикатов и даже рецептов с необходимыми ингредиентами. Еще одной заметной тенденцией стало развитие так называемых "темных кухонь" (dark kitchens) - специализированных производств, работающих исключительно в формате доставки и не предусматривающих физического присутствия посетителей.

## В России рынок доставки еды начал активно развиваться в 2010-х годах. По данным Research and Markets, объем российского рынка в 2023 году превысил $5 млрд при ежегодном росте около 15%. Пандемия COVID-19 стала катализатором роста индустрии: по данным McKinsey, в 2020 году спрос на услуги доставки вырос на 300% .

## Однако, как отмечает российский эксперт А. Иванов в своей работе о цифровой трансформации ресторанного бизнеса, многие локальные службы доставки до сих пор используют устаревшие методы управления заказами. Это создает значительный потенциал для автоматизации и оптимизации бизнес-процессов в данном сегменте.

## Современные системы управления заказами должны решать комплекс задач:

## автоматизация приема и обработки заказов;

## интеграция с ресторанами и курьерскими службами;

## управление клиентской базой и лояльностью;

## аналитика и прогнозирование спроса;

## оптимизация логистики доставки.

## Развитие технологий продолжает трансформировать индустрию. По данным Statista, к 2025 году мировой объем рынка доставки еды достигнет $200 млрд, при этом доля мобильных заказов составит более 70%. Эти тенденции делают разработку эффективных систем управления заказами особенно актуальной.

## 1.2 Организационная структура службы доставки

## Современная служба доставки еды представляет собой сложную организационную систему, эффективность которой напрямую зависит от четкого распределения функций между структурными подразделениями. В данном разделе рассматривается типичная организационная структура крупной службы доставки, анализируются функции каждого подразделения и их взаимодействие.

## Рассмотрим основные структурные подразделения:

## Директора: финансовый директор, технический директор, директор по маркетингу. Высшее руководство осуществляет стратегическое управление компанией, определяет основные направления развития, контролирует выполнение ключевых показателей эффективности (KPI).

## Отдел разработки и IT: разработчики программного обеспечения, системные администраторы, аналитики данных, специалисты по кибербезопасности. Данный отдел отвечает за: разработку и поддержку программного обеспечения, интеграцию с ресторанами-партнерами, обслуживание мобильных приложений и веб-платформы, защиту персональных данных клиентов, анализ больших данных для оптимизации бизнес-процессов.

## Операционный отдел: менеджеры по работе с ресторанами, координаторы заказов, диспетчеры, менеджеры по качеству. Основные функции: взаимодействие с ресторанами-партнерами, контроль выполнения заказов, решение конфликтных ситуаций, обеспечение качества услуг.

## Логистический отдел: менеджеры по логистике, координаторы курьеров, аналитики маршрутов. Отвечает за: оптимизацию маршрутов доставки, распределение заказов между курьерами, контроль сроков доставки, внедрение систем GPS-мониторинга.

## Отдел маркетинга и продаж: маркетологи, SMM-специалисты, менеджеры по продажам, копирайтеры. Основные задачи: разработка маркетинговых стратегий, продвижение бренда, управление социальными сетями, проведение рекламных кампаний, анализ конкурентов.

## Клиентский сервис: операторы call-центра, менеджеры по работе с клиентами, специалисты по лояльности. Функции: консультирование клиентов, обработка жалоб и предложений, программы лояльности, обратная связь с клиентами.

## Финансовый отдел: бухгалтерия, финансовые аналитики, специалисты по расчетам. Отвечает за: финансовое планирование, взаиморасчеты с партнерами, налоговую отчетность, анализ финансовых показателей.

## Взаимодействие подразделений

Эффективная работа службы доставки возможна только при четкой координации всех структурных подразделений. Процесс обработки заказа начинается с того, что клиент оформляет заказ через мобильное приложение, за работу которого отвечает отдел разработки. После этого информация о заказе поступает в операционный отдел, где она обрабатывается и передается в соответствующий ресторан-партнер. Параллельно логистический отдел определяет наиболее подходящего курьера с учетом его местоположения и загруженности, обеспечивая быструю и эффективную доставку. В ходе выполнения заказа сотрудники клиентского сервиса остаются на связи, предоставляя оперативные ответы на вопросы клиента. По завершении доставки финансовый отдел проводит расчеты с рестораном и курьером, включая начисление комиссий и бонусов. Завершающим этапом является аналитическая работа отдела маркетинга, который изучает поведение клиента с целью последующей персонализации предложений и повышения лояльности.

