System Tech Spec.

Technical descriptions overview.

- This document describes the technical implementation of Red Alert system, Mobile Application and Website.
- This document reviews the suggested technologies which best match the product.
- This document describes the use of server, database, storage, and front sides (websites and admins), the main features, description of functionalities, and the project's technical environments.

Table of contents

Project Overview

Description

This project will contain the following interfaces:

1. Global site.
2. Admin for the global site for content editing.
3. Mobile App (Android & IOS).
4. RSS.

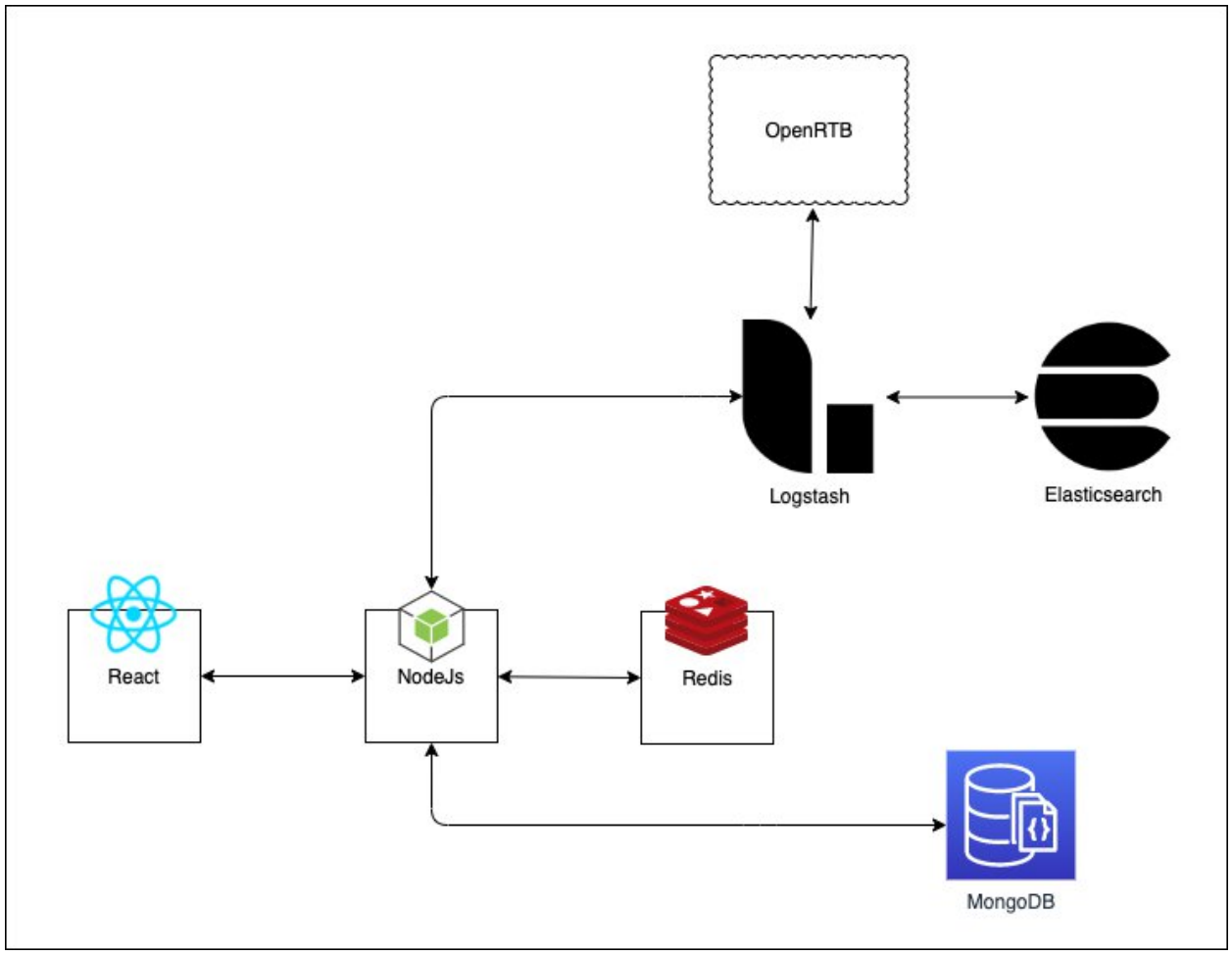
Development environments & Languages:

- **Global site -** The Global site will be developed using Wordpress template.
- **Admin -** The Admin site will be developed using Angular/Reactjs.
- **Mobile App -** The Mobile app will be developed using React Native platform.
- **Backend side-** The Backend side will be developed using Nodejs with ExpressJS and Typescript.
- **Cache management -** The application cache will be managed using Redis.
    - **Redis -** A key-value in-memory data structure store, used as a database, cache and message broker. Will be managed by Azure's service - Azure cache for Redis.
- **Database -** Server's DB will be MongoDB - document-based, distributed database built for modern application. Will be managed by MongoDB Atlas on Azure.
- **Elasticsearch (ELK)** will be as a service on Azure, ELK stands for -
    - **Elasticsearch -** a search and analytic engine.
    - **Logistash -** a pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to Elasticsearch.
    - **Kibana -** lets users visualize data with charts and graphs in Elasticsearch.

Terminology:

- **Azure** - Microsoft cloud services, Major supplier for VMs(Virtual Machines), Cloud Storage solutions and many more.
- **App Service** - Azure App Service is an HTTP-based service for hosting web applications, REST APIs, and mobile back ends.
- **Cosmos DB** - Azure Cosmos DB is Microsoft's globally distributed, multi-model database service.
- **NodeJS** - Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside a web browser.
- **ExpressJs** - Express.js, is a web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.
- **Redis -** Is an in-memory data structure project implementing a distributed, in-memory key–value database with optional durability. Redis supports different kinds of abstract data structures, such as strings, lists, maps, sets, sorted sets, HyperLogLogs, bitmaps, streams, and spatial indexes.
- **MongoDB** - MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.
- **Elasticsearch** - Elasticsearch is a search engine based on the Lucene library. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.

Architecture



Server side

- **Backend application** -The Backend application will be developed using Nodejs with ExpressJS and Typescript. The application will use Azure App Service for hosting the application. The application interface will be developed as a REST API.
- **Caching** - The backend application will support caching layer, implementing with redis, to minimize time responses.
- **Databases** - The CRUD operations will be triggered from node application. The data will be saved into mongo documents. part of the data, for complex queries will be saved also in Elasticsearch database.
    - **Mongodb**, NoSQL database, The mongo database will store all the application entities.
    - **Elasticsearch** - Elasticsearch database will be used for complex queries and data fusion.
- **Authentication** - The communication with the server will be through a REST API, all the routes will enforce authentication. For authentication we will use **JWT**.
- **Authorization -** The application will implement authorization layers to manage user access control. the "CASL" library for implementing this layer.

Client side:

- Admin - The application will be hosted on azure App service.
    - **Web application** - will be developed with ReactJS. framework features:
        - Typescript
        - Redux & Redux Thunk - state management
        - Unit testing using Jest
    - **Data visualization** - will be composed of several libraries and depending on the final design:
        - Charts.js
    - **Map view** - using MapBox for map related visualization.
- Global site - The application will be hosted on azure App service.
- Mobile App - The application will be distributed by Google play and Apple Store.
- RSS - The RSS feed will be saved on the server and will be served through a special route.

