

Product Requirements Document (PRD) – Retail Management System (Uzbekistan)

1. Обзор продукта и сфера использования

Описание: Данный продукт – облачная система управления розничной торговлей, ориентированная на малый и средний бизнес в Узбекистане. Система объединяет функции кассовой программы (POS), складского учёта, управления продажами, учёта финансов и базового CRM (работа с клиентами/лояльностью) в одном решении, аналогично тому, что предлагают существующие платформы как Billz и МойСклад ¹. Цель – предоставить **все необходимые инструменты в одной системе**, избавив бизнес от ручного ведения учёта и разрозненных приложений.

Сценарий использования: Продажа любых непродовольственных товаров в розницу (магазины одежды, электроники, бытовых товаров и пр.). Система **не нацелена на общепит** (кафе, рестораны) – для них есть отдельное решение. Логистические процессы (доставка, курьеры, маршрутизация) также вне сферы данного продукта. Основной упор – на **простоту и удобство** для продавцов, администраторов магазинов и владельцев бизнеса.

Примеры-конкуренты: - *BILLZ* – «all-in-one» платформа для ритейла, совмещающая POS, склад, CRM, e-commerce и аналитику ¹. Показывает направление, в котором должна развиваться наша система, включая омниканальность и аналитику. - *МойСклад* – облачная ERP для малого бизнеса, включающая продажи, закупки, склад, финансы и производство в одной системе ². Делает акцент на быстром запуске и понятном интерфейсе ³, а также на соответствии локальному законодательству (фискализация, маркировка) ⁴. Эти моменты учитываются в требованиях нашего продукта.

2. География и валюта

- **Рынок:** Запуск планируется в Узбекистане, с возможностью дальнейшей экспансии в соседние регионы. На первом этапе система заточена под особенности узбекского рынка.
- **Валюта:** Все операции ведутся в национальной валюте – узбекский сум (UZS). Мультивалютность в MVP не требуется (цены, отчёты и транзакции только в UZS).
- **Локальные стандарты:** Интерфейсы отображают формат денежных сумм, дат, чисел согласно местным стандартам (разделители тысяч, формат даты DD.MM.YYYY и т.д. для RU/UZ локалей).

3. Регуляторные требования: Фискализация и маркировка

- **Фискализация продаж:** Система должна соответствовать законодательству Узбекистана по онлайн-кассам. Каждый продажный чек необходимо **фискализировать** – отправить в налоговую через зарегистрированную онлайн-ККМ и напечатать фискальный чек с QR-кодом ⁴.
- Предусмотрен слой абстракции **FiscalProvider** с драйверами под конкретные модели онлайн-касс/фискальных принтеров, используемых в Республике Узбекистан.

- Поддерживаются основные операции: **открытие смены, регистрация продажи (чека), регистрация возврата и закрытие смены**. Эти операции ставятся в очередь для отправки на фискальный регистратор. В случае неудачи реализованы повторные попытки (retry) – например, если принтер офлайн, команда будет повторяться, чтобы чек обязательно фискализировался.
- **QR-код на чеке**: Фискальный чек содержит QR-код для проверки в налоговой (обязательно по требованиям законодательства).
- **Интеграция с E-POS**: В перспективе, интеграция с государственной системой E-POS (если требуется) для передачи данных о продажах одним кликом ⁵. Пока в MVP можно ограничиться работой через физические ККМ, но архитектура готова к API E-POS.
- **Маркировка товаров**: В Узбекистане введена национальная система цифровой маркировки товаров (AslBelgisi) для отслеживания подакцизных и определённых групп товаров.
- Система должна учитывать возможность работы с маркированными товарами. Например, при продаже сканировать DataMatrix код маркировки и передавать информацию в систему маркировки (через предусмотренные интеграции/API).
- **Интеграция AslBelgisi**: Планируется модуль интеграции с национальной системой маркировки, чтобы автоматизировать операции с маркированными товарами (проверка и списание марки при продаже, отражение кода марки в учётных документах) ⁴. В MVP это опционально; приоритет – заложить возможность, даже если полный функционал маркировки появится в обновлениях.
- **Коды ТАСНИФ и ИКПУ**: Для соответствия бухгалтерским требованиям, система должна позволять указывать коды классификаторов (например, ТАСНИФ, ИКПУ) для товаров и операций, чтобы документы (накладные, отчёты) соответствовали законодательным шаблонам ⁴. В MVP можно предусмотреть справочники для этих кодов.
- **Соответствие локальным стандартам печатных форм**: Все печатные формы (чеки, накладные, отчёты) должны соответствовать требованиям узбекского законодательства. Например, наличие необходимых реквизитов на кассовом чеке, использование узбекского языка для определённых полей (по требованию закона), формат дат и сумм и т.д. В системе должны быть шаблоны печатных форм, адаптированные под Узбекистан ⁴.

4. Многоарендность и иерархия данных

- **SaaS-модель**: Система разрабатывается как многоарендная (multi-tenant) SaaS-платформа. Один запущенный экземпляр сервиса обслуживает сразу несколько бизнесов (клиентов), изолируя их данные.
- **Изоляция данных**: У каждой записи в БД (товар, чек, склад и т.д.) есть идентификатор арендатора (`tenant_id`), соответствующий конкретному бизнесу. Реализована *Row-Level Security* (RLS) на уровне PostgreSQL, чтобы гарантировать, что запросы пользователя возвращают только данные его компании.
- **Структура предприятия**: Предусмотрена следующая иерархия сущностей:
- **Бизнес (компания)** – основной арендатор. Например, юридическое лицо или индивидуальный предприниматель, владеющий сетью магазинов.
- **Магазин (торговая точка)** – конкретная точка продаж/филиал компании. У бизнеса может быть множество магазинов (розничных точек).

- **Касса (терминал)** – рабочее место кассира внутри магазина. Может соответствовать одному компьютеру с POS-приложением. В каждом магазине может быть несколько касс (если несколько кассиров работают параллельно).
- **Разграничение по магазину:** Данные можно сегментировать по магазинам: складские остатки, продажи, кассовые смены привязаны к конкретному магазину. Пользователь с ролью, ограниченной магазином, видит только свой магазин. Управляющие или владельцы могут видеть данные по всем магазинам в бизнесе.
- **Единый аккаунт для нескольких бизнесов:** (Опционально) В будущем, если один пользователь управляет несколькими бизнесами, можно предусмотреть переключение контекста арендатора после логина. Но в MVP достаточно поддержки одного бизнеса на учетную запись владельца.

5. POS-клиент (точка продаж)

- **Платформа POS:** Кассовое приложение реализуется как **desktop-приложение** для Windows, созданное с использованием Electron (веб-технологии внутри настольного приложения). Это позволит использовать единый стек (JavaScript/HTML/CSS) и совместную кодовую базу с веб-версией, но при этом получить доступ к локальным ресурсам (файловая система, устройство печати) и устанавливать приложение на Windows-терминалы.
- **Оффлайн-режим:** Критическое требование – POS должен работать без постоянного интернета. Все основные операции (регистрация продаж, возвратов, открытие/закрытие смен) выполняются локально даже при отсутствии сети. Для этого в POS-клиенте используется **локальная база данных** (SQLite) для хранения настроек, каталога товаров и очереди транзакций.
- При появлении интернета происходит **синхронизация** с облаком: продажи и другие изменения, сделанные оффлайн, отправляются на сервер, и наоборот – обновления (например, новые товары или изменения цен, проведенные через веб-интерфейс) загружаются на клиент.
- Реализован механизм **Outbox/Inbox**: на POS изменения пишутся в локальный Outbox, а фоновые задачи отправляют их на сервер. Серверные обновления кладутся в локальный Inbox для применения на клиенте.
- Конфликты решаются по принципу *«last write wins»* для продаж (последняя операция считается актуальной, хотя обычно конфликты маловероятны для транзакций продаж) и по определенным правилам для справочников (например, если один товар параллельно редактировали на двух кассах, сервер может оставить последнюю версию или объединить изменения полей, если они разные).
- Такой подход аналогичен решениям конкурентов, где POS продолжает работу даже без интернета ⁶, обеспечивая непрерывность продаж.
- **Интерфейс и UX кассы:**
 - POS-интерфейс должен быть **максимально простым и быстрым** для кассира. Операции продажи должны выполняться **минимальным числом кликов/действий**.
 - **Сканирование штрих-кодов:** Поддерживается ввод товара сканером штрих-кодов – при сканировании курсор не должен отвлекать кассира (например, скрытое поле ввода), товар мгновенно добавляется в чек. Это ускоряет работу и снижает ошибки.
 - **Горячие клавиши:** Предусмотреть сочетания клавиш для основных действий (новый чек, поиск товара, применение скидки, выбор способа оплаты, печать копии чека и т.д.), чтобы опытные пользователи могли работать не трогая мышь. Особое внимание – работе с клавиатурой, т.к. многие кассиры предпочитают быстрые нажатия клавиш.

- **Макет экрана:** На экране кассы должно отображаться:
 - Список товаров в текущем чеке (с ценами, количеством, суммами и скидками) – обычно справа либо снизу, чтобы постоянно быть на виду.
 - Область поиска/сканирования товара и быстрые кнопки товарных категорий или популярных товаров – слева, для добавления позиций.
 - Итоги по чеку (итого, скидки, налоги) и выбранный способ оплаты – обычно внизу, крупно выделено.
 - Кнопки действий: Завершить продажу, Отмена/очистка чека, Применить скидку, Возврат и т.п. – крупные кнопки, удобные для тач-экрана.
- **Быстрые операции:** Добавить поддержку быстрого сканирования дисконтной карты клиента (или ввода номера телефона) для применения программы лояльности, быстрый ввод суммы наличными (например, кнопки "50,000 сум", "100,000 сум" для быстрого выбора сдачи).
- **Удобство для кассира:** Интерфейс кассы рассчитан на сенсорные экраны и различные разрешения, минимальный размер интерактивных элементов – 40px (для пальца). Цветовая схема – нейтральная, не утомляющая глаза при длительной работе, с возможностью ночного режима (dark theme) для работы вечером.
- **Поддержка оборудования:** POS-клиент должен интегрироваться с типичным торговым оборудованием:
- **Фискальный регистратор/принтер** – подключение к локальному драйверу или через `fiscal-gateway` для печати чеков. Приложение должно уметь печатать чеки на 80мм термоленте, используя шаблоны ESC/POS. Для Узбекистана – поддержка принтеров, сертифицированных в стране.
- **Сканер штрих-кодов** – обычно подключается как клавиатура (key input) или через USB/COM. Система должна обрабатывать скан ввода так же, как ручной ввод кода товара.
- **Денежный ящик** – открытие ящика синхронизируется с печатью чека (через команду принтера kick-out или напрямую, если поддерживается).
- **Электронные весы** – (опционально) для магазинов, торгующих весовыми товарами, предусмотреть интеграцию с весами: считывание веса товара напрямую или обработка штрих-кода с весовым кодом.
- **Платёжный терминал** – на первом этапе интеграция может быть не автоматизирована (кассир вручную пробивает сумму на банковском терминале). В перспективе, если банки предоставят API или через партнёров (Payme, Click), можно реализовать связь POS с терминалом по локальной сети для автоматической отправки суммы на оплату. Пока это обозначается как будущее улучшение.
- **Обновления и поддержка:** POS-клиент должен получать обновления (например, через авто-обновление Electron приложения) с минимальным участием пользователя. В случае отсутствия интернета, он продолжает работать на старой версии, а обновление применяется при следующем соединении.

6. Каталог товаров (товароучёт)

- **Структура каталога:** Система позволяет вести каталог товаров со следующими характеристиками:
- **Товар** – общее наименование (например, футболка), содержит основные свойства товара (наименование, категория, бренд, и т.д.).
- **Варианты товара (SKU)** – конкретные варианты/модификации, например размер и цвет. Каждый вариант имеет свой артикул или SKU, собственный штрих-код и остатки на складе. Пользователь может заводить товар либо как единицу без вариантов, либо с одним или двумя измерениями вариативности (цвет, размер, модель и т.п.).

