ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 4](#_Toc200357244)

[ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА ЗАДАЧ 6](#_Toc200357245)

[1.1 История и развитие ресторанного бизнеса 6](#_Toc200357246)

[1.2 Организационная структура ресторана 9](#_Toc200357247)

[1.3 Функции администратора ресторана 13](#_Toc200357248)

[1.4 Постановка задачи 16](#_Toc200357249)

[ГЛАВА 2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ 18](#_Toc200357250)

[2.1 Требования к программе 18](#_Toc200357251)

[2.2 Руководство пользователя 19](#_Toc200357252)

[2.3 Руководство программиста. 22](#_Toc200357253)

[ЗАКЛЮЧЕНИЕ 29](#_Toc200357254)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 31](#_Toc200357255)

[ПРИЛОЖЕНИЕ А 32](#_Toc200357256)

# ****ВВЕДЕНИЕ****

В современном мире автоматизация бизнес-процессов играет ключевую роль в повышении эффективности работы предприятий, особенно в таких динамичных отраслях, как ресторанный бизнес. Внедрение информационных систем позволяет оптимизировать управление заказами, бронированием столиков, учетом клиентов и персонала, а также финансовыми операциями, минимизируя человеческий фактор, ускоряя обработку данных и повышая качество обслуживания клиентов. Разработка программной системы для управления рестораном является актуальной задачей, так как она обеспечивает администратору инструменты для эффективной координации работы заведения, точного учета ресурсов и улучшения клиентского опыта. Такие решения способствуют оптимизации внутренних процессов, повышению прозрачности операций и укреплению конкурентных позиций ресторана на рынке.

Целью курсовой работы является разработка программной системы для управления рестораном, обеспечивающей автоматизацию ключевых процессов. Для достижения этой цели была глубоко изучена предметная область ресторанного бизнеса, проанализированы основные бизнес-процессы, а также реализована функциональность, охватывающая бронирование столиков, учет клиентов, официантов, блюд и заказов, анализ данных о популярности блюд, среднем чеке и выручке, управление персоналом и обеспечение безопасности данных. Система реализована в виде консольного приложения с интуитивно понятным интерфейсом, прошла тестирование и сопровождается подробной документацией, что делает её готовой к использованию и дальнейшему развитию. [1 c. 30]

Объектом исследования является ресторанный бизнес, а предметом — процесс разработки программной системы для его управления. В ходе работы применялись методы анализа для изучения предметной области, синтеза для создания системы и формализации для проектирования структуры данных и алгоритмов. Разработанная система предоставляет администратору ресторана эффективный инструмент для управления процессами, повышения производительности труда, улучшения качества обслуживания клиентов и обеспечения устойчивого развития заведения. [2 c. 30]

# ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА ЗАДАЧ

## **История и развитие ресторанного бизнеса**

Ресторанный бизнес является из старейших отраслей, тесно связанных с культурой, экономикой и образом жизни людей. Его истоки уходят в глубокую древность, когда первые одной формы общественного питания начали формироваться для удовлетворения потребностей путешественников, торговцев и паломников. Сегодня ресторанный бизнес — это глобальная индустрия, которая не только обеспечивает людей едой, но и создаёт уникальный клиентский опыт, стимулирует туризм и вносит значительный вклад в экономику многих стран.

Первые прототипы ресторанов появились ещё в древних цивилизациях. В Древнем Египте (около 2000 г. до н. э.) существовали заведения, где путникам предлагали пищу и напитки. В Древнем Риме таверны (popinae) и харчевни (thermopolia) были популярны среди горожан: здесь подавали горячие блюда, вино и закуски, а клиенты могли есть за стойкой или за столами. Эти заведения уже имели элементы современного ресторана: фиксированное меню, обслуживание и оплата за еду. [3 c. 30]

В Китае в период династии Тан (618–907 гг. н. э.) в крупных городах, таких как Чанъань, начали появляться чайные дома и закусочные, где подавали лапшу, пельмени и другие блюда. Эти заведения стали местом встреч для торговцев и чиновников, что подчеркивает социальную роль общественного питания.

На Востоке, в Средней Азии и на Ближнем Востоке, караван-сараи (с X века) предоставляли не только ночлег, но и питание для купцов, путешествующих по Великому шёлковому пути. Такие заведения были важными узлами торговли и культурного обмена, где формировались традиции гостеприимства.

В Средние века в Европе (V–XV века) ресторанный бизнес развивался в виде постоялых дворов, трактиров и таверн. Эти заведения обслуживали путников, паломников и местных жителей. Постоялые дворы, расположенные вдоль торговых путей, предлагали еду, ночлег и смену лошадей. Трактиры, популярные в городах, специализировались на питании и напитках, а таверны часто были местом для встреч и развлечений.

В монастырях Европы организовывались «госпиции» для паломников, где предоставляли бесплатную еду. Со временем некоторые из них начали взимать плату, что стало шагом к коммерциализации. В Англии XII века таверны (alehouses) начали предлагать простые блюда, такие как хлеб, сыр и мясные пироги, что заложило основу для развития гастрономической культуры.

Современная концепция ресторана зародилась во Франции в XVIII веке. Слово «ресторан» происходит от французского restaurer («восстанавливать»), так как первые рестораны позиционировались как заведения, где можно восстановить силы с помощью питательной еды. В 1765 году в Париже Буланже открыл заведение, где подавали супы и бульоны, названные restaurants (восстанавливающие). Это считается отправной точкой ресторанного бизнеса в современном понимании.

К концу XVIII века в Париже начали появляться рестораны с фиксированным меню, индивидуальным обслуживанием и возможностью выбора блюд. Революция во Франции (1789–1799) сыграла ключевую роль: многие повара, ранее работавшие в аристократических домах, открыли собственные заведения, что привело к расцвету гастрономии. В 1782 году Антуан Бовилье основал ресторан La Grande Taverne de Londres, который считается первым рестораном высокого класса с элегантным интерьером и профессиональным обслуживанием. [4 c. 30]

В XIX веке ресторанный бизнес начал активно развиваться в Европе и Америке благодаря индустриализации, росту городов и развитию транспорта. Железные дороги и пароходы сделали путешествия доступнее, что увеличило спрос на общественное питание. В крупных городах, таких как Лондон, Париж и Нью-Йорк, рестораны стали центрами социальной жизни, где собирались представители буржуазии, интеллигенции и элиты.

В США в XIX веке появились diner (закусочные) и saloons, обслуживавшие рабочих и путешественников. В 1831 году в Нью-Йорке открылся ресторан Delmonico’s, который стал эталоном американской высокой кухни, предлагая меню с десятками блюд и винную карту.

В России ресторанная культура начала формироваться в XVIII веке. Трактиры, известные с XII века, к XIX веку трансформировались в рестораны, особенно в Москве и Санкт-Петербурге. Например, ресторан Яръ в Москве (основан в 1826 году) стал символом роскоши, привлекая аристократов и купцов. К началу XX века в России насчитывалось около 6000 трактиров и ресторанов, большинство из которых были частными.

