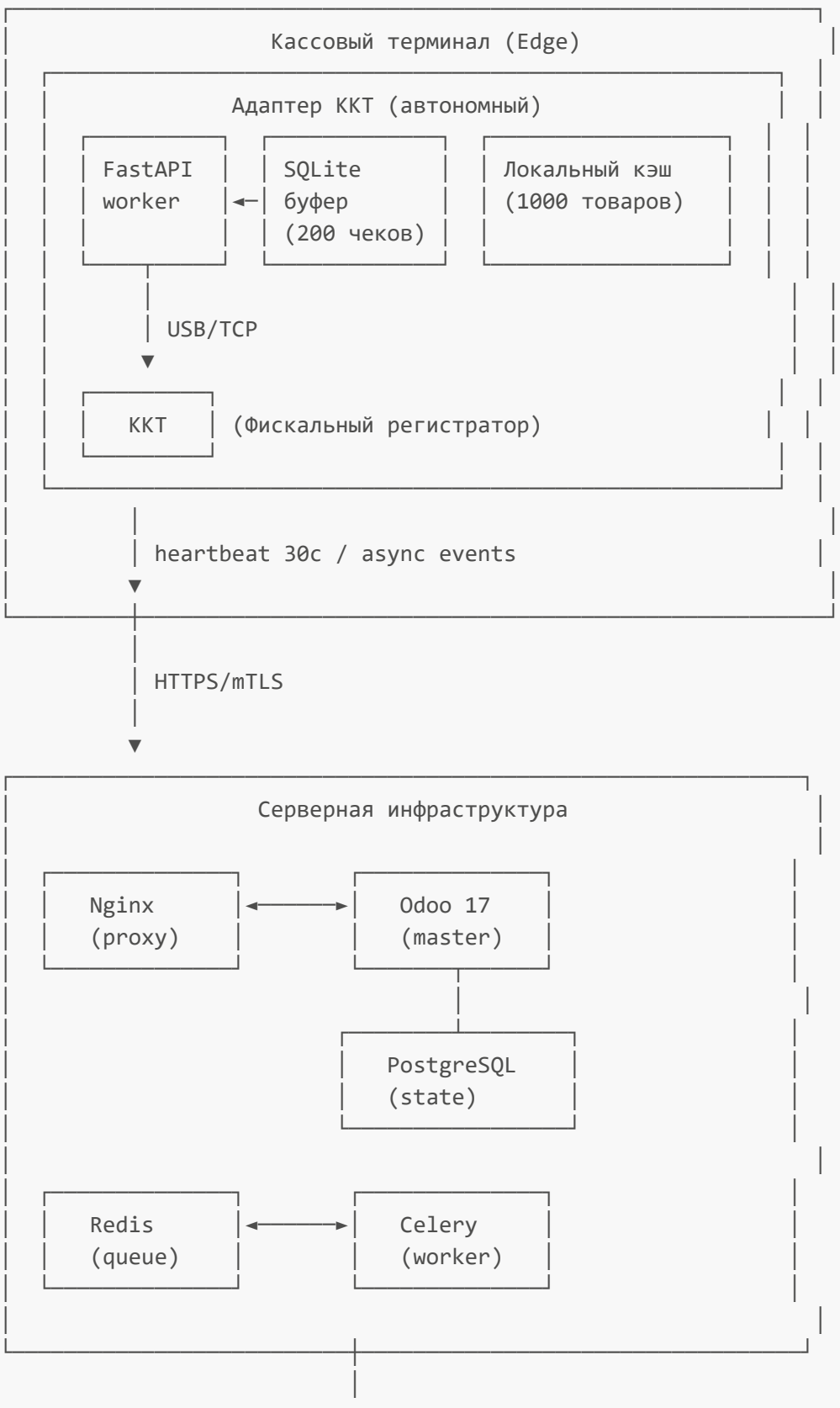
3.1 Архитектурная схема (offline-first)

> **Критичное изменение:** Архитектура переработана под приоритет автономной работы кассы

Принципиальная схема



Ключевые принципы архитектуры

1. Автономность адаптера ККТ

- **Deployment:** отдельный контейнер/процесс на кассовом терминале (edge computing)
- **State management:**
  - SQLite для офлайн-буфера (персистентность)
  - Локальный кэш каталога (1000 последних товаров + цены)
  - Heartbeat к Odoo каждые 30с
- **Режимы работы:**
  - **Online:** синхронная отправка чеков в ОФД + синхронизация с Odoo
  - **Degraded:** связь с Odoo потеряна → работа из кэша, буферизация событий
  - **Offline:** связь с ОФД потеряна → локальная фискализация, накопление в буфере

2. Двухфазная фискализация

```
Фаза 1 (локальная, всегда успешна):
┌──────────────────────────────────────────────┐
│ 1. Генерация receipt_id (UUIDv4)             │
│ 2. Сохранение в SQLite (WAL mode)            │
│ 3. Печать чека на ККТ                        │
│ 4. Отдача товара клиенту                     │
└──────────────────────────────────────────────┘

Фаза 2 (отложенная, best-effort):
┌──────────────────────────────────────────────┐
│ 1. Проверка связи с ОФД (ping)               │
│ 2. Если online → отправка из SQLite          │
│ 3. Если offline → остается в буфере          │
│ 4. Фоновая синхронизация (backoff)           │
└──────────────────────────────────────────────┘
```

3. Взаимодействие компонентов

Взаимодействие	Режим	Таймаут	Ретраи	Fallback
Odoo → Адаптер ККТ	Async (queue)	30с	3 × exp backoff	Буфер в Redis
Адаптер → ККТ	Sync	5с подключение, 10с чтение	3 × exp backoff	Офлайн-буфер SQLite

Взаимодействие	Режим	Таймаут	Ретраи	Fallback
Адаптер → ОФД	Async	10с	∞ (фоновая синхронизация)	Офлайн-буфер SQLite
Адаптер ↔ Odoo (heartbeat)	Async	2с	3	Автономный режим
POS UI → Odoo	Sync	3с	1 (немедленный feedback)	Кэш + ручной ввод

