🧾 Техническое задание (ТЗ)

Проект: FUCENT (маркетплейс-платформа в стиле Wildberries + Instagram)
Версия: v 2.1 **Дата:** ноябрь 2025 **Backend:** Java (Spring Boot + PostgreSQL) **Frontend:** Angular **Mobile:** Flutter

---

1. Цели проекта

Создать универсальную торгово-социальную платформу, объединяющую магазины, рынки и покупателей в одном цифровом пространстве.
Основные функции:

- просмотр и покупка товаров;
- публикации магазинов (лента, сторис, прямые эфиры);
- чаты и взаимодействие продавец ↔ покупатель;
- карта точек продаж;
- возможность онлайн-оплаты и учёта остатков через POS-устройство.

---

2. Основные роли пользователей

Роль	Описание	Основные права
Клиент (User)	Покупатель, использующий веб или мобильное приложение.	Просмотр каталога, добавление в корзину, заказы, подписки, лайки, комментарии, чат с магазинами.
Продавец (Merchant Owner)	Юр/физлицо, владелец одного или нескольких магазинов.	Управление товарами, заказами, публикациями, аналитикой, настройками выплат и POS.
Магазин (Shop)	Конкретная торговая точка продавца (в т. ч. рынок).	Имеет адрес, контакты, геолокацию, товары и статус POS.

Администратор (Admin)	Модератор и оператор площадки.	Управление пользователями, жалобами, категориями, тарифами, модерацией публикаций и товаров.

---

3. Архитектура системы

3.1 Общая схема

Backend:

- Spring Boot 3 + JPA (Hibernate) + PostgreSQL;
- структура слоёв: Controller → Service → Repository;
- JWT-аутентификация, BCrypt-хэширование паролей;
- REST API v1;
- Swagger / OpenAPI 2.6.0;
- Docker-развёртывание, GitLab CI/CD.

Frontend (Angular Web):

- SPA с маршрутизацией по разделам;
- глобальная навигационная панель (Лента / Каталог / Чат / Корзина / Профиль);
- поддержка Push/SMS/email уведомлений.

Mobile (Flutter):

- единое приложение с режимами Покупатель/Продавец;
- навигация адаптирована под мобильный UX.

---

4. Модули и подсистемы

4.1 Аутентификация и пользователи

- Регистрация / логин через email + пароль.
- JWT-токены (access + refresh).
- BCrypt для хэширования.
- Audit log действий пользователей и админов.
- Возможность 2FA для продавцов и админов (опционально).

Поля таблицы `app_user`:
`id UUID`, `email`, `password_hash`, `role ENUM(user, merchant, admin)`, `name`, `surname`, `avatar_url`, `created_at`.

4.2 Торговые сущности

4.2.1 Merchant

```
id UUID PK
owner_user_id UUID (FK→app_user)
name
description
payout_account_number nullable
payout_bank_name nullable
payout_status ENUM(pending, active, suspended)
buy_eligibility ENUM(manual_contact, online_purchase)
settings_json (jsonb)
created_at timestamp
```

4.2.2 Shop

```
id UUID PK
merchant_id FK→merchant
name
address
phone
lat, lon
pos_status ENUM(inactive, active)
created_at
```

Магазин может иметь несколько точек.
POS интеграция обновляет остатки и чеки в реальном времени.

4.3 Каталог товаров

4.3.1 Category

- централизованная структура категорий (аналог Wildberries).
- поля: `id`, `parent_id`, `name_ru`, `name_kg`, `name_en`, `slug`.

4.3.2 Product

```
id UUID
shop_id FK→shop
category_id FK→category
name
description
status ENUM(active, hidden, archived)
created_at
```

4.3.3 ProductVariant (JSON-атрибуты)

```
id UUID
product_id FK→product
sku STRING unique
barcode STRING unique or null
attributes_json (json) – size, color, gender …
price DECIMAL
stock_qty INT
```

- Атрибуты хранятся в JSON.
- Для online_purchase SKU и barcode обязательны.