XX век стал переломным для ресторанного бизнеса. Развитие транспорта (автомобили, авиация) и туризма увеличило мобильность населения, что стимулировало открытие новых заведений. В США в 1920-х годах начали появляться сетевые рестораны, такие как White Castle (1921), которые предлагали стандартизированное меню и быстрое обслуживание.

После Второй мировой войны ресторанный бизнес пережил бум благодаря росту среднего класса и увеличению доходов населения. В 1955 году открылся первый McDonald’s, который стал символом фастфуда и глобализации ресторанной индустрии. Сети быстрого питания (Burger King, KFC, Starbucks) начали экспансию по всему миру, внедряя стандарты управления, автоматизацию и массовое производство.

В СССР ресторанный бизнес был частью государственной системы общепита. В 1930-е годы создавались столовые и рестораны при предприятиях, а в крупных городах открывались элитные заведения, такие как Метрополь и Арагви в Москве. К 1980-м годам в СССР насчитывалось около 80 000 предприятий общественного питания, обслуживавших миллионы людей.

В XXI веке ресторанный бизнес стремительно эволюционирует под влиянием технологий, глобализации и изменения потребительских предпочтений. Внедрение POS-систем (Point of Sale) обеспечивает эффективное управление заказами и финансами, а онлайн-бронирование через платформы, такие как OpenTable и BookTable, упрощает доступ клиентов к услугам. Мобильные приложения для заказа и доставки, включая Uber Eats и DoorDash, стали неотъемлемой частью индустрии, а искусственный интеллект позволяет анализировать предпочтения клиентов и прогнозировать спрос. Экологичность и устойчивое развитие приобретают всё большее значение: рестораны используют местные и органические продукты, минимизируют пищевые отходы в рамках концепции zero waste и переходят на экологичную упаковку для доставки.

Персонализация клиентского опыта выходит на первый план, предлагая индивидуальные меню с учётом диетических предпочтений, таких как веганские или безглютеновые блюда, и создавая уникальную атмосферу через тематические интерьеры и мероприятия.

Пандемия COVID-19 (2020–2021) значительно ускорила развитие доставки еды, и, по данным Statista, в 2023 году мировой рынок доставки достиг 600 миллиардов долларов, что также привело к появлению «тёмных кухонь» (cloud kitchens), работающих исключительно на доставку. Гастрономическое разнообразие продолжает расширяться благодаря популяризации этнических кухонь, таких как японская, мексиканская и индийская, а также росту интереса к высокой кухне (fine dining) и ресторанам, отмеченным звёздами Michelin, что отражает стремление клиентов к новым гастрономическим впечатлениям.

## **1.2 Организационная структура ресторана**

Организационная структура ресторана определяет распределение функций, ответственности и взаимодействие между сотрудниками, обеспечивая эффективное управление заведением. Она зависит от масштаба ресторана, его концепции, целевой аудитории и уровня сервиса. Структура ресторана формируется с учетом потребностей бизнеса, таких как обслуживание клиентов, управление персоналом, контроль финансов и обеспечение качества продукции. Разнообразие подходов к организации позволяет ресторанам адаптироваться к рыночным условиям, оптимизировать процессы и повышать конкурентоспособность.

Организационная структура ресторана включает три ключевых элемента: управление, правила и распределение труда. Управление осуществляется руководящим звеном, которое принимает стратегические и операционные решения. В небольших заведениях эту роль выполняет владелец или администратор, в крупных ресторанах — команда менеджеров, включая управляющего, финансового директора и шеф-повара.

Правила представляют собой внутренние стандарты, регулирующие процессы: рецептуры блюд, протоколы обслуживания, расписание смен, санитарные нормы. Распределение труда определяет, кто и какие задачи выполняет, обеспечивая разделение функций между кухней, залом, администрацией и вспомогательным персоналом. Например, официанты взаимодействуют с клиентами, повара готовят блюда, а администратор координирует работу всех подразделений.

Эффективная структура минимизирует дублирование функций, ускоряет принятие решений и улучшает коммуникацию. В ресторанах с высокой проходимостью, таких как сетевые заведения, чёткое распределение ролей позволяет справляться с большим потоком клиентов, тогда как в небольших семейных кафе структура может быть более гибкой, но менее формализованной. [4 c. 30]

Рестораны используют различные типы организационных структур в зависимости от их размера и целей. Основные подходы включают линейную, функциональную и линейно-функциональную структуры, каждая из которых имеет свои преимущества и ограничения.

Линейная структура характерна для небольших ресторанов, таких как кафе или семейные заведения с численностью персонала до 10–15 человек. В такой структуре администратор или владелец напрямую управляет всеми сотрудниками: поварами, официантами, уборщиками. Преимуществами являются простота коммуникации, быстрые решения и низкие управленческие расходы. Однако линейная структура ограничена в масштабируемости: при росте бизнеса администратор может не справляться с увеличившимся объёмом задач, что приводит к перегрузке и снижению эффективности. Например, в небольшом кафе с пятью столиками владелец может одновременно выполнять функции администратора, кассира и закупщика, но в ресторане с 50 столиками это становится невозможным.

Функциональная структура применяется в крупных ресторанах или сетевых заведениях, где задачи разделены по функциональным направлениям: кухня, зал, бухгалтерия, маркетинг, логистика. Каждое направление имеет своего руководителя (например, шеф-повар, менеджер зала, финансовый менеджер), который подчиняется управляющему. Такая структура повышает специализацию и эффективность, так как сотрудники сосредоточены на конкретных задачах. Например, в ресторане White Rabbit (Москва) кухня управляется шеф-поваром, а зал — менеджером по сервису, что позволяет поддерживать высокий уровень качества. Недостатком является сложность координации между подразделениями, что может привести к конфликтам или задержкам в коммуникации. [5 c. 30]

Линейно-функциональная структура сочетает преимущества двух предыдущих подходов и широко используется в средних и крупных ресторанах. Управляющий осуществляет общее руководство, а функциональные руководители отвечают за свои направления, но при этом подчиняются единому центру. Например, в сети Dodo Pizza управляющий координирует работу пиццерий, а локальные менеджеры отвечают за кухню, доставку и сервис. Эта структура обеспечивает баланс между централизованным контролем и специализацией, но требует чёткой регламентации процессов, чтобы избежать бюрократии. В ресторанах с такой структурой часто внедряются информационные системы, такие как iiko или rPOS, для автоматизации управления и обмена данными между подразделениями.

Основные службы ресторана

Работа ресторана строится на взаимодействии двух ключевых служб: фронт-офиса и бэк-офиса, каждая из которых выполняет свои функции.

Фронт-офис включает сотрудников, непосредственно взаимодействующих с клиентами. Основные роли:

Хостес: встречает гостей, провожает к столикам, управляет бронированием. В ресторанах с высокой загрузкой хостес использует системы бронирования для оптимизации размещения.

