Client web

📱 Техническая спецификация БД: Client App

Контекст: Публичное приложение (Super App) для заказа еды и продуктов. Единая точка входа для B2C клиентов и сотрудников B2B (корпоративное питание).

📚 0. Словарь Сущностей (Entity Reference)

Этот список определяет язык общения между бэкендом, мобильной разработкой и бизнесом.

Сущность	Описание	Критичность
Client User	Физическое лицо. Может быть "Просто клиентом" или "Сотрудником". Вход по телефону.	🔥 High
User Device	Смартфон пользователя. Хранит FCM/APNS токены для Push-уведомлений.	🟡 Med
Verification Code	Временный OTP код для входа (SMS).	🔥 High
User Address	Личные адреса пользователя (Дом, Друзья). *Корп. адреса здесь не храним.*	🟡 Med
Cart (Session)	Корзина товаров. Должна переживать закрытие приложения (Persistent).	🟡 Med
Checkout Group	"Мега-заказ". Единая транзакция глазами пользователя ("Я заплатил 100с"). Группирует технические заказы.	🔥 High
Order	Технический заказ конкретному Исполнителю (Ресторану). Единица логистики.	🔥 High
Transaction	Движение *личных* средств (Пополнение, Списание с карты).	🔥 High
Analytics Event	Сырой поток действий пользователя (Clickstream).	🟢 Low (Tech)

🗺️ 1. ER-Диаграмма

```
erDiagram
    %% Identity & Auth
    CLIENT_USERS ||--o{ USER_DEVICES : owns
    CLIENT_USERS ||--o{ USER_ADDRESSES : manages
    CLIENT_USERS ||--|{ PERSONAL_WALLETS : owns
    CLIENT_USERS ||--o{ SAVED_CARDS : saves

    %% Commerce
    CLIENT_USERS ||--|| CARTS : has_active
    CARTS ||--|{ CART_ITEMS : contains

    %% Ordering Core
    CLIENT_USERS ||--|{ CHECKOUT_GROUPS : initiates
    CHECKOUT_GROUPS ||--|{ ORDERS : splits_into
    ORDERS ||--|{ ORDER_ITEMS : contains
    ORDERS ||--o{ ORDER_REVIEWS : rated_by

    %% Finance
    CLIENT_USERS ||--|{ CLIENT_TRANSACTIONS : history
    CHECKOUT_GROUPS ||--|| CLIENT_TRANSACTIONS : paid_by_personal

    %% Analytics
    CLIENT_USERS ||--o{ ANALYTICS_EVENTS : generates

    orders {
        bigint id PK
        jsonb address_snapshot "Копия адреса"
        string status
    }
    order_items {
        decimal price_snapshot "Копия цены"
    }
```

2. Детальная SQL Схема и Комментарии

👤 Модуль 1: Identity & Auth (Профиль и Безопасность)

client_users

Единый профиль пользователя. *Связь с B2B:* **Отсутствует на уровне БД.** Мы не храним `company_id`. При логине Бэкенд делает запрос в Business CRM: *"Есть ли сотрудник с телефоном X?"*. Если да — фронтенд включает B2B-интерфейс.

```
CREATE TABLE client_users (
    id BIGSERIAL PRIMARY KEY,
    phone VARCHAR(20) NOT NULL UNIQUE, -- E.164 (+992...). Главный ID.

    -- Личные данные (Заполняются по желанию [CA-USER-03])
    full_name VARCHAR(150),
    email VARCHAR(150), -- Для отправки чеков
    birth_date DATE,
    gender VARCHAR(20),
    avatar_url TEXT,

    -- Безопасность
    password_hash VARCHAR(255), -- Опционально. Если юзер хочет вход п
o паролю.
    is_phone_verified BOOLEAN DEFAULT FALSE, -- Критично для Lazy Auth

    -- Системные
    status VARCHAR(50) DEFAULT 'ACTIVE', -- ACTIVE, BLOCKED (Фрод), DEL
ETED
    language_code CHAR(2) DEFAULT 'ru', -- ru, tj, en

    -- Метаданные
    registered_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    last_login_at TIMESTAMP WITH TIME ZONE
);
```

> 💡 Комментарии архитектора:
>
> - `is_phone_verified`: Пользователь может накидать корзину как Гость. Запись в этой таблице создается только при вводе OTP кода. До этого момента корзина привязана к `guest_session_id`.
> - `email`: Не уникален (один email может быть у семьи), но желателен для фискальных чеков.

verification_codes

Временные коды для SMS-авторизации.

```
CREATE TABLE verification_codes (
    id BIGSERIAL PRIMARY KEY,
    phone VARCHAR(20) NOT NULL,
    code VARCHAR(6) NOT NULL,

    attempts INT DEFAULT 0, -- Защита от перебора
    is_used BOOLEAN DEFAULT FALSE,
    expires_at TIMESTAMP WITH TIME ZONE NOT NULL,

    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

> ⚙️ Рекомендации:
>
> - **TTL:** Установить короткое время жизни (5 минут).
> - **Rate Limit:** На уровне API запретить отправку SMS чаще 1 раза в минуту.

user_devices

Устройства пользователя. Критично для Push-уведомлений ("Курьер подъехал").