- **Категории** – древовидная классификация товаров (например, Электроника → Мобильные телефоны). Товар может относиться к одной категории. Категории используются для навигации и для отчётов (продажи по категориям).
- **Единицы измерения** – поддержка продажи и учёта товаров в разных единицах (шт, кг, м и др.). Для каждого товара указывается базовая единица. Возможна привязка коэффициентов (например, коробка = 12 шт) для учета больших упаковок.
- **Штрих-коды** – хранение одного или нескольких штрих-кодов для каждого SKU. Например, международный EAN-13, внутренний код, код производителя – чтобы на кассе можно было сканировать любой из них.
- **Налог (НДС)** – возможность указать ставку НДС или другого налога для товара. В Узбекистане многие малые предприятия работают без НДС (UPSimplified tax), но система должна поддерживать учёт НДС для тех, кто на нём работает, или на будущее. Налоговая информация используется для расчёта налоговых сумм в чеке и для отчётности.
- **Оптовые и розничные цены** – заложить возможность нескольких цен для товара (например, розничная цена и оптовая цена, цены для разных типов клиентов). В MVP достаточно одной основной цены продажи, но архитектура готова к добавлению ценовых уровней.
- **Дополнительные атрибуты:** возможно, нужны поля как артикул, бренд, сезон, размерная сетка и т.п. Пока конкретные атрибуты вариативности не определены, но стоит предусмотреть механизм свойств (custom attributes) для гибкости, чтобы не кодировать жёстко.
- **Комплекты (наборы):** Предусмотрена возможность создать *комплект* – товар, состоящий из других товаров (набор, bundle). Продажа комплекта списывает несколько компонентов. В MVP это опционально (если быстро реализуемо). Это полезно для акций (2+1) или готовых наборов. Если сложно – переносим в будущее.
- **Серийные номера:** Если бизнес продаёт товар с уникальными серийными номерами (электроника, например), система должна позволить при продаже указать/сканировать серийник, и вести учёт проданных серийных номеров. Для MVP – опционально (требует доработки интерфейсов), но заложить возможность.
- **Импорт/экспорт каталога:** Для удобства начальной настройки предусмотреть импорт списка товаров из CSV/Excel. Например, конкурент МойСклад позволяет импортировать накладные и прайс-листы поставщиков из Excel ⁷ – аналогично, мы даём возможность загрузить файл с товарами (наименование, цена, штрих-код и т.д.) для ускорения внедрения.
- **Массовое редактирование:** Удобство использования требует функций массового обновления цен или категорий. В админ-панели веб-приложения можно реализовать массовое выделение товаров и применение операций (например, +10% к цене всех выделенных или переместить в другую категорию).
- **Поиск и фильтрация:** В списке товаров – быстрый поиск по названию, фильтры по категории, наличию на складе, атрибутам. Это поможет менеджеру быстро находить нужные позиции в большом каталоге.
- **UI/UX для каталога:** Табличный вид списка товаров с возможностью сортировки и кастомизации колонок. Например, отображение колонки «Цена», «Остаток на складе (по выбранному магазину или суммарно)». Поддержка виртуализации для очень длинных списков. Форма создания/редактирования товара – во всплывающем окне (drawer или modal) с разделением на вкладки, если много полей (основное, цены, остатки, изображения и т.д.).
- **Изображения товаров:** (Опционально) хранение изображений товаров. Полезно для идентификации товара на кассе или в электронной витрине. Можно реализовать загрузку картинок и отображение на POS (например, плиткой для выбора товара без сканера). В

MVP не критично, но как дополнительный полезный функционал – хотя бы в веб-админке отображать картинки, если привязаны.

7. Ценообразование и скидки

- **Цены:** У каждого SKU есть базовая цена продаж. Все цены хранятся и отображаются в суммах (UZS). Поддержка копеек (тийин) не нужна, т.к. сумма – валюта без дробной части.
- Можно заложить несколько ценовых типов (как упомянуто выше, розница/опт), но на старте достаточно одной цены.
- Возможность изменения цен: Интерфейс для обновления цен (массово или индивидуально). Возможно, реализовать документ "Переоценка" для смены цен с определённой даты (не в MVP, но перспективно).
- **Скидки:**
 - **Ручные скидки в чеке:** Кассир может применить скидку на отдельную позицию или на весь чек. Скидка может быть процентной (например, 10%) или абсолютной суммой. Необходимо учитывать ограничения (например, право кассира давать скидку не более N%, это можно через роли/права настроить).
 - **Правила скидок/акции:** В системе можно настроить простые правила промоакций. Например, скидка 5% на определённую категорию товаров на этой неделе, или "купи 2 получи 3-й бесплатно" – хотя последнее сложнее и может не попасть в MVP. Для MVP достаточно механизма фиксированных скидок.
 - **Купоны/промокоды:** Опционально, поддержка промокодов, которые кассир вводит, и получаются скидки. Если не успеваем – не включаем.
 - **Дисконтные уровни:** Если планируется работа с сегментами клиентов (обычный/VIP клиент), возможно, разные уровни скидок по уровню. Но поскольку лояльность выносится во внешний сервис Jowi Club, то ценовые правила могут приходить оттуда.
 - **Отчёт по скидкам:** Отдельно отмечаем, что все предоставленные скидки логируются – в отчётах управляющий сможет видеть, сколько скидок предоставлено, кем, и как это влияет на выручку.
- **Программа лояльности (интеграция с Jowi Club):**
 - Система **не будет хранить баллы и уровни** клиентов внутри себя. Вместо этого интегрируется с внешним сервисом **JOWi Club** (существующая платформа лояльности).
 - При сканировании карты лояльности или вводе номера телефона клиента, POS обращается к API Jowi Club для получения информации о клиенте: имя, текущий баланс баллов, уровень скидки и т.п.
 - **Начисление баллов:** После завершения продажи POS передает данные о покупке (сумма, товары) в Jowi Club, чтобы там начислились баллы или другие бонусы согласно правилам программы лояльности.
 - **Списание баллов:** Если клиент желает оплатить баллами или использовать скидку по уровню, POS должен отправить запрос/получить подтверждение от Jowi Club о доступном бонусе, применить скидку в чеке, и зафиксировать списание баллов через API.
 - **Журнал транзакций лояльности:** В нашей системе хранится только журнал того, какие транзакции отправлены в Jowi Club (на случай расхождений или для локальной аналитики), но расчет баллов, хранение информации о клиентах – на стороне Jowi Club.
 - **Простота для кассира:** интеграция должна быть максимально бесшовной – например, после сканирования карты система автоматически подтянет скидку. В UI кассы – индикация, что применена такая-то карта, скидка X% либо Y баллов будут списаны.
 - **Будущие промо-механики** (после MVP):
 - Сложные акции (набор правил, комбинации условий) – планируются в дальнейшем. Например, скидки по времени дня, персонализированные предложения, кросс-акции между категориями и пр. Архитектура должна позволять добавлять модуль промо-движка.

- **Подарочные сертификаты:** не в MVP, но помним как возможность – продажа/погашение gift cards.
- **Сезонные распродажи:** возможно, массовое снижение цен на период (можно через документ переоценки делать).
- **Ценообразование закупок:** (это больше к модулю закупок) – система хранит закупочную цену в приходных накладных для вычисления себестоимости, но ценообразование (наценка) пока вне автоматизации – пользователь сам устанавливает цены. В будущем можно добавить рекомендации (например, подсказка оптимальной цены на основе наценки).

8. Складской учёт и движение товаров

- **Управление остатками:** В системе фиксируются **остатки товаров** по каждому складу (магазину). При продажах и возвратах остатки автоматически списываются/приходятся. У пользователя всегда должна быть возможность видеть актуальный остаток по каждому SKU в каждом магазине в реальном времени (при условии синхронизации с POS). Важное требование – **предотвращение отрицательных остатков:** система либо блокирует продажу, если товара нет на складе, либо предупреждает и позволяет продать в минус (опция для администратора). В идеале, для розницы – не продавать того, чего нет.
- **Складские документы:** Для операций движения предусмотрены документы:
- **Приходная накладная** – оформление поступления товара от поставщика. Пользователь (кладовщик или менеджер) вводит накладную: выбирает поставщика, список товаров, количество, закупочные цены, дата прихода. Проведение накладной увеличивает остатки на складе и фиксирует задолженность перед поставщиком (если закуплено в кредит).
- **Перемещение** – документ для перемещения товара с одного склада на другой (например, со склада компании в конкретный магазин, или между магазинами). Документ указывает склад-отправитель, склад-получатель, товары и кол-ва. Проведение списывает остаток с отправителя и приходит получателю.
- **Возврат поставщику** – оформляет возврат бракованного или лишнего товара обратно поставщику. Списывает товар со склада, уменьшает задолженность перед поставщиком (или формирует долг поставщика перед нами, если уже было оплачено).
- **Акт списания** – (если требуется) документ для списания товара по причине порчи, кражи, истечения срока годности. Уменьшает остатки и отражается в учёте как расход/убыток.
- **Инвентаризация** – документ пересчета остатков. Кладовщик проводит инвентаризацию (полную или частичную): сканирует/вводит фактическое наличие товара, сравнивает с данными системы, и документ фиксирует расхождения (излишки или недостачи). Проведение инвентаризации корректирует остатки на складе и может генерировать соответствующие списания/оприходования для учёта.
- Каждый документ имеет номер, дату, ответственного и проводки в учётной системе. Проведенные документы нельзя задним числом редактировать (только сторно или исправления новым документом), чтобы сохранять целостность истории учета.
- **Метод учёта себестоимости:** Для вычисления себестоимости проданных товаров и оценки складских запасов используется метод **FIFO** (First In, First Out) по партиям:
- Система отслеживает **партии** товара: каждая приходная накладная создает партии с определённой ценой закупки. При продаже товара ему сопоставляется цена из первой нереализованной партии (FIFO). Таким образом можно рассчитать валовую прибыль (разница между ценой продажи и себестоимостью).
- В MVP достаточно ориентироваться на FIFO, без поддержки сложных методов (LIFO, средняя и т.д.). ClickHouse аналитика может рассчитывать прибыль, опираясь на эти данные.

- **Операции списания при продаже:** Каждый чек при проведении списывает товар с конкретного склада магазина. Если в магазине несколько складов (например, витрина и запас), MVP рассматривает один основной склад на магазин. Продажа списывает с основного склада.
- **Контроль запасов:** Менеджер должен иметь инструменты контроля:
- **Отчёт остатки на складе** – список товаров с текущим количеством по складам, возможность фильтра по критически низким остаткам.
- **Неснижаемый остаток:** опционально, возможность задавать для товара мин. уровень запаса. Тогда отчёты/интерфейсы помечают, что товар нужно заказать, если достиг минимума. (В МойСклад есть автопополнение до неснижаемого остатка ⁷ – у нас пока без авто, но флаг/отметка).
- **Сроки годности/партионный учёт:** Если работаем с товарами с expiration date (например, фармацевтика), нужен учёт по срокам годности. Это специфично, можно отложить. В MVP можно указать, что поддержка expire date возможна через партии (атрибут партии – дата).
- **Интеграция с закупками и финансами:** Приходные накладные влияют на расчёты с поставщиками (долги), а возвраты поставщику – наоборот. Эти связи отражаются в модуле Финансы (см. далее).
- **Многоскладовость:** Если бизнес желает вести несколько складов в пределах одного магазина (например, фронт-склад и основной склад), система должна позволять это настроить, но можно упростить: один склад = один магазин в MVP, а межскладовое перемещение – между магазинами. В будущем – добавить поддержку множества складов на магазин.
- **UI для склада:**
- Веб-интерфейс: Раздел *Склад* со списком документов (приходы, перемещения, инвентаризации). Реализовать удобный **мастер создания документов** – например, пошаговый «Wizard» для инвентаризации (выбор склада, список позиций для пересчета, ввод фактов, подтверждение).
- Поддержка **сканера**: при работе с приходом или инвентаризацией, возможность сканировать товары для добавления строки или быстрого поиска позиции.
- **Экспорт/печать:** Документы должны печататься на А4 (как отчёт или унифицированная форма накладной) – в узбекском/русском языках, с нужными полями (номер, дата, организации, ИНН, код товара, кол-во, цена, подписи и печати – если требуется). Можно реализовать генерацию PDF на сервере.
- **Циклические инвентаризации:** вместо ежегодной полной, система должна поддерживать частые частичные (например, каждый день определённую категорию товаров пересчитывать). Для этого – журнал инвентаризаций, чтобы пользователь мог планировать и отмечать что когда пересчитано.
- **Простота использования:** Несмотря на богатый функционал, интерфейс складского учёта должен быть понятен даже неопытному пользователю. Например, при создании приходной накладной – автозаполнение текущей даты, удобный поиск по поставщикам и товарам, проверка на дублирование позиций, предупреждение о превышении кредитного лимита поставщика (если введём учёт, но, возможно, не сейчас).