Main Features

Global Web App main features:

- The user will be able to see all relevant notifications in a list or on a map.
    - The user will be able to filter notifications by:
        - Location
        - Time base
        - Type

Admin Web App main features:

- The user will be able to create/edit/remove an event.
- The user will be able to manage the push notifications for each event.
- The user will be able to see all analytics data per event:
    - Push data:
        - Users who received push notifications
        - Users who opened push notifications
        - Users mobile data
        - Users location
- The admin will be able to see all relevant notifications in a list or on a map.
    - The user will be able to filter notifications by:
        - Location
        - Time base
        - Type

Mobile App main features:

- The user will be able to see specific events that relevant to him by:

        - Location
        - Time base
        - Type

- The user will have to accept location permissions.
- The user will have to accept notification permissions.
- The user will get notifications based on his current location.
- The application will sample every X hours the user location and will save it “locally”.
- The application will send the user location to the server by network availability.

Server side REST API main features:

The server REST API application will be divided into 2 main sections:

1. **REST API for Admin application** -
    a. All routes will be private and enforced authentication before receiving any data.
    b. Authentication process will use the microsoft active directory to authenticate users.
2. **Global / Mobile application** -
    a. All routes will be public, no Authentication will be required.
    b. Getting data on active events
    c. Save user mobile location every X hours.
    d. Serve RSS file.
    e. RSS counting / logs

Database

There will be two main sources of data :

1. **Mobile application -** The application will supply most of the geo location data for a user.
2. **OpenRTB** - This source of the data will supply more insight on the applications, device metadata and geo location about users.

The fusion of the data can be made by using the user IFA (i.e. Identifier for Advertisers), which is an ID that the advertisers use to track devices. Hence, the IFA can be used to track users. This identifier can be extracted from both sources, Mobile application (Red Alert) and OpenRTB source.

Entities:

- OpenRTB Data
    - IFA - (Identifier for advertisers)
    - app_id
    - app_name
    - device_carrier
    - device_make,
    - device_model
    - device_ip
    - device_os
    - device_osv
    - device_language
    - geo_city
    - geo_country
    - geo_lat
    - geo_lon

    - geo_region
    - geo_type
    - Geo_zip

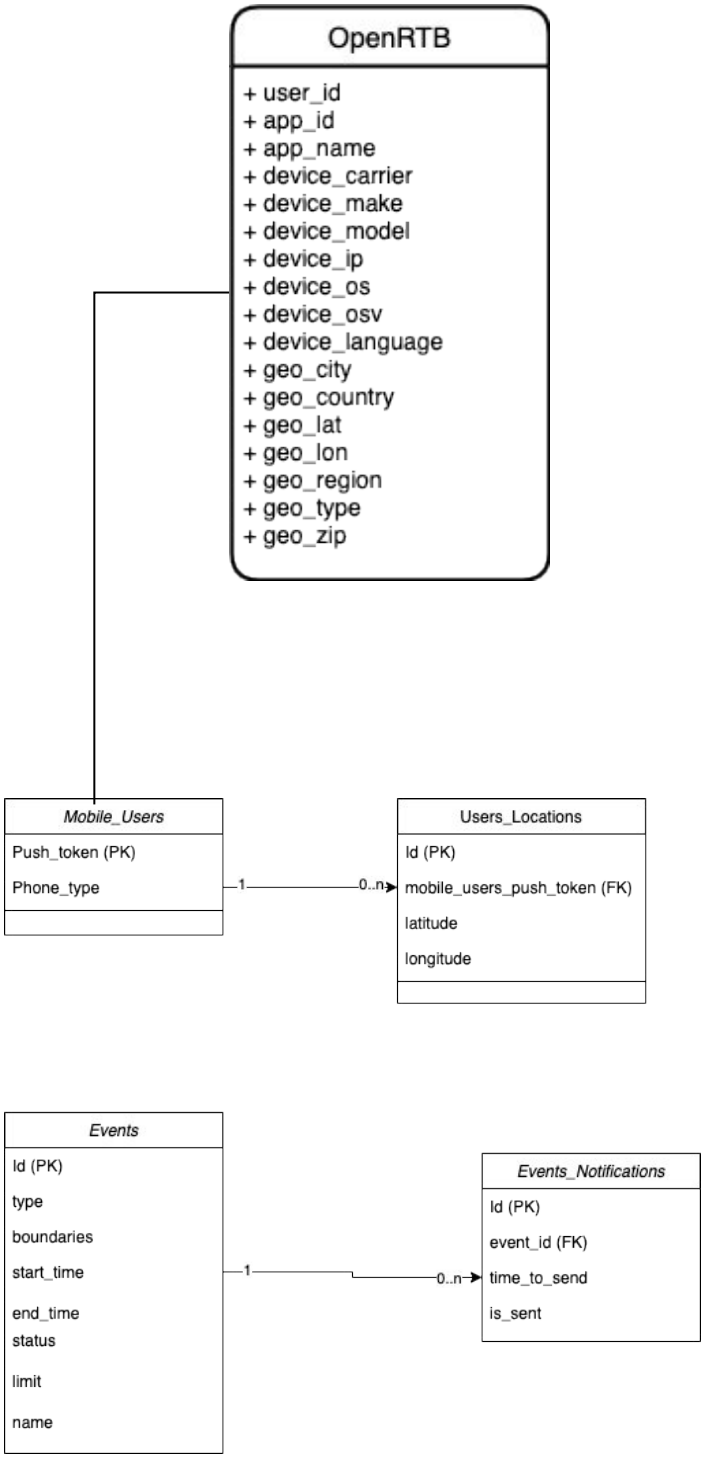
- Mobile Users
    - Push Token id (PK) - String
    - Phone type - String