Официанты: принимают заказы, обслуживают гостей, передают заказы на кухню. В некоторых заведениях официанты также выполняют функции сомелье или бариста.

Администратор (менеджер зала): контролирует работу фронт-офиса, решает конфликтные ситуации, следит за качеством сервиса. В крупных ресторанах администратор также анализирует отзывы клиентов и координирует маркетинговые акции.

Бармены: готовят напитки, обслуживают барную зону, поддерживают ассортимент бара.

Бэк-офис отвечает за внутренние процессы, невидимые для гостей. Основные роли:

Повара: готовят блюда в соответствии с рецептурами, контролируют качество продуктов. Шеф-повар разрабатывает меню, обучает команду и следит за стандартами.

Закупщики и логисты: обеспечивают поставки продуктов, управляют складом, контролируют сроки годности. В сетевых ресторанах логистика централизована для оптимизации затрат.

Бухгалтеры: ведут финансовый учёт, рассчитывают зарплаты, анализируют доходы и расходы. В небольших ресторанах эти функции может выполнять администратор.

Уборщики и технический персонал: поддерживают чистоту в зале, на кухне и в подсобных помещениях, обеспечивают работу оборудования.

Организационная структура ресторана является основой его успешной работы, обеспечивая координацию процессов, разделение функций и эффективное управление. Линейная, функциональная и линейно-функциональная структуры адаптируются под масштаб и цели заведения, а взаимодействие фронт- и бэк-офиса гарантирует качество обслуживания. Современные информационные системы, такие как разработанная в данной работе, усиливают эффективность структуры, автоматизируя ключевые процессы и предоставляя администратору инструменты для контроля и аналитики. [6 c. 30]

## **1.3 Функции администратора ресторана**

Администратор ресторана является ключевой фигурой, обеспечивающей бесперебойную работу заведения и координацию всех бизнес-процессов. Его роль охватывает управление персоналом, контроль качества обслуживания, финансовый учёт, взаимодействие с клиентами и внедрение технологических решений. Функции администратора направлены на достижение высокой эффективности работы ресторана, повышение удовлетворённости гостей и обеспечение конкурентоспособности заведения на рынке. В условиях современного ресторанного бизнеса, где технологии играют всё большую роль, администратор должен сочетать организаторские навыки с умением использовать информационные системы для оптимизации процессов.

Одной из основных функций администратора является управление персоналом, включающее планирование, обучение и контроль. Администратор составляет расписание смен, распределяя официантов, поваров, хостес и уборщиков таким образом, чтобы обеспечить равномерную загрузку и избежать простоев. Например, в ресторане с высокой проходимостью, таком как McDonald’s, администратор рассчитывает количество сотрудников на пиковые часы, чтобы минимизировать время ожидания клиентов. Обучение персонала также входит в обязанности: администратор проводит инструктажи по стандартам обслуживания, санитарным нормам и работе с оборудованием. В авторских ресторанах, таких как White Rabbit (Москва), администратор может организовывать тренинги по презентации блюд или винной карте.

Контроль работы персонала предполагает мониторинг выполнения задач и соблюдения дисциплины. Администратор следит за тем, чтобы официанты оперативно обслуживали гостей, повара соблюдали рецептуры, а хостес корректно управляли бронированием. В случае конфликтов между сотрудниками администратор выступает медиатором, решая споры и поддерживая рабочую атмосферу. В крупных ресторанах, таких как Twins Garden, администратор использует системы управления персоналом, например, Bitrix24 или iiko, для отслеживания продуктивности и обратной связи.

Администратор отвечает за организацию бронирования столиков и размещение гостей, что особенно важно в ресторанах с высокой посещаемостью. Он принимает заявки на бронирование по телефону, через онлайн-платформы (OpenTable, BookTable) или мобильные приложения, а также управляет доступностью столиков в реальном времени. Например, в ресторане на 50 столиков администратор должен учитывать количество мест, время пребывания гостей и особые пожелания, такие как размещение у окна или в отдельном зале.

Внедрение информационных систем, таких как разработанная в данной работе, позволяет автоматизировать этот процесс, минимизируя ошибки и пересечения бронирований. [7 c. 30]

Размещение гостей требует от администратора навыков тайм-менеджмента и клиентоориентированности. Он координирует работу хостес, чтобы гости были оперативно рассажены, а также решает нестандартные ситуации, например, когда клиент приходит без брони в переполненный зал. В ресторанах высокой кухни, таких как Noma (Копенгаген), администратор может лично приветствовать VIP-гостей, создавая индивидуальный опыт. Кроме того, администратор анализирует данные о бронированиях, чтобы прогнозировать загрузку и оптимизировать использование залов.

Администратор играет важную роль в финансовом управлении рестораном, контролируя доходы, расходы и рентабельность. Он отслеживает выручку, анализирует средний чек и выявляет популярные блюда, чтобы оптимизировать меню. Например, в сети Dodo Pizza администраторы используют аналитические отчёты для определения блюд с низкой маржой, что позволяет корректировать закупки. В небольших ресторанах администратор может самостоятельно вести учёт кассовых операций, тогда как в крупных заведениях он взаимодействует с бухгалтерией, предоставляя данные о продажах и затратах.

Аналитика клиентских данных также входит в обязанности администратора. Он изучает предпочтения гостей, частоту посещений и суммы заказов, чтобы разрабатывать маркетинговые акции и программы лояльности. Например, в российских ресторанах, таких как Теремок, администраторы используют системы лояльности для персонализации предложений, что увеличивает повторные визиты. Информационные системы, подобные описанной в данной работе, автоматизируют сбор и анализ данных, предоставляя администратору готовые отчёты о выручке, среднем чеке и клиентской активности.

Администратор отвечает за соблюдение санитарных, противопожарных и юридических норм. Он организует регулярные проверки оборудования, следит за чистотой помещений и контролирует сроки годности продуктов. В России, согласно требованиям Роспотребнадзора, рестораны обязаны вести журналы санитарного контроля, и администратор обеспечивает их заполнение. Например, в сети KFC администраторы используют цифровые чек-листы для ежедневных проверок, что упрощает соблюдение норм.

Безопасность данных клиентов также входит в зону ответственности администратора. Он следит за защитой информации о бронированиях, заказах и платежах, особенно при использовании онлайн-платформ. Информационные системы, такие как разработанная в данной работе, включают механизмы шифрования и резервного копирования, что минимизирует риски утечек данных.

Функции администратора ресторана охватывают управление персоналом, бронированием, качеством обслуживания, финансами, заказами, безопасностью и технологиями. Эта роль требует сочетания организаторских, аналитических и технических навыков, а также умения адаптироваться к динамичным условиям ресторанного бизнеса. Информационные системы, подобные разработанной в данной работе, значительно упрощают выполнение этих функций, автоматизируя рутинные процессы и предоставляя данные для принятия решений. Эффективная работа администратора обеспечивает успех ресторана, повышая удовлетворённость клиентов и финансовые показатели.