9. Продажи и операции в POS

(Этот раздел дополняет раздел о POS-клиенте, фокусируясь на процессах оформления продажи и связанных действиях.)

- **Продажа (чек):** Основная транзакция – розничная продажа:

- Кассир сканирует или выбирает товар, он добавляется в список чека с указанной ценой. По умолчанию цена берется из текущего прайс-листа (обычно розничная цена из каталога).
- Кассир может изменять количество (например, если покупают 2 штуки одного товара). Если товар измеряется весом – интерфейс позволяет ввести вес вручную или получать от весов.
- Применение скидок: кассир может выбрать позицию и ввести скидку (%) или сумму, если имеет право. Либо применить скидку на весь чек (например, по дисконтной карте – это делается автоматически через Jowi Club, или вручную по решению администратора).
- **Добавление клиента к продаже:** Если покупатель предоставляет карту лояльности или просит накопить баллы, кассир вводит клиента в чек (поиск по номеру телефона, сканирование штрих-кода карты и т.д.). Система отображает имя клиента, его статус, доступные баллы/скидку. После этого продажи могут учитываться на этого клиента (для CRM аналитики) и применяться правила лояльности.
- **Итоги:** По мере добавления позиций система считает итоги – промежуточный итог, скидки, налоги (если выделяются), итоговую сумму к оплате. Эти поля обновляются в реальном времени и видны кассиру.
- Когда клиент готов оплатить, кассир нажимает «Оплата» и выбирает способ оплаты (или несколько, если частями).
- **Фискализация:** После подтверждения оплаты чек отправляется на фискальный принтер для регистрации ⁴. Кассиру отображается состояние (успех/ошибка). В случае ошибки фискализации – система должна выдать понятное сообщение (например, «Ошибка связи с кассой»), предложить повторить. В оффлайн-режиме, если фискальный аппарат не подключен к интернету, чек всё равно печатается (с отметкой офлайн, если таковое разрешено законодательством), и данные отправляются когда связь появится.
- **Печать чека:** POS печатает клиентский чек (ширина ~58-80 мм, моноширинный шрифт, 48 символов в строке). В чеке – наименование магазина, адрес, ИНН, дата-время, список товаров (название, кол-во, цена, сумма, налог), итог, сумма НДС (если есть), скидка отдельной строкой, QR-код, фискальные реквизиты (номер фискального документа, заводской номер кассы и пр.). Чек печатается на локальном принтере сразу же после проведения.
- **Хранение данных продажи:** Продажа сохраняется локально (в SQLite) сразу при проведении, присваивается временный ID. Если интернет есть – отправляется на сервер и получает облачный ID. Если интернета нет – ждет в очереди отправки.
- **Возврат товара (refund):** Необходим механизм оформления возврата:
- Возврат может осуществляться **по чеку**: кассир находит оригинальный чек (через поиск по номеру, дате или сканированию QR-кода чека, если наш), затем выбирает позиции и количество к возврату. Система создаёт отрицательную транзакцию с пометкой «Возврат», и также отправляет её на фискальный регистратор как чек возврата ⁵.
- Если оригинальный чек не найден (например, утерян или система его не находит – хотя если чек выбит в системе, он должен быть), то возврат можно оформить вручную через отдельный документ «Возврат без чека» – кассир сканирует товар, указывает количество к возврату, возможна причина возврата. Такой возврат тоже фискализируется (отдельный тип).
- При возврате происходит **приход товара на склад** (то есть увеличение остатка) и **возврат денег покупателю** (наличными или отмена транзакции по карте на терминале). Финансово – уменьшается выручка, корректируется прибыль.
- Учитывать ограничения законодательства (например, возврат в тот же день может аннулировать продажу, а возврат после смены – отдельной транзакцией). Разрешить частичный возврат.

- **Отмена продажи:** Если кассир начал пробивать чек, но покупатель передумал до оплаты, достаточно нажать «Отмена» – текущий чек просто очищается (не сохраняется). Если же чек уже оплачен/напечатан, его отмена возможна только через возврат.
- **Открытие/закрытие смены:**
- **Открытие смены:** При начале работы кассир (или старший) должен открыть кассовую смену на терминале. Вводится начальный остаток наличности в кассе (сколько денег было в ящике на начало дня). Смена открывается через команду фискальному регистратору (открытие фискальной смены) и фиксируется в системе (создается объект «Смена» с меткой времени и ответственным).
- **Закрытие смены:** В конце дня (или при пересменке) кассир выполняет закрытие смены. Система должна напечатать **Z-отчёт** – итоговый отчет по смене (количество чеков, сумму наличных, сумму безналичных, возвраты, итоговую выручку, суммы налогов). Фискально – происходит закрытие смены на кассе. В системе сохраняется запись с итогами.
- При закрытии смены кассир подсчитывает фактические деньги в ящике и вводит сумму. Система сравнивает с ожидаемым остатком по данным (начальный + продажи - возвраты - внесения + изъятия) и показывает **разницу (излишек/недостача)**. Это сохраняется в отчёте.
- Без закрытия предыдущей смены невозможно открыть новую (контролируется).
- **Управление наличностью в течение дня:**
- **Внесение денег:** Операция, когда в кассу кладут наличные (например, размен, либо инкассация – хотя инкассация скорее изъятие). Кассир выбирает «Внесение», вводит сумму, и эта сумма добавляется к кассе (не является выручкой, а просто увеличивает наличные).
- **Изъятие денег:** Операция выемки из кассы (например, передают деньги в офис или инкассаторам). Указывается сумма, причина. Эта сумма вычитается из остатков кассы. В Z-отчёте такие операции должны учитываться отдельно.
- Эти операции не фискализируются как чеки продажи, но могут требовать просто записи в журнал. Однако для полноты учёта кассы они важны.
- **Несколько типов продаж:** Пока ориентируемся на розничную продажу со 100% оплатой. Продажи с отсрочкой платежа, счётами и т.п. – не в MVP (это больше B2B функция). В будущем можно добавить **счёт на оплату** (invoice) и **продажу в кредит** (для постоянных оптовых клиентов), но сейчас нет.
- **Удобство для кассира:**
- **Интуитивность:** Процесс продажи должен быть обучаем «с нуля» за 10-15 минут. Конкуренты заявляют «старт за 15 минут» ⁸ – у нас аналогично: понятные кнопки, подсказки на экране («Отсканируйте или введите код товара»).
- **Минимум ошибок:** Система предупреждает о нетипичных ситуациях: продажа без открытия смены, попытка продать товар при нулевом остатке, повторное сканирование одного товара (возможно, просто увеличивает количество, что правильно), превышение допустимой скидки.
- **Скорость:** Время от начала пробития чека до печати – ключевой показатель. Будем стремиться оптимизировать поиск товаров (быстрые запросы к локальной БД), рендеринг списка (виртуализация для длинных чеков), и задержку печати (быстрый драйвер печати, параллельная отправка на фискализацию).
- **Обучение:** При первом запуске POS можно отобразить режим обучения/демо: подсветка основных кнопок, краткая инструкция (например, интерактивный туториал, который можно пропустить).
- **Продажи через другие каналы:** В MVP фокус на офлайн-магазине. Продажи через интернет-магазин или маркетплейс будут учитывать как интеграции (см. раздел Интеграции), но сам процесс оформления интернет-заказа вне системы (поступает уже как готовый заказ, возможно, мы его проводим как отдельный тип документа).

10. Финансы и кассовые операции

- **Кассовая книга:** Система должна автоматически формировать **кассовую книгу** по каждому магазину/кассе – т.е. полный список всех движений наличности за день/смену:
- Начальный остаток (при открытии смены).
- Поступления от продаж (каждый наличный чек суммой).
- Возвраты наличные (вычитаются).
- Внесения/изъятия.
- Итог на конец дня (который должен совпасть с фактом).
- Эти данные выводятся в отчёте (Z-отчёт) и сохраняются для бухгалтерского учёта предприятия. Формально, для небольших бизнесов возможно ведение упрощённое, но если потребуется – экспорт кассовой книги за период для бухгалтерии.
- **Учёт доходов и расходов:** Внутренняя подсистема учёта (бухгалтерии) записывает **проводки** при каждом финансово значимом событии:
- Продажа: Дт "Касса" / Кт "Выручка от продаж" (грубо говоря, и списание товара: Дт "Себестоимость продаж" / Кт "Товар на складе").
- Закупка: Дт "Товар на складе" / Кт "Расчёты с поставщиком".
- Оплата поставщику: Дт "Расчёты с поставщиком" / Кт "Деньги".
- И т.д. – полный бухгалтерский учёт пока не обязателен, но **двойная запись** заложена для последующего получения финансовых отчётов (баланс, P&L) внутри системы.
- **Важно:** Мы не стремимся в MVP выдать полностью регламентированную бух. отчетность, но движение средств и товаров взаимосвязано и должно консистентно учитываться.
- **Взаиморасчёты с поставщиками и клиентами:**
- **Поставщики:** У каждого поставщика можно отслеживать баланс задолженности. Когда проводим приходную накладную с типом оплаты "Отсрочка" – система увеличивает долг перед поставщиком. Когда фиксируем факт оплаты поставщику (через отдельную операцию «оплата поставщику» или отмечая накладную оплаченной) – долг гасится. Пользователь (бухгалтер/менеджер) может видеть список задолженностей по всем поставщикам.
- **Клиенты (долги покупателей):** В рознице, как правило, нет продаж в долг. Но если вдруг предусмотрена работа с некоторыми постоянными клиентами по постоплате (например, корпоративный клиент покупает товары и оплачивает раз в месяц по счету) – можно вести учёт дебиторской задолженности аналогичным образом. В MVP, вероятно, не востребовано, но можно иметь базовый функционал: документ "Реализация в кредит" и "Платёж от клиента".
- **Оплаты и платежи:**
- **Наличные:** фиксируются по кассе (как описано).
- **Эквайринг (карты):** Каждая продажа по безналу отмечается как оплата картой. В кассовой смене они суммируются отдельно. Взаимодействие с банком может быть ручным (кассир сам провёл терминал, результат просто отметили). Должна быть возможность сверять суммы терминала и системы в конце дня. Возможно, в отчёте – отдельная секция «безнал по терминалу».
- **Переводы (электронные кошельки):** В Узбекистане популярны платежи через Click, Payme и др. Если магазин принимает оплату через сканирование QR кода (клиент сканирует и платит) – кассир по факту просто отмечает чек как оплачен *по переводу*. Желательно, чтобы система могла генерировать QR для оплаты на кассе, но это требует интеграции с Payme/Click API. В MVP, скорее всего, это вне скопа, поэтому просто учитываем как тип оплаты "Перевод/QR оплата" (вручную).
- **Смешанная оплата:** Поддерживаем частичную оплату разными способами. Например, клиент часть наличными, часть картой. Интерфейс оплаты позволяет добавить несколько

строк оплаты (тип и сумма). Контролируем, чтобы общая сумма оплаты равнялась итогу чека.