- Mobile Users Locations
    - Mobile user Push Token id (FK) - String
    - Latitude - Double, user location latitude
    - Longitude - Double, user location longitude

- Events
    - Id (PK)
    - Type - Event type
    - Boundaries - Array of geo points (Polygon)
    - Start time - Date
    - End time - Date when the event is irrelevant
    - Status - Active/Not Active
    - Limit - Number, using for indicate maximum number of users inside a polygon

- Event Notification
    - Id (PK)
    - Event_id (FK)
    - Time to sent - Date, the date to send the notification
    - isSent - Boolean, notification already sent?

Schema:


OpenRTB
+ user_id
+ app_id
+ app_name
+ device_carrier
+ device_make
+ device_model
+ device_ip
+ device_os
+ device_osv
+ device_language
+ geo_city
+ geo_country
+ geo_lat
+ geo_lon
+ geo_region
+ geo_type
+ geo_zip
Mobile_Users
Push_token (PK)
Phone_type
1
0..n
Users_Locations
Id (PK)
mobile_users_push_token (FK)
latitude
longitude
Events
Id (PK)
type
boundaries
start_time
end_time
status
limit
name
1
0..n
Events_Notifications
Id (PK)
event_id (FK)
time_to_send
is_sent

Environments

There will be 3 environments for the Server, Global site and Admin site, Develop, QA and production.

- **Production** - should use load-balancer and auto-scaling services.
- **Staging** - for QA proposes, this environment should be identical to the production environment in order to predict any issue before each production deployment.
- **Dev** - this environment should be used only by developers in order to test their development functionality and behavior on air, many cases during development require this ability of testing.

CI/CD:

- Deployment process - All Deployment processes will be automatic with automation servers such as Azure DevOps service.
- All code bases will be hosted on gitHub.

Mobile App CI/CD:

- Deploy code process:
    - Developers update the master branch on the site repository via git.
    - Automated process will sign the application and distribute the new version for the QA process.
    - When the new version will be approved (by a person) an automated process will be deployed to google and apple store, using the Fastlane framework.

Web & Server App CI/CD:

- Deploy code process:
    - Developers update the master branch on the site repository via git.
    - Automated processes will build and deploy the new code to the staging slot on the Web app for QA process.
    - When the new version will be approved (by a person) an automated process will be started and deploy the new version to the production slot.