Уточнения по инфраструктуре

- **Брокер очередей для Celery:** Redis (пилот) или RabbitMQ (прод) — для импортов 10k+ и фоновых задач высокой нагрузки.
- **Синхронно ↔ асинхронно:**
  - ~~Синхронно: Odoo → адаптер ККТ для фискализации/печати чека~~ → **ИЗМЕНЕНО:** асинхронно через очередь + двухфазный коммит
  - Асинхронно: импорт крупных файлов (10k+ строк), рассылка электронных чеков, построение тяжёлых отчётов, **синхронизация офлайн-буферов**.
- **Таймауты/повторы (circuit breaker):**
  - Соединение ≤ 5 с, чтение ≤ 10 с, попыток = 3, экспоненциальный backoff
  - При превышении — размыкание на 30 с, логирование события
  - **Для ОФД:** бесконечные ретраи с exp backoff (cap 5 мин) — данные в буфере до успешной доставки
- **Отказоустойчивость:**
  - При падении ККТ/сети — офлайн-буфер чеков (ёмкость ≥ 200), гарантированная доставка при восстановлении
  - Уведомление оператора (алерт P1 при переполнении буфера ≥80%)
  - **При падении адаптера ККТ:** автоматический перезапуск контейнера (Docker restart policy), восстановление состояния из SQLite ≤2 мин
- **Обязательность Celery:**
  - Для объёма импортов > 10k строк — Celery обязателен
  - Для ≤ 10k и непиковых окон достаточно Odoo cron
  - **Для синхронизации офлайн-буферов:** Celery для масштаба ≥5 касс одновременно
- **Безопасность адаптера ККТ:**
  - Изолированная подсеть/VLAN
  - Доступ к устройству ККТ (USB/TCP) только с хоста-адаптера
  - mTLS/JWT между Odoo и адаптером (JWT ≤15 мин TTL)
  - Минимальные права к БД (read-only кэш каталога)
  - Запрещён прямой доступ из внешней сети

    - **Дополнительно:** read-only filesystem кроме `/data` (SQLite + логи), SELinux/AppArmor

- **Odoo Community 17** (Python, PostgreSQL) в Docker.
- **Сервис-адаптер ККТ** (Python 3.11+, FastAPI): очередь, ретраи, драйвер ККТ/облако.
- **Nginx** как реверс-прокси, TLS 1.3.
- **Фоновые задания**: Odoo cron + Celery (опц.) для тяжёлых импортов + **приоритетная очередь для синхронизации офлайн-буферов**.

3.2 Компоненты и модули

- `optics_core`: модели рецептов/линз, workflow, печать бланков.
- `optics_pos_ru54fz`: API к ККТ, X/Z-отчёты, электронный чек, **офлайн-буфер (SQLite), двухфазная фискализация, UI-индикация офлайн-режима**.
- `connector_b`: импорт Excel/CSV, профили маппинга, превью → upsert, логи, **блокировка импорта при несинхронизированных буферах**.
- `ru_accounting_extras`: кассовые счета по точкам, движения, отчёты GP/по точкам.

Новый компонент: `kkt_adapter` (автономный сервис)

Стек:

- Python 3.11+, FastAPI, uvicorn
- SQLite (WAL mode) для офлайн-буфера
- aiohttp для асинхронных запросов к ОФД/Odoo
- APScheduler для heartbeat и фоновой синхронизации

Структура:

```
kkt_adapter/
├── app/
│   ├── main.py             # FastAPI app
│   ├── models.py           # Pydantic models
│   ├── buffer.py           # SQLite buffer manager
│   ├── cache.py            # Product cache (1000 SKU)
│   ├── kkt_driver.py       # KKT driver (USB/TCP)
│   ├── ofd_client.py       # ОФД API client
│   ├── sync_worker.py      # Background sync (APScheduler)
│   └── heartbeat.py        # Odoo heartbeat (30s)
├── data/
│   ├── buffer.db           # SQLite (офлайн-буфер)
│   └── cache.json          # Product cache
├── config.toml             # Configuration
├── Dockerfile              # Multi-stage build
└── docker-compose.yml      # Local dev
```

API endpoints (новые для офлайн-режима):

```
POST   /v1/kkt/receipt           # Create receipt (двухфазный)
GET    /v1/kkt/buffer/status     # Статус буфера
POST   /v1/kkt/buffer/sync       # Ручная синхронизация (admin)
GET    /v1/kkt/buffer/logs       # Логи буфера (пагинация)
POST   /v1/kkt/cache/refresh     # Обновление кэша каталога
GET    /v1/health                # Health check (для мониторинга)
```

3.3 Хранилище данных

- **PostgreSQL (Odoo master state)**:
  - Одна БД `optics_erp`
  - Бэкапы `pg_dump` ежедневно, хранение ≥ 14 дней
  - Политики vacuum/analyze
  - **Не содержит офлайн-буферы** (только синхронизированные чеки)
- **SQLite (адаптер ККТ, офлайн-буфер)**:
  - WAL mode для concurrent reads
  - Схема:

```
CREATE TABLE receipts (
  id TEXT PRIMARY KEY,           -- UUIDv4
  pos_id TEXT NOT NULL,
  created_at INTEGER NOT NULL,   -- Unix timestamp
  fiscal_doc TEXT,               -- JSON ФФД
  status TEXT NOT NULL,          -- pending/syncing/synced/failed
  retry_count INTEGER DEFAULT 0,
  last_error TEXT,
  synced_at INTEGER
);
CREATE INDEX idx_status ON receipts(status);
CREATE INDEX idx_created_at ON receipts(created_at);
```

  - Автоочистка: записи со статусом `synced` старше 7 дней удаляются
  - Бэкап: включается в общий бэкап кассового терминала (daily snapshot)
- **Файлы (Odoo)**:
  - Загрузки Excel, логи импорта
  - Очистка через ретеншн-политику (30 дней)
  - **НЕ содержит офлайн-буферы** (только на кассовых терминалах)

