НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники Направление подготовки 09.03.04 Программная инженерия Дисциплина «Информационные системы»

ОТЧЕТ

По курсовой работе

«CRM для Карсти Крабс»

Студенты:
Ильин Н. С., Алхимовици А., гр. P3310

Преподаватель:
Воронина Д. С.

Санкт-Петербург
2025 г.

Содержание

Подробное текстовое описание предметной области:

CRM для «Красти Крабс» — это информационная система для автоматизации бизнес-процессов ресторана «Красти Крабс», специализирующегося на приготовлении и продаже крабсбургеров. Она предназначена для учета заказов, управления клиентами, контроля за запасами ингредиентов и анализа показателей продаж. Система помогает повысить эффективность работы ресторана, улучшить обслуживание посетителей и поддерживать доступность возможности заказа легендарных бургеров.

Система объединяет данные о клиентах, заказах, меню, сотрудниках в единую платформу. Это позволяет владельцу (Мистеру Крабсу) контролировать все аспекты деятельности ресторана: от количества заказов до остатка ингредиентов на складе.

Зачем нужна информационная система:

Информационная система «CRM для Красти Крабс» предназначена для оптимизации процессов ресторанного бизнеса и повышения эффективности легендарной закусочной мистера Крабса. Она обеспечивает централизованное управление клиентами, заказами, меню, складом и сотрудниками. Система помогает автоматизировать рутинные операции, сократить издержки, повысить качество обслуживания и сделать бизнес более прозрачным и прибыльным.

CRM не только ускоряет обслуживание гостей и экономит ресурсы, но и создает основу для лояльности клиентов, анализа продаж и дальнейшего роста заведения.

Основные задачи, которые решает система:

1. **Управление заказами и клиентами**

- Поддержка приёма и обработки заказов (на вынос, в зале, онлайн-заказ с доставкой).
- Ведение истории покупок и общих предпочтений клиентов.
- Формирование клиентской базы для программ лояльности и персонализированных предложений.

2. **Контроль меню и ингредиентов**

- Автоматизация учёта меню: актуальные цены, состав блюд, доступность.
- Взаимосвязь заказов с остатками ингредиентов.

3. **Управление сотрудниками и рабочими сменами**

- Хранение информации о персонале: роли, расписание работы, зарплаты.
- Учет производительности (например, среднее время приготовления заказа Спанч Бобом).

4. **Финансовый учет и отчётность**

- Подсчёт дневной, недельной, месячной выручки.
- Контроль расходов на ингредиенты и зарплаты.

5. **Программы лояльности**

- Реализация скидок для VIP-клиентов.

6. Мониторинг качества обслуживания

- Сбор обратной связи от посетителей.

7. Централизованное управление и контроль владельца

- Владелец имеет доступ к сводным данным обо всех процессах в реальном времени.
- Возможность контролировать персонал и состояние склада через единую панель.

Требования к разрабатываемой информационной системе:

Роли

- **Гость** — неавторизованный офлайн-посетитель, может оформить заказ только на кассе через кассира; личных данных не предоставляет.
- **Клиент** — авторизованный пользователь (веб/мобильный интерфейс), может оформлять онлайн-заказы, видеть историю своих заказов и оставлять отзывы.
- **Кассир** — сотрудник кассы; принимает оффлайн заказы у кассы, принимает наличные. Не имеет доступа к интерфейсам, доступным повору или управляющему.
- **Повар** — сотрудник кухни; видит очередь заказов, меняет только производственные статусы («Готовится», «Готов»). Не имеет доступа к интерфейсам, доступным кассиру или управляющему.
- **Управляющий** — административная роль; может просматривать статистику продаж, историю заказов и отзывов, управляет меню и персоналом. Не имеет доступа к интерфейсам, доступным повору или кассиру.

Функциональные требования

- **FR-001** Система должна позволять пользователю оформлять заказ (на вынос, в зале или с доставкой).
- **FR-002** Система должна предоставлять пользователю доступ к актуальному меню с ценами и составом блюд.
- **FR-003** Система должна обеспечивать возможность отслеживания статуса заказа.
- **FR-004** Система должна предоставлять возможность оставления пользователем обратной связи после оформления заказа.
- **FR-005** Система должна поддерживать участие пользователей в программе лояльности (накопление бонусов, получение скидок).
- **FR-006** Система должна обеспечивать возможность редактирования пользователем своих персональных данных (телефон, адрес доставки, предпочтения).
- **FR-007** Система должна предоставлять пользователю доступ к истории его заказов.
- **FR-008** Система должна позволять сотрудникам принимать и обрабатывать заказы (подтверждение, изменение статуса).