4.4 POS интеграция

- Продавец указывает, есть ли у него POS-устройство.
- При активации POS — включается автоматический учёт продаж и остатков.
- Таблицы:

    - `pos_sale` – каждая продажа или возврат;
    - `pos_summary_daily` – агрегированные данные по дням.
- Если POS отправляет heart-beat > 10 мин назад → кнопка «Купить» скрывается.

🧩 Часть 2 — Социальный функционал, взаимодействие и контент

---

5. Лента (Feed)

5.1 Общие принципы

- Публикации создаются магазинами (и в будущем — пользователями).
- Поддерживаются изображения, видео и текстовое описание.
- В каждой публикации может быть **привязка к товару (`product_id`)**, что позволяет пользователю перейти на карточку товара.
- Если привязка отсутствует — кнопка **«Перейти на товар»** не отображается.

Поля таблицы `post`:

```
id UUID
shop_id FK→shop
author_user_id FK→app_user
media_urls TEXT[]
caption TEXT
product_id FK→product nullable
likes_count INT
comments_count INT
created_at TIMESTAMP
```

API `/api/v1/feed`

- `GET /feed/main` — общая лента (рекомендации + тренды);
- `GET /feed/subscriptions` — публикации из подписок;
- `POST /feed` — создать пост (с optional productId);
- `POST /feed/{id}/like` — лайкнуть;
- `POST /feed/{id}/comment` — добавить комментарий;
- `GET /feed/{id}` — получить публикацию;
- `GET /feed/{id}/comments` — список комментариев.

5.2 Разделы ленты

Раздел	Описание
Подписки	Публикации магазинов, на которые подписан пользователь.
Рекомендации	Похожие публикации по интересам, активности и регионам (аналог Instagram Explore).
Лента (Тренды)	Популярные публикации, выбранные по алгоритму трендов.

5.3 Сторис и прямые эфиры

- В верхней части ленты — **аватарки продавцов**, у которых активны сторис или прямые эфиры.
- Продавцы могут загружать сторис (видео/фото), срок хранения — 24 часа.
- Прямой эфир отображается с красной обводкой аватарки.
- API: `/api/v1/stories` и `/api/v1/stream`.

5.4 Алгоритм трендов (v1)

Цель: формировать вкладку “Тренды” с учётом вовлечённости и свежести контента.

Факторы ранжирования:

Показатель	Вес
Лайки / комментарии / сохранения	0.4
CTR (клики на пост / время просмотра)	0.2
Свежесть публикации	0.2
Гео-близость пользователя	0.1
Репутация магазина (рейтинг, активность)	0.1

Механизм:

- расчет рейтинга в отдельной таблице `post_trend_score`;
- обновление каждые 30 минут (через cron или scheduler);
- конфигурация весов хранится в `feed_trend_weights`.

---

6. Чаты и коммуникации

6.1 Общая структура

Единая таблица чатов:

```
chat:
id UUID
type ENUM('user_user','user_shop')
initiator_id UUID
recipient_id UUID
created_at TIMESTAMP
last_message_at TIMESTAMP
```

Сообщения:

```
message:
id UUID
```

```
chat_id FK→chat
sender_id UUID
content TEXT
type ENUM('text','product','post')
created_at TIMESTAMP
```

6.2 Функциональность

- Продавцы и покупатели общаются напрямую.
- Для покупателей — чат создаётся по запросу.
- Для магазинов — чат создаётся автоматически при первом сообщении.
- В чате можно:
    - писать текстовые сообщения;
    - отправлять ссылки на товары или публикации;
    - делиться контентом через встроенное окно (попап 30–70% экрана).
- Сообщения сортируются по `last_message_at`.

6.3 Поиск и фильтрация

- Встроенный поиск по имени, фамилии, username или названию магазина.
- UI-аналог поиска в Instagram: показываются имя, username, аватар.
- Результаты фильтруются по типу (user или shop).

