

Third Party Web API Implementation in Web Application Development



Author(s) Title Number of Pages Date	Humberto Hetu Bampende Third party web API implementation in web application development 31 pages + 2 appendices 15 April 2015
Degree	Bachelor of Engineering
Degree Program	Media Engineering
Specialisation option	Digital Media
Instructor(s)	Ilkka Kylmäniemi, Senior Lecturer
<p>The rapid expansion of the MunaEggsPress business has caused the need for a presence online and a booking system to facilitate the management of the logistics behind the delivery of goods and to provide sales points in advance as available options to customers.</p> <p>As a result, this thesis aim to provide a part of the solution to the booking system for the sales point locations, a login alternative, and messaging modules using third-party APIs approach. Previously, the business essentially used Facebook as way to communicate with their customers and to inform their next sales points. Some customers were active on the social media and some not. Ignoring a group that does not use social media would result in a huge loss for the growing business so this thesis tackles these issues in three steps.</p> <p>First, the purpose was to create a website for the business to interact with their customers, which in this case means a frontend and backend application. This directly provides an alternative to the social media the company long relied on.</p> <p>Second, another purpose was to provide a solid location system using the Google Maps API and geocoding including a database structure that handles all information and data that should be manipulated and made available to the public.</p> <p>Third, the final goal was to implement an alternative login system using the Facebook API and to implement the Multisender API for text messaging.</p> <p>The combination of the named third party APIs resulted in a mash-up application that takes advantage of powerful tools available to developers to provide quality services to the client.</p>	
Keywords	API, REST, RESTful, Web Service, Web Development

Contents

List of Abbreviations

1	Introduction	1
2	API Theory	2
	History	3
	Background	4
3	API Types	5
	Web Service	5
	RESTful API	5
	Data	7
	Testing Tools	7
4	Google APIs	8
	Data as Service	9
	Map JavaScript API v3	10
	Geocoding API	11
5	Facebook API	13
	System Login	14
	Authentication	14
	Database	15
	Configuration	16
	Implementation	16
	Checking the Login Status	17
	Permission and API Calls	18

Graph API	20
Logout	21
6 Multisender API	22
POST	22
Status-Code	25
GET	26
PUT	26
DELETE	26
Multisender Endpoint	26
Accept 27	
Content-Type	27
Multisender Authentication	27
Multisender SMS	28
New SMS 28	
Groups 29	
7 Conclusion	31
References	33

Appendices

Appendix 1. Response Body of a Successfully Created Text Message

Appendix 2. Response of an Account list

List of Abbreviations

AJAX	Asynchronous JavaScript Object
PHP	Personal Home Page
MySQL	My Structured Query Language
HTTP	Hyper Text Transfer Protocol
CRUD	Create, Read, Update and Delete
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JSON	JavaScript Object Notation
URL	Uniform Resource Locator
XML	Extensible Markup Language
API	Application Programming Interface
UI	User Interface
GUI	Graphic User Interface
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
WSDL	Web Service Description Language
SDK	Software Development Kit

1 Introduction

According to the Finnish Commerce Federation, the e-commerce growth rate has accelerated by about one-fifth in the last three years. E-commerce involving goods and services grew by almost 9 % in Finland in 2013. This is proof of the importance of online visibility in the competitive market of goods and services. [1.]

MunaEggsPress is not an exception when it comes to consolidating the position of the company in the online market by offering customers an alternative way to how the company have communicate and reaches to the customers. The business of MunaEggsPress used to rely only on Facebook as the main medium for communication, where they announced their next trajectory, which included sales points where their customers could meet the company representatives and purchase fresh products from the farm.

With the rapid growth of MunaEggsPress and the huge media exposure, they now acknowledge the need of an online booking system to facilitate the management of the logistics behind the transportation of goods, reaching out new potential customers, and giving an alternative option of communication to those who do not use Facebook.

This thesis discusses third party APIs implementation in web development and how third party APIs can be of great help to companies like MunaEggsPress. The first chapter introduces the subject. The second chapter explains the terms and dives into the history and background of APIs. The third chapter describes types of APIs and data transfer. The fourth, fifth, and sixth chapters elaborate on the subject and demonstrate implementations of third party APIs as the solution to what the client asked from us by the client. The seventh and last chapter concludes the thesis.

It is impossible to talk about third party APIs in this study without mentioning the web application that hosts the modules itself. Regardless of the design and the look of the website, functionalities are the focus. The functionalities this thesis will focus on are the incorporation of third party APIs in the development. The main application that hosts the result of this thesis was developed in modules and this thesis will have a closer look to at the third parties APIs used in the final year project the thesis describes and what has influenced the choice of the following.

2 API Theory

API is an acronym for Application Programming Interface, in computer programming and web development. In simple terms, it is defined as an interface that enables a software program to interact with other programs. [2.] It is a set of pre-defined methods, accessible by chosen protocol, regardless of the execution of the output.

The term can also be broken in three parts for a better understanding of the acronym.

Application

The application is a set of functions written in some programming language that works together following some logic to execute certain tasks. As opposed to humans, applications do not need a graphical point of interaction to access methods or functions used by other applications.

Programming

Programming can be defined as the process of writing a programming language logically to perform a task.

Interface

The interface is the way through which people interact with programs [3]. Most of the programs humans interact with have a graphical or visual user interface that consists of a window with buttons to send commands.

An API can be understood as a point of interaction for an application that wishes to use other application functions in order to perform a particular task. APIs are exposed for developers to make use of them. In other words, developers connect their programs and take advantage of services offered by those API providers. Some are free of charge and some services need to be purchased.

History

In order to understand the concept of web APIs, it is important to look back at the period when this concept was introduced sometime before the year 2000, slightly before the .COM bubble. Software developers in need of standard ways to create a bridge between applications and allowing them to communicate with each other lead to the development of APIs. Unlike earlier existing models of software communication, APIs came as a solution to this preoccupation because it utilised the ready proof HTTP protocol.

APIs originate from the software development, during the Service Oriented Architecture (SOA) movement, which sometime around 2000. A portion of the SOA experiment left the industry and found a more fertile environment in the world of start-ups business. [3]

SalesForce and eBay are some of the pioneers by launching their first distinct APIs in 2000, followed by many others the following years.

"The concept of an API pre-dates even the advent of personal computing, let alone the Web, by a very long time! The principal of a well documented set of publicly addressable "entry points" that allow an application to interact with another system has been an essential part of software development since the earliest days of utility data processing. However, the advent of distributed systems, and then the web itself, has seen the importance and utility of these same basic concepts increase dramatically." - Martin Bartlett [4.]