- **Возврат оплаты:** При возврате товара – если оплата была наличными, кассир выдаёт наличные из кассы (и это отражается в книге как расход). Если оплата была по карте, обычно возврат делается на карту – тут кассир отмечает, что возврат по карте (и на терминале проводит сторно). Система должна учесть это, чтобы корректно отражалось в смене (уменьшение безнала).
- **Без интеграции с внешней бухгалтерией:** Подчеркнём, что интеграция с 1С или другими бухгалтерскими системами **не нужна** в рамках MVP. Идея продукта – пользователь получает все необходимые отчёты из нашей системы, чтобы не вести отдельный бухгалтерский учёт.
- Однако, для продвинутых пользователей, можно **экспортировать данные:** например, выгрузка в Excel/CSV регистра продаж, движений по складу или оборотно-сальдовой ведомости. Возможно, PDF-отчёты баланса/прибыли. Но сам факт – закладываем, что наша система *сама ведёт учёт всех приходов и расходов и отображает все транзакции и отчёты по ним.*
- **Проводки и аудит:** Все финансовые действия порождают **проводки** (как упомянуто) и сохраняются в неизменяемом журнале аудита. Это важно для доверия к системе: любое изменение или задним числом правка – либо невозможна, либо фиксируется. Например, если администратор решит удалить чек (чего делать нельзя, возврат вместо удаления), то в системе это должно отразиться. В MVP удаление финансовых документов лучше запретить совсем.
- **Разделение по юр.лицам:** Если бизнес имеет несколько юридических лиц, могло бы потребоваться разделять фин.учёт. В MVP предполагаем, что один бизнес (tenant) = одно юрлицо, и все чеки и документы выписываются от его имени. Если нужно другое – можно завести отдельный аккаунт. (В дальнейшем можно развить, но пока не усложняем.)
- **Отчёты финансовые:**
 - **Отчёт о продажах** (выручка) за период – сколько наличных, сколько безнала, возвраты, итог.
 - **Отчёт о прибыли** – валовая прибыль = выручка - себестоимость, по периодам/по товарам.
 - **Движение денежных средств** – похоже на кассовую книгу, но объединяет все способы оплаты: сколько денег поступило (наличные, на счёт – если фиксируем платежи) и сколько выбыло.
 - **Баланс взаиморасчётов с поставщиками** – список поставщиков и сколько мы должны (и просрочка, если есть).
 - **Баланс по кассе/счёту** – сколько денег в кассе фактически и по учёту.
 - **Прочие доходы/расходы:** если магазин учитывает расходы (аренда, зарплата) – это вне нашего скопа, т.е. управленческий учёт расходов пока не делаем. Счета-фактуры, НДС-отчётность – тоже вне скопа (так как бух интеграции нет).

11. Отчёты и аналитика

- **Базовые отчёты MVP:**
 - **Продажи по периодам:** отображение объёма продаж (сумма и количество чеков) по дням, месяцам или выбранному периоду. Можно график тренда. Фильтры по магазину, кассиру.
 - **Продажи по товарам:** топ продаж по товарам или категориям. Например, за выбранный месяц – список товаров с количеством проданных и суммой. Выявление ходовых товаров.
 - **Продажи по часам:** например, поминутная или почасовая выручка в течение дня (полезно для анализа пиковых нагрузок, чтобы планировать персонал).

- **Эффективность сотрудников:** сколько каждый кассир/продавец сделал продаж (кол-во чеков, сумма выручки, средний чек). Помогает при расчёте KPI или бонусов сотрудникам.
- **Остатки на складе:** текущий остаток товаров на складе (с возможностью фильтрации по складу/магазину и наличию, например «показать товары, где остаток >0» или «товары, заканчиваются»).
- **Движение товаров:** отчёт о приходах/расходах товара за период – сколько поступило, сколько продано, списано, итоговый остаток (для инвентаризации и контроля).
- **Финансовый отчёт (упрощенный P&L):** за период – выручка, себестоимость, валовая прибыль, расходы (если учитываем списания как убыток), чистая прибыль. Без сложной амортизации и т.д., только операционные показатели.
- **Интерактивность:** Отчёты доступны в веб-админке. Пользователь выбирает период, при необходимости – фильтры (магазин, категория, сотрудник и пр.). Отчёты представляются **в виде таблиц и графиков:**
 - Таблицы можно выгрузить в CSV/Excel.
 - Графики (диаграммы) – столбчатые, линейные или круговые (для долей категорий, например). Используем библиотеку Recharts для отрисовки, легенда и подсказки включены [9](#) [10](#).
- **Детализация:** В отчетах желательно поддерживать drill-down – напр., кликнув на день в отчёте продаж по дням, увидеть детализацию по чекам этого дня.
- **Большие данные:** Предполагается, что с ростом числа транзакций простой запрос по PostgreSQL может тормозить. Поэтому архитектурно:
 - Основная БД – Postgres – хранит все транзакции (чеки, документы).
 - Настроен **CDC (Change Data Capture)** через Debezium + Kafka, который транслирует изменения (продажи, движения) в аналитическую базу **ClickHouse**.
 - В ClickHouse поддерживаются агрегированные материализованные представления или просто быстрые запросы по большим таблицам. Отчёты могут строиться путем запросов к ClickHouse, что даёт мгновенные результаты даже на миллионах строк.
 - В MVP можно обойтись Postgres для простоты, если нагрузки невелики. Но закладываем инфраструктуру, чтобы добавить ClickHouse без переписывания логики отчётов (например, используя абстракцию репозитория, который может подменяться).
- **Витрина данных:** Реализуем концепцию «витрина отчётов» – модуль, куда легко добавлять новые отчёты. Т.е. новый отчёт – это, например, новый SQL запрос (или несколько) + шаблон отображения. Разработчики и даже опытные пользователи смогут расширять пакет аналитики.
- **Метрики для SLA:** Встроим некоторые системные отчёты/метрики:
 - % успешной синхронизации офлайн-продаж (чтобы отслеживать проблемы с офлайн данными).
 - Среднее время оформления чека (метрика UX, рассчитывается как время между открытием и закрытием чека).
 - Количество сбоев печати или ошибок по кассам.
 - Эти вещи больше внутренние, но могут быть полезны для команды разработки и для бизнес-аналитиков сети.
- **Сравнительный анализ:** В будущем, возможно, предложим бизнесам сравнение с анонимными данными по рынку (как делает, например, Square – бенчмарки). Пока не планируется, но если соберётся большая база, можно добавить инсайты.
- **UI/UX отчётов:**
 - Раздел *Отчёты* в веб-админке с перечнем доступных отчётов. Каждому отчёту – страница или модальное окно с параметрами (фильтры) и сгенерированными графиками/таблицами.

- Предусмотреть **дашборд** на главной странице админки: ключевые показатели (выручка сегодня, топ товар, продажи по магазинам) в виде карточек KPI и мини-графиков. Администратор, зайдя в систему, сразу видит общую картину.
- Важно, чтобы отчёты **были понятны**: использовать ясные названия метрик, пояснения при наведении. Возможно, сделать шаблоны экспорта в PDF, чтобы владелец мог распечатать и предоставить инвесторам/налоговой при необходимости (упрощённо).
- **Расширяемость**: Добавление новых отчётов должно быть несложным – система спроектирована масштабируемо, как витрина. Это позволит быстро реагировать на запросы пользователей, добавляя, например, отчёт по оборачиваемости товара, по эффективности акций и т.д. (которые сейчас не первоочередны).

12. Пользователи и роли доступа

- **Ролевая модель (RBAC)**: Доступ к функционалу определяется ролями пользователей. В MVP достаточно предусмотреть **базовые роли** (защиты в системе), с возможностью добавлять новые при необходимости:
- **Администратор** – обычно это владелец бизнеса или ИТ-администратор. Имеет полный доступ ко всем данным своего бизнеса. Может добавлять/удалять пользователей, менять настройки системы, видеть и редактировать все справочники, просматривать все отчёты, финансовые данные.
- **Управляющий/Менеджер** – роль для директора магазина или операционного менеджера. Может видеть продажи и остатки своего магазина (или всех магазинов, если указано), управлять товарным каталогом, оформлять приход/списание товаров, видеть отчёты по своему магазину, просматривать данные сотрудников своего магазина. Не имеет доступа к настройкам бизнеса (оплате подписки, например, если такое будет) и, возможно, к конфиденциальным финансовым отчетам всего бизнеса.
- **Кассир/Продавец** – минимальные права: работа с POS (оформление продаж, возвратов), открытие/закрытие смены. **Не имеет доступа** к разделам справочников, к отчётам (кроме своих продаж за смену, возможно), к финансовой информации. Может видеть только свой интерфейс кассы и, например, историю своих чеков.
- **Кладовщик** – роль для сотрудника склада/завсклада. Может создавать приходные накладные, инвентаризации, перемещения. Видит остатки. Не имеет доступа к продажам и финансам. (В маленьком магазине эта роль может совмещаться с менеджером.)
- **Бухгалтер** – (опционально в будущем) если подключится бухгалтер, можно дать роль с просмотром всех документов, формированием отчетов, но без права оформлять продажи или изменять настройки. В MVP можно обойтись без отдельной роли бухгалтера, т.к. владелец или менеджер сам просматривает финансы.
- **Гибкость**: Роли реализуются через таблицу прав (permissions). Для MVP достаточно зафиксировать разрешения, но кодом заложим, что новые роли можно добавить (например, в БД таблица Roles, Permissions, UserRoles, чтобы потом через интерфейс настраивать). Пока UI для кастомизации прав может не быть, но архитектурно – RBAC, не hardcode.
- **Разделение по магазинам**: Пользователю можно назначить, к каким магазинам у него доступ. Например, управляющий может видеть 1 магазин из 5. Кассир обычно привязан к конкретному магазину (и, возможно, к конкретной кассе, хотя кассир может сесть за любую кассу своего магазина).
- **Аутентификация**: Используем современный подход Auth - например, Auth.js или сервис Clerk. Авторизация (проверка прав) – на сервере (через NestJS Guards, Middleware) и дублируется на клиенте для условного рендера элементов UI.
- Сессии – JWT токены, в которых зашит `tenant_id` и роль/права. Это будет основой для бекенда, чтобы по каждому запросу знать, кто и от какого бизнеса обращается.

- **2FA:** Для безопасности можно включить опциональную двухфакторную аутентификацию для важных аккаунтов (через приложение аутентификации). В MVP – не обязательное требование, но хорошо бы иметь возможность, т.к. финансовые данные важны.
- **Audit Log:** Как упомянуто, система ведёт аудит: важные действия (создание/удаление пользователей, изменение прав, изменение настроек системы, удаление документов, крупные возвраты) логируются с указанием пользователя, времени, и доступно администратору в разделе "Журнал действий". Это повышает доверие и безопасность.
- **Пароли и безопасность:** Требовать надёжные пароли при регистрации пользователей. Возможность сброса пароля (email/SMS отправка ссылки). Так как система SaaS, обеспечение безопасного хранения (хеширование паролей, защита от brute force – лимиты попыток).
- **UI доступа:**
 - Администратор через веб-интерфейс имеет раздел "Пользователи", где может приглашать новых пользователей (например, по email), назначать им роль и магазин. Также можно временно блокировать пользователя (уволился сотрудник – блокируем доступ).
 - Для простоты, можно объединить понятия *сотрудник* и *пользователь системы*. Каждый сотрудник, который использует систему, имеет логин. Если есть сотрудники, которые не работают в системе (например, грузчики), их можно не заносить.
 - При входе пользователь выбирает, возможно, магазин (если у него доступ к нескольким магазинам и он запускает POS, важно выбрать нужный контекст).
- **Многофакторность контекста:** Убедиться, что `tenant_id` всегда присутствует во всех ключевых запросах. Это решается автоматически, если JWT содержит и backend фильтрует.
- **Возможность расширения:** Добавление новых прав (например, право "просмотр себестоимости" – может быть только у админа, кассирам не показывать себестоимость товаров). На будущее – гибкая матрица разрешений, но MVP минимально.