- **FR-009** Система должна предоставлять возможность сотрудникам вводить и обновлять информацию о наличии ингредиентов.
- **FR-010** Система должна автоматически фиксировать время приготовления заказа на основе изменения статусов («Готовится» → «Готов»).
- **FR-011** Система должна обеспечивать доступ сотрудникам к расписанию рабочих смен.
- **FR-012** Система должна поддерживать ведение учёта проданных блюд.
- **FR-013** Система должна показывать повару новые заказы.
- **FR-014** Система должна предоставлять управляющему доступ к статистике продаж (дневная, недельная, месячная).
- **FR-015** Система должна обеспечивать управляющему возможность контролировать остатки ингредиентов на складе.
- **FR-016** Система должна предоставлять управляющему возможность управления меню (цены, состав, доступность).
- **FR-017** Система должна предоставлять управляющему возможность управления персоналом (назначение расписания, роли, зарплаты).
- **FR-018** Система должна предоставлять управляющему доступ к обратной связи от клиентов.
- **FR-019** Система должна поддерживать кассовый сценарий оплаты наличными: кассир вводит сумму полученных средств, система рассчитывает и отображает сумму сдачи.
- **FR-020** Система должна предоставлять управляющему доступ к расширенной статистике бизнеса: истории заказов и отзывов.
- **FR-021** Система должна вести журналы (логи) действий сотрудников и изменений заказов.

Нефункциональные требования

Удобство использования

- **UR-001** Система должна обеспечивать выполнение операции оформления заказа не более чем за 3 минуты.
- **UR-002** Система должна предоставлять навигацию, позволяющую найти любое блюдо не более чем за 30 секунд.

Производительность

- **PR-001** Система должна загружать главный экран (меню) за ≤5 секунд при стабильном подключении к интернету со скоростью ≥12 Мбит/с.
- **PR-002** Система должна обновлять статус заказа и отображать изменения не позднее чем через 5 секунд.
  - **PR-003** Система должна выдерживать нагрузку ≥50 запросов/с в ходе приёмочного теста TPL-LOAD-001 (непрерывно 10 минут).

- **PR-004** Необходимо, чтобы интернет-подключение имело пропускную способность не менее 12 Мбит/с.
- **PR-005** Система должна обновлять информацию о состоянии складских остатков в течение 10 секунд после ввода изменений.

SLI/SLO/SLA для производительности

- **Определения**
  - **SLI** (Service Level Indicator) - измеримая метрика качества сервиса.
  - **SLO** (Service Level Objective) - целевое значение SLI.
  - **SLA** (Service Level Agreement) - доля соблюдения SLO за отчётный период.
- **Предпосылки измерений**
  - Подключение клиента к интернету: ≥12 Мбит/с, средняя RTT ≤100 мс.
  - Для серверных SLI учитываются только успешные ответы (HTTP 2xx/3xx).
  - **SLI и SLO (пороговые, для базовых проверок)**
    - **SLI-1: menu_load_good_ratio** - good_loads(≤5 с) / total_loads; событие = одна проверка загрузки меню.
      - **SLO-1:** ≥ 98% за месяц.
    - **SLI-2: status_update_good_ratio** - good_updates(≤5 с) / total_updates; событие = одна проверка доставки статуса в UI.
      - **SLO-2:** ≥ 98% за месяц.
    - **SLI-3: inventory_propagation_good_ratio** - good_propagations(≤10 с) / total_propagations; событие = одна проверка отражения остатков.
      - **SLO-3:** ≥ 98% за месяц.
    - **SLI-4: throughput_acceptance** - проверяется только в рамках приёмочного нагрузочного теста (см. ниже), не входит в ежемесячный SLA.
- **SLA и метрики**
  - Отчётный период: календарный месяц.
  - **Метрика соблюдения SLA** для каждого SLI: percent_compliance = (кол-во событий, удовлетворяющих SLO) / (все события) · 100%.
  - **SLA-уровень:** percent_compliance ≥ 98.0% по каждому из SLI-1..3 за отчётный месяц. Нагрузочная пропускная способность (SLI-4) проверяется однократно при приёмке.
  - **Error budget:** до 2.0% событий в месяц могут превышать SLO без нарушения SLA.
  - **Сбор и верификация**
    - Источник данных: замеры времени выполнения HTTP-запросов и проверки содержания ответов.
    - Нарушение SLA фиксируется при percent_compliance < 98% по любому SLI за отчётный период.
  - **Автоматические проверки**
    - **P1: Загрузка меню ≤5 с** - GET главной страницы/меню; успех, если общее время ≤5 с.
      - пример: curl с измерением общего времени ответа; писать «ok», если ≤5.0 с.