APIs are picked according to the service they provide and their reliability. A stable API, meaning one that maintains its services and is well documented, will see the amount of its clients increasing considerably. An unstable API on the other hand, will fade away for the only reason that bugs and malfunctions in applications will force clients to terminate with the bad service.

Background

APIs appear to be everywhere. In fact, most of the heavy weight applications from the social media to e-commerce use them extensively. Amazon is one of the pioneers in making available their API to developers who intended to build e-commerce storefront that sells items listed on www.amazon.com their. [4]

In 2000, to respond to the growing number of their licensed partners and developers, EBay launched the eBay Application Program Interface (API), along with the eBay Developers Program. EBay goal was to standardize how client applications interact with Ebays services and made it possible for their clients to implement functionalities and specialized operations not otherwise afforded by the eBay interface. [5.]

After seeing applications developers built without Twitter's help, Twitter revealed their internal methods to the developer community in 2006 with The Twitter API launch to empower developers and allowed third part applications to host Twitter services. [7.]

SalesFroce also published in 2000 their enterprise-class, web-based, sales force automation as "Internet as a service". The API came as a solution for enterprises to share data across their different business applications. [8.]

The expansion of APIs on the Internet has not been easy for most of the services. From the shy start of Google Maps in 2000 to the launch of the Facebook long-awaited development platform and API version 1.0 in 2006, the web community has learned more and some API service providers had to re-think the way they published their methods. Twitter for instance had to force their developers to switch to different authorization methods. Some developers even qualified the Facebook API of unstable so forth.

While getting more standardized, APIs have opened new possibilities for developers out there to experiment and create interesting mash-ups. Serving data from different origin on the same application became easier and faster. There is no limitation to the amount of APIs, which can be mash-up to create an outstanding application if not to be aware of the performance. That is beyond the scope of this report

3 API Types

Web APIs are mostly available on two levels. A high level is accessible with specific languages such as Java or JavaScript. Low level uses direct HTTP requests to series of endpoints for accessing different parts of the application interface to their client. [9.]

Regardless of the type of the web API, the core remains in the HTTP request. The client request will contain necessary information to be passed to the method invoked on the API for processing and information about the client browser.

Web service APIs are bound to provide good documentation to their clients, which is crucial for a better understanding of the services they dispense and its strength and its limitations. Failing to provide a good documentation will result in an incomprehensible API to use for developers.

Web Service

A web service is a set of functions deployed on a server, which can be accessed by computer programs on the clients through the Internet. Those functions can be utilized to designate and retrieve the target information requested by the computer programs ran by the client. Web Services are not limited to the use of browsers or hypertext markup language (HTML). Thus, Web Services are sometimes referred to as *application services*. [10.]

RESTful API

The word REST is an acronym for Representational State Transfer. REST is an architecture design pattern. RESTful designates a simple interface that transfers data over a standardized HTTP protocol without the need of an extra layer protocol, as opposed to Simple Object Access Protocol (SOAP), which makes use of other schemes like FTP and SMTP. However it is a little low in performance. Each resource can be identified by its unique Universal Resource locator endpoint [11].

RESTful Web services are web services that are developed following REST architecture rules. Therefore, they use common HTTP methods such as GET to read resources or contents on the API, POST to create contents, PUT to update contents, and DELETE contents [11]. The whole operation is commonly referred to in one word which is CRUD.

Skeleton of a RESTful HTTP request:

`http://domain/methodname.format?access=¶m1=¶m2=&...¶m N=...`

- domain: Indicates the address of server
- methodname: Refer to name of the method being invoked
- format: The format of return data
- access: Developer's unique 'Access Token'
- param1...param N: Parameters of the method.

Figure 1 is a simplified graphical representation of the use of a RESTful API by multiple developers in multiple applications and published on different mediums with unique experiences for each user, yet consuming the same information from the same API .

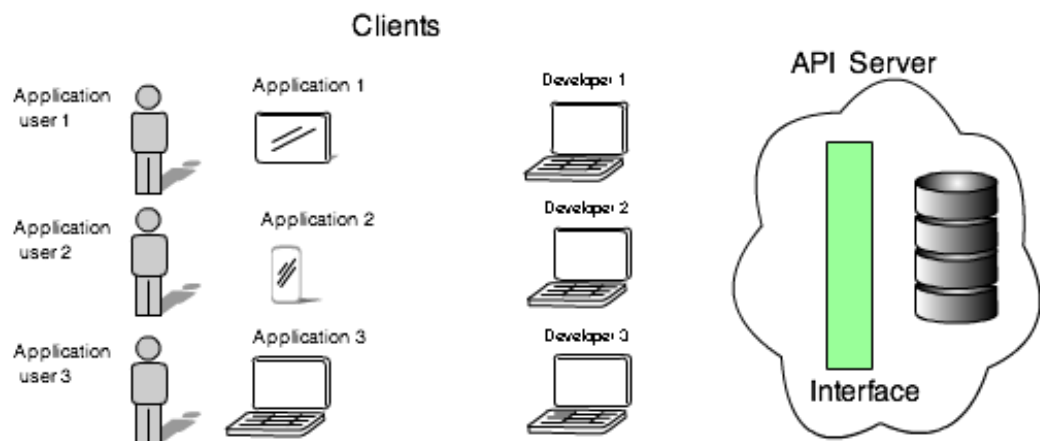


Figure 1. API consumption

Data

The whole idea behind APIs is to exchange data and manipulate them. Resources are available in various formats depending on the service provider. For efficiency reasons of how fast the data is transferred across, data are not most of the time packed in human readable format. Back in the early age of APIs, the XML format was pretty much the standard data format. With time came JSON (JavaScript Object Notation), which gained a lot of ground, and it is becoming a standard due to its flexibility and lightweight compared to XML. In fact, some APIs only have one format available, JSON, which is much easier to parse and has a direct in-memory representation in most programming languages.

Testing Tools

A variety of tools is available on the web, helping to ease the process of testing and visualizing results of requests and to translate the retrieved data into a more human readable content. Plugins can be found for most of the modern browsers, such as Google Chrome Postman, Mozilla Firefox Poster.

Testing tools are just for testing purposes before implementing the calls and the data processing in the application. They are helpers that help developers to speed up the development. Consoles are also excellent tools while working with data transfer of any nature.

4 Google APIs

Google offers a range of flexible and powerful API web services that can be utilised in numerous types of application projects on different platforms. Google APIs are reliable and well documented, among them the Maps API. Google Maps JavaScript is customizable and interactive. APIs enrich users experience on applications by offering visualization for places on the globe.