13. Интеграции внешние

- **Программа лояльности – JOWi Club:** (См. **Ценообразование и скидки** раздел) – основная интеграция MVP. Все аспекты работы с бонусными баллами и картами лояльности выполняются через REST API Jowi Club:
- Необходимо настроить безопасное соединение (API key или OAuth) между нашим приложением и Jowi Club.
- Обработать вебхуки от Jowi (например, если клиент проверяет баланс через приложение – не критично для нас, или если нужно уведомление о начислении).
- **Отказоустойчивость:** Если Jowi Club недоступен, кассир всё равно должен иметь возможность пробивать чеки (просто без применения скидки, либо с резервным офлайн-режимом лояльности – это сложно, скорее, просто предупреждение).
- **Маркетплейсы / e-commerce:** В MVP **не реализуем**, но **закладываем возможность**:
- Создаём абстракцию или модуль *Integration Hub*, куда можно добавлять коннекторы к внешним системам. Например, коннектор к **Uzum Market, Wildberries, Ozon** и т.д. (MoySklad уже имеет готовые интеграции с Uzum и WB ¹¹).
- В будущем это позволит пользователям синхронизировать товарный остаток с онлайн-площадками, выгружать товары на маркетплейсы, загружать заказы.
- Важное требование – архитектура API должна позволять безопасно обращаться к сторонним сервисам и хранить credenшлы (например, токены API маркетплейсов).
- В MVP ограничимся проектированием интерфейсов: например, таблица `integration` с полями `type` (marketplace name) и токенами, базовые endpoints.
- **Онлайн-магазин:** Если клиенты захотят собственный интернет-магазин, возможны пути:
- Предоставить простой встроенный онлайн-витрину (как делает BILLZ – у них, видимо, есть e-commerce модуль ¹). Это вне MVP – требует фронтенд магазин, корзину и т.п.

- Либо интегрироваться с популярными CMS (WooCommerce, Shopify) – тоже позже.
- На уровне MVP можно только отметить, что заказы из интернет-каналов можно будет ручным импортом оформлять (менеджер получил заказ – руками завёл в систему как расходную накладную на клиента).
- **Платёжные сервисы:** В перспективе:
- **Payme, Click** – локальные платёжные системы. MoySkлад, например, упоминает регистрацию продаж через QR-коды Click/Payme ⁶. Мы можем интегрироваться с их API, чтобы генерировать QR для оплаты и получать подтверждение платежа. В MVP, скорее всего, не реализуем, но **учитываем, что это желательная фича** для рынка.
- **Банковские эквайринги** – возможно, сотрудничество с Payme (который предоставляет терминалы) или TBC Bank (партнёры Billz) для интеграции. **Пример:** TBC Bank приобретает Billz, обещая интеграцию с банковскими сервисами (кредиты, эквайринг) ¹². Наш продукт тоже может в будущем получить плюсы от партнерств с финтехом.
- **Национальные сервисы:**
- **AslBelgisi** – см. Маркировка выше. По сути, это внешняя система, куда надо посылать данные о маркированных товарах. Интегрируем через API (вероятно, REST или SOAP). Сложность – в требовании в реальном времени при продаже проверять код. Это учтём в POS (доп. шаг сканирования марки).
- **Электронные счета-фактуры (E-SF):** Если для оптовых или B2B продаж понадобится выписывать счет-фактуры и отправлять их в налоговую электронно, можно подумать об интеграции с системами вроде Didox (электронный документооборот) – MoySkлад уже анонсировал такое ¹³. Но для чисто розницы (B2C) это не нужно. Поэтому MVP без этого.
- **Фискальные регистраторы:** (Повторим) – интеграция с оборудованием выносится в отдельный сервис `fiscal-gateway`. Для реализации драйверов под разных производителей (Штрих-М, Атол, НСР и др. – какие доступны в Узбекистане). С POS и бэкенда общение с этим сервисом через HTTP или gRPC – т.е. POS отправляет JSON команду `registerSale` → `fiscal-gateway` → принтер, получает ответ. Такой подход упростит сертификацию и поддержку (можно обновлять драйверы отдельно).
- **Мобильное приложение:** В MVP мобильных клиентов нет. Но, возможно, интеграция с **мобильным приложением Jowi** (если у них есть клиентское приложение или приложение для владельцев). Например, получение push-уведомлений о выручке за день. Но конкретных требований нет, поэтому просто имеем ввиду, что REST API нашего сервера должен позволять реализовать mobile app в будущем.
- **API для клиентов:** Некоторым продвинутым пользователям может понадобиться API для их внутренних систем. Планируем после MVP опубликовать API (на основе тех же tRPC контрактов) для основных операций: получить остатки, создать продажу (например, если они свою кассу хотят прикрутить). В MVP — не приоритет, но архитектура (NestJS + tRPC-like) фактически уже предоставляет API, нужно будет лишь документировать.
- **Шина событий:** За кулисами NATS может использоваться для трансляции событий (продан товар – событие, которое может слушать модуль интеграции). Это внутренняя архитектурная деталь, чтобы новые интеграции добавлялись подпиской на события, а не переписыванием логики продаж.

14. Локализация (интернационализация)

- **Поддерживаемые языки:**
- **Русский** – основной язык интерфейса по умолчанию (большая часть бизнес-пользователей им владеет).
- **Узбекский** – обязателен, т.к. государственный язык и многие сотрудники/клиенты предпочитают его. Нужно перевести весь интерфейс на узбекский (латиница или

кириллица – лучше оба варианта предусмотреть; официально сейчас узбекский в латинице, но многие привыкли к кириллице. Вероятно, используем латиницу).

- **Переключение языка:** Пользователь должен иметь возможность выбрать язык интерфейса. Либо при входе (в форме логина), либо внутри приложения (например, выпадающий список языков в меню профиля). Можно хранить предпочтение языка в профиле пользователя.
- **Перевод данных:** Справочники (товары, категории) могут быть заведены на одном языке. В MVP не будем требовать дублировать названия товаров на двух языках, это усложняет. Скорее всего, пользователи выберут либо русский, либо узбек для ввода данных. Интерфейс при этом переводится, но названия товаров – как введены. (В будущем, для сетей, может понадобиться мультиязычный каталог, но не сейчас).
- **i18n технология:** Используем i18next (или аналог) с поддержкой **неймспейсов** для модулей (`common`, `pos`, `inventory`, `finance`, `reports` и т.д.), чтобы избежать конфликтов ключей и упростить перевод. **Все текстовые константы** в коде должны быть вынесены в словари.
- **Форматы:** Библиотеки локализации должны форматировать валюту, даты и время на выбранном языке. Например, дата на узбекском может отображаться как `12-Oktabr 2025` или т.п. Надо проверить предпочтения.
- **Правила языка:** Учесть направления текста (оба языка – лев-to-right, проблем нет), множественное число, падежи – i18next может с этим помогать, но для MVP можно простые фразы.
- **Документация и поддержка:** Возможно, документацию (help) тоже стоит сделать на двух языках. Пока можно ограничиться русским, но в будущем – да.
- **Дополнительные языки:** Если планируется выход за пределы Узбекистана, стоит заложить возможность добавления новых языков (казахский, английский и т.д.). Благодаря i18n это просто вопрос добавления файлов переводов.
- **Локализация печатных форм:** Кассовый чек должен печататься на языке, который требует закон – скорее всего, на государственном языке (узбекском). Нужно уточнить требования налоговой. Возможно, хотя бы на узбекском или двуязычный. Можно реализовать настройку: язык печати чеков (отдельно от интерфейса).
- При локализации чека – такие слова как "Чек фискальный", "Итого", "Сумма", наименование налога – должны переводиться. Названия товаров можно печатать как введены (если введены на русском, печатать на русском, закон это допускает, обычно).
- **Тестирование переводов:** Проверить, что все экраны на двух языках отображаются корректно, тексты не обрезаны (учесть, что узбекские слова могут быть длиннее, или наоборот).
- **Элементы Radix/UI:** Пользуемся компонентами, которые поддерживают aria-атрибуты – там тоже возможна локализация некоторых встроенных слов (например, "Close" на кнопке закрытия модала – перевести).

15. Обзор архитектуры решения

(Здесь описывается техническая архитектура, чтобы подтвердить, что она соответствует требованиям продукта.)

- **Клиентские приложения:**
- **Desktop POS:** Настольное приложение на Electron + React. Бандлирование через Vite (для быстрого запуска и HMR при разработке). Код POS написан на TypeScript, UI-слои разделены на компоненты из общего набора (см. ниже). Electron позволяет доступ к файловой системе (для хранения SQLite базы) и к нативным функциям OS (например, управление печатью на локальном принтере).

- **Web Admin (Back-Office):** Веб-приложение для администратора и менеджеров, основанное на Next.js (с использованием App Router для сегментации маршрутов и потенциала SSR/SSG, хотя в основном будет CSR для админки). React + TypeScript.
- **Shared UI Library:** Создаётся отдельный пакет `@jowi/ui` – набор React-компонентов и стилей, используемый и в POS, и в веб-админке для единообразия. В нём будут реализованы общие компоненты интерфейса (кнопки, таблицы, модальные окна и т.п.) согласно дизайн-гайдам (см. UI/UX раздел).
- **Стили и темы:** Используем Tailwind CSS для быстрой и единообразной стилизации. Настроены CSS Variables для цветовых тем (light/dark), чтобы переключение тем влияло на набор переменных (например, background, text color, primary color). Две темы (светлая и тёмная) должны поддерживаться с самого начала.
- **Radix UI Primitives:** используем Radix (<https://www.radix-ui.com/>) для доступных сниппетов UI (диалоги, поповеры, листбоксы и пр.), чтобы обеспечить хорошую UX и доступность. Компоненты ShadCN/UI оборачивают Radix, предлагая готовые стилизованные блоки на Tailwind – это ускорит разработку.
- **Серверная часть:**
 - **Backend Framework:** NestJS (Node.js + TypeScript) – модульный, поддерживает инъекцию зависимостей, имеет интеграцию с WebSockets, а также можно реализовать RPC-шлюз. Мы планируем реализовать API двумя способами:
 - **REST API** для стандартных запросов (создать чек, получить список товаров и т.д.).
 - **tRPC-подобный** (или GraphQL) для типобезопасного общения Web Admin → Backend. Возможно, Next.js (App Router) будет вызывать серверные действия (route handlers) напрямую на том же хосте.
 - **База данных (OLTP):** PostgreSQL, как надёжное хранилище для основной бизнес-информации. Структура данных спроектирована с учётом multi-tenant (tenant_id в нужных таблицах + RLS политик).
 - **Миграции:** управляются с помощью Prisma Migrate или dbmate. Храним схему в Git, миграции версионные.
 - **ORM:** Prisma позволит типобезопасно работать с БД. Также можно использовать Prisma для генерации типов, которые затем валидировать Zod'ом.
 - **Взаимодействие POS с сервером:** POS может напрямую обращаться к API (REST) при наличии интернета. Но для оффлайна – POS может накапливать операции.
 - Когда интернет появляется, POS шлёт пачку запросов или один batched request. Можно реализовать sync-эндпоинт, который принимает список изменений (продажи, новые клиенты и т.д.).
 - Сервер обрабатывает каждую операцию идемпотентно (повторная отправка того же чека не создаст дубликат, API использует уникальные идентификаторы, генерируемые на клиенте, например UUID для чеков).
 - Ответом сервер может прислать и обновления (например, «у тебя версия справочника устарела, вот новые данные» или подтверждение синхронизации).
- **Очереди и фоновые задачи:**
 - Используем **Redis** для простых очередей и кэша. Например, очередь на отправку фискальных команд: Backend кладёт задачу в Redis, `fiscal-gateway` (отдельный сервис) её забирает.
 - Redis также пригодится для кэширования часто запрашиваемых данных (справочники, настройки) чтобы разгрузить Postgres.
 - **NATS (опционально):** если решим построить событийную архитектуру, NATS Streaming (или Kafka, но Kafka тяжеловеснее) будет брокером. Можно отправлять событие "sale.created" и на него подпишется модуль лояльности, модуль отчетов, и т.д. Пока это не строго необходимо, но возможно при внедрении ClickHouse через Debezium Kafka станет фактом.