## **1.4 Постановка задачи**

Разработка программной системы для управления рестораном направлена на автоматизацию ключевых процессов: бронирования, заказов, учёта клиентов, персонала и финансов. Цель — создать консольное приложение на C#, которое повысит эффективность работы администратора, минимизирует ошибки и обеспечит удобство использования.

Цель: разработать систему для автоматизации управления рестораном. Основные задачи:

* Автоматизировать бронирование столиков и учёт их доступности.
* Управлять заказами: создание, обработка, расчёт суммы.
* Вести учёт клиентов (ФИО, телефон, история заказов).
* Управлять официантами: добавление, удаление, назначение столиков.
* Предоставлять аналитику: выручка, средний чек, популярные блюда.
* Обеспечить безопасность данных.

Объект: процесс управления рестораном (бронирование, заказы, персонал, финансы).

Предмет: программная система на C# для автоматизации этих процессов.

В итоге работы предполагается получение полностью функциональной информационной системы администратора ресторана, которая будет способствовать оптимизации процесса работы администратора, повышению уровня обслуживания клиентов. [8 c. 30]

# ГЛАВА 2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

## **2.1 Требования к программе**

Для работы программы требуется минимум 100 Мб свободного места на диске. Поддерживаются операционные системы Microsoft Windows 7 и выше (64-битные версии). Программа обеспечивает стабильность при длительном использовании и поддерживает восстановление данных после сбоев через автоматическое сохранение в текстовые файлы.

Разработка велась на следующей конфигурации:

* Процессор: Intel Core i5, 2.4 ГГц.
* Оперативная память: 8 ГБ DDR4.
* Видеокарта: Интегрированная Intel UHD Graphics.
* Операционная система: Windows 10 Pro (64-бит).
* Накопитель: SSD, 256 ГБ.
* Дисплей: 15.6 дюйма, Full HD (1920x1080).

Минимальные требования для использования:

- Операционная система: Windows 7, 10 (64-бит).

- Процессор: Intel Core 2 Duo, 1.8 ГГц.

- Оперативная память: 4 ГБ.

- Свободное место: 10 МБ.

- Минимальное разрешение экрана: 1024x768.

Эти требования обеспечивают стабильную работу программы и доступ к её функционалу.

Функциональные требования:

Система позволяет вести учёт столиков, клиентов, официантов, блюд, заказов и бронирований, выполнять аналитические запросы, такие как поиск доступных столиков, определение популярных блюд, расчёт среднего чека и выручки, управлять данными через добавление, редактирование и удаление записей, сортировать и искать информацию, предоставляет консольный интерфейс с меню и валидацией ввода.

## **2.2 Руководство пользователя**

Руководство подробно описывает использование консольного приложения для администратора ресторана. Оно включает пошаговые инструкции по запуску, навигации, выполнению операций, примеры ввода-вывода и иллюстрации для каждого действия.

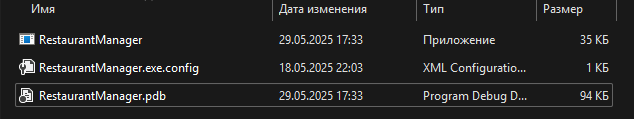
Для запуска дважды щёлкните по файлу RestaurantManager.exe. После старта отображается первое меню с командами, активируемыми цифрами на клавиатуре. Интерфейс оформлен с использованием цветового выделения заголовков для удобства (рис. 1).

Рис.1 Запуск приложения

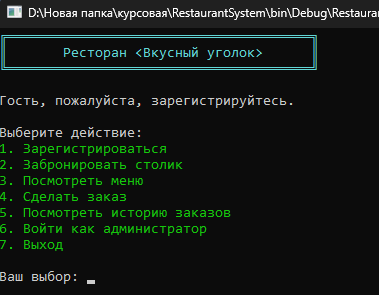
После запуска приложения откроется первое меню, его можно увидеть на (рис. 2), на нем располагается функционал для переключения, реализованный при помощи клавиш. Нажав на клавишу на клавиатуре возможен переход на действие, которое необходимо пользователю.

Рис.2. Первое меню.

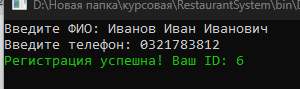
При переходе на первое действие, программа предлагает пользователю ввести свои данные: инициалы и номер телефона (рис. 3). Это нужно для того, чтобы в дальнейшем можно было использовать другие функции (бронирование столиков, оформление заказов).

Рис.3. Успешная регистрация

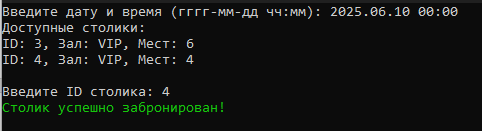
После регистрации пользователь может забронировать столик. Для этого нужно выбрать второе действие, программа запросит ввести дату и время, на которое нужна бронь на столик и ключ зала (рис. 4).

Рис.4. Успешное бронирование

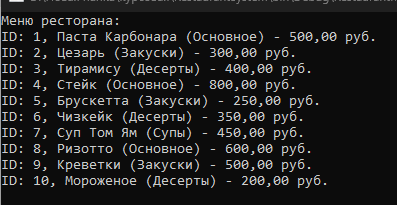
Выбрав третье действие клиент переходит в меню ресторана (рис. 5). Оно необходимо для просмотра всех блюд, что поможет определиться с заказом.

Рис.5. Просмотр меню

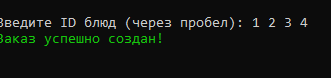
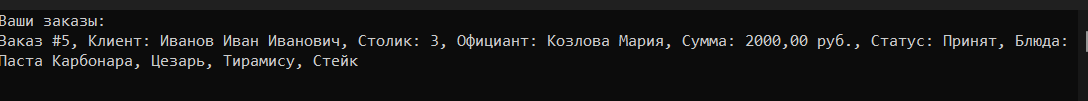
При оформлении заказа, нужно выбрать четвертое действие. Программа запросит у клиента ввести номера блюд, которые он выбрал (рис. 6).

Рис.6. Оформление заказа

Так же в программе можно посмотреть историю заказов, выбрав пятое действие. Это как личный архив, для того чтобы клиент видел, что он заказал. (рис. 7).

Рис.7. История заказов

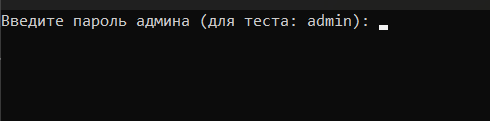
Для администратора есть отдельное меню, где он может просматривать работу ресторана и управлять рабочим персоналом. Для этого нужно выбрать шестой пункт и ввести пароль. (рис. 8).

Рис.8. Вход для администратора.

