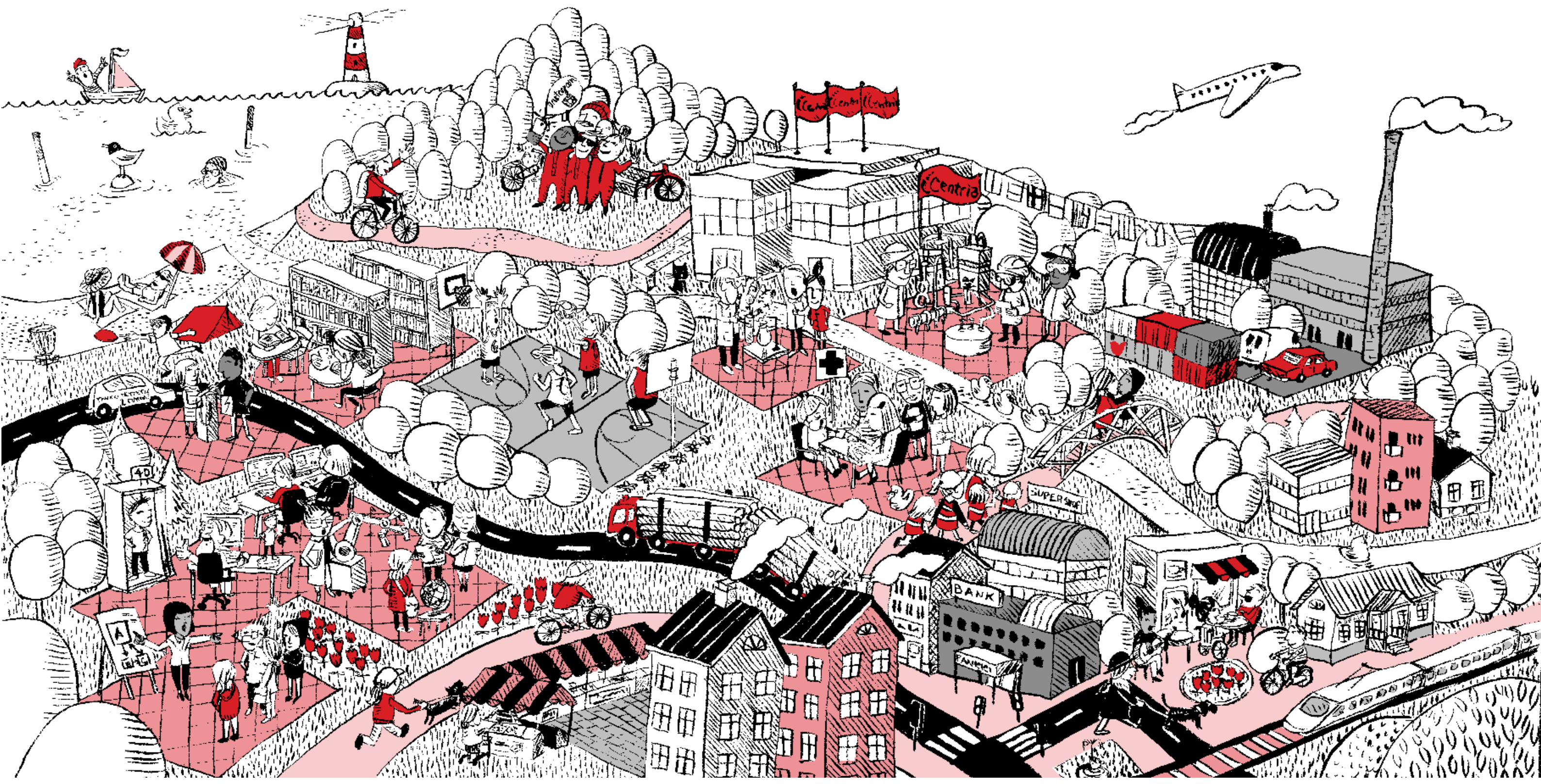
**Viktor Spasenkov**

# SINGLE PAGE WEB APPLICATION

**Train tracking web application**

**Thesis**
**CENTRIA UNIVERSITY OF APPLIED SCIENCES**
**Bachelor of engineering in information technology**
**February 2022**



**ABSTRACT**

| | | |
|---|---|---|
| **Centria University of Applied Sciences** | **Date** Februrary 2022 | **Author** Viktor Spasenkov |
| **Degree programme** Bachelor of engineering in information technology | | |
| **Name of thesis** SINGLE PAGE WEB APPLICATION. Train tracking web application | | |
| **Centria supervisor** Jari Isohanni | | **Pages** 34 + 20 |
| **Instructor representing commissioning institution** Jari Isohanni | | |

The main purpose of this thesis work was to create a RESTful single page web application that will show train location on Google maps using Digitraffic API. Train location data is loaded onto the map in a GeoJSON format with some specific train information being loaded simultaneously and shown upon clicking or tapping on a specific train.

The theoretical background consists of different parts. The first part presents web architecture it covers how it works and communicates with users. The second part describes RESTful interfaces their request-URI method and the way these interfaces are formed. The third part describes JSON as a data format and GeoJSON as a special variant of JSON for location data and features with examples of JSON and GeoJSON responses from Digitraffic API. The fourth part describes the way web maps work, how they obtain data with a figure describing architecture, its variants, and describes some web maps available for customers to use.

The practical part describes the architecture of the web page, and the different types it can be. It also covers the implementation process of designing, how data is obtained and settled on the map.

Finally, conclusion summarises development process and describes further possible development.

**Key words**
HTML, CSS, JavaScript, JSON, GeoJSON, REST, API, Frontend

## CONCEPT DEFINITIONS

IoT – Internet of Things. Network of physical objects integrated with sensors, software, and other technologies in order to connect and exchange data with other devices and systems over the internet.
API – Application Programming Interface. A software interface that establishes a connection between computers, systems, devices, or other programs.
JSON – JavaScript Object Notation
XML – Extensible Markup Language
GUI – Graphical User Interface
SDK – Software Development Kit
URL – Uniform Resource Locator
URI – Uniform Resource Identifier
TLC – Transport Layer Security
SSL – Secure Sockets Level

**ABSTRACT**
**CONCEPT DEFINITIONS**
**CONTENTS**

**FIGURES**

# 1 INTRODUCTION

People started using maps thousands of years ago. First, they were used to obtain basic geographical information about the nearby surrounding area. With people exploring areas around them on a bigger scale, the need for bigger maps with more data appeared. The more people were advancing in technology the more detailed maps got. Nowadays maps not only cover all areas on the earth but cover all sorts of different data leading to a variety of map types: resource, weather, political, topographic, time zone.

With the appearance of the internet, maps got a digital form, and with a blistering development in IoT technology devices, sensors, meters, and other equipment can share data and provide it to be used to automatically create different types of maps or to put new data on an existing web map. There is one chapter about digital map services to cover.

This project aims to develop a single-page web application that manipulates data provided by a Digitraffic API. The train data is collected and then used as a web map marker.  The user can access the website and see current locations of trains and then click or tap on a specific train to get its information. There is also a possibility for further development to get additional features.

# 2 WEB ARCHITECTURE

Web architecture is the conceptual structure of the World Wide Web (WWW). WWW is also called the internet and is a constantly loping and changing medium. WWW enables the communication between users and technical communication between different systems. The base for this is a variety of components and data formats, layering on top of each other. Overall, they form the internet infrastructure, which is possible with core components of data transition protocols, representation formats, and routing standards. Term web architecture is separated from the terms website and information architecture. Communication is established by different types of devices and systems (FIGURE 1).

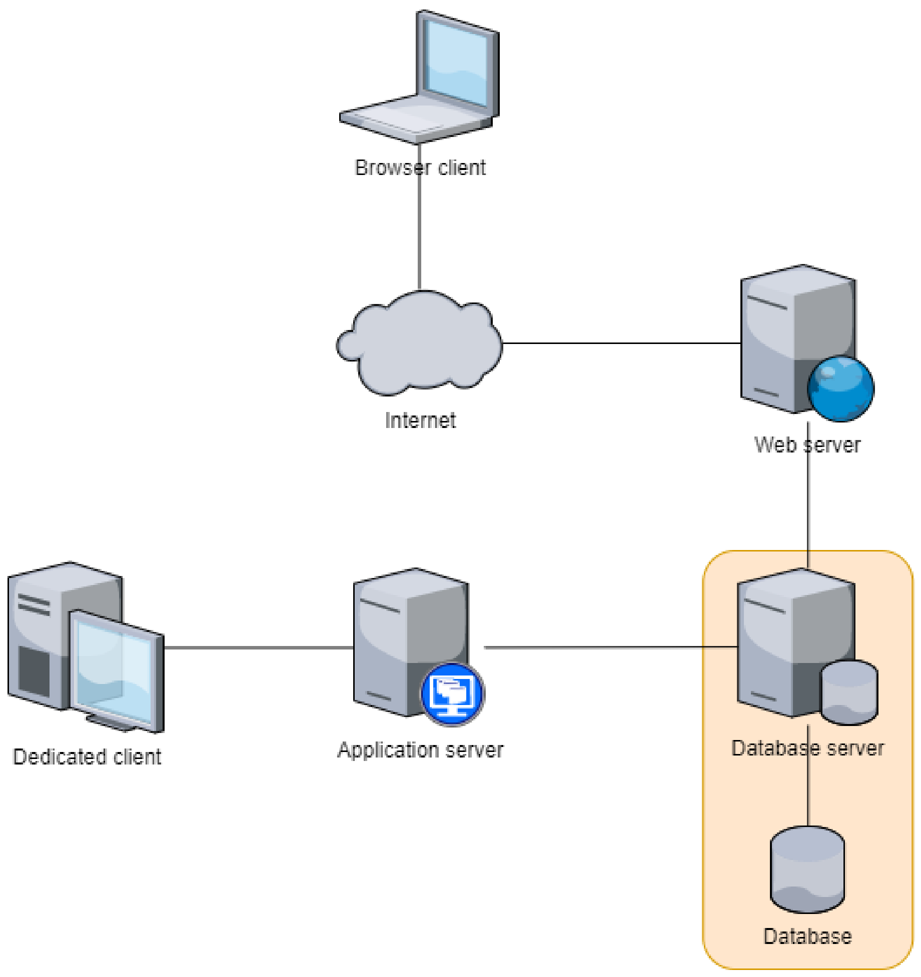


FIGURE 1. Classic web architecture (Baldric Mosley 2014.)