For the project described in this thesis, we the map API was chosen to handle maps and the geocoding API to handle coordinates generated from the map. The combination of those two APIs results in a powerful mash-up, which can be an application on its own.

The goal is to provide a backend tool for MunaeggsPress to set specific locations on the map and to store those locations as sets of trajectories, which can be published on the frontend for their customers. This backend is part of the solution to arm the web application manager with a flexible and easy to use tool, as opposed to long lines of daunting code he/she might not be familiar with.

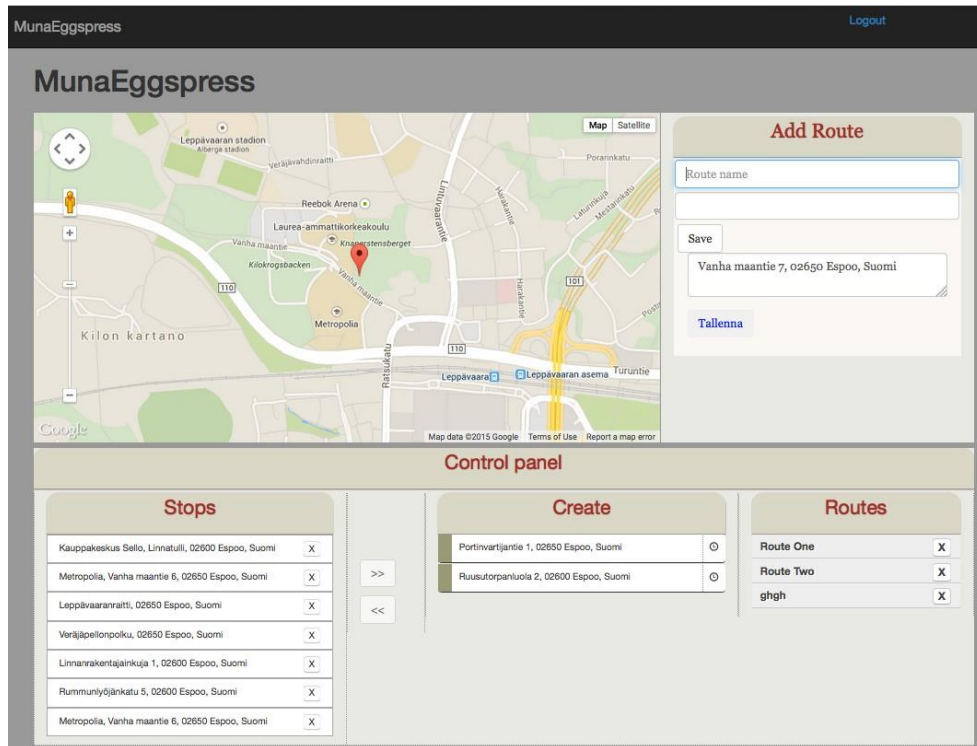


Figure 2. Backend administrator panel. Screenshot [25].

Figure 2 shows the administration tool. The administrator has the means to plan his/her sells spots on the map and store trajectories as he pleases. A marker on the map represents a sell's stop. The stops list in the panel can be re-ordered or simply deleted. Stops can also be grouped into sets of trajectories and published on the frontend to indicate customers next sells stops.

The administrator panel user interface is just one way of arranging the data that is loaded on this page. With the idea of sharing some of methods in mind, we have been thinking on ways to servicing the next available routes for example.

Data as Service

The data saved and displayed in the application is not bound to this specific web application. If we were about to make available some data the application uses to other applications that wish to use the same information on their site as they please. One way to do it is by servicing the date through an endpoint as shown in figure 3.



Figure 3. Data exposed from our web service

Twitter, Flickr, Facebook, and others do something similar. Their own applications use the same API they share. This is in a minimalistic scale, just to show how the data in JSON format can be shared as a service as figure 5 shows. The routes panel at the right lower corner of the administration panel (see figure 4) can be called by other developers who wish to use this information on their applications.

Map JavaScript API v3

The Google Map API is mostly used to quickly display maps interfaces. It is accessible by HTTP request from the application. The Map API is a useful tool when it comes to accessing geographic data for applications that utilise maps like this control panel (see figure 4).

To use the Google API, developers should register and acquire a key, necessary for the use of the service. Google advises developers not to include the key in the request [12.]. Google API provides a step-by-step documentation, and reference information including code examples, which are available on Google developer's site on how to get maps loading and start interaction with them. There are other choices for the development environment and languages. JavaScript API was chosen for the project

The Google Map API enable the application to access the coordinates latitude and longitude of any point the user chooses on the map using JavaScript methods as shown in the code below (see listing 1).

```
google.maps.event.addListener(map, 'click', function (event) {  
    var lat = event.latLng.lat();  
    var lng = event.latLng.lng();  
})
```

Listing 1. JavaScript Map event listener.

This will benefit the application in the long term. It was mentioned earlier that the client (MunaEggsPress) picks sells spots on the map. Those spots are locations where the mobile shop will stop and sell their merchandises.

With that in mind, we invoke the geocoding Google API, which works perfectly in combination with the Google Map to achieve the goal.

Geocoding API

The Google geocoding as the name implies does all the heavy lifting when it comes to converting a human-readable address to coordinates like latitude and longitude, which the map application understands.

The process of converting coordinates to human-readable address format as we commonly use them is called reverse geocoding.

The geocoding request resemble the following:

<https://maps.googleapis.com/maps/api/geocode/output?parameters>

The *output* in the geocoding URL can either be in XML or JSON format. The *Parameters* include the coordinates latitude and longitude. The language can also be specified as a parameter.

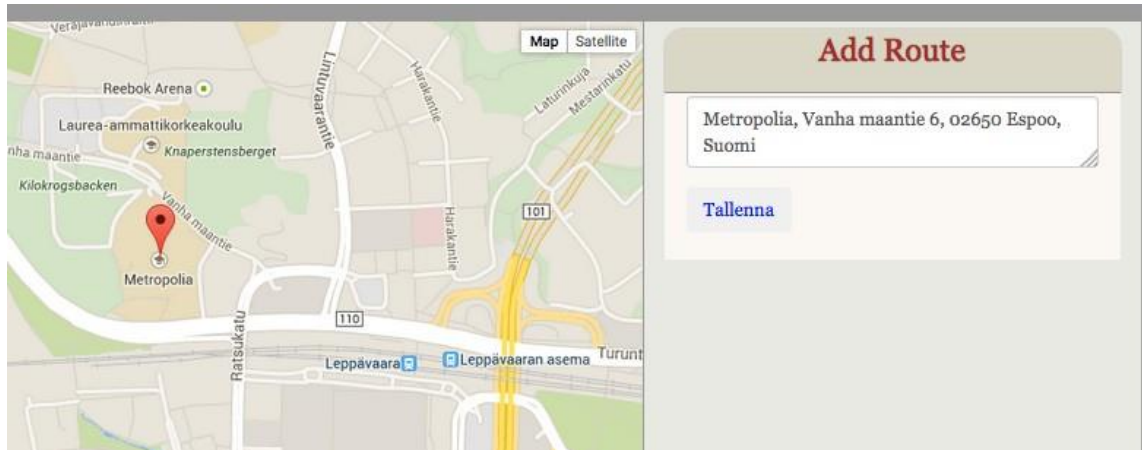


Figure 4. Conversion of instant coordinates to address. Screenshot [25].