- **P2: Обновление статуса заказа ≤5 с** - в тестовом заказе изменить статус и опросом 1 раз в секунду проверять, что API/страница отразила изменение ≤5 с.
  - успех, если изменение наблюдается не позже 5-й секунды.
- **P3: Отражение остатков ≤10 с** - изменить тестовый остаток и опросом проверить, что изменение видно в API/интерфейсе ≤10 с.
- **P4: Пропускная способность (опционально для мониторинга)** - при необходимости краткий прогон; основной нагрузочный тест выполняется один раз на приёмке (см. ниже).
- **Агрегация SLA** - percent_compliance = ok/total по каждой проверке; порог 98% за месяц.

▸ **Нагрузочное тестирование при приёмке (однократно)**
- **Цель** - подтвердить, что система выдерживает базовую целевую нагрузку до запуска.
- **Сценарий PR-LOAD-001**
  - Продолжительность: 10 минут непрерывной нагрузки.
  - Нагрузка: ключевые эндпоинты - получение меню, создание заказа.
  - Критерии успешности (pass/fail):
    - Requests/sec (среднее за прогон) ≥ 50.
    - Доля ответов не-2xx/3xx ≤ 1%.
    - 95-й перцентиль времени ответа ≤ 5 с.
- **Фиксация результата** - сохранить summary инструмента как артефакт приёмки.

Надежность

- **RR-002** Среднее время на устранение проблем должно составлять 3 часа; в случае критического сбоя система должна быть восстановлена в течение 1 дня.

Безопасность

- **SEC-001** Система должна обеспечивать аутентификацию пользователей через Яндекс.ID.
- **SEC-002** Система должна обеспечивать ролевую модель доступа (гость, клиент, кассир, повар, управляющий) с ограничением операций по ролям.
- **SEC-003** Система должна быть защищена от SQL-инъекций.
- **SEC-004** Использовать актуальные версии сторонних библиотек и зависимостей.

Сопровождаемость

- **MAINT-001** Кодовая база хранится в репозитории GitHub; вся разработка ведётся через ветки и pull request.
- **MAINT-002** Для единообразия стиля применяется автоматическое форматирование кода (форматтер); проверка форматирования входит в проверки перед слиянием.
- **MAINT-003** К качеству кода применяются правила статического анализа (линтер); наличие ошибок линтера блокирует слияние изменений.
- **MAINT-004** Запуск форматтера и линтера является частью конвейера CI; при несоответствии стиль-гайду или при ошибках конвейер помечается как неуспешный.

- **MAINT-005** Необходимо поддерживать базовые модульные тесты для ключевой функциональности; падение тестов блокирует слияние.
- **MAINT-006** Минимальное покрытие кода модульными тестами — не менее 50%; порог измеряется и проверяется в CI, несоответствие блокирует слияние.

CI/CD

- **CI-001** В экосистеме GitHub настраивается CI для автоматической сборки, запуска тестов и проверок качества кода при каждом push и pull request.
- **CI-002** Базовый сценарий конвейера: 1) подготовка окружения; 2) сборка; 3) запуск линтеров; 4) проверка форматирования; 5) запуск модульных тестов; 6) публикация артефактов сборки/отчётов как артефактов CI.
- **CI-003** Развертывание по требованию: требуется, чтобы из основной ветки репозитория можно выполнить развёртывание собранного артефакта на целевое окружение проекта.
- **CI-004** Конвейер pull request должен быть зелёным (все шаги успешны), иначе слияние запрещено.

Прецеденты:

Прецедент: Заказ на кассе
ID: 1
Краткое описание: Клиент приходит в ресторан, выбирает блюда и оплачивает заказ у кассы. Своих личных данных клиент не сообщает.
Главный актер: Клиент.
Предусловия: Клиент находится в ресторане и стоит у кассы. Система работает и отображает актуальное меню.
Основной поток: 1. Клиент говорит кассиру, какие блюда он хочет. 2. Сотрудник кассы через систему формирует заказ. 3. Система проверяет наличие ингредиентов и добавляет блюда в чек. 4. Система рассчитывает стоимость. 5. Кассир принимает оплату наличными: вводит сумму полученных средств, система рассчитывает и отображает сумму сдачи; кассир выдаёт сдачу. 6. Система сохраняет заказ и передаёт его на кухню.
Альтернативный поток: 3а. Если нужное блюдо недоступно: 1. Сотрудник кассы сообщает об этом клиенту. 2. Клиент выбирает другое блюдо или отказывается от заказа.

