**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 7](#_Toc165050060)

[1.3 Функции, выполняемые менеджером заказов 9](#_Toc165050061)

[1.4 Постановка задачи 12](#_Toc165050062)

[ГЛАВА 2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ 14](#_Toc165050063)

[2.1 Требования к программе 14](#_Toc165050064)

[2.2 Руководство пользователя 15](#_Toc165050065)

[2.3 Руководство программиста 18](#_Toc165050066)

[ЗАКЛЮЧЕНИЕ 21](#_Toc165050067)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 22](#_Toc165050068)

[ПРИЛОЖЕНИЕ А 23](#_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 компании:

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

Отдел разработки и IT включает разработчиков программного обеспечения, системных администраторов, аналитиков данных и специалистов по кибербезопасности. Этот отдел отвечает за разработку и поддержку программного обеспечения, интеграцию с ресторанами-партнерами, обслуживание мобильных приложений и веб-платформы, защиту персональных данных клиентов, а также анализ больших данных для оптимизации бизнес-процессов.

Операционный отдел состоит из менеджеров по работе с ресторанами, координаторов заказов, диспетчеров и менеджеров по качеству. Основные функции этого подразделения — взаимодействие с ресторанами-партнерами, контроль выполнения заказов, решение конфликтных ситуаций и обеспечение высокого уровня качества предоставляемых услуг.

Логистический отдел включает менеджеров по логистике, координаторов курьеров и аналитиков маршрутов. Его задача — оптимизация маршрутов доставки, распределение заказов между курьерами, контроль сроков выполнения доставки и внедрение систем GPS-мониторинга для повышения эффективности работы.

Отдел маркетинга и продаж объединяет маркетологов, SMM-специалистов, менеджеров по продажам и копирайтеров. Главные задачи этого подразделения — разработка маркетинговых стратегий, продвижение бренда, управление социальными сетями, проведение рекламных кампаний и анализ деятельности конкурентов.

Клиентский сервис представлен операторами call-центра, менеджерами по работе с клиентами и специалистами по лояльности. Их функции включают консультирование клиентов, обработку жалоб и предложений, реализацию программ лояльности и поддержание обратной связи с клиентской базой.

Финансовый отдел состоит из бухгалтерии, финансовых аналитиков и специалистов по расчетам. Он отвечает за финансовое планирование, взаиморасчеты с партнерами, налоговую отчетность и анализ финансовых показателей, обеспечивая устойчивость и прозрачность финансовой деятельности компании.

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.

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).

Сотрудники отдела занимаются приёмом и обработкой 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, при необходимости оперативно вносятся изменения в заказы, включая замену блюд, изменение времени или адреса доставки. Параллельно осуществляется координация с ресторанами: передаются заказы, получаются подтверждения об их приёме, отслеживаются сроки приготовления, а возникающие сложности, такие как отсутствие некоторых блюд или задержки, решаются в кратчайшие сроки. Ведётся актуальная база данных ресторанов, включающая меню, график работы и контактную информацию ответственных лиц.

Организация доставки заказов осуществляется с учётом текущей загруженности и местоположения курьеров. Для сокращения времени ожидания клиентов оптимизируются маршруты, а передвижение курьеров отслеживается с помощью GPS-систем. Также решаются внештатные ситуации — транспортные затруднения, отмена заказов, непредвиденные задержки. С клиентами ведётся постоянная работа: предоставляется консультационная поддержка по вопросам меню, акций и условий доставки, на всех этапах оформления и выполнения заказа клиенты получают информацию о его статусе. Отдел обрабатывает жалобы, разрешает спорные ситуации, организует сбор обратной связи и развивает программы лояльности.

Финансовая часть работы включает контроль оплаты заказов — онлайн-платежей, наличных расчётов и корпоративных переводов. Проводится сверка финансовых данных с ресторанами и курьерами, рассчитываются бонусы и санкции. На регулярной основе формируются отчёты по выручке, количеству выполненных заказов и возвратам, проводится анализ ключевых показателей: среднего чека, времени доставки, уровня отмен. Важную роль играет работа с ИТ-системами — ведение базы заказов и клиентов в CRM, обновление информации о доступности блюд и акциях, а также взаимодействие с техническими специалистами в случае сбоев в работе сайта или приложений.

