UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESIGN AND IMPLEMENTATION OF A SOCIAL MEDIA MONITORING WEB APPLICATION BASED ON REACT FRAMEWORK AND ELASTICSEARCH

ÁLVARO FERNÁNDEZ RIVERO
ENERO 2019

TRABAJO FIN DE GRADO

Título:	Diseño e implementación de una aplicación web basada en React y Elasticsearch para la monitorización de medios de comunicación
Título (inglés):	Design and implementation of a Social Media Monitoring Web Application based on React framework and Elasticsearch
Autor:	Álvaro Fernández Rivero
Tutor:	Carlos A. Iglesias Fernández
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:

Vocal:

Secretario:

Suplente:

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

DESIGN AND IMPLEMENTATION OF A SOCIAL MEDIA MONITORING WEB APPLICATION BASED ON REACT FRAMEWORK AND ELASTICSEARCH

Álvaro Fernández Rivero

Enero de 2019

Resumen

Esta memoria es el resultado de un proyecto cuyo objetivo ha sido el diseño y desarrollo de una interfaz web basada en React y Elasticsearch, que permite la monitorización de noticias de terrorismo islámico.

Uno de los objetivos propuestos al comienzo del proyecto era la integración de una de las tecnologías más punteras en el desarrollo web (React) con uno de los mejores motores de búsqueda (Elasticsearch). Para hacer esto posible, se ha utilizado el framework de ReactiveSearch, que nos ofrece componentes de React con conexión a un índice de Elasticsearch.

La utilización de ReactiveSearch nos ha permitido la integración de componentes de librerías externas de React. Esto nos permite crear nuestros propios componentes de filtrado utilizando por ejemplo Google Maps o gráficos.

Además, se ha implementado un sistema de alertas personalizadas para cada usuario, pudiendo seleccionar los temas que más le interesan. Para ello, se ha utilizado un servidor NodeJS con conexión a una base de datos MongoDB para almacenar los datos de los usuarios.

Por último, una de las grandes ventajas del uso de React es la fácil migración de la aplicación web a una aplicación móvil nativa de Android o iOS. La lógica de los componentes es reutilizable, y únicamente hemos de cambiar los elementos HTML utilizados y el diseño de los mismos. Esto nos ha permitido desarrollar una aplicación móvil para poder filtrar y ver los resultados obtenidos.

Palabras clave: React, Terrorismo, Elasticsearch, Monitorización, Noticias, Alertas, Móvil, Aplicación

Abstract

This thesis is the result of a project whose objective has been the design and the development of a web interface based on React and Elasticsearch, which allows the monitoring of news related to Islamic terrorism.

One of the objectives purposed when the project started, was the integration of one of the most popular technologies in web development (React) with one of the best search engines (Elasticsearch). To make this possible, ReactiveSearch framework has been used, which offers us React components connected to an Elasticsearch index.

The usage of ReactiveSearch has allowed us making the integration of components of external libraries from React. This enables us to create our own filtering components using for instance Google Maps or charts.

Furthermore, a custom alerts system for each user has been implemented, so users will be able to select topics which they are interested in. For making this possible, a NodeJS server with connection to a MongoDB database has been used in order to store users data.

Finally, one of the main features of React consists on the easy migration from a web application to a native application of Android or iOS. The logic of the components is reusable, so we only have to change HTML elements used and their design. This has allowed us to develop a native mobile application to filter and see results obtained.

Keywords: React, Elasticsearch, Monitoring, News, Alerts, Mobile, Application

Agradecimientos

Gracias a mis padres, a mis hermanos y a Sofía por ayudarme desde el primer momento.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Context

Nowadays, when millions of bytes of data travel around the Internet, data analysis has become one of the most interesting topics in the world. Millions of users who surf the Web are looking for platforms which can analyse those data.

Social media monitoring tools offers us the possibility of making data analysis. They provide us interfaces to filter between millions of data or send alerts from recent news of topics we are interested.

In this project we are working on that: making interfaces which help us to manage data. So what we need is a fast search engine to filter data and a fast website to see data. The future requires emerging technologies, so the more recent the technology used, the more future the project will have.

The topic we are mainly managing is jihadist attacks. It is one of the main problems in the world. In the recent years, there have been thousands of terrible attacks in Europe, so it is under our responsibility to help as much as we can to stop that. This wave of recent attacks mainly from Daesh started in 2015, when Charlie Hebdo offices were assaulted by two terrorists provoking twelve deaths.

Between that moment and today, tens of terrible attacks have occurred in Europe. The most relevant have caused hundreds of victims such as the attack in Bataclan (Paris) which provoked 130 deaths and more than 300 wounded people, or the assault in Nice, which caused 85 deaths and more than 100 wounded people. *Data extracted from [4]*

This project takes part of a global project called Trivalent (Terrorism pReventIon Via rAdicaLisation countEr-NarraTive) [12], which aims to give a preventive solution to the problem described by studying the narratives used by terrorist organisations in order to develop effective countermeasures to prevent the processes of radicalization.

This project has already modules done. In 2018, the module responsible for scrapping news from the main newspapers and store them in Elasticsearch (Sect. 2.4.2) was developed [11]. It consists on getting the news from newspapers such as Al Jazeera, CNN or The New York Times, and then analyse them through Senpy [25] modules. The information about these news is collected in Elasticsearch, showing fields such as the full article, the author, the organisations which take part in the article, the places extracted from it, an analysis of sentiments, etc.

Our mission is to build a web and a mobile applications to visualise those data, having the possibility of filter between news and receive alerts when a new article arrives to the index in Elasticsearch. The mobile application will offer us more facility to access to the news, meanwhile the web application will give us the possibility of receive alerts.

To make this possible, we will use modern technologies for user interfaces, the best search engines and one of the most used technologies for server-side applications.

1.2 Project goals

When we thought about this project, we got a clear conclusion that was the main goal in this project is to design a React (Sect. 2.3.1) interface which is connected with Elasticsearch. The research Group of Intelligent Systems (GSI) at Universidad Politécnica de Madrid (UPM) have developed several applications that use Elasticsearch with Polymer components [23], so we are taking a step over and introducing React. Main goals of this project are listed below:

- Design an interface to interact with news, having the possibility of receive alerts or filter between news.

- Build a server and a database which can store data related with users, helping them to manage their own alerts.
- Provide access to an index from a React application where we can find all the news related with jihadists.
- Build a mobile application connected to Elasticsearch in order to provide news monitoring

1.3 Structure of this document

In this section, a brief overview is provided of the chapters included in this document. The structure is as follows:
Chapter 1 explains an introduction, an overview to the project and the goals to achieve
Chapter 2 describes the technologies used in this project, from visualisation system to management of data
Chapter 3 presents the architecture of the project. It is divided into different modules which compose the whole application
Chapter 4 explains how the visualisation module is designed and the components it is made of.
Chapter 5 discusses the conclusions drawn from this project, problems faced and suggestions for a future work.

CHAPTER 2

Enabling Technologies

2.1 Introduction

This chapter describes the technologies used in this project. We have decided to use the ones which better suit the project. In this case, two applications are built: a web application and a mobile application, so the best option is to use React and React Native.

There are a lot of options to develop a website and those which are currently more used are chosen, and also those that will be important in the near future web development. Different options between Javascript frameworks, such as Angular [18] or React, have been considered, but finally, but finally we have selected React because it is the best option to migrate the web application to the mobile application.

Firstly, the chosen technology for visual aspects will be explained in detail, in this case React. Below are the technologies used for data storage, where Elasticsearch will be the one which give us a fast and easy data search engine, and MongoDB (Sect. 2.4.1.1) and ExpressJS (Sect. 2.4.1.3) will provide us user's data storage.

Finally, the technologies used for building the Android and iOS application will be described.

2.2 Execution environment

This section details the global technologies used on this project, in both client and server side. NodeJS (Sect. 2.2.1) will provide us an environment to make a server-side development for our application, and NPM (Sect. 2.2.2) will manage all the libraries needed in the project.

2.2.1 NodeJS

NodeJS [26] is a free code Javascript[1] run-time environment. NodeJS uses V8 Chrome engine to execute our Javascript code. It can generate dynamic page content and it can add, modify or delete data from our database. NodeJS lets developers use JavaScript to write Command Line tools and for server-side scripting.

2.2.2 NPM

NPM [1] is a Javascript package manager which contains reusable components and libraries designed for other communities. It allows different payment options, but also a free one, which has all the free-code components and libraries. NPM controls the whole dependencies of the application, and saves their definitions in the file called *package.json*. Those dependencies are installed in a local folder *node_modules*.

NPM will give us the components we are going to use to build the application, so our work is reduced to search those components which fit our design.

Some of the NPM orders in the console are:

- **npm create-react-app myapp:** It creates a React application from scratch. It makes a folder with several dependencies and the architecture of React.
- **npm install package –save:** It creates the package we desire. For instance, one of the packages chosen in this application is bootstrap.
- **npm start:** This command runs the application in a local server.

In this project, NPM is used to install React components from different sources, such as Google Maps or ChartJS, and also to install packages for managing tools such as mongoose for MongoDB connections, or ExpressJS for back-end routing.

[1] https://nodejs.org/es/docs/

2.3 Data visualization

React is the responsible for displaying data. It is a Javascript framework which provides us some free code libraries with components.

2.3.1 React

React [14] is a Javascript library, owned by Facebook and launched in 2013, for build user interfaces. One of its essential features is that it helps us to build applications which data is permanently changing.

React is based on components which have their own state. Those components exchange data with others easily to build more complex applications. Data flow only goes in one direction, from parent components to child components (one-way data-binding), meanwhile events go in the opposite way.

Once the component is created, the only thing the developer has to do is to declare it in its parent and gives it the data he wants. He does not have to worry about telling it what to do with them, because it has its own logic.

React manages the user interface in one page, which has the whole logic and components (Single page application SPA), and takes part from model view controller standard (MVC). It is compatible with the rest technologies, and React components can be built from others created before with other technologies, such as Polymer web components.

In the last recent years, the progress of this framework is very striking. Let's take a look in the recent interest in different Javascript frameworks:

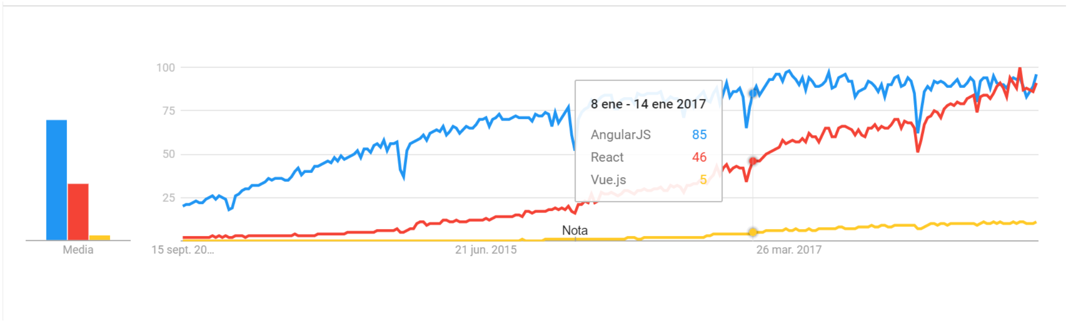


Figure 2.1: Frameworks interest in the last 5 years

In the chart obtained from Google trends [16], we can see that React has an exponential growing trend, meanwhile Angular and VueJS [24] are starting to decrease. One of the reasons why React can be the perfect framework for the closest future.

