**1. Основная идея проекта**

Разрабатываемая система:

Информационная система (веб + мобильный модуль) для автоматизации учёта и управления запасами в розничной сети «МегаМаркет». Система поддерживает учёт товаров на складах и в филиалах, прогнозирование спроса, автоматизацию заказов у поставщиков, перемещения между филиалами, аналитические отчёты и интеграцию с кассовыми/ERP-системами.

Какие проблемы решает:

* Слишком высокие запасы и связанные с этим издержки (складские расходы, замороженный капитал).
* Частые острые дефициты товара на полках (out-of-stock).
* Ручное формирование заказов и ошибки при пересчёте остатков.
* Отсутствие прозрачных данных о движении между филиалами.
* Низкая точность планирования из-за отсутствия прогнозирования спроса.

Целевая аудитория:

* Оперативный персонал магазинов (кладовщики, менеджеры филиалов).
* Cлужба закупок / category managers.
* Логистика (координаторы перемещений между складами).
* Руководство и аналитики (отдел BI).
* IT/интеграторы (для настройки интеграций с 1C/ERP/кассами).

Роли и задачи в системе (пример):

* Администратор системы — настройка справочников, управление пользователями, права.
* Кладовщик / Оператор склада — приём/отгрузка, инвентаризация, перемещения.
* Менеджер по закупкам — формирование и утверждение закупок, подбор поставщиков.
* Аналитик / Руководитель филиала — мониторинг остатков, отчёты, KPI.
* Система/Интеграция (машинная роль) — обмен данными с POS/ERP/поставщиками.

**2. Анализ требований к проекту**

**2.1. Нефункциональные требования**

* Доступность: 99.5% SLA для веб-интерфейса; возможность оффлайн-работы мобильного приложения в филиалах с синхронизацией.
* Масштабируемость: поддержка десятков филиалов и сотен тысяч товарных позиций.
* Производительность: отклик UI < 1,5 с для основных операций; отчёты — до 10 с.
* Безопасность: аутентификация, RBAC, шифрование данных в покое и при передаче.

**2.2. Функциональные требования (основные)**

1. Учёт товаров: карточки товара, серийные номера/сроки годности, партии.
2. Приём/Расход/Перемещения: операции прихода, расхода, внутренние перемещения между филиалами и складами.
3. Инвентаризация: разовые и плановые инвентаризации, сопоставление факта и учёта, списания.
4. Управление закупками: формирование заказов поставщикам (ручное/авто), снижение/увеличение заказов по правилам.
5. Прогнозирование спроса: модель прогнозирования для каждой SKU/филиала; рекомендации по заказу.
6. Автоматические уровни пополнения: MOQ, min/max/ROP (reorder point), safety stock, lead time.
7. Планирование поставок и контроль исполнения: статусы заказов, сроки, уведомления.
8. Отчёты и дашборды: обороты, оборачиваемость, стареющие товары, точность прогноза, fill rate, out-of-stock incidents.
9. Интеграции: POS/кассы, ERP (например 1С), поставщики (эл. заказы), WMS/логистика, бухгалтерия.
10. Пользовательские уведомления и workflow: алерты по дефициту, подтверждение заказов, согласования.
11. Аудит и логирование всех операций.

**2.3. Сценарии использования (user stories)**

* Как менеджер по закупкам, я хочу получать автоматические рекомендации по заказу для SKU с учётом прогноза и остатка.
* Как кладовщик, я хочу быстро оформить приход партии с указанием срока годности.
* Как директор филиала, я хочу видеть KPI по товарообороту и % недостающих товаров.

**3. Архитектура проекта**

**3.1. Компоненты системы**

1. Веб-приложение (Frontend) — админ-панель, дашборды, отчёты (React/Vue).
2. Мобильный клиент — iOS для кладовщиков.
3. Backend API — REST/GraphQL сервис(ы) (микросервисы или монолит с модульной структурой).
4. Сервис прогнозирования — отдельный ML-микросервис (Python) для обучения и выдачи прогнозов. (на будущее)
5. База данных транзакций — SQL (PostgreSQL).
6. Data Warehouse / OLAP — для аналитики: ClickHouse / BigQuery / Redshift (опционально).
7. Message Broker —Kafka для асинхронных задач (обмен с POS, очереди задач).
8. ETL / Integration layer — коннекторы к 1С, POS, поставщикам (API, SFTP, EDI).
9. UI/Reports — BI-инструмент (Metabase / Superset / Power BI) или встроенные отчёты.
10. Auth / IAM — сервис аутентификации (JWT).
11. Monitoring & Logging — Prometheus, Grafana.
12. CI/CD и инфраструктура — Docker, Kubernetes (k8s) / облачный PaaS.

**3.2. Взаимодействие компонентов (описание потока)**

* POS/кассы отправляют продажи → через ETL/Message Broker → Backend обновляет остатки в БД → триггер на пересчёт прогноза/оповещения.
* Backend предоставляет REST API для фронта и мобил.
* ML-сервис регулярно получает исторические данные (через ETL) → обучает модели → публикует прогнозы в БД/через API.
* Сервис закупок формирует заказ (ручной/авто) → отправляет поставщику через Integration layer (API/EDI) → статус обновляется в системе.
* Отчёты собираются из DWH/OLAP для быстрых аналитических запросов.

