**Техническое задание  
на разработку веб‑приложения  
«Приёмка объектов недвижимости»**

**Версия:** 1.0  
**Язык интерфейса:** RU (на первом этапе)  
**Модель развёртывания:** локальный сервер/ВМ (с перспективой Azure)  
**Целевая платформа клиента:** десктоп/планшет (в т.ч. iPad), мобильные браузеры  
**Пользователи на первом этапе:** 1 специалист (без мульти-тенантности)

**1) Цель и ключевая ценность**

Собрать единое веб-приложение, которое позволяет:

* быстро находить и отбирать **дефекты** строительства/ремонта по фильтрам (страна → блок → тип, локация, критичность, текстовый поиск);
* отмечать нужные позиции чекбоксами, дополнять их **измерениями**, привязками к **локациям** и **нормативам**;
* формировать **отчёты** с обязательными **нормативными ссылками/выдержками** и **фото**, экспортировать в **PDF/DOCX/XLSX**;
* хранить и обрабатывать **медиафайлы** (галерея: теги, подписи, сортировка, пакетное удаление) через Supabase Storage c Signed URLs.

**Отдельные требования, подтверждённые пользователем:**

* Все нормы (РК/РФ и др.) — в **одной базе**, в отчёте обязательны ссылки/цитаты.
* **DOCX**-экспорт: для каждого дефекта вставлять **2–3 фото**, если доступны.
* Галерея: **теги** и **поиск по тегам** («трещина», «окно», «радиатор» и т.п.).
* **PWA/офлайн** + **Background Sync** для POST-операций с визуальными статусами.

**2) Роли и доступ**

* **Гость (anon):** только чтение (через RLS).
* **Авторизованный (authenticated):** чтение + добавление измерений, изменение атрибутов дефекта (критичность, локация), добавление **новых локаций**, **динамическое пополнение** справочников **блоков/типов**.
* **Админ:** техподдержка/миграции (вне UI).
* На первом этапе: **один пользователь**, без менеджеров/ревьюеров и без «владельца записи».

**3) Технологический стек**

* **Frontend/Backend:** Next.js 15 (App Router), TypeScript, Tailwind CSS.
* **БД/аутентификация:** Supabase (Postgres + Auth + RLS).
* **Экспорт:**
  + **PDF:** серверный рендер через Playwright/Puppeteer (для больших выборок).
  + **DOCX:** пакет **docx** (серверный); встраивание 2–3 фото на дефект.
  + **XLSX:** пакет **xlsx** (сервер).
* **Состояние выбора:** localStorage.
* **PWA:** манифест + сервис-воркер + IndexedDB для справочников и кэша GET.
* **Фулл-текст поиск:** PostgreSQL **pg\_trgm** (+ ранжирование).

**4) Переменные окружения (.env.local)**

NEXT\_PUBLIC\_SUPABASE\_URL=… # Публичный URL

NEXT\_PUBLIC\_SUPABASE\_ANON\_KEY=… # Только anon key для браузера

# ВАЖНО: service\_role ключ хранить ТОЛЬКО на сервере для миграций/сидов.

*Примечание:* пользователь предоставил доступы отдельно; использовать **service\_role** исключительно для серверных миграций/импорта, затем ключ **ротировать**.