Контроль качества сервиса охватывает соблюдение стандартов обслуживания, включая своевременность и корректность ответов на обращения. Анализируются причины негативных отзывов, разрабатываются меры по их устранению. Кроме того, организуются обучающие программы и тренинги для новых сотрудников, включая менеджеров и курьеров. Аналитическая работа направлена на оптимизацию процессов: исследуется логистическая цепочка, выявляются слабые места, например, рестораны с частыми задержками, прогнозируются периоды повышенной нагрузки и разрабатываются меры реагирования. Также проводится тестирование новых функций, таких как 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**

Разработать программную систему для управления заказами в службе доставки еды, которая будет учитывать и обрабатывать информацию о клиентах, ресторанах-партнёрах, меню, заказах, курьерах и оплатах. Основной целью системы является упрощение процесса оформления и контроля доставки заказов, а также повышение эффективности работы службы доставки.

В системе должна храниться структурированная информация о клиентах с уникальным идентификатором, ФИО, контактными данными, адресом доставки и историей заказов. Аналогично для ресторанов-партнёров необходимо ввести данные с уникальным ключом, названием, адресом, контактной информацией и актуальным меню. Каждое блюдо будет иметь свой ключ, название, категорию, цену и принадлежность к определённому ресторану. Заказы должны содержать сведения о клиенте, списке выбранных блюд, ресторане, дате оформления, времени доставки, текущем статусе и назначенном курьере. Курьеры будут описаны с указанием уникального идентификатора, ФИО, контактных данных, транспортного средства и текущего статуса. Для оплаты будет предусмотрена информация о заказе, дате транзакции, сумме, методе и статусе оплаты.

Система должна обеспечивать управление обобщёнными списками всех ключевых сущностей: клиентов, ресторанов, меню, заказов, курьеров и оплат. Пользовательский интерфейс должен позволять выполнять основные операции — оформление новых заказов, назначение курьеров на конкретные заказы, а также изменение статуса заказа в процессе его обработки (в обработке, готовится, доставляется, выполнен).

Кроме того, система должна поддерживать выполнение типовых запросов: получение списка заказов за определённый период времени, отслеживание статуса доставки конкретного заказа, определение самых популярных блюд среди клиентов, подсчёт количества выполненных заказов каждым курьером и вычисление общей выручки за выбранный месяц.

# В итоге, программная система призвана автоматизировать ключевые процессы службы доставки еды, обеспечивая удобство управления заказами, прозрачность и оперативность работы с клиентами и курьерами, а также формирование отчетности для анализа деятельности.

# 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.

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).

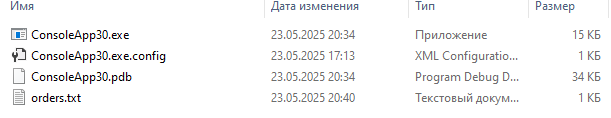


Рис. 1 Запуск приложения

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

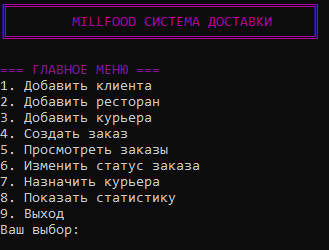


Рис. 2 Главное меню программы

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

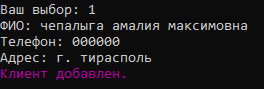


Рис. 3 Добавление клиента

Пункт 2 позволяет добавить рестораны, программа требует ввести название, адрес, контактные данные, блюда ресторана (Рис. 4).

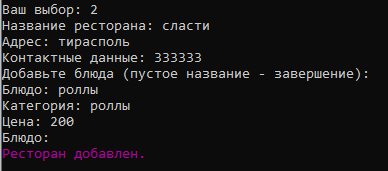


Рис. 4 Добавление ресторана

Пункт 3 позволяет ввести имя нового курьера, после чего он будет добавлен в список доступных для распределения заказов (Рис. 5).

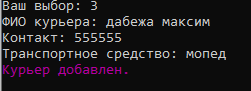


Рис. 5 Добавление курьера

Выберите пункт 4, чтобы создать заказ. Программа предложит выбрать клиента, ресторан, блюда, время доставки (Рис. 6).