The conversion came in handy for this particular project. It is important to note that when the administrator drops a marker on a specific point on the map, the geocoding API gets the marker's coordinates from the map API. After the geocoding API grabs the coordinates and processes them by converting them to a corresponding address on the map equivalent to the postal address ("Metropolia. Vanha maantie 6, 002650 Espoo, Suomi ") as shown in figure 4. The administrator is not concerned of the process in the background. The two APIs combined together are communicating with one another transferring data. This is a typical illustration of how APIs can be used efficiently in mash-up applications (See figure 5).

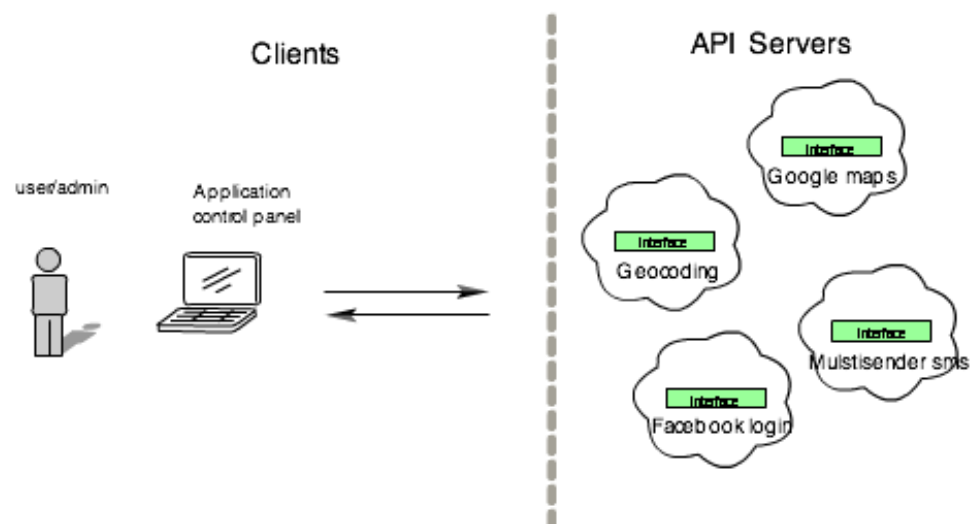


Figure 5. Illustration of the backend mash-up.

5 Facebook API

Facebook RESTful API v1 was launched in 2007 along with the Facebook development platform. The release allowed developers to write social applications and mash-ups to empower Facebook users. Although considered as unstable by some back in the days, the API has developed considerably by giving developers a free hand in developing applications that can talk to each other and create new exciting ways of Facebook users to interact [15, 2]. The Facebook platform has become an essential tool in the Facebook social network development itself. The platform offers multiple options for application developments. Among these options, the login API has been used in this project.

The application built in this study has a purchasing system. This process requires a login in order to authenticate customers. Besides the application's own login system, we need to provide alternative login for customers who do not wish to sign in the system and avoid the registration process, which logically involves filling information in forms. This project will utilise the Facebook login API as an alternative to the site login system as mentioned earlier.

A good candidate for alternative login is the Facebook login, which is widely used for application logins. There were over 10 billion Facebook logins in 2013, and it is now integrated with over 80% of the top hundred online stores on IOS and Android applications in the US. It is going to be one of the first choice social media logins in the world. [12.]

The alternative Facebook was implemented at the model of what have already been done on numerous applications. Salesforce for instance offers Google plus login as an option for login, side to side with the Facebook button, LinkedIn, and their own login system. <https://www.salesforce.com/eu/form/demo/demo-sales.jsp?d=70130000000ryrm&internal=true>.

Figure 6. Login panel showing the alternative logging option

One way to interact with the user is to provide a form, where the user can submit to us required information for processing before accessing what is considered as sensible content of the application, that needs a certain level of security.

System Login

The login system will rely on the application's own user authentication first, and then integrate the Facebook login as the alternative for login (see figure 6). The authentication will help identify users and respectively determine their privilege level in the application.

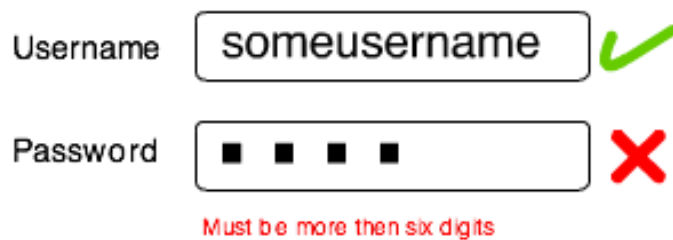
Authentication

The authentication is the process of validating the user information. It is important to check what the user inputs against what has been previously saved in the database. If there is a match, the application allows access to the appropriate content.

This process is one of the most challenged on the web, for the obvious reason that it gives access to sensible content that can be misused by malicious individuals.

Although the server validation is the most important process using scripting language such as PHP, the validation at client side is essential. Good practices recommend that validation is done first at the client side emphasising the user to write inputs in the correct format with the help of AJAX for instance before the user submits the form to the server for a further validation.

The reason why we used AJAX is the capabilities it offers. We will be able to update specific area of the DOM without having to submit the form to the server for final validation. This method is not to be confused with the secure validation the data must go through. The advantage of client validation is that it is a filter and simultaneously a guide for users.



Username ✓

Password ✗

Must be more then six digits

Figure 7. Live validation. Screenshot [25].

Database

For the application authentication process, the application needs to store related information about users and be able to check any new information against stored data. This requires a database from which the application can check stored information.

Table 1. Database table for user authentication

Data	Description
ID	Unique identifier number
username	Nickname of the user
password	Secret word for identification
email	User email address
lp	To help recognize users who are not logged in

Table 1 is the representation of the database. The choice of the database is not of much importance for this study, but just to mention, a MySQL database is used.

