### **Детализированное Техническое Задание: Платформа мониторинга метеоданных**

#### **1. Архитектура решения**

**Стек технологий**:

* **Backend**: Laravel 11 (PHP 8.2+)
* **Frontend**: Vue 3 + Vuetify 3 + Pinia
* **БД**: PostgreSQL 14+ с PostGIS
* **Очереди**: Redis + Laravel Horizon
* **Карты**: Leaflet.js
* **Графики**: Chart.js

#### **2. Расширенная схема БД (PostgreSQL)**

sql

-- Пользователи и роли

CREATE TABLE users (

id SERIAL PRIMARY KEY,

email VARCHAR(255) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL,

full\_name VARCHAR(100) NOT NULL,

role VARCHAR(20) CHECK (role IN ('farmer', 'agronomist', 'technician', 'admin')),

phone VARCHAR(20),

created\_at TIMESTAMPTZ DEFAULT NOW(),

updated\_at TIMESTAMPTZ DEFAULT NOW()

);

-- Культуры растений

CREATE TABLE crops (

id SERIAL PRIMARY KEY,

name VARCHAR(100) UNIQUE NOT NULL, -- "Пшеница", "Кукуруза"

description TEXT

);

-- Фермерские хозяйства

CREATE TABLE farms (

id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

contact\_phone VARCHAR(20),

created\_at TIMESTAMPTZ DEFAULT NOW()

);

-- Поля хозяйств

CREATE TABLE fields (

id SERIAL PRIMARY KEY,

farm\_id INTEGER NOT NULL REFERENCES farms(id) ON DELETE CASCADE,

name VARCHAR(100) NOT NULL,

area\_hectares NUMERIC(10,2),

boundary GEOGRAPHY(POLYGON, 4326) NOT NULL, -- Границы поля

created\_at TIMESTAMPTZ DEFAULT NOW()

);

-- История севооборота

CREATE TABLE crop\_rotations (

id SERIAL PRIMARY KEY,

field\_id INTEGER NOT NULL REFERENCES fields(id) ON DELETE CASCADE,

crop\_id INTEGER NOT NULL REFERENCES crops(id) ON DELETE CASCADE,

planting\_date DATE NOT NULL,

harvest\_date DATE,

created\_at TIMESTAMPTZ DEFAULT NOW()

);

-- Метеостанции

CREATE TABLE stations (

id SERIAL PRIMARY KEY,

serial\_number VARCHAR(50) UNIQUE NOT NULL,

field\_id INTEGER REFERENCES fields(id) ON DELETE SET NULL,

gps\_coordinates GEOGRAPHY(POINT, 4326) NOT NULL,

expected\_location GEOGRAPHY(POINT, 4326), -- Ожидаемая позиция

battery\_level SMALLINT CHECK (battery\_level BETWEEN 0 AND 100),

signal\_strength INTEGER, -- dBm

movement\_speed NUMERIC(5,1), -- км/ч

last\_seen TIMESTAMPTZ,

is\_active BOOLEAN DEFAULT true,

created\_at TIMESTAMPTZ DEFAULT NOW()

);

-- Данные с датчиков

CREATE TABLE sensor\_data (

id BIGSERIAL PRIMARY KEY,

station\_id INTEGER NOT NULL REFERENCES stations(id) ON DELETE CASCADE,

timestamp TIMESTAMPTZ NOT NULL,

temperature NUMERIC(4,1), -- °C

humidity SMALLINT, -- %

soil\_moisture SMALLINT, -- %

precipitation NUMERIC(5,1), -- мм

wind\_speed NUMERIC(5,1), -- м/с

wind\_direction SMALLINT, -- 0-360°

pressure NUMERIC(6,1), -- гПа

created\_at TIMESTAMPTZ DEFAULT NOW()

);

-- Оповещения

CREATE TABLE alerts (

id SERIAL PRIMARY KEY,

station\_id INTEGER REFERENCES stations(id) ON DELETE CASCADE,

user\_id INTEGER REFERENCES users(id) ON DELETE CASCADE,

type VARCHAR(20) CHECK (type IN (

'soil\_dry', 'soil\_flood', 'offline',

'battery', 'movement', 'frost\_risk'

)),

message TEXT NOT NULL,

is\_resolved BOOLEAN DEFAULT false,

triggered\_at TIMESTAMPTZ DEFAULT NOW(),

resolved\_at TIMESTAMPTZ

);