Постусловия: Заказ успешно создан и передан на кухню, или отменён.

Прецедент: Онлайн-заказ с доставкой
ID: 2
Краткое описание: Клиент через сайт или приложение выбирает блюда, указывает адрес доставки и способ оплаты. Система формирует заказ и передаёт курьеру.
Главный актер: Клиент, Курьер.
Предусловия: Клиент зашёл на сайт/приложение. Меню загружено и актуально.
Основной поток: 1. Клиент открывает меню. 2. Система отображает список блюд с ценами и составом. 3. Клиент добавляет выбранные позиции в корзину. 4. Клиент вводит контактные данные и адрес доставки. 5. Система рассчитывает итоговую стоимость заказа. 6. Клиент выбирает способ оплаты: онлайн (картой) или при получении (наличными). 7. Клиент подтверждает заказ. 8. Система сохраняет заказ и передаёт информацию курьеру.
Альтернативный поток: 3а. Если выбранное блюдо недоступно: 1. Система уведомляет клиента. 2. Клиент может удалить/заменить блюдо или отменить заказ. 6а. Если онлайн-оплата не прошла: 1. Система уведомляет клиента. 2. Клиент выбирает другой метод оплаты или отменяет заказ.
Постусловия: Заказ успешно создан и передан курьеру, либо отменён.

Прецедент: Управление складом ингредиентов
ID: 3
Краткое описание: Сотрудник вводит данные о приходе или расходе ингредиентов в систему.
Главный актер: Сотрудник склада.
Предусловия: Сотрудник авторизован в системе.
Основной поток: 1. Сотрудник открывает раздел «Склад».

2. Вводит количество поступивших или списанных ингредиентов.

3. Система обновляет остатки.

4. Информация становится доступна для кассы, кухни и отчётности.

Альтернативный поток:

2а. Если введено некорректное значение:

1. Система уведомляет сотрудника об ошибке.
2. Сотрудник исправляет данные.

Постусловия: Складские данные актуализированы.

Прецедент: Просмотр отчёта о продажах

ID: 4

Краткое описание: Управляющий анализирует финансовые показатели заведения через отчёты в системе.

Главный актер: Управляющий.

Предусловия: Управляющий авторизован в системе.

Основной поток:

1. Управляющий выбирает период отчёта (день, неделя, месяц).
2. Система формирует отчёт по продажам.
3. Управляющий просматривает структуру продаж (по блюдам, сотрудникам, времени суток).
4. Система сохраняет факт просмотра и формирует статистику для истории.

Альтернативный поток:

2а. Если данных за выбранный период нет:

1. Система уведомляет управляющего.
2. Управляющий выбирает другой период.

Постусловия: Управляющий получил отчёт о продажах или уведомление об отсутствии данных.

Прецедент: Регистрация клиента в системе (сайт/приложение)

ID: 5

Краткое описание: Новый клиент создаёт учётную запись, чтобы оформлять онлайн-заказы и участвовать в программе лояльности.

Главный актер: Клиент.

Предусловия: Сайт/приложение доступно.

Основной поток:

1. Клиент открывает страницу регистрации. 2. Вводит личные данные (имя, телефон, email/логин, пароль). 3. Система проверяет корректность данных. 4. Система создаёт новый профиль клиента.
Альтернативный поток: 1а. Если данные некорректны: 1. Клиент получает уведомление 2. Клиент исправляет ошибки. 1б. Если телефон/email уже зарегистрирован: 1. Система уведомляет клиента 2. Система предлагает вход.
Постусловия: Клиент зарегистрирован в системе, учётная запись создана.

Прецедент: Авторизация клиента в системе
ID: 6
Краткое описание: Клиент авторизуется для просмотра истории заказов, персональных данных и участия в бонусных программах.
Главный актер: Клиент.
Предусловия: Клиент зарегистрирован в системе.
Основной поток: 1. Клиент вводит логин и пароль. 2. Система проверяет данные. 3. В случае успешной проверки клиент получает доступ в личный кабинет.
Альтернативный поток: 2а. Если данные некорректны: 1. Система сообщает об ошибке. 2. Клиент может попробовать снова или восстановить пароль.
Постусловия: Клиент получил доступ к системе или остался неавторизованным.

Прецедент: Управление личным кабинетом
ID: 7
Краткое описание: Клиент меняет свои данные (адрес доставки, пароль).
Главный актер: Клиент.
Предусловия: Клиент авторизован в системе.

Основной поток: 1. Клиент авторизуется в системе. 2. Открывает «Личный кабинет». 3. Редактирует данные. 4. Система сохраняет изменения.
Альтернативный поток: 1а. Если данные некорректны: 1. Клиент получает уведомление 2. Клиент исправляет ошибки.
Постусловия: Данные клиента обновлены.