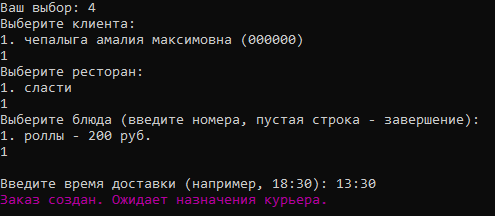


Рис. 6 Создание нового заказа

Пункт 5 позволяет просмотреть список всех заказов с отображением клиента, блюда, цены, даты, статуса, курьера, время, ресторан (Рис. 7).



Рис. 7 Просмотр заказов

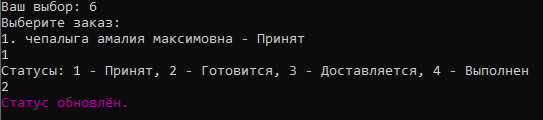
Выберите пункт 6 для изменения статуса заказа (Рис. 8). 

Рис. 8 Изменение статуса заказа

Пункт 7 позволяет назначить свободного курьера для доставки заказа (Рис.8)

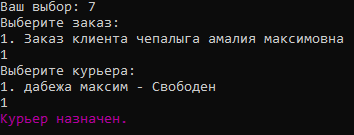


Рис. 8 Назначение курьера

Пункт 8 позволяет увидеть статистику заказов: количество выполненных заказов, популярные блюда, а также сколько было заказов на определенное блюдо (рис. 9).

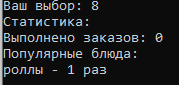


Рис. 9 Статистика

Выбор пункта 9 завершает работу приложения. Все заказы автоматически сохраняются в файл orders.txt при их создании, изменении или удалении. Структура файла упрощает последующую загрузку данных (Рис. 10).

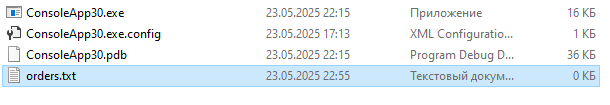


Рис. 10 Файловое сохранение

## Руководство программиста

Консольное приложение Millfood — это система для оформления, управления и отслеживания заказов на доставку еды. Программа написана на языке C# с использованием стандартных библиотек .NET и демонстрирует основные принципы объектно-ориентированного программирования, включая работу с объектами, инкапсуляцию и взаимодействие компонентов внутри системы.

Центральным элементом системы является главный класс Program. В нем сосредоточена логика взаимодействия с пользователем, отображение меню, а также обработка действий: добавление клиентов, ресторанов, курьеров, оформление заказов, обновление их статусов, назначение курьеров, просмотр статистики и выход из программы. В методе Main реализован цикл, отображающий главное меню и обрабатывающий ввод пользователя. В зависимости от выбранного действия вызываются соответствующие методы.

Класс Client представляет клиента, оформляющего заказ. Он содержит свойства Id, Name, Phone и Address, где хранится уникальный идентификатор клиента, его ФИО, номер телефона и адрес доставки соответственно. Метод AddClient реализует добавление нового клиента в систему. Пользователь вводит имя, номер телефона и адрес, после чего создается объект Client и добавляется в список клиентов.

Класс Restaurant представляет заведение, из которого осуществляется доставка еды. Он содержит свойства Id, Name, Address, ContactInfo и Menu. В них хранятся уникальный идентификатор ресторана, его название, адрес, контактная информация и меню, представленное списком объектов класса Dish. Метод AddRestaurant позволяет пользователю ввести основные данные ресторана и последовательно добавить блюда, каждое из которых включает название, категорию и цену. После ввода данных создается объект Restaurant, который добавляется в список ресторанов.

Класс Dish описывает блюдо, доступное для заказа. У каждого объекта есть Id, Name, Category и Price. Эти данные используются при создании меню ресторана и при выборе блюд пользователем в процессе оформления заказа.

Класс Courier представляет курьера, ответственного за доставку. Он содержит свойства Id, Name, ContactInfo, Vehicle и Status. Здесь хранятся данные о курьере: его идентификатор, имя, контактная информация, тип транспортного средства и текущий статус (свободен или занят). Метод AddCourier используется для добавления курьеров в систему. В процессе выполнения метода пользователь вводит данные о курьере, после чего создается объект и добавляется в список.