## Тенденции развития организационных структур

Современные тенденции в организации служб доставки отражают стремление компаний к повышению эффективности и адаптивности в условиях быстро меняющейся рыночной среды. Одним из ключевых направлений является укрупнение и централизация основных функций, что позволяет оптимизировать управление ресурсами и сократить издержки. При этом второстепенные процессы, такие как бухгалтерия, техническая поддержка или логистика, все чаще передаются на аутсорсинг, что способствует концентрации на ключевых компетенциях бизнеса. Широкое распространение получают гибкие организационные структуры, обеспечивающие быстрое реагирование на изменения спроса и повышение уровня клиентского сервиса. Значительное внимание уделяется развитию франчайзинговых моделей, позволяющих ускорить масштабирование бизнеса при сохранении единых стандартов качества. Наконец, важным трендом стало появление так называемых "виртуальных ресторанов" — формата, при котором заведения работают исключительно на доставку, не имея физических точек обслуживания клиентов, что снижает операционные расходы и расширяет возможности для экспериментальных концепций.

Организационная структура продолжает эволюционировать под влиянием технологических изменений и меняющихся потребностей потребителей. В ближайшие годы ожидается дальнейшая автоматизация процессов и интеграция искусственного интеллекта в систему управления доставкой.

## 1.3 Функции, выполняемые менеджером заказов

Менеджер заказов в службе доставки еды играет центральную роль в организации и координации всех процессов, связанных с приемом, обработкой и выполнением заказов. Его деятельность охватывает широкий спектр задач, направленных на обеспечение бесперебойной работы сервиса, удовлетворение потребностей клиентов и поддержание эффективного взаимодействия между всеми участниками процесса (курьерами, ресторанами, клиентами и техническими системами).

1. Прием и обработка заказов

Сотрудники отдела осуществляют регистрацию входящих заказов, поступающих через мобильное приложение, сайт, телефонные звонки и чат-боты. Особое внимание уделяется проверке корректности введённых данных: адреса доставки, состава заказа и дополнительных пожеланий клиента. Контролируется наличие выбранных позиций в меню ресторанов-партнёров, а также оперативно вносятся изменения в заказы, включая замену блюд, изменение времени или адреса доставки.

1. Координация с ресторанами-партнерами

Обеспечивается передача заказов в рестораны и получение подтверждения их принятия. Отслеживаются сроки приготовления, а возникающие затруднения, такие как отсутствие отдельных блюд или задержки, решаются в оперативном порядке. Ведётся актуальная база данных ресторанов с указанием меню, графика работы и контактной информации ответственных лиц.

1. Управление курьерской службой

Доставка заказов организуется с учётом текущей загруженности и местоположения курьеров. Для сокращения времени ожидания клиентов осуществляется оптимизация маршрутов, а передвижение курьеров контролируется с помощью GPS-систем. Отдел также отвечает за урегулирование внештатных ситуаций, таких как транспортные затруднения, отмена заказа или другие задержки.

1. Взаимодействие с клиентами

Пользователям предоставляется консультационная поддержка по вопросам меню, акционных предложений и условий доставки. На всех этапах оформления и выполнения заказа клиенты получают информацию о его статусе. Отдел также обрабатывает обращения, связанные с жалобами, разрешением спорных ситуаций, а также организует сбор обратной связи и развитие программ лояльности.

1. Финансовые операции и отчетность

Осуществляется контроль оплаты заказов, включая онлайн-платежи, расчёты наличными и корпоративные переводы. Проводится сверка финансовых данных с ресторанами и курьерами, включая расчёт бонусов и штрафных санкций. Формируются регулярные отчёты по выручке, количеству выполненных заказов и возвратам, а также проводится анализ ключевых показателей, таких как средний чек, время доставки и уровень отмен.

1. Работа с ИТ-системами

Обеспечивается ведение базы данных заказов и клиентов с использованием CRM-систем. Регулярно обновляется информация о доступности блюд и акциях. В случае технических неполадок в работе приложений или сайта сотрудники взаимодействуют с IT-отделом для их оперативного устранения.

1. Контроль качества сервиса

Проводится контроль соблюдения стандартов обслуживания, включая своевременность и корректность ответов на обращения клиентов. Анализируются причины негативных отзывов и разрабатываются мероприятия по их устранению. Кроме того, организуются тренинги и обучающие программы для новых сотрудников, в том числе менеджеров и курьеров.