Configuration

Although the use of the platform is free of charge, a registration as a developer is needed in order to use the platform for development. After registration, the developer can create a new app in the dashboard. This step walks you through the app registration and generates an application ID, which is the application identifier. Facebook App ID is a requirement for developers before starting further development [14].

Implementation

The Facebook platform offers easy and comprehensive step-by-step documentations on how to configure a Facebook application. There are various choices of development platforms for the Facebook Login API. Platforms such as Android, IOS, and for the web-based are supported. The application in this study is for a web-based platform. Therefore, the choice of the web platform workflow. The Facebook's JavaScript SDK is the recommended method to add the login feature to the web applications. [14.]

There is more than one way to implement the login. The target platform determines which flow to follow. Likely, for us there are two flows possible for websites. The JavaScript library and the PHP 4.0 and above are libraries for accessing the graph, and taking advantage of the Facebook login.

The PHP library adds server-side functionalities to the application, that already utilises the JavaScript SDK at the front end. Unlike the JavaScript SDK, the PHP files need to

be hosted physically on the server for the login. The PHP SDK can be downloaded from Facebook documentation [15].

Facebook recommends the following six steps in the implementation of the login API regardless of which library is used feature:

- Check the login status
- If the user is not logged, invoke the login dialog and ask for a set of permissions
- Verify the user identity
- Store the resulting access token
- Make API calls
- Provide the logout option to the user

Checking the Login Status

It is very important to verify the status in order to determine if the user has already logged into Facebook or not. Any time the user log into Facebook; a session is saved by the browser. The same session helps the SDK to determine the status and than proceed to the next step.

On window load, the application initializes the JavaScript SDK. As followed in the code below:

```
window.fbAsyncInit = function() {  
  FB.init({  
    appId      : '3123145677',  
    cookie     : true,  // enable cookies to allow  
    xfbml      : true,  // parse social plugins c  
    version    : 'v2.2' // use version 2.2  
  });  
};
```

Listing 2. Initialization function.

The initialization makes sure the FB object from the Facebook JavaScript SDK is loaded as part of the application setup before invoking any other FB method. It is important to note that the initializer will load the SDK asynchronously. This will not stop the rest of the application from loading normally. Once the initialization is complete, the application can proceed to check the user status.

```
FB.getLoginStatus(function(response) {  
    statusChangeCallback(response);  
});
```

Listing 3. Status check.

A response object is returned from the callback function as such:

```
{  
  status: 'connected',  
  authResponse: {  
    accessToken: '...',  
    expiresIn: '...',  
    signedRequest: '...',  
    userID: '...'   
  }  
}
```

Listing 4. Status response in JSON object.

At this stage, the developer knows whether the user is already logged on Facebook or not. The user can be prompted for login if necessary. The authentication response contains permissions information, how long until the cookie expires, the signed Request, and the user ID.

Permission and API Calls

As shown earlier in listing 3, it is a fact that the authentication response returned from the `getLoginStatus` function contains permission information. The access token is a temporary random string generated by the API. This string references the set of per-

missions the Facebook users grant to a third party application for accessing their content from Facebook.

Developers ask for permission when the user wants to complete an action that requires authorization. To give an example, the user might want to share a post from a third party application to Facebook. When the user expresses the need to use this service by clicking on the share button, it is the right moment for the developer through the third party application to ask for adequate permission for the post to Facebook.

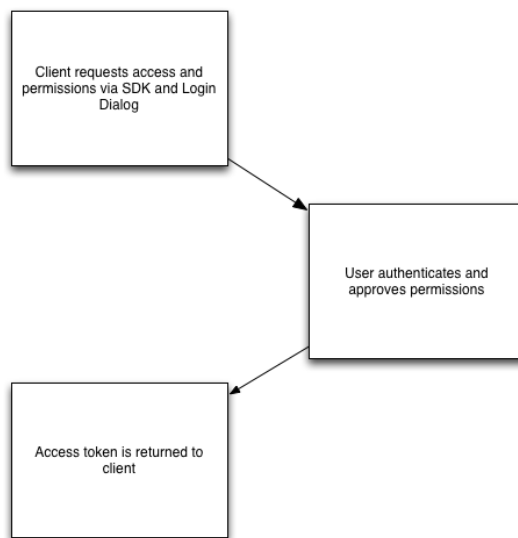


Figure 8. Client Token [16].

Example of an access token:

“CAALQ7EKM3xkBAJPkuWdivaG2u9PoHV6Hd3hPIhKFHgZCQRZCSO4OP4upbcElzFMDfquZADdVW5kw4cgfBvMFEFctlorLV2Feo0fur3lsmqZBAIn6CJmlyRAVL9ZB0w0nBriTuPHpj5n0GxYZCcxCn-ZAcouP06401ghHloNvjJV59INq0Eseupk32vxipF1ScZBsZCsYpgLL0GJMTa5qU5s21Eo1tNZA3S2o0MZD”.

Facebook recommends developers to ask only the minimum permission in the context. If there is a need for more privilege it is possible to ask for more permission. This approach helps application’s user for instance understand the reason of each one of the permissions he or she is about to give [16].



Figure 9. Dialog box asking for permission [15]

Permissions are asked according to the experience the developer wants to share with the user. Users can accept or decline permissions to their information. Developers can ask for additional permissions at any time, even after a person logs in for the first time [15].

Graph API

The graph API is the main tool that allows the communication between third party applications and Facebook. Queries can be performed directly on the console. The Facebook Graph API can also be used as a regulator of data that the application can ask.

All requests to the graph require an access token. This means that an access token can be compromised by malicious individuals for their own agenda. A solution to this is the introduction of the app_secret proof. This parameter can be enabled in the advance section of the Facebook application. This way of setting the app will stop a malicious application from sending spam to Facebook on behalf of the registered application [19].

Logout

It is important to arm users with the way to log out of the application. Logging out is equally important as logging in the application. Otherwise, users do not trust the application, causing all kind of unwanted effects that can follow.

```
FB.logout(function(response) {  
  // user is now logged out  
});
```

Listing 5. Logout snippet.