**3.3. Хранилище данных — модель (вкратце)**

* Таблицы: products, sku\_attributes, warehouses, stocks (warehouse\_id, sku\_id, quantity, reserved), movements, purchase\_orders, sales\_transactions, suppliers, forecast\_daily и т.д.

**4. Технологии (рекомендованный стек)**

**4.1. Backend & API**

* Язык: Python (FastAPI) или Node.js (NestJS).
* Архитектурный стиль: REST + GraphQL (для гибких запросов).
* Асинхронность: uvicorn/asyncio (если Python).
* Message Broker: Kafka (масштабируемость) или RabbitMQ (проще).

**4.2. Frontend & Mobile**

* Frontend: React + TypeScript, UI-библиотека (Ant Design / Material UI).
* Mobile: React Native или Flutter (одна кодовая база для Android/iOS).

**4.3. Базы данных и хранилища**

* OLTP: PostgreSQL (ACID и расширяемость).
* OLAP/аналитика: ClickHouse или Apache Druid (быстрые агрегации).
* Кеширование: Redis (сессии, кеш отчётов).
* Объектное хранилище: S3-совместимое для дампов, отчётов.

**4.4. ML / прогнозирование**

* Язык: Python (pandas, scikit-learn, xgboost, prophet, pytorch if needed).
* MLOps: базовая оркестрация (Airflow / Prefect) для ETL и расписания обучения.

**4.5. Инфраструктура и DevOps**

* Контейнеризация: Docker.
* Оркестрация: Kubernetes (EKS/GKE/AKS), либо Docker Swarm для малого масштаба.
* CI/CD: GitHub Actions / GitLab CI.
* Мониторинг: Prometheus + Grafana, логирование через ELK.

**4.6. Безопасность и аутентификация**

* OAuth2/OpenID Connect (Keycloak или Auth0).
* TLS everywhere (HTTPS), шифрование полей (PII) в БД.

**5. Вопросы безопасности (конкретные меры)**

**5.1. Аутентификация и авторизация**

* Внедрить RBAC (ролевой доступ) с тонкой гранулярностью.
* Поддержка SSO (AD/LDAP) для корпоративных пользователей.

**5.2. Защита передачи и хранения данных**

* TLS 1.2+ для всех соединений.
* Шифрование чувствительных полей (например, номера карт поставщиков) в БД с KMS (Key Management Service).
* Ротация ключей доступа.

**5.3. Защита API**

* Ограничение скорости (rate limiting), защита от DDoS (WAF).
* Валидация входных данных, защита от SQL-инъекций, XSS, CSRF.
* Использование подготовленных выражений/ORM.

**5.4. Аудит и логирование**

* Журнал всех критичных операций (кто, что, когда). Логи в централизованном хранилище (ELK) с защитой доступа.
* Настроить оповещения при подозрительной активности.

**5.5. Безопасность ML/данных**

* Псевдонимизация персональных данных в тренировочных датасетах.
* Тесты на целостность данных и мониторинг дрейфа модели.

**5.6. Процедуры безопасности**

* Регулярные pentest/сканирование уязвимостей.
* Процедуры резервного копирования и тестирование восстановления (RTO/RPO определены).

**6. План разработки (этапы, сроки и ресурсы)**

Оценка: проект средней сложности. Примерный план — 7 месяцев (можно адаптировать к доступным ресурсам).

**Фазы и сроки (пример)**

1. Предпроектный этап / Сбор требований — 2 недели
   * Встречи с заказчиком, картирование процессов, сбор сценариев.
2. Техническое задание и проектирование (UX + архитектура) — 3 недели
   * Функциональные спецификации, ERD, API contract, макеты.
3. Proof of Concept (PoC) и ML-прототип — 4 недели
   * Минимальный прототип учёта + простой ML прогноз для 100 SKU.
4. Реализация ядра (MVP) — 12 недель
   * Backend API, Frontend дашборд, мобильный приёмщик, интеграция с 1С/POS (минимально).
5. Тестирование и доработки — 6 недель
   * Функциональное, интеграционное, нагрузочное тестирование.
6. Внедрение и пилот в 1–2 филиалах — 3 недели
   * Настройка, миграция данных, обучение персонала.
7. Масштабирование и ввод в эксплуатацию — 2–4 недели
   * Производственный запуск, поддержка, сбор обратной связи.

**Команда (рекомендуемая)**

* Проектный менеджер (1)
* Аналитик/BA (1)
* Архитектор/Lead backend (1)
* Backend-разработчики (2)
* Frontend-разработчик (1–2)
* Mobile-разработчик (1)
* Data Engineer (1)
* ML-инженер/DS (1)
* QA-инженер (1–2)
* DevOps (1)
* Бизнес-специалист / представитель заказчика (1–2)

**Управление проектом**

* Методология: Agile / Scrum (спринты 2 недели).
* Еженедельные демо, backlog grooming, retrospective.
* Риски: некорректные входные данные, сложная интеграция с 1С, задержки со стороны поставщиков — подготовить план смягчения.