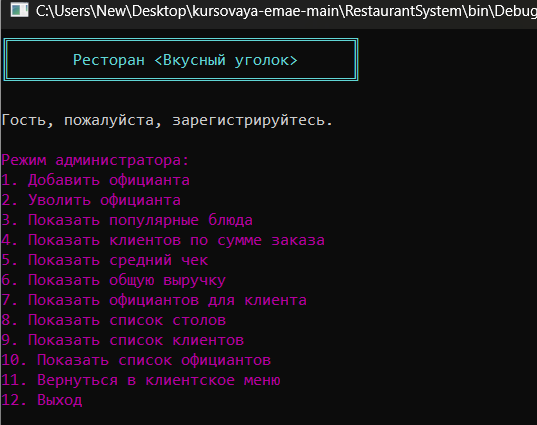
Вот как выглядит меню администратора (рис. 9):

Рис.9. Меню администратора.

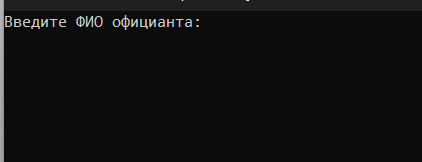
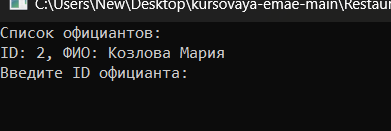
Выбрав первый или второй пункт, администратор может уволить или добавить официантов. Для этого в первом пункте нужно ввести имя официанта и столики, которые он будет обслуживать (рис. 10). А во втором выбрать ID официанта что бы уволить (рис. 11).

Рис.10. Добавление официанта. Рис.11. Увольнение официанта.

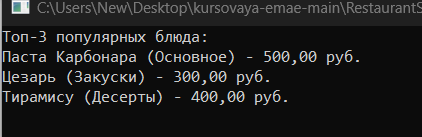
 Так же администратор может посмотреть популярные блюда, для этого он должен выбрать третий пункт (рис. 12). Это удобно для того чтобы администратор понимал на какие блюда есть спрос и мог корректировать цены или убрать те блюда, которые меньше всего заказывают.

Рис.12. Просмотр популярных блюд.

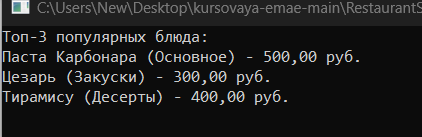
Можно посмотреть клиентов по сумме заказа, выбрав четвертый пункт (рис. 13). Это удобно для того чтобы понимать какие клиенты тратят больше всех и можно сделать им скидку.

Рис.13. Клиенты по сумме заказа.

Так же администратор может просматривать список гостей, официантов и столиков. Это очень удобно для контроля работы ресторана, обслуживающего персонала и систематизации работы.

## **Руководство программиста.**

Данный раздел описывает программную реализацию консольного приложения «Система управления рестораном», разработанного на языке C# в среде Microsoft Visual Studio. Программа предназначена для управления данными о клиентах, официантах, столиках, блюдах и заказах, их хранения в текстовых файлах и выполнения операций, таких как регистрация клиентов, бронирование столов, создание заказов и аналитические запросы (например, популярные блюда, средний чек). Руководство адресовано программистам, которые будут поддерживать или модифицировать код. Здесь описаны структура программы, классы, методы, алгоритмы и рекомендации по доработке. [9 c. 30]

Программа организована в двух пространствах имён: ConsoleApp1 (классы Client, Waiter, Table, Dish, Order, RestaurantSystem) и RestaurantManagement (класс Program, реализующий основную логику). Основной функционал сосредоточен в классах RestaurantSystem (управление данными) и Program (пользовательский интерфейс).

Данные хранятся в текстовых файлах tables.txt (столы), clients.txt (клиенты), waiters.txt (официанты), dishes.txt (блюда) и orders.txt (заказы). Программа использует объектно-ориентированный подход, разделяя функциональность на классы и методы.

Основные компоненты:

**Классы данных:** Client (клиент), Waiter (официант), Table (стол), Dish (блюдо), Order (заказ).

* **Логика управления:** Класс RestaurantSystem для операций с данными и класс Program для интерфейса.
* **Хранение данных**: Текстовые файлы с разделителями (запятая, точка с запятой).
* **Интерфейс:** Консольный интерфейс с многоуровневыми меню для клиентов и администраторов, цветовым оформлением (ConsoleColor.Cyan для заголовков, ConsoleColor.Green/Magenta для пунктов).

Программа включает пять классов для данных и два для управления. Ниже описаны их свойства и методы. [10 c. 30]

Класс Client

Описывает сущность «клиент».

**Свойства:**

* Id (тип int): Уникальный идентификатор клиента.
* FullName (тип string): ФИО клиента.
* Phone (тип string): Номер телефона.
* OrderIds (тип List<int>): Список ID заказов клиента.
* ReservationTableId (тип int?): ID забронированного столика (может быть null).

Класс Waiter:

Описывает сущность «официант».

**Свойства:**

* Id (тип int): Уникальный идентификатор официанта.
* FullName (тип string): ФИО официанта.
* AssignedTableIds (тип List<int>): Список ID столов, закрепленных за официантом.
* ServedOrderIds (тип List<int>): Список ID обслуженных заказов.

Класс Table:

Описывает сущность «стол».

**Свойства:**

* Id (тип int): Уникальный идентификатор стола.
* Hall (тип string): Название зала (например, «Основной», «VIP»).
* Seats (тип int): Количество мест.
* IsAvailable (тип bool): Доступность стола.

Класс Dish:

Описывает сущность «блюдо».

**Свойства:**

* Id (тип int): Уникальный идентификатор блюда.
* Name (тип string): Название блюда.
* Category (тип string): Категория (например, «Основное», «Десерты»).
* Price (тип decimal): Цена в рублях.

Класс Order:

Описывает сущность «заказ».

**Свойства:**

* Id (тип int): Уникальный идентификатор заказа.
* ClientId (тип int): ID клиента.
* TableId (тип int): ID стола.
* WaiterId (тип int): ID официанта.
* DishIds (тип List<int>): Список ID блюд.
* TotalAmount (тип decimal): Общая сумма заказа.
* Status (тип string): Статус (например, «Принят», «Оплачен», «Отменён»).
* OrderDate (тип DateTime): Дата и время заказа.

Класс RestaurantSystem:

Реализует логику управления данными.

**Поля:**

* Tables (тип List<Table>): Список столов.
* Clients (тип List<Client>): Список клиентов.
* Waiters (тип List<Waiter>): Список официантов.
* Dishes (тип List<Dish>): Список блюд.
* Orders (тип List<Order>): Список заказов.
* tablesFile, clientsFile, waitersFile, dishesFile, ordersFile (тип string): Пути к файлам данных.

Класс Program

Реализует пользовательский интерфейс и взаимодействие с RestaurantSystem.