```
CREATE TABLE user_devices (
    id BIGSERIAL PRIMARY KEY,
    user_id BIGINT NOT NULL REFERENCES client_users(id) ON DELETE CASC
ADE,
```

```
    device_token TEXT NOT NULL, -- FCM / APNS Token
    platform VARCHAR(20) NOT NULL, -- 'ANDROID', 'IOS', 'WEB'
    device_id VARCHAR(100), -- Unique Hardware ID

    last_active_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    UNIQUE(user_id, device_id)
);
```

user_addresses

Личные адреса. *Логика:* Корпоративные адреса приходят по API Business CRM. Если юзер хочет добавить комментарий к офису ("Оставьте на ресепшене"), это сохраняется в самом заказе (`orders`), а не здесь.

```
CREATE TABLE user_addresses (
    id BIGSERIAL PRIMARY KEY,
    user_id BIGINT NOT NULL REFERENCES client_users(id) ON DELETE CASC
ADE,

    name VARCHAR(100), -- "Дом", "Родители"
    address_line TEXT NOT NULL, -- Геокодированный адрес ("ул. Рудаки 1
0")
    geo_point POINT NOT NULL, -- (lat, long) для проверки зон доставки [CA-
FOOD-01]

    -- Детали
    apartment VARCHAR(20),
    floor VARCHAR(20),
    entrance VARCHAR(20),
    intercom VARCHAR(20),
    comment_for_courier TEXT,

    is_default BOOLEAN DEFAULT FALSE,
    deleted_at TIMESTAMP WITH TIME ZONE, -- Soft Delete
```

```
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

> ⚙️ Рекомендации:
>
> - **Validation:** При вставке проверять через Geo-Service, попадает ли `geo_point` в зону обслуживания хотя бы одного партнера.

🛒 Модуль 2: Cart (Корзина)

carts

Персистентная корзина. *Логика:* Храним корзину вечно, пока не купит или не очистит. Позволяет начать на вебе, закончить в приложении.

```
CREATE TABLE carts (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id BIGINT REFERENCES client_users(id), -- Nullable для Гостей
    guest_session_id VARCHAR(100), -- Fingerprint для Гостей

    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

    CONSTRAINT uq_cart_user UNIQUE (user_id) -- У юзера только 1 активна
я корзина
);
```

cart_items

Товары. *Связь:* Ссылаемся на `merchant_id` и `menu_item_id`, которые живут в **Merchant Admin** (через API Gateway или Shared DB в MVP).

```
CREATE TABLE cart_items (
    id BIGSERIAL PRIMARY KEY,
    cart_id UUID NOT NULL REFERENCES carts(id) ON DELETE CASCADE,

    -- Внешние ссылки (Merchant Admin)
    merchant_id BIGINT NOT NULL, -- ID Ресторана/Магазина
```

```
    menu_item_id BIGINT NOT NULL, -- ID Товара

    quantity INT NOT NULL CHECK (quantity > 0),

    -- Выбор пользователя
    selected_modifiers JSONB, -- [{"id": 10, "name": "Сыр", "price": 5}]
    comment VARCHAR(255), -- "Без лука"

    -- Тип сервиса (Задел на будущее Ритейла)
    service_type VARCHAR(20) DEFAULT 'RESTAURANT', -- 'RESTAURANT', 'G
ROCERY'

    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

> 💡 Комментарии архитектора:
>
> - `selected_modifiers`: Храним JSON, так как структура модификаторов может быть сложной (деревья, группы). Важно хранить и ID, и цену опции на момент добавления в корзину (для предварительного расчета суммы на клиенте).

📦 Модуль 3: Ordering (Ядро Заказов)

`checkout_groups`

"Мега-заказ". Группировка транзакции. *Пример:* Юзер заказал из Ресторана А и Магазина Б. Это **один** `checkout_group`, но **два** `orders`. *Зачем:* Чтобы списать деньги одной транзакцией и показать в истории "Заказ №123".