The code in listing 5, logs the user out of both the Facebook page and the application. It also destroys the access token used in the session. [16]

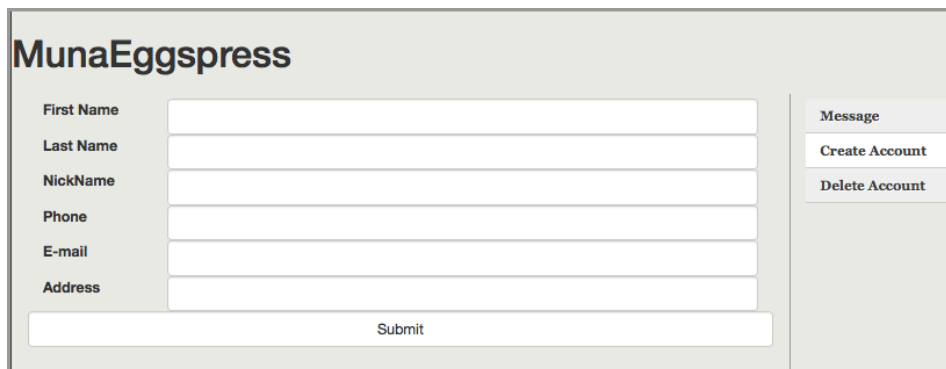
6 Multisender API

Holding an online store is not very different from having an infrastructure store from the perspective of customer communication and satisfaction. The owner of the store must be able to communicate with customers, replay to their questions, and keep them informed about products. This is an easy task when people visit a store's infrastructure. Displays, boards, and pamphlet distribution contribute to getting the message across. Likewise, online stores need to keep customers informed about products but also other activities and promotions. Hence, the need for communication tools. Most customers can be emailed but some prefer getting text messages directly to their mobile phone.

The multisender API is a RESTFul API, which allows it clients to send multiple SMS simultaneously to different mobile phones. It includes functionalities such as creating, reading, updating, and deleting content. It uses classic HTTP: POST, GET, PUT, and DELETE methods [17].

POST

The POST method is used to create content on the multisender service. From the administration panel the administrator fills in and submits the form to create a new account. In the form process, the application sends a POST method along with the data to the multisender API. The endpoint is `http://api.multisender.net/v1/accounts`.



MunaEggspress	
First Name	<input type="text"/>
Last Name	<input type="text"/>
NickName	<input type="text"/>
Phone	<input type="text"/>
E-mail	<input type="text"/>
Address	<input type="text"/>
Submit	
<div>Message</div> <div>Create Account</div> <div>Delete Account</div>	

Figure 10. Administration panel for account creation Screenshot [25].

The administrator creates new accounts by filling in the appropriate form. Fields to be filled are not chosen randomly. The API documentation specifies what parameters to use. Based on that, developers build forms to collect the information from the user. It is

important to note that the developer ultimately decides what information to ask from the user if the API does not require all parameters. Developer X might omit a few parameters when developer Y might want to use all of them in his/her application. This is a typical example of how two applications that utilize the same API have free hands on what experience they would like to share with their users.

Parameters			
Attribute	Description	Type	Validation
firstName	First name.	String	
lastName	Last name	String	
nickName	Nick name	String	
phone	The phone number where the sms or call is delivered to.	phone_number	valid_e164_phone
email	The email address which is used for delivering the email.	Varchar	valid_email
address	Street address	String	
zip	Zip Code	number	
city	City	String	
companyName	Company Name	String	
website	Website URL	URL	
note	Notes	String	

Figure 11. List of parameters provided by the multisender API [17]

The application collects the information entered by the user and store it in a JavaScript object as followed.

```
var params ={
    "firstName": "Mary",
    "lastName": "Robinson",
    "nickName": "Red Paron",
    "phone": "+358912345678",
    "email": "exam-
ple@example.com",
    "address": "Street"
};
```

Listing 6. Account parameters.

```
$.post('http://api.multisender.net/v1/access_token=xxxxxxxxx&accounts?Accept=Application/json&ContentType=Application/json', params, function (response) {  
    console.log(response);  
}, 'json');
```

Listing 7. AJAX call.

With the help of JQuery and AJAX the application can pass the params object using the POST method and send the data to the API in order to create a new user account.

A JSON formatted response is returned from the server if the request was successful. The code below is an example of a successful response:

```
{  
    "accountId": "1856",  
    "firstName": "Mary",  
    "lastName": "Robinson",  
    "nickName": "Red Paron",  
    "phone": "+358443349999",  
    "email": "example@example.com",  
    "address": "Street",  
    "updatedAt": "2015-03-10 12:59:00",  
    "createdAt": "2014-03-10 12:59:00"  
}
```

Listing 8. JSON response from the API.

In case of an error the server returns a status code of 401 with a message. The application will elaborate a little more about the status-code inside the response in the next section.

```
{  
  "error": true,  
  "type": "Authentication",  
  "message": "Your access token is either missing or incor-  
rect. Please          check the X-Access-Token header and try  
again."  
}
```

Listing 9. Error response.

Status-Code

After the request has been sent to the server a response is returned to the client (see figure 11). The response contains descriptive information about the operation at the server side. The Status-code is included in the response and is intended for use by automata. Most importantly, this code defines the class of the response. [16] By taking a glance at the status-code return by the server, we can tell if the operation was a success or a failure.

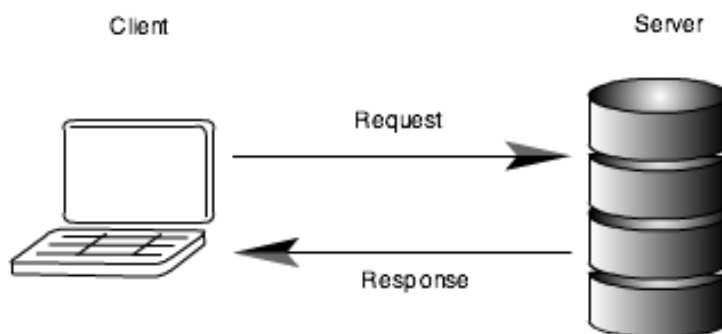


Figure 12. Illustration of the client–server communication

The 2xx class and the 4xx are the most frequently encountered with the multisender communication, 200 for success and 401 for failure due to unauthorized request.

- 2xx: Success The action was successfully received, understood, and accepted.
- 4xx: Client Error The request contains bad syntax or cannot be fulfilled.

GET

The GET method is invoked when fetching the list of existing accounts. This method is used while fetching text message by IDs, and when listing all text messages.

As an example, the endpoint for fetching the list of text messages from the API using the GET method. <http://api.multisender.net/v1/smses>

PUT

The PUT method is invoked when updating content. Whether smses or accounts the endpoint is sent using PUT.

An example of the endpoint for updating text message by IDs using the PUT method. <http://api.multisender.net/v1/smses/:smsId>

DELETE

The DELETE method, as the name indicates, is utilized when deleting content from the multisender API. It applies to both, existing text message and accounts. The response for the delete method contains status-code 204 and is empty.

Example of the DELETE method's endpoint. <http://api.multisender.net/v1/smses/:smsId>

Multisender Endpoint

Like most APIs the multisender is accessed by the HTTP request. The endpoint to the service is: <http://api.multisender.net/v1>

Two header fields must be included in all the calls to the multisender service, the ACCEPT header-field and the CONTENT-TYPE header-field.