1. Аналитика и оптимизация процессов

Осуществляется комплексный анализ логистической цепочки с целью выявления узких мест, например, ресторанов с регулярными задержками. Прогнозируются периоды пиковых нагрузок (праздничные дни, неблагоприятные погодные условия) и разрабатываются превентивные меры. Также проводится тестирование новых функций сервиса, включая возможность оформления предзаказа на заданное время.

Менеджер заказов в службе доставки еды сочетает функции организатора, коммуникатора и аналитика. Его работа требует не только внимательности к деталям, но и навыков многозадачности, стрессоустойчивости и умения быстро принимать решения. Автоматизация части этих функций через специализированную информационную систему (например, интеграцию CRM с системами ресторанов и курьерскими сервисами) значительно повышает эффективность работы и удовлетворенность клиентов.

Исходя из описанных выше функций, выполняемых администратором гостинице можно перейти к постановке задачи для разработки информационной системы.

## 1.4 Постановка задачи

Общая задача курсовой работы заключается в разработке программной системы для управления заказами в службе доставки еды, которая будет включать следующий функционал:

1. Управление заказами: прием и обработка заказов (онлайн, через мобильное приложение, телефонные звонки). Отслеживание статуса заказа (принят, готовится, передан курьеру, доставлен). Возможность изменения или отмены заказа клиентом.
2. Учет клиентов: хранение данных о клиентах (ФИО, контактные данные, история заказов, предпочтения). Система лояльности (бонусные программы, скидки, персональные предложения).
3. Учет ресторанов-партнеров: Ведение базы ресторанов (название, меню, время работы, контакты). Контроль доступности блюд и актуальности цен.
4. Управление курьерами: распределение заказов между курьерами. GPS-трекинг для отслеживания местоположения курьеров. Учет рабочего времени и выплат.
5. Финансовые операции: прием платежей (онлайн, наличные, корпоративные счета). Формирование отчетов по выручке, комиссиям, расходам. Интеграция с бухгалтерскими системами.
6. Аналитика и отчетность: анализ популярных блюд, среднего чека, времени доставки. Выявление проблемных зон (частые отмены, задержки). Генерация отчетов для руководства.

Задачи курсовой работы:

1. Анализ предметной области: исследование существующих решений на рынке. Определение функциональных и нефункциональных требований к системе.
2. Проектирование системы: разработка архитектуры (клиент-северная часть, база данных), выбор технологий (язык программирования, СУБД).
3. Реализация: написание кода для основных модулей (заказы, клиенты, курьеры, финансы). Создание интерфейсов для администратора, клиентов и курьеров.
4. Тестирование: проверка работоспособности системы на тестовых данных. Выявление и исправление ошибок.
5. Оценка эффективности: Анализ скорости обработки заказов, удобства интерфейса. Разработка рекомендаций по дальнейшему развитию (например, внедрение AI для прогнозирования спроса).

Ожидается, что итогом станет работоспособная система, способная существенно упростить процесс управления заказами, обеспечивая их быструю и точную обработку. За счет оптимизации маршрутов доставки предполагается значительное сокращение времени выполнения заказов, что напрямую отразится на эффективности логистики. Дополнительным преимуществом станет повышение уровня удовлетворенности клиентов, достигаемое благодаря прозрачности всех этапов заказа и удобству взаимодействия с сервисом. Кроме того, система должна автоматизировать рутинные операции, такие как учет и формирование отчетности, тем самым снижая нагрузку на персонал и минимизируя вероятность ошибок.

Система будет полезна для малых и средних служб доставки, стремящихся оптимизировать свои бизнес-процессы.

В итоге работы предполагается получение полностью функциональной информационной системы для управления заказами в службе доставки еды, которая будет способствовать оптимизации процесса работа администратора, повышению уровня обслуживания клиентов.

# ГЛАВА 2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

## Требования к программе

Для корректного функционирования программы требуется как минимум 120 МБ свободного пространства на жестком диске. Операционная система должна быть не ниже Microsoft Windows 7 (64-бит), включая Windows 8 и Windows 10. Программа должна обеспечивать стабильную работу при длительной эксплуатации. В случае сбоя аппаратного обеспечения должна быть предусмотрена возможность восстановления программы без потери данных, включая автоматическое резервное копирование.

