Technical Report - Train Commander

FREVILLE Titouan
PETRIGNET Julien

December 14, 2015

Contents

Abstract

Part I

Software Solutions

NodeJS or go Solution.

Chapter 1

Engine core

1.1 Provide information to the engine

Call sncf api to provide real time data. If nodeJS solution: Ajax Call. If GO solution : go program.

1.2 Find way

Process on the data collected in the api using Node or GO function. Function with 3 arguments (Departure, Arrival, Time zone).

1.3 Process reservation

Call webview in AngularJS to fulfill the reservation form + make access to a virtual paypal payment.

1.4 Generate PDF bill and ticket

Generate the pdf using node or go + send it.

Chapter 2

Accounts

need a DataBase to store the accompts. Should be a MariaDB or Mysql one (MariaDB is a Mysql "framework").

2.1 Create account

2.1.1 From mail

Classic register form catch by go or node and Process in the database

2.1.2 From Facebook

Use Facebook connection api. Store information in the database.

2.1.3 From Google Account

Use Google connection api. Store information in the database.

2.2 Review travel

Standart call on the database using node or go to read the travel link to the accompt.

2.3 Replicate a previous travel

Need to store an historic of the travel from a user in database. Standart call on it to Replicate the valid data.

2.4 Print tickets/receipts

Call to the database to confirm he payed ... Standart call to node or go to launch print function (can be a pdf view printed from the navigator).

Chapter 3

Access to all feature

3.1 From website

Easy :p Just think about making the reservation process a separate web view who wil be called by the web site.

3.2 From Mobile application

Should provide a Android(JAVA) and IOS(Obj. C/swift) application. Think to use the web view for reservation (avoid replication in the work :))

Part II

Supporting Architecture Solutions

Chapter 4

Virtualization

4.1 Infra to power the app

Server should be on a linux system Supporting Vagrant. Call Vagrant to up the virtual machine. The VM should run on coreOS using dockers.

4.2 2 sites

Make sure to be able to copy states when the main is forced down.

Chapter 5

Cluster

5.1 Engine

Using docker + coreOS + Vagrant as system to run the app, we just have to manage the Cluster in the vagrantfile.

5.2 Web

Using docker + coreOS + Vagrant as system to run the app, we just have to manage the Cluster in the vagrantfile.

Chapter 6

Web CDN

6.1 Always pass by CDN

Need to find a CDN solution.

6.2 Load balancing