**5) Модель данных (Supabase / Postgres)**

**5.1 Таблицы (основные)**

* **countries**(id uuid PK, name text unique, is\_enabled bool)
* **blocks**(id uuid PK, name text unique)
* **defect\_types**(id uuid PK, block\_id → blocks.id, name text, **unique(block\_id,name)**)
* **norms**(id uuid PK, country\_id → countries.id, code text, point text, source\_url text, full\_text text, **unique(country\_id,code,point)**)
* **severity\_levels**(id bigserial PK, name text unique, sort\_order int) — 4 уровня (Критично/Существенно/Незначительно/Информативно)
* **location\_categories**(id uuid PK, name text unique) — «Квартира», «Частный дом/Коттедж», «Офис», …
* **locations**(id uuid PK, category\_id → location\_categories.id, name text, **unique(category\_id,name)**)
* **defects**(id uuid PK, country\_id, block\_id, type\_id, problem text, description text, solution text, norm\_id uuid, severity\_id bigint, location\_id uuid, created\_at timestamptz default now())
  + **уникальный индекс (дедуп):**  
    (country\_id, block\_id, type\_id, problem, coalesce(norm\_id,'00000000-0000-0000-0000-000000000000'))
* **defect\_measurements**(id bigserial PK, defect\_id uuid → defects.id, name text, value text, unit text, created\_at timestamptz default now(), created\_by uuid)
* **media\_files**(id uuid PK, project\_id uuid, defect\_id uuid?, location\_id uuid?, path text, preview\_path text?, exif jsonb?, tags text[], created\_at timestamptz default now())
* **import\_logs**(id bigserial PK, source text, rows int, started\_at timestamptz, finished\_at timestamptz, details jsonb)

**5.2 Представления (для UI)**

* **defects\_full\_v2** — развёрнутая запись дефекта: имена страны/блока/типа/локаций, норма (код/пункт/ссылка/выдержка), критичность (имя, порядок), удобный norm\_display = code + point.
* **defects\_view\_v2** — компактная выборка для списка/таблицы (с norm\_display и severity\_order).

**5.3 Политики RLS**

Включены на всех таблицах.

* SELECT: доступны **anon** и **authenticated** (read-only).
* INSERT/UPDATE/DELETE:
  + **defect\_measurements** — разрешить authenticated все операции;
  + **defects** — UPDATE (критичность, локация, problem/description/solution);
  + **locations** — INSERT (пользователь может добавлять свои локации);
  + **media\_files** — INSERT (добавление медиа/метаданных).

**5.4 Индексы и производительность**

* Индексы на всех FK.
* defects\_dedupe\_idx (см. выше).
* Для фулл-текста: gin + pg\_trgm на (problem, description, solution, norm\_display).
* Бюджет ответа списка: **≤300–600 мс** при pageSize=50 и типовых фильтрах.

**6) UI/UX**

**6.1 Главная «/»**

**Фильтры (верхняя панель):**

* **Страна** (из countries, is\_enabled=true),
* **Блок** (из blocks),
* **Тип** (из defect\_types, зависит от блока),
* **Категория локации** (из location\_categories),
* **Локация** (из locations, зависит от категории),
* **Критичность** (из severity\_levels),
* **Поиск** (фри-текст по problem/description/solution/norm\_display; при включённом pg\_trgm — ранжирование).

**Таблица дефектов (источник defects\_view\_v2):**  
колонки: checkbox, ID, Страна, Блок, Тип, Проблема, Описание, Решение, Норма (display), Критичность, Категория локации, Локация.

* Сортировка по всем колонкам.
* Скрытие пустых/нерелевантных значений при активных фильтрах.
* Пагинация (серверная): **50/100/200**, стрелки, «в начало/в конец», «go to page N».
* Чекбоксы выбора строк (сохранение в localStorage по id).
* Кнопки: «Выбрать страницу / Снять выбор», «Выбрать всё (по текущим фильтрам)».

**Панель действий:**

* Экспорт **выбранных**: PDF/DOCX/XLSX (по ids из localStorage).
* Экспорт **всех по фильтрам**: PDF/DOCX/XLSX (сервер формирует по фильтрам, без ограничений страниц).
* «Сброс фильтров», «Очистить выбор».