2.3.2 Redux

Redux [3] is a predictable state container for Javascript applications. It helps us to build applications that behave consistently, run in different environments and are easy to test. Its main function is to extract the global state of a web application from visualization part. It can be used with different frameworks such as Angular, VueJS or React.

Redux converts the process of modification state in an easier way. It builds a tree in which the only way to modify the state is by emitting an action. Reducers are just pure functions that take the previous state and an action, and return the next state.

Redux uses the Flux [5] architecture, which consists on dispatcher/s, stores and views that have assigned different functions, making a tree in which data flows only in one direction.

2.3.3 ReactiveSearch

ReactiveSearch [2] is a React and React native (Sect. 2.5) component's library. It is designed to work with Elasticsearch indices, so result components are updated by search components following the search criteria. It can be installed through NPM package manager. ReactiveSearch has the possibility of managing HTTP requests to the database. The only thing we have to do in our application is to implement components and to inject them an Elasticsearch index. Our own components can also be created based on those already existing. Should we wish make our own requests, they can be modified and then send them to the database.

Finally, some of the components used in the project are detailed:

- **Search elements:** These elements make HTTP requests to the Elasticsearch index and they update the results. Elements that belong to this group are text search boxes, multiple selectors or calendars to filter by dates.
- **Result components:** They are updated by the filter chosen in the search elements. They show fields of data of the index.
- **Custom components:** These components are built from scratch. They inherit the properties from Reactivebase and can be updated when other components change

their aggregations. They also allow us to make queries to the Elasticsearch index. Custom components can be both: result and search elements.

2.3.4 React modules for displaying data

This section details some modules used in the project for displaying data. We have selected those which fit the application according to news data. These modules have been developed for using them in React. They are installed through NPM package manager.

- **Google Maps React:** [15] It allows us to render any component inside a Google Map. We have to specify the longitude and latitude for each component, and it will be rendered there.
- **Material-UI:** [28] Material-UI provides us Material design components built in React. It has UI components for data visualization and also its own theme.
- **ChartJS:** [6] This package provides us graphics or charts built in React in order to display statistics of data.

2.4 Data storage

The process followed during data storage is explained in this section. It consists on two parts. The first part is the data storage for users authentication, meanwhile the second one is to get data from different sources to store them in the Elasticsearch index. First of all, let's introduce the technologies used, and then, the process will be described.

2.4.1 Authentication storage: MongoDB and ExpressJS

The process of authentication requires the storage of some data related with user properties, such as usernames, passwords, etc. To make this possible, we are going to use the MERN stack (MongoDB, ExpressJS, React and NodeJS).

2.4.1.1 MongoDB

MongoDB [7] is a NoSql database system, which store data based on JSON (Javascript Object Notation) structures. It offers a great scalability but also flexibility. The main purpose of using this technology is the use of Javascript code in the whole application, from front-end to back-end, so it will make easier to develop the tool.

MongoDB is free code database, so we can use it. Some of the features[2] it has are the following:

1. High availability
2. Horizontal scalability
3. Internal document validation
4. Monitoring, automation and backup tools
5. Comprehensive security
6. Permanent assistance

The following example [21] shows an object to be stored in MongoDB databases:

JSON object to be stored in MongoDB:

```
{
  name: ’Kate Monster’,
  ssn: ’123-456-7890’,
  addresses : [
     { street: ’123 Sesame St’, city: ’Anytown’, cc: ’USA’ },
     { street: ’123 Avenue Q’, city: ’New York’, cc: ’USA’ }
  ]
}
```

As we can see, this is a Javascript object, so we have the possibility of access to all the properties the object has. Another option we have is to group objects making hierarchy. To explain this, let’s take a look to the previous example, and we could see the object has another object (called ’adresses’), with its properties inside it.

2.4.1.2 Mongoose

Mongoose [19] provides a straight-forward[3], schema-based solution to model your application data. It includes built-in type casting, validation, query building, business logic hooks and more, out of the box.

[2]https://www.mongodb.com/what-is-mongodb?lang=es-es

[3]https://mongoosejs.com/

Mongoose helps us with the connection between NodeJS server and MongoDB database. It is a library that we can install through NPM package manager in our server application.

2.4.1.3 ExpressJS

ExpressJS [20] is a fast, unopinionated, minimalist web framework for NodeJS [4]. It is a free code library under the MIT License. It provides us a set of tools for developing webs and native applications. It is very easy to connect it to MongoDB database, because the only thing we have to do is to import the mongoose object modeling tool, and make APIs to send and receive data.

We can use it to make a lot of APIs, which can be accessed from different front-end web applications. In this case, we are going to use React, but for instance, we can also use Angular and call those APIs by the same method we do in React.

ExpressJS is a framework which give us the possibility to make URL routing (Get, Put, Post, etc). We make APIs which are being called by one or many applications. Routing refers to how an application's endpoints (URIs) respond to client requests. This routing responds with a callback function, which is invoked when we called a route.

In the following example [13], we can see the function called when we make a HTTP GET request to the route '/' of the application:

Routing example for GET method:

```
var express = require('express')
var app = express()

// respond with "hello world" when a GET request is made to the
   homepage
app.get('/', function (req, res) {
  res.send('hello world')
})
```

When we make a HTTP GET request to the route '/', ExpressJS will return us "hello world".

[4]https://expressjs.com/

2.4.2 News storage: Elasticsearch

Elasticsearch [17] is a distributed real-time search and analytics engine. It is a full-text search engine with an HTTP web interface and schema-free JSON documents. Its top features are:

- **Speed:** Make searches and obtain results is very fast. Elasticsearch is implemented by state traducers for full-text querying. It works slicing the index in small blocks, in order to optimize the searches.
- **Scalability:** Elasticsearch can be deployed on a personal computer or hundreds of servers. This fact allows developers to scale their business easily.
- **Flexibility:** We can manage a lot of types of data in the indices.

2.4.2.1 Adding data

Adding data to Elasticsearch engine can be done by multiple ways. We have made tests with Logstash.

Logstash [27] is a server-side pipeline, which ingest data from multiple sources, and send them to an Elasticsearch server, which is the responsible of storing them. We have proved it ingesting real-time data from Twitter. To make it possible, we need a configuration file like this [10]:

Logstash example for catching real-time tweets:

```
input {
  twitter {
    consumer_key => "lZ8ZIFoTriCwMs4vjvGaTQ"
    consumer_secret => "rSgEQyhYWyeXZOW1udog593rAxUOvDIycymOjZrQCo"
    oauth_token => "377381275-AfMBAQYZbZMnAjxEVH5an8NuiBrtxQegfLyaBjM7"
    oauth_token_secret => "ZL9anQWQx6I0adcp8i5SHd5YuxEbcefD9Bp8XWgNhCDo8"
    keywords => ["elasticsearch", "kibana", "react"]
    full_tweet => true
    ignore_retweets => true
  }
}

filter {
}

output {
  elasticsearch {
    hosts => ["localhost:9200"]
    index => "elastic"
    document_type => "tweet"
  }

}
```

Analyzing the file, we can observe it obtains tweets which contain the following words: elasticsearch, kibana o react. Furthermore, we need credentials from Twitter which allow us to obtain data using its API.

2.4.3 GSI crawler

GSI Crawler [8] is an innovative and useful framework which aims to extract information from web pages enriching following semantic approaches. There are a lot of sources included into GSI crawler. We can divided them into three categories: social networks (Twitter and Facebook), jihadist propaganda magazines such as Rumiyah and Dabiq, and finally news from different sources such as CNN, AlJazeera or The New York Times.

The process of storing data in GSI crawler Elasticsearch index consists on three phases:

- **Data ingestion:** This is the first part of the process. It consists on extracting data from the multiple sources described above. It works thanks to the use of web and PDF scrapers.
- **Data analysis:** It consists on the analysis of the data we have ingested from the different sources. This analysis is done by multiple modules which can be concatenated forming pipelines. This modules are Senpy plugins.
- **Data storage:** Once data is analyzed, the process finish with the storage of them. Finally, we have made different objects from multiple sources, but they have the same structure as JSON objects. We can make three groups of objects, the same we have detailed before. So the best way to make a fast and scalable engine to search that data is Elasticsearch.

To make the whole process from data ingestion to data storage possible, we need a pipeline. The orchestration is done through the Luigi framework, which allows us to build a sequence of tasks in the desired pipeline form.

The figure below explain in detail all the process followed:

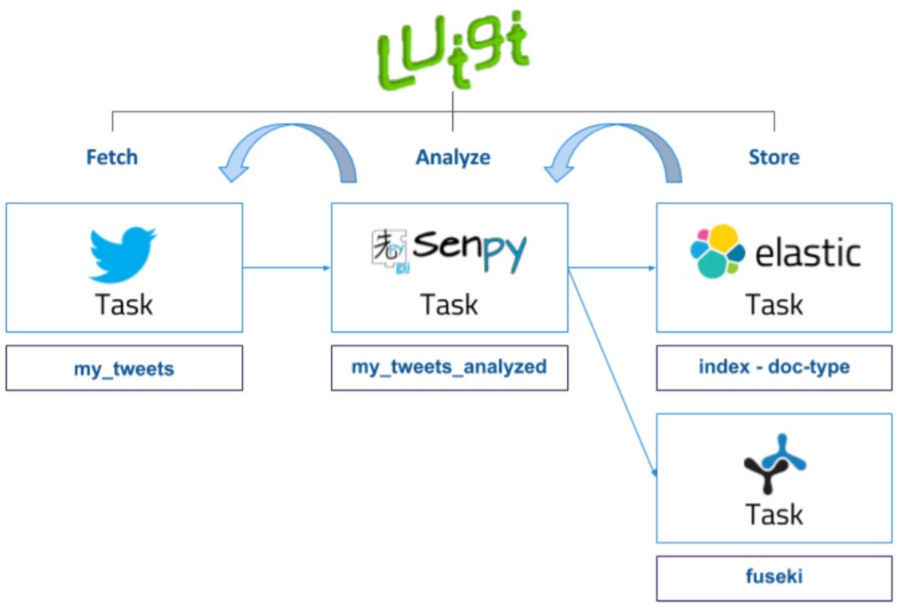


Figure 2.2: GSI Crawler pipeline structure

Luigi [5] is a package that helps us to build complex pipelines of batch jobs. It handles dependency resolution, workflow management, visualization, handling failures, command line integration, and much more. It allows us to make the process possible.

Fetch refers to the part of obtaining data (Twitter data in the figure), but also magazines or news data. Senpy plugin is the responsible of doing all the analysis to those data. And finally, we store them in an Elasticsearch index.

2.5 React Native

One of the main features of React consists on the facilities offered to migrate the web application to Android or iOS. To make this possible, there is a framework for building native applications from React called React Native.

React Native [22] lets us to build mobile applications using only Javascript code, using the same design as React, letting us compose a mobile UI from declarative components.

React Native uses the same fundamental UI building blocks as iOS or Android applications. The only difference between this and native mobile applications built in Java or

[5]https://github.com/spotify/luigi

Swift is the use of Javascript to connect those blocks. If we need to use components made in other languages, we can do so. React Native combines smoothly with components written in Objective-C, Java, or Swift. It's simple to drop down to native code if we need to optimize a few aspects of our application.

If we think about making a native application using Javascript, we realised that we can not use HTML elements or CSS styles. To replace HTML elements, React Native has elements such as Text, Image, View, etc. which are native elements. React Native also provides APIs to access native functionalities such as *CameraRoll* to access photo gallery, *Alert* to access alert dialogs, etc.