Класс Order представляет заказ и объединяет информацию о клиенте, ресторане, курьере (если он назначен), списке заказанных блюд, дате оформления, времени доставки и текущем статусе. Метод CreateOrder выполняет весь процесс оформления заказа. Пользователь выбирает клиента и ресторан, затем выбирает блюда из меню ресторана. После этого вводится желаемое время доставки, и создается объект Order, который добавляется в список заказов. Если курьер еще не назначен, заказ ожидает назначения.

Метод AssignCourier позволяет назначить курьера на заказ, если он еще не был назначен. Пользователь выбирает один из заказов без курьера и одного из доступных курьеров. После выбора курьер назначается, а его статус меняется на «занят». Это обеспечивает отслеживание занятости курьеров.

Метод UpdateOrderStatus предоставляет возможность изменить статус заказа. Пользователь выбирает заказ из общего списка, а затем указывает новый статус. Доступны следующие состояния: «Принят», «Готовится», «Доставляется» и «Выполнен». Изменение статуса позволяет отслеживать этапы выполнения заказа.

Метод ShowOrders отображает все оформленные заказы, включая дату оформления, имя клиента, название ресторана и текущий статус. Это позволяет пользователю быстро получить общее представление о состоянии заказов.

Метод ShowStatistics предоставляет аналитическую информацию о выполненных заказах и самых популярных блюдах. Сначала выводится общее количество выполненных заказов, затем анализируется частота встречаемости блюд в оформленных заказах. На основе этой информации выводятся наиболее заказываемые блюда.

Класс Payment представляет возможную модель для хранения информации об оплате, включая идентификатор, связанный заказ, дату, сумму, способ оплаты и статус. Хотя в текущей реализации класс не задействован, он может быть использован в будущих версиях приложения для реализации функциональности оплаты.

Перечисления OrderStatus и CourierStatus определяют возможные состояния заказов и курьеров соответственно. В OrderStatus предусмотрены состояния «Принят», «Готовится», «Доставляется» и «Выполнен», а в CourierStatus — «Свободен» и «Занят». Эти значения используются в соответствующих методах для обновления и отображения текущего состояния объектов.

Вся структура приложения построена на принципах модульности и инкапсуляции. Каждый класс отвечает за отдельную часть бизнес-логики, а методы обеспечивают взаимодействие между компонентами. Программа предоставляет удобный интерфейс для управления основными процессами доставки еды в рамках небольшого предприятия.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы была успешно разработана и реализована информационная система для управления заказами в службе доставки еды. Созданное консольное приложение на языке C# обеспечивает удобное и эффективное взаимодействие диспетчера с базой данных клиентов, курьеров и заказов, хранящейся в текстовых файлах.

Разработанная система демонстрирует высокую простоту использования благодаря интуитивно понятному интерфейсу и позволяет быстро вводить, искать и обновлять необходимую информацию, что значительно облегчает повседневные задачи службы доставки. В процессе тестирования были выявлены перспективные направления для дальнейшего развития, такие как внедрение функционала отслеживания статуса заказа в реальном времени, переход к более современным графическим интерфейсам и интеграция с внешними сервисами для расширения возможностей системы.

Таким образом, поставленная цель была достигнута: создано программное решение, способное оптимизировать процесс управления заказами и повысить качество обслуживания клиентов. Полученный опыт и результаты работы послужат хорошей основой для дальнейшего совершенствования и масштабирования системы в соответствии с потребностями бизнеса.

