WeatherAlerts

WeatherAlerts е дистрибуиран систем насочен за алармирање на временски непогоди кој обезбедува следење, обработка и известување во реално време за метеоролошките услови низ повеќе градови. Системот интегрира повеќе микросервиси кои даваат податоци (на пример нивоа на PM2.5, PM10, NO_2, брзина на ветерот, интензитет на врнежи од дожд, UV индекс), ги насочуваат низ брокер за пораки, ги обработуваат и доставуваат известувања до мобилни клиенти преку WebSocket и push notifications.

Проектот демонстрира практична примена на RabbitMQ.

Функционални барања

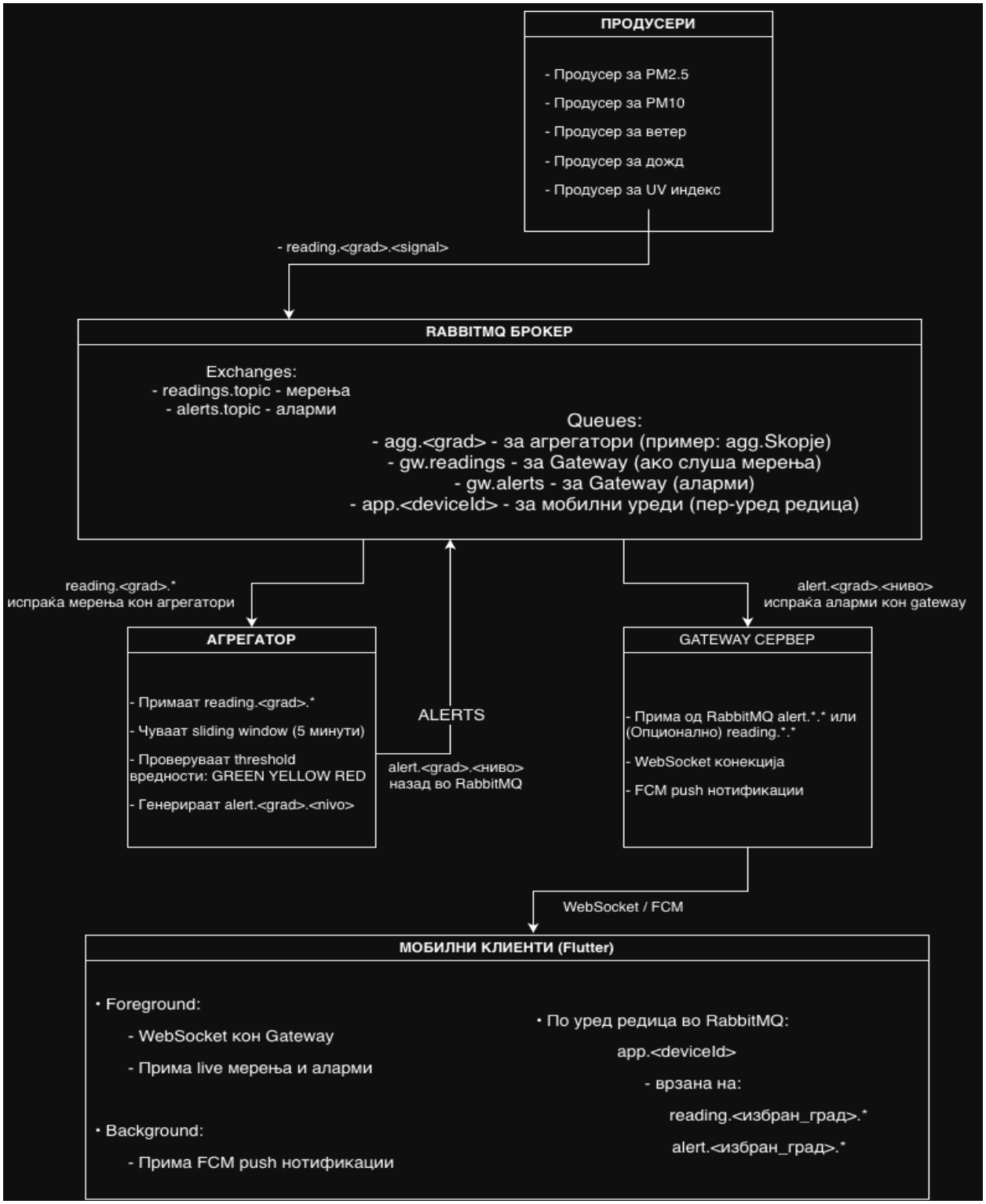
1. Поддршка за повеќе градови (Скопје, Битола, Велес, Охрид...).
2. Следење повеќе параметри:
    - PM2.5, PM10, NO_2
    - Брзина на ветер
    - Интензитет на дожд
    - UV индекс
3. Продусерите континуирано испраќаат мерења.
4. Системот генерира аларми:
    - GREEN
    - YELLOW
    - RED
5. Мобилните уреди примаат:
    - WebSocket live stream (foreground)
    - Push нотификации (background)