- **Сервис фискализации (fiscal-gateway):** Выделен отдельный микросервис, написанный, например, на C# или Go (или даже NodeJS, если найдутся SDK) – его задача: работать с драйверами ККМ.
 - Интерфейс `fiscal-gateway` — HTTP API или RPC. Например, `POST /openShift` с данными магазина, возвращает `{ success, shiftNumber,... }` или ошибку.
 - Обработка сложных случаев (принтер не отвечает, бумаги нет) – `fiscal-gateway` должен сигнализировать обратно. POS/Backend должны это обрабатывать и отображать оператору.
 - `fiscal-gateway` изолирует все локальные тонкости фискальных регистраторов. Его можно установить локально (рядом с POS) или запускать централизованно (если используется облачная фискализация). В Узбекистане, скорее всего, ККМ подключаются к локальному компу, значит `fiscal-gateway` либо встроен в POS-клиент, либо работает как Windows-служба.
- **Логи и мониторинг:**
 - Внедряем **OpenTelemetry** SDK в бекенд и в POS (через Electron main process) для сбора трасс (trace) – например, последовательность действий при синхронизации.
 - Разворачиваем стек наблюдаемости: **Grafana** для метрик (можно Prometheus подключить), **Loki** для логов, **Tempo** для трейсинга. Это важно для быстрого отладки проблем в продакшене.
 - **Ошибки фронтенда:** В веб и POS можно подключить Sentry или аналоги, чтобы ловить исключения UI.
- **Безопасность:**
 - Все внешние соединения должны быть по HTTPS (SSL).
 - Хранение секретов (например, ключи API интеграций) в защищённом виде (HashiCorp Vault или хотя бы .env на сервере).
 - Защита от SQL-инъекций, XSS – обеспечивается использованием ORM и тщательно эскейп пользовательский ввод при выводе.
 - Rate limiting на публичные API (например, на попытки логина).
 - JWT токены шифруются, могут иметь ограниченный срок действия (refresh tokens).
- **Аналитическая подсистема:**
- **Debezium + Kafka:** Debezium коннектор читает WAL логи Postgres, и публикует события (insert/update/delete) в топик Kafka. Интересующиеся сервисы (например, коннектор ClickHouse) читают эти события.
- **ClickHouse:** Разворачивается БД ClickHouse, настроены таблицы под нужные отчёты (например, таблица фактов продаж с нужными агрегатами). Используем Materialized Views для агрегированных показателей (продажи по дням, топ товаров) чтобы отчёты были мгновенными.
- **Отчётный API:** Backend, получая запрос на отчёт, либо:
 - А) идёт в Postgres (для небольших выборок, например, остатки на текущий момент),
 - В) или идёт в ClickHouse (для масштабных, типа продажи за год по дням).
 - Это можно скрыть за Repo слоем – запросы определяются в зависимости от объёма/типа отчёта.
- **План Б без ClickHouse:** Если MVP нужно срочно, можно без сложной аналитики – Postgres справится с первыми клиентами. Но архитектуру описали с прицелом на рост.
- **Масштабирование:**
 - Приложение stateless (NestJS) – может масштабироваться горизонтально (несколько инстансов за load balancer'ом) для обслуживания большего числа пользователей.
 - Postgres – можно настроить реплики для чтения, шардинг по tenant (если очень вырастет база, но на старте не нужно).
 - Отделение сервисов: Фискальный сервис отдельно, может несколько инстансов для разных регионов.

- Обработка очередей – отдельно, чтобы тяжелые задачи (например, генерация отчёта PDF) не тормозили основные запросы.
- **DevOps:**
 - Контейнеризация: Все компоненты (NestJS, ClickHouse, Redis, etc.) можно упаковать в Docker. Деплой в облако (например, AWS, Azure, либо локальный сервер).
 - CI/CD: Настроить pipeline для автоматического тестирования и деплоя (например, GitHub Actions → build Docker → push).
 - Backups: Регулярный бэкап базы (и сохранение на внешний storage).
 - Учет версий API: если будем развивать API, следить за версионностью, чтобы не поломать совместимость с уже установленными POS-клиентами (POS может обновляться автоматом, но всё же).
- **Производительность:**
 - Целевые показатели: время отклика основных API < 200мс, время синхронизации чека – пару секунд макс (в оффлайне не влияет на кассира, но при онлайн стараемся быстро).
 - POS должен работать на обычном офисном ПК (Windows 10+, 4GB RAM) без лагов.
 - Тестировать на больших данных: каталог 100k товаров, 1 млн продаж – интерфейсы должны выдерживать (виртуализированные списки, страничные загрузки).
- **Observability (Наблюдаемость):**
 - **Логирование действий:** Пишем инфо-логи при ключевых событиях (открытие смены, ошибка синхронизации, запрос к внешнему API и т.д.) – чтобы потом можно было диагностировать.
 - **Метрики:** Собираем метрики по количеству чеков в час, среднему времени транзакции, памяти, CPU – для оптимизации.
 - **Audit log неизменяемый:** Как говорилось, критичные данные (финансы) – записываются в отдельную таблицу, где UPDATE/DELETE запрещены, только INSERT. Это наш «след операций», который можно использовать для разборов инцидентов и доверия.

(Архитектура обеспечивает основу для всех перечисленных в требованиях возможностей, с упором на масштабируемость и надёжность.)

16. Доменная модель (основные сущности MVP)

Для ясности, приведём ключевые сущности предметной области и их взаимоотношения:

- **Бизнес (Business)** – компания-клиент SaaS. Атрибуты: название, ИНН/ОГРН, юридические данные, тариф (если коммерческая модель), настройки (валюта, язык по умолчанию и т.п.). Связан со множеством пользователей и магазинов.
- **Магазин (Store/Outlet)** – торговая точка. Атрибуты: название/адрес, привязка к бизнесу. Может иметь собственные настройки (например, отдельный префикс нумерации чеков). Один бизнес имеет несколько магазинов.
- **Терминал/Касса (POS Terminal)** – конкретное рабочее место. Атрибуты: идентификатор (может быть серийный номер устройства или логическое имя), привязка к магазину. Нужен для учёта смен и идентификации, с какого устройства пришёл чек. Один магазин может иметь несколько терминалов.
- **Пользователь / Сотрудник (User/Employee)** – учетная запись человека. Атрибуты: имя, email/телефон (для логина), пароль (хэш), роль, активность, возможно ссылка на сотрудника (вне системы). Может быть привязан к определённому магазину (для кассира). Один пользователь принадлежит одному бизнесу (в MVP).
- **Роль (Role)** – роль доступа, с набором разрешений. Фиксированные записи: Админ, Менеджер, Кассир, Склад. Связь многие-ко-многим: пользователь может иметь одну роль (или несколько, но в MVP достаточно одну основную).

- **Смена (Shift)** – кассовая смена. Атрибуты: номер смены, терминал, открыт кем/когда, закрыт кем/когда, начальная сумма, конечная сумма, рассчитанная выручка, фактическая выручка, отклонение, статус (открыта/закрыта). Связана с терминалом и, через него, с магазином и бизнесом.
- **Клиент / Покупатель (Customer)** – контрагент типа "покупатель". В рознице часто не идентифицируется, но для лояльности и CRM мы храним базу клиентов. Атрибуты: имя, телефон, email, возможно номер карты лояльности, ссылка на внешнюю систему (ID в Jowi Club). Можно хранить дату рождения, пол (для акций), адрес (если доставки).
- **Поставщик (Supplier)** – контрагент типа "поставщик". Нужен для доходов и расчетов. Атрибуты: название, контактные данные, ИНН. Баланс с поставщиком – расчетный на основе документов.
- **Товар (Product)** – товарная номенклатура (родитель для вариантов). Атрибуты: название, артикул (если есть общий), категория, бренд, единица измерения по умолчанию, налог (ставка налога), активен/неактивен.
- **Вариант товара / SKU (ProductVariant)** – конкретная разновидность товара. Если товар без вариантов, все равно создается один вариант "по умолчанию". Атрибуты: собственный код/SKU, связанный товар, значения атрибутов варианта (например, цвет=красный, размер=L), штрих-код (основной), вес (для весового товара, может хранить штучный вес?), цена закупки последняя (для справки), и пр.
- **Штрих-код (Barcode)** – отдельная сущность или просто поле списка в SKU. Наверное, можно как отдельную сущность, связанную со SKU, чтобы хранить несколько. Атрибуты: код, тип кода (EAN13, Code128 и т.д.), основной признак.
- **Категория (Category)** – категория/группа товаров (дерево). Атрибуты: название, родительская категория (для иерархии), признак типа (вдруг отделяем услуги или еще что, но пока нет).
- **Склад (Warehouse)** – место хранения. В MVP, фактически один склад = один магазин, но сущность все равно нужна. Атрибуты: название, магазин, тип (магазин, центральный склад и т.д.). Используется в остатках и движениях.
- **Остаток (Stock)** – текущий остаток товара на складе. Атрибуты: склад, SKU, количество, стоимость (средняя или суммарная? Можно не хранить, а вычислять). Можно организовать как Materialized View на основе движений, но, скорее, обновлять при каждой транзакции для быстрого доступа. Важна целостность – не потерять изменения при параллельных операциях.
- **Партия (Batch/Lot)** – партия товара. Атрибуты: SKU, склад, уникальный ID партии, количество в партии (текущее), закупочная цена, дата поступления, возможно срок годности. Новая партия создается приходом. При продаже списывается из первой по очереди.
- **Документ движения (InventoryDocument)** – абстрактная сущность для складских операций (приход, перемещение, инвентаризация и т.п.). Общие атрибуты: тип документа, дата, склад-отправитель (для перемещения), склад-получатель, контрагент (для доходов/возвратов), статус (черновик/проведен), комментарий.
- **Строка документа (DocumentLine)** – строка товара в документе: SKU, количество, цена (для доходов – закупочная, для перемещений можно 0 или тоже цена для оценки).
- Конкретные документы могут иметь специфические поля: для инвентаризации – кол-во по учёту и кол-во фактическое, для списания – причина и т.п.
- **Чек (SaleReceipt)** – документ продажи (розничный чек). Атрибуты: уникальный номер (может быть составной: номер смены + порядковый, или глобальный), дата-время, кассир (пользователь), клиент (если указали), магазин/терминал, итоги: сумма, скидка, налог, статус (пробит/возврат/аннулирован), ссылки на фискальные данные (номер фискального чека, фискальный признак).





- **Позиция чека (SaleItem)** – строка в чеке: SKU, товар, количество, цена, сумма, скидка на строку, налог на строку (если нужен).
- **Оплата (Payment)** – привязанная к чеку запись: тип оплаты (наличные, карта, баллы, сертификат и т.д.), сумма. Обычно чек состоит либо из одной оплаты на всю сумму, либо нескольких (split payment).
- **Связь с возвратом:** если это чек возврата, может быть поле reference на оригинальный чек.
- **Скидка (Discount)** – как отдельная сущность не требуется, т.к. скидка фиксируется как поле (процент или сумма) в SaleItem или SaleReceipt. Но если бы были купоны, можно хранить как сущность Promotion/DiscountCampaign. В MVP – не делаем, достаточно полей.
- **Кассовая операция (CashTransaction)** – для внесений/изъятий. Атрибуты: дата, кассир, тип (внесение/изъятие), сумма, комментарий. Привязано к смене или магазину.
- **Проводка (LedgerEntry)** – элемент бухгалтерской записи. Поля: дата, описание, счет Дт, счет Кт, сумма. Счета можно закодировать (например, 401 "Товары", 701 "Выручка"). Это для внутреннего учета, у пользователя интерфейса к этому может не быть (кроме фин. отчетов). Если не успеваем, можно опустить реализацию двойной записи, но упомянуть как базу для будущего.
- **Интеграция (IntegrationConfig)** – хранит настройки интеграций для конкретного бизнеса:
 - Тип (JowiClub, конкретный маркетплейс, платежный сервис).
 - Поля аутентификации (API key, секрет).
 - Статус интеграции (активен/не активен).
 - Могут быть связанные сущности, напр. **IntegrationLog** (журнал обмена, чтобы отлаживать).
- **Логи и служебные:**
 - **AuditLog** – запись аудита: кто, когда, что сделал (например, "User X deleted product Y" или "Price of product Z changed from A to B by User U").
 - **SyncQueue** – таблицы очередей Outbox/Inbox для оффлайн: список изменений (например, модель, ID, тип операции: insert/update/delete, payload).
 - **Notification** – если делаем всплывающие уведомления (не критично).
 - **Settings** – различные настройки бизнесов (например, включена ли продажа в минус, обязательна ли карта клиента для чека и т.п.). Может быть JSON поле.