### 2.1 Client-server model

From the beginning web was formed by two-level structure: clients and servers. They divided and shared services and tasks system was supposed to perform. Client could send a request for a service from a server, then server replies by providing service. One example of a two-layer model is getting website by using URL-address, which leads to a server, to load site on user's internet browser.
Web architecture must have representation formats with a fixed standard, the protocol for data transfer, the standard for addressing, so that distributed application systems to communicate with each other. (Ryte Wiki 2022.)

#### 2.1.1 Representation formats with a fixed standard

There should be list of fixed standards, in order to provide availability to websites, services and made communication between user and server possible. An example of this is the browser war in the middle of the 90s. Two main competitors Netscape and Microsoft decided to add their own unique sets of elements into HTML. It led to confusion of various designs to work in WWW to be compatible in only on browser or another. Main problem was creating cross-browser programs in JavaScript. To solve this, developers had to do different versions of the same code or page to make it workable on all major web browsers. Some of the most frequent formats used nowadays are HTML, CSS and XML. (Ryte Wiki 2022.)

HyperText Markup Language (HTML) is a language used in most web pages and web applications. Hypertext is used to reference other pieces of text and markup language is a set of marks that indicate to web servers the structure and style of a document. The average website can have several HTML pages. They have a series of HTML elements, including a set of tags and attributes. A tag tells where an element begins and ends, attribute contain a description of an element. (Astari S. 2022)
Cascading Style Sheets (CSS) specifies style including page layout, colours and fonts. CSS can be placed inline in each element, internally in an HTML file or externally in a separate CSS file with the link in an HTML file. (Skillcrush 2022.)

Extensible Markup Language (XML) is a markup language. It is like HTML but has no predefined tags. In XML developer defines its own tags for specific needs. The main purpose is to store data in a format that can be searched, shared, and stored.  It is accepted across systems and platforms due to standardized syntax. (Mozilla 2022.)

### 2.1.2 Protocols for data transfer

Nowadays there is a variety of data being transferred between users and servers. To make it possible, a set of protocols for data transfer must be utilized by every software. One of the first protocols is protocol 1822, created as a starting point for host-to-host communication. An initial version was developed in 1969 as a part of ARPANET. It includes the physical layer, data link layer, and network layer. Every 1822 message has a message type, a numeric host address – a predecessor of an IP address, and a data field. 1822 protocol was not suitable for processing multiple connections among different applications on a host computer. This problem was fixed with the Network Control Program (NCP). It provided a standard method to provide communication, among different processes in different host computers. The NCP interface provided software connection across ARPANET with high-level communication protocols. It is a TCP/IP set of protocols. Two main protocols are most important for it: Transmission Control Protocol and Internet Protocol. This set of protocols has 4 layers: Application layer, Transport layer, Internet layer, and Network Access layer. (Michael Hauben 1998.)

Some of the commonly used protocols nowadays are HTTP (Hypertext Transfer Protocol), HTTPS (Hypertext Transfer Protocol Secure). They are part of the application layer. HTTP is a protocol for presenting information for transferring data over the network. Most data sent nowadays like website content and API calls use HTTP. There are usually HTTP requests and responses to that request. A common example of this is RESTful interfaces. All data is sent in plain text without any security. To resolve this, HTTPS was developed. TLC was used to transmit data in that protocol by public-key encryption. Nowadays public key is shared with the client by the server's SSL certificates. Every certificate is cryptographically signed by Certificate Authority. Each browser contains a list of that certificates. (Mozilla 2021.)

### 2.1.3 The standard for addressing

The main standard for addressing nowadays is URL. It was created in tin the 90s as an instance of a URI (Uniform Resource Identifier). This standard includes already pre-existing domain names system slashes to separate directories. Valid URL points to the unique resource. Every URL is composed of different parts. Some of them are mandatory to be in URL. (Mozilla 2022.) The most important parts are the scheme, domain name, port, the path of the file, parameters and anchor (FIGURE 2).

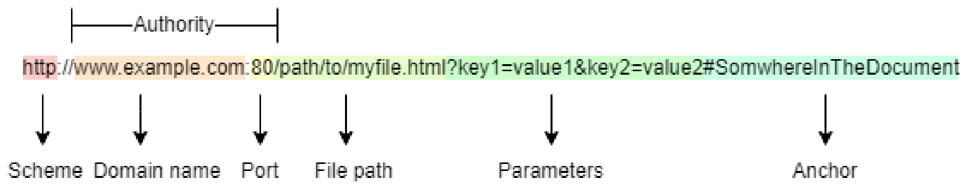

Authority
http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#SomwhereInTheDocument
Scheme
Domain name
Port
File path
Parameters
Anchor

FIGURE 2. Example URL composition (Mozilla 2022.)

The scheme is an indicator of the protocol that the browser uses to request the recourse, usually HTTP or HTTPS. After the scheme, there is the authority which consists of domain – an indicator of web server being registered, usually a domain name or IP address, and port – an indicator of the technical gate used to access the web server. If the web server uses standard ports of the HTTP protocol, it is not mandatory to have a port in the URL. Next goes the path to the resource on the web server. To communicate with the web server parameters can be added to the URL as key value pairs. The server uses those parameters to do extra tasks specified in the parameter. Each server has its own way of dealing with the parameters. The anchor is used to direct to another part of the resource. (Mozilla 2022.)

Most languages use the Latin alphabet, so does URL. But with further development of the internet, some non-Latin alphabet countries wanted URLs in their native language. To make it possible, internationalized URL was created. Links take form like http://xn--90ahbyb4h.xn--p1ai will be shown on user browser as мебель.рф, xn-- in this link is an indicator that character is not originally Latin or ASCII. Same works when only part of the URL is not in the Latin alphabet. For example, this link https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B1%D0%B5%D0%BB%D1%8C will be shown to the user as https://ru.wikipedia.org/wiki/мебель. (Ryte Wiki 2022.)

### 2.2 Three-tier model

Three-tier models include logic between client and server, which does data processing and allows a certain state of interaction. For example, an application server is used only for data processing, while a database server is only for data storage (FIGURE 3). This means that content can be dynamically loaded and saved. Generally, there is a difference between server-side and user-side data processing. (Ryte Wiki 2022.)

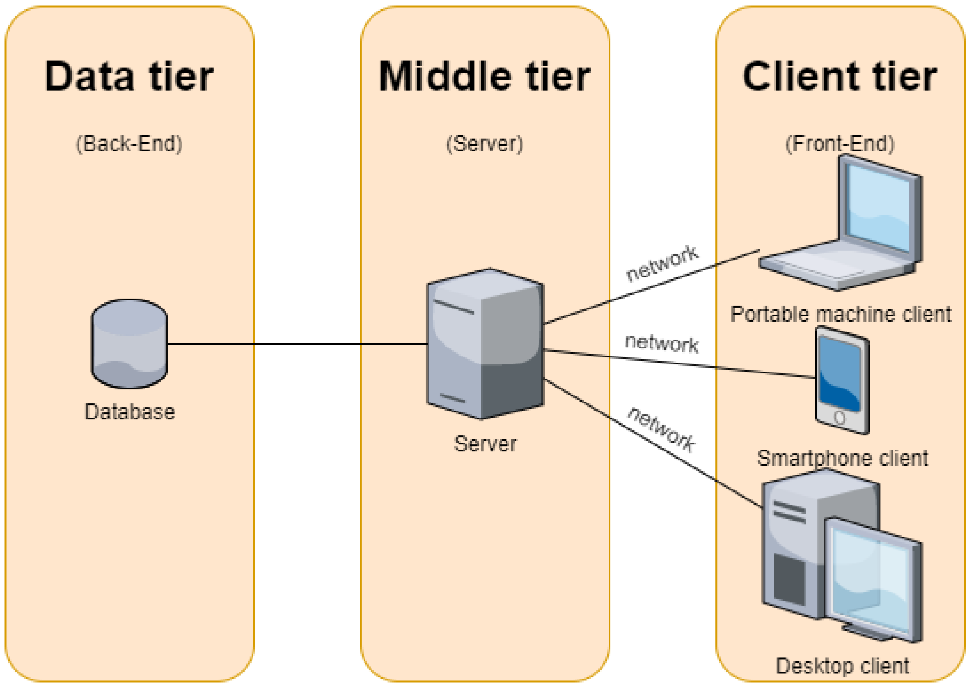


FIGURE 3. Three-tier model (Susan Goodwin 2019)

Dynamic websites allow to data being changed on user-side without any new connections between server and user. Actions of users are influenced by scripts or scenarios, so no asynchronous data transfer needed. On server side, changed content is saved by application server on the database server, it can also be a virtual server, which simulates work of a physical one. (Ryte Wiki 2022.)

### 2.2.1 WSDL and SOAP

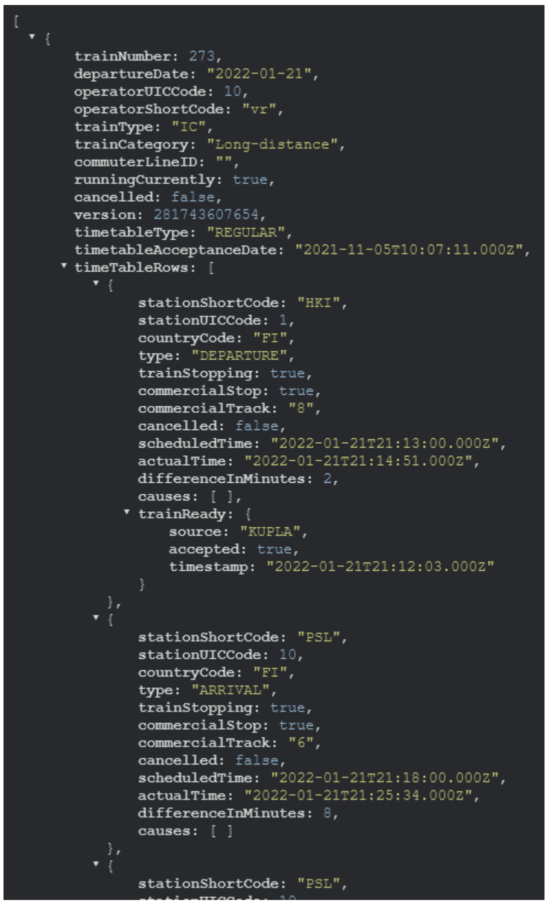
Web Services Description Language (WSDL) is a meta-language for describing network services based on XML, enabling web-service to implement and perform certain specific tasks. WSDL can also define interface to a web service.