# 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 А**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text.Json;

using System.Text.Json.Serialization;

using System.Text.Encodings.Web;

namespace FoodDeliverySystem

{

class Program

{

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

static List<Restaurant> restaurants = new List<Restaurant>();

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

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

static List<Payment> payments = new List<Payment>();

static void Main(string[] args)

{

LoadOrdersFromFile();

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

Console.ForegroundColor = ConsoleColor.Magenta;

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

Console.WriteLine("║ MILLFOOD СИСТЕМА ДОСТАВКИ ║");

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

Console.ResetColor();

while (true)

{

Console.ForegroundColor = ConsoleColor.DarkMagenta;

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

Console.ResetColor();

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

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

Console.WriteLine("3. Добавить курьера");

Console.WriteLine("4. Создать заказ");

Console.WriteLine("5. Просмотреть заказы");

Console.WriteLine("6. Изменить статус заказа");

Console.WriteLine("7. Назначить курьера");

Console.WriteLine("8. Показать статистику");

Console.WriteLine("9. Выход");

Console.Write("Ваш выбор: ");

var choice = Console.ReadLine();

switch (choice)

{

case "1": AddClient(); break;

case "2": AddRestaurant(); break;

case "3": AddCourier(); break;

case "4": CreateOrder(); break;

case "5": ShowOrders(); break;

case "6": UpdateOrderStatus(); break;

case "7": AssignCourier(); break;

case "8": ShowStatistics(); break;

case "9": return;

default:

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Неверный ввод.");

Console.ResetColor();

break;

}

}

}

static void AddClient()

{

Console.Write("ФИО: ");

string name = Console.ReadLine();

Console.Write("Телефон: ");

string phone = Console.ReadLine();

Console.Write("Адрес: ");

string address = Console.ReadLine();

clients.Add(new Client { Id = Guid.NewGuid(), Name = name, Phone = phone, Address = address });

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Клиент добавлен.");

Console.ResetColor();

}

static void AddRestaurant()

{

Console.Write("Название ресторана: ");

string name = Console.ReadLine();

Console.Write("Адрес: ");

string address = Console.ReadLine();

Console.Write("Контактные данные: ");

string contact = Console.ReadLine();

var menu = new List<Dish>();

Console.WriteLine("Добавьте блюда (пустое название — завершение):");

while (true)

{

Console.Write("Блюдо: ");

string dishName = Console.ReadLine();

if (string.IsNullOrWhiteSpace(dishName)) break;

Console.Write("Категория: ");

string category = Console.ReadLine();

Console.Write("Цена: ");

decimal price = decimal.Parse(Console.ReadLine());

menu.Add(new Dish { Id = Guid.NewGuid(), Name = dishName, Category = category, Price = price });

}

restaurants.Add(new Restaurant { Id = Guid.NewGuid(), Name = name, Address = address, ContactInfo = contact, Menu = menu });

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Ресторан добавлен.");

Console.ResetColor();

}

static void AddCourier()

{

Console.Write("ФИО курьера: ");

string name = Console.ReadLine();

Console.Write("Контакт: ");

string contact = Console.ReadLine();

Console.Write("Транспортное средство: ");

string vehicle = Console.ReadLine();

couriers.Add(new Courier { Id = Guid.NewGuid(), Name = name, ContactInfo = contact, Vehicle = vehicle, Status = CourierStatus.Свободен });

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Курьер добавлен.");

Console.ResetColor();

}

static void CreateOrder()

{

if (!clients.Any() || !restaurants.Any())

{

Console.WriteLine("Необходимо добавить клиентов и рестораны перед оформлением заказа.");

return;

}

Console.WriteLine("Выберите клиента:");

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

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

int clientIndex = int.Parse(Console.ReadLine()) - 1;

var client = clients[clientIndex];

Console.WriteLine("Выберите ресторан:");

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

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

int restIndex = int.Parse(Console.ReadLine()) - 1;

var restaurant = restaurants[restIndex];

var selectedDishes = new List<Dish>();

Console.WriteLine("Выберите блюда (введите номера, пустая строка — завершение):");

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

Console.WriteLine($"{i + 1}. {restaurant.Menu[i].Name} — {restaurant.Menu[i].Price} руб.");

while (true)

{

string dishInput = Console.ReadLine();

if (string.IsNullOrWhiteSpace(dishInput)) break;

int dishIndex = int.Parse(dishInput) - 1;

selectedDishes.Add(restaurant.Menu[dishIndex]);

}

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

string deliveryTime = Console.ReadLine();

var order = new Order

{

Id = Guid.NewGuid(),

Client = client,

Restaurant = restaurant,

Dishes = selectedDishes,

Date = DateTime.Now,

DeliveryTime = deliveryTime,

Status = OrderStatus.Принят

};

orders.Add(order);