Разработка программного продукта осуществлялась с использованием следующей конфигурации аппаратного и программного обеспечения:

* Среда разработки: Visual Studio 2019
* Язык программирования: C# версии 7.3
* Процессор: Intel(R) Pentium(R) CPU N4200 @ 1.10GHz 1.10 GHz
* Оперативная память: 4,00 ГБ (доступно: 3,84 ГБ)
* Операционная система: Windows 10 Pro 21H1

Данная конфигурация обеспечивала необходимую производительность для эффективной разработки, отладки и тестирования программы.

Минимально необходимая конфигурация пользователя:

* Операционная система: Windows 7, 8 или 10 (только 64-битные версии)
* Процессор: Intel Core 2 Duo или аналогичный с частотой от 1.8 ГГц
* ОЗУ: не менее 4 ГБ
* Свободное место на диске: не менее 1 ГБ
* Среда выполнения .NET: .NET Framework, совместимая с Visual Studio 2019 и C# 7.3

Такие технические параметры обеспечивают корректную установку, запуск и стабильную работу программы.

## 2.2 Руководство пользователя

Руководство пользователя по использованию консольного приложения системы управления заказами в службе доставки еды предназначено для ознакомления пользователя с возможностями системы и обучению основным действиям. Оно описывает интерфейс, структуру меню и последовательность действий при работе с программой.

Запуск приложения

Для запуска приложения откройте папку на диске C:\ (C:\Users\New\source\repos\ConsoleApp30\bin\Debug), в которую установлена программа, и дважды щелкните по файлу с расширением .exe. Это откроет консольное окно с главным меню (Рис. 1).

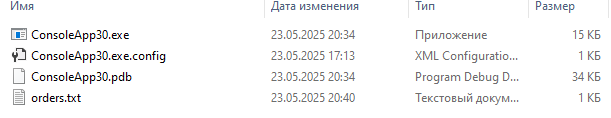


Рис. 1 Запуск приложения

Главное меню

После запуска программы пользователь увидит основное меню, которое отображает доступные функции приложения (Рис. 2). Выбор действия осуществляется путем ввода соответствующей цифры на клавиатуре.

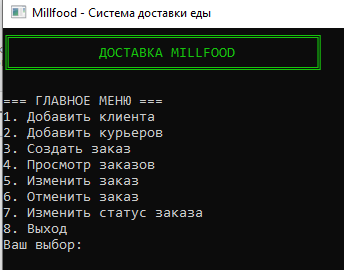


Рис. 2 Главное меню программы

Добавление клиента

Чтобы добавить нового клиента, выберите пункт 1. Введите ФИО и номер телефона клиента, после чего он будет добавлен в базу данных. Это действие отображается в консоли как подтверждение (Рис. 3).

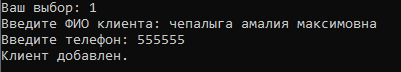


Рис. 3 Добавление клиента

Добавление курьера

Пункт 2 позволяет ввести имя нового курьера, после чего он будет добавлен в список доступных для распределения заказов (Рис. 4).

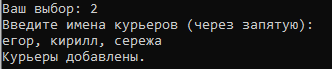


Рис. 4 Добавление курьера

Создание заказа

Выберите пункт 3, чтобы создать заказ. Программа предложит выбрать клиента, курьера, адрес, время (по готовности или к определенному времени), блюда, цена. После ввода данных заказ сохраняется, клиент получает бонусные баллы (Рис. 5)

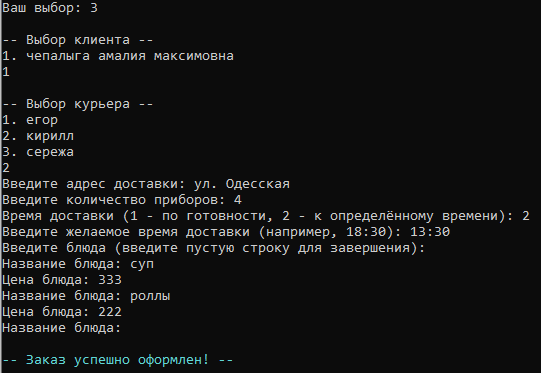


Рис. 5 Создание нового заказа

Просмотр заказов

Пункт 4 позволяет просмотреть список всех заказов с отображением клиента, блюда, цены, даты, статуса, курьера, время (Рис. 6).