(Эта модель охватывает основную информацию, необходимую для MVP. При разработке могут появиться дополнительные сущности или связи, но ключевые домены – продажа, товар, склад, деньги – описаны выше.)

17. Требования к UI/UX и дизайн-гайдлайны

(В этом разделе перечислены принципы дизайна интерфейса и опыт взаимодействия, чтобы продукт был удобным и единообразным.)

- **Единый дизайн-систем:** Разработчики интерфейса придерживаются принципа, что **код – единственный источник правды** для дизайна. Используется единая библиотека компонентов `@jowi/ui` без макетов в Figma, то есть стиль оттачивается непосредственно в компонентах. Это обеспечивает единообразие и предсказуемость UI на всех экранах.
- **Цветовая схема и темы:**
 - Используем CSS-переменные для цветов, шрифтов и т.д. Это позволяет легко менять тему. Изначально будет **светлая тема** (адаптированная под дневную работу) и **тёмная тема** (удобна для работы вечером или в слабо освещённых помещениях, напр. склад).

- Цветовая палитра нейтральная, без кислотных цветов. Основной цвет (primary) – например, синий или зелёный, ассоциирующийся с брендом. Ошибка – красный, предупреждение – оранжевый/желтый, успешное – зелёный.
- Важно соблюдать **контраст $\geq 4.5:1$** для текста на фоне ¹⁰, чтобы интерфейс был читаем даже на не самых новых мониторах и для людей с небольшими нарушениями зрения.
- **Типографика:**
 - Шрифты используем системные (для Windows – Segoe UI / для веб – sans-serif по умолчанию), чтобы не было проблем с загрузкой и чтобы текст выглядел привычно.
 - Размеры шрифта: базовый 14px для интерфейса (может 15px, чтобы чуть крупнее). Заголовки – 16-18px. На POS, возможно, делаем чуть крупнее (16px базово), учитывая расстояние до экрана.
 - Моноширинный шрифт – для печатных форм (чек), и возможно для каких-то таблиц (необязательно).
- **Сетка и отступы:**
 - Интерфейс строится на 8pt grid. Все отступы, поля – кратны 8 (или 4 для мелких). Это придаёт визуальную гармонию.
 - Максимальная ширина контейнера для содержимого ~1200-1440px на широком экране. Этого достаточно для отображения таблиц и форм. На очень больших экранах контент центрируется в этой ширине, чтобы не «растягивался» (улучшается читаемость).
 - Адаптивность: Веб-админка должна корректно отображаться на ширине от ~1024px (планшет в горизонтале) и выше. На мобиле веб-админку можно не оптимизировать (мало кто будет пользоваться с телефона бэкэндом), но страницы логина/отчётов можно хотя бы прокручивать.
- **Навигация и компоновка:**
 - **App Shell:** Для веб – левое боковое меню (Sidebar) со списком основных разделов: *Главная (дашборд), Товары, Склад, Продажи (или Чеки), Отчёты, Настройки, Пользователи, Интеграции* (если нужно). Вверху – панель (Topbar) с названием текущего раздела, переключателем магазина (если админ хочет смотреть данные по конкретной точке), и меню пользователя (профиль, выход, язык).
 - В POS-приложении – можно обойтись без сложного меню: основной экран – это оформление продажи. Остальные функции (например, X-отчёт, возврат без чека, настройки оборудования) можно делать через меню, вызываемое по нажатию (например, клавиша F10 "Меню" или кнопка шестерёнки). POS интерфейс должен быть максимально сфокусирован на продаже, чтобы кассир не отвлекался.
 - **Хлебные крошки:** для веб-админки, если есть вложенные сущности (например, открыли карточку товара из списка) – можно отображать навигационную цепочку. Но часто достаточно просто модальных окон для редактирования.
 - **Иконки:** Используем набор lucide-react – современные векторные иконки. Размеры 16px внутри текста/кнопок, 20-24px для основных меню/иконок. Иконки с понятной метафорой для каждого раздела: товар – , склад – , отчёты – , настройки –  и т.д. (Выберем из lucide близкие). Иконки добавляют визуальные подсказки пользователю.
- **Компоненты интерфейса:**
 - **Таблицы (DataTable):** на страницах списков (товары, документы, отчёты) используем мощный табличный компонент на базе TanStack Table. Он поддерживает:
 - Серверную пагинацию (чтобы не загружать тысячи записей сразу).
 - Виртуализацию строк для производительности на больших списках.
 - Фильтры по колонкам (например, поиск по названию товара прямо в заголовке колонки).
 - Сортировку по колонкам.
 - Возможность скрывать/показывать колонки.
 - Кнопку экспорта данных в CSV (например, текущий фильтрованный список).

- Выбор строк (чекбоксами) для пакетных операций (удалить, экспортировать выбранное и т.п.).
- **Формы (Form, FormField):** Создан единый компонент `Form` с интеграцией React Hook Form + Zod. Это даёт:
 - Валидацию вводимых данных по схеме (например, цена должна быть ≥ 0 , обязательные поля, формат email).
 - Отображение ошибок под соответствующим полем (`FormField` компонент выводит label, input, и если есть ошибка – текст ошибки).
 - Единый стиль для всех форм: расстояние между полями, поведение при фокусе, подсветка ошибок (красная рамка, например).
 - Поля ввода: текстовые, числовые (с форматированием), выпадающие списки (Select – на базе Radix Select), чекбоксы, переключатели, даты (DatePicker – можно использовать библиотеку или кастом).
 - **Диалоги форм:** Модальные окна с формой (например, создание нового товара, проведение оплаты) используют компонент `FormDialog` – он автоматически добавляет заголовок, кнопки "Сохранить/Отмена", блокирует закрытие по ESC при незаполненных обязательных, и т.д.
 - **Мастер (Wizard):** Для сложных процессов (например, первичная настройка или инвентаризация) можно применить компонент Wizard – последовательность шагов с «Далее/Назад». Это улучшает UX при множестве вводимых данных.
- **Диалоги и уведомления:**
 - **Dialog/Drawer:** используем Radix Dialog для модальных окон и Radix Drawer для выезжающих панелей сбоку. Например, просмотр детали чека – как боковая панель, чтобы не переключаться на другую страницу.
 - **Toast (уведомления):** Компонент для всплывающих уведомлений (в правом верхнем углу, например). Используется для кратких сообщений: "Товар сохранен", "Ошибка: нет подключения" и т.п. Дизайн тостов единообразный, автоматически скрываются через несколько секунд.
 - **Loading states:** Компонент спиннера и **скелетонов** (серые мигающие блоки вместо еще не загруженного списка) – общий. Например, при загрузке таблицы товаров сначала показываем скелет-строки, потом данные.
 - **Empty states:** Если список пуст – отображаем дружелюбное сообщение с иконкой, объясняющее, что пока данных нет, и возможно СТА ("Добавить первый товар").
 - **Error states:** При ошибке загрузки – показываем сообщение об ошибке с возможностью повторить попытку (Retry).
- **Навигационные компоненты:** Tabs (вкладки) – для страниц, где несколько под-разделов (например, на странице товара: «Общие», «Цены», «Остатки», «Изображения» как вкладки).
- **Карточки и графики:** Для дашборда – компонент KPI-Card (отображает показатель и подпись, может быть стрелочка тренда). Графики – примитивы для BarChart, LineChart, PieChart, с Legend и Tooltip по умолчанию ⁹.
- **Специфика POS UX:**
- **Горячие клавиши:** Ограничимся самыми нужными, но документируем их. Например: F1 – помощь/справка, F2 – поиск товара, F3 – переключить фокус на список чека, F4 – удалить выбранный товар из чека, F5 – применить скидку, F9 – оплата, F12 – закрыть смену (примерно, уточним).
- **Режим сканера:** Когда кассир сканирует штрих-код, ввод фокусируется на скрытом поле ввода, которое ловит цифры, после Enter (сканер обычно посылает Enter) – система ищет товар. Если найден – добавляет в чек. Если не найден – сигнал ошибки (звуковой, визуальный) и предложение ввести код вручную или создать товар на лету (можно сразу добавить новый товар, если такая бизнес-логика разрешена).

- **Чек на экране:** Отображается в виде списка строк с возможностью прокрутки. Последний добавленный товар выделяется на секунду (вспышка фона) для визуального подтверждения.
- **Крупные интерактивные элементы:** Кнопки оплаты, отмены – большие и вынесены в понятные зоны. Цветовое кодирование: "Оплата/Завершить" – зелёная кнопка, "Отмена" – серая/красная, "Возврат" – оранжевая, например.
- **Состояние кассы:** Где-то на экране POS небольшая панель: имя кассира, текущее время, статус связи (онлайн/оффлайн индикатор), и, возможно, сумма в кассе (но это спорно показывать кассиру постоянно). Главное – индикатор Offline Mode, чтобы кассир знал, что сейчас чек офлайн и синхрон. Это можно сделать иконкой (зеленая если онлайн, если нет).
- **Печать чеков:** После печати успешной – выдать сигнал (звук "дзинь"). Если печать провалилась – сигнал ошибки и всплывающее окно с инструкцией (например, "Проверьте подключение принтера. Повторить печать?").
- **Обучение кассира:** Возможно, сделать раздел "Тренировка" – переключение в режим, где чеки не идут в учёт, а просто для обучения (как демонстрационный режим). Это опционально.
- **Адаптация под разные устройства:**
 - POS предполагается на Windows ПК. Но Electron позволяет и Linux/Mac. Если клиенты попросят – можно будет дать.
 - Web Admin – желательно, чтобы работал и на планшете. Вдруг менеджер смотрит отчёты с iPad. Поэтому, тестируем на разрешении ~768px ширины (портретный планшет): сайдбар можно скрывать (гамбургер меню).
 - Мобильные телефоны: можно не оптимизировать админку. Но авторизация/просмотр отчёта в крайних случаях должна хотя бы не ломаться.
- **Accessibility (доступность):**
 - Придерживаемся ARIA-меток и стандартов. Radix компоненты уже имеют хорошую поддержку клавиатуры и экранных читалок.
 - Все интерактивные элементы помечены `tabindex` правильно, последовательность табуляции логичная.
 - **Focus rings:** Не отключаем обводку фокуса для клавиатурной навигации. По дизайну: при фокусе элемент обводится синим или контрастным ободком, так пользователь понимает, где он находится, особенно при использовании Tab.
 - Текстовые альтернативы для иконок (aria-label) – добавляем на кнопках, где только иконка.
 - Контраст цветов, размер шрифта – уже упомянуто.
- **Сообщения и текст:**
 - Текстовки интерфейса хранятся в i18n словарях. Они должны быть **лаконичными и понятными**. Лучше использовать простые слова, понятные рядовому пользователю, избегать излишнего канцелярита.
 - Например, вместо "Данный товар отсутствует на складе" можно вывести "Нет остатка". Вместо "Операция выполнена успешно" – просто "Сохранено".
 - В уведомлениях об ошибках – писать, что случилось и что делать: "Не удалось напечатать чек. Проверьте подключение принтера." – чтобы пользователь знал, как реагировать.
- **Онбординг (введение):**
 - Для новых бизнесов можно реализовать мастера первичной настройки: сразу после регистрации аккаунта администратору предлагается пройти шаги: заполнить информацию о компании (название, налоговые данные), добавить первый магазин (адрес), добавить пару товаров (чтобы понять как), создать пользователя-кассира. Это поможет быстрее начать (в духе "Быстрый старт – понятный интерфейс, помощь в настройке" ³).
 - Можно включить ссылки на видео-уроки или поддержку прямо в интерфейсе (например, раздел "Помощь").