6.4 Структура API

```
GET  /api/v1/chat       – список чатов
GET  /api/v1/chat/{id}  – сообщения в чате
POST /api/v1/chat       – создать новый чат (если не существует)
POST /api/v1/chat/{id}/message – отправить сообщение
GET  /api/v1/chat/search?q=... – поиск пользователей/магазинов
```

---

7. Каталог и карта магазинов

7.1 Каталог товаров

- Основная витрина со всеми активными товарами.
- Фильтры по категориям, цене, магазину, наличию.
- Возможность сортировки по рейтингу и популярности.
- В верхней части — кнопка **“Карта”**, открывающая карту с магазинами.
- При выборе магазина — всплывающее окно с товарами и контактами.

7.2 Карта магазинов

- Отображает все магазины с координатами `lat`, `lon`.
- Возможна кластеризация точек на больших масштабах.
- Клик по маркеру → открывает карточку магазина.
- Карта интегрируется через Google Maps / 2GIS API.
- При необходимости — локальный кэш тайлов.

---

8. Корзина и заказы

8.1 Корзина

- Пользователь добавляет товары в корзину (`cart_item`).
- Возможность изменить количество, удалить, оформить заказ.
- Проверка доступности товара при переходе к оплате:
    - если магазин с POS → проверка через API;
    - если без POS → отображается предупреждение “наличие уточняйте”.

```
cart_item:
id UUID
user_id FK→app_user
product_variant_id FK→product_variant
qty INT
created_at
```

8.2 Оформление заказа

- Один заказ — один магазин (несколько товаров).
- Резервирование товара на 30 минут до оплаты.
- После оплаты — резерв на 24 часа до самовывоза.
- Поля заказа:

```
order:
id UUID
user_id
shop_id
status ENUM('pending','paid','cancelled','completed')
total_amount DECIMAL
payment_status ENUM('pending','paid','refunded')
created_at
```

---

9. Профиль пользователя

9.1 Настройки и персонализация

- Изменение имени, фамилии, email и аватара.
- Настройка языка интерфейса (ru, kg, en, zh, kz, uzb).
- Раздел “Сохранённые публикации” и “Избранные товары”.
- Возможность деактивировать аккаунт.

9.2 UI и навигация

Навигационная панель (Web):

- фиксирована снизу;
- кнопки: **Лента**, **Каталог**, **Чат**, **Корзина**, **Профиль**.

Навигация (Mobile Flutter):

- адаптивный низ экрана (TabBar);
- кнопка “+” для быстрой публикации;
- “Чат” и “Корзина” — плавающие иконки.

---

10. Социальные взаимодействия и контент

10.1 Подписки и лайки

- Пользователь может подписываться на магазины.
- Лайки и комментарии доступны всем пользователям.
- Отображается бейдж “Verified Purchase”, если пользователь реально купил товар.

10.2 Комментарии

- Возможность оставлять комментарии под постами и товарами.
- Антиспам:
    - задержка между постами/комментариями 10–30 сек;
    - фильтрация запрещённых слов;
    - лимит длины комментария;
    - автоудаление при репортах.

---

11. Алгоритмы рекомендаций и аналитика

11.1 Персонализация

- Формирование рекомендаций по лайкам, подпискам и гео.
- Умеренный уровень персонализации (без heavy ML на старте).
- Весовая модель:
    - взаимодействия пользователя;
    - активность магазина;
    - CTR публикации;
    - время жизни контента.

11.2 Аналитика магазинов

- Метрики:
    - просмотры профиля;
    - переходы на товары;
    - клики, лайки, комментарии;
    - конверсия “в корзину” и “в заказ”.
- Отчёты доступны в личном кабинете продавца.
- Возможность выгрузки в CSV/XLSX.