Содржина

WeatherAlerts користи слоевита архитектура:

1. **Producer слој** – Ги праќаат мерењата.
2. **RabbitMQ брокер** – рутира пораки преку topic exchanges.
3. **Агрегаторен слој** – sliding window + правила за аларм.
4. **Интерфејсен Gateway слој** – WebSocket интерфејс + Firebase Cloud Messaging (FCM) push.
5. **Мобилна апликација** – визуелен интерфејс за корисникот.

Дијаграм


ПРОДУСЕРИ
- Продусер за PM2.5
- Продусер за PM10
- Продусер за ветер
- Продусер за дожд
- Продусер за UV индекс
- reading.<grad>.<signal>
RABBITMQ БРОКЕР
Exchanges:
- readings.topic - мерења
- alerts.topic - аларми
Queues:
- agg.<grad> - за агрегатори (пример: agg.Skopje)
- gw.readings - за Gateway (ако слуша мерења)
- gw.alerts - за Gateway (аларми)
- app.<deviceId> - за мобилни уреди (пер-уред редица)
reading.<grad>.*
испраќа мерења кон агрегатори
alert.<grad>.<ниво>
испраќа аларми кон gateway
АГРЕГАТОР
- Примаат reading.<grad>.*
- Чуваат sliding window (5 минути)
- Проверуваат threshold вредности: GREEN YELLOW RED
- Генерираат alert.<grad>.<nivo>
ALERTS
alert.<grad>.<ниво>
назад во RabbitMQ
GATEWAY СЕРВЕР
- Прима од RabbitMQ alert.*.* или (Опционално) reading.*.*
- WebSocket конекција
- FCM push нотификации
WebSocket / FCM
МОБИЛНИ КЛИЕНТИ (Flutter)
• Foreground:
- WebSocket кон Gateway
- Прима live мерења и аларми
• Background:
- Прима FCM push нотификации
• По уред редица во RabbitMQ:
app.<deviceId>
- врзана на:
reading.<избран_град>.*
alert.<избран_град>.*

Дизајн на идеата

- Продусери

Микросервиси (мали програми) кои генерираат или читаат сензорски податоци и ги праќаат во системот.

- Producer за PM2.5 (загадување)
- Producer за ветер
- Producer за дожд
- Producer за UV индекс

Responsibilities:

- Читаат податоци од сензори.
- Нормализираат вредности.
- Публикуваат пораки кон `readings.topic`.
- Се хоризонтално скалабилни и stateless.

Како функционира?

1. Добиваат некакво мерење;
2. Го стават во JSON формат;
3. Го праќаат во RabbitMQ со routing key:

```
reading.<grad>.<signal>
```

Пример JSON:

```
{
      "city": "Skopje",
      "signal": "pm25",
      "value": 87.4,
      "unit": "µg/m³",
      "timestamp": "2025-11-25T19:00:00Z"
}
```

- RabbitMQ Message Broker

RabbitMQ обезбедува асинхронска комуникација помеѓу компонентите.

Секој комуницира само со RabbitMQ. Producer-от не мора да знае за aggregator, aggregator-от не мора да знае за gateway, gateway-от не мора да знае за уредите.

RabbitMQ ги распределува пораките меѓу потрошувачите по round-robin. Со приватна редица по град, секој уред добива *сè* што е врзано за неговиот град каде тој е поврзан.

Како функционира?