3.4 API-контракты (обновлено для офлайн)

Общие принципы

- **Версионирование:** все REST-пути начинаются с `/v1`.
- **Аутентификация:** mTLS на уровне канала + короткоживущие JWT (≤ 15 мин) со скоупами.
- **Идемпотентность:** заголовок `Idempotency-Key` (UUIDv4) обязателен для мутирующих запросов; повтор с тем же ключом возвращает прежний ответ.
- **Ошибки:** унифицированная модель JSON `{code, message, details, id}`; коды: 400/401/403/404/409/422/429/5xx.
- **Ограничение скорости:** базово 60 rps/интеграция (пилот); при превышении — `429` + `Retry-After`.
- **Ретраи/бэкофф:** экспоненциальный, не более 3 попыток для синхронных API; **бесконечные ретраи для фискализации в ОФД** (с exp backoff, cap 5 мин).
- **Заголовки:** `Content-Type: application/json; charset=utf-8`.

Обязательные поля ФФД 1.2 для `/v1/kkt/receipt`:
`type` (sale/refund/correction), позиционные атрибуты (наименование, ставка НДС, сумма, признак предмета/способа расчёта), итоги по чеку, способ оплаты; при маркировке — `mark_code`.

Примеры API (OpenAPI, фрагмент с офлайн-поддержкой)

```
openapi: 3.0.3
info:
  title: OpticsERP Adapter API
  version: 1.0.0
  description: |
    API адаптера ККТ с поддержкой offline-first.
    Критичные изменения в v1.0.0:
    - Двухфазная фискализация (локальная → ОФД)
    - Офлайн-буфер на 200+ чеков
    - Автоматическая синхронизация при восстановлении связи

paths:
  /v1/kkt/receipt:
    post:
      summary: Create fiscal receipt (двухфазная фискализация)
      description: |
        Фаза 1: чек сохраняется локально (SQLite) + печатается на ККТ
        Фаза 2: асинхронная отправка в ОФД (если online) или буферизация (если
offline)

        ВАЖНО: response 200 означает успешную печать и локальное сохранение,
        но НЕ гарантирует немедленную отправку в ОФД.

        Для проверки статуса фискализации используйте GET /v1/kkt/receipt/{id}
      parameters:
        - in: header
          name: Idempotency-Key
          required: true
          schema: { type: string, format: uuid }
          description: UUIDv4 для предотвращения дубликатов
      requestBody:
        required: true
```

```
        content:
          application/json:
            schema:
              type: object
              required: [pos_id, type, items, payments]
              properties:
                pos_id: { type: string, example: "POS-001" }
                type: { type: string, enum: [sale, refund, correction] }
                items:
                  type: array
                  items:
                    type: object
                    required: [sku, name, qty, price, vat]
                    properties:
                      sku: { type: string }
                      name: { type: string }
                      qty: { type: number }
                      price: { type: number }
                      vat: { type: string, enum: ["20%", "10%", "0%", "no_vat"]
}
                      discount: { type: number }
                      mark_code: { type: string, nullable: true }
                payments:
                  type: array
                  items:
                    type: object
                    required: [method, amount]
                    properties:
                      method: { type: string, enum: [cash, card, acquiring] }
                      amount: { type: number }
                customer:
                  type: object
                  properties:
                    email: { type: string, format: email }
                    phone: { type: string }
      responses:
        '200':
          description: |
            OK - чек напечатан и сохранен локально.
            Фискализация в ОФД происходит асинхронно.
          content:
            application/json:
              schema:
                type: object
                properties:
                  status:
                    type: string
                    enum: [printed, buffered]
                    description: |
                      printed - чек напечатан и отправлен в ОФД
                      buffered - чек напечатан и сохранен в офлайн-буфере
                  receipt_id: { type: string, format: uuid }
                  fiscal_doc:
```

```
                    type: object
                    nullable: true
                    description: Заполняется только если status=printed
                    properties:
                      fn: { type: string, description: "Номер ФН" }
                      fd: { type: string, description: "Номер ФД" }
                      fp: { type: string, description: "Фискальный признак" }
                  buffer_info:
                    type: object
                    nullable: true
                    description: Заполняется только если status=buffered
                    properties:
                      queue_position: { type: integer }
                      total_queued: { type: integer }
                      estimated_sync_time: { type: string, format: date-time }
        '409':
          description: Duplicate Idempotency-Key (чек уже создан)
        '422':
          description: Validation error (неверные поля ФФД)
        '429':
          description: Rate limit exceeded
        '503':
          description: |
            Service unavailable - офлайн-буфер переполнен (≥200 чеков).
            Требуется ручная синхронизация администратором.
          headers:
            Retry-After:
              schema: { type: integer }
              description: Время в секундах до повторной попытки

  /v1/kkt/receipt/{id}:
    get:
      summary: Get receipt status
      parameters:
        - in: path
          name: id
          required: true
          schema: { type: string, format: uuid }
      responses:
        '200':
          description: OK
          content:
            application/json:
              schema:
                type: object
                properties:
                  id: { type: string, format: uuid }
                  status:
                    type: string
                    enum: [pending, syncing, synced, failed]
                  created_at: { type: string, format: date-time }
                  synced_at: { type: string, format: date-time, nullable: true
}
```

```
                fiscal_doc:
                  type: object
                  nullable: true
                retry_count: { type: integer }
                last_error: { type: string, nullable: true }

  /v1/kkt/buffer/status:
    get:
      summary: Офлайн-буфер статус
      description: Информация о текущем состоянии офлайн-буфера
      responses:
        '200':
          description: OK
          content:
            application/json:
              schema:
                type: object
                properties:
                  total_capacity: { type: integer, example: 200 }
                  current_queued: { type: integer, example: 47 }
                  percent_full: { type: number, format: float, example: 23.5 }
                  oldest_receipt:
                    type: object
                    nullable: true
                    properties:
                      id: { type: string, format: uuid }
                      created_at: { type: string, format: date-time }
                      age_hours: { type: number }
                  network_status:
                    type: string
                    enum: [online, offline, degraded]
                  last_sync:
                    type: object
                    properties:
                      timestamp: { type: string, format: date-time }
                      synced_count: { type: integer }
