Technical Specifications Year 2022-23 Project 2

Maxime PAGES

December 14, 2022

Abstract

This is a document describing the Technical specifications for the project group 7 during the year 2022-23 at ALGOSUP. This document describes the practical aspects of the project such as its functionalities, which technologies were used and where, as well as the architectures used for both the front and back ends.

Contents

1 Introduction

The Object of the project is to build a robust and elastic service able to handle peak load of thousands of users per second. The service should appear as a publicly available website where any user would be able to enter any location and be served the remaining time until Santa's arrival.

1.1 The Team

The project will be made by a team of 5 over a period of 6 weeks, Louis is the Project Manager and is in charge of organising the work and dividing the task to the team, Ivan is the Program Manager, his job is to define the high level functionalities, and to promote the product to the customers, Maxime is the Tech Lead, his job is to chose how the functionalities will be implemented, be it the technologies used, or the server hosting. Max is in charge of the Quality Assurance, he will be testing every functionality of the project to find and point out any bug found, finally, David is our software engineer, and will be the one responsible for implementing most of the functionalities

2 Scope

Item	In Scope	Out of Scope
Website	X	
Geocoding	X	
Database	X	
Solar Time	X	
Countdown	X	
Docker	X	
Scaling	X	
Promoting	X	
Further Maintenance		X
Deliver Gifts		X
Santa Geo Tracker		X
Data Analytics		X

Table 1: Scope & Out of Scope

3 Approach

We will use a Micro-service like architecture that can take full advantage of the Docker and Kubernetes technologies. When a user tries to connect to the website, the request should be handled by Load Balancer, that will then route the request to one of the multiple servers that are running in parallel, this way of handling things allows us to

scale up and down the capacity in case of peak usage or to lower the cost during base load.

The server will then serve the HTML page to be displayed on the user's browser. On this page the user will able to do different things:

- Look up when will Santa Arrive
- Access the Advent Calendar
- Access a "About Us" page
- Go to the other groups websites
- Donate to Charity

4 Technology Stack

The front-end should be a standard website made of a HTML page that will use JS and CSS to display the Web Page on the user's browser, to serve the page from our servers, we will use Golang along with it's net package. Golang is a compiled language that is know for being faster than most other server languages such as Java.

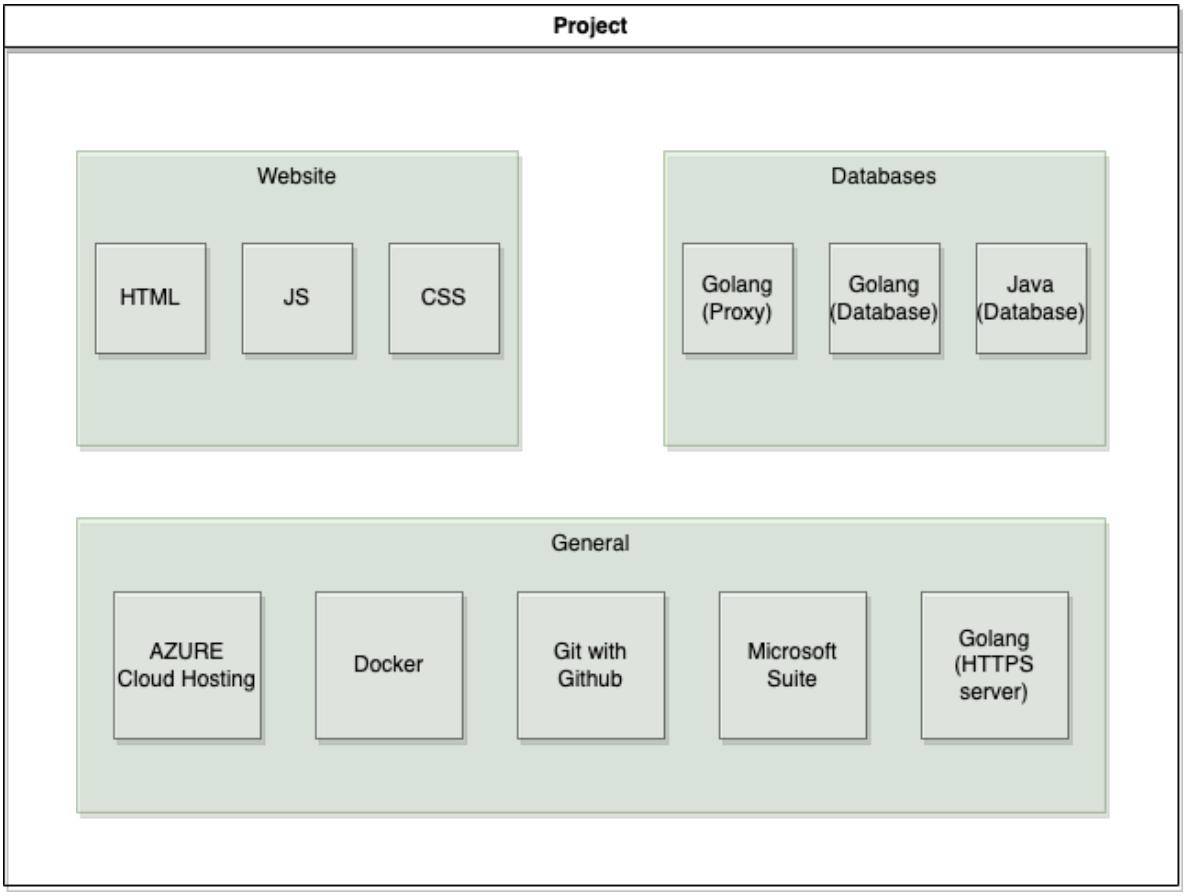

Figure 1: Technologies used

5 Server Architecture Overview

To respect our budget constraints, we will split the website into 2 azure accounts, 1 for the initial testing phase and prototyping the server architecture, and another one to host the final databases.