Simple Object Access Protocol (SOAP) is also based on XML and allows to control of web services in form of procedure calls. Those calls are realized by remote procedure call protocol. (Ryte Wiki 2022.) SOAP has four layers of architecture: header, body, envelope, fault. WSDL has three main elements: types, binding and operations. SOAP has generally more community and programming models support. WSDL is often used with SOAP to provide web services. A client program that connects to web service can use WDSL file to determine what an operation server can do. Then, the client can use SOAP to perform one of the operations listed in WSDL file. (Wikipedia 2021.)

### 2.2.2 JSON

JSON is minimal, simplistic, and easy-to-read, structuring data format. It is widely used specifically to transfer data between web applications and severs, alternatively to XML. It was created in early 2000s as a real-time server-to-browser communication alternative to the use of browser plugins. JSON is based on JavaScript but can be used as a data format in a wide range of other programming languages. (Ilari Léman 2018.)

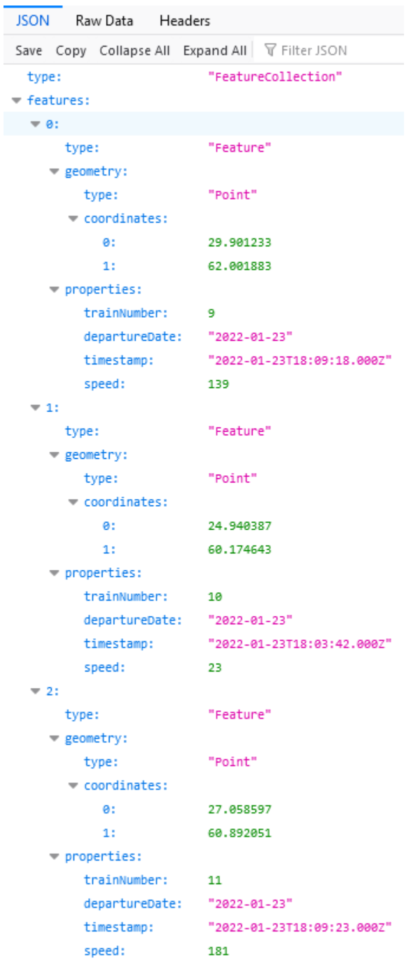
Values in JSON can take different forms. It can be an array, real or integer number, Boolean value, line, and JSON object. JSON in most cases is an array with key/value pairs. Key is the name of the parameter that is passed to the server and server responses with suitable value. Every developer decides which JSON is valid, and which is not. But there is a set of rules, used to describe well-formed JSON that must not be broken: data is written in ley/value pairs, data is separated by commas, and an object is inside curly braces {}, and each array is inside square brackets []. JSON is mostly used in REST APIs (FIGURE 4). Digitraffic uses JSON to output different traffic data, like trains, route restrictions, route works.



```
[
  ▾ {
        trainNumber: 273,
        departureDate: "2022-01-21",
        operatorUICCode: 10,
        operatorShortCode: "vr",
        trainType: "IC",
        trainCategory: "Long-distance",
        commuterLineID: "",
        runningCurrently: true,
        cancelled: false,
        version: 281743607654,
        timetableType: "REGULAR",
        timetableAcceptanceDate: "2021-11-05T10:07:11.000Z",
      ▾ timeTableRows: [
          ▾ {
                stationShortCode: "HKI",
                stationUICCode: 1,
                countryCode: "FI",
                type: "DEPARTURE",
                trainStopping: true,
                commercialStop: true,
                commercialTrack: "8",
                cancelled: false,
                scheduledTime: "2022-01-21T21:13:00.000Z",
                actualTime: "2022-01-21T21:14:51.000Z",
                differenceInMinutes: 2,
                causes: [ ],
              ▾ trainReady: {
                    source: "KUPLA",
                    accepted: true,
                    timestamp: "2022-01-21T21:12:03.000Z"
                }
            },
          ▾ {
                stationShortCode: "PSL",
                stationUICCode: 10,
                countryCode: "FI",
                type: "ARRIVAL",
                trainStopping: true,
                commercialStop: true,
                commercialTrack: "6",
                cancelled: false,
                scheduledTime: "2022-01-21T21:18:00.000Z",
                actualTime: "2022-01-21T21:25:34.000Z",
                differenceInMinutes: 8,
                causes: [ ]
            },
          ▾ {
                stationShortCode: "PSL",
```

FIGURE 4. Example REST JSON response from Digitraffic API

GeoJSON is based on a JSON open standard format, specifically made to represent geographical features with its non-spatial attributes. It is always a JSON object and single entity. GeoJSON can have an arbitrary number of key/value pairs. It must have type property of one of the following values: "Point", "MultiPoint", "LineString", "MultiLineString", "Polygon", "MultiPolygon", "GeometryCollection", "Feature", or "FeatureCollection". GeoJson can also have ”crs” property and “bbox” property (Howard Butler 2008). GeoJSON is used in Digitraffic API to show different types of locations like trains, road restrictions, or stations (FIGURE 5).



```
  type:                     "FeatureCollection"
▼ features:
  ▼ 0:
        type:               "Feature"
      ▼ geometry:
          type:             "Point"
        ▼ coordinates:
            0:              29.901233
            1:              62.001883
      ▼ properties:
          trainNumber:      9
          departureDate:    "2022-01-23"
          timestamp:        "2022-01-23T18:09:18.000Z"
          speed:            139
  ▼ 1:
        type:               "Feature"
      ▼ geometry:
          type:             "Point"
        ▼ coordinates:
            0:              24.940387
            1:              60.174643
      ▼ properties:
          trainNumber:      10
          departureDate:    "2022-01-23"
          timestamp:        "2022-01-23T18:03:42.000Z"
          speed:            23
  ▼ 2:
        type:               "Feature"
      ▼ geometry:
          type:             "Point"
        ▼ coordinates:
            0:              27.058597
            1:              60.892051
      ▼ properties:
          trainNumber:      11
          departureDate:    "2022-01-23"
          timestamp:        "2022-01-23T18:09:23.000Z"
          speed:            181
```

FIGURE 5. Example GeoJSON response from Digitraffic API

### 2.2.3 REST

Representational State Transfer is a similar approach, used for communication between machines in distributed systems. It is based on a client-server architecture but characterized by unified interface. This feature simplifies usage of REST with variety of resources or objects. REST is designed for internet-scale usage, so the connection between client and server must be as lightweight as possible to allow large scale implementation. (Oreilly 2022.)

While this concept is in the base of the World Wide Web, the term REST was announced in 2000 by Roy Fielding (Red Hat 2020.). He also proposed six principles as basis of REST. Client-Server – Concerns should be separated between client and server. This will allow both sides components to update independently, so that system can scale. Stateless – Communication between server and client should be stateless. Server should not remember state of the client. Clients should include in all the necessary information in the request instead, so that server can understand and process it. Layered System – Multiple hierarchical layers including gateways, proxies, and firewalls can exist between client and server. These layers can be changed, reordered or removed transparently for better scalability. Cache – Responses from server must be announced as cacheable or noncacheable. This will allow users or its intermediate components to cache responses and reuse them for next requests. This lowers server load and helps to improve performance. Uniform Interface – All interactions between client, server, and intermediary components are based on uniformity of their interfaces. This makes overall architecture simpler, because components can develop independently, as long as they implement the agreed-on contract. Universal interface limitation is further divided into four subliminations – identification of resources, representation of resources, self-descriptive messages and hypermedia as mechanism of application state. Code on demand – Clients can extend their functionality by downloading and executing code on demand. Some examples of it are JavaScript scripts, Java applets. This constraint is optional. If application follows these constrains it can be considered as RESTful. (REST API 2021.)

# 3 RESTFUL INTERFACES

The concept of uniformity of interfaces is central to REST architecture. While each component follows this general principle, each module in architecture can be developed and deployed independently, overall complexity of architecture gets simplified, and server technology can be scaled easier. Simple interfaces also allow for an arbitrary number of intermediary blocks to be used and placed between server and client within communication. The uniformity of RESTful interfaces is built upon four guiding principles: identification of resources through URI mechanism, manipulation of resources through their representations, use of self-descriptive messages, and implementing Hypermedia as Engine of Application State. (Oreilly 2022.)
To communicate with RESTful interfaces, application makes the request and gets interface responses. This communication consists of several basic elements: Request-URI and HTTP Methods in requests, Status-Line in responses, headers and special headers to specify details of attachments when a message consists of multiple parts, and message body. (Oracle 2015.)

## 3.1 Request-URI and HTTP Methods in requests

There are four methods that are used to request a required action that will be performed on an abstract or physical resource: "GET", "POST", "PUT", or "DELETE". Resource has a specific URI. The most important part of any request that an application makes to the RESTfull interfaces is that the Request-URI identifies the abstract or physical resource that an HTTP method acts upon or uses. (Oracle 2015.)

### 3.1.1 GET

The GET method retrieves the state of previously set up specific resource and does not modify it in any way. It is said to be a safe method. GET APIs should be idempotent. When the multiple identical GET requests are made, the result must be same every time, until another API (POST or PUT) changes state of the resource on the server. If the Request-URI is referred to as a process of data producing, this produced data shall be returned as the entity in the response and not the source text of the process, unless the output of the process is that text. “HTTP GET http://www.appdomain.com/users” - an example of a GET URI. For every given GET request, if the resource is found on a server, there must be a return of an HTTP response code 200 (OK) with a response body. If the resource is not found on the server there must be a return of an HTTP response code 404 (NOT FOUND). Same way, if it is determined that GET request is formed incorrectly, then the server must return an HTTP response code 400 (BAD REQUEST). (REST API Tutorial 2021.)