Example of an HTTP request for the multisender RESTFul API,
`http://api.multisender.net/v1/Accept=application/json&ContentType=application/json&access_token=123mnb`

Accept

Accept is used to specify which media type is acceptable for the response. It indicates the format of the desired type. Application/json is the only type supported presently. This option will make sense, as other media type will be supported in the future. [17]

Content-Type

The Content-Type entity-header field indicates the media type of the entity-body sent to the recipient [17]. Browsers need this information in order to handle the entity body correctly. The parameters passed are the attribute/value, as shown in the example of the HTTP request above.

Multisender Authentication

One of the requirements for the multisender is the authentication. All requests to the API must contain an access token in order to be authenticated. The multisender checks for a valid key, the access token. The access token can be passed three different ways: using an HTTP header, using a GET parameter, and using a POST parameter. The access token is issued by the service. [17]

The fact that applications can access API interfaces is what makes APIs such a tool of preference for this kind of solution. What the application's administrator will need to do is to write a single message, choose the receivers, and send.



The screenshot displays a web interface for creating a message. On the left, there is a large text area with the placeholder 'SMS goes here...' and a red 'Send' button below it. On the right, there is a sidebar with a 'Message' header and two buttons: 'Create Account' and 'Delete Account'.

Figure 13. Message creation view for the administrator. Screenshot [25].

The rest of the operation is handled by the communication between the web application and the multisender API. The administrator will be able to get feedback of the background operations for success or failure.

Multisender SMS

Maintaining a healthy communication between the business and customers. This has direct impact on the customer and the business owner. It was mentioned at the initial of chapter 6 the reasons why this is important. One might ask, what sets the multisender API apart from normal routine such as sending text messages directly from the mobile phone to customers. The power of the multisender resides in the fact that it is an API. Therefore, this application can access methods automatically if a triggering event is fired in the application. An automatic text message could be sent to remind customers living in the premises of the next sell stop as reminder. A lot more can be accomplished than using traditionally a mobile phone to send text messages.

New SMS

A new text message implies a creation on the server side of a new entry with an associated timestamp, a new ID, a message to be sent to accounts, and the receivers list of accounts to whom the message is sent.

The endpoint for creating a new text message is the POST method which has the following URL: <http://api.multisender.net/v1/smses>

```
{
  "actionTime": "2014-01-0 12:00:01",
  "message": "Hello World!",
  "sender": "0123456789",
  "receivers": [
    {
      "accountId": XXX
    }
  ]
}
```

Listing 10. Request body of the POST method

The server replays either positively with a success message, a status-code of 201, and a newly create message (see appendix 1) including a recipient which received the text message. In case of failure, and an error response, JSON is returned with the status-code indicating the class error, 401 in this case.

```
{
  "error": true,
  "type": "Validation",
  "message": "Validation error.
The Message field is required.</p>\n"
}
```

Listing 11. Error response JSON

The update of the text message is a similar process to the creation except that it utilizes the PUT method.

Groups

The multisender offers the possibility to organize accounts into groups and address them as such. This comes in handy specially when texting different groups of users on the website. For example, users of the same interest for a certain article could be grouped together and selected quickly as one container. Groups work very much similarly as individual accounts for all the core methods.

Parameters			
Attribute	Description	Type	Validation
name	Group name	String	required
description	Description	String	

Figure 14. Parameters for group creation

It is important to note that groups are just containers. Accounts that belong to a group can also be addressed individually or belong to multiple groups. The group is an array containing JSON objects (see figure 17), which are valid accounts.

As an example, the endpoint used in the POST method for adding accounts to group, POST <http://api.multisender.net/v1/groups/:groupId/accounts>.

The server returns a group array that contains accounts (see figure 17):

```
[
  {
    "accountId": 1
  },
  {
    "accountId": 2
  }
]
```

Listing 12. Group array

7 Conclusion

Web APIs help create a bridge for the data transfer between applications. In fact, one of the main uses of a web application interface is to do just that. The role of third party APIs starting from their origin have been extensively discussed throughout this thesis. Early implementations of APIs like Amazone were more of a one on one connection between the service provider and the client. APIs evolved into a more complex network where combined with each other they allow developers to create useful and astonishing applications that enhance their users experience in a short time.

With the boom in Internet device use, it is only normal that the consumption of content and information has increased considerably and will never be the same. It is difficult to imagine how painful it could be to write map applications like described in this thesis without map APIs. There are applications that could never have been thought of without the possibilities made available by data transfer between applications using APIs. The application that was create for MunaEggsPress in the final year project this thesis describes fits into the category of an application that is useful, taking advantage of third party APIs to provide a quality service to their customers.

Some have claimed that APIs are nothing new. It is true that earlier systems preceding APIs have served to create the communication between software, but there has been nothing like APIs in the the way developers use them today. Web APIs have redefined the way developers publish interfaces, and allowing other developers to use their methods and enhance the user's experience. Amazon, eBay, Twitter, Flicker, Facebook, Google and almost all major players on the web have recognized the importance of APIs and continue to work on them to make them even more useful and available to communities of developers, which help by pushing in the same dynamic.

The use of web APIs is quite broad due to their flexibility. With a bit of programming skills and imagination, results can be outstanding applications that give satisfaction to users. One API on one application or many APIs to one application, it is up to developers to determine what services serve better their vision.

Disadvantages

A study on the impact of API evolution on developer's adoption and how client applications react to those changes shows that APIs do evolve to a high rate causing developers to change codes [24.] and it is generally known that bugs are never far when changes happen.

Client applications the security of which depends only on an API can find themselves in situations where security holes exploitable by hackers could be exposed.

It is well established by now that applications built based on a third party APIs will rely heavily on the API's health. As a result, there is a need for constant maintenance and updates. Changes in the application-programming interface will directly affect built applications. The choice of an application-programming interface (API) by the developer is an important step to take with caution.

Developers should consider these facts while designing, just to be prepared to respond immediately in case of emergency.