                      failed_count: { type: integer }
                      duration_seconds: { type: number }
                  alerts:
                    type: array
                    items:
                      type: object
                      properties:
                        level: { type: string, enum: [warning, critical] }
                        message: { type: string }

  /v1/kkt/buffer/sync:
    post:
      summary: Принудительная синхронизация буфера
      description: |
        Запускает немедленную синхронизацию офлайн-буфера с ОФД.
        Требует роли администратора.
      security:
```

```
        - AdminAuth: []
      requestBody:
        content:
          application/json:
            schema:
              type: object
              properties:
                force:
                  type: boolean
                  default: false
                  description: Если true, игнорирует backoff и пытается
синхронизировать сразу
      responses:
        '202':
          description: Accepted - синхронизация запущена
          content:
            application/json:
              schema:
                type: object
                properties:
                  job_id: { type: string, format: uuid }
                  estimated_duration_seconds: { type: integer }
        '409':
          description: Conflict - синхронизация уже идет
        '503':
          description: Service unavailable - сеть недоступна

  /v1/kkt/buffer/logs:
    get:
      summary: Paged buffer logs
      parameters:
        - in: query
          name: page
          schema: { type: integer, default: 1 }
        - in: query
          name: limit
          schema: { type: integer, default: 100, maximum: 500 }
        - in: query
          name: level
          schema:
            type: string
            enum: [INFO, WARN, ERROR]
        - in: query
          name: since
          schema: { type: string, format: date-time }
      responses:
        '200':
          description: OK
          content:
            application/json:
              schema:
                type: object
                properties:
```

```
                total: { type: integer }
                page: { type: integer }
                limit: { type: integer }
                items:
                  type: array
                  items:
                    type: object
                    properties:
                      timestamp: { type: string, format: date-time }
                      level: { type: string }
                      event: { type: string }
                      receipt_id: { type: string, nullable: true }
                      message: { type: string }

  /v1/connector/import:
    post:
      summary: Import catalog/stock file
      description: |
        ВАЖНО: импорт блокируется если существуют несинхронизированные
        офлайн-буферы (≥1 чек в статусе pending/syncing).
      parameters:
        - in: query
          name: dry_run
          schema: { type: boolean }
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              required: [profile_id, file_id]
              properties:
                profile_id: { type: string }
                file_id: { type: string }
      responses:
        '202':
          description: Accepted; import job started
          content:
            application/json:
              schema:
                type: object
                properties:
                  job_id: { type: string }
                  limits:
                    type: object
                    properties:
                      max_file_mb: { type: integer, example: 50 }
                      max_rows: { type: integer, example: 100000 }
        '409':
          description: |
            Conflict - существуют несинхронизированные офлайн-буферы.
            Дождитесь синхронизации или выполните ручную синхронизацию.
          content:
```

```
            application/json:
              schema:
                type: object
                properties:
                  code: { type: string, example: "UNSYNC_BUFFERS" }
                  message: { type: string }
                  unsync_terminals:
                    type: array
                    items:
                      type: object
                      properties:
                        terminal_id: { type: string }
                        queued_receipts: { type: integer }
        '422': { description: Validation error }
        '429': { description: Rate limit exceeded }

  /v1/connector/import/{job_id}/logs:
    get:
      summary: Paged import logs
      parameters:
        - in: path
          name: job_id
          required: true
          schema: { type: string }
        - in: query
          name: page
          schema: { type: integer, default: 1 }
        - in: query
          name: limit
          schema: { type: integer, default: 100 }
      responses:
        '200':
          description: OK
          content:
            application/json:
              schema:
                type: object
                properties:
                  total: { type: integer }
                  page: { type: integer }
                  limit: { type: integer }
                  items:
                    type: array
                    items:
                      type: object
                      properties:
                        row: { type: integer }
                        level: { type: string, enum: [INFO, WARN, ERROR] }
                        message: { type: string }

  /v1/health:
    get:
      summary: Health check (для мониторинга)
```

```
      responses:
        '200':
          description: OK - сервис работает
          content:
            application/json:
              schema:
                type: object
                properties:
                  status: { type: string, enum: [healthy, degraded, unhealthy]
}
                  components:
                    type: object
                    properties:
                      kkt_device:
                        type: object
                        properties:
                          status: { type: string }
                          last_check: { type: string, format: date-time }
                      ofd_connection:
                        type: object
                        properties:
                          status: { type: string }
                          last_successful_sync: { type: string, format: date-
time }
                      odoo_connection:
                        type: object
                        properties:
                          status: { type: string }
                          last_heartbeat: { type: string, format: date-time }
                      buffer:
                        type: object
                        properties:
                          status: { type: string }
                          percent_full: { type: number }
        '503':
          description: Service unavailable - критичные компоненты недоступны
```

Примеры отказов/повторов:

- `502/504` (ККТ) → клиент повторяет до 3 раз с тем же `Idempotency-Key`
- `503` (ОФД) → чек автоматически буферизируется, фоновая синхронизация
- `503` (буфер переполнен) → клиент НЕ повторяет, алерт P1, ручная синхронизация администратором
- `409` (duplicate key) → клиент НЕ повторяет, получает данные оригинального чека

3.5 Требования к качеству и безопасности

- Линтеры: `black`, `flake8`, `isort`, `mypy` (type checking).
- Тесты: `pytest` (модели, сервисы), мок эмулятора ККТ, **интеграционные тесты офлайн-буфера (SQLite in-memory)**.