With regards to styles, React Native has a prop named style and an API to apply styles to components called StyleSheet. All the components accept that prop. The style names and values usually match with CSS styles, changing little details such as the use of capital letters (fontSize instead of font-size). React Native also allows us to create cascading styles, similar to CSS, through the use of style arrays.

To make our application adaptable to different screen sizes, React Native has also a dynamic system. Flex expands the components in the available space they have, or their parents delimit. Flex disposes the application like a grid, allowing us to place components horizontally or vertically, and dividing space proportionally or not between elements.

As regards to connect mobile application and Elasticsearch, ReactiveSearch has also components for React Native as described before.

To start building our mobile application, an initial template has been created through NPM and 'expo' help. Expo [6] is a free and open source toolchain built around React Native to help you build native iOS and Android projects using JavaScript and React.

Expo helps us to launched the application in a server. To display it in our devices, we have to read a QR code. Then, we have also the possibility to debug it.

[6]https://expo.io/

CHAPTER 3

Architecture

3.1 Introduction

In this chapter, the architecture of this project is detailed, from the design to the implementation phase. First of all, in the overview we are going to make an explanation of the main modules of the application, identifying both the server and client sides, and also the data storage. Then, we will detail each module explaining its purpose in this project.

3.1.1 Context

Our goal is to build two applications to visualise the data obtained from the analysis of the news and stored in Elasticsearch.

On the one hand, the web application includes a custom alert system for each user, so we need to store data related with users. To make this possible, the technologies more useful will be used: NodeJS, ExpressJS and MongoDB written in Javascript which allow us to better manage the objects passed from the React client to the server and the database. The second one is the mobile application, created to better and faster access to the articles, and visualise the analysis.

With regards to the web application architecture, it is more complex than the architecture of the mobile application because it includes the alerts system. This architecture consists on three main blocks. Firstly, the React client is the core of this architecture. Secondly, we find the authentication server responsible for receiving user data and storing them in the MongoDB database. Finally, the MongoDB database is found, which stores user data.

On the other hand, the mobile application architecture consists on the React Native client. This will provide us the possibility to obtain data from Elasticsearch through ReactiveSearch and then visualise them.

This two clients make up this project, and it can be visualised in Fig. 3.1 in the global module 'visualisation server'.

3.2 Overview

In this section we will present the global architecture of the project, defining the global modules that take part in this project.

The following modules can be identified and will be explained later:

- ***Visualisation module (Sect. 3.3):*** This is the main part of the project. It consists in the visualisation of the data stored in Elasticsearch. To make it possible, we have used React framework, and also one module that helps us to make queries to the index, which is ReactiveSearch.
- ***Server authentication (Sect. 3.4):*** This module refers to user data storage, needed for making different queries according to user requirements. We manage it by two technologies: ***ExpressJS*** to make the server and to make HTTP requests and responses, and ***MongoDB*** to store all these data.
- ***GSI Crawler (Sect. 3.5):*** This is the module which stores all the data related with the news.
- ***Mobile application (Sect. 3.6):*** This module is also a visualisation module. Its main function is to display data obtained from the index in mobile applications built in React Native.

In Fig. 3.1 (global application architecture), the different systems listed before are included, and they will be explained in detail later. The main system is the web application, which manages server and Elasticsearch connections, and displays data collected from them. Visualisation server includes all the work related to this project: two visualisation applications (mobile and web) and a server and database for managing user data.

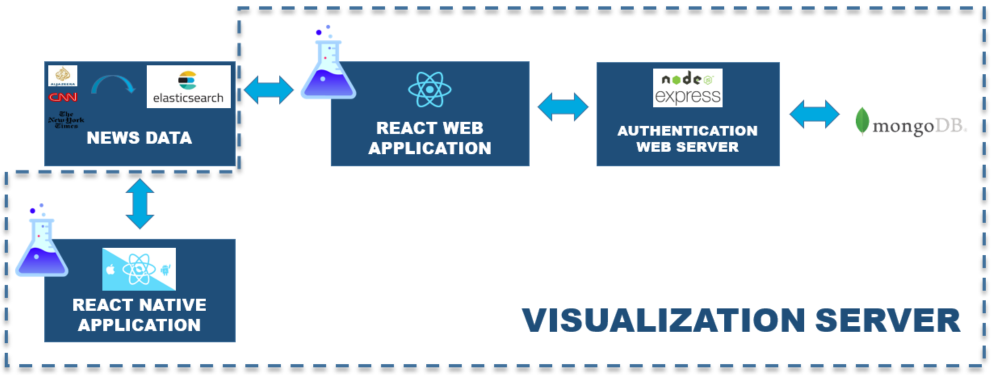


Figure 3.1: Global architecture

In the following sections, the subsystems which composed the application are described.

3.3 Visualisation module

In this section we are going to detail the union between all the technologies used in the visualisation system.

To visualise different use cases offered in the web application, we have made an UML (Unified Modeling Language) diagram in *draw.io* [1] application:

[1] https://www.draw.io/

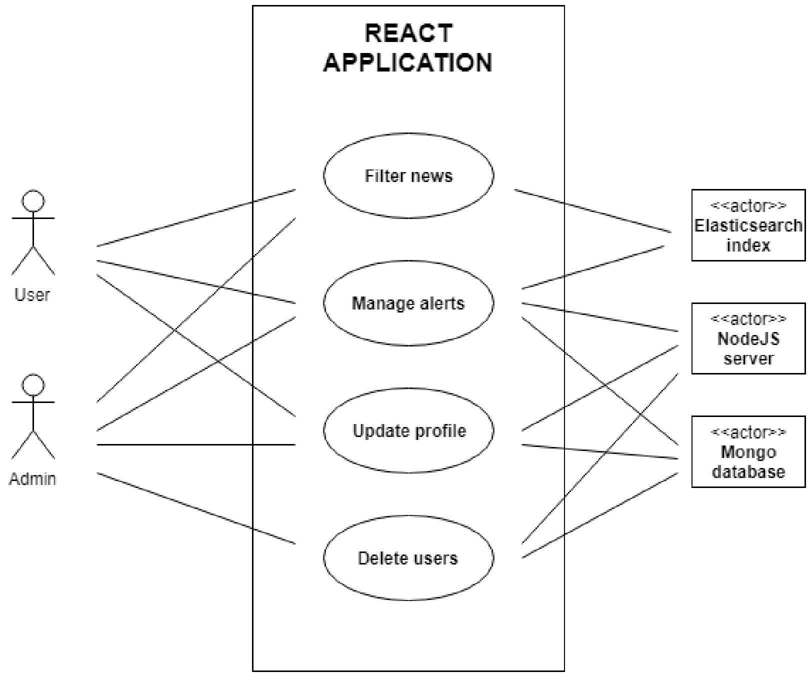


Figure 3.2: Use cases

There are two types of users: admin and normal users. The only difference between them is that the admin has the possibility of deleting users who are making a bad use of the application. All of them can: filter news through the elements built, manage alerts choosing their favourite topics or update their profiles.

The main technology used in the visualisation module is React, the core of this project. React controls connections between Elasticsearch and application, and collect data from the index to display it in the website.

First of all, the core of the project is explained in detail. Then, sub-modules which composed the whole application will be described. The main sub-module is the one which help us to connect the web application to the Elasticsearch index.

Let´s introduce React explaining its main features and specific peculiarities.

3.3.1 React: the core

We have made a React application from scratch and then we have implemented different modules to our application through NPM package manager. As mentioned, React makes the core of the application, so our code is mainly written in Javascript. But React offers us the possibility of using JSX syntax, which has some differences with classic JS syntax.

JSX is a Javascript extension built by Facebook in order to better development of React applications. React bases its development in the creation of components, so what JSX pretends to do is to help us to create those components. Let's going to see how changes JSX with regard to JS in the following example:

JSX syntax:

```
const name = 'Josh Perez';
const element = (
  <div>
    <h1>Hello! {name}</h1>
    <h2>Good to see you here.</h2>
  </div>
);
```

JS syntax:

```
'use strict';
var name = 'Josh Perez';
var element = React.createElement(
  'div',
  null,
  React.createElement(
    'h1',
    null,
    'Hello! ',
    name
  ),
  React.createElement(
    'h2',
    null,
    'Good to see you here.'
  )
);
```

We have seen the difference between both. Meanwhile React offers the possibility of using Javascript expressions inside JSX by wrapping it in curly braces, Javascript requires a more difficult to understand and longer syntax.

We have made the different components in multiple files, so we can use them as many times as we want. The main component is App, which will be rendered once the application is started. Each component is responsible for doing something and rendering HTML expressions.

3.3.2 ReactiveSearch

The module responsible for Elasticsearch querying will be explained in this section. ReactiveSearch is the union between React and data stored in Elasticsearch. If offers us components to make queries to the index and also the possibility to make our own components, inheriting all the properties from ReactiveSearch.

The figure below shows the connection between Elasticsearch and React module ReactiveSearch:

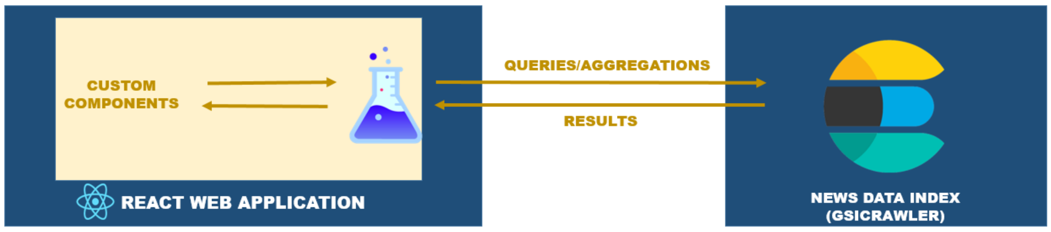


Figure 3.3: Obtaining data

As figure shows, ReactiveSearch makes the union between Elasticsearch and React possible. It collects data from the index to display them in the application. There are two options to display those data: using constituents provided by ReactiveSearch or making our own elements which consume data from the GSICrawler index through ReactiveSearch.

Data flows in only one direction from the application to the GSICrawler index. Meanwhile queries and aggregations flow in the opposite sense.

The main component in ReactiveSearch is the one that manages everything: ReactiveBase. In this component, we can specify multiple factors. The connection to the Elasticsearch index is stated in the 'app' and the 'url' properties. We can also indicate the Google Maps key and the theme all of the ReactiveSearch components are going to use.

As stated, elements of ReactiveSearch have their properties, which we specify when we implement them. There are common properties and specific features for each component. Here are some of the common properties:

- **componentId:** This is the component's identifier, which must be unique. We can refer to that component in another through it.

- **defaultSelected:** This parameter contains the pre-selected value in the filter.

- **dataField:** In this parameter we configure the Elasticsearch field which we want to get its values for filtering. This is probably the most important parameter because we are defining each filter by its field.

- **react:** This parameter joins all the queries that different components make and specify dependent components to reactively update component's options. You can configure each component combining clause. If one of them is updated, you can configure if you want it to be updated or not. It admits three possibilities: and (if you want to update results by all of the associated components), or (if you want to update results by at least one of the associated components states) and not (updated by an inverse match of the associated component states).

- **onValueChange:** In this property we specify which function will be called when we change the value selected.