References

- 1 Finnish Commerce Federation. Press release [online].
URL: http://www.kauppa.fi/eng/press_releases/e_commerce_of_goods_increasing_despite_recession_24804. Accessed 18 February 2015.
- 2 Definition of 'Application Programming Interface API' [online]. 2015
URL: <http://www.investopedia.com/terms/a/application-programming-interface.asp>. Accessed 21 January 2015.
- 3 A Language Engineering System for Graphical User Interface. Information and Communication Technologies [online]. 2006.
URL: <http://ieeexplore.ieee.org.ezproxy.metropolia.fi/stamp/stamp.jsp?tp=&arnumber=1684996>. Accessed 11 March 2015.
- 4 Lane, K. 2013a. API Evangelist: History of APIs [online]. 2015.
URL: <http://history.apievangelist.com>. Accessed 2 February 2015.
- 5 Amazon api documentation. Audience [online]. 2015.
URL: <http://docs.aws.amazon.com/AWSECommerceService/latest/DG/Welcome.html>. Accessed 15 February 2015.
- 6 Functionality and specialized operations not otherwise afforded by the eBay interface [online]. 2006.
URL: <https://blog.twitter.com/2006/introducing-twitter-api>. Accessed 15 February 2015.
- 7 Introducing the Twitter API [online]. 2006.
URL: <https://blog.twitter.com/2006/introducing-twitter-api>. Accessed 17 February 2015.
- 8 Salesforce API documentation [online].
URL: https://www.salesforce.com/us/developer/docs/api_rest/. Accessed 23 February 2015.
- 9 W3c working draft [online]. URLs 2014. URL: <http://www.w3.org/TR/url-1/#concept-url-scheme-data>. Accessed 5 February 2015.
- 10 How Does Web Service API Evolution Affect Clients [online]. 2013.
URL: <http://ieeexplore.ieee.org.ezproxy.metropolia.fi/stamp/stamp.jsp?tp=&arnumber=6649592> Header fields definitions. Accessed 12 March 2015.
- 11 Oracle.com. Introduction to restful web services [online].
URL: <https://docs.oracle.com/middleware/1212/wls/RESTF/intro-restful-service.htm#RESTF105>. Accessed 28 January 2015.
- 12 Facebook Developers. Facebook Login Version [online].
URL: <https://developers.facebook.com/docs/facebook-login/v2.3>. Accessed 11 February 2015.
- 13 Google Developers. Obtaining an API key [online].
URL: <https://developers.google.com/maps/documentation/javascript/tutorial>. Accessed 20 January 2015.

- 14 Graham Wayne. Facebook API. Developer's Guide. SPRINGER. New york 2008.
- 15 Facebook Developers. Facebook Login version [online]. Login for the Web with JavaScript SDK. 2015.
URL: <https://developers.facebook.com/docs/facebook-login/login-flow-for-web/v2.2>. Accessed 11 February 2015.
- 16 Facebook Developers. Facebook SDK for PHP [online]. 2015.
URL: <https://developers.facebook.com/docs/reference/php/4.0.0>. Accessed 27 February 2015.
- 17 Facebook Developers. Access Tokens [online]. 2015.
URL: <https://developers.facebook.com/docs/facebook-login/access-tokens>. Accessed 27 February 2015.
- 18 Facebook Developers. Facebook Login Version [online]. 2014
URL: <https://developers.facebook.com/docs/facebook-login/v2.3>. Accessed 2 March 2015.
- 19 Facebook Developers. JavaScript SDK. Best practices [online].
URL: <https://developers.facebook.com/docs/reference/javascript/FB.logout>. Accessed 2 March 2015.
- 20 Facebook Developers. Securing Graph API Requests [online]. 2015.
URL: <https://developers.facebook.com/docs/graph-api/securing-requests>. Accessed 21 March 2015.
- 21 Hypertext transfer protocol HTTP version 1. Header fields definitions [online]. 2014.
URL: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>. Accessed 3 February 2015.
- 22 Multisender documents [online]. 2014.
URL: <http://api.multisender.net/docs>. Accessed 21 February 2015.
- 23 Multisender documents [online]. 2014.
URL: <http://api.multisender.net/docs/groups>. Accessed 21 February 2015.
- 24 An Empirical Study of API Stability and Adoption in the Android Ecosystem. IEEE International Conference on Software Maintenance [online]. 2013
URL: <http://ieeexplore.ieee.org.ezproxy.metropolia.fi/stamp/stamp.jsp?tp=&arnumber=667687>. Accessed 7 April 2015.
- 25 Administrator panel of the final year project [Internet application].
URL: The administration url is not public information.

Response Body of a successfully Create Text Message

```

{
  "smsId": "405",
  "customerId": "9",
  "applicationId": "43",
  "transmissionId": "489",
  "message": "Hello Moon!",
  "sender": "012345xxxx",
  "sendRetrys": null,
  "maxResend": null,
  "resendInterval": null,
  "deleted": "n",
  "updatedAt": null,
  "createdAt": "2014-09-16 13:17:19",
  "transmissionTypeId": "2",
  "environmentName": "live",
  "actionTime": "2014-09-16 00:00:00",
  "price": "909.0000",
  "workerId": "541838613ecb4",
  "editable": "n",
  "receivers": [
    {
      "receiverId": "account-1856",
      "accountId": "1856",
      "groupId": null,
      "updatedAt": null,
      "createdAt": "2014-09-16 13:17:19",
      "name": "Mary Robinson",
      "firstName": "Mary",
      "lastName": "Robinson",
      "email": "example@example.com",
      "phone": "+358443349999"
    },
    {
      "receiverId": "account-1857",
      "accountId": "1857",
      "groupId": null,
      "updatedAt": null,
      "createdAt": "2014-09-16 13:17:19",
      "name": "Jake Last",
      "firstName": "Jake",
      "lastName": "Last",
      "email": "jake@example.com",
      "phone": "+358443349999"
    },
    {
      "receiverId": "group-26",
      "accountId": null,
      "groupId": "26",
      "updatedAt": null,
      "createdAt": "2014-09-16 13:17:19",
      "receiverType": "group",
      "name": "My First Group",
      "numberOfAccounts": 2
    }
  ]
}
```


Response of account list

```
[
  {
    "accountId": "1856",
    "firstName": "Mary",
    "lastName": "Robinson",
    "nickName": "Red Paron",
    "phone": "+358443349999",
    "email": "example@example.com",
    "address": "Street",
    "zip": "000100",
    "city": "City of example",
    "companyName": "Acme Co.",
    "website": "www.example.com",
    "note": "This text is a note.",
    "updatedAt": "2015-02-02 12:59:33",
    "createdAt": "2014-01-31 10:55:00"
  },
  {
    "accountId": "1857",
    "firstName": "Jake",
    "lastName": "Last",
    "nickName": "Nick",
    "phone": "+358443349999",
    "email": "jake@example.com",
    "address": "Street 2",
    "zip": "00200",
    "city": "HKI",
    "companyName": "Acme Co.",
    "website": "www.example.com",
    "note": "This text is a note.",
    "updatedAt": "2015-02-02 12:59:33",
    "createdAt": "2014-01-31 10:55:00"
  }
]
```