⚙️ Часть 3 — Уведомления, платежи, тарифы, мультиязычность, безопасность и инфраструктура

---

12. Уведомления и коммуникация с пользователем

12.1 Типы уведомлений

Система уведомлений поддерживает три канала:

Тип уведомления	Назначение	Канал
Push	Основной канал для мобильных клиентов (Flutter).	Firebase Cloud Messaging (FCM)
SMS	Альтернатива при отсутствии интернет-доступа.	Провайдер: TBD (будет выбран для КР)
Email	Системные уведомления, активация, восстановление пароля.	SMTP / Mailgun / SendGrid (TBD)

12.2 Архитектура уведомлений

- Все уведомления обрабатываются асинхронно через очередь сообщений (`notification_queue`).
- Каждое событие (новый заказ, сообщение, лайк, ответ и т.д.) → публикуется в очередь, затем обработчик формирует сообщение и отправляет по нужному каналу.
- Используются отдельные таблицы:
    - `notification_template` — хранение шаблонов;
    - `notification_event` — события и статусы отправки;
    - `user_preferences` — настройки уведомлений по типам.

Пример структуры таблицы `notification_event`:

```
id UUID
user_id UUID
type ENUM('order','chat','system','marketing')
channel ENUM('push','sms','email')
status ENUM('pending','sent','failed')
payload JSONB
created_at TIMESTAMP
```

12.3 События-триггеры

Событие	Описание	Каналы
Новый заказ	Клиент оформил заказ	Push + Email продавцу
Новое сообщение	Пришло сообщение в чат	Push
Новый комментарий	На публикацию или товар	Push
Новый лайк	На публикацию	Только Push
Смена статуса заказа	“Оплачен”, “Готов к выдаче”	Push + Email
Обновление POS/остатков	Для владельца магазина	Email

12.4 Антиспам и ретраи

- Повторная отправка (`retry_count <= 3`) при сбое.
- Уведомления дедуплицируются по типу и контексту (`unique_hash`).
- Push отправляются с TTL 24ч.
- Email и SMS — сохраняются в журнал для отчётности.

---

13. Платежи и выплаты

13.1 Архитектура

- Платёжная схема поддерживает **оплату онлайн и выплаты продавцам**.
- Используется абстракция `PayoutProvider` с режимами:
  - `instant` — мгновенная выплата (по факту оплаты);
  - `T+1` — выплата на следующий день.
- В MVP: фактические платежи не активны, реализованы как мок-интерфейсы.

13.2 Комиссия и эквайринг

Параметр	Значение
Комиссия площадки (take rate)	% от заказа или фикс/гибрид. В БД хранится значение, по умолчанию 0%.
Эквайринг оплачивает	Продавец.
Выплаты продавцу	Моментальные после подтверждения оплаты.
Окно возврата	7–14 дней, гибко настраивается.
Частичный возврат	Не поддерживается.
Ответственность за возврат	В зависимости от причины — продавец или платформа.

13.3 Таблицы и поля

payout_account (nullable пока):

```
merchant_id FK→merchant
account_number
bank_name
provider ENUM('manual','providerX','providerY')
status ENUM('pending','verified','blocked')
```

transaction:

```
id UUID
order_id FK→order
amount DECIMAL
commission DECIMAL
status ENUM('pending','success','failed')
created_at
```

---

14. Подписки и тарифы

14.1 Общая модель монетизации

- Базовый доступ — **FREE**;

- Расширенные возможности — **PRO / ENTERPRISE**.

Уровень	Особенности
FREE	До N товаров, базовые метрики.
PRO	Больше лимитов, реклама, буст публикаций, аналитика.
ENTERPRISE	Полный доступ, приоритетная поддержка, API-интеграции.

14.2 Подписки и триалы

- Продавцу при регистрации активируется триал 14 дней.
- Автопродление включено по умолчанию, можно отключить вручную.
- Данные о подписке:

```
subscription:
id UUID
merchant_id FK→merchant
plan ENUM('free','pro','enterprise')
start_date DATE
end_date DATE
auto_renew BOOLEAN
status ENUM('active','expired','cancelled')
```

---

15. Мультиязычность

15.1 Поддерживаемые языки

Платформа работает с шестью языками (в порядке приоритета):
`ru`, `kg`, `en`, `zh`, `kz`, `uzb`.