- **customQuery:** If we want to make specific queries that contains the value selected, we must define this parameter. A query that the function returns is called when we change the value selected. Here is a sample custom query we have made:

```
customQuery={
  function(value, props) {
    return {
      match: {
        '@type': "schema:NewsArticle"
      }
    }
  }
```

  This customQuery is a function that returns a query that matches all the hits whose @type property is "schema:Article". We have made this for getting all the news hits, because the index has hits of three categories: news, social networks and magazines. So, to resume, we have filtered the index, which is the main feature of Elasticsearch engine.

3.3.2.1 Custom components in ReactiveSearch

ReactiveSearch has its own elements such as single lists, calendar pickers, range items, etc. As mentioned, we can make our own components for ReactiveSearch. The union between components from other libraries and ReactiveSearch will be explained below.

To implement custom components in ReactiveSearch, we must specify inside the ReactiveBase component, a ReactiveComponent. Props related with ReactiveSearch will be inherited by the components inside the ReactiveComponent, so we have aggregations and hits. Inside it, we can introduce any type of React components, and we access the props as we usually do. To make queries from this components, we have to call a props function (this.props.setQuery). It will update the hits that Elasticsearch returns and also the components whose have the react property referring to this component, as we mentioned in common properties.

The following code shows a ReactiveSearch custom component we have made:

```
    //Updating the query
    setValue(value) {
  this.props.setQuery({
    query: {
      term: {
        'entities.schema:name.keyword': value,
      },
    },
    value,
  });
      }
      //Getting props and building maps
      render() {
          let hits = this.props.hits;
          let allLocations = [];
          if(this.props.hits.length > 0){
            for(let i=0; i<hits.length; i++){
              let entities = hits[i]._source.entities;
              let locations = entities.filter(entitie => entitie['@type']
                  === "schema:Place");
              for(let j=0; j<locations.length; j++){
                allLocations.push(locations[j]);
              }

            }
            return   <MapaComponent locationClick={this.setValue} data={
                allLocations} />;
          }
          return null;
        }
```

In the first part of the code, we have made the function explained before, for updating the Elasticsearch index. In the second part, we have got the props from ReactiveSearch, and we have filtered the locations between all the hits. We have built an array with those locations, and then we have rendered the component 'MapaComponent' sending them two props: data (which contains the locations array) and locationClick (which has the function to be updated when we select a location in the map).

3.3.3 React dependencies

We can make applications in React creating our own components, but it would be a hard work to build map or chart components from scratch. If we want to use external elements from different libraries, we have to install them through NPM package manager. Once installed them, libraries are stored in node_modules folder. Libraries for assisting us with connections or data flowing are also stated here.

We can see dependencies used on this project in package.json file. The following code shows this file:

```
{
    "name": "aplicacion",
    "version": "0.1.0",
    "private": true,
    "dependencies": {
      "@appbaseio/reactivesearch": "^2.12.0",
      "@material-ui/core": "^3.2.0",
      "@material-ui/icons": "^3.0.1",
      "chart.js": "^2.7.3",
      "google-map-react": "^1.0.9",
      "history": "^4.7.2",
      "react": "^16.5.1",
      "react-bootstrap": "^0.32.4",
      "react-chartjs-2": "^2.7.4",
      "react-dom": "^16.5.1",
      "react-redux": "^5.1.0",
      "react-router-dom": "^4.3.1",
      "react-scripts": "1.1.5",
      "redux-logger": "^3.0.6",
      "redux-thunk": "^2.3.0"
    },
    "scripts": {
      "start": "react-scripts start",
      "build": "react-scripts build",
      "test": "react-scripts test --env=jsdom",
      "eject": "react-scripts eject"
    }
  }
```

The file contains the dependencies used in this project and scripts to be executed through NPM package manager. Both are explained below:

- **Dependencies**: External libraries for providing us components or tools. Version appears in the right side.
  - @appbaseio/reactivesearch: It contains ReactiveSearch dependency.
  - @material-ui: Dependency for material design components.
  - chart.js and react-chartjs-2: Definitions for chart components.
  - google-map-react: Definitions for chart components.
  - react-redux, redux-logger and redux-thunk: Contain definitions for redux.
- **Scripts**: Here it is the relation between NPM and react-scripts commands. For instance, we can launch *npm start* that corresponds to *react-scripts start* to start the application in the localhost.

3.3.3.1 Google Maps for React

To develop APIs which obtain data from Google, the first thing we have to do is to register us in Google in order to obtain credentials to access Google APIs. Once we have credentials, we can setting them in the ReactiveBase component.

As result of analysing news, we have in JSON fields the locations which appear in each new. We have created a Material design component on each location, with the objective of showing the count of news in which the location is involved.

When you click on each location, a function that updates Elasticsearch query will be launched, as explained before in custom components.

3.3.3.2 ChartJS for React

As mentioned in 'Enabling technologies' chapter (Sect. 2.3.4), ChartJS allows us to make charts. This library is written in Javascript, but a community (react-chart-js-2) has built a compatible library for JSX.

We have decided to use charts for showing stats of the most common locations, people or organisations of a list of news. For instance, if you are looking for CNN news, you will also know the most popular people, locations and organisations involved in these news.

When you click on each entity, a function that updates Elasticsearch query will be launched.

3.3.3.3 Material-UI

In order to create a beautiful and intuitive interface, we have chosen Material UI, because it is one of the most popular UI design systems.

Material gives us elements to make a better application. Some of the elements used to display results in the project are detailed below:

- **Expansion Panel:** To show results of news with filters applied, expansion panels are going to be used. Headline, source and organisations involved in the new are showed in the list, and if the panel is expanded, analysis will be seen.

- **Dialog:** The main purpose of this element is the same that expansion panels. Alerts results are showed in modal views.

- **Grid list:** This element gives us a different way of viewing results. Images provided by news results are listed with their headlines.

- **Chip:** Allows users to make selections or filter content. It is used to display entities or count of news related with a place.

3.3.4 Redux

As mentioned in 'Enabling technologies' chapter (Sect. 2.3.2), Redux provides us a simple way for managing global state of the application. We have not used Redux in the whole application, because there are components whose state is very small and do not depend on other components.

For instance, Google Maps component state is only used inside that element. If we need to modify or send actions to that state, we will do it inside that component.

However, we have user state. It is used by the majority of elements. So the best option to manage it is Redux.

3.4 Server authentication

The connections between React application and NodeJS server, and also between NodeJS server and MongoDB database are going to be explained below. To make this possible, we have used the most useful technologies for Javascript: ExpressJS and MongoDB.

ExpressJS offers us the chance of making APIs which can be accessed by client applications. ExpressJS is also known as Javascript server side, so we send Javascript objects from React application to ExpressJS server.

MongoDB has also a helpful feature. It can store Javascript objects in its database.

This server and this database will help us to store information related with users, for the purpose of creating custom alerts for each one. We will store some fields such as name, login, password and topics user wants to be informed.

The following figure shows the whole process of storing users data: data flows in both directions, from client to server and back.

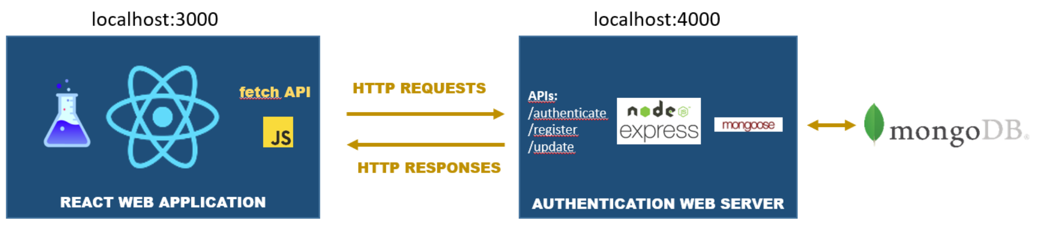


Figure 3.4: Server connections

In the image, we can see React web application makes HTTP requests (GET, PUT, POST, DELETE) to NodeJS server through Javascript fetch API. ExpressJS provides us routing to access to those APIs: for instance route *http://localhost:4000/users/authenticate* will callback a function to check a user identity.

Then, differents subsystems which composed the transport and storage of user data will be detailed. First of all, connections to NodeJS server from web application are explained. Secondly, we will take a look to the connections between server and database.

3.4.1 Connections to NodeJS server

This section explains the data transportation from client to server and back. In this project's development, two ports in localhost are being used: first port (3000) is used to run React application, and second port (4000) is responsible for running NodeJS and ExpressJS server.

ExpressJS is used to build the server. In port 4000, we have made some APIs to be accessed from React, and will be explained below:

- **users:** This is the generic route for managing user data.
- **users/register:** This API is called when a user requests for an authentication in React register's page.
- **users/authenticate:** An HTTP request is forwarded to this API when a user logs in the application.
- **users/delete:** API used for deleting users.
- **users/update:** Route for updating user. For instance, when someone decides to change his alerts to be informed.

Let's see an example of a route made with ExpressJS:

```
router.put('/:id',
  function update(req, res, next) {
      userService.update(req.params.id, req.body)
          .then(() => res.json({}))
          .catch(err => next(err));
  }
);
```

The previous code shows a route which can be accessed with HTTP method PUT. It requires a parameter, which is user ID. A function is called automatically when we invoked that PUT request. Then, the function will reply with another HTTP response to the client, completing communication process.

To access these routes from React's client, Fetch provides us an interface for fetching resources (including across the network).

3.4.2 Connections to MongoDB

Connections to MongoDB database from the server are described below. Firstly, an introduction to MongoDB DB is detailed, and then, we will describe the library which allow us to connect NodeJS server to database.

In a MongoDB database we can store Javascript objects. That is the main reason for using it in this project whose base is that language.

3.4.2.1 Mongoose

Mongoose provides us a model to store Javascript objects from server in database. It is very useful when we have the whole application built in Javascript language.

The following code shows how it works:

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const schema = new Schema({
    username: { type: String, unique: true, required: true },
    password: { type: String, required: true },
    firstName: { type: String, required: true },
    lastName: { type: String, required: true },
    createdDate: { type: Date, default: Date.now },
    topicone: { type: String, required: true },
    topictwo: { type: String, required: true }
});
schema.set('toJSON', { virtuals: true });
module.exports = mongoose.model('User', schema);
```

This code shows the model used in the project for storing users data. Mongoose is imported in the first line. Then we have made the schema with data related to user.

We can define an object properties such as username, password, topics to search, etc. Then we convert the schema to a JSON object and we export it as 'User'. In future processes, we have a Javascript object, so we can operate it in NodeJS server and also in React web application.

3.4.3 Alerts service

As mentioned, the main purpose of integrating a server and a database to this project is to manage custom alerts for each user.

Alerts inform users about fresh news. The process follows different steps:

- **Choose topics:** First of all, in register's page, user chooses topics he wants to be informed. Those topics should be related with the main issue of application. In this case, we have chosen jihadist news.
- **Visualize alerts:** Once topics are configured, we can log in the application and go to alerts page. We will see all the news related with those topics.
- **Update dates:** When user clicks on the button to update alerts, last time access date will be updated. Next time user goes to alerts page, he will see news from last date to today.
- **Change alerts:** Users can change alerts received. There is a configuration page visible for all the users where they can change topics to be informed.

Alert services contains entire cycle of data: it gets data from Elasticsearch and compare them to user data got from MongoDB. Below there is a UML (Unified Modeling Language) sequential diagram which shows the whole process:

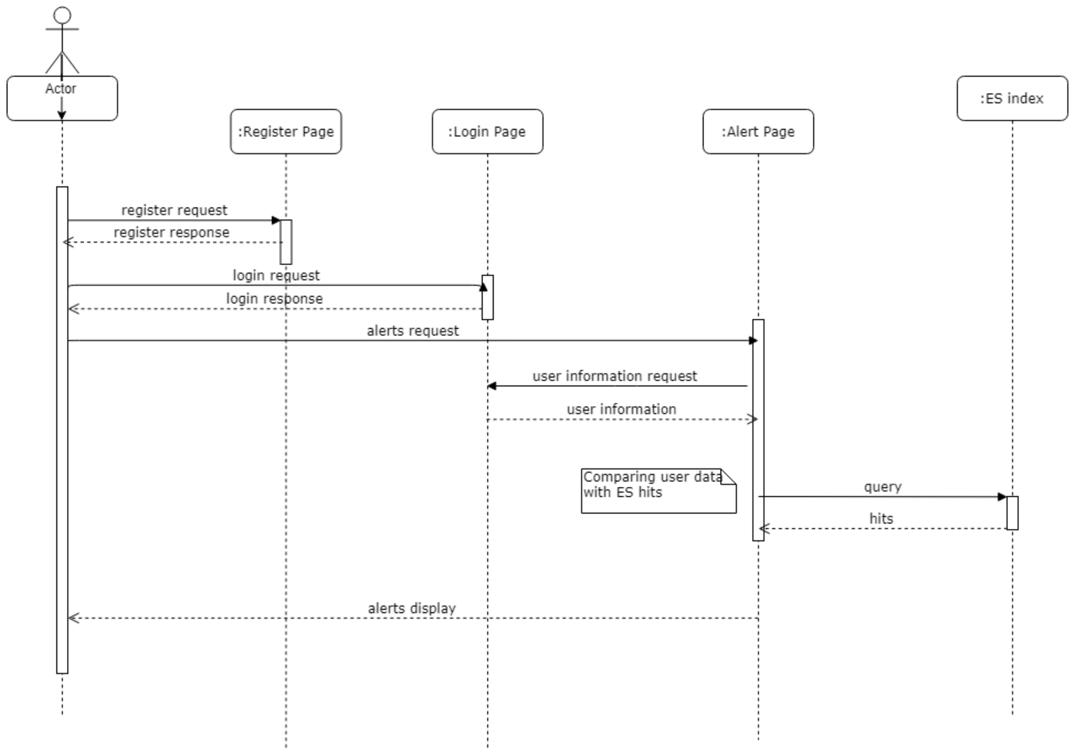


Figure 3.5: Sequential diagram

3.5 GSI crawler

As mentioned in 'Enabling technologies' chapter (Sect. 2.4.3), GSI crawler is an innovative and useful framework which aims to extract information from web pages enriching following semantic approaches.

This GSI crawler has been developed by members of GSI department of Technical University of Madrid. Our goal is making an interface in order to be able of visualizing results obtained from data analysis.

We are focusing on jihadism information from different newspapers. Firstly, we have to filter all the data obtained from Elasticsearch, extracting only news articles type. Once we got them, queries and aggregations can be launched from web application. Extracting only news articles is the main query of this project.

The following figure shows data flows inside GSI crawler, from scrapping news to storing them in Elasticsearch:

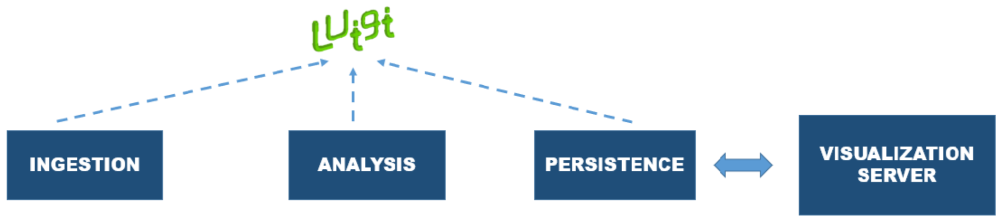


Figure 3.6: GSI crawler

The process consists of three steps: scrapping data from newspapers, analysing them through Senpy modules and finally storing them in Elasticsearch. Our goal in this project is to visualise the result of this process.

3.6 Mobile application

In this section, visualisation server module for mobile application is detailed. The core of this module is the same as web application core: Javascript as programming language, HTTP requests to Elasticsearch index and ReactiveSearch as the module which connects the application to the index.

Once we have made the web application, the migration to mobile native applications should be an easy task. As mentioned, React Native follows same structure of components as React, replacing HTML elements by native components, also those written in Swift or Java.

As explained in 'Enabling technologies' chapter (Sect. 2.5), we have started by creating a template through NPM. We have also installed some dependencies in our project such as ReactiveSearch and Material Design UI. Then, we got everything to build our application.

Firstly, connections between React Native and Elasticsearch index will be explained. In React Native we follow the same process as in React, implementing elements of ReactiveSearch which are responsible for managing connections to the index. Some of ReactiveSearch components used in this application are listed below:

- **ReactiveBase:** Same as ReactiveBase component for React application. It details connection parameters to the index and includes the rest of ReactiveSearch elements in the application.
- **Single List:** List of selectable items used for filtering results. We can make an aggregation of a field in the index and get their results to build the list.

- **DateRange:** Calendar to select range of dates to search results.
- **ReactiveList:** Customizable list of search results filtered by the rest of the filter components. An action can be launched when we click an item.
- **SelectedFilters:** List of selected filters with their values. For instance, dateRange should show a range of dates.

Secondly, different elements of Material Design and React Native used in this project are described. The application consists on two main blocks. First block includes all the filters, meanwhile second block contains search results.

To display first block, ReactiveSearch elements have been used, and also a extensible panel to show/hide filters. To visualise results, ReactiveList has been displayed, listing View React Native elements with headlines and images. When you select one View in the list, a modal will be displayed, showing the whole article and the entities that take part on it.

CHAPTER 4

Visualisation server

4.1 Introduction

The main goal of this module is to present the visual aspects of the two applications we have made. As explained before, our objective is to show and alert users of news related with jihadism. For making this possible, we have built two different applications: a website and a native application. Both extract data from Elasticsearch thanks to the components provided by ReactiveSearch.

First of all, we are going to explain the structure followed in the web application, describing each page and its function. Then, components used on each page are detailed. The main goal is to show the structure and the different components that compose the dashboard. This website is based on React components from different libraries and its main theme follows material design principles. ReactiveSearch has provided us also some interactive elements for selecting filters in order to update results obtained.

Finally, native application is explained, detailing the structure and the components used on it.

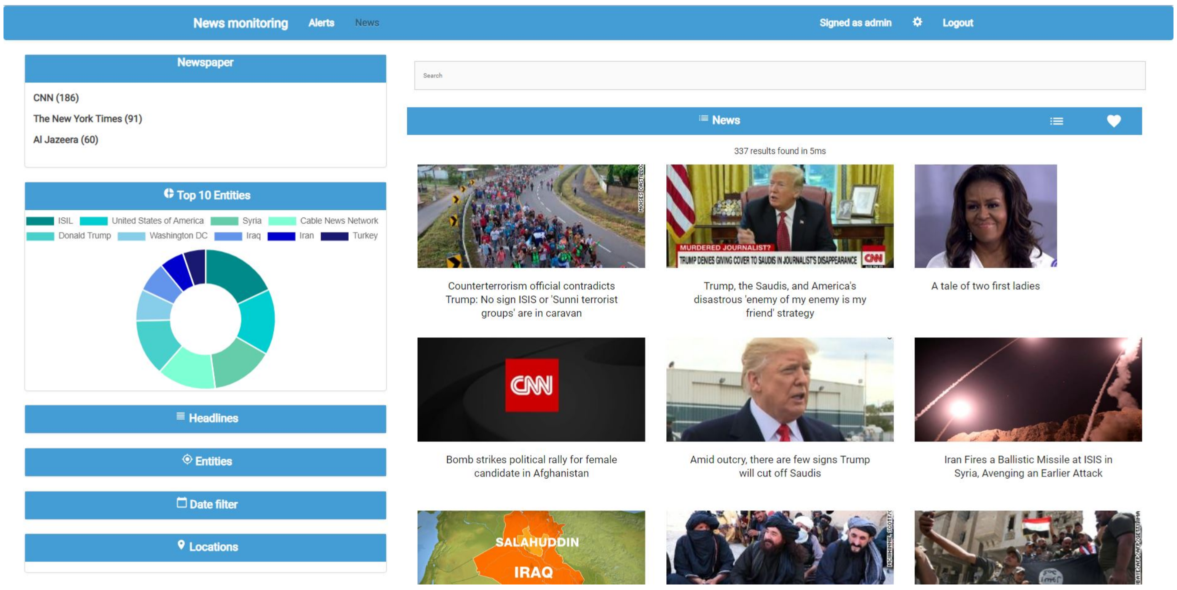
4.2 Structure

React web application consists of five pages. Three of them are used for managing users, and the function of two remaining is showing us an interface to monitoring news. The pages are listed below:

- **Login:** This is a small form which allows users to access in the system. The users must have been registered before.
- **Registration:** The function of this page is to register users in the system, requesting information about them and storing their data.
- **Configuration:** The function of this page is provide users an interface to change their profile, what allows them to select different topics to received alerts or to change their data.
- **News:** This is the main page of the application. Its function is monitoring news, offering user the chance of selecting different filters depending on he wants.
- **Alerts:** The alerts page shows users the news results related to the topics they have chosen before in registration or configuration pages.

The Fig. 4.1 is a general screenshot of the application. It shows the menu and the News page. The filters are displayed on the left side of the page meanwhile the results are listed on the right side.

The pages listed before are detailed below.



Counterterrorism official contradicts Trump: No sign ISIS or 'Sunni terrorist groups' are in caravan

Trump, the Saudis, and America's disastrous 'enemy of my enemy is my friend' strategy

A tale of two first ladies

Bomb strikes political rally for female candidate in Afghanistan

Amid outcry, there are few signs Trump will cut off Saudis

Iran Fires a Ballistic Missile at ISIS in Syria, Avenging an Earlier Attack