**7. Макеты / прототипы интерфейса (текстовые прототипы)**

Ниже — упрощённые wireframe-описания основных экранов. Они помогут быстро представить UX; позже можно сделать кликабельный прототип (Figma).

**7.1. Дашборд руководителя**

* Заголовок: «Дашборд — МегаМаркет»
* KPI сверху: Общие остатки, % заполнения полок, Fill Rate (за период), Средняя оборачиваемость (дни), Value стареющих запасов.
* Графики: продажи по категориям (за 30д), прогноз спроса (по ключевым SKU), heatmap по филиалам (дефицит/избыток).
* Список предупреждений: товары с низким запасом, задержанные поставки.

**7.2. Карточка товара (SKU)**

* Фото, название, категория, артикул, UPC.
* Текущие остатки по складам (таблица), зарезервировано, ожидается (PO).
* История движения (таблица/фильтр по датам).
* Рекомендации: рекомендованный заказ (кол-во), safety stock, lead time.

**7.3. Модуль формирования заказов**

* Фильтры: филиал/поставщик/категория.
* Авто-сгенерированный список SKU с предложениями; чекбоксы для подтверждения; кнопка «Сформировать PO», вариант — «Отправить поставщику».

**7.4. Мобильный экран (для кладовщика)**

* Экран «Приём партии»: сканирование штрих-кода → выбор PO → ввод количества и срока годности → подтвердить.
* Экран «Инвентаризация»: список SKU, ввод факта, комментарий, сохранить и синхронизировать.

(Если нужно, могу сгенерировать простые SVG/PNG прототипы — напишите, желаете ли визуальные макеты.)

**8. Стратегия тестирования**

**8.1. Типы тестов**

* Unit tests — логика backend, ML-препроцессинг.
* Integration tests — интеграции с POS/1С/поставщиками, очередями.
* End-to-end (E2E) — сценарии: приём товара, продажа, автоматический заказ.
* Load / Performance tests — моделирование пиковых продаж, нагрузка на отчёты.
* Security tests / Penetration tests — внешние и внутренние.
* User Acceptance Testing (UAT) — пилотные пользователи из «МегаМаркет».

**8.2. Метрики успешности**

* Точность прогноза: MAPE < целевой (например, < 15% для основных SKU) — согласовать с заказчиком.
* Снижение out-of-stock incidents на X% в пилоте (цель — 30–50%).
* Снижение среднего уровня запасов (капитала) на Y% без ухудшения fill rate.
* Время отклика основных операций < 1.5 с.
* Уровень багов: критических багов = 0 в продакшене в течение 30 дней после релиза.

**9. Документация проекта**

Документация должна быть мультиуровневая и доступна в репозитории / Confluence:

1. Техническое задание (ТЗ) — функциональные и нефункциональные требования.
2. Архитектурная документация — диаграммы, ERD, sequence diagrams, API contracts (OpenAPI/Swagger).
3. Руководства пользователя — по ролям: кладовщик, менеджер закупок, аналитик.
4. DevOps / Infra docs — деплой, конфигурации, backup/restore, runbooks.
5. Security policy — RBAC matrix, правила хранения и обработки данных.
6. MLOps docs — pipeline обучения, метрики моделей, валидация.
7. Тестовая документация — планы тестирования, чек-листы, отчёты о тестах.
8. API документация — OpenAPI спецификации, примеры запросов.
9. Release Notes — журнал обновлений.
10. FAQ и Troubleshooting — частые проблемы и их решения.

**10. Общий вывод — знания и улучшения**

**Что можно получить в ходе разработки**

* Практический опыт проектирования интегрированных решений (WMS + POS + ERP).
* Навыки построения ML-пайплайна для прогнозирования спроса в ритейле.
* Опыт работы с микросервисами, очередями сообщений и высоконагруженными системами.
* Понимание бизнес-процессов розничной сети и оптимизации запасов.

**Возможные доработки и улучшения (фичи «на будущее»)**

* Автоматизация динамического ценообразования (markdowns для стареющих товаров).
* Оптимизация маршрутов перемещения на основе налогов/стоимости перевозки.
* Advanced ML: модели с учётом промо-кампаний, погоды, событий.
* Симулятор запасов (what-if analysis) для оценки сценариев закупок.
* Полная интеграция с поставщиками через EDI и автоматическое согласование сроков.
* Интернет вещей (IoT): датчики на складах для контроля температуры/влажности (для скоропортящихся товаров).

Если хотите, могу сразу:

* Сгенерировать чёткий план задач (backlog) с задачами для спринтов (в формате Jira-эпиков/тасков).
* Сделать ER-диаграмму и пример OpenAPI для ключевого API (products, stocks, movements, PO).
* Подготовить визуальные макеты (PNG) основных экранов (dashboard, карточка товара, мобильный экран приёма).

Скажите, что предпочтительнее — я сразу подготовлю выбранный артефакт в следующем сообщении (backlog / ERD / OpenAPI / макеты).