- В MVP отображаются только **RU**, остальные — пустые поля, но в БД и API уже поддерживаются.
- Все текстовые поля в таблицах `category`, `attribute`, `product` имеют префиксы `_ru`, `_kg`, `_en` и т.д.

- Интерфейс клиентских приложений поддерживает переключение языка.
- Категории и атрибуты переводятся централизованно.

---

16. Безопасность и аудит

16.1 Аутентификация

- JWT-токены (access + refresh).
- Хэширование паролей — BCrypt.
- Возможность включить **двухфакторную аутентификацию** для продавцов и админов.

16.2 Логирование

- Логи аудита (действия пользователей и админов):
    - входы/выходы;
    - изменения товаров, заказов, публикаций;
    - операции с POS и выплатами.
- Хранятся в таблице `audit_log` (логические типы: user_action, system, error).

16.3 Политика хранения данных

Тип данных	Срок хранения
Сырые логи	90 дней
Медиа (посты, фото)	бессрочно
События аналитики	13 месяцев
Уведомления и транзакции	1 год

---

17. Система аналитики

17.1 Метрики

- **Traffic analytics:** клики, просмотры, CTR.
- **Sales analytics:** конверсии, средний чек, товарооборот.
- **Behavior analytics:** время на странице, взаимодействия, пути перехода.

17.2 Структура хранения

- Вся аналитика сохраняется в PostgreSQL.
- На будущее — возможно внедрение ClickHouse для high-load запросов.
- Основные таблицы:
    - `analytics_event` — все сырые события;
    - `analytics_aggregate_daily` — суточные агрегаты.

17.3 Метрики продавца

- Просмотры карточек товаров;
- Количество переходов в чат;
- CTR на кнопки “Купить”;
- Конверсии в заказы;
- Средний чек.

---

18. Партнёрская доставка

- В MVP не используется централизованный склад.
- Доставка — партнёрская (внешние службы).
- При разработке API учитывается возможность подключения через `DeliveryProvider`:

- расчёт стоимости доставки;
- статусы заказа;
- подтверждение вручения (код/фото).

---

19. Маркетинг и реклама

19.1 Ads и промо

- Магазины могут продвигать публикации (“boost post”).
- Форматы: баннеры, подборки, рекомендации.
- Рекламные кампании ограничиваются частотой показов (frequency cap = 3–5/сутки).

19.2 Модерация

- Автофильтр запрещённых слов/категорий.
- Модерация фото и описаний.
- SLA модерации — 24ч.

🧱 Часть 4 — POS-интеграция, инфраструктура, CI/CD, API и план релиза

---

20. POS (Smart-POS) интеграция

20.1 Цель

Обеспечить синхронизацию продаж офлайн-магазинов с онлайн-платформой: остатки, цены, чеки, продажи и возвраты обновляются автоматически через POS-устройство.

20.2 Устройство и модель данных

Таблица	Назначение	Ключевые поля
pos_device	Регистрирует устройство продавца	`id`, `shop_id`, `serial_number`, `status`, `last_heartbeat`
pos_sale	Фиксирует каждую продажу или возврат	`id`, `variant_id`, `qty`, `price`, `type (sale/refund)`, `timestamp`
pos_summary_daily	Ежедневная агрегация для аналитики	`shop_id`, `date`, `total_sales`, `total_refunds`, `items_sold`

20.3 Принцип работы

1. POS-терминал магазина подключается к API `/api/v1/pos/sync`.
2. Отправляет heartbeat каждые 5–10 минут.
3. При продаже — создаёт запись в `pos_sale`, обновляя `product_variant.stock_qty`.
4. Если heartbeat не получен > 10 минут → магазин считается **offline**, кнопка “Купить” скрывается.
5. Раз в сутки (в 03:00) агрегируются данные в `pos_summary_daily`.