-- Эталонные параметры для культур

CREATE TABLE reference\_data (

id SERIAL PRIMARY KEY,

crop\_id INTEGER NOT NULL REFERENCES crops(id) ON DELETE CASCADE,

parameter VARCHAR(30) NOT NULL, -- "soil\_moisture\_min"

value NUMERIC(5,1) NOT NULL

);

**Индексы**:

sql

CREATE INDEX idx\_sensor\_data\_timestamp ON sensor\_data (station\_id, timestamp DESC);

CREATE INDEX idx\_stations\_location ON stations USING GIST (gps\_coordinates);

CREATE INDEX idx\_fields\_boundary ON fields USING GIST (boundary);

CREATE INDEX idx\_crop\_rotations\_dates ON crop\_rotations (planting\_date, harvest\_date);

#### **3. Модели Laravel**

**User.php**:

php

namespace App\Models;

use Laravel\Sanctum\HasApiTokens;

use Illuminate\Notifications\Notifiable;

use Illuminate\Database\Eloquent\Model;

class User extends Model

{

use HasApiTokens, Notifiable;

protected $fillable = ['email', 'password', 'full\_name', 'role', 'phone'];

public function farms() {

return $this->belongsToMany(Farm::class, 'farm\_users');

}

public function alerts() {

return $this->hasMany(Alert::class);

}

}

**Field.php**:

php

namespace App\Models;

use MatanYadaev\EloquentSpatial\Objects\Polygon;

use MatanYadaev\EloquentSpatial\SpatialBuilder;

class Field extends Model

{

protected $fillable = ['name', 'area\_hectares', 'farm\_id'];

protected $casts = [

'boundary' => Polygon::class,

];

public function newEloquentBuilder($query): SpatialBuilder

{

return new SpatialBuilder($query);

}

public function farm() {

return $this->belongsTo(Farm::class);

}

public function stations() {

return $this->hasMany(Station::class);

}

public function currentCrop() {

return $this->hasOne(CropRotation::class)

->whereDate('planting\_date', '<=', now())

->whereDate('harvest\_date', '>=', now())

->latest();

}

}

**Station.php**:

php

namespace App\Models;

use MatanYadaev\EloquentSpatial\Objects\Point;

use MatanYadaev\EloquentSpatial\SpatialBuilder;

class Station extends Model

{

protected $fillable = ['serial\_number', 'battery\_level', 'signal\_strength'];

protected $casts = [

'gps\_coordinates' => Point::class,

'expected\_location' => Point::class,

];

public function newEloquentBuilder($query): SpatialBuilder

{

return new SpatialBuilder($query);

}

public function field() {

return $this->belongsTo(Field::class);

}

public function sensorData() {

return $this->hasMany(SensorData::class)

->orderBy('timestamp', 'DESC');

}

public function latestData() {

return $this->hasOne(SensorData::class)

->latest('timestamp');

}

}

**SensorData.php**:

php

namespace App\Models;

class SensorData extends Model

{

protected $fillable = [

'temperature', 'humidity', 'soil\_moisture',

'precipitation', 'wind\_speed', 'wind\_direction',

'pressure', 'timestamp'

];

protected $casts = [

'timestamp' => 'datetime'

];

public function station() {

return $this->belongsTo(Station::class);

}

}

**CropRotation.php**:

php

namespace App\Models;

class CropRotation extends Model

{

protected $fillable = ['planting\_date', 'harvest\_date'];

protected $casts = [

'planting\_date' => 'date',

'harvest\_date' => 'date',

];

public function field() {

return $this->belongsTo(Field::class);

}

public function crop() {

return $this->belongsTo(Crop::class);

}

public function referenceValues() {

return $this->hasManyThrough(

ReferenceData::class,

Crop::class,

'id',

'crop\_id',

'crop\_id',

'id'

);

}

}

#### **4. API Endpoints**

**Для метеостанций**:

* POST /api/v1/stations/data - Прием данных

json

{

"serial": "ST-001",

"token": "station\_secret\_key",

"data": {

"timestamp": "2025-06-17T15:25:00Z",

"temperature": 22.4,

"humidity": 65,

"soil\_moisture": 42,

"precipitation": 0,

"wind\_speed": 3.2,

"wind\_direction": 270,

"pressure": 1013.2,

"battery": 85,

"signal": -70,

"gps": [50.4504, 30.5245]

}

}

**Для фронтенда**:

* GET /api/fields - Список полей пользователя
* GET /api/stations/{id}/data?period=7d - Данные станции
* POST /api/alerts/{id}/resolve - Закрытие оповещения