**6.2 Детальная «/defect/[id]»**

* **Карточка** (read-only для anon, editable для authenticated):  
  страна (R/O), блок (R/O), тип (R/O), **problem/description/solution** (editable), **критичность** (select), **категория/локация** (select → зависимый).  
  Норма — **кликабельная** (открывает source\_url).  
  Кнопка «Сохранить» (PATCH).
* **Измерения**: табличный список (name, value, unit, created\_at) + форма добавления; правка/удаление (для автора/authenticated).
* **Добавить локацию**: поле + выбор категории → insert в locations, **сразу** появляется в select.

**6.3 Галерея медиа**

* Источник: media\_files.
* Фичи: **теги** (tags[]), **подписи**, **сортировка** по дате/имени, **поиск по тегам/имени**, **мультивыбор** и **пакетное удаление**.
* Превью (800 px) и EXIF (дата съемки, камера и др.) — см. §8.
* Привязки: project\_id/location\_id/defect\_id (опционально).

**6.4 Авторизация**

* Вход: **Email magic-link/OTP**. OAuth/SSO — не требуется на первом этапе.

**6.5 Доступность и адаптивность**

* Адаптив: десктоп/планшет/мобайл.
* A11y: aria-label, фокус-стили, управление с клавиатуры.

**7) Поведение фильтров и зависимостей**

* Выбор **блока** ограничивает список **типов**.
* Выбор **категории локации** ограничивает **локации**.
* Выбор **страны** не ограничивает типы/локации, но фильтрует строки результатов.
* **Повторяемость** дефектов по помещениям — допустима; вывод строится по стране/блоку/типу с учётом выбранных локаций.

**8) Экспорт отчётов**

**8.1 Общие правила**

* Серверные API-роуты Next.js формируют **PDF/DOCX/XLSX**.
* **Сортировка экспорта**: severity\_order, country, block, type.
* **Лимит защиты**: максимум 10 000 строк за один экспорт; для больших наборов — ZIP-архив со связками (PDF + XLSX + ссылки на медиа).

**8.2 DOCX (обязательно с фото)**

* Шаблон: логотип, титул, оглавление, колонтитулы, **QR-подпись** (хэш/ссылка на карточку отчёта), фирменные цвета.
* По каждой записи-дефекту:
  + заголовок (локация → категория → блок/тип → проблема),
  + таблица полей: описание, решение, критичность, **норма (код/пункт/выдержка + source\_url)**,
  + **встраивание 2–3 фото** при наличии: "media/{project\_id}/{location}/{defect\_id}/photos/\*", с подписями «Фото 1: …».
* Перенос больших таблиц по страницам, повтор шапок.

**8.3 PDF**

* Генерация из HTML сервером (Playwright/Puppeteer) c предсказуемой версткой (нумерация, колонтитулы, оглавление).

**8.4 XLSX**

* Лист data с колонками списка + отдельные листы по группировкам (опционально).

**9) API (Next.js App Router, v1)**

Базовый префикс: /api/v1

**9.1 Дефекты/список**

GET /defects  
**query:**

* country, block, type, severity, locCat, loc, q,
* sort=col:asc|desc, page, pageSize (50/100/200)  
  **resp:**

{

"rows": [ /\* из defects\_view\_v2 \*/ ],

"total": 1234,

"page": 1,

"pageSize": 50

}

**9.2 Дефект/деталь**

GET /defects/{id} → из defects\_full\_v2.

PATCH /defects/{id}  
**body (partial):**

{

"problem": "...",

"description": "...",

"solution": "...",

"severity\_id": 2,

"location\_id": "uuid"

}

**9.3 Измерения**

POST /defects/{id}/measurements

{ "name": "Зазор", "value": "3", "unit": "мм" }

PATCH /measurements/{id} / DELETE /measurements/{id}

**9.4 Локации**

POST /locations

{ "category\_name": "Квартира", "name": "Гардеробная" }

**9.5 Экспорт**

* **Выбранные:** GET /export?format=(pdf|docx|xlsx)&ids=uuid,uuid,…
* **По фильтрам:** GET /export?format=…&all=true&country=…&block=…&…  
  **Поведение:** сервер повторяет выборку как /defects, но без пагинации; рендерит файл; ответ application/octet-stream с Content-Disposition: attachment.