**Поля:**

* Нет собственных полей, использует экземпляр RestaurantSystem.

Основные методы класса RestaurantSystem

Методы разделены на группы: инициализация данных, загрузка/сохранение файлов, управление сущностями и аналитические запросы.

Методы инициализации и работы с файлами:

InitializeData(): Вызывает методы загрузки данных (LoadTables, LoadClients, LoadWaiters, LoadDishes, LoadOrders) или создаёт файлы с тестовыми данными (CreateDefault\*).

* LoadTables(), LoadClients(), LoadWaiters(), LoadDishes(), LoadOrders(): Читают данные из файлов, создавая объекты соответствующих классов.
* SaveTables(), SaveClients(), SaveWaiters(), SaveDishes(), SaveOrders(): Сохраняют данные в файлы в формате с разделителями.
* CreateDefaultTables(), CreateDefaultClients(), CreateDefaultWaiters(), CreateDefaultDishes(), CreateDefaultOrders(): Создают файлы с тестовыми данными, если они отсутствуют.

Методы управления сущностями:

* RegisterClient(string fullName, string phone): Создаёт нового клиента с уникальным ID, сохраняет в файл.
* ReserveTable(int clientId, int tableId, DateTime dateTime): Бронирует стол, обновляя доступность и клиента.
* CreateOrder(int clientId, int tableId, List<int> dishIds): Создаёт заказ, проверяя наличие стола, клиента и официанта, рассчитывает сумму.
* AddWaiter(string fullName, List<int> assignedTableIds): Добавляет официанта с указанными столами.
* RemoveWaiter(int waiterId): Удаляет официанта, отменяя его заказы.
* GetClientOrders(int clientId): Возвращает заказы клиента.
* GetAvailableTables(DateTime dateTime): Возвращает доступные столы.
* GetMenu(): Возвращает список блюд.

**Пример** AddWaiter:

|  |
| --- |
| public void AddWaiter(string fullName, List<int> assignedTableIds) |
| { |
| int newId = Waiters.Any() ? Waiters.Max(w => w.Id) + 1 : 1; |
| Waiters.Add(new Waiter |
| { |
| Id = newId, |
| FullName = fullName, |
| AssignedTableIds = assignedTableIds ?? new List<int>() |
| }); |
| SaveWaiters(); |

Листинг 1

Методы аналитических запросов:

* GetPopularDishes(): Использует LINQ для выбора трёх самых популярных блюд по количеству заказов.
* GetClientsByOrderAmount(decimal minAmount): Возвращает клиентов с заказами на сумму не менее указанной.
* GetAverageCheck(DateTime start, DateTime end): Рассчитывает средний чек за период.
* GetTotalRevenue(): Подсчитывает общую выручку по оплаченным заказам.
* GetWaitersForClient(int clientId): Возвращает официантов, обслуживавших клиента.
* GetWaiters(), GetTables(), GetClients(): Возвращают списки соответствующих сущностей.

Основные методы класса Program:

Методы разделены на группы: отображение меню, обработка выбора и вывод сообщений.

Методы отображения меню

* DisplayHeader(Client client): Отображает заголовок с названием ресторана и приветствием клиента.
* DisplayClientMenu(RestaurantSystem, ref Client, ref bool): Показывает меню клиента (регистрация, бронирование, заказ и т.д.).
* DisplayAdminMenu(RestaurantSystem, ref bool, ref Client): Показывает меню администратора (управление официантами, аналитика).

Методы обработки выбора

* ProcessChoice(string, RestaurantSystem, ref Client, ref bool): Перенаправляет выбор в ProcessClientChoice или ProcessAdminChoice.
* ProcessClientChoice(string, RestaurantSystem, ref Client, ref bool): Обрабатывает клиентские действия (регистрация, бронирование, заказ).
* ProcessAdminChoice(string, RestaurantSystem, ref bool): Обрабатывает действия администратора (добавление/удаление официанта, аналитика).

Методы вывода сообщений:

* PrintSuccess(string): Выводит сообщение об успехе (зелёный цвет).
* PrintError(string): Выводит сообщение об ошибке (красный цвет).
* PrintMessage(string, ConsoleColor): Выводит сообщение с указанным цветом.

Рекомендации по доработке:

1. Улучшение валидации: Добавить проверки формата телефона и уникальности ID блюд/столов.
2. Расширение аналитики: Реализовать запросы, такие как статистика заказов по категориям блюд или загрузка официантов.
3. Логирование: Внедрить запись действий и ошибок в лог-файл для отладки.
4. Графический интерфейс: Перенести приложение на WPF или ASP.NET для удобства пользователей.
5. База данных: Заменить текстовые файлы на SQLite или SQL Server для повышения производительности и надёжности.
6. Авторизация: Улучшить механизм входа администратора, добавив хеширование пароля и роли пользователей.

# ЗАКЛЮЧЕНИЕ

В данной курсовой работе была разработана информационная система для управления рестораном, реализованная на языке программирования C# в среде Microsoft Visual Studio. Для хранения данных используются пять текстовых файлов (tables.txt, clients.txt, waiters.txt, dishes.txt, orders.txt), что обеспечивает простоту реализации и доступность для небольших заведений. Система предоставляет функциональность для клиентов и администраторов, включая регистрацию, бронирование столов, создание заказов, управление официантами и аналитические запросы, такие как расчет среднего чека и определение популярных блюд.

Пользовательский интерфейс создан с использованием консольных средств ввода-вывода с цветовым оформлением (ConsoleColor), что делает навигацию интуитивно понятной и визуально различимой. Интерфейс разделён на клиентский и административный режимы, обеспечивая удобное взаимодействие с системой для разных категорий пользователей. Данные о клиентах, официантах, столах, блюдах и заказах хранятся в отдельных файлах, что упрощает управление и доступ к информации.

Основные преимущества разработанной системы включают:

1. Интуитивно понятный интерфейс, обеспечивающий лёгкость освоения для клиентов и администраторов.
2. Модульная структура программы, реализованная с использованием объектно-ориентированного подхода, что упрощает поддержку и расширение.
3. Поддержка аналитических функций, позволяющих оптимизировать управление рестораном (например, анализ выручки и популярности блюд).

В процессе разработки были выявлены области для потенциальных улучшений:

1. Расширение функциональности, например, добавление возможности отмены бронирования, управления меню или интеграции с системой лояльности.
2. Улучшение консольного интерфейса или его замена на графический (например, на базе WPF или ASP.NET) для повышения удобства и презентабельности.
3. Переход от текстовых файлов к реляционной базе данных (например, SQLite или SQL Server) для повышения производительности, надёжности и масштабируемости.
4. Усиление безопасности, включая хеширование паролей для входа в административный режим и более строгую валидацию входных данных.