### 3.1.2 POST

The POST method is used to access a resource factory to create a resource that does not have URI yet. Multiple requests can create multiple new resources. Responses to this method do not cache unless there is an appropriate Cache-Control or Expires header fields in response. Call of two identical POST requests will result in two different resources with the same information which means that POST is neither safe nor idempotent. “HTTP POST http://www.appdomain.com/users” - an example of a POST URI. When a resource is created on the origin server, there should be an HTTP response code 201 (Created) and contain an entity describing the status of the request and referring to the new resource, and a Location header. If the post request does not result in a resource that can be identified by a URI, the response code is 200 (OK) or 204 (No Content). (REST API Tutorial 2021.)

### 3.1.3 PUT

The PUT method is used to create a resource with a predetermined URI or to update an existing resource. When the request goes through a cache, and Request-URI identifies that one or more entities are currently cached, they should be treated as outdated. Responses to this method are not cacheable. “HTTP PUT http://www.appdomain.com/users/123” - an example of a PUT URI. When a new resource is created by this method, the origin server informs the user agent by the HTTP response code 201 (Created). If there is an existing resource and it is got modified, then there should be sent either 200 (OK) or 204 (No Content) responses to indicate that the request is completed successfully. (REST API Tutorial 2021.)

### 3.1.4 DELETE

The DELETE method is used to remove a specified resource, identified by the Request-URI. This method operations are idempotent which means if resource is deleted, it is removed from the collection of resources. When the request goes through a cache and Request-URI identifies one or more currently cached entities, those entitles are treated as state. Responses to DELETE are not cacheable. “HTTP DELETE http://www.appdomain.com/users/123” - an example of a DELETE URI. Successful DELETE request gets response code 200 (OK) if the response contains an entity describing the status.

When there is response after a performed action, but without entity response code is 204 (No Content). If the action is queued, response code is 202 (Accepted). (REST API Tutorial 2021.)

## 3.2 Status-Line

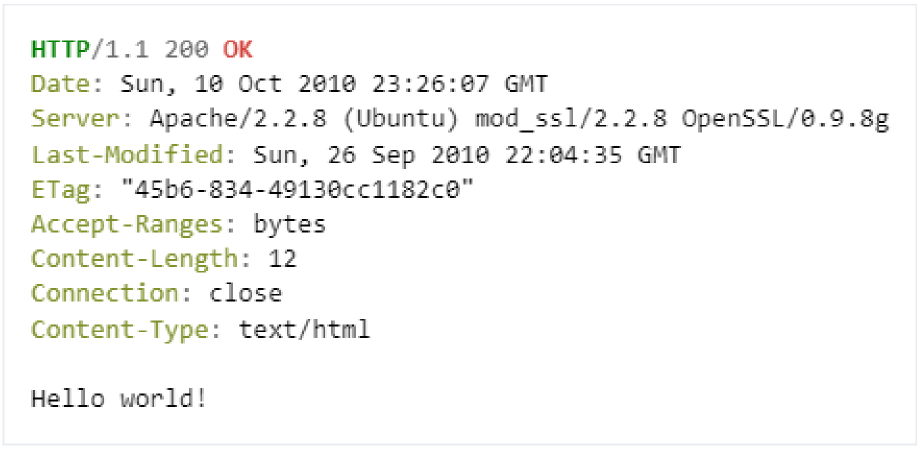
Status-Line or response code is the first line of every response that an application receives after interaction with a RESTful interface. Each Status-Line has Status-Code – three-digit indicator of success or failure and Reason-Phrase – a brief description of the performed successful action, or reason for failure. “HTTP/1.1 Status-Code Reason-Phrase” - is an example of Status-Line. (Oracle 2015.)

## 3.3 Headers

All requests and responses for RESTful operations contain several header fields. The authorization header field, which is required in all requests, is made to indicate the type of authentication and security. The Session ID is present in all request messages to identify the application. It is the first task of an application to get a session ID from the Session Manager Web service. Service correlation ID if an application needs service correlation. Tunnel parameters that are to supply parameters that are not supported in the RESTful interface but need to be passed to the network. Location header, located in responses for certain requests, is used to identify a new resource or redirect to a location different from the Request-URI. Current-Length that is to indicate the length of the request or response body. Content-Type that is to specify content in the request. (Oracle 2015.)

## 3.4 Message Body

Message body is only for a request or response and is present only when required. It is a JSON object. Request body provides extra data that is required to complete the specific request, or it can send data just to update the server. Bodies are divided into two categories: single-resource body and multiple-resource body. The single-resource body consists of one single file and is defined by content-type to indicate the original media type of the resource and content-length headers to specify the size of the message body. The multiple-resource body has several parts where each one has different bits of information. Each part is its own entity with its own HTTP headers: Content-Disposition and Content-Type (FIGURE 6). The response body provides data that the application will need for later action. (Oracle 2015.)



```
HTTP/1.1 200 OK
Date: Sun, 10 Oct 2010 23:26:07 GMT
Server: Apache/2.2.8 (Ubuntu) mod_ssl/2.2.8 OpenSSL/0.9.8g
Last-Modified: Sun, 26 Sep 2010 22:04:35 GMT
ETag: "45b6-834-49130cc1182c0"
Accept-Ranges: bytes
Content-Length: 12
Connection: close
Content-Type: text/html

Hello world!
```

FIGURE 6. Example of the message body

# 4 WEB MAP COMPONENTS

A web map is an interactive display of geographic data and information. The map consists of a basic map, used as a base, a set of layers of data an extent, and a set of instruments including navigation for planning and zoom option. These maps sometimes are called “slippy maps”. They achieve smooth usability by the use of tiling. Web map contains small tiles, that are loaded individually, instead of whole map load. When a person moves around the map, there is no reload of the whole map, just an additional load of new tiles. There are two types of tiles: raster tiles and vector tiles. Raster tiles are simply pre-rendered images with map data. It is an original standard. Vector tiles are small chunks of the data itself. It is now frontline technology because it made a live rendering of map data and information more achievable. Web maps use a standard tiling scheme. Each tile is identified by three numbers: zoom level which relates to an initial scale of the whole world fitting in one tile, a horizontal position (an X coordinate) of a tile at a zoom level, and a vertical position (a Y coordinate) of a tile at a zoom level. The web map is often created and maintained by the use of different servers and machines: a database or file servers to hold GIS data, a geospatial server for specialized software, and power to draw maps and perform operations to analyse GIS (FIGURE 7). (Sterling Quinn, John A. Dutton e-Education Institute, College of Earth and Mineral Sciences, The Pennsylvania State University 2022.)

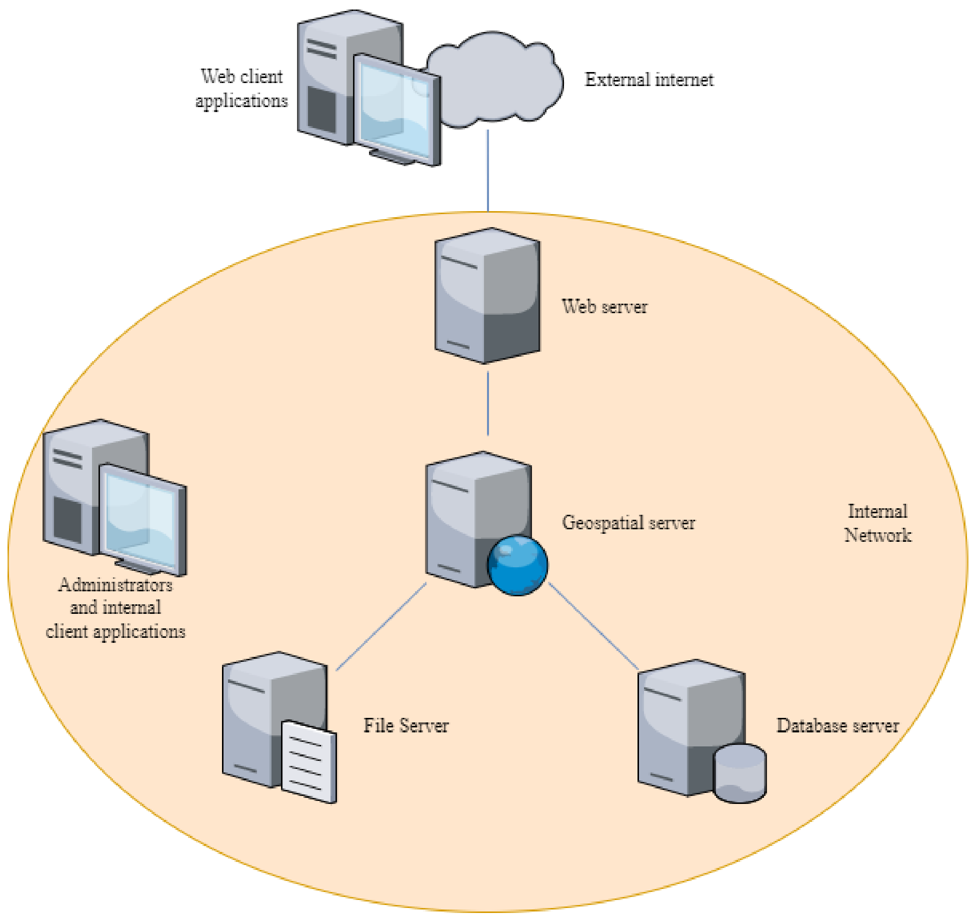
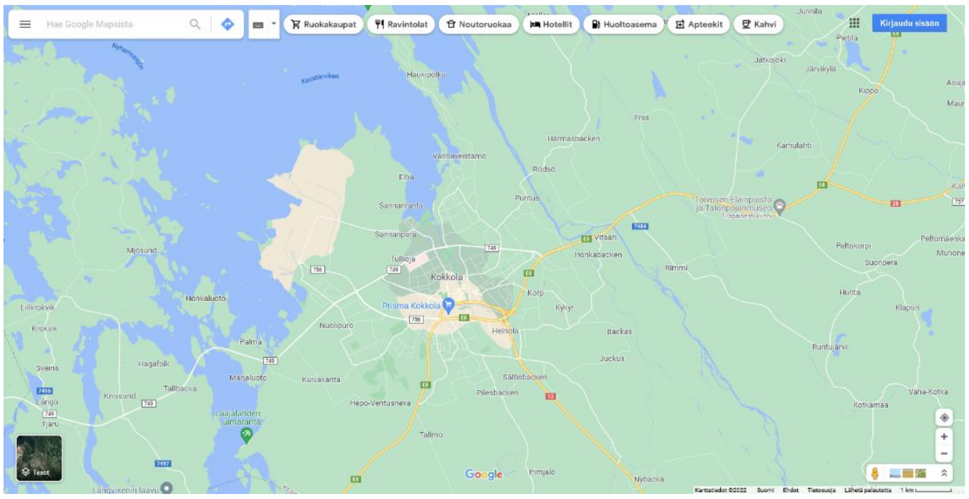


FIGURE 7. The example system architecture of web mapping (Sterling Quinn, John A. Dutton e-Education Institute, College of Earth and Mineral Sciences, The Pennsylvania State University 2022.)