- **Производительность UI:**
 - Используем lazy loading (ленивую загрузку) модулей – Next.js нам поможет (разделение по страницам). POS – сразу грузит всё нужное, но можем выкинуть неиспользуемые части.
 - Виртуализация списков, мемоизация компонентов – чтобы интерфейс не фризил на больших объёмах.
 - Проверяем, чтобы никаких дорогих операций в основном потоке Electron не было (все тяжелое – либо в web worker, либо на бэкенд).
- **Постоянство данных:**
 - Если пользователь что-то вводил в форму и закрыл случайно – можем предусмотреть предупреждение "Вы уверены? Данные не сохранены." (особенно в веб-админке).
 - Автосохранение черновиков: например, если заполняли накладную и вдруг обновили страницу – можно сохранять черновик документа в локальное хранилище.
- **Визуальное соответствие платформе:**
 - На Windows приложение может вписываться в окружение: использовать системные диалоги для выбора принтера, например. Но в целом, кастомный дизайн, унифицированный между веб и desktop.
- **Обратная связь от пользователей:**
 - В продукт можно встроить механизм отправки отзывов/репортов (например, форма "Сообщить об ошибке/предложить идею"). На старте можно собирать эту инфу для улучшений.
 - Также можно анонимно трекать взаимодействия (с согласия) – какие функции наиболее используются, где пользователи проводят больше времени – чтобы улучшать UX.

18. Ограничения и исключения (что не входит в MVP)

Чтобы четко очертить границы первой версии, перечислим функционал, **который пока не реализуется**, но может появиться впоследствии:

- **Интеграция с маркетплейсами и онлайн-торговлей:** Как отмечено, прямые коннекторы к площадкам (Uzum Market, Wildberries и др.) пока не будут готовы. Сейчас система фокусируется на офлайн-рознице. В будущем планируем добавить, т.к. конкуренты активно предлагают продажу на маркетплейсах из системы ¹¹.
- **E-commerce (интернет-магазин):** Встроенный интернет-магазин (витрина) для клиента не делаем. Нет модуля заказов интернет-магазина, корзины, сайта. Бизнес, если торгует онлайн, пока должен вести эти заказы отдельно (или в будущем – через интеграцию).
- **Продвинутые акции и кампании:** Например, сложные скидочные акции с расписанием, персонализированный маркетинг, рекомендации товаров, кросс-селл – всё это вне MVP. Будет реализовано базовое дисконтирование и интеграция с Jowi Club, не более.
- **Производство / рецептуры:** Если магазин сам производит товары (например, выпечка, или сборка комплектов с разуконплектацией) – функционал производства не включён. Нет поддержки сборок по рецептам, списания ингредиентов. Такие вещи можно решить через списание/приход вручную.
- **Логистика и доставка:** Как указано, модуль доставки (курьеры, маршруты, статусы доставок) отсутствует. Предполагается, что это для другого ПО или ненужно для формата наших пользователей. Если будет спрос (например, сеть магазинов хочет доставлять товары клиентам) – интеграция с сервисами доставки или простой модуль отгрузок может появиться позже.
- **Мобильные приложения (Android/iOS):** Отсутствуют приложения для смартфонов для кассира или менеджера. Менеджеры могут использовать веб-админку из браузера на планшете/телефоне, но специально не оптимизировано. Кассиру требуется Windows-

устройство. В будущем, возможно, появится POS на Android (для небольших киосков), но MVP – только Windows.

- **Внешняя бухгалтерия:** Не будет обмена данными с 1С, бухгалтерскими сервисами. Считаем, что встроенных отчётов хватит. В крайнем случае – экспорт в Excel и бухгалтер вручную загрузит. **Ценность нашего продукта именно в том, чтобы избавиться от постоянного ведения бухучёта вручную**, поэтому мы стараемся покрыть потребности внутри. Однако, официальные бухгалтерские проводки, баланс для налоговой – это может делаться внешне, наш софт ориентирован на управленческий учет.
- **CRM beyond loyalty:** Глубокий CRM-функционал (управление лидами, воронкой продаж, задачи продавцам, рассылки клиентам) – не делаем. Наша CRM-часть ограничена хранением клиентов, историей покупок и базовой сегментацией. Например, пока не будет модулей наподобие постановки задач отделу продаж, как есть в МойСклад ¹⁴.
- **HR и зарплаты:** Учёт рабочего времени сотрудников, расчет зарплаты, мотивация – вне области. Можно только выгружать отчёт кто сколько продал для начисления бонусов вручную.
- **Развитая аналитика:** Некоторые сложные аналитические отчёты (ABC/XYZ анализ ассортимента, RFM анализ клиентов, прогнозирование продаж) – не включены. MVP ограничится базовой аналитикой, перечисленной ранее.
- **Сложные конфигурации оборудования:** Например, поддержка POS-терминалов на Linux, интеграция с фискальными регистраторами через облако – возможно, не всё это в MVP (может быть реализуем только 1-2 модели ККМ, а остальные добавим по мере необходимости).
- **Поддержка множества юрлиц в одном бизнесе:** Если сеть имеет разные ООО под одним брендом, пока решение – вести раздельно. В дальнейшем, может быть, дадим мульти-юрлицо в рамках одного аккаунта.
- **Глобальная мультивалютность/многоналоговость:** Не требуется, так как Узбекистан – одна валюта, налоговая система тоже единая для нашего сегмента. Если будем выходить в другие страны, тогда добавим. Пока в коде можно предусмотреть валюта=UZS как константа.
- **Тонкие настройки прав:** Пока роли жёстко запрограммированы. UI для кастомизации ролей (создать свою роль с галочками прав) – не делаем. Если понадобится изменить права – через обновление системы.
- **Тарифные ограничения:** Если SaaS предполагает тарифы (например, бесплатный на 1 кассу, платный на больше) – возможно, пока MVP будет бесплатным/единым. В требованиях этого нет, но можно пометить, что будет монетизация. МойСклад, например, имеет бесплатный тариф с ограничениями ¹⁵. Мы можем позже ввести тарифные ограничения (по кол-ву точек, пользователей или объёму данных).
- **24/7 поддержка:** Организационно обеспечить саппорт 24/7 пока не требуется. Но в интерфейсе хорошо бы иметь раздел "Помощь" с FAQ. В будущем, когда будет коммерческий релиз, организуется техподдержка.
- **Регулирование цен поставщиков:** Механизмы типа импорта прайсов и обновления цен на основе закупок (хотя частично упомянули импорт Excel) – это больше nice-to-have, без них можно работать вручную.

(Всё вышеперечисленное исключено из первой версии, чтобы сосредоточиться на ключевом функционале. Однако, архитектура и дизайн закладываются с учётом возможного появления этих элементов, чтобы их внедрение прошло максимально естественно.)

19. Ключевые метрики успеха MVP

Чтобы оценить эффективность и успех запуска продукта, определим несколько **метрик**, за которыми будем наблюдать:

- **Скорость оформления продажи:** Среднее время, требуемое кассиру от начала сканирования товаров до выдачи чека покупателю. Цель – это время не больше, чем в кассовых решениях конкурентов. Идеально, если система позволяет оформить типовой чек (3-5 позиций, один способ оплаты) за **15-20 секунд**. Будем измерять время между открытием нового чека и печатью фискального чека. Улучшение этого показателя напрямую означает удобство для кассира и удовлетворенность покупателей (меньше очередь).
- **Надёжность оффлайн-продаж:** Процент оффлайн-транзакций, успешно синхронизированных при восстановлении связи. Стремимся к **≈ 100%** – т.е. ни один чек не теряется. Отслеживаем число случаев, когда данные потерялись или требовали ручного вмешательства. Также время, за которое данные доходят до сервера после восстановления связи (желательно в пределах нескольких минут).
- **Доступность системы (uptime):** Время бесперебойной работы сервера. Цель – **>99.9%** uptime. Так как POS офлайн, небольшой простой сервера не мешает продажам, но может мешать получению отчетов или синку. Поэтому мы мониторим, чтобы сервер и базаданные работали стабильно. В случае обновлений – минимизировать даунтайм (развёртывание без остановки, миграции без блокировок).
- **Производительность операций:**
 - Время отклика ключевых API: например, загрузка каталога товаров (должно занимать <2 секунды для 1000 товаров, <5 секунд для 10000 товаров), формирование отчёта продаж за месяц (<3 секунды).
 - Время открытия POS приложения и готовности к работе: <10 секунд на среднем ПК.
 - Память, CPU на клиенте – следить, чтобы Electron-приложение не потребляло чрезмерно ресурсы (должно работать плавно даже на недорогом оборудовании).
- **Время инвентаризации:** Не прямой системный показатель, но важный бизнес-процесс. Если ранее магазину требовалось, скажем, 8 часов на полную инвентаризацию вручную, с нашей системой они могут сократить это время. Метрика – время проведения инвентаризации X позиций, или количество ошибок при пересчёте. Мы можем собрать обратную связь, что с нашим ПО инвентаризация проходит, например, на 30% быстрее благодаря сканеру и удобному интерфейсу.
- **Метрика удовлетворенности:**
 - Косвенно измеряется удержанием клиентов (сколько продолжают пользоваться через 3 месяца) и NPS (Net Promoter Score) — можно собирать отзывы. Положительные отзывы об удобстве будут индикатором успеха.
 - Прямо внутри системы можем отслеживать, используют ли пользователи все функции (например, сколько % подключили лояльность, сколько смотрят отчёты ежедневно).
- **Ошибки и сбои:** Количество аварийных сбоев приложения (крашей) на кассах. Стремиться к 0 критических. Ловить исключения через мониторинг – и быстро исправлять.
- **Процент выполненных синхронизаций:** Если, к примеру, из 1000 оффлайн операций все 1000 дошли успешно – 100%. Если что-то застряло – анализируем. Цель – максимизировать.
- **Обратная связь от пилотных клиентов:** Качественная метрика – отзывы пилотных магазинов. Например, "стало проще обучать новых кассиров (обучение занимает 1 час вместо 1 дня)" или "владелец теперь видит картину по продажам сразу, раньше вручную считал". Такие кейсы показывают ценность.

- **Активность пользователей:** Количество активных пользователей (DAU/MAU) в системе, количество чеков в день. Рост этих показателей после запуска – признак востребованности.
- **Сравнение с конкурентами:** Например, время внедрения: цель – **быстрый старт**. Если МойСклад заявляет "старт за 15 минут" ⁸, мы тоже хотим, чтобы новый клиент настроил базу и начал работу в считанные минуты. Будем проверять, удастся ли пользователям самостоятельно запуститься, или им нужна помощь (если нужна – упрощаем UI/онбординг).

Отслеживая эти метрики, команда будет понимать, где нужно улучшать продукт в следующих итерациях – будь то оптимизация скорости, упрощение интерфейсов или добавление недостающих функций.

Итого, данный PRD описывает полнофункциональную систему для розничной торговли, ориентированную на **комплексность и удобство**. Мы ориентируемся на функционал лидеров рынка (как Billz, МойСклад), предлагая аналогичные возможности: объединение POS, складского учёта, CRM-элементов и аналитики в одном решении ¹. При этом особое внимание уделяется **локальным требованиям Узбекистана** – фискализация чеков, поддержка узбекского языка, интеграция с национальной маркировкой ⁴. Простота и дружелюбный интерфейс – приоритет: система должна быть понятна пользователям без специального образования, позволяя начать работу практически сразу ³. Такой продукт позволит автоматизировать магазин "под ключ", убрав необходимость в отдельных программах для разных задач и сведя ручной труд (особенно бухучёт и инвентаризацию) к минимуму.

¹ ¹² TBC Bank Group acquires majority stake in BILLZ, Uzbekistan's leading retail management SaaS platform | TBC Bank

<https://www.tbcbankgroup.com/news-and-media/press-releases/news/2025/tbc-bank-group-acquires-majority-stake-in-billz-uzbekistan-s-leading-retail-management-saas-platform/>

² ³ ⁴ ⁵ ⁶ ⁷ ⁸ ⁹ ¹⁰ ¹¹ ¹³ ¹⁴ ¹⁵ Облачная ERP-система МойСклад — складской учет товаров онлайн, программа автоматизации торговли и производства для Узбекистана

<https://www.moysklad.uz/>