HikingWorld



Nikolaj Ørberg Hansen, Matti Andreas Nielsen & Victor-Emil Rossil Andersen

Software Development – System Integration – 1st semester spring 2016

31/03-2016

Indhold

[Brief Introduction 3](#_Toc447195355)

[Technologies 4](#_Toc447195356)

[Application Programming Interface (API) 5](#_Toc447195357)

[Server specifications 6](#_Toc447195358)

# Brief Introduction

# Technologies

Our main programming language throughout this project will be Node.js. Node.js is a server side language for developing web applications and is built on top of the Javascript core with added functionality. Node.js comes with a premade http server inside its standard library. This is the one we will be using for handling http requests, which will be the main protocol for our API. We will go in depth with our API in the next section.

We have chosen to implement the RESTful architecture in our application. To us the RESTful architecture provides us with a lot of freedom in how we will develop this application. At the moment we have a plan that we will be using JSON as the primary format of data. That is also a reason why we have chosen REST. If we had chosen SOAP we would be required to use XML as the format for all the data. The only way to bypass this would have been to wrap JSON inside of XML. This would require the network to transport heavy chunks of data around. By using REST, we will be granted the freedom to design and develop the application in a way that the client can decide its own content type for the response data. If we implement this functionality a developer using the API will be able to choose which format the data should be in.   
  
In addition to our own API we will also be using the Google Maps API for showing the predefined routes that the website will offer to the costumers. The Google Maps API is also accessible as a public REST service. The Google maps API is well documented and pretty easy to use. If we were to develop this functionality ourselves, we would have to drop out of school. We will also be using a weather API, but we have not decided on which one to use yet. The data from this API will show the weather prognoses of the different routes.

Another web integration we will be using is the continuous integration build cycle of Jenkins CI. This is not a tool that will provide functionality to our application, but rather a tool that will strengthen our development cycle. Jenkins CI is a tool that can automate a lot of tasks for us. In our situation we do not want to run all unit tests and integration tests locally on our computers. This will take time and pause the current development. Instead we will deploy Jenkins CI on our server which will run tests and build our project remotely every time we push to out git repository. It will then give a status report of the remote build and inform us of any errors. Jenkins CI will be called via a webhook which is basically a http request that will trigger whenever Github detect that a push has been made.

# Application Programming Interface (API)

**The restful convention we follow is:**  
GET 200 /api/resource/:id  
POST 201 /api/resource  
PUT 200 /api/resource/:id  
DELETE 200 /api/resource/:id  
  
**Our current server endpoints:**  
GET 200 /api/route/:id  
GET 200 /api/route  
  
GET 200 /api/review  
  
POST 200 api/user/login  
POST 201 api/user/register  
POST 201 api/review  
  
PUT 200 api/user/:id  
PUT 200 api/review/:id  
  
DELETE 200 api/user/:id  
DELETE 200 api/review/:id  
DELETE 200 api/routes/:id  
  
Client requests:   
-Google translate: [Https://www.googleapis.com/language/translate/v2?q=some\_text?target=](Https://www.googleapis.com/language/translate/v2?q=some_text%3Ftarget%3D)"dk"?key=some\_private\_key  
-Wheater api request:  
-Google map api:  
-Our own server endpoints:  
  
github webhook:  
PUSH [https://jenkins.it-kartellet.dk](https://jenkins.it-kartellet.dk/) forwarded for --> 10.10.10.15:8081 internally where our own server is hosting jenkins.

# Server specifications

The application will be hosted on a Linux server owned by our group member Matti. The server only allows connections through SSH with a public/private key combination. Regular username/password connectivity has been deactivated on the server for better security. Every person in the group will store a private key locally. The server will store all the public keys, which allows for authentication when connecting remotely. The Linux server is configured to only allow access on 8081 (Jenkins CI) and 3000 (Node.js).