## 4.1 Google Maps

Google Maps is a web service for mapping and application for consumers provided by Google LLC, based on a Google GIS, which is also used in Google Earth. It contains several different services for end-users, businesses, and developers. This web map is available on desktop and mobile. It contains services including accurate satellite view of the whole world, street view with panoramic view of streets (with some countries done fully) and other places, path planning for different means of transportation, real-time traffic jam conditions, place search. It is used by more than 1 billion people. Google maps has customization enabled. It is performed by coding an API. While Google Earth can be customized without any code. Google maps can be used independently or by integrating it into custom website. Google maps are also available on mobile device with special applications, allowing 3D world view. Google Maps provide a variety of APIs depending on  needs. An API to implement map to the website. A Geocoding API to convert addresses to coordinates and vice versa. A places API to provide time zone data in any location. A Maps Distance Matrix API to calculate distance and travel time between two points. A Driving Direction API to provide direction between locations. Free use of APIs is limited by 200$ free usage per month. More than that is 2$ per 1000 calls. (RapidAPI Staff 2021.)

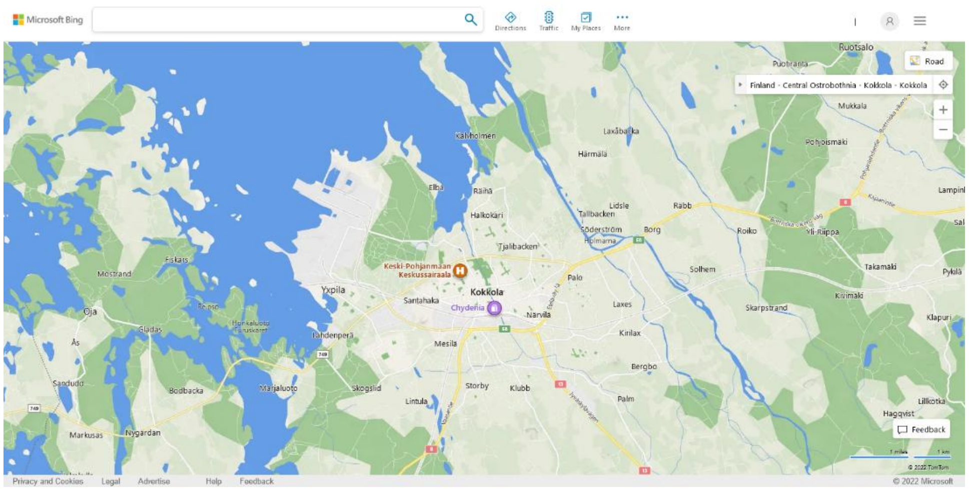


PICTURE 1. Google Maps

Google Maps are connected to other Google services, which provide location information of places, businesses, and organisations. Users and business owners can add locations on map. Google has more advanced GIS based on a Google Maps and several other services similar to Google maps like space, moon. (Google Maps 2021.)

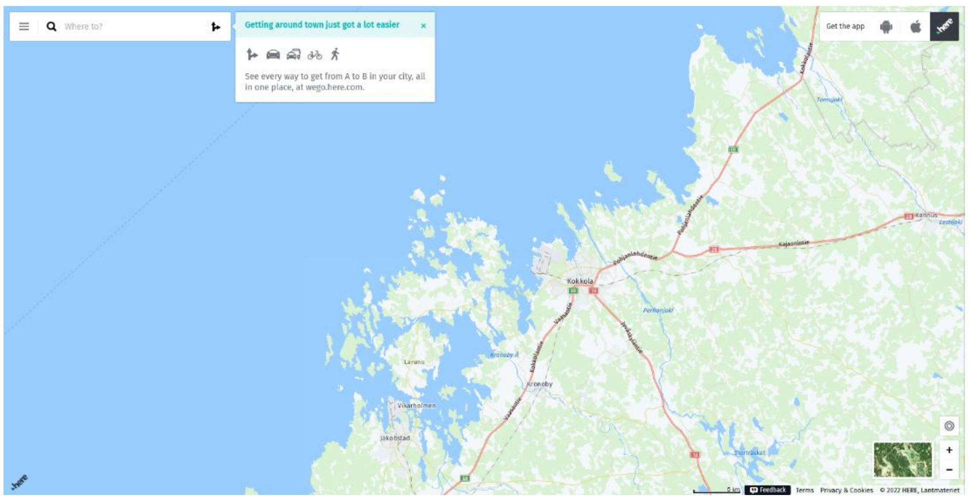
## 4.2 Bing Maps

Bing Maps is a mapping service in web, provided as a part of Microsoft Bing search engine, based on a Microsoft MapPoint platform. After purchasing Nokia in 2012, Bings Maps is partly based on Here maps. It has all basic features of modern web map, but special future is bird-eye view, which allows to see aerial photos of upper perspective from four to different locations. It also has own API for customer users and user map layering and shaping through Bing Maps GUI. API also provides 3D imagery, high-resolution aerial imagery, traffic information, route data, geocoding, interactive and static maps, and spatial data sources. Street view availability is limited. Bing Maps has a variety of pricing options, but basic subscriptions start with 4500$ per 100000 transactions. (RapidAPI Staff 2021.)


PICTURE 2. Bing Maps

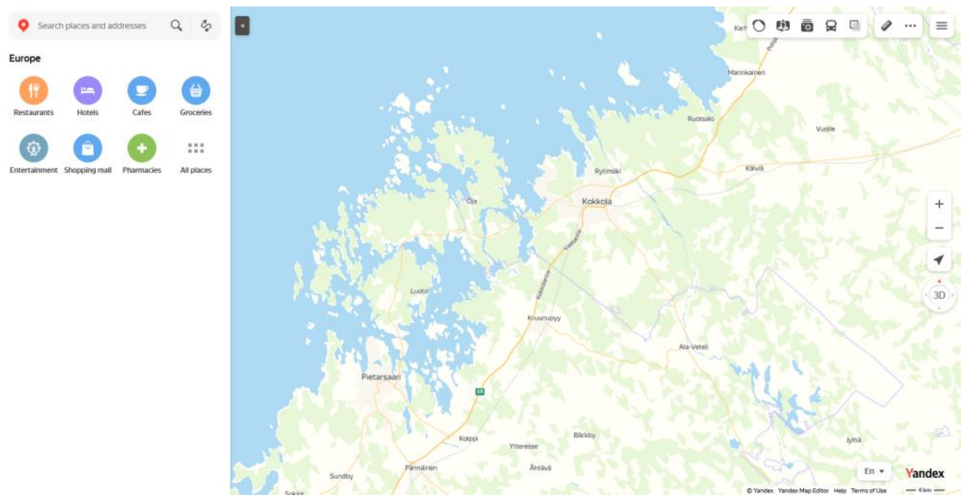
## 4.3 HERE WeGo

Is a web map service originally developed by Nokia and currently under HERE Technologies operation. It uses in-house data including satellite view, traffic condition, locations information and other features of modern web map. Its location information is integrated with TripAdvisor service where it gets all suitable data. Users can also use its API and SDK for own development. Here WeGo is available on web and mobile. Due to its partly ownership by Audi, BMW, Mercedes and Mitsubishi its platform is widely used in automobile industry (HERE 2022). HERE provides APIs for routing, geocoding, traffic, public transit, weather and vector tile. HERE API map use for 1000 daily transactions is free, more than that is base plan with pay as-you-grow transaction, price itself is not listed. (HERE 2022.)


PICTURE 3. HERE WeGo

## 4.4 Yandex.Maps

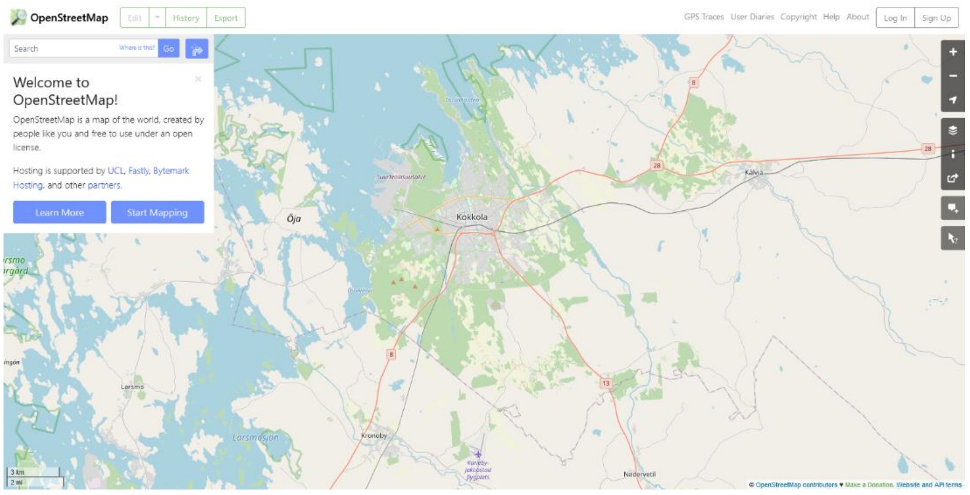
Yandex.Maps is a Russian web mapping service created by Yandex. Due to its location, it is mostly targeted to post-Soviet countries, with some most common features are only available in those countries. It has most detailed maps for Russia, Belarus, Ukraine and some other post-soviet countries, including Turkey. Street view is limited to post-Soviet countries with unique places like Chernobyl, Pripyat and Mount Everest. User can add new data to map using Yandex Map editor. It has web version and mobile applications. Its API includes geocoder for address by coordinates and vice versa, places API to find certain organization, static API for static image of map section, MapKit and several JavaScript APIs. API is free for non-commercial use with some limitations to 25000 requests per day. Commercial use pricing can be requested. (Yandex 2022.)