20.4 API POS-интеграции

```
POST /api/v1/pos/register      – регистрация устройства
POST /api/v1/pos/heartbeat     – обновление статуса
POST /api/v1/pos/sale          – фиксация продажи
POST /api/v1/pos/refund        – фиксация возврата
GET  /api/v1/pos/summary       – статистика за день
```

Пример запроса sale:

```
{
  "deviceId": "UUID",
  "variantId": "UUID",
```

```
  "qty": 2,
  "price": 1500.0,
  "timestamp": "2025-11-14T15:00:00Z"
}
```

---

21. Деплой, окружения и CI/CD

21.1 Среда исполнения

- **ОС:** Linux (Ubuntu 22.04 LTS)
- **Runtime:** Docker + Docker Compose
- **База данных:** PostgreSQL 16.x (отдельный контейнер)
- **Брокер сообщений (на будущее):** RabbitMQ / Kafka (для уведомлений и POS)
- **Frontend:** Angular → билд через GitLab CI и деплой на Nginx

---

21.2 Dev / Stage / Prod окружения

Среда	Назначение	Особенности
Dev	тестирование локальных фич	localhost + docker-compose
Stage	предпродакшн, QA	общий сервер
Prod	реальная эксплуатация	сертифицированный датацентр в Бишкеке

21.3 CI/CD

- **GitLab CI/CD**
    - сборка Docker-образов;
    - автоматический деплой (по тегу версии);

    - тесты и линтеры перед билдом.
- **Docker Registry:** локальный приватный реестр.
- **Мониторинг:**
    - Prometheus + Grafana (CPU, RPS, ошибки, очереди);
    - алерты на сбои heartbeat POS, выплаты и уведомления.

21.4 Команды для ручного деплоя (MVP)

```
git pull origin main
docker compose down
docker compose up -d --build
```

База данных — отдельный контейнер или физический инстанс рядом с backend.
На этапе MVP — без Flyway (используется `ddl-auto=update`).

---

22. API и структура endpoints

22.1 Принципы REST API

- Все маршруты начинаются с `/api/v1/`.
- Ответы — JSON (UTF-8).
- Аутентификация — Bearer JWT.
- Ошибки в формате:

```
{
  "timestamp": "...",
  "status": 400,
  "error": "Bad Request",
  "message": "Invalid parameter",
  "path": "/api/v1/..."
}
```

22.2 Основные группы endpoints

Категория	Префикс	Назначение
Auth	`/auth`	Регистрация, логин, refresh
User	`/users`	Профиль, подписки, сохранённые
Merchant	`/merchant`	Управление продавцами
Shop	`/shop`	Магазины, адреса, POS
Catalog	`/catalog`	Категории, товары, варианты
Order	`/orders`	Заказы и корзина
Chat	`/chat`	Сообщения и чаты
Feed	`/feed`	Публикации и комментарии
Analytics	`/analytics`	Метрики, отчёты
Notification	`/notify`	Push, SMS, Email
Admin	`/admin`	Управление пользователями и модерацией

---

23. Безопасность и права доступа

23.1 Роли и пермишены

Роль	Разрешённые действия
User	Просмотр каталога, ленты, покупка, чат
Merchant	Управление товарами, магазинами, публикациями
Admin	Управление всеми сущностями, модерация контента
System	внутренние сервисные запросы POS, уведомления

23.2 Политика CORS и доступов

- CORS разрешён только для доменов, указанных в `app.cors.origins`.
- Поддерживаемые методы: `GET, POST, PUT, DELETE, PATCH, OPTIONS`.
- Разрешённые заголовки: `Authorization, Content-Type, Accept`.

---

24. Мониторинг и алертинг

24.1 Сбор метрик

- **Prometheus:** метрики CPU, память, RPS, error rate.
- **Grafana:** визуализация дашбордов.
- **Alertmanager:** уведомления в Telegram/email о:
    - heartbeat POS;
    - ошибках выплат;
    - задержках уведомлений;
    - 5xx на API.

24.2 Логирование

- **Logback JSON** формат для серверных логов.
- Логи сохраняются в `/var/log/fucent/` с ротацией каждые 7 дней.
- Ошибки уровня ERROR и выше отправляются в централизованный storage.