Цель курсовой работы была успешно достигнута: разработана информационная система, обеспечивающая автоматизацию ключевых процессов управления рестораном. Система является надёжной основой для эксплуатации в реальных условиях и может быть адаптирована под конкретные потребности заведения с учётом предложенных доработок.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

* 1. Албахари Дж., Албахари Б. C# 8.0. Карманный справочник. – Москва: ООО «И. Д. Вильямс», 2020. – 240 с.
  2. Марченко А. *Основы программирования на C# 2.0. Учебное пособие*. – 2017. – 557 с.
  3. Шилдт Г. *C# 4.0: полное руководство*. – Москва: ООО «И. Д. Вильямс», 2015. – 1056 с.
  4. Лекция «Службы гостиницы». – [https://ru.scribd.com/document/603037793/](https://ru.scribd.com/document/603037793/%D0%9B%D0%B5%D0%BA%D1%86%D0%B8%D1%8F-%D0%A1%D0%BB%D1%83%D0%B6%D0%B1%D1%8B-%D0%B3%D0%BE%D1%81%D1%82%D0%B8%D0%BD%D0%B8%D1%86%D1%8B)
  5. *История гостиничного дела*. – <https://hoteladvisor.livejournal.com/32450.html>
  6. Троелсен Э., Джепикс Ф. *Язык программирования C# 8.0 и платформа .NET Core 3.0. Том 1: Основы*. – СПб: Питер, 2020. – 832 с.
  7. Фримен А. *Pro ASP.NET Core MVC 2*. – Apress, 2017. – 1017 p.
  8. Гаевский В.М., Соловьева Е.В. *Гостиничное дело. Организация деятельности службы приема и размещения*. – М.: КноРус, 2020. – 312 с.
  9. *C# documentation – Microsoft Learn*. – <https://learn.microsoft.com/en-us/dotnet/csharp/>
  10. Кравченко Е.А. *Гостиничный бизнес: основы управления*. – М.: Юрайт, 2018. – 285 с.

# ПРИЛОЖЕНИЕ А

using ConsoleApp1;

using System;

using System.Linq;

namespace RestaurantManagement

{

class Program

{

static void Main(string[] args)

{

var system = new RestaurantSystem();

try

{

system.InitializeData();

}

catch (Exception ex)

{

PrintError($"Ошибка при загрузке данных: {ex.Message}");

return;

}

Client currentClient = null;

bool isAdmin = false;

while (true)

{

Console.Clear();

DisplayHeader(currentClient);

if (isAdmin)

DisplayAdminMenu(system, ref isAdmin, ref currentClient);

else

DisplayClientMenu(system, ref currentClient, ref isAdmin);

Console.Write("\nВаш выбор: ");

string choice = Console.ReadLine();

ProcessChoice(choice, system, ref currentClient, ref isAdmin);

}

}

static void DisplayHeader(Client client)

{

Console.ForegroundColor = ConsoleColor.Cyan;

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

Console.WriteLine("║ Ресторан «Вкусный уголок» ║");

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

Console.ResetColor();

Console.WriteLine(client == null

? "\nГость, пожалуйста, зарегистрируйтесь."

: $"\nДобро пожаловать, {client.FullName}!");

}

static void DisplayAdminMenu(RestaurantSystem system, ref bool isAdmin, ref Client currentClient)

{

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("\nРежим администратора:");

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

Console.WriteLine("2. Уволить официанта");

Console.WriteLine("3. Показать популярные блюда");

Console.WriteLine("4. Показать клиентов по сумме заказа");

Console.WriteLine("5. Показать средний чек");

Console.WriteLine("6. Показать общую выручку");

Console.WriteLine("7. Показать официантов для клиента");

Console.WriteLine("8. Показать список столов");

Console.WriteLine("9. Показать список клиентов");

Console.WriteLine("10. Показать список официантов");

Console.WriteLine("11. Вернуться в клиентское меню");

Console.WriteLine("12. Выход");

Console.ResetColor();

}

static void DisplayClientMenu(RestaurantSystem system, ref Client currentClient, ref bool isAdmin)

{

Console.WriteLine("\nВыберите действие:");

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("1. Зарегистрироваться");

Console.WriteLine("2. Забронировать столик");

Console.WriteLine("3. Посмотреть меню");

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

Console.WriteLine("5. Посмотреть историю заказов");

Console.WriteLine("6. Войти как администратор");

Console.WriteLine("7. Выход");

Console.ResetColor();

}

static void ProcessChoice(string choice, RestaurantSystem system, ref Client currentClient, ref bool isAdmin)

{

Console.Clear();

if (isAdmin)

ProcessAdminChoice(choice, system, ref isAdmin);

else

ProcessClientChoice(choice, system, ref currentClient, ref isAdmin);

Console.ReadKey();

}

static void ProcessAdminChoice(string choice, RestaurantSystem system, ref bool isAdmin)

{

switch (choice)

{

case "1": // Добавить официанта

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

string waiterName = Console.ReadLine();

Console.Write("Введите ID столиков (через пробел): ");

try

{

var tableIds = Console.ReadLine().Split().Select(int.Parse).ToList();

system.AddWaiter(waiterName, tableIds);

PrintSuccess("Официант добавлен!");

}

catch

{

PrintError("Неверный ввод ID столиков.");

}

break;

case "2": // Уволить официанта

Console.WriteLine("Список официантов:");

foreach (var waiter in system.GetWaiters())

Console.WriteLine($"ID: {waiter.Id}, ФИО: {waiter.FullName}");

Console.Write("Введите ID официанта: ");

if (int.TryParse(Console.ReadLine(), out int waiterId))

{

system.RemoveWaiter(waiterId);

PrintSuccess("Официант уволен!");

}

else

{

PrintError("Неверный ID.");

}

break;

case "3": // Популярные блюда

Console.WriteLine("Топ-3 популярных блюда:");

foreach (var dish in system.GetPopularDishes())

Console.WriteLine($"{dish.Name} ({dish.Category}) - {dish.Price:F2} руб.");

break;

case "4": // Клиенты по сумме

Console.Write("Введите минимальную сумму: ");

if (decimal.TryParse(Console.ReadLine(), out decimal minAmount))

{

Console.WriteLine($"Клиенты с заказами от {minAmount:F2} руб.:");

foreach (var client in system.GetClientsByOrderAmount(minAmount))

Console.WriteLine($"ID: {client.Id}, ФИО: {client.FullName}, Телефон: {client.Phone}");

}

else

{

PrintError("Неверная сумма.");

}

break;

case "5": // Средний чек

Console.Write("Введите дату начала (гггг-мм-дд): ");

if (DateTime.TryParse(Console.ReadLine(), out DateTime start))

{

Console.Write("Введите дату конца (гггг-мм-дд): ");

if (DateTime.TryParse(Console.ReadLine(), out DateTime end))

{

var avgCheck = system.GetAverageCheck(start, end);

Console.WriteLine($"Средний чек: {avgCheck:F2} руб.");

}

else

{

PrintError("Неверный формат даты конца.");