PICTURE 4. Yandex Maps

## 4.5 OpenStreetMaps

A collaborative web map project with a goal to create free editable maps with usual functionality. It was made with use of free sources and user collected GPS data and aerial photography. Users can add new maps, roads, places and map layers, so range of map data is limited to users. It is used by Facebook, WikiMedia, Uber and others. It is owned by non-profit organisation. Has some map project based on it called OpenHistoricalMap (OpenStreetMap 2022.). OpenStreetMap has an editing API to add raw geodata to map database and SDK to implement map to website. (OpenStreetMap wiki 2020.)



Welcome to OpenStreetMap!

OpenStreetMap is a map of the world, created by people like you and free to use under an open license.

Hosting is supported by UCL, Fastly, Bytemark Hosting, and other partners.

PICTURE 5. OpenStreetMaps

## 4.6 Map service choice

By default, all these maps have basic features, which can be used in Finland. In most of them users can see more than just a map. All of them have an API to work with and integrate map to your own website. In this project Google maps is used because it is the most used web map in Finland, it has easy to use API with some tutorials to work with and it has free version with some limitations not related to this project. (RapidAPI Staff 2021.)

TABLE 1. Web map service comparison (RapidAPI Staff 2021.)

| Feature | Google Maps | Bing Maps | HERE WeGo | Yandex.Maps | OpenStreet-Maps |
|---|---|---|---|---|---|
| Licence | Proprietary | Proprietary | Proprietary | Proprietary | ODbl |
| Map zoom | 22 | 22 | 18 | 19 | 19 |
| Map types | 6 | 9 | 7 | 5 | 5 |
| API | Available | Available | Available | Available | Available |
| Mobile version | Yes | Yes | Yes | Yes | Yes |
| Is API free | $200 free per month | Unpublished | With limitations | For non-commercial use | Free |
| 3D mode | Limited to some areas | Yes | Yes | Yes | No |

# 5 DIGITRAFFIC API

Digitraffic is a service by Fintraffic which provides real-time traffic information. The provider of this service is state-owned under the control of Finland’s Ministry of Transport and Communications. Digitraffic covers the road, rail and marine traffic information. The development of an API was started in 2002 by the Helsinki University of Technology and VTT Technical Research Center of Finland. Digitraffic uses JSON and GeoJSON to represent traffic data. Road data information includes weather cameras, weather station data, weather forecasts, Traffic Measurement System data, traffic messages and road maintenance information. Marine data is made by marine warnings, harbour schedules, vessel location, Aton faults, waterway traffic, sea state estimation data and Dirways from a Baltice-system. The project developed for this thesis is concentrated on rail traffic. For that purpose, API shares data about trains that are operated in Finland. It contains data about train location, speed, route taken, stations, active trains, type, number, route history, wagon and locomotive composition, and timetables. (Digitraffic 2022.)

# 6 PRACTICAL SOLUTION

This part will be concentrated on the process of creating single page application. The main requirements were to create a web application that will draw trains or ship locations in real-time on the map and will work both on desktop and mobile. As a source of train or ship location data, Digitraffic API is meant to be used. There were other extra requirements including train or ship filtering, the ability to find a specific train or ship, zoning, and routes.

## 6.1 Architecture

It is important to understand web application architecture, because it will show the way application is constructed and works. It describes the interaction between components of a web application, databases, and middleware systems and ensures that they work simultaneously. When the user clicks on a URL or types it on a web browser address bar the request for a particular web address or web application is sent. Then server replies by sending files to the browser, so it can execute them to show the page requested and give the user ability to interact with it. Interaction is made through code with or without special instructions to the web browser on how to respond to different types of user's input. Any web application contains two types of code running side by side. Client-side code that is in the browser and made to respond user input and server-side code that is on the server to respond HTTP requests. This project will be concentrated on client-side code which is a combination of HTML, CSS, and JavaScript.

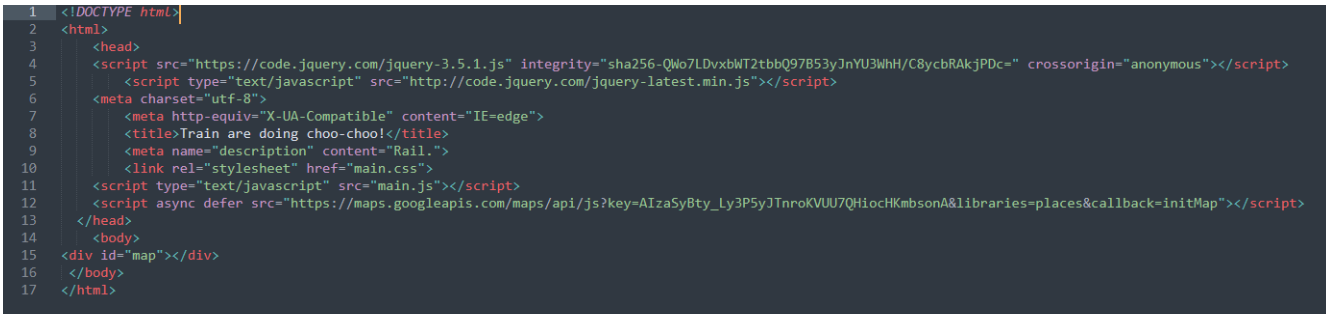
Every web application can have several components. This includes UI/UX web application components – interface layout of the web application that includes components like logs, notifications, settings. Structural components are client and server components. And the client component is part of the web application’s functionality for end-user to interact with. It is mostly developed in HTML, CSS, and JavaScript. The server component consists of at least two parts: app logic taking the role of the control centre of web application and database to store all persistent data.

Depending on the number of these web components web application model can be one web server, one database or multiple web servers, one database or multiple web servers, multiple databases. According to the way, different components of a web application interact with each other and the way, application logic is spread among client and server-side, there are 3 main types of web application architecture:

single-page applications, microservices, and serverless architectures. A single-page application is a website where instead of loading one of the multiple pages each time it is necessary, it dynamically changes data on the single page by providing content updates to it. There are no interrupts in user experience, so it can resemble a traditional desktop application in a way. A microservice is a small and lightweight service made for a single functionality. Components of microservice do not depend on each other, so they do not have to be built using the same programming language. Application with a server and infrastructure management outsourced on a third-party cloud infrastructure. So, an application can avoid tasks related to infrastructure. (Swapnil Banga 2021.)

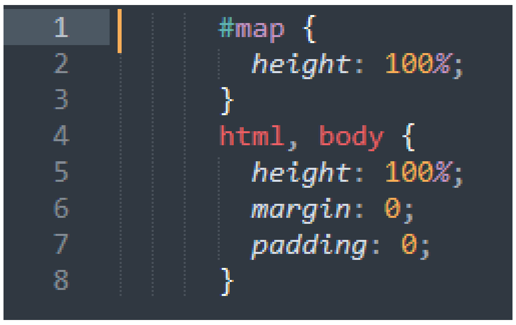
## 6.2 Implementation

This project is carried out as a single page web application. Consists of three files: HTML, CSS, and JavaScript. Most of the user interface is based on Google maps and its functionality. The main difference so far is train markers. In that case, CSS is not involved much. HTML is used as a core to load CSS and JavaScript files and handle the Google Maps API key. Google Maps API key is used to link a website to a Google Cloud Platform and billing account and make it possible to use Google Maps in a project.



```html
<!DOCTYPE html>
<html>
    <head>
    <script src="https://code.jquery.com/jquery-3.5.1.js" integrity="sha256-QWo7LDvxbWT2tbbQ97B53yJnYU3WhH/C8ycbRAkjPDc=" crossorigin="anonymous"></script>
        <script type="text/javascript" src="http://code.jquery.com/jquery-latest.min.js"></script>
    <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>Train are doing choo-choo!</title>
        <meta name="description" content="Rail.">
        <link rel="stylesheet" href="main.css">
    <script type="text/javascript" src="main.js"></script>
    <script async defer src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBty_Ly3P5yJTnroKVUU7QHiocHKmbsonA&libraries=places&callback=initMap"></script>
  </head>
    <body>
<div id="map"></div>
 </body>
</html>
```

FIGURE 8. HTML code

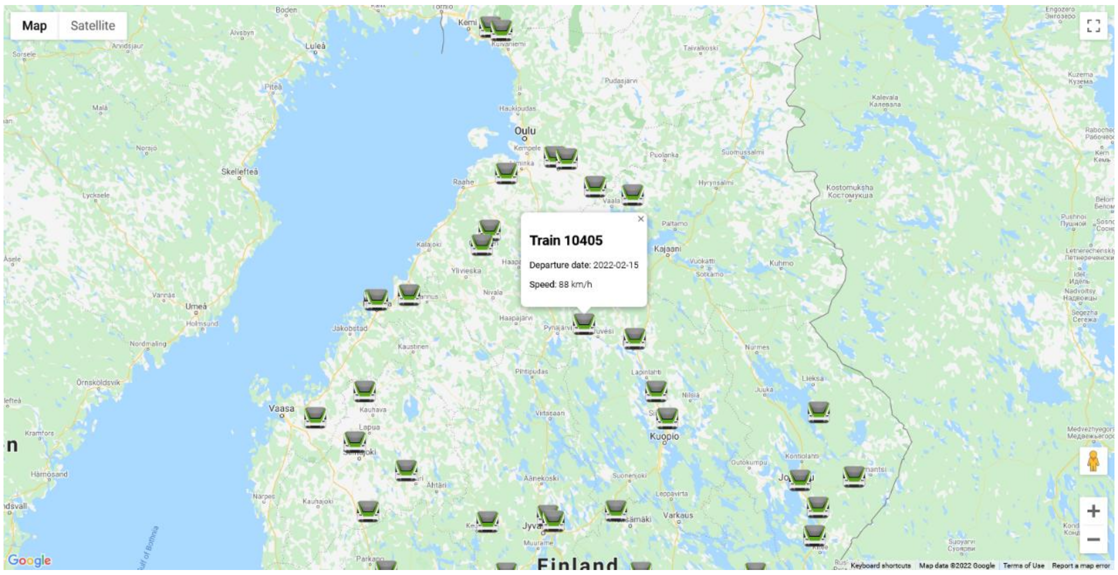


```css
#map {
  height: 100%;
}
html, body {
  height: 100%;
  margin: 0;
  padding: 0;
}
```

FIGURE 9. CSS code

Train marker design choice is based on the free-to-use marker of Stadler FLIRT (Sm3) train currently used in Finland in the Helsinki region with VR colours to resemble InterCity trains too (PICTURE 6.).