**9.6 Медиа/поиск**

GET /media/search  
**query:** q (имя), tag, date\_from, date\_to, project\_id, defect\_id, location\_id.  
**resp:** массив объектов media\_files + **Signed URLs** (через серверный прокси-роут).

**9.7 Вебхуки (опционально)**

* POST /webhooks/report-ready — отчёт сформирован (внутреннее уведомление).
* POST /webhooks/new-measurement, …/new-location — события добавления.

**Ошибки API:** JSON { "error": { "code": "…", "message": "…" } }, корректные HTTP-коды.

**10) Медиа-хранилище (Supabase Storage)**

**10.1 Бакет и структура**

* Бакет: media
* Папки: media/{project\_id}/{location}/{defect\_id}/{photos|videos|docs}/filename.ext

**10.2 Signed URLs + прокси**

* Доступ медиа — только по **Signed URL** (TTL настраиваемый), выдаёт **серверный прокси**; прямые пути не раскрываются.

**10.3 Превью/EXIF (пояснение)**

* **Превью** — автоматическая генерация уменьшенных изображений (≈800 px по длинной стороне) для быстрой галереи; хранится путь preview\_path.
* **EXIF** — извлечение метаданных (дата/время съемки, камера, гео при наличии) в поле exif (jsonb). Это не «обязаловка», но сильно ускоряет поиск/сортировку и повышает информативность фото-вставок в отчёт.
* **Теги** — tags[] (например: ["трещина","окно","радиатор"]), участвуют в поиске.
* **Ограничения размеров:** «жёсткого» максимума **не требуется** (по согласованию). Рекомендовано предупреждение при загрузке >25 МБ (UX).

**11) PWA и офлайн-режим**

* **Манифест** + **сервис-воркер**.
* Кэширование GET-запросов и справочников в **IndexedDB** (countries/blocks/types/norms/severity/locations).
* **Background Sync**: очередь POST-операций (измерения, новые локации, медиа-метаданные).
* **Статусы** на клиенте: **«в очереди» → «отправлено»**; повторные попытки при восстановлении сети.
* Обработка конфликтов: при неуспехе PATCH — явно показать тост «Не удалось сохранить, попробуйте снова/перезагрузите страницу».

**12) Нефункциональные требования**

* **Производительность:** ответ списка ≤300–600 мс на 50 строк при типовых фильтрах; серверный экспорт больших наборов.
* **Надёжность:** централизованный обработчик ошибок API; ретраи в экспорте; внутренние уведомления о готовности отчёта.
* **Логи/мониторинг:** Sentry (frontend/backend), Logflare/pg\_audit; трассировки длительных запросов.
* **Бэкапы:** собственные **snapshots** БД **после каждого подтверждённого проекта** (+ возможность «ручного» snapshot).
* **Безопасность:**
  + только публичные ключи в браузере;
  + вся запись — только для authenticated через RLS;
  + ограничение экспорта для анонима (без «выбрать всё»);
  + rate-limit в API можно опустить на первом этапе (1 пользователь), но оставить готовность включить.

**13) Импорт/сидирование данных**

* Исходные мастер-файлы: CSV/Excel (страны, блоки/типы, локации, нормы, дефекты).
* Порядок импорта:
  1. countries, blocks, severity\_levels, location\_categories
  2. defect\_types (map: block\_name → blocks.name)
  3. locations (map: category\_name → location\_categories.name)
  4. norms (map: country\_name)
  5. defects (map по именам и norm code+point, дедуп по индексу)
  6. defect\_measurements (по defect\_id после вставки дефектов)
* Объём на запуск: **≈500 дефектов/страна**, возможный рост не более ×2.
* Подготовлен **полный server\_seed\_full.sql**: создаёт схему/вьюхи/RLS и функции upsert\_\* (страна, блок, тип, локация, норма, дефект).
* (Опционально) \copy-скрипт или Node-утилита для пакетного вызова upsert\_defect(...) по CSV.