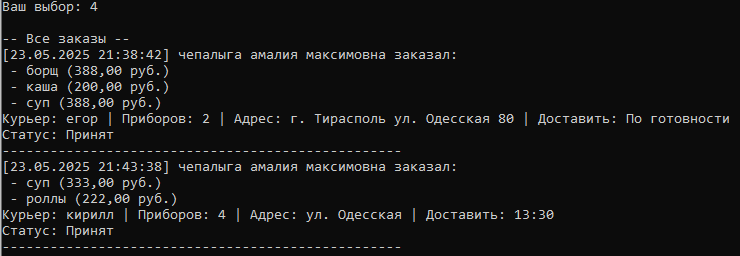


Рис. 6 Просмотр заказов

Изменение заказа

Выберите пункт 5 для изменения заказа (например, адрес, время и т.д.) (Рис. 7).

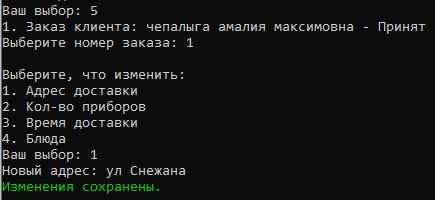


Рис. 7 Изменение заказа

Отмена заказа

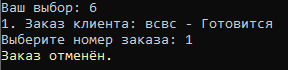
Пункт 6 позволяет выбрать заказ для его удаления из базы (рис. 8). 

Рис. 8 Отмена заказа

Изменение статуса заказа

Пункт 7 позволяет изменить статус заказа. Доступны статусы: принят, готовится, передан курьеру, доставлен (рис. 9).

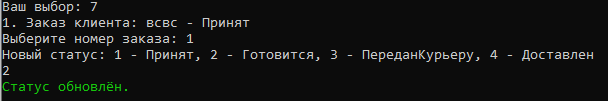


Рис. 9 Изменение статуса

Выход из программы

Выбор пункта 8 завершает работу приложения. Все заказы автоматически сохраняются в файл orders.txt при их создании, изменении или удалении. Структура файла упрощает последующую загрузку данных.

Файловое сохранение

Все заказы автоматически сохраняются в файл orders.txt при их создании, изменении или удалении. Структура файла упрощает последующую загрузку данных (Рис. 10).

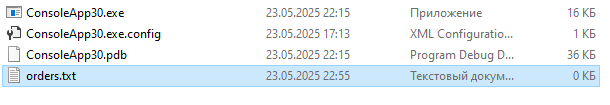


Рис. 10 Файловое сохранение

## Руководство программиста

Представленный программа является консольным приложением, реализующим систему для оформления и управления заказами на доставку еды. Программа позволяет добавлять клиентов и курьеров, оформлять заказы с индивидуальными параметрами, а также редактировать, просматривать, отменять и отслеживать заказы. Приложение написано на языке программирования C# с использованием стандартных библиотек .NET.

Структура приложения

Структура приложения включает в себя следующие основные компоненты:

* Главный класс Program - содержит логику пользовательского интерфейса и управления взаимодействием между сущностями.
* Класс Client - представляет клиентов.
* Класс Courier - описывает курьеров.
* Класс Order - описывает заказ. Перечисление OrderStatus - определяет статус заказа.

Используемые библиотеки

* System - базовая функциональность .NET.
* System.Collections.Generic - коллекции, такие как List<T>.
* System.Linq - функциональные методы работы с коллекциями.
* System.IO - работа с файлами.

Описание классов

1. Класс Client - описывает клиента, оформляющего заказ.

Свойства:

Name: тип string - ФИО клиента.

Phone: тип string - номер телефона клиента.

Назначение: используется при создании заказа и хранится внутри объекта Order.

1. Класс Courier - описывает курьера, который доставляет заказ.

Свойства:

Name: тип string - имя курьера.

Назначение: связывается с заказом для указания исполнителя доставки.

1. Класс Order - сновной класс, описывающий заказ клиента.

Свойства:

Client: тип Client - клиент, сделавший заказ.

Courier: тип Courier - курьер, доставляющий заказ.

Dishes: List<string> - список блюд в заказе.

Prices: List<decimal> - соответствующие цены блюд.

Address: string - адрес доставки.

Utensils: int - количество приборов.

DeliveryTime: string - время доставки (по готовности или конкретное время).

Status: OrderStatus — текущий статус заказа.

Date: DateTime — дата и время оформления заказа.

1. Перечисление OrderStatus - используется для отслеживания состояния заказа.

Функциональность интерфейса (Методы в классе Program)

Главное меню: добавить клиента, добавить курьеров, создать заказ, просмотреть все заказы, изменить заказ, отменить заказ, обновить статус заказа, выход.