1. Ги прима сите пораки од producer-ите.
2. Им ги доставува на агрегаторите.
3. Ги прима алармите и ги доставува до gateway и мобилни уреди.
4. Секоја порака оди само таму каде што е врзана преку routing rules.

Exchanges (тука влегуваат пораките):

- `readings.topic` – мерења
- `alerts.topic` – аларми

Routing Keys (ги адресираат пораките):

- Мерења → `reading.<grad>.<signal>`
- Аларми → `alert.<grad>.<nivo>`

Queues (редови каде чекаат пораките):

- `agg.Skopje`, `agg.Bitola`, `agg.Veles`, `agg.Ohrid` → агрегатори
- `gw.readings`, `gw.alerts` → gateway
- `app.<deviceId>` → мобилни уреди

- Агрегатори

Агрегаторите се микросервиси за локална анализа кои ги примаат сите мерења и одлучуваат дали е потребно да се активира аларм.

Тие се потребни за паметна обработка на податоци и да детектираме опасни ситуации (црвен аларм)

Како функционира?

1. Ги примаат сите мерења за еден град.
2. Ги чуваат последните 5 минути податоци (sliding window).
3. Пресметуваат статистики (avg, max, min).
4. Применуваат правила за аларм → Проверуваат дали се надминати границите (thresholds).
5. Ако да → Публикуваат аларми во `alerts.topic`..
6. Алармот оди назад во RabbitMQ.

Пример JSON ALERT:

```
{
      "city": "Skopje",
      "signal": "pm25",
      "level": "RED",
      "value": 87,
      "threshold": 75,
      "timestamp": "2025-11-25T19:02:00Z"
}
```

Скалабилност:

- Може да има повеќе агрегатори по град. Ако еден падне, другиот продолжува.

- Gateway (API/WebSocket сервер)

Gateway е јавниот backend кој комуницира со мобилните уреди.

Мобилните уреди не можат директно да зборуваат со RabbitMQ. Gateway е посредник меѓу RabbitMQ и уредите.

Responsibilities:

- Live WebSocket конекции.
- Push нотификации Firebase Cloud Messaging (FCM) при критични аларми.
- Компатибилност со RabbitMQ queues.
- Stateless – не чува податоци во меморија
- Може лесно да се скалира (повеќе Gateway сервери истовремено)

Како функционира?

1. Прима аларми од RabbitMQ.
2. Ако мобилниот уред е „отворен“ → испраќа WebSocket порака (real-time).
3. Ако уредот е во background → испраќа push нотификација (FCM).
4. Им ги праќа и сите нови мерења на активните клиенти.

WebSocket пример:

```
wss://gateway/weather?city=Skopje&deviceId=abcd1234
```

- Мобилна Апликација - Flutter

Мобилната апликација е крајната дестинација за корисниците.

Foreground функционалност

- WebSocket streaming.
- Live графици.
- Визуелни аларми.

Background функционалност

- Firebase Cloud Messaging push нотификации.

Карактеристики:

- Одбирање град.
- Приказ на сегашни мерења.
- Приказ на аларм ниво.
- Историски податоци (опционално).

RabbitMQ Топологија

Чекор 1 – Мерења:
Producer → `readings.topic` → Aggregators → Devices → Gateway

Чекор 2 – Аларми:
Aggregators → `alerts.topic` → Gateway → Devices

Чекор 3 – Device Routing:
Секој уред ги добива ***сите*** пораки за избраниот град.

Тек на податоците (Data Flow Sequence)

1. Продусер испраќа мерење.
2. RabbitMQ го рутира кон агрегаторот и уредите.
3. Агрегатор го додава во sliding window.
4. Ако threshold е надминат → генерира аларм.
5. RabbitMQ го испраќа алармот кон Gateway + уредите.
6. Gateway испраќа WebSocket + FCM нотификации.
7. Мобилната апликација го прикажува алармот.

JSON Примери

Reading:

```
{
      "city": "Ohrid",
      "signal": "wind",
      "value": 12.1,
      "unit": "m/s",
      "timestamp": "2025-11-25T20:11:00Z"
}
```

Alert:

```
{
      "city": "Ohrid",
      "level": "YELLOW",
      "signal": "wind",
      "value": 12.1,
      "threshold": 10,
      "timestamp": "2025-11-25T20:11:05Z"
}
```