**14) Деплой/инфраструктура**

* **Локальное развёртывание** (Docker/ВМ/ПК/ноутбук).
* **Node.js:** актуальная **LTS** (рекомендуем 20.x).
* **Домен/HTTPS/CDN:** не требуются на первом этапе.
* **Azure** — как перспектива, конфигурация совместима.
* **Supabase Storage**: настроить CORS и политику Signed URLs.

**15) Лицензирование и политика данных**

* Лицензионные планы/партнёрский доступ — **не требуются** на первом этапе.
* Политика данных:
  + регион хранения — по региону Supabase проекта;
  + право на удаление/экспорт пользовательских данных;
  + отчёты/медиа хранятся в Supabase (Retention — по усмотрению пользователя).

**16) Приёмочные тесты (Acceptance)**

**Данные/импорт**

* Импорт миграций и CSV → записи видны во defects\_full\_v2/defects\_view\_v2.
* Дедупликация: повторная загрузка не создаёт дублей.

**Фильтры/список**

* Фильтры по стране/блоку/типу работают; тип зависит от блока;
* Сортировка по всем столбцам; пустые значения «не мешают».
* Пагинация 50/100/200; «выбрать страницу/всё» корректно управляет выбором.

**Деталь дефекта**

* Можно присвоить критичность, категорию/локацию, править problem/description/solution.
* Измерения: добавить/править/удалить; created\_at фиксируется.
* «Добавить локацию» — запись появляется в селекте без перезагрузки.

**Экспорт**

* «Выбранные» → файл содержит только отмеченные строки;
* «По фильтрам» → файл содержит **все** строки, а не только текущую страницу;
* **DOCX** содержит 2–3 фото под дефект при наличии; нормы отображаются (код/пункт/выдержка/ссылка).

**Галерея**

* Теги добавляются и учитываются в поиске; превью генерируются; пакетное удаление работает.

**PWA/офлайн**

* Установка на устройство;
* Кэш справочников в IndexedDB;
* Очередь Background Sync: статусы «в очереди/отправлено», авто-ретраи.

**Безопасность/RLS**

* anon — только чтение; authenticated — запись согласно политике;
* прямой доступ к Storage без Signed URL невозможен.

**Смоук-тесты и E2E**

* Смоук PWA/офлайн;
* E2E (Playwright): сценарии списка, фильтров, экспорта, детальной карточки, измерений, медиа.

**17) Будущие улучшения (опционально)**

* Сохранённые пресеты фильтров (персональные).
* История правок/комментарии (audit trail).
* Импорт CSV из UI для админа.
* Группировки/кластеризация в отчётах (по блокам/типам/локациям).
* Мультиязычность (EN).
* Мульти-тенантность/роли «менеджер/ревьюер» по проектам.

**18) Приложения**

* **Список нормативов** и примеры формулировок — закладываются в norms с code, point, source\_url, full\_text.
* **DOCX-шаблон**: титул с логотипом, оглавление, колонтитулы, QR-подпись; разделы «Данные об объекте», «Инструменты», «Нормативно-правовая база», «Фотоотчёт» и др.
* **Медиа-структура**: media/{project\_id}/{location}/{defect\_id}/{photos|videos|docs}/…

Ниже привожу полный **server\_seed\_full.sql** для Supabase/Postgres — в том виде, чтобы можно было загрузить его в SQL Editor Supabase (под **service\_role**) или выполнить через psql.

Файл создаёт:

* все **таблицы** (countries, blocks, defect\_types, norms, severity\_levels, location\_categories, locations, defects, defect\_measurements, media\_files, import\_logs),
* все **вьюхи** (defects\_full\_v2, defects\_view\_v2),
* включает **RLS** + политики чтения/записи,
* вставляет сиды критичности,
* добавляет **функции upsert\_**\* для словарей, норм и дефектов.