Методы

1. Метод AddClient - позволяет ввести имя и телефон клиента, после чего добавляет объект Client в список клиентов.
2. Метод AddCouriers - позволяет добавить одного или несколько курьеров, имена которых вводятся через запятую.
3. Метод CreateOrder - создает заказ: выбор клиента из списка, выбор курьера, ввод адреса, количества приборов, времени доставки, ввод блюд и цен. Формирование объекта Order и сохранение его в список заказов. Вывод информации о заказе и сохранение в файл.
4. Метод ShowOrders - отображает список всех заказов с детализацией: клиент, курьер, блюда, стоимость, статус и прочее.
5. Метод EditOrder - позволяет изменить параметры существующего заказа: адрес доставки, количество приборов, время доставки, список блюд и цен.
6. Метод CancelOrder - удаляет заказ из списка заказов на основе выбора пользователя.
7. Метод UpdateOrderStatus - позволяет изменить статус заказа (принят, готовится, передан курьеру, доставлен).
8. Метод SaveOrdersToFile - сохраняет текущие заказы в файл orders.txt в текстовом формате. Структура строки: копировать, редактировать, форматы данных.

Файл orders.txt создается в корне директории и может быть открыт любым текстовым редактором.

Данное приложение демонстрирует принципы объектно-ориентированного программирования (ООП), такие как инкапсуляция и работа с коллекциями объектов. Оно представляет собой удобный инструмент для небольшого бизнеса по доставке еды и может быть расширено под конкретные нужды.

# ЗАКЛЮЧЕНИЕ

В данной курсовой работе была разработана информационная система для диспетчера службы доставки еды. Программа реализована на языке программирования C#, а в качестве хранилища данных используются три текстовых файла.

Пользовательский интерфейс создан с использованием консольных средств ввода-вывода, что обеспечивает простоту навигации и интуитивно понятные команды для работы с программой. Интерфейс позволяет удобно взаимодействовать с системой: вводить новые заказы, осуществлять поиск по базе и выводить необходимую информацию.

Информация о клиентах, курьерах и заказах организована отдельно и хранится в трёх текстовых файлах.

Основные преимущества разработанной системы включают в себя:

1. Интуитивно понятный интерфейс, не требующий длительного обучения, что делает систему удобной для повседневного использования.
2. Быстрое внесение и поиск информации, что важно для оперативной работы службы доставки.
3. В процессе разработки и тестирования были выявлены направления для дальнейшего улучшения системы:
4. Расширение функциональности, например, добавление возможности отслеживания статуса заказа в реальном времени.
5. Улучшение пользовательского интерфейса – переход от консольного приложения к графическому, что повысит удобство работы.
6. Интеграция с внешними сервисами и базами данных для получения более полной информации о клиентах и курьерах (адреса, контактные данные, история заказов и пр.).

Цель работы была успешно достигнута – создана информационная система, позволяющая эффективно управлять заказами в службе доставки еды.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Албахари Дж., Албахари Б.: C# 8.0. Карманный справочник - Москва: ООО «И. Д. Вильямс», 2020. – 240 с.
2. Антон Марченко: Основы программирования на С# 2.0. Учебное пособие – 2017. – 557 с.
3. Шилдт, Г. C# 4.0: полное руководство. - Москва: ООО «И. Д. Вильямс», 2015. - 1056 с.
4. Иванов А. В. Цифровая трансформация ресторанного бизнеса. – М.: Альпина Паблишер, 2021.)
5. «История доставки еды» https://en.wikipedia.org/wiki/Food\_delivery

**ПРИЛОЖЕНИЕ**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace ConsoleApp30

{

class Program

{

static List<Order> orders = new List<Order>();

static List<Client> clients = new List<Client>();

static List<Courier> couriers = new List<Courier>();

static void Main(string[] args)

{

Console.Title = "Millfood - Система доставки еды";

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("╔══════════════════════════════════════╗");

Console.WriteLine("║ ДОСТАВКА MILLFOOD ║");

Console.WriteLine("╚══════════════════════════════════════╝");

Console.ResetColor();

while (true)

{

Console.WriteLine("\n=== ГЛАВНОЕ МЕНЮ ===");

Console.WriteLine("1. Добавить клиента");

Console.WriteLine("2. Добавить курьеров");

Console.WriteLine("3. Создать заказ");

Console.WriteLine("4. Просмотр заказов");

Console.WriteLine("5. Изменить заказ");

Console.WriteLine("6. Отменить заказ");

Console.WriteLine("7. Изменить статус заказа");