Прецедент: Управление меню
ID: 8
Краткое описание: Управляющий обновляет информацию о блюдах: меняет цену, состав и доступность.
Главный актер: Управляющий.
Предусловия: Управляющий авторизован в системе.
Основной поток: 1. Управляющий открывает раздел «Меню». 2. Выбирает блюдо для изменения. 3. Редактирует цену, состав или статус доступности. 4. Система сохраняет обновлённые данные и обновляет меню.
Альтернативный поток: 3а. Если управляющий вводит некорректные данные: 1. Система сообщает об ошибке. 2. Управляющий корректирует данные.
Постусловия: Меню в системе актуализировано.

Прецедент: Управление персоналом
ID: 9
Краткое описание: Управляющий назначает роли, рабочие смены и зарплаты сотрудников.
Главный актер: Управляющий.
Предусловия: Управляющий авторизован в системе.
Основной поток:

1. Управляющий открывает раздел «Персонал». 2. Выбирает сотрудника. 3. Редактирует данные сотрудника (роль, смены, оклад). 4. Система сохраняет изменения и обновляет расписание.
Альтернативный поток: 3а. При вводе некорректных данных: 1. Система сообщает об ошибке. 2. Управляющий исправляет ввод.
Постусловия: Информация о сотруднике актуализирована.

Прецедент: Отслеживание статуса онлайн-заказа
ID: 10
Краткое описание: Клиент отслеживает текущее состояние своего заказа.
Главный актер: Клиент.
Предусловия: Клиент оформил заказ.
Основной поток: 1. Клиент открывает раздел «Мои заказы». 2. Система отображает список заказов и их статусы. 3. Клиент просматривает информацию о своём заказе.
Альтернативный поток: 2а. Если данные не загрузились: 1. Система уведомляет клиента об ошибке связи. 2. Клиент повторяет попытку позже.
Постусловия: Клиент получил актуальную информацию об онлайн-заказе.

Прецедент: Обратная связь
ID: 11
Краткое описание: После завершения заказа клиент оставляет отзыв о блюде и сервисе.
Главный актер: Клиент.
Предусловия: Заказ оплачен и зарегистрирован в системе.
Основной поток: 1. Клиент получает в интерфейсе предложение оставить отзыв. 2. Заполняет форму: оценку, комментарий. 3. Система сохраняет отзыв.

4. Управляющий получает доступ к обратной связи.
Альтернативный поток: 2а. Клиент не оставляет отзыв: 1. Система ничего не сохраняет.
Постусловия: Отзыв сохранён и доступен управляющему или отсутствует.

Прецедент: Финансовый учёт
ID: 12
Краткое описание: Система подсчитывает выручку и расходы ресторана.
Главный актер: Управляющий.
Предусловия: Управляющий авторизован в системе.
Основной поток: 1. Управляющий открывает раздел «Отчёты». 2. Система формирует отчёт выручки и расходов (ингредиенты, зарплаты). 3. Управляющий просматривает итоговую статистику.
Альтернативный поток: 2а. Если данных нет (например, новый день): 1. Система уведомляет управляющего об отсутствии информации.
Постусловия: Управляющий получил сводку финансов.

Прецедент: Принятие и обработка заказов
ID: 13
Краткое описание: Кассир принимает поступивший заказ; Повар изменяет только производственные статусы («Готовится», «Готов»).
Главный актер: Сотрудник кассы.
Предусловия: Сотрудник авторизован в системе.
Основной поток: 1. Кассир открывает список заказов. 2. Выбирает новый заказ. 3. Подтверждает приём заказа. 4. Повар в своей панели меняет производственные статусы по мере готовности («Готовится», «Готов»). 5. Система обновляет данные и отображает статус клиенту.
Альтернативный поток: 2а. Если заказ отменён клиентом до подтверждения:

1. Система снимает заказ с обработки.
Постусловия: Статус заказа изменён и отображается всем заинтересованным сторонам.

Прецедент: Авторизация сотрудника/управляющего
ID: 16
Краткое описание: Сотрудник входит в систему с учётом своей роли доступа.
Главный актер: Сотрудник кассы, Повар, Управляющий.
Предусловия: Сотрудник зарегистрирован в системе.
Основной поток: 1. Сотрудник вводит логин и пароль. 2. Система проверяет данные и роль сотрудника. 3. В случае успеха система предоставляет доступ к рабочему интерфейсу в зависимости от роли.
Альтернативный поток: 2а. Если данные некорректны: 1. Система сообщает об ошибке. 2. Пользователь может повторить попытку.
Постусловия: Пользователь получил доступ к системе в рамках своей роли или остался неавторизован.