---

25. План релиза

25.1 MVP (этап 1)

- Регистрация и логин;

- Профиль пользователя;
- Каталог товаров и магазинов;
- Лента публикаций (без рекомендаций);
- Чаты и сообщения;
- POS heartbeat и фиксация продаж;
- Минимальная аналитика и отчёты;
- Web-интерфейс (Angular).

25.2 Этап 2

- Алгоритм трендов и рекомендаций;
- Онлайн-оплата и выплаты продавцам;
- Подписки PRO/Enterprise;
- Рекламные кампании (boost постов);
- Мобильное приложение (Flutter);
- Push/SMS уведомления.

25.3 Этап 3

- Полноценная доставка и склад;
- ClickHouse и расширенная аналитика;
- Гео-поиск и персональные рекомендации;
- Автоматизация маркетинга и ретаргетинг;
- Мультиязычность интерфейса (все языки).

---

26. Ключевые принципы масштабирования

1. **Модульность:** независимые модули (auth, catalog, chat, feed, pos, payout).
2. **Горизонтальное масштабирование:** контейнеризация каждого сервиса.
3. **Отделение аналитики:** ClickHouse или TimescaleDB.
4. **Хранилище медиа:** S3 или MinIO с CDN.
5. **Очереди:** Kafka/RabbitMQ для POS и уведомлений.

---

27. Резюме архитектуры

- **Backend:** Spring Boot (Java 21), PostgreSQL, JWT, Docker.
- **Frontend:** Angular SPA.
- **Mobile:** Flutter (единое приложение).
- **POS:** REST-интеграция с heartbeat и продажами.
- **CI/CD:** GitLab, Docker Registry, manual deploy.
- **Монетизация:** комиссии, подписки, реклама.
- **Отчётность:** CSV/XLSX, аналитика в админке.
- **Цель:** единая экосистема для магазинов и покупателей в Кыргызстане с масштабированием по СНГ.

🆕 ЧАСТЬ 5 — Обновления FUCENT v2.2 (ноябрь 2025)

---

28. Расширенные роли и модель управления магазинами

28.1 Список ролей системы

Роль	Назначение	Возможности
Admin	Администратор системы	Полный доступ, модерация, блокировка контента
Buyer	Покупатель / клиент	Просмотр, подписки, чат, покупки
Merchant	Владелец магазина	Создание точек (shops), управление продавцами и товарами
Seller	Продавец / менеджер магазина	Управление товарами и публикациями конкретного филиала

28.2 Иерархия и управление пользователями

- У **Merchant** может быть несколько **Shop** (филиалов).
- У каждого **Shop** — один или несколько **Seller**.
- При создании магазина (`/api/v1/merchant/register`) владелец получает роль **MERCHANT**.
- При добавлении сотрудника к филиалу (`/api/v1/shops/{id}/add-seller`) назначается роль **SELLER**, и ему открывается доступ только к данным своего филиала.
- **Admin** может повышать или понижать роли через админ-панель.

---

29. Правила публикаций и контент

29.1 Кто может публиковать

- Публикации и товары создаются **только магазинами и продавцами** (MERCHANT, SELLER).
- Роль **Buyer** не имеет доступа к созданию публикаций.

29.2 Мультимедиа публикаций

- Публикация может содержать:
  - **до 10** изображений или
  - **1 видео**.
- При просмотре публикации — **горизонтальный свайп (← →)** для переключения между изображениями внутри публикации.
- Публикации одного магазина могут быть привязаны к товарам (`post.product_id` nullable).
- Если есть привязка — у клиента отображается кнопка **«Перейти к товару»**.
- Если привязки нет — кнопка не отображается.

---

30. Отображение и поведение ленты

30.1 Общий принцип

Лента работает в формате **интерактивного скролла**, аналогично **Instagram / TikTok**:

- Каждая публикация занимает весь экран.
- Вертикальный свайп (↑ ↓) — переключает на следующую публикацию.
- Слабый свайп тоже фиксируется как переход (snap-to-next-post).