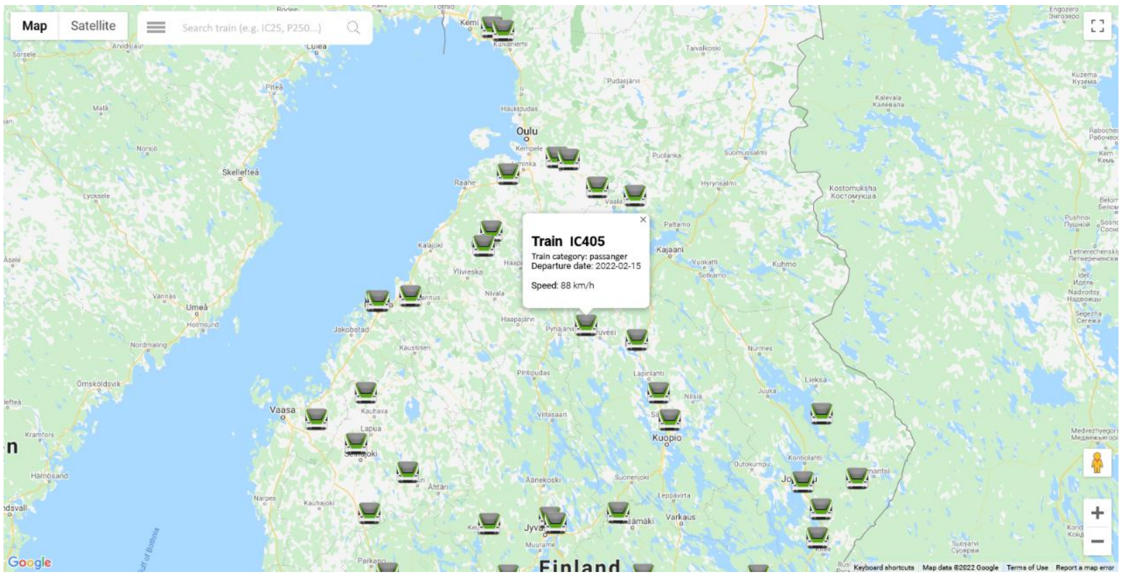
PICTURE 6. Train marker



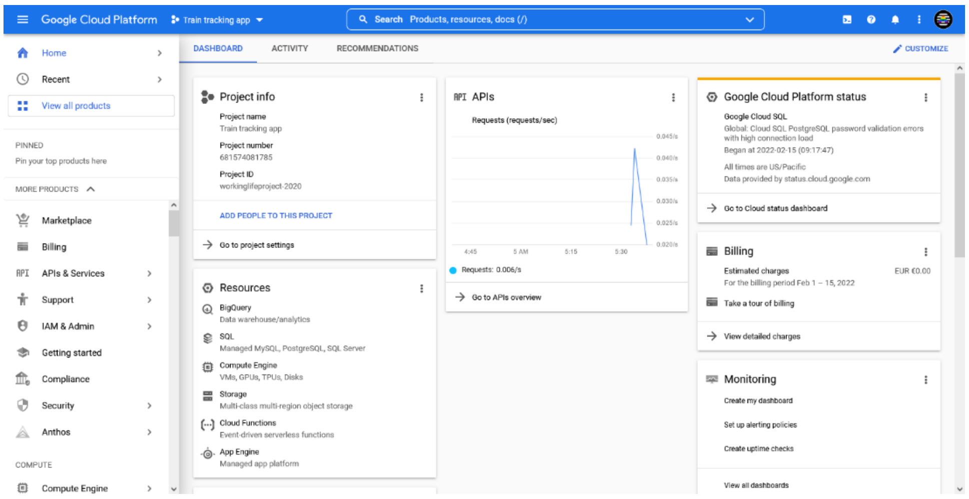
Train 10405
Departure date: 2022-02-15
Speed: 88 km/h

PICTURE 7. User interface

Usage of Google maps makes it familiar to users, further development of user interface is expected to be based on similar logic and design as Google maps itself and depend on the features added. Main possible update features can include search for a specific train, train filtering on the map by its type, different train logos to indicate different train types, route train took, area zoning (PICTURE 8.).


Map
Satellite
Search train (e.g. IC25, P250...)
Train IC405
Train category: passanger
Departure date: 2022-02-15
Speed: 88 km/h
Oulu
Kajaani
Umeå
Vaasa
Kuopio
Finland
PICTURE 8. Mock-up of further developed user interface

This project base is Google maps, so it requires working with Google maps API. All of the Google APIs are located on the Google Cloud Platform. It is a part of Google Cloud and includes cloud infra-structure, Google Workspace, the enterprise version of Google operating systems, APIs and enterprise mapping services (PICTURE 9). Through that platform, it is possible to enable APIs, get API key, monitor it and enable billing. Using Google Maps API costs money and depends on the number of requests to the API. But u also get free usage worth 200$ (28,500 map loads per month).



Project info

Project name
Train tracking app

Project number
681574081785

Project ID
workinglifeproject-2020

Resources

BigQuery
Data warehouse/analytics

SQL
Managed MySQL, PostgreSQL, SQL Server

Compute Engine
VMs, GPUs, TPUs, Disks

Storage
Multi-class multi-region object storage

Cloud Functions
Event-driven serverless functions

App Engine
Managed app platform

APIs

Requests (requests/sec)

Requests: 0.006/s

Google Cloud Platform status

Google Cloud SQL
Global: Cloud SQL PostgreSQL password validation errors with high connection load
Began at 2022-02-15 (09:17:47)

All times are US/Pacific
Data provided by status.cloud.google.com

Billing

Estimated charges EUR €0.00
For the billing period Feb 1 – 15, 2022

Monitoring

PICTURE 9.  Google Cloud Platform for the current project

Use case diagram identifies roles involved in web application and how they interact with functions. In the current version of the project, there is 3 roles and 3 functions. Roles involved are user, Google Maps API, and Digitraffic API. When a user loads a web application, Maps API simultaneously loads a map and adds train markers based on location data, provided by Digitraffic API. User can click/tap on each train icon to get some data about this train including current speed, number, and departure date.

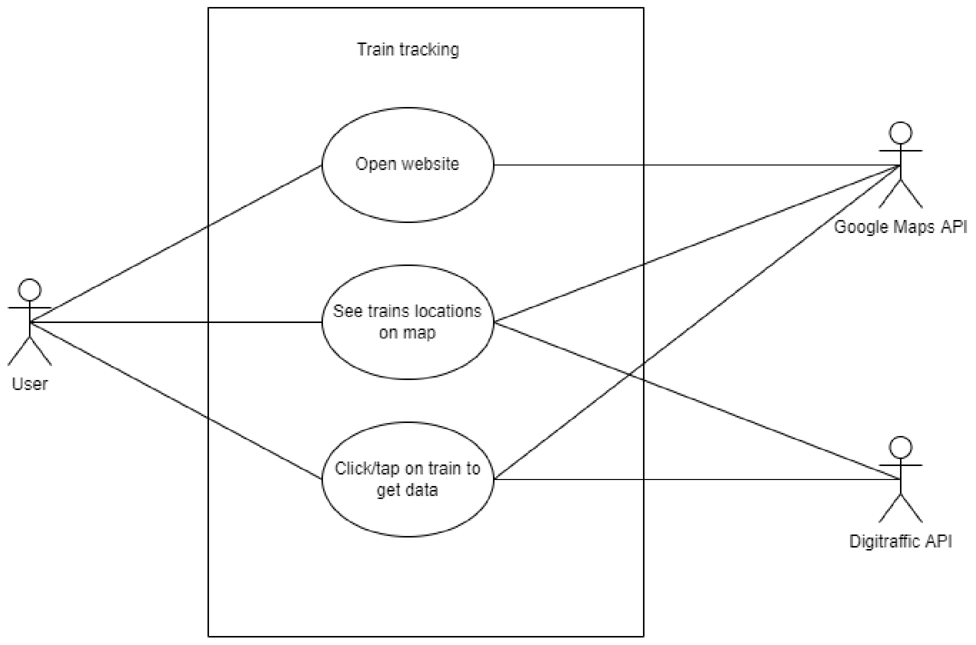
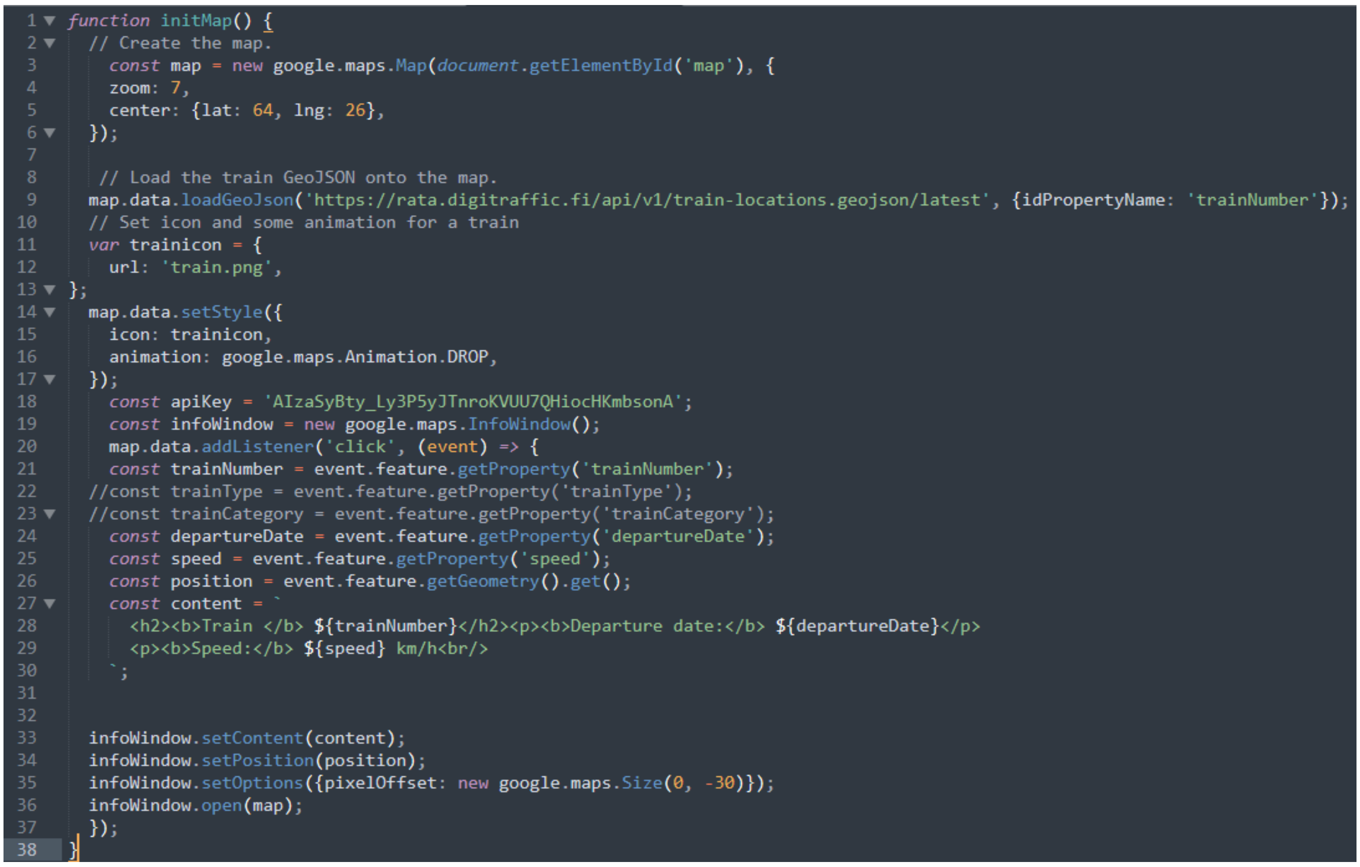