- CI: pre-commit, unit, **интеграционные тесты офлайн-сценариев**, сборка образов.
- Безопасность: RBAC, минимальные привилегии, аудит действий (цен, POS, **офлайн-событий**), TLS 1.3.
- **Дополнительно для адаптера ККТ:**
  - Read-only filesystem (кроме `/data`)
  - Secrets в env vars (не в git)
  - Регулярные обновления base image (Alpine Linux)
  - SAST-сканирование (Bandit, Safety)

3.6 Производительность и надёжность

- **Цели (связь с Док. 2):**
  - P95 печати чека ≤ **7 с** (**независимо от связи с ОФД**)
  - Синхронизация офлайн-буфера: 200 чеков ≤ **10 мин**
  - Throughput синхронизации: ≥ **20 чеков/мин** (P50), ≥ **15 чеков/мин** (P95)
  - Импорт **10k строк ≤ 2 мин**
  - **Дубликаты чеков = 0** (идемпотентность по `receipt_id`)
- **Пулы соединений:**
  - **PostgreSQL:** `max_connections ≥ 200`; при наличии pgbouncer — пул `pool_size 50–100` (transaction mode)
  - **Odoo:** `workers = 2×CPU`; `db_maxconn = 16` на worker (итоговая сумма ≤ лимита БД); `limit_memory_hard/soft` по стандартным рекомендациям
  - **SQLite (адаптер ККТ):** WAL mode, `journal_mode=WAL`, `synchronous=NORMAL` (баланс производительности/надёжности)
- **Индексы и модель данных:**
  - Уникальный индекс по **SKU**
- **Индексы и модель данных:**
  - Уникальный индекс по **SKU**
  - Индекс по **EAN/штрихкоду**
  - Прикладные внешние ключи (FK) на ключевых таблицах заказов/позиционных строк
  - Аудит изменений цен/складских остатков
  - **SQLite (офлайн-буфер):** индексы по `status`, `created_at` для эффективной FIFO-выборки
  - **Event Sourcing таблица** (опционально, для полного аудита):

```
CREATE TABLE product_events (
    id BIGSERIAL PRIMARY KEY,
    aggregate_id TEXT NOT NULL,          -- SKU продукта
    event_type TEXT NOT NULL,            -- price_changed,
stock_adjusted, ...
    event_data JSONB NOT NULL,
    metadata JSONB,                      -- user_id, pos_id,
```

```
timestamp, trace_id
    created_at TIMESTAMP NOT NULL DEFAULT NOW(),
    sequence_number BIGINT NOT NULL
);
CREATE INDEX idx_product_events_aggregate ON
product_events(aggregate_id, sequence_number);
```

- **Офлайн-буфер чеков:**
  - Ёмкость ≥ **200** записей (FIFO)
  - **SQLite настройки для максимальной durability:**

```
conn.execute("PRAGMA journal_mode=WAL")
conn.execute("PRAGMA synchronous=FULL")      # КРИТИЧНО: защита от
power loss
conn.execute("PRAGMA wal_autocheckpoint=100")
conn.execute("PRAGMA cache_size=-64000")     # 64 MB кеш
```

  - **Circuit Breaker для ОФД:** прекращение попыток отправки при 5 ошибках подряд, размыкание на 60с
  - Политика повторной доставки — экспоненциальный бэкофф (стартовая задержка 5с, макс 5 мин)
  - **DLQ (Dead Letter Queue):** после 20 неудачных попыток чек переносится в DLQ для ручной обработки
  - Инструмент «повторить из DLQ» в UI (роль: администратор)
  - Автоочистка синхронизированных чеков через **7 дней**, DLQ — **90 дней**
  - **Персистентность:** SQLite WAL mode, автоматический checkpoint каждые 100 транзакций
  - **UPS-защита:** graceful shutdown при работе от батареи (скрипт для apcupsd)
- **Throughput импорта:**
  - ≥ **5k строк/мин** (≈ **80 строк/с**) при P50
  - P95 ≥ **60 строк/с** на стандартном профиле
  - Прогресс-бар и пагинация логов
  - **Блокировка импорта:** если существуют несинхронизированные буферы (≥1 чек в pending/syncing)
- **SLA очередей (Bulkhead Pattern):**
  - **Критичная очередь** (фискализация, синхронизация): ≤ **20 задач**, старше ≤ **1 мин**
  - **Высокий приоритет** (электронные чеки): ≤ **50 задач**
  - **Обычная очередь** (импорты): ≤ **100 задач**
  - **Низкий приоритет** (отчёты): без лимита
  - Алерт при превышении порога критичной очереди ≥ **5 мин** подряд
  - **Разделение workers:**

```
celery-worker-critical: 4 concurrency, только critical queue
celery-worker-high: 2 concurrency, high + critical overflow
celery-worker-default: 2 concurrency, default + low
```

- **Backpressure стратегия:**
  - **При превышении 50 задач в критичной очереди:**
    - Новые некритичные задачи блокируются (HTTP 429)
    - Приоритет отдается фискализации и синхронизации офлайн-буферов
    - Алерт администратору (уровень Warning)
  - **При превышении 100 задач:**
    - Автоматическое масштабирование critical workers (+2, max 10)
    - Эскалация P2 → P1
    - Уведомление on-call инженера

3.6.1 Circuit Breaker для ОФД (защита от каскадных отказов)

Паттерн Circuit Breaker

Состояния:

- **CLOSED** (норма): все запросы идут к ОФД
- **OPEN** (авария): все запросы немедленно fail → чеки буферизируются
- **HALF_OPEN** (восстановление): пробный запрос каждые 60с

Реализация:

```
from circuitbreaker import circuit

class OFDCircuitBreaker:
    def __init__(self):
        self.failure_threshold = 5        # 5 ошибок подряд → OPEN
        self.recovery_timeout = 60        # 60с в OPEN → пробный запрос
        self.success_threshold = 3        # 3 успеха подряд → CLOSED
        self.expected_exception = (TimeoutError, ConnectionError, HTTPError)

    @circuit(failure_threshold=5, recovery_timeout=60, expected_exception=
(TimeoutError,))
    async def send_receipt(self, receipt):
        """
        Отправка чека в ОФД через Circuit Breaker

        CLOSED: запрос идёт к ОФД
        OPEN: немедленно CircuitBreakerError → чек в буфер
        HALF_OPEN: пробный запрос
        """
        response = await ofd_client.post("/receipts", json=receipt, timeout=10)
        return response
```

```
# Использование
ofd_cb = OFDCircuitBreaker()

async def fiscalize_receipt(receipt):
    try:
        result = await ofd_cb.send_receipt(receipt)
        return {"status": "synced", "fiscal_doc": result}
    except CircuitBreakerError:
        # Circuit Breaker открыт → ОФД недоступен
        logger.warning(f"Circuit breaker OPEN, buffering receipt
{receipt['id']}")
        buffer_db.execute(
            "INSERT INTO receipts (id, fiscal_doc, status) VALUES (?, ?,
'pending')",
            (receipt['id'], json.dumps(receipt))
        )
        return {"status": "buffered", "reason": "ofd_circuit_open"}
```