}

}

else

{

PrintError("Неверный формат даты начала.");

}

break;

case "6": // Общая выручка

Console.WriteLine($"Общая выручка: {system.GetTotalRevenue():F2} руб.");

break;

case "7": // Официанты для клиента

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

if (int.TryParse(Console.ReadLine(), out int clientId))

{

Console.WriteLine("Официанты, обслуживавшие клиента:");

foreach (var waiter in system.GetWaitersForClient(clientId))

Console.WriteLine($"ID: {waiter.Id}, ФИО: {waiter.FullName}");

}

else

{

PrintError("Неверный ID.");

}

break;

case "8": // Показать список столов

Console.WriteLine("Список столов:");

foreach (var table in system.GetTables())

Console.WriteLine($"ID: {table.Id}, Зал: {table.Hall}, Мест: {table.Seats}, Доступен: {(table.IsAvailable ? "Да" : "Нет")}");

break;

case "9": // Показать список клиентов

Console.WriteLine("Список клиентов:");

foreach (var client in system.GetClients())

{

var orders = string.Join(", ", client.OrderIds);

var reservation = client.ReservationTableId.HasValue ? $"Столик {client.ReservationTableId}" : "Нет";

Console.WriteLine($"ID: {client.Id}, ФИО: {client.FullName}, Телефон: {client.Phone}, Заказы: {(string.IsNullOrEmpty(orders) ? "Нет" : orders)}, Бронь: {reservation}");

}

break;

case "10": // Показать список официантов

Console.WriteLine("Список официантов:");

foreach (var waiter in system.GetWaiters())

{

var tables = string.Join(", ", waiter.AssignedTableIds);

var orders = string.Join(", ", waiter.ServedOrderIds);

Console.WriteLine($"ID: {waiter.Id}, ФИО: {waiter.FullName}, Столики: {(string.IsNullOrEmpty(tables) ? "Нет" : tables)}, Заказы: {(string.IsNullOrEmpty(orders) ? "Нет" : orders)}");

}

break;

case "11": // Вернуться в клиентское меню

isAdmin = false;

break;

case "12": // Выход

PrintMessage("Спасибо за посещение!", ConsoleColor.Yellow);

Environment.Exit(0);

break;

default:

PrintError("Неверный выбор.");

break;

}

}

static void ProcessClientChoice(string choice, RestaurantSystem system, ref Client currentClient, ref bool isAdmin)

{

switch (choice)

{

case "1": // Регистрация

if (currentClient != null)

{

PrintError("Вы уже зарегистрированы!");

return;

}

Console.Write("Введите ФИО: ");

string fullName = Console.ReadLine();

Console.Write("Введите телефон: ");

string phone = Console.ReadLine();

try

{

currentClient = system.RegisterClient(fullName, phone);

PrintSuccess($"Регистрация успешна! Ваш ID: {currentClient.Id}");

}

catch

{

PrintError("Ошибка при регистрации.");

}

break;

case "2": // Бронирование столика

if (currentClient == null)

{

PrintError("Сначала зарегистрируйтесь!");

return;

}

Console.Write("Введите дату и время (гггг-мм-дд чч:мм): ");

if (DateTime.TryParse(Console.ReadLine(), out DateTime dateTime))

{

Console.WriteLine("Доступные столики:");

var tables = system.GetAvailableTables(dateTime);

foreach (var table in tables)

Console.WriteLine($"ID: {table.Id}, Зал: {table.Hall}, Мест: {table.Seats}");

Console.Write("\nВведите ID столика: ");

if (int.TryParse(Console.ReadLine(), out int tableId))

{

if (system.ReserveTable(currentClient.Id, tableId, dateTime))

PrintSuccess("Столик успешно забронирован!");

else

PrintError("Ошибка: столик недоступен или не существует.");

}

else

{

PrintError("Неверный ID столика.");

}

}

else

{

PrintError("Неверный формат даты.");

}

break;

case "3": // Просмотр меню

Console.WriteLine("Меню ресторана:");

foreach (var dish in system.GetMenu())

Console.WriteLine($"ID: {dish.Id}, {dish.Name} ({dish.Category}) - {dish.Price:F2} руб.");

break;

case "4": // Сделать заказ

if (currentClient == null)

{

PrintError("Сначала зарегистрируйтесь!");

return;

}

if (currentClient.ReservationTableId == null)

{

PrintError("Сначала забронируйте столик!");

return;

}

Console.WriteLine("Меню:");

foreach (var dish in system.GetMenu())

Console.WriteLine($"ID: {dish.Id}, {dish.Name} ({dish.Category}) - {dish.Price:F2} руб.");

Console.Write("\nВведите ID блюд (через пробел): ");

try

{

var dishIds = Console.ReadLine().Split().Select(int.Parse).ToList();

if (system.CreateOrder(currentClient.Id, currentClient.ReservationTableId.Value, dishIds))

PrintSuccess("Заказ успешно создан!");

else

PrintError("Ошибка при создании заказа.");

}

catch

{

PrintError("Неверный ввод ID блюд.");

}

break;

case "5": // История заказов

if (currentClient == null)

{

PrintError("Сначала зарегистрируйтесь!");

return;

}

Console.WriteLine("Ваши заказы:");

var orders = system.GetClientOrders(currentClient.Id);

if (!orders.Any())

Console.WriteLine("Заказов нет.");

foreach (var order in orders)

{

var client = system.GetClients().FirstOrDefault(c => c.Id == order.ClientId);

var waiter = system.GetWaiters().FirstOrDefault(w => w.Id == order.WaiterId);

var dishNames = string.Join(", ", order.DishIds.Select(id => system.GetMenu().FirstOrDefault(d => d.Id == id)?.Name ?? "Неизвестно"));

Console.WriteLine($"Заказ #{order.Id}, Клиент: {client?.FullName ?? "Неизвестно"}, Столик: {order.TableId}, Официант: {waiter?.FullName ?? "Неизвестно"}, Сумма: {order.TotalAmount:F2} руб., Статус: {order.Status}, Блюда: {dishNames}");

}

break;

case "6": // Вход как администратор

Console.Write("Введите пароль админа (для теста: admin): ");

if (Console.ReadLine().Trim() == "admin")

{

isAdmin = true;

PrintSuccess("Вход в режим администратора успешен!");

}

else

{

PrintError("Неверный пароль.");

}

break;

case "7": // Выход

PrintMessage("Спасибо за посещение!", ConsoleColor.Yellow);

Environment.Exit(0);

break;

default:

PrintError("Неверный выбор.");

break;

}

}

static void PrintSuccess(string message)

{

PrintMessage(message, ConsoleColor.Green);

}

static void PrintError(string message)

{

PrintMessage(message, ConsoleColor.Red);

}

static void PrintMessage(string message, ConsoleColor color)

{

Console.ForegroundColor = color;

Console.WriteLine(message);

Console.ResetColor();

}

}

}