Бизнес-процессы

1. Приём заказа в ресторане

Цель быстро принять заказ гостя и передать его на кухню.
Участники Клиент; Сотрудник кассы; Повар; Система

Ход процесса

1. Клиент сообщает кассиру желаемые блюда.
2. Сотрудник кассы вносит заказ в систему.
3. Система проверяет наличие ингредиентов.
   - Если ингредиенты есть, заказ формируется.
   - Если чего-то не хватает, кассир предлагает замену.
4. Система рассчитывает стоимость.
5. Клиент оплачивает заказ наличными; кассир вводит сумму полученных средств и выдаёт сдачу.
6. Заказ фиксируется в системе и уходит на кухню.

Результат Заказ создан, оплачен и готовится.

2. Онлайн-заказ с доставкой

Цель предоставить клиенту возможность заказать с сайта или приложения.
Участники Клиент; Курьер; Система

Ход процесса

1. Клиент открывает меню.
2. Добавляет блюда в корзину.
3. Указывает адрес и контактные данные.
4. Система рассчитывает итоговую стоимость.
5. Клиент выбирает способ оплаты: онлайн (картой) или при получении (наличными).
6. Система формирует заказ.
7. Заведение получает данные и сотрудники передают заказ в доставку.

Результат заказ передан в доставку или отменён.

3. Приготовление заказа

Цель обеспечить своевременное приготовление блюд.
Участники Повар; Система

Ход процесса

1. Повар видит новые заказы в панели.
2. Отмечает начало приготовления — система фиксирует время.
3. Готовит заказ.
4. Отмечает статус «Готов».
5. Система сохраняет время приготовления.

Результат заказ готов, клиент получает уведомление.

4. Управление складом ингредиентов

Цель поддерживать актуальные остатки ингредиентов.
Участники Сотрудник склада; Система; Управляющий

Ход процесса

1. Сотрудник вносит приход/расход ингредиентов.
2. Система обновляет остатки.
3. Информация отображается кассе и кухне.
4. При критическом остатке система уведомляет управляющего.

Результат складские данные актуальны, принятие заказов не сбоит.

5. Управление персоналом

Цель планировать рабочие смены, роли и оплату.
Участники Управляющий; Система; Сотрудник кассы; Повар; Сотрудник склада; Курьер

Ход процесса

1. Управляющий открывает раздел «Персонал».
2. Назначает смены, роли, зарплаты.
3. Система сохраняет изменения.
4. Сотрудники видят актуальное расписание.

Результат персонал распределён, план работы синхронизирован.

6. Финансовый учёт и отчётность

Цель контролировать прибыль и расходы.
Участники Управляющий; Система

Ход процесса

1. Система фиксирует каждую транзакцию (выручка, списания ингредиентов, оплата труда).
2. Управляющий выбирает период отчёта.
3. Система строит аналитику продаж и расходов.
4. Управляющий принимает решения (например, оптимизация затрат).

Результат отчёт готов, есть основа для управленческих решений.

7. Программа лояльности

Цель удержание клиентов.
Участники Клиент; Система; Управляющий

Ход процесса

1. Клиент регистрируется в системе.
2. При заказах накапливаются бонусы или скидки.
3. Система хранит историю заказов и предпочтений.
4. Управляющий формирует акционные предложения.

Результат клиенты возвращаются охотнее, бизнес растёт.

8. Обратная связь клиентов

Цель улучшение качества сервиса.
Участники Клиент; Система; Управляющий

Ход процесса

1. После заказа клиенту предлагается оставить отзыв.
2. Клиент пишет оценку и комментарий.
3. Система сохраняет отзыв.
4. Управляющий анализирует отзывы и принимает решения.

Результат качество сервиса улучшается.

Список технологий и фреймворков:

Технологии области представления системы

- React v19
- Nginx v1.29.1

Технологии на серверной части

- Spring Boot 4
- Spring Security
- PostgreSQL
- OpenAPI

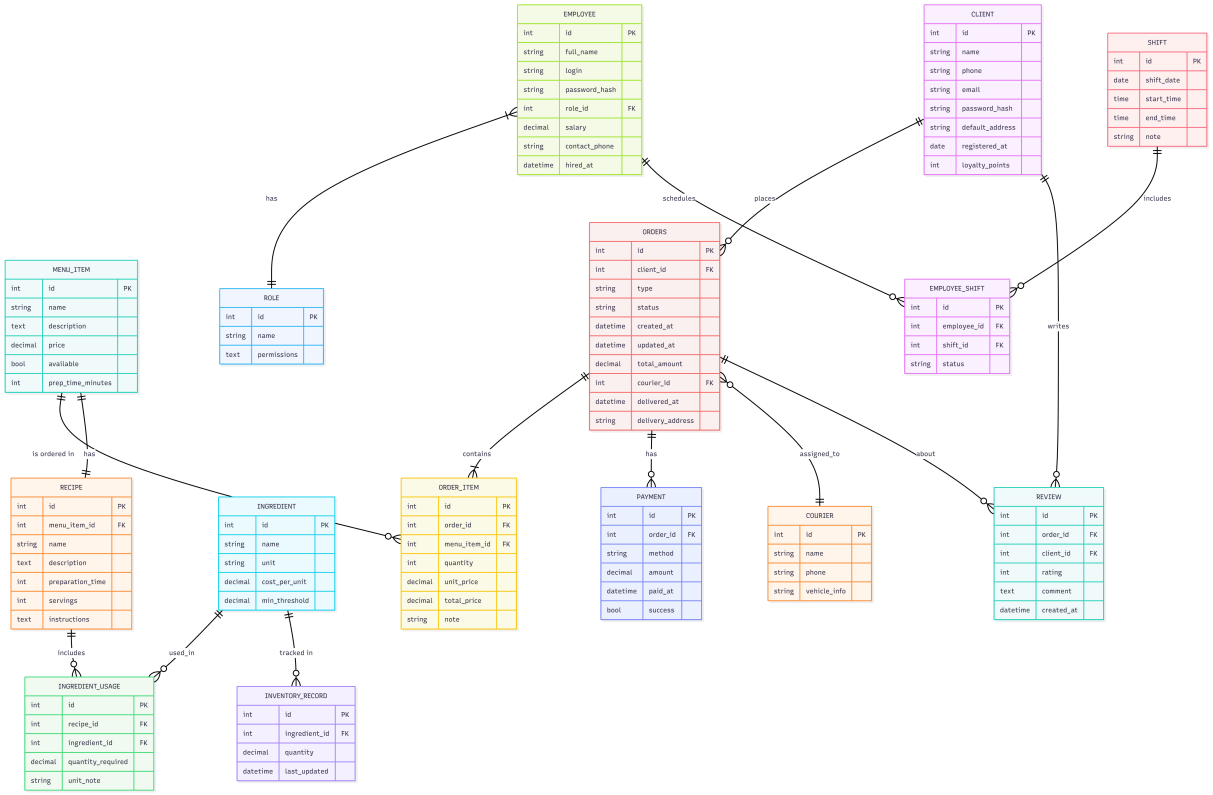
Технологии для развертывания инфраструктуры

- Docker
- Docker-compose

Технологии для обеспечения качества и надежности систем

- JUnit5
- Mockito
- Checkstyle
- CI/CD Github Actions

ER-модель:



Даталогическая модель и целостность данных:

Реализована даталогическая модель в PostgreSQL (db/schema.sql): таблицы clients, menu_items, recipes, ingredients, ingredient_usages, inventory_records, orders, order_items, payments, employees, roles, shifts, employee_shifts, reviews, couriers.

CHECK-ограничения: неотрицательные суммы/количества, rating 1–5, валидность статусов/типов заказа, обязательность адреса/курьера для доставки.

Индексы на все FK, а также полезные составные индексы (см. раздел про бенчмарки ниже).

Триггеры для обеспечения целостности

- orders.updated_at - автоматическое обновление при изменении заказа.
- Пересчёт итоговой суммы заказа и проверка оплаты: recalc_order_total_and_validate() пересчитывает orders.total_amount и, при наличии успешного платежа, требует точного равенства суммы платежа итогу заказа.
- order_items_default_price - подстановка текущей цены блюда при вставке позиции, если цена не передана.
- payments_set_paid_at - BEFORE-триггер, автоматически проставляющий paid_at при success = true.
- inventory_touch - автообновление last_updated в записях склада.
- employee_shifts_no_overlap - запрет пересекающихся смен у одного сотрудника в рамках даты.

PL/pgSQL-функции и процедуры:

- place_order(client_id, type, delivery_address, items jsonb) → id заказа: создание заказа с валидацией и добавлением позиций из JSON.
- update_order_status(order_id, new_status): допустимые переходы статусов; для доставки устанавливается delivered_at.
- process_payment(order_id, method) → id платежа: единоразовый платёж на сумму заказа с success = true.
- get_kitchen_queue(): очередь кухни (заказы в confirmed/preparing с агрегированным списком позиций).
- sales_summary(from, to): выручка/средний чек за период по успешным платежам.
- top_menu_items(from, to, limit): топ блюд по количеству и выручке.
- low_stock(threshold_factor): ингредиенты ниже порога; restock_ingredient(id, delta): пополнение/коррекция остатков.