Мониторинг:

```
# Prometheus metrics
ofd_circuit_state = Gauge('ofd_circuit_breaker_state', 'Circuit breaker state',
['pos_id'])
# 0 = CLOSED, 1 = OPEN, 2 = HALF_OPEN

# Алерт в Grafana
ALERT OFDCircuitOpen
  IF ofd_circuit_breaker_state == 1
  FOR 5m
  LABELS { severity="P2" }
  ANNOTATIONS {
    summary="ОФД circuit breaker открыт на {{ $labels.pos_id }}",
    description="ОФД недоступен более 5 минут, чеки буферизируются"
  }
```

Преимущества:

- **Быстрый fail:** не тратим 10с на каждый таймаут (вместо этого мгновенная буферизация)
- **Защита ресурсов:** CPU/память не расходуются на бесполезные запросы
- **Автовосстановление:** через 60с пробный запрос (half-open)
- **Метрики:** легко отслеживать состояние (closed/open/half-open)

3.7 Развёртывание

Docker Compose структура

```yaml
version: '3.8'

services:
  odoo:
    image: odoo:17
    depends_on:
      - postgres
      - redis
    environment:
      - DB_HOST=postgres
      - DB_USER=${DB_USER}
      - DB_PASSWORD=${DB_PASSWORD}
    volumes:
      - ./addons:/mnt/extra-addons
      - odoo-data:/var/lib/odoo

  postgres:
    image: postgres:15-alpine
    environment:
      - POSTGRES_DB=optics_erp
      - POSTGRES_USER=${DB_USER}
      - POSTGRES_PASSWORD=${DB_PASSWORD}
    volumes:
      - postgres-data:/var/lib/postgresql/data

  redis:
    image: redis:7-alpine
    command: redis-server --appendonly yes
    volumes:
      - redis-data:/data

  celery-worker:
    image: odoo:17
    depends_on:
      - postgres
      - redis
    command: celery -A odoo worker -l info -Q default,priority
    environment:
      - CELERY_BROKER_URL=redis://redis:6379/0

  nginx:
    image: nginx:alpine
    ports:
      - "443:443"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
      - ./certs:/etc/nginx/certs:ro
    depends_on:
      - odoo

  # Адаптер ККТ (деплоится на кассовом терминале)
  kkt-adapter:
```

```
    build: ./kkt_adapter
    restart: unless-stopped
    devices:
      - /dev/ttyUSB0:/dev/ttyUSB0  # ККТ по USB
    environment:
      - ODOO_URL=${ODOO_URL}
      - ODOO_JWT_SECRET=${JWT_SECRET}
      - OFD_URL=${OFD_URL}
      - OFD_API_KEY=${OFD_API_KEY}
      - BUFFER_CAPACITY=200
      - HEARTBEAT_INTERVAL=30
      - SYNC_INTERVAL=300  # 5 мин
    volumes:
      - kkt-data:/app/data  # SQLite buffer + logs
      - /etc/localtime:/etc/localtime:ro  # Синхронизация времени

volumes:
  odoo-data:
  postgres-data:
  redis-data:
  kkt-data:
```

Переменные окружения (`.env`)

```
# Odoo
DB_USER=odoo
DB_PASSWORD=<secure_password>
ODOO_ADMIN_PASS=<admin_password>

# Адаптер ККТ
ODOO_URL=https://odoo.example.com
JWT_SECRET=<jwt_secret_key>
OFD_URL=https://ofd.example.com/api/v1
OFD_API_KEY=<ofd_api_key>

# Безопасность
POSTGRES_PASSWORD=<pg_password>

# Мониторинг
SENTRY_DSN=<sentry_dsn>  # Опционально
```

Миграции

- Через data files и `migrations/` в модулях Odoo
- **Для SQLite (офлайн-буфер):** Alembic для версионирования схемы
- **Rollback-процедура:**
  - Миграции Odoo: down-скрипты в `migrations/{version}/pre.py` и `post.py`
  - SQLite: Alembic downgrade

- Откат Docker-образов: `docker-compose down && docker-compose up -d --scale kkt-adapter=0` → откат → `--scale kkt-adapter=1`

3.8 Логирование и мониторинг

Журналы (структурированный JSON)

- **Odoo:** POS события, вызовы адаптера ККТ, результаты импорта
- **Адаптер ККТ:**
  - Все события офлайн-буфера (добавление/синхронизация/переполнение)
  - Heartbeat к Odoo (каждые 30с)
  - Попытки фискализации (success/failure)
  - Изменения статуса сети (online↔offline)
- **Формат:**

```
{
  "timestamp": "2025-10-05T14:23:45.123Z",
  "level": "INFO",
  "component": "kkt_adapter",
  "event": "receipt_buffered",
  "receipt_id": "550e8400-e29b-41d4-a716-446655440000",
  "pos_id": "POS-001",
  "buffer_size": 47,
  "network_status": "offline"
}
```

Алерты (критичные для офлайн-режима)