-- server\_seed\_full.sql

-- Полный сидинг схемы и функций для Supabase/Postgres

-- ВАЖНО: запускать только под service\_role, в браузере не выполнять!

begin;

-- Extensions -----------------------------------------------------------------

create extension if not exists "uuid-ossp";

create extension if not exists pgcrypto;

-- Справочники ---------------------------------------------------------------

create table if not exists countries(

id uuid primary key default gen\_random\_uuid(),

name text unique not null,

is\_enabled boolean not null default true

);

create table if not exists blocks(

id uuid primary key default gen\_random\_uuid(),

name text unique not null

);

create table if not exists defect\_types(

id uuid primary key default gen\_random\_uuid(),

block\_id uuid not null references blocks(id) on delete cascade,

name text not null,

unique(block\_id, name)

);

create table if not exists norms(

id uuid primary key default gen\_random\_uuid(),

country\_id uuid not null references countries(id) on delete cascade,

code text not null,

point text not null,

source\_url text,

full\_text text,

unique(country\_id, code, point)

);

create table if not exists severity\_levels(

id bigserial primary key,

name text unique not null,

sort\_order int not null

);

create table if not exists location\_categories(

id uuid primary key default gen\_random\_uuid(),

name text unique not null

);

create table if not exists locations(

id uuid primary key default gen\_random\_uuid(),

category\_id uuid not null references location\_categories(id) on delete cascade,

name text not null,

unique(category\_id, name)

);

-- Дефекты -------------------------------------------------------------------

create table if not exists defects(

id uuid primary key default gen\_random\_uuid(),

country\_id uuid not null references countries(id),

block\_id uuid not null references blocks(id),

type\_id uuid not null references defect\_types(id),

problem text not null,

description text,

solution text,

norm\_id uuid references norms(id),

severity\_id bigint references severity\_levels(id),

location\_id uuid references locations(id),

created\_at timestamptz not null default now()

);

create unique index if not exists defects\_dedupe\_idx

on defects(country\_id, block\_id, type\_id, problem, coalesce(norm\_id,'00000000-0000-0000-0000-000000000000'));

-- Измерения -----------------------------------------------------------------

create table if not exists defect\_measurements(

id bigserial primary key,

defect\_id uuid not null references defects(id) on delete cascade,

name text not null,

value text,

unit text,

created\_at timestamptz not null default now(),

created\_by uuid

);

-- Медиа ---------------------------------------------------------------------

create table if not exists media\_files(

id uuid primary key default gen\_random\_uuid(),

project\_id uuid,

defect\_id uuid references defects(id) on delete set null,

location\_id uuid references locations(id) on delete set null,

path text not null,

preview\_path text,

exif jsonb,

tags text[],

created\_at timestamptz not null default now()

);

-- Логи импорта ---------------------------------------------------------------

create table if not exists import\_logs(

id bigserial primary key,

source text,

rows int,

started\_at timestamptz default now(),

finished\_at timestamptz,

details jsonb

);

-- Вьюхи ---------------------------------------------------------------------

create or replace view defects\_full\_v2 as

select d.\*,

c.name as country\_name,

b.name as block\_name,

t.name as type\_name,

l.name as location\_name,

lc.name as location\_category\_name,

n.code as norm\_code,

n.point as norm\_point,

n.source\_url as norm\_source\_url,

n.full\_text as norm\_full\_text,

s.name as severity\_name,

s.sort\_order as severity\_order,

(coalesce(n.code,'') || ' ' || coalesce(n.point,'')) as norm\_display

from defects d

join countries c on c.id = d.country\_id

join blocks b on b.id = d.block\_id

join defect\_types t on t.id = d.type\_id

left join locations l on l.id = d.location\_id

left join location\_categories lc on lc.id = l.category\_id

left join norms n on n.id = d.norm\_id

left join severity\_levels s on s.id = d.severity\_id;

create or replace view defects\_view\_v2 as

select id, country\_id, block\_id, type\_id, problem, description, solution,

norm\_id, severity\_id, location\_id,

country\_name, block\_name, type\_name,

location\_category\_name, location\_name,

norm\_display, severity\_order

from defects\_full\_v2;