Индексы и сравнительный анализ производительности:

Использованные индексы

- orders(status, created_at) - для очереди кухни (фильтр по статусам + сортировка по времени).
- orders(client_id, created_at desc, id) - покрывающий для ленты заказов клиента (index-only scan).
- payments(paid_at) where success = true - частичный под отчёты по выручке (окна по времени).
- menu_items using gin(name gin_trgm_ops) - триграммный для ILIKE '%…%'.

Нагрузка и методика

- Генераторы: 10k клиентов/меню, 50k заказов, 100k+ позиций, 50k платежей (db/generate_test_data.sql).
- Замеры: EXPLAIN ANALYZE, 3 прогона «до» и «после», усреднение.
- Среда: PostgreSQL 16 (Docker), scripts/benchmark.py.

Результаты замеров

- Очередь кухни: status IN ('confirmed','preparing') ORDER BY created_at LIMIT 5000
  - До: 9.135 ms → После: 5.249 ms (≈1.7× быстрее)
  - Размер индекса: 1896 kB
- История клиента: client_id = 8471 ORDER BY created_at DESC LIMIT 1000 (covering)
  - До: 0.122 ms → После: 0.042 ms (≈2.9× быстрее)
  - Размер индекса: 2008 kB
- Выручка (30 дней): success = true AND paid_at ∈ [now()-30d, now()]
  - До: 2.823 ms → После: 2.071 ms (≈1.4× быстрее)
  - Размер индекса: 1056 kB
- Поиск по меню: name ILIKE '%Patty%'
  - До: 3.094 ms → После: 1.759 ms (≈1.8× быстрее)
  - Размер индекса: 408 kB

Выводы

- Все индексы показали ускорение целевых запросов, при умеренной стоимости по памяти (сотни килобайт — единицы мегабайт).
- Частичный индекс по payments минимизирует размер и ускоряет агрегации в окнах времени.
- Покрывающий индекс по orders обеспечивает index-only сканирование для ленты клиента.
- Триграммный GIN по menu_items.name сильно ускоряет гибкий поиск, что важно для UX.
- Итог: индексы целесообразны для заявленных бизнес-процессов и подтверждены измерениями.

Инвентаризация индексов (SQL-запрос):

```
SELECT ns.nspname AS schema, c.relname AS table, i.relname AS index, pg_get_indexdef(i.oid) AS indexdef, pg_size_pretty(pg_relation_size(i.oid)) AS index_size, s.idx_scan, s.idx_tup_read, s.idx_tup_fetch FROM pg_class c JOIN pg_index x ON c.oid = x.indrelid JOIN pg_class i ON i.oid = x.indexrelid JOIN pg_namespace ns ON ns.oid = c.relnamespace LEFT JOIN pg_stat_all_indexes s ON s.indexrelid = i.oid WHERE ns.nspname = 'public' ORDER BY pg_relation_size(i.oid) DESC;
```

Результат (топ-20 по размеру):

table	index	index_size	idx_scan	idx_tup_read	idx_tup_fetch
orders	orders_pkey	3320 kB	300002	626296	300002
orders	idx_orders_client_id	2344 kB	5	50060	0
order_items	order_items_pkey	2208 kB	0	0	0
orders	idx_orders_client_created_at_id	2008 kB	4	60	0
orders	idx_orders_status_created_at	1896 kB	8	50256	0
order_items	idx_order_items_order_id	1848 kB	200001	433339	0
orders	idx_orders_courier_id	1584 kB	0	0	0
payments	uq_payments_order	1560 kB	150001	50000	50000
payments	idx_payments_order_id	1560 kB	0	0	0
order_items	idx_order_items_menu_item_id	1248 kB	0	0	0
payments	payments_pkey	1112 kB	0	0	0
payments	idx_payments_paid_at_success	1056 kB	4	31544	0
clients	uq_clients_email	680 kB	0	0	0
menu_items	idx_menu_items_name_trgm	408 kB	4	14016	0
clients	uq_clients_phone	328 kB	0	0	0
clients	clients_pkey	240 kB	262504	262504	262504
menu_items	menu_items_pkey	240 kB	100001	100001	100001
couriers	uq_couriers_phone	16 kB	0	0	0
couriers	couriers_pkey	16 kB	31853	31853	31853
employees	uq_employees_login	8192 bytes	0	0	0