Алерт	Условие	Уровень	Действие
Офлайн-буфер заполнен ≥80%	`buffer_size >= 160`	P2 (Warning)	Уведомление администратора, мониторинг
Офлайн-буфер заполнен ≥90%	`buffer_size >= 180`	P1 (Critical)	Эскалация, подготовка к ручной синхронизации
Офлайн-буфер переполнен	`buffer_size >= 200`	P1 (Critical)	Блокировка продаж, экстренная синхронизация
Задержка чеков > 60 сек	`receipt_latency_p95 > 60s`	P2	Проверка ККТ/сети
Ошибка фискализации (ККТ)	`kkt_error`	P1	Диагностика устройства ККТ
Провал импорта	`import_failed`	P3	Проверка логов, повтор
Heartbeat к Odoo потерян >5 мин	`heartbeat_lost > 300s`	P2	Проверка сети, перезапуск адаптера

Алерт	Условие	Уровень	Действие
Синхронизация буфера >20 мин	`sync_duration > 1200s`	P1	Проверка канала до ОФД, повтор
ФН заполнен <100 чеков	`fn_capacity < 100`	P1	Замена ФН в течение 24ч
NTP drift >5 секунд	`ntp_drift > 5s`	P2	Проверка NTP-синхронизации
DLQ содержит >10 чеков	`dlq_size > 10`	P2	Ручная обработка чеков из DLQ

Метрики (для Prometheus/Grafana)

```
# Примеры метрик адаптера ККТ
kkt_buffer_size{pos_id="POS-001"}  # Gauge: текущий размер буфера
kkt_buffer_capacity  # Gauge: максимальная ёмкость (200)
kkt_receipts_total{status="synced|failed|pending"}  # Counter
kkt_sync_duration_seconds  # Histogram
kkt_receipt_latency_seconds  # Histogram: от создания до печати
kkt_network_status{status="online|offline|degraded"}  # Gauge (1/0)
kkt_heartbeat_failures_total  # Counter
kkt_ofd_errors_total{error_code="..."}  # Counter
kkt_dlq_size  # Gauge
```

Дашборд (критичные панели)

1. **Статус касс (real-time):**
   - Карта касс с цветовой индикацией (зелёный/жёлтый/красный)
   - Размер офлайн-буфера (progress bar)
   - Последний heartbeat
2. **Офлайн-буфер:**
   - График заполнения буфера по времени
   - Топ-5 касс по размеру буфера
   - Средняя длительность синхронизации
3. **Производительность:**
   - P95 латентности печати чека
   - Throughput синхронизации (чеков/мин)
   - Доля чеков с ретраями
4. **Надёжность:**
   - Uptime касс (бизнес-доступность)
   - Доля успешных синхронизаций

- Размер DLQ по кассам

3.8.1 Стратегия хранения и архивации логов

Проблема

40 касс × 100 событий/день × 365 дней = **1.46M событий/год**
Средний размер события: ~500 bytes
Итого: ~730 GB/год логов без компрессии

Решение: многоуровневое хранилище

Ротация логов (logrotate):

```
# /etc/logrotate.d/kkt_adapter
/var/log/kkt_adapter/*.log {
    daily
    rotate 30           # 30 дней онлайн (горячие)
    compress            # gzip (коэффициент ~10x)
    delaycompress       # не сжимать последний файл
    missingok
    notifempty
    postrotate
        # Отправка в долгосрочное хранилище (S3/Minio)
        aws s3 cp /var/log/kkt_adapter/*.gz \
          s3://optics_erp-logs/$(date +%Y/%m/%d)/ \
          --storage-class STANDARD
    endscript
}
```

Многоуровневое хранилище:

Период	Хранилище	Формат	Доступность	Стоимость/GB/мес	Retention policy
0-30 дней	Локальный SSD	JSON.gz	Мгновенная (grep)	$0.10 (вкл. в сервер)	rotate daily
31-90 дней	S3 Standard	JSON.gz	Секунды (S3 API)	$0.023	lifecycle → Glacier
91-365 дней	S3 Glacier	JSON.gz	Минуты (restore)	$0.004	lifecycle → Deep Archive
365+ дней	S3 Deep Archive	JSON.gz	Часы (restore)	$0.00099	delete after 1825 days (5 лет, 54-ФЗ)

Оценка затрат (20 точек, 40 касс):

```
events_per_day = 40 касс × 100 событий = 4000
size_per_event = 500 bytes
daily_size = 4000 × 500 = 2 MB

# С компрессией (10x)
compressed_daily = 2 MB / 10 = 0.2 MB

# Годовые затраты (при 40 кассах)
ssd_30d = 0.2 MB × 30 × $0.10 / 1024 = $0.0006/day
s3_60d = 0.2 MB × 60 × $0.023 / 1024 = $0.00027/day
glacier_275d = 0.2 MB × 275 × $0.004 / 1024 = $0.00021/day
deep_275d = 0.2 MB × 1000 × $0.00099 / 1024 = $0.00019/day

total_annual = ($0.0006 + $0.00027 + $0.00021 + $0.00019) × 365 = $0.43/год
```

Вывод: Затраты минимальны (<$1/год), архивация критична для compliance (90 дней онлайн + 5 лет архив по 54-ФЗ)

Lifecycle policy (S3)

```
{
  "Rules": [
    {
      "Id": "optics_erp-logs-lifecycle",
      "Status": "Enabled",
      "Transitions": [
        {
          "Days": 30,
          "StorageClass": "GLACIER"
        },
        {
          "Days": 365,
          "StorageClass": "DEEP_ARCHIVE"
        }
      ],
      "Expiration": {
        "Days": 1825
      }
    }
  ]
}
```

Поиск по архивным логам

Восстановление из Glacier (для инцидента 3-месячной давности):# OpticsERP — 3. Технологический стек и требования к реализации