Figure 4.1: General view of the application

4.3 Login

Login page is the default page of the application. If a user is not registered in the system, he can not access to other page except Registration and Login pages. If he wants to access to alert routes or news routes, access will be denied and user will be redirect to Login page.

Login page consists on a small form with two fields: user and password.

4.4 Registration

In this page, user is registered in the system. It consists on a form with some obligatory fields detailed below:

- **Username:** Nickname to be used when a user try to access to the system. It must be unique.
- **Password:** Secret to authenticate users.
- **First name:** Name of user.
- **Last name:** Last name of user.
- **Topic one and topic two:** Topics user chooses to be alerted.

4.5 Configuration

Configuration form follows the same structure as registration form. The main purpose of this module is to allow users to update their topics. For instance, a user who wants to change topics related to China to United States.

In this page, the user 'admin' has the ability to remove users from the system, if they are doing a bad use of it.

4.6 News

This is the main page of the web. It is divided into two sections: the filters and the results. To make our searches faster, we have used Elasticsearch and components of ReactiveSearch.

4.6.1 Filters

Filters make up the main part of this page. There are two types of filters: proper components of ReactiveSearch and custom components made using another React libraries. All of them are connected, which means that if one of the filters is updated, the rest are updated with the new results. All the filters are listed below:

- **Top 10 entities:** This filter is a chart which shows the top 10 most popular entities which appears in the news. It is a chart from the chartJS components library adapted to work with Elasticsearch. All the sections available in the chart are clickable, and so the filters and the results are updated modifying Elasticsearch query.

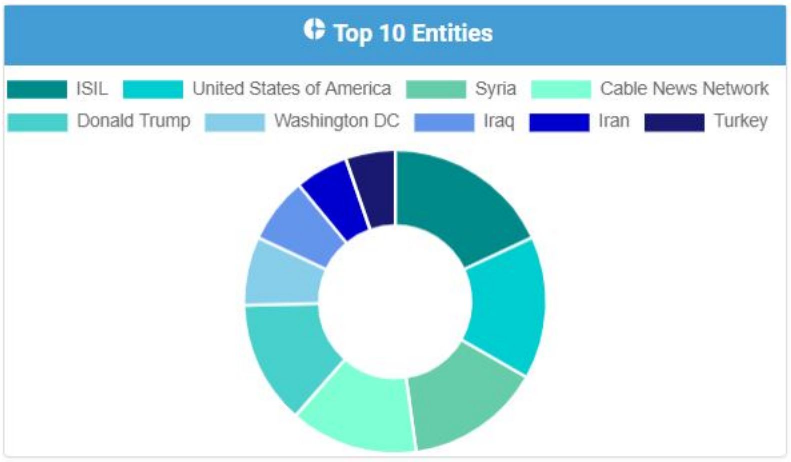


Figure 4.2: Chart filter

- **Entities:** This is also a ReactiveSearch component. It is a word cloud who shows the most popular entities that appear in the news. Each word is selectable and updates the results.

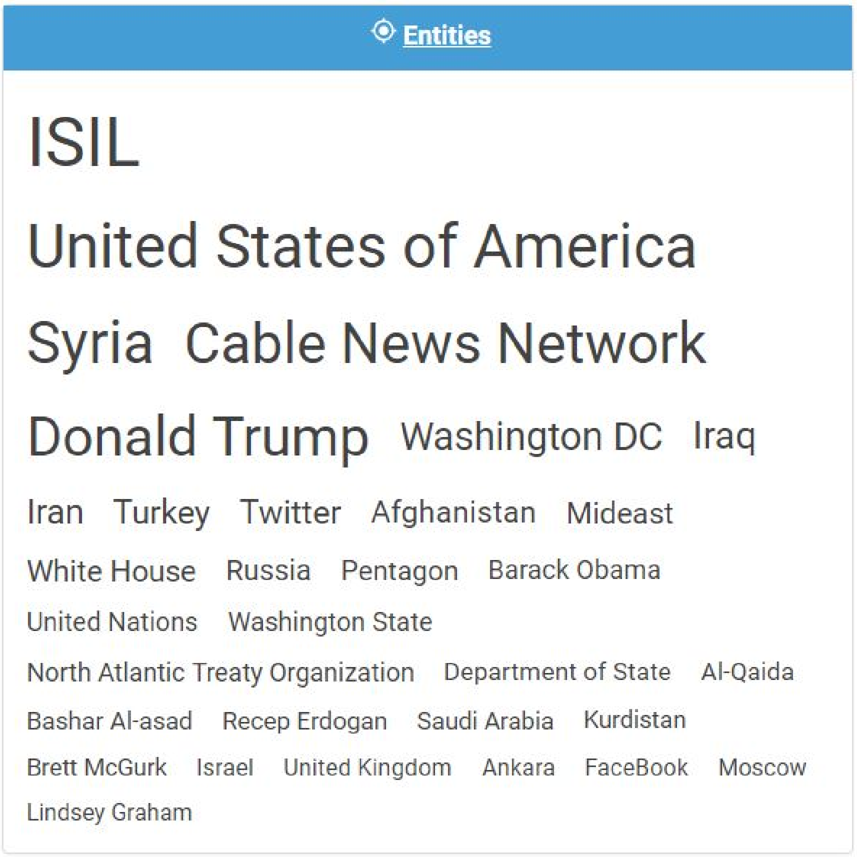


Figure 4.3: Word cloud filter

- **Source filter:** This is a ReactiveSearch component which selects news source. Depending on where they came from, there are three sources: CNN, Al Jazeera and The New York Times.

- **Headlines:** This component comes from ReactiveSearch. It has all the headlines available and a text searcher to better find the headline wanted.

- **Date filter:** It is a ReactiveSearch element which allows us to filter by date. It is a range filter which means that we can select in the calendar a range of days. News results will have been in those days.

- **Locations:** This is a Google Maps component of React. It is one of the custom components made from another library and implemented inside ReactiveSearch framework. We have developed an algorithm to obtain all the places which appear in the news, and then we have presented them in each geo-location through their latitudes and longitudes. Users can select each location to filter results.

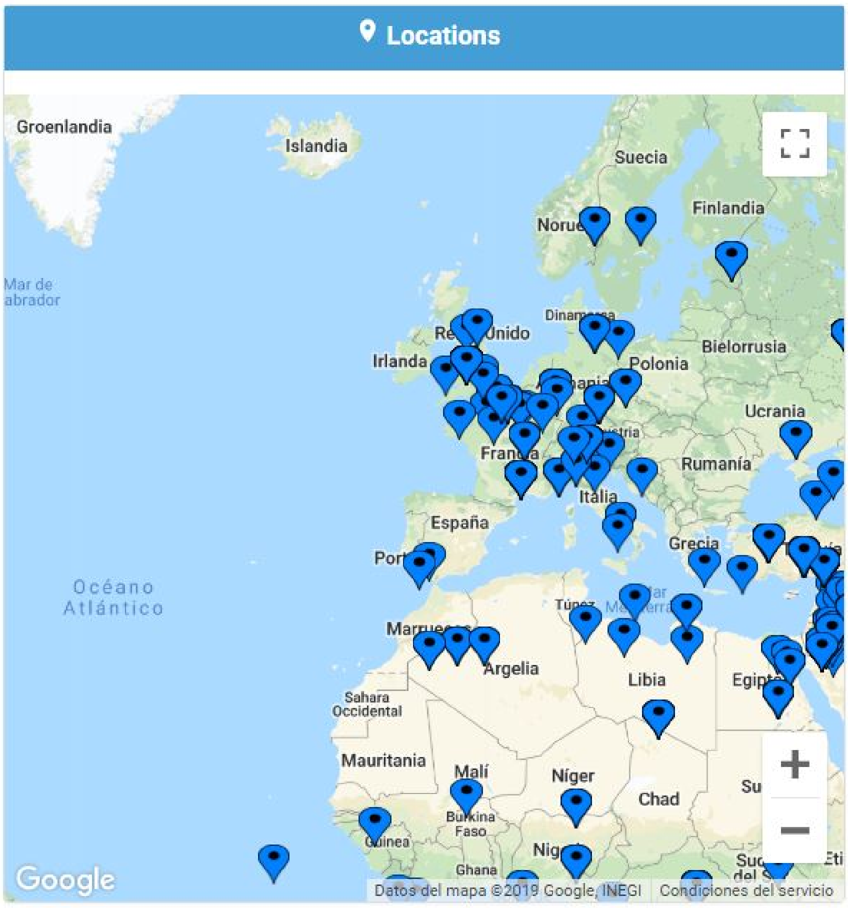


Figure 4.4: Maps filter

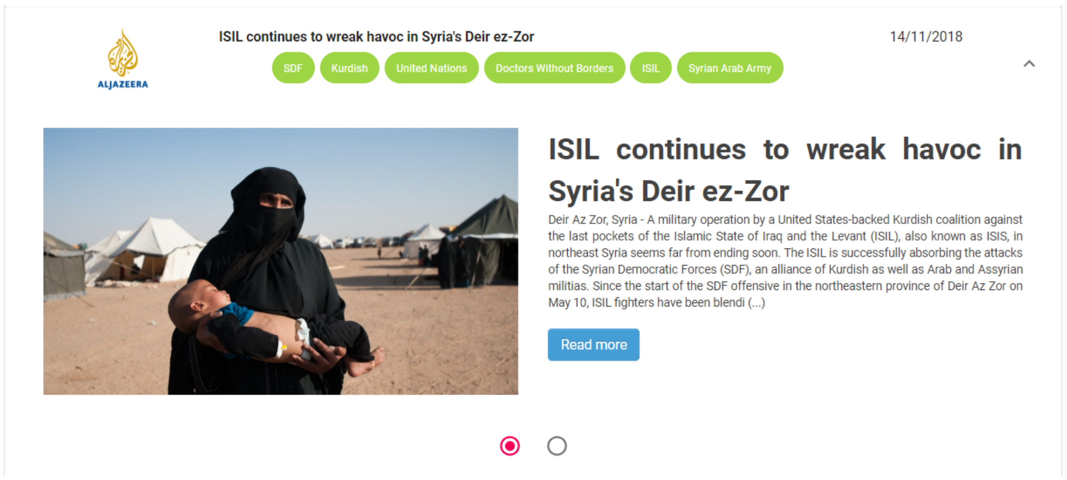
- **Selected filters:** ReactiveSearch has a component to display the selections user has made. Each filter can be customized and also clicked for doing an action. When a selected filter is deactivated, the filter is cleaned and the components are updated with the new results.

4.6.2 Results

Results are listed on the right side of the page. They are updated when a user changes his filtering parameters. There are two options to visualise results: a list of expansible panels or a list of images and headlines.

Each news item has many properties such as the article, the headline, the entities which appear in the news, the date of publication, etc. These components are going to list these news with their characteristics.

The list of expansible panels shows the headline of the article, the date of publication and organisations that appear in the news. When you expand the panel, you will see the full article and an analysis of it. The following image presents this component:



ISIL continues to wreak havoc in Syria's Deir ez-Zor

14/11/2018

ISIL continues to wreak havoc in Syria's Deir ez-Zor

Deir Az Zor, Syria - A military operation by a United States-backed Kurdish coalition against the last pockets of the Islamic State of Iraq and the Levant (ISIL), also known as ISIS, in northeast Syria seems far from ending soon. The ISIL is successfully absorbing the attacks of the Syrian Democratic Forces (SDF), an alliance of Kurdish as well as Arab and Assyrian militias. Since the start of the SDF offensive in the northeastern province of Deir Az Zor on May 10, ISIL fighters have been blendi (...)

Figure 4.5: News results

This component has also two options to do. If the second radio button is selected, an article analysis will be displayed, showing a map and a list of people that take part on the news. Furthermore, if 'read more' button is clicked, full article will be display in a dialog modal.

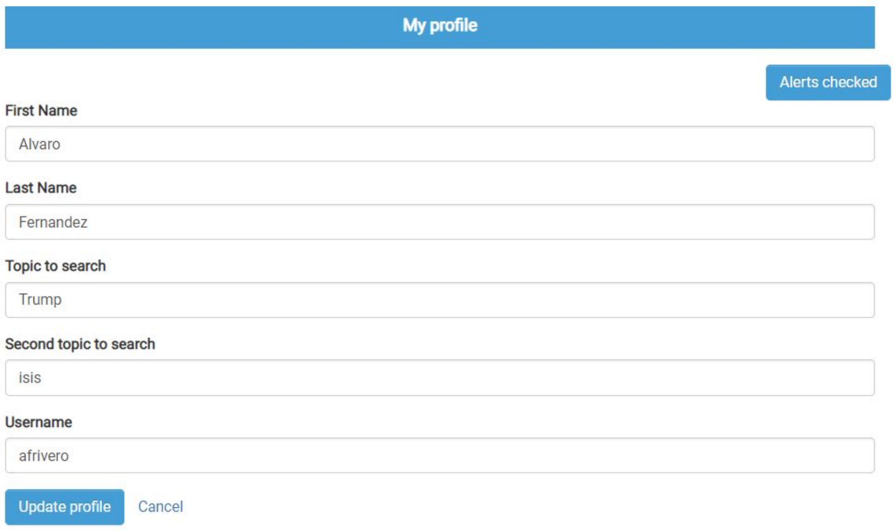
There is a second option to visualise results. This is one of the features that React gives us: the chance to create dynamic pages following the SPA (Single Page Application) criteria. In this case, news are displayed showing the headline and the main image of the article, offering users the vision of select news visualising images. When an image is selected, a modal will be displayed with information about entities and the full article.

4.7 Alerts

As previously explained, alerts represent the whole cycle of data: from obtaining users data from MongoDB database and news from Elasticsearch to compare them by displaying custom alerts for each user.

The alerts make up another page in our application. The structure of this page follows the same criteria as in news page: a block which includes some filters and the block of results. In this case, there are two blocks of results corresponding to the two topics chosen by the user in configuration page.

Let’s see an example of this process for user afrivero. This user has the following profile:



First Name: Alvaro

Last Name: Fernandez

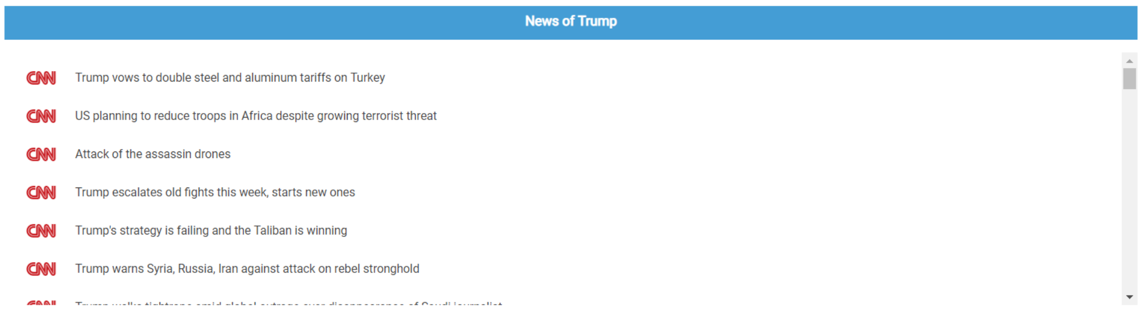
Topic to search: Trump

Second topic to search: isis

Username: afrivero

Figure 4.6: Profile of user afrivero

The user afrivero has chosen receive alerts from two topics: ’isis’ and ’Trump’. It can be updated when the user wants. In alerts page, news which ’Trump’ or ’isis’ take part will be displayed:



CNN Trump vows to double steel and aluminum tariffs on Turkey

CNN US planning to reduce troops in Africa despite growing terrorist threat

CNN Attack of the assassin drones

CNN Trump escalates old fights this week, starts new ones

CNN Trump's strategy is failing and the Taliban is winning

CNN Trump warns Syria, Russia, Iran against attack on rebel stronghold

Figure 4.7: Alerts of user afrivero

A list of news with their source are displayed. When one item is selected, a modal will be showed detailing the full article and the entities which take part on it. When the user selects the button to update alerts, last time access date will be updated automatically, so the user will not see any item until a new article gets in the index.

4.8 Mobile application

The structure of mobile application follows the same schema than News page of the website. It consists on two main blocks:

- **Filters block:** This container has all the filters used in the application. It can be expanded to show/hide the filters. When a filter is updated, the rest are also updated by the new hits obtained from the index.
- **Results block:** The results block contains all the results extracted by the queries selected in the filters by the user.

4.8.1 Filters

In order to filter between all the news stored in the index, we have implemented components from ReactiveSearch. They are listed inside an expansible block, whose function is to hide or show filters allowing users to have an interactive and intuitive experience. The different elements used can filter by:

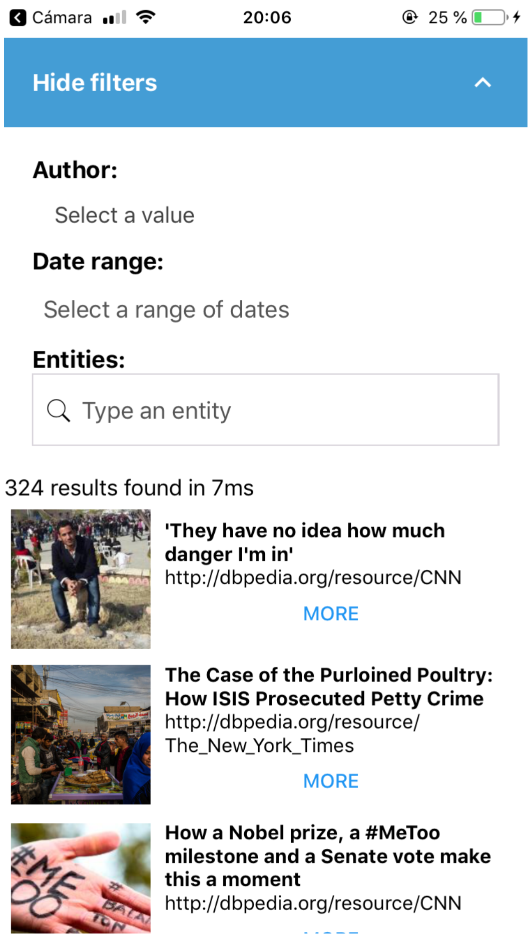
- **Date:** This component filters by selecting a range of dates in which an article is published.