-- RLS -----------------------------------------------------------------------

alter table countries enable row level security;

alter table blocks enable row level security;

alter table defect\_types enable row level security;

alter table norms enable row level security;

alter table severity\_levels enable row level security;

alter table location\_categories enable row level security;

alter table locations enable row level security;

alter table defects enable row level security;

alter table defect\_measurements enable row level security;

alter table media\_files enable row level security;

-- Политики select для anon/authenticated

create policy if not exists "read\_all\_countries" on countries for select using (true);

create policy if not exists "read\_all\_blocks" on blocks for select using (true);

create policy if not exists "read\_all\_types" on defect\_types for select using (true);

create policy if not exists "read\_all\_norms" on norms for select using (true);

create policy if not exists "read\_all\_severity" on severity\_levels for select using (true);

create policy if not exists "read\_all\_loc\_cat" on location\_categories for select using (true);

create policy if not exists "read\_all\_locations" on locations for select using (true);

create policy if not exists "read\_all\_defects" on defects for select using (true);

create policy if not exists "read\_all\_measurements" on defect\_measurements for select using (true);

create policy if not exists "read\_all\_media" on media\_files for select using (true);

-- Политики записи для authenticated

create policy if not exists "auth\_upd\_defects" on defects for update to authenticated using (true);

create policy if not exists "auth\_insupd\_measurements" on defect\_measurements for all to authenticated using (true) with check (true);

create policy if not exists "auth\_ins\_locations" on locations for insert to authenticated with check (true);

create policy if not exists "auth\_ins\_media" on media\_files for insert to authenticated with check (true);

-- Seed критичности -----------------------------------------------------------

insert into severity\_levels(name, sort\_order) values

('Критично',1),

('Существенно',2),

('Незначительно',3),

('Информативно',4)

on conflict (name) do update set sort\_order=excluded.sort\_order;

-- Функции upsert -------------------------------------------------------------

create or replace function upsert\_country(p\_name text, p\_enabled boolean default true)

returns uuid language plpgsql as $$

declare v\_id uuid;

begin

insert into countries(name, is\_enabled) values (p\_name, coalesce(p\_enabled,true))

on conflict(name) do update set is\_enabled=excluded.is\_enabled

returning id into v\_id;

return v\_id;

end $$;

create or replace function upsert\_block(p\_name text)

returns uuid language plpgsql as $$

declare v\_id uuid;

begin

insert into blocks(name) values (p\_name)

on conflict(name) do update set name=excluded.name

returning id into v\_id;

return v\_id;

end $$;

create or replace function upsert\_type(p\_block text, p\_type text)

returns uuid language plpgsql as $$

declare v\_bid uuid; v\_tid uuid;

begin

v\_bid := upsert\_block(p\_block);

insert into defect\_types(block\_id, name) values(v\_bid, p\_type)

on conflict(block\_id,name) do update set name=excluded.name

returning id into v\_tid;

return v\_tid;

end $$;

create or replace function upsert\_loc\_category(p\_name text)

returns uuid language plpgsql as $$

declare v\_id uuid;

begin

insert into location\_categories(name) values (p\_name)

on conflict(name) do update set name=excluded.name

returning id into v\_id;

return v\_id;

end $$;

create or replace function upsert\_location(p\_cat text, p\_loc text)

returns uuid language plpgsql as $$

declare v\_cid uuid; v\_lid uuid;

begin

v\_cid := upsert\_loc\_category(p\_cat);