FIGURE 10. Use case diagram

To load map function initMap() is used. It contains all other code related to map implementation. A map.data class with loadGeoJson method to load train data through URL. A trainicon variable that is assigned to a custom icon file. A map.data class with setstyle method to set custom icon using trainicon variable and drop animation. An infoWindow const declaration to create an InfoWindow for each train icon. A map.data class with event listener so, that when the user clicks/taps on a train icon event with getProperty method adds data to icon InfoWindow with data from properties provided by Digitrafic API in GeoJSON file. Each const loads single property of train and const content is to position it properly in InfoWindow. InfoWindow constructor with different methods. A setContent method to assign content, shown in each infoWindow. A setPosition method to make each infoWindow locate properly to its train icon. A setOptions to customise infoWindow e.g., size. An open method to open infoWindow on the given map. (FIGURE 11)



```javascript
function initMap() {
  // Create the map.
    const map = new google.maps.Map(document.getElementById('map'), {
    zoom: 7,
    center: {lat: 64, lng: 26},
  });

   // Load the train GeoJSON onto the map.
  map.data.loadGeoJson('https://rata.digitraffic.fi/api/v1/train-locations.geojson/latest', {idPropertyName: 'trainNumber'});
  // Set icon and some animation for a train
  var trainicon = {
    url: 'train.png',
};
  map.data.setStyle({
    icon: trainicon,
    animation: google.maps.Animation.DROP,
  });
    const apiKey = 'AIzaSyBty_Ly3P5yJTnroKVUU7QHiocHKmbsonA';
    const infoWindow = new google.maps.InfoWindow();
    map.data.addListener('click', (event) => {
    const trainNumber = event.feature.getProperty('trainNumber');
  //const trainType = event.feature.getProperty('trainType');
  //const trainCategory = event.feature.getProperty('trainCategory');
    const departureDate = event.feature.getProperty('departureDate');
    const speed = event.feature.getProperty('speed');
    const position = event.feature.getGeometry().get();
    const content = `
      <h2><b>Train </b> ${trainNumber}</h2><p><b>Departure date:</b> ${departureDate}</p>
      <p><b>Speed:</b> ${speed} km/h<br/>
    `;


  infoWindow.setContent(content);
  infoWindow.setPosition(position);
  infoWindow.setOptions({pixelOffset: new google.maps.Size(0, -30)});
  infoWindow.open(map);
  });
}
```

FIGURE 11. JavaScript code

# 7 CONCLUSION

This thesis intention was to create a single page application to show train location in Finland on Google Maps with the use of Digitraffic API to get train location and data. To achieve that, the Google Maps Cloud platform and Google Maps API was used with methods, classes, and constructors to get data provided by Digitraffic API and load it onto the map. Lack of knowledge led to np further development with an only main feature provided.

During the development and theoretical process, the knowledge about REST API, what applications can be called RESTful rose. Understanding of how web maps work, the way they get data and work with it is higher. Got a picture of what JSON is and its special format GeoJSON. Web application architecture, and its types were described.

This project still needs improvements and extra features to become a project it is supposed to be by all the requirements. At the moment all train markers are static and to change their positioning user needs to reload the page. It would be better to make these markers dynamic with positioning changed simultaneously. Though GeoJSON contains some information about trains like train number, it is difficult to identify, which type train is: InterCity, Pendolino, or cargo. Some of the supposed features include some train location reporting, an account management system that can be developed to allow user login and work with some features.

## REFERENCES

Astari S. 2022 “What Is HTML?”. Available at: https://www.hostinger.com/tutorials/what-is-html#What_Is_HTML. Accessed 21.02.2022

Baldric Mosley 2014 “Architectures”. Available at: https://slideplayer.com/slide/7640839/. Accessed 21.02.2022

Christoph Körner 2016 “Learning Responsive Data Visualization” Birmingham.

Digitraffic 2022 “Service Overview”. Available at: https://www.digitraffic.fi/en/service-overview/. Accessed 22.02.2022

Google Maps 2021 “About Google Maps”. Available at: https://www.google.com/maps/about/#!/. Accessed 22.02.2022

Google 2021 “Adding a Google Map with a Marker to Your Website”. Available at: https://developers.google.com/maps/documentation/javascript/adding-a-google-map. Accessed 16.02.2022.

HERE 2022 “About us”. Available at: https://www.here.com/company/about-us. Accessed 25.02.2022

HERE 2022 “HERE Base Plan Pricing”. Available at: https://www.here.com/pricing. Accessed 21.02.2022

Howard Butler (Hobu Inc.), Martin Daly (Cadcorp), Allan Doyle (MIT), Sean Gillies (UNC-Chapel Hill), Tim Schaub (OpenGeo), Christopher Schmidt (MetaCarta) 2008 “The GeoJSON Format Specification”. Available at: https://geojson.org/geojson-spec.html. Accessed 21.02.2022

Ilari Léman 2018 “CREATING A BUILDING HEALTH IOT APPLICATION”. Available at: https://www.theseus.fi/bitstream/handle/10024/151039/Leman_Ilari.pdf. Accessed 16.02.2022.

Michael Hauben 1998 “View of Behind the Net: The Untold Story of the ARPANET and Computer Science (Chapter 7)”. https://firstmonday.org/ojs/index.php/fm/article/view/612/533. Accessed 25.02.2022

Mozilla 2021 “HTTP”. Available at: https://developer.mozilla.org/en-US/docs/Web/XML/HTTP. Accessed 25.02.2022

Mozilla 2022 “What is a URL?”. Available at: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL. Accessed 20.02.2022

Mozilla 2022 “XML introduction”. Available at: https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction. Accessed 21.02.2022

OpenStreetMap 2022 “About OpenStreetMap”. Available at: https://wiki.openstreetmap.org/wiki/About_OpenStreetMap. Accessed 25.02.2022

OpenStreetMap wiki 2020 “API”. Available at: https://wiki.openstreetmap.org/wiki/API. Accessed 21.02.2022

Oreilly 2022 “Chapter 1. From Hypertext to Hyperdata”. Available at: https://www.oreilly.com/library/view/restful-rails-development/9781491910849/ch01.html. Accessed 21.02.2022

Oracle 2015 “3 Using the RESTful Interfaces”. Available at: https://docs.oracle.com/cd/E50778_01/doc.60/e50769/rst_interact.htm#SGAPP968. Accessed 16.02.2022.

RapidAPI Staff 2021 “The Top 10 Mapping & Maps APIs (for Developers in 2018)”. Available at: https://rapidapi.com/blog/top-map-apis/. Accessed 21.02.2022

Red Hat 2020 “What is REST API?”. Available at: https://www.redhat.com/en/topics/api/what-is-a-rest-api. Accessed 25.02.2022

REST API Tutorial 2021 “HTTP Methods”. Available at: https://restfulapi.net/http-methods/. Accessed 16.02.2022.

REST API 2021 “What is REST”. Available at: https://restfulapi.net/. Accessed 16.02.2022.

Ryte Wiki 2022 “Web Architecture”. Available at: https://en.ryte.com/wiki/Web_Architecture. Accessed 16.02.2022.

Susan Goodwin 2019 “Web Technologies Basics”. Available at: https://slideplayer.com/slide/13650486/. Accessed 21.02.2022

Skillcrush 2022 “Tech 101: The Ultimate Guide To CSS”. Available at: https://skillcrush.com/blog/css/. Accessed 21.02.2022

Sterling Quinn, John A. Dutton e-Education Institute, College of Earth and Mineral Sciences, The Pennsylvania State University 2022 “System architecture for web mapping”. Available at: https://www.e-education.psu.edu/geog585/node/684. Accessed 16.02.2022

Swapnil Banga 2021 “What is Web Application Architecture? Components, Models, and Types”. Available at: https://hackr.io/blog/web-application-architecture-definition-models-types-and-more. Accessed 16.02.2022.

Wikipedia 2021 “Web Services Description Language”. Available at: https://en.wikipedia.org/wiki/Web_Services_Description_Language#Example_WSDL_file. Accessed 21.02.2022

Yandex 2022 “Products and features”. Available at: https://yandex.com/dev/maps/mapsapi/. Accessed 21.02.2022