#### **5. Визуализация интерфейсов (ASCII)**

**Дашборд фермера**:

text

+-------------------------------------------------------------------------+

| [Лого] МетеоАгро | Добро пожаловать, Иван Петров! |

+-------------------+-------------------------------+-----------------------+

| ПОЛЕ: Поле-12 (Пшеница) | СТАНЦИЯ: ST-789 | Батарея: 85% |

| Площадь: 120 га | Сигнал: ████▂ 70 dBm | Онлайн |

| Координаты: 50.4504, 30.5245 | Последние данные: 15:25 (10 мин.назад|

+----------------------------------+---------------------------------------+

| ТЕКУЩИЕ ПОКАЗАТЕЛИ: |

| Температура: 22.4°C Влажн.возд: 65% Влажн.почвы: 42% (оптимум) |

| Ветер: 3.2 м/с (СЗ) Осадки: 0 мм Давление: 1013.2 гПа |

+----------------------------------+---------------------------------------+

| [График] Температура/влажность: ▁▂▅▇█▇▅▂▁ |

| [Кнопка "История"] [Кнопка "Сравнить"] [Кнопка "Отчет за сезон"] |

+----------------------------------+---------------------------------------+

| СОСЕДНИЕ ПОЛЯ (радиус 10 км): |

| 1. Поле-8 (Кукуруза): темп. +1.2°C, влажн. -5% |

| 2. Поле-15 (Соя): темп. -0.8°C, влажн. +3% |

+-------------------------------------------------------------------------+

**Страница севооборота**:

text

+-------------------------------------------------------------------------+

| ИСТОРИЯ СЕВООБОРОТА: Поле-12 |

+------+----------------+----------------+----------------+----------------+

| Год | Культура | Дата посева | Дата сбора | Урожайность |

+------+----------------+----------------+----------------+----------------+

| 2024 | Пшеница | 15.03.2024 | 20.08.2024 | 45 ц/га |

| 2023 | Рапс | 10.04.2023 | 18.09.2023 | 28 ц/га |

| 2022 | Кукуруза | 01.05.2022 | 15.10.2022 | 85 ц/га |

+------+----------------+----------------+----------------+----------------+

| ПЛАНИРОВАНИЕ: |

| [Выбрать культуру: ▾] [Дата посева: ▸] [Прогноз урожайности: 50 ц/га] |

| [Сохранить план] |

+-------------------------------------------------------------------------+

**Админ-панель (мониторинг)**:

text

+-------------------------------------------------------------------------+

| АДМИНИСТРАТОР | Статус системы: 99.8% uptime | 24 новых оповещения |

+---------+------------+-----------+----------+-------------+-------------+

| ID стан.| Хозяйство | Заряд | Сигнал | Посл. данные | Статус |

+---------+------------+-----------+----------+-------------+-------------+

| ST-789 | АгроШанс | 85% | ████▂ 70 | 15:25 | ONLINE |

| ST-790 | ПолеVision | 12% ! | █▂▂▂▂ 92 | 14:40 | OFFLINE |

| ST-791 | Зеленый | 100% | █████ 45 | 15:30 | MOVEMENT! |

+---------+------------+-----------+----------+-------------+-------------+

| [ДЕЙСТВИЯ]: Добавить станцию | Настройки оповещений | Массовое обновление|

+-------------------------------------------------------------------------+

| КАРТА СТАНЦИЙ: |

| [⚠] ST-791 - Перемещение на 1.2 км |

| [●] ST-789 - Норма |

| [✖] ST-790 - Офлайн > 1 час |

| |

| +------------------------------------------------------------------+ |

| | .,-:::::/. ST-791 ⚠ | |

| | ST-790 ✖ ,' ;-''`;-''. ST-789 ● | |

| | ;' ; ' \ | |

| +------------------------------------------------------------------+ |

+-------------------------------------------------------------------------+

#### **6. Ключевые бизнес-процессы**

**1. Регистрация данных от станции**:

php

class StationDataController extends Controller

{

public function store(StationDataRequest $request)

{

$station = Station::where('serial\_number', $request->serial)

->where('api\_token', $request->token)

->firstOrFail();

// Сохранение данных

$data = $station->sensorData()->create(

array\_merge($request->data, ['timestamp' => now()])

);

// Обновление статуса станции

$station->update([

'battery\_level' => $request->battery,

'signal\_strength' => $request->signal,

'gps\_coordinates' => new Point($request->gps[0], $request->gps[1]),

'last\_seen' => now(),

]);

// Проверка перемещения

if ($station->expected\_location &&

$station->expected\_location->distanceTo($station->gps\_coordinates) > 1000)