30.2 Разделы ленты

Раздел	Описание
Подписки	Посты магазинов, на которых пользователь подписан
Рекомендации	Публикации, подобранные алгоритмом (по ER, CTR, свежести, гео)
Тренды	Посты с наибольшим вовлечением за последние 24 часа

30.3 Поиск в разделе «Рекомендации»

- В разделе **Рекомендации** доступен встроенный поиск.
- Пользователь может искать магазины по:
  - нику (`username`),
  - названию (`shop.name`),
  - имени (`merchant.name`).
- Результаты показываются в виде карточек магазинов с аватаркой и кнопкой **«Перейти»**.

30.4 UI и навигация (Frontend Angular / Flutter)

Навигационное меню отображается на всех страницах, включая ленту.

Кнопка	Назначение
Лента	Главная страница с публикациями

Каталог	Список товаров и кнопка перехода на карту магазинов
Чат	Все переписки и запросы на переписку
Корзина	Добавленные товары
Профиль	Настройки, фото, имя, сохранённые ленты и публикации

> 💡 На мобильной версии (Flutter) кнопки расположены в нижней навигации; на вебе — в верхнем баре.

---

31. Поведение чата

31.1 Общая структура

- Используется единая таблица **chat** для всех типов диалогов:
  - user ↔ user,
  - user ↔ shop,
  - shop ↔ shop (для B2B в будущем).
- Сообщения хранятся в **chat_message**, привязаны к `chat_id`.

31.2 Логика общения

- Для покупателей: необходимо **запросить переписку** (pending → accepted).
- Для продавцов: чат создаётся **сразу при первом сообщении**.
- Поддерживается поиск по пользователям и магазинам:
  - по нику (`username`);

- по имени (`first_name`, `last_name`);
- по названию магазина (`shop.name`).

---

32. Обновление алгоритма трендов (детализированный подпункт)

32.1 Цель

Показывать пользователям публикации с высоким вовлечением и актуальностью.

32.2 Формула оценки

Каждой публикации присваивается рейтинг:

```
score = (likes * 0.4) + (comments * 0.3) + (saves * 0.2) + (views * 0.1) - (age_hours * 0.05)
```

32.3 Обновление

- Расчёт выполняется cron-job каждые 30 минут.
- Используется отдельная таблица `post_trend_score`.
- Публикации с наивысшим рейтингом попадают в ленту «Тренды».

32.4 Факторы для персонализации (в будущем)

- История лайков и подписок;
- Гео-позиция пользователя;
- Время активности;
- Категории интересов.

---

33. Изменения в БД

Таблица	Поле	Тип	Назначение
post	`product_id`	UUID (nullable)	Привязка публикации к товару
post_media	`post_id`, `url`, `type`	UUID, TEXT, ENUM(image, video)	Мультимедиа для публикаций
user	`role`	ENUM(admin, buyer, merchant, seller)	Расширенный список ролей
shop_seller	`shop_id`, `user_id`	UUID	Связь магазина и продавца
post_trend_score	`post_id`, `score`, `updated_at`	float, timestamp	Оценка трендов

---

34. Дополнения для фронтенда и UX

- Переход между лентой и товарами плавный, с анимацией (fade/swipe).
- При свайпе вверх/вниз видео автоматически останавливается или проигрывается.
- При открытии публикации с привязкой — отображается кнопка **«Перейти к товару»** под описанием.
- В разделе **Каталог** → **Карта** пользователи видят магазины с активным POS и кнопкой “Онлайн-покупка”.

---

35. Итого

Новые возможности версии **FUCENT v2.2:**

✅ Расширенные роли: Admin, Buyer, Merchant, Seller
✅ Полноценная иерархия магазинов и продавцов
✅ Возможность поиска магазинов в разделе «Рекомендации»
✅ Полноценный просмотр ленты как в Instagram / TikTok
✅ До 10 изображений или одно видео в публикации
✅ Привязка публикаций к товарам
✅ Алгоритм трендов и cron-обновление
✅ Обновлённые связи в БД