```
CREATE TABLE checkout_groups (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(), -- Публичный ID зака
за
    user_id BIGINT NOT NULL REFERENCES client_users(id),

    -- Финансы (Общие)
    total_amount DECIMAL(15, 2) NOT NULL, -- Сумма всех под-заказов
```

```
    -- Источники оплаты (Split Payment)
    payment_method VARCHAR(50) NOT NULL, -- 'PERSONAL', 'CORP', 'SPLIT'
    paid_by_personal DECIMAL(15, 2) DEFAULT 0.00,
    paid_by_corp DECIMAL(15, 2) DEFAULT 0.00,

    -- Промо
    promo_code_applied VARCHAR(50),
    discount_amount DECIMAL(15, 2) DEFAULT 0.00,

    -- Глобальный статус
    status VARCHAR(50) DEFAULT 'PENDING_PAYMENT', -- PENDING, PAID, FA
ILED, PARTIALLY_REFUNDED

    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

orders

Единица логистики. Уходит конкретному Мерчанту.

```
CREATE TABLE orders (
    id BIGSERIAL PRIMARY KEY,
    checkout_group_id UUID NOT NULL REFERENCES checkout_groups(id),

    -- Исполнитель
    merchant_id BIGINT NOT NULL, -- Внешний ID (Ресторана)
    service_type VARCHAR(20) NOT NULL, -- 'RESTAURANT' (сейчас), 'GROC
ERY' (потом)

    -- Статусы
    status VARCHAR(50) DEFAULT 'NEW', -- NEW → COOKING → ON_WAY →
DELIVERED | CANCELED
    cancellation_reason TEXT, -- Если отменен

    -- 📷 СНАПШОТЫ (Защита истории)
```

```
    -- Мы копируем адрес целиком. Даже если это B2B адрес, пользовател
ь мог добавить комментарий.
    delivery_address_json JSONB NOT NULL,
    -- { "lat": ..., "lon": ..., "text": "...", "comment": "Код 123" }

    -- Логистика
    courier_id BIGINT, -- ID курьера (из Yalla Logistics)
    delivery_cost DECIMAL(10, 2) DEFAULT 0.00,
    eta_minutes INT, -- Расчетное время прибытия

    -- Даты
    schedule_time TIMESTAMP WITH TIME ZONE, -- Если предзаказ
    delivered_at TIMESTAMP WITH TIME ZONE,

    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

> 💡 Комментарии архитектора:
>
> - `merchant_id`: При создании заказа мы должны проверить, открыт ли ресторан (API Call в Merchant Admin / Shifts).
> - `status`: Статусы синхронизируются с Merchant App (через вебхуки или общую шину событий).

order_items

Состав заказа (Immutable History).

```
CREATE TABLE order_items (
    id BIGSERIAL PRIMARY KEY,
    order_id BIGINT NOT NULL REFERENCES orders(id),

    menu_item_id BIGINT NOT NULL,

    -- 📸 СНАПШОТЫ (Защита от изменения цен)
```

```
    name_snapshot VARCHAR(255) NOT NULL, -- "Бургер Чиз" (даже если пе
реименуют)
    price_snapshot DECIMAL(10, 2) NOT NULL, -- 25.00 (даже если станет 30.
00)

    quantity INT NOT NULL,
    modifiers_json JSONB, -- Выбранные опции с ценами: [{"name": "Сыр",
"price": 5}]

    total_price DECIMAL(10, 2) NOT NULL -- (price + mods) * qty
);
```

order_reviews

Оценки и отзывы [CA-ORDER-04].

```
CREATE TABLE order_reviews (
    id BIGSERIAL PRIMARY KEY,
    order_id BIGINT NOT NULL REFERENCES orders(id),
    user_id BIGINT NOT NULL REFERENCES client_users(id),

    rating INT NOT NULL CHECK (rating BETWEEN 1 AND 5),
    tags VARCHAR(50)[], -- ['TASTY', 'FAST_DELIVERY']
    comment TEXT,

    is_ticket_created BOOLEAN DEFAULT FALSE, -- Если оценка < 4, создаем
тикет в Support

    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

> ⚙️ Рекомендации:
>
> - **Integration:** При вставке записи с `rating <= 3`, код должен отправлять событие в Yalla CRM для создания Инцидента (Incident).

💳 Модуль 4: Finance (Личный кошелек)

personal_wallets

Кошелек B2C.

```
CREATE TABLE personal_wallets (
    user_id BIGINT PRIMARY KEY REFERENCES client_users(id),
    balance DECIMAL(15, 2) DEFAULT 0.00 CHECK (balance >= 0),
    currency_code CHAR(3) DEFAULT 'TJS',
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

client_transactions

История личных средств. *Логика:* Здесь только списания с личного баланса. Списания корпоративных средств хранятся в `Business CRM → company_transactions`.