{

AlertService::triggerMovementAlert($station);

}

// Проверка агро-параметров

if ($station->field && $station->field->currentCrop) {

AlertService::checkSoilConditions($data, $station->field->currentCrop);

}

return response()->json(['status' => 'success']);

}

}

**2. Проверка состояния почвы**:

php

class AlertService

{

public static function checkSoilConditions(SensorData $data, CropRotation $cropRotation)

{

$references = $cropRotation->crop->referenceValues;

$soilMoistureRef = $references->firstWhere('parameter', 'soil\_moisture\_optimal');

$tempRef = $references->firstWhere('parameter', 'temperature\_frost\_threshold');

// Проверка влажности

if ($soilMoistureRef && $data->soil\_moisture < ($soilMoistureRef->value \* 0.8)) {

self::createAlert(

$data->station,

'soil\_dry',

"Критическая сухость почвы: {$data->soil\_moisture}%"

);

}

// Проверка заморозков

if ($tempRef && $data->temperature < $tempRef->value) {

self::createAlert(

$data->station,

'frost\_risk',

"Риск заморозков: {$data->temperature}°C"

);

}

}

}

**3. Сравнение с соседними полями**:

php

class FieldCompareService

{

public function getComparativeData(Field $field, $radiusKm = 10)

{

$currentFieldData = $field->latestData;

$neighbors = Field::where('id', '!=', $field->id)

->whereRaw(

"ST\_DWithin(boundary, ?, ?)",

[$field->boundary, $radiusKm \* 1000]

)

->with(['currentCrop', 'latestData'])

->get();

return [

'current' => $currentFieldData,

'neighbors' => $neighbors->map(function($neighbor) use ($currentFieldData) {

return [

'name' => $neighbor->name,

'crop' => $neighbor->currentCrop->name,

'temperature\_diff' => $neighbor->latestData->temperature - $currentFieldData->temperature,

'humidity\_diff' => $neighbor->latestData->humidity - $currentFieldData->humidity,

];

})

];

}

}

#### **7. Инфраструктура**

* **Сервер**: 2x Intel Xeon Silver 4210, 128 ГБ RAM, 2x1TB NVMe (RAID 1)
* **Балансировка**: Nginx + Keepalived
* **Резервирование**: Daily backups + репликация БД в другой ЦОД
* **Мониторинг**: Prometheus + Grafana
* **СД/CI**: GitLab CI, Docker, Ansible

#### **8. Оптимизация для масштабирования**

1. **Партицирование данных**:
   * Таблица sensor\_data партицирована по месяцам
   * Использование TimescaleDB для временных рядов
2. **Кэширование**:
   * Redis для кэша метрик станций
   * Varnish для статических данных API
3. **Обработка очередей**:
   * Распределенная обработка оповещений через Laravel Horizon
   * Приоритетные очереди для критических алертов

yaml

# horizon.yaml

environments:

production:

supervisor-1:

connection: redis

queue: 'alerts,default'

balance: auto

processes: 10

max-time: 60

max-jobs: 1000

#### **9. Безопасность**

1. **Аутентификация**:
   * Пользователи: Laravel Sanctum + JWT
   * Станции: HMAC подпись запросов
2. **Защита данных**:
   * Шифрование данных сенсоров в rest и transit
   * Регулярные аудиты безопасности
3. **Роли**:  
   php

Gate::define('view-station', function (User $user, Station $station) {

return $user->farms->contains($station->field->farm);

1. });

#### **10. Тестирование**

1. **PHPUnit тесты**:
   * Тестирование обработки данных станций
   * Проверка бизнес-логики оповещений
   * Тесты геопространственных запросов
2. **Фронтенд тесты**:
   * Jest + Vue Test Utils для компонентов
   * Cypress для e2e тестирования
3. **Нагрузочное тестирование**:
   * Имитация 500 станций с интервалом 5 мин
   * Тест на 1000 одновременных пользователей

### **Заключение**

Предложенная архитектура обеспечивает:

* Масштабируемость до 1000+ станций
* Надежное хранение исторических данных
* Реальное время обновления информации
* Интеграцию агрономических параметров
* Кросс-ролевой доступ к данным
* Автоматизацию критических оповещений

Для реализации MVP рекомендуется поэтапное внедрение:

1. Базовый сбор данных и дашборд (2 мес)
2. Система севооборота и аналитика (1.5 мес)
3. Мобильное приложение + push-уведомления (1 мес)
4. Интеграция с системами полива (фаза 2)