insert into locations(category\_id, name) values (v\_cid, p\_loc)

on conflict(category\_id,name) do update set name=excluded.name

returning id into v\_lid;

return v\_lid;

end $$;

create or replace function upsert\_norm(p\_country text, p\_code text, p\_point text, p\_url text, p\_full text)

returns uuid language plpgsql as $$

declare v\_cid uuid; v\_nid uuid;

begin

v\_cid := upsert\_country(p\_country, true);

insert into norms(country\_id, code, point, source\_url, full\_text)

values (v\_cid, p\_code, p\_point, p\_url, p\_full)

on conflict(country\_id, code, point) do update set

source\_url = excluded.source\_url,

full\_text = excluded.full\_text

returning id into v\_nid;

return v\_nid;

end $$;

create or replace function upsert\_defect(

p\_country text,

p\_block text,

p\_type text,

p\_problem text,

p\_description text,

p\_solution text,

p\_norm\_code text,

p\_norm\_point text,

p\_severity text,

p\_loc\_cat text,

p\_location text

) returns uuid language plpgsql as $$

declare

v\_cid uuid;

v\_bid uuid;

v\_tid uuid;

v\_sid bigint;

v\_lid uuid;

v\_nid uuid;

v\_id uuid;

begin

v\_cid := upsert\_country(p\_country, true);

v\_tid := upsert\_type(p\_block, p\_type);

select block\_id into v\_bid from defect\_types where id=v\_tid;

if p\_loc\_cat is not null and p\_location is not null then

v\_lid := upsert\_location(p\_loc\_cat, p\_location);

end if;

if p\_norm\_code is not null and p\_norm\_point is not null then

v\_nid := upsert\_norm(p\_country, p\_norm\_code, p\_norm\_point, null, null);

end if;

if p\_severity is not null then

select id into v\_sid from severity\_levels where name = p\_severity;

end if;

insert into defects(country\_id, block\_id, type\_id, problem, description, solution, norm\_id, severity\_id, location\_id)

values (v\_cid, v\_bid, v\_tid, p\_problem, p\_description, p\_solution, v\_nid, v\_sid, v\_lid)

on conflict (country\_id, block\_id, type\_id, problem, coalesce(norm\_id,'00000000-0000-0000-0000-000000000000'))

do update set description = excluded.description, solution = excluded.solution, severity\_id = excluded.severity\_id, location\_id = excluded.location\_id

returning id into v\_id;

return v\_id;

end $$;

commit;

📌 Рекомендации по использованию:

1. Выполните скрипт в **Supabase SQL Editor** или psql под **service\_role**.
2. Для пакетной загрузки CSV используйте \copy + вызовы upsert\_defect(...) по строкам.
3. После запуска вьюхи defects\_full\_v2 и defects\_view\_v2 будут готовы для UI.
4. RLS включены — доступ anon/authenticated ограничен согласно политикам.

Доступы Supabase:

NEXT\_PUBLIC\_SUPABASE\_URL:

Direct connection: postgresql://postgres:Cerebrum!986@db.atpmulcyoiiobpbwgzvf.supabase.co:5432/postgres

Transaction pooler: postgresql://postgres.atpmulcyoiiobpbwgzvf:Cerebrum!986@aws-1-eu-north-1.pooler.supabase.com:6543/postgres

Session pooler: postgresql://postgres.atpmulcyoiiobpbwgzvf:Cerebrum!986@aws-1-eu-north-1.pooler.supabase.com:5432/postgres

Выбери нужный линк.

NEXT\_PUBLIC\_SUPABASE\_ANON\_KEY:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6ImF0cG11bGN5b2lpb2JwYndnenZmIiwicm9sZSI6ImFub24iLCJpYXQiOjE3NTc2NzEzMzcsImV4cCI6MjA3MzI0NzMzN30.W6x5TocWD-QpE0Ign2s9FGp\_T2i90hZ0NR0A-SmxsMo