Console.WriteLine("8. Выход");

Console.Write("Ваш выбор: ");

var input = Console.ReadLine();

switch (input)

{

case "1": AddClient(); break;

case "2": AddCouriers(); break;

case "3": CreateOrder(); break;

case "4": ShowOrders(); break;

case "5": EditOrder(); break;

case "6": CancelOrder(); break;

case "7": UpdateOrderStatus(); break;

case "8": return;

default:

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Неверный ввод.");

Console.ResetColor();

break;

}

}

}

static void AddClient()

{

Console.Write("Введите ФИО клиента: ");

string name = Console.ReadLine();

Console.Write("Введите телефон: ");

string phone = Console.ReadLine();

clients.Add(new Client { Name = name, Phone = phone });

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Клиент успешно добавлен.");

Console.ResetColor();

}

static void AddCouriers()

{

Console.Write("Введите имена курьеров (через запятую): ");

string input = Console.ReadLine();

var names = input.Split(',').Select(n => n.Trim());

foreach (var name in names)

couriers.Add(new Courier { Name = name });

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Курьеры добавлены.");

Console.ResetColor();

}

static void CreateOrder()

{

if (clients.Count == 0 || couriers.Count == 0)

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("Добавьте хотя бы одного клиента и курьера.");

Console.ResetColor();

return;

}

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine("\n-- Выбор клиента --");

Console.ResetColor();

for (int i = 0; i < clients.Count; i++)

Console.WriteLine($"{i + 1}. {clients[i].Name}");

int clientIndex = Convert.ToInt32(Console.ReadLine()) - 1;

var client = clients[clientIndex];

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine("\n-- Выбор курьера --");

Console.ResetColor();

for (int i = 0; i < couriers.Count; i++)

Console.WriteLine($"{i + 1}. {couriers[i].Name}");

int courierIndex = Convert.ToInt32(Console.ReadLine()) - 1;

var courier = couriers[courierIndex];

Console.Write("Введите адрес доставки: ");

string address = Console.ReadLine();

Console.Write("Введите количество приборов: ");

int utensils = Convert.ToInt32(Console.ReadLine());

Console.Write("Время доставки (1 — по готовности, 2 — к определённому времени): ");

int timeChoice = Convert.ToInt32(Console.ReadLine());

string deliveryTime;

if (timeChoice == 1)

deliveryTime = "По готовности";

else

{

Console.Write("Введите желаемое время доставки (например, 18:30): ");

deliveryTime = Console.ReadLine();

}

List<string> dishes = new List<string>();

List<decimal> prices = new List<decimal>();

Console.WriteLine("Введите блюда (введите пустую строку для завершения):");

while (true)

{

Console.Write("Название блюда: ");

string dish = Console.ReadLine();

if (string.IsNullOrWhiteSpace(dish)) break;

Console.Write("Цена блюда: ");

decimal price = Convert.ToDecimal(Console.ReadLine());

dishes.Add(dish);

prices.Add(price);

}

var order = new Order

{

Client = client,

Courier = courier,

Dishes = dishes,

Prices = prices,

Address = address,

Utensils = utensils,

DeliveryTime = deliveryTime,

Status = OrderStatus.Принят,

Date = DateTime.Now

};

orders.Add(order);

SaveOrdersToFile();

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("\n╔══════════════════════════════════════╗");

Console.WriteLine("║ ЗАКАЗ УСПЕШНО ОФОРМЛЕН ║");

Console.WriteLine("╚══════════════════════════════════════╝");

Console.ResetColor();

Console.WriteLine($"\nКлиент: {client.Name}");

Console.WriteLine($"Курьер: {courier.Name}");

Console.WriteLine($"Адрес доставки: {address}");

Console.WriteLine($"Количество приборов: {utensils}");

Console.WriteLine($"Время доставки: {deliveryTime}");

Console.WriteLine("Блюда:");

decimal total = 0;

for (int i = 0; i < dishes.Count; i++)

{

Console.WriteLine($" - {dishes[i]} — {prices[i]:0.00} руб.");

total += prices[i];

}

Console.WriteLine($"ИТОГО: {total:0.00} руб.");

}

static void ShowOrders()

{

if (orders.Count == 0)

{

Console.WriteLine("Нет заказов.");

return;

}

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine("\n-- Все заказы --");

Console.ResetColor();

foreach (var o in orders)

{

Console.WriteLine($"[{o.Date}] {o.Client.Name} заказал:");

for (int i = 0; i < o.Dishes.Count; i++)