- **Author:** This element has three options listed according to the three sources of articles: CNN, Al Jazeera or The New York Times.
- **Entities:** This component is a searcher to find entities which take part in the article. It filters by the entity selected.

The Fig. 4.8 illustrates the filters block.

4.8.2 Results

Results are presented in the ReactiveList component from ReactiveSearch. The items of this list are customizable, so we have defined an action on each item to display the whole article and the entities which take part on it.

The following figures illustrates both blocks, the filters and the results:



Author:

Select a value

Date range:

Select a range of dates

Entities:

Type an entity

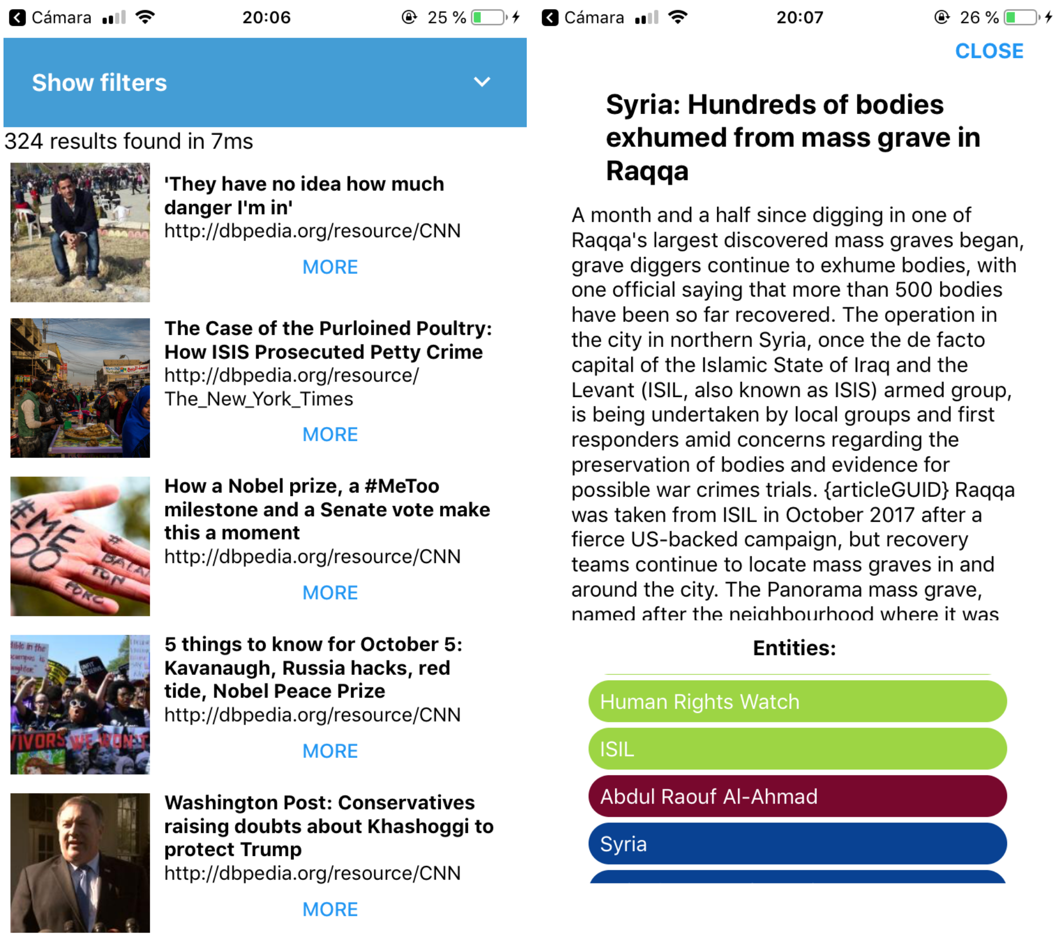
324 results found in 7ms

'They have no idea how much danger I'm in'
http://dbpedia.org/resource/CNN
MORE

The Case of the Purloined Poultry: How ISIS Prosecuted Petty Crime
http://dbpedia.org/resource/The_New_York_Times
MORE

How a Nobel prize, a #MeToo milestone and a Senate vote make this a moment
http://dbpedia.org/resource/CNN

Figure 4.8: Filters in the mobile application



324 results found in 7ms

'They have no idea how much danger I'm in'
http://dbpedia.org/resource/CNN
MORE

The Case of the Purloined Poultry: How ISIS Prosecuted Petty Crime
http://dbpedia.org/resource/The_New_York_Times
MORE

How a Nobel prize, a #MeToo milestone and a Senate vote make this a moment
http://dbpedia.org/resource/CNN
MORE

5 things to know for October 5: Kavanaugh, Russia hacks, red tide, Nobel Peace Prize
http://dbpedia.org/resource/CNN
MORE

Washington Post: Conservatives raising doubts about Khashoggi to protect Trump
http://dbpedia.org/resource/CNN
MORE

Syria: Hundreds of bodies exhumed from mass grave in Raqqa

A month and a half since digging in one of Raqqa's largest discovered mass graves began, grave diggers continue to exhume bodies, with one official saying that more than 500 bodies have been so far recovered. The operation in the city in northern Syria, once the de facto capital of the Islamic State of Iraq and the Levant (ISIL, also known as ISIS) armed group, is being undertaken by local groups and first responders amid concerns regarding the preservation of bodies and evidence for possible war crimes trials. {articleGUID} Raqqa was taken from ISIL in October 2017 after a fierce US-backed campaign, but recovery teams continue to locate mass graves in and around the city. The Panorama mass grave, named after the neighbourhood where it was

Entities:

Human Rights Watch
ISIL
Abdul Raouf Al-Ahmad
Syria

Figure 4.9: Results in the mobile application

CHAPTER 5

Conclusions and future work

5.1 Introduction

In this chapter, conclusions extracted from this project are described. Firstly, we will do an overview about conclusions extracted. Secondly, we will described problems faced. Thirdly, achievements reached during this project are detailed. Finally, we will suggest future work or improvements to do.

5.2 Conclusions

In this project we have created a dashboard for monitoring news based on an innovative framework and useful technologies to provide storage. This dashboard is made of components extracted from different libraries, which have been connected to each other. The web application provides a lot of information and a system to manage alerts.

This project is formed by different subsystems. Our goal was to build a visualisation system to data stored in the index, but inside that block there are a lot of modules. From React website to MongoDB database, through an authentication server to register and log in into the application.

The use of React has also allowed us to migrate easily the web application to mobiles. This is an important achievement because mobile phones are nowadays an important alerts system. Throughout the day, tens of notifications are received, so it has been an important step to build a little native application.

5.3 Achieved goals

In the following section, achieved goals will be detailed explaining in detail each one.

Build an innovative dashboard with leading technologies This goal has been achieved thanks to React. As mentioned in 'Enabling technologies' chapter (Sect. 2), React is actually one of the most popular frameworks in the market. To build a React dashboard connected to Elasticsearch has been an important step over to future developments.

Make filters using React libraries ReactiveSeach help has been crucial because it has been responsible for managing connections to the index. Furthermore, the multiple libraries from React have been very useful to develop innovative widgets and filters in this project. React offers an extensive catalog of free code libraries which can be accessed through GitHub and installed in our project through NPM.

Build an authentication server To access the website, user credentials are required. This allows us to manage dynamic pages based on topics chosen by the user. Alerts consist on inform users about recent news emerged in the last days. Users can updated their profile as they want. To achieve this goal, we have used useful technologies which combine with Javascript code such as MongoDB and NodeJS.

Build a mobile application connected to Elasticsearch One of the motivations to use React was the easy migration to build native applications. To make this possible, React Native offers native components build in React. ReactiveSearch has also been important to manage connections to the Elasticsearch index.

5.4 Problems faced

During the development of this project we had to face some problems. These problems are listed below:

- **Integrate external libraries with ReactiveSearch:** ReactiveSearch has his own components, but one of the goals of this project was to build our own filters. Select those libraries which better fit with our data structures was important to make the best filters. Once one made, all the rest are an easy task.
- **Integrate ReactJS, NodeJS and MongoDB:** the architecture of the project has been also a problem faced. Different connections between them and different technologies used have been a hard work to understand.
- **Adapt to changes in the index:** the index has changed during the development of this project. To adapt the application to different structures of data has been a problem faced.

5.5 Future work

In the following section, suggestions of possible improvements of the project are explained. This project can grow as much as we want, so these are only a few details:

- **Multiple topics to search:** While it is true that we can update as many times as we want our topics, the possibility to select as many topics as we want could be a future work to do.
- **Add relations between entities:** entities which take part in the news have usually got a relation. For instance, if Donald Trump and United States are taken from the article, we can display that Donald Trump is from United States. This could be possible thanks to Sparql [9].
- **Integrate Java or Swift components to mobile application:** there are a lot of free code components available of Swift or Java, and integrate them with the application should be an important step over. Also the connection to mobile native functionalities should be interesting.
- **Login and send mobile notifications:** One of the main pages of the web application consists on managing custom alerts for each user. A login page can be made in the mobile application for managing users and then, send notifications when an article is added to the index through native functionalities of Android or iOS. It should probably be the best future development.

APPENDIX A

Possible impacts of this project

In this chapter, possible impacts and responsibilities taken from this project are explained. This project is attempting to include an alerts system in order to receive notifications when a new article is published. Furthermore, this project is attempting to evolve in the technologies used for making web applications connected to Elasticsearch, and to build a mobile application which has an easier accessibility.

A.1 Social context

The main use case of this project consists on jihadists news. Below are detailed some data about attacks made by Islamic organisations. Data extracted from 'Statista' [1] expose that the main terrorism attacks suffered in the last year are provoked by Islamic organisations. They have caused hundreds of deaths in the period between 2017 and 2018.

The attacks have taken place in many popular cities around the world: Barcelona, Manchester, Mogadiscio, etc.