SaveOrdersToFile();

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Заказ создан. Ожидает назначения курьера.");

Console.ResetColor();

}

static void AssignCourier()

{

var pendingOrders = orders.Where(o => o.Courier == null).ToList();

if (!pendingOrders.Any())

{

Console.WriteLine("Нет заказов без курьера.");

return;

}

Console.WriteLine("Выберите заказ:");

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

Console.WriteLine($"{i + 1}. Заказ клиента {pendingOrders[i].Client.Name}");

int orderIndex = int.Parse(Console.ReadLine()) - 1;

Console.WriteLine("Выберите курьера:");

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

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

int courierIndex = int.Parse(Console.ReadLine()) - 1;

pendingOrders[orderIndex].Courier = couriers[courierIndex];

couriers[courierIndex].Status = CourierStatus.Занят;

SaveOrdersToFile();

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Курьер назначен.");

Console.ResetColor();

}

static void UpdateOrderStatus()

{

if (!orders.Any())

{

Console.WriteLine("Нет заказов.");

return;

}

Console.WriteLine("Выберите заказ:");

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

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

int index = int.Parse(Console.ReadLine()) - 1;

Console.WriteLine("Статусы: 1 — Принят, 2 — Готовится, 3 — Доставляется, 4 — Выполнен");

int status = int.Parse(Console.ReadLine());

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

SaveOrdersToFile();

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Статус обновлён.");

Console.ResetColor();

}

static void ShowOrders()

{

foreach (var o in orders)

{

Console.WriteLine($"{o.Date}: Клиент {o.Client.Name}, Ресторан: {o.Restaurant.Name}, Статус: {o.Status}");

}

}

static void ShowStatistics()

{

Console.WriteLine("Статистика:");

int completed = orders.Count(o => o.Status == OrderStatus.Выполнен);

Console.WriteLine($"Выполнено заказов: {completed}");

var topDishes = orders.SelectMany(o => o.Dishes)

.GroupBy(d => d.Name)

.OrderByDescending(g => g.Count())

.Take(3);

Console.WriteLine("Популярные блюда:");

foreach (var dish in topDishes)

Console.WriteLine($"{dish.Key} — {dish.Count()} раз");

}

static void SaveOrdersToFile()

{

var options = new JsonSerializerOptions

{

WriteIndented = true,

ReferenceHandler = ReferenceHandler.IgnoreCycles,

Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping

};

string json = JsonSerializer.Serialize(orders, options);

File.WriteAllText("orders.txt", json);

}

static void LoadOrdersFromFile()

{

if (File.Exists("orders.txt"))

{

var options = new JsonSerializerOptions

{

PropertyNameCaseInsensitive = true,

ReferenceHandler = ReferenceHandler.IgnoreCycles

};

string json = File.ReadAllText("orders.txt");

orders = JsonSerializer.Deserialize<List<Order>>(json, options) ?? new List<Order>();

}

}

}

class Client

{

public Guid Id { get; set; }

public string Name { get; set; }

public string Phone { get; set; }

public string Address { get; set; }

}

class Restaurant

{

public Guid Id { get; set; }

public string Name { get; set; }

public string Address { get; set; }

public string ContactInfo { get; set; }

public List<Dish> Menu { get; set; }

}

class Dish

{

public Guid Id { get; set; }

public string Name { get; set; }

public string Category { get; set; }

public decimal Price { get; set; }

}

class Order

{

public Guid Id { get; set; }

public Client Client { get; set; }

public Restaurant Restaurant { get; set; }

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

public Courier Courier { get; set; }

public DateTime Date { get; set; }

public string DeliveryTime { get; set; }

public OrderStatus Status { get; set; }

}

class Courier

{

public Guid Id { get; set; }

public string Name { get; set; }

public string ContactInfo { get; set; }

public string Vehicle { get; set; }

public CourierStatus Status { get; set; }

}

class Payment

{

public Guid Id { get; set; }

public Order Order { get; set; }

public DateTime Date { get; set; }

public decimal Amount { get; set; }

public string Method { get; set; }

public string Status { get; set; }

}

enum OrderStatus { Принят, Готовится, Доставляется, Выполнен }

enum CourierStatus { Свободен, Занят }

}

# 