Console.WriteLine($" - {o.Dishes[i]} ({o.Prices[i]:0.00} руб.)");

Console.WriteLine($"Курьер: {o.Courier.Name} | Приборов: {o.Utensils} | Адрес: {o.Address} | Доставить: {o.DeliveryTime}");

Console.WriteLine($"Статус: {o.Status}");

Console.WriteLine("--------------------------------------------------");

}

}

static void EditOrder()

{

if (orders.Count == 0)

{

Console.WriteLine("Нет заказов для изменения.");

return;

}

for (int i = 0; i < orders.Count; i++)

Console.WriteLine($"{i + 1}. Заказ клиента: {orders[i].Client.Name} — {orders[i].Status}");

Console.Write("Выберите номер заказа: ");

int index = Convert.ToInt32(Console.ReadLine()) - 1;

var order = orders[index];

Console.WriteLine("\nВыберите, что изменить:");

Console.WriteLine("1. Адрес доставки");

Console.WriteLine("2. Кол-во приборов");

Console.WriteLine("3. Время доставки");

Console.WriteLine("4. Блюда");

Console.Write("Ваш выбор: ");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

Console.Write("Новый адрес: ");

order.Address = Console.ReadLine();

break;

case "2":

Console.Write("Новое кол-во приборов: ");

order.Utensils = Convert.ToInt32(Console.ReadLine());

break;

case "3":

Console.Write("Новое время доставки: ");

order.DeliveryTime = Console.ReadLine();

break;

case "4":

order.Dishes.Clear();

order.Prices.Clear();

Console.WriteLine("Введите блюда заново (пустая строка — завершение):");

while (true)

{

Console.Write("Блюдо: ");

string d = Console.ReadLine();

if (string.IsNullOrWhiteSpace(d)) break;

Console.Write("Цена: ");

decimal p = Convert.ToDecimal(Console.ReadLine());

order.Dishes.Add(d);

order.Prices.Add(p);

}

break;

default:

Console.WriteLine("Неверный выбор.");

return;

}

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Изменения сохранены.");

Console.ResetColor();

SaveOrdersToFile();

}

static void CancelOrder()

{

if (orders.Count == 0)

{

Console.WriteLine("Нет заказов для отмены.");

return;

}

for (int i = 0; i < orders.Count; i++)

Console.WriteLine($"{i + 1}. Заказ клиента: {orders[i].Client.Name} — {orders[i].Status}");

Console.Write("Выберите номер заказа: ");

int index = Convert.ToInt32(Console.ReadLine()) - 1;

orders.RemoveAt(index);

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("Заказ отменён.");

Console.ResetColor();

SaveOrdersToFile();

}

static void UpdateOrderStatus()

{

if (orders.Count == 0)

{

Console.WriteLine("Нет заказов для обновления.");

return;

}

for (int i = 0; i < orders.Count; i++)

Console.WriteLine($"{i + 1}. Заказ клиента: {orders[i].Client.Name} — {orders[i].Status}");

Console.Write("Выберите номер заказа: ");

int index = Convert.ToInt32(Console.ReadLine()) - 1;

Console.WriteLine("Новый статус: 1 — Принят, 2 — Готовится, 3 — ПереданКурьеру, 4 — Доставлен");

int status = Convert.ToInt32(Console.ReadLine());

orders[index].Status = (OrderStatus)(status - 1);

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Статус обновлён.");

Console.ResetColor();

SaveOrdersToFile();

}

static void SaveOrdersToFile()

{

var lines = orders.Select(o =>

$"{o.Date}|{o.Client.Name}|{o.Courier.Name}|{string.Join(",", o.Dishes)}|" +

$"{string.Join(",", o.Prices)}|{o.Address}|{o.Utensils}|{o.DeliveryTime}|{o.Status}");

File.WriteAllLines("orders.txt", lines);

}

}

class Client

{

public string Name { get; set; }

public string Phone { get; set; }

}

class Courier

{

public string Name { get; set; }

}

class Order

{

public Client Client { get; set; }

public Courier Courier { get; set; }

public List<string> Dishes { get; set; }

public List<decimal> Prices { get; set; }

public string Address { get; set; }

public int Utensils { get; set; }

public string DeliveryTime { get; set; }

public OrderStatus Status { get; set; }

public DateTime Date { get; set; }

}

enum OrderStatus

{

Принят,

Готовится,

ПереданКурьеру,

Доставлен

}

}

# 