3.9 Конфигурация проекта

`config.toml` (адаптер ККТ)

```toml
[adapter]
version = "1.0.0"
environment = "production"  # dev|staging|production

[buffer]
capacity = 200
fifo_enabled = true
auto_sync_interval = 300  # 5 мин
max_retry_attempts = 20  # Перед отправкой в DLQ
initial_backoff = 5  # секунды
max_backoff = 300  # 5 мин
dlq_retention_days = 90
synced_retention_days = 7

[cache]
max_products = 1000
refresh_interval = 3600  # 1 час
fallback_timeout = 2  # секунды для запроса к Odoo

[heartbeat]
interval = 30  # секунды
timeout = 2
max_failures = 3  # Перед переходом в автономный режим

[kkt]
device_path = "/dev/ttyUSB0"  # Или TCP: "tcp://192.168.1.100:5000"
connection_timeout = 5
read_timeout = 10
retry_attempts = 3

[ofd]
api_url = "${OFD_URL}"
api_key = "${OFD_API_KEY}"
timeout = 10
verify_ssl = true

[security]
jwt_secret = "${JWT_SECRET}"
jwt_ttl = 900  # 15 мин
mtls_cert = "/etc/kkt/certs/client.crt"
mtls_key = "/etc/kkt/certs/client.key"
mtls_ca = "/etc/kkt/certs/ca.crt"

[monitoring]
sentry_dsn = "${SENTRY_DSN}"
log_level = "INFO"  # DEBUG|INFO|WARNING|ERROR
metrics_port = 9090  # Prometheus exporter
```

`config.toml` (проект Odoo)

См. оригинальный `config.toml` (проект) + дополнения:

```
[features]
offline_mode = true  # Обязательно для продуктива
marking_enabled = false  # Feature flag для маркировки
electronic_receipt = true

[pos]
receipt_timeout = 7  # P95 SLA, секунды
max_discount_percent = 50
x_report_roles = ["cashier", "manager"]
z_report_roles = ["manager", "admin"]

[offline]
buffer_warning_threshold = 0.8  # 80%
buffer_critical_threshold = 0.9  # 90%
alert_channels = ["email", "telegram"]  # Каналы для алертов
import_block_on_unsync = true  # Блокировать импорт при несинхронизированных
буферах

[retention]
audit_logs_online_days = 30
audit_logs_archive_days = 60
import_logs_days = 30
backup_retention_days = 14
```

3.10 UX/Интерфейсы

Формы и UI (общие)

- Формы рецептов и заказов на изготовление с валидацией диапазонов.
- POS-интерфейс: быстрый поиск по SKU/EAN, горячие клавиши, виджеты скидок.
- Превью импорта с пагинацией и фильтрами ошибок.

UI офлайн-режима (новое)

Статус-бар (постоянная индикация):

```
┌──────────────────────────────────────────────────────┐
│ [●] Касса: POS-001  |  [●] Сеть: Офлайн 2ч 15мин     │
│ [!] Буфер: 34/200 чеков (17%)                        │
└──────────────────────────────────────────────────────┘

Цветовая схема:
[●] Зелёный  - online, буфер <80%
[●] Жёлтый   - degraded или буфер 80-90%
```

```
[●] Красный  - offline или буфер >90%
[!] Оранжевый - предупреждение
```

Tooltip при наведении на индикатор:

```
Офлайн-режим активен
─────────────────────

Последняя синхронизация: 2ч 15мин назад
Чеков в буфере: 34 из 200 (17%)
Ожидаемое время синхронизации: ~2 мин

Все чеки сохранены и будут отправлены
в ОФД автоматически при восстановлении связи.
```

Предупреждение при достижении 80% буфера:

```
┌──────────────────────────────────────────────┐
│ ⚠  ВНИМАНИЕ: Офлайн-буфер заполнен на 80%    │
│                                              │
│ В буфере 160 чеков из 200.                   │
│ При достижении 200 чеков продажи будут       │
│ автоматически заблокированы.                 │
│                                              │
│ [Вызвать администратора]  [Понятно]          │
└──────────────────────────────────────────────┘
```

Блокировка при переполнении буфера (100%):

```
┌──────────────────────────────────────────────┐
│ 🔘  НЕВОЗМОЖНО ПРИНЯТЬ ОПЛАТУ                │
│                                              │
│ Офлайн-буфер переполнен (200/200 чеков).     │
│ Дождитесь восстановления связи или           │
│ вызовите администратора для ручной           │
│ синхронизации.                               │
│                                              │
│ [Вызвать администратора]  [Отмена]           │
└──────────────────────────────────────────────┘
```

Страница "Офлайн-буфер" (для администратора):

```
Офлайн-буфер - POS-001
═══════════════════════════════════════
```

```
Статус: [●] Офлайн           Буфер: 47/200 (24%)
Последняя синхронизация: 45 мин назад

[Принудительная синхронизация]  [Обновить]

Последние чеки (10 из 47):
┌──────────────┬──────────────┬──────────┬──────────┬──────────┐
│ ID           │ Время        │ Сумма    │ Статус   │ Действ.  │
├──────────────┼──────────────┼──────────┼──────────┼──────────┤
│ 550e8400-... │ 14:23:45     │ 2500₽    │ Pending  │ [Лог]    │
│ 661f9511-... │ 14:18:32     │ 1800₽    │ Pending  │ [Лог]    │
│ ...          │ ...          │ ...      │ ...      │ ...      │
└──────────────┴──────────────┴──────────┴──────────┴──────────┘

DLQ (Dead Letter Queue): 2 чека требуют ручной обработки
[Открыть DLQ]
```

Логи офлайн-событий (доступ: кассир → последние 10, администратор → все):

```
14:23:45  [INFO]   Чек 550e8400 сохранен в буфер (офлайн-режим)
14:18:32  [INFO]   Чек 661f9511 сохранен в буфер (офлайн-режим)
14:15:00  [WARN]   Потеряна связь с ОФД (переход в офлайн)
12:00:00  [INFO]   Синхронизация завершена: 28 чеков отправлено
11:58:30  [INFO]   Восстановлена связь с ОФД
```

Индикация во время синхронизации:

```
┌──────────────────────────────────────────────┐
│  ⏳ Синхронизация офлайн-буфера...           │
│                                              │
│ [████████████▒▒▒▒▒▒▒▒] 65% (32/47 чеков)     │
│                                              │
│ Примерное время: 1 мин                       │
│                                              │
│ Не выключайте кассу.                         │
└──────────────────────────────────────────────┘
```