```
CREATE TABLE client_transactions (
    id BIGSERIAL PRIMARY KEY,
    user_id BIGINT NOT NULL REFERENCES client_users(id),

    type VARCHAR(50) NOT NULL, -- 'DEPOSIT_CARD', 'ORDER_PAYMENT', 'R
EFUND'
    amount DECIMAL(15, 2) NOT NULL, -- +/-

    -- Связь
    checkout_group_id UUID REFERENCES checkout_groups(id),

    -- Платежный шлюз
    payment_provider VARCHAR(50), -- 'ALIF', 'DC', 'CORTI_MILLI'
    external_txn_id VARCHAR(100), -- ID транзакции в банке

    balance_after DECIMAL(15, 2) NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

📊 Модуль 5: Analytics (Big Data)

Реализация требования "Трекать поведение с первого дня".

analytics_events

Clickstream. Лог всех действий (от клика по баннеру до покупки).

```
CREATE TABLE analytics_events (
    id BIGSERIAL, -- Используем для совместимости, но лучше TimescaleDB / ClickHouse
    event_time TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

    -- Контекст
    user_id BIGINT, -- Nullable (Гость)
    session_id VARCHAR(100), -- Fingerprint / Cookie ID
    device_type VARCHAR(50), -- 'IOS', 'ANDROID', 'WEB'
    app_version VARCHAR(20),

    -- Событие
    event_name VARCHAR(100) NOT NULL, -- 'view_restaurant', 'add_to_cart', 'checkout_start'

    -- Объект
    entity_type VARCHAR(50), -- 'RESTAURANT', 'DISH', 'SCREEN'
    entity_id VARCHAR(50),

    -- Детали (JSON)
    properties JSONB -- {"screen": "home", "search_query": "plov", "referrer": "instagram"}
) PARTITION BY RANGE (event_time);

-- Создание партиций (Автоматизировать через cron!)
CREATE TABLE analytics_events_2025_12 PARTITION OF analytics_events
    FOR VALUES FROM ('2025-12-01') TO ('2026-01-01');
```

> ⚙️ Рекомендации:

> - **Insert Strategy:** Вставка должна быть асинхронной (Fire & Forget). Нельзя, чтобы тормоза аналитики замедляли заказ еды.
> - **No Foreign Keys:** Не связывать эту таблицу ключами с `client_users` для скорости вставки.

🔗 3. Связи с внешними продуктами

Так как мы строим Экосистему, базы данных логически разделены, но функционально связаны через API.

1. Client App -> Business CRM (B2B)

- **Цель:** Использовать корп. лимит для оплаты.
- **Метод:** API-First (Синхронные вызовы).
- **Поток:**
  1. При логине: `GET /b2b/check-employee?phone=...` → Получаем `is_active`, `daily_limit` и `address`.
  2. При заказе (Сплит):
     - Client App блокирует личные средства.
     - Client App шлет запрос `POST /b2b/charge { amount: 40, user_phone: ... }`.
     - Business CRM списывает деньги и возвращает `OK`.
     - Client App завершает заказ (`PAID`).

2. Client App -> Merchant Admin (Рестораны)

- **Цель:** Получить меню, проверить наличие, отправить заказ.
- **Метод:** API Gateway.
- **Поток:**
  - `cart_items` хранит `menu_item_id`.
  - При отображении корзины, мы делаем запрос `GET /merchant/menu-check`, чтобы проверить актуальность цены и стоп-листы.

3. Client App -> Yalla CRM (Support)

- **Цель:** Жалобы и рейтинги.
- **Метод:** Очередь событий (RabbitMQ / Kafka).
- **Поток:**
  - Пользователь ставит 1 звезду.
  - Client App кидает событие `ORDER_RATED_LOW`.
  - Yalla CRM слушает очередь и создает Тикет для оператора.

⚠️ 4. Developer Checklist

🔴 Critical (Нельзя игнорировать)

1. **Snapshots (Снапшоты):** Всегда копируйте `price` и `name` блюда из Меню в `order_items`. Меню меняется, история заказов — нет.
2. **Idempotency (Идемпотентность):** При оплате картой (`client_transactions`), генерируйте `idempotency_key` на клиенте. Если сеть моргнет, мы не должны списать деньги дважды.
3. **Phone Normalization:** Все телефоны хранить строго в формате E.164 (без пробелов, скобок, начиная с +992). Иначе матчинг с B2B сотрудниками не сработает.

🟡 Warning (Логика)

1. **Guest Cart:** Корзина привязана к `guest_session_id`. При логине (Verify Phone), нужно сделать `UPDATE carts SET user_id = X` для старой гостевой корзины (Merge Carts).
2. **Split Rollback:** Если списание B2B прошло успешно, а последующее списание Карты упало — обязательно дернуть API отмены B2B транзакции (Compensating Transaction / Refund).

🔵 Suggestions

1. **JSONB Indexing:** Для `analytics_events` создать GIN индекс на поле `properties`, чтобы аналитики могли фильтровать по JSON-полям.

2. **Cache:** Кэшировать B2B статус сотрудника в токене или Redis, чтобы не дергать Business API при каждом переходе на экран "Профиль".