In 2017, Daesh (an islamic organisation) has provoked 10.326 deaths[2], being the most

[1]https://es.statista.com/estadisticas/577222/ataques-terroristas-en-el-mundo-por-numero-de-muertes/
[2]https://www.elespanol.com/reportajes/20170818/239976868_0.html

affected countries Irak (309), Afghanistan (115) and Nigeria (69).

The 95% of victims which Daesh has produced are of Islamic origin. Attacks affect us and we are in risk of suffering one, so it is under our responsibility to do as much as we can to combat them. In this project we have made an application with two main features. The first consists on receive alerts when an article is published in one of the main newspapers around the world. The second one allows us to keep track of news published over the last few months. Furthermore, an analysis has been extracted from those news, so we can visualise the relation between them. For instance, an organisation which takes part on several news.

A.2 Social impacts

Make an application to track articles about jihadism can have important social impacts. It is important to do as much as we can to mitigate Islamic attacks, so we must collaborate with the Spanish police and the Intelligence Nacional Center (CNI). This project takes part of a global project against terrorism in collaboration to these two organisations. Our mission is to build a visualisation module to receive alerts and track news.

The possible impacts this development supposes will be described.

- **Better tracking of news:** The alerts system and the filtering page allows the users to make a better tracking of news, making relations between news and getting conclusions of that.
- **Articles analysis done:** Organisations, places and people which take part in the articles are extracted through an analysis made of them. It could help us to avoid possible attacks in cities or with people affected.
- **Possibility of relate articles to help police investigations:** Police and research organisations could use this application in order to get a better investigation of new attacks. Furthermore, if they get a relation between these attacks and possible future attacks in progress, they will be able to detect and avoid them.
- **Better system accessibility:** An application for mobile phones could help us to access the news in an easier and faster way.

To resume, this project has two main impacts. It could help to analyse Islamic organizations and prevent future attacks. It could help police and research organizations to better access to this analysis from platforms such as a mobile application.

A.3 Economic impacts

This development means savings in police and government investigations, both time and money. Saving of time, because the application allows the visualisation of the analysis of the articles in order to get as much information as possible to relate the news. And saving of money, because the use of the application makes researches faster.

APPENDIX B

Economic budget

This appendix presents the costs of the system. Logistic and personal costs should be considered during the project development. This project has been carried out in 5 months, including software development of the two applications and the writing of this thesis.That means that the month costs will be multiplied by 5.

With regards to logistic aspects, several resources should be considered. First of all, the computer equipment, which includes an HP computer with an i5 processor of 8th generation. The total amount of this computer equipment is 1.000 €. Furthermore, a mobile phone is needed to develop the mobile application. An iPhone 6S has been used which supposes a total amount of 600 €.

In respect of shared Internet connection, it supposes a monthly cost of 20 €, which represents 100 € in total.

Software licences are not needed because all the software used is open-source. Meanwhile React is a Javascript open-source framework owned by Facebook, ReactiveSearch has several payment plans, but also a free one. The library can be installed for free through NPM, so its components can be used for free. However, there is a possibility to house the Elasticsearch index. In our case, that index is stored in the GSI department servers.

With regards to personal costs, we have to estimate the total of hours the project has lasted. As mentioned before, the project has been carried out in five months, from September until January 8. Counting only the working days, there is an amount of 84 days, excluding Spanish holidays. The work per day has been approximately 4 hours, so the total of hours is *84 days x 4 hours/day = 336 hours.* The perceived salary per hour by a Telecommunications Engineer Intern in Spain is about 7 €, which supposes *336 hours x 7 €/hour = 2.352 €*

The next table shows a resume of total costs:

Economic budget	
Entry	**Cost**
Telecommunications Engineer Intern (Partial time)	2.352 €
HP computer	1.000 €
iPhone 6S used to develop mobile applications	600 €
Software licenses	0 €
Internet connections and technical equipment	20 €/month
Total	4.052 €

Finally, we must apply the Spanish tax VAT. It supposes an increment of the 21% of the total cost, so the final budget is *4.052 € + 21 % =* ***4.902,92 €***

Bibliography

[1] npm. https://www.npmjs.com/.

[2] appbase.io. Reactivesearch - react and react native ui components for elasticsearch. https://opensource.appbase.io/reactivesearch/.

[3] Alex Banks and Eve Porcello. *Learning React: Functional Web Development with React and Redux.* " O'Reilly Media, Inc.", 2017.

[4] Marta Ley Belén Belmonte. Principales atentados yihadistas en europa desde 2004 — españa home — el mundo. https://www.elmundo.es/grafico/espana/2017/08/18/5997190c46163fd4128b4597.html.

[5] Adam Boduch. *Flux architecture.* Packt Publishing Ltd, 2016.

[6] ChartJS. React-chartjs-2 - npm. https://www.npmjs.com/package/react-chartjs-2.

[7] Kristina Chodorow. *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage.* " O'Reilly Media, Inc.", 2013.

[8] GSI department. What is gsi crawler? — gsi crawler 1.0 documentation. https://gsicrawler.readthedocs.io/en/latest/gsicrawler.html.

[9] Bob DuCharme. *Learning SPARQL: querying and updating with SPARQL 1.1.* " O'Reilly Media, Inc.", 2013.

[10] Elasticsearch. Twitter input plugin — logstash reference [6.5] — elastic. https://www.elastic.co/guide/en/logstash/current/plugins-inputs-twitter.html.

[11] Rodrigo Barbado Esteban. Design of a prototype of a big data analysis system of online radicalism based on semantic and deep learning technologies. http://www.gsi.dit.upm.es/administrator/components/com_jresearch/files/publications/tfm_rodrigo.pdf, 2018.

[12] EU. Trivalent. https://trivalent-project.eu/.

[13] ExpressJS. Express routing. https://expressjs.com/en/guide/routing.html.

[14] Artemij Fedosejev. *React. js Essentials.* Packt Publishing Ltd, 2015.

[15] Google. Github - google-map-react: Google map library for react that allows rendering components as markers. https://github.com/google-map-react/google-map-react.

[16] Google. Google trends. `https://trends.google.com/trends/?geo=US`.

[17] Clinton Gormley and Zachary Tong. *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine.* " O'Reilly Media, Inc.", 2015.

[18] Brad Green and Shyam Seshadri. *AngularJS.* " O'Reilly Media, Inc.", 2013.

[19] Simon Holmes. *Mongoose for Application Development.* Packt Publishing Ltd, 2013.

[20] Azat Mardan. *Express. js Guide: The Comprehensive Book on Express. js.* Azat Mardan, 2014.

[21] MongoDB. Mongodb schema design: Part 1 — mongodb. `https://www.mongodb.com/blog/post/6-rules-of-thumb-for-mongodb-schema-design-part-1`.

[22] Vladimir Novick. *React Native-Building Mobile Apps with JavaScript.* Packt Publishing Ltd, 2017.

[23] Jarrod Overson and Jason Strimpel. *Developing Web Components: UI from jQuery to Polymer.* " O'Reilly Media, Inc.", 2015.

[24] Stuart Ratcliffe. *ASP. NET Core 2 and Vue. js: Full Stack Web Development with Vue, Vuex, and ASP. NET Core 2.0.* Packt Publishing Ltd, 2018.

[25] J Fernando Sánchez-Rada, Carlos A Iglesias, Ignacio Corcuera, and Oscar Araque. Senpy: A pragmatic linked sentiment analysis framework. In *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*, pages 735–742. IEEE, 2016.

[26] Pedro Teixeira. *Professional Node. js: Building Javascript based scalable software.* John Wiley & Sons, 2012.

[27] James Turnbull. *The Logstash Book.* James Turnbull, 2013.

[28] Material UI. The world's most popular react ui framework - material-ui. `https://material-ui.com/`.