We will also use the Google Cloud Services to make use of their Kubernetes integration to ease the scaling of the servers, and because their quotas on Kubernetes Pods and virtual CPUs are higher

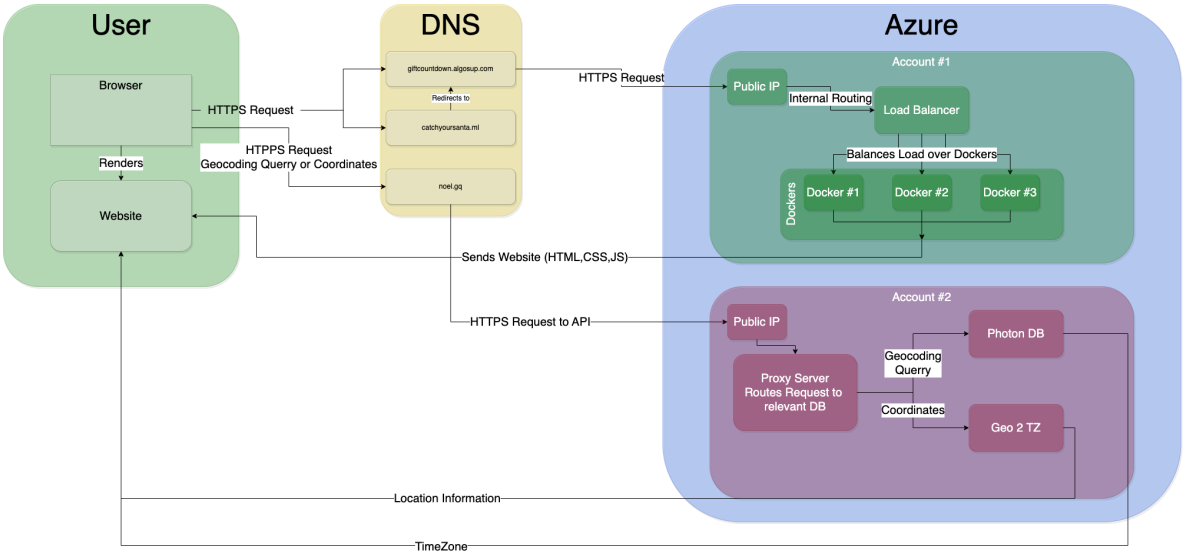

Figure 2: Software Architecture Diagram

6 Front End

The Front End of the website will be made using Hyper Text Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript (JS).

HTML is used to build a tree of the website (also known as DOM) that can then be understood by any browser on any hardware to display a Website, we then use CSS to modify the apparence of the website to make it more pleasing to the eye, which will result in a more attractive website and more users. Finally, JS is used to handle events and modify the webpage directly from the browser, such as displaying a timer after the user enter an address.

The website should be responsive so that any user, no matter the used hardware (Desktop, Smartphone, Nintendo 2DS, Smart Fridge) can access the website and make use of it's full functionalities without being constrained by their screen.

6.1 Searching

The main functionality of the Website is the search-bar used to display the countdown until the Solar Midnight on the 25^{th} of December (henceforth refereed to as "Solar Midnight").

6.1.1 Querying

The main way of interacting with the Search bar is by clicking on it and then typing a query in any language to look up an address or city. The search bar should first return and display a list of possible location after a second. Clicking any of the suggestions or directly pressing *Enter* should then display the countdown until that location's Solar Midnight.

6.1.2 GPS

The other way of displaying the countdown is by using the "*Use my Location*" button next to the search bar, this button should use the browser's integrated Geo location to display the countdown, or warn the user that this service isn't available if the service is blocked either at the browser or the kernel level.

6.1.3 Countdown

The time remaining until the Solar Midnight will be displayed as an animated timer akin to an analog scoreboard such as in sports events.

Only a single timer should display at a time, and the timer should update itself in case of a new query

6.2 Advent Calendar

To help user retention, it's helpful to have a reason to come back to the website more than once, we can take advantage of the Christmas theme by implementing an Advent Calendar. Users should be able to access the calendar trough a button in the top left. The calendar is made out of 24 square labelled from 1 to 24, clicking on an accessible square (meaning a square whose number is inferior to the current day) will randomly roll one of the 6 available images for the day depending on a rarity system.

The image that were won by the user for each day will be saved as a JSON formatted cookie in the user's browser, only the image won will be loaded from the GitHub repository to avoid having to download all of the images at once.

6.3 Translating

There should be a way for the user to translate the website to a language they understand, we will use the Google translate API to automatically translate the website in any language the user may want it to.

6.4 Calculating Solar Midnight

The Formula for calculating the Solar Midnight was found in this paper by William B. Stine and Michael Geyer

6.4.1 Calculating Solar Time

We must first define the local solar time of any point on Earth at any time. With:
t_{slr} = Local Solar Time t_{std} = Local Standard Time
M_{std} = Local Standard Meridian = $15 * (t_{std} -$ Greenwich time)
L_{loc} = Local Longitude

n = Current day between 1 and 365

$B = \frac{360\pi(n-1)}{65743.56} rad$

$E = 0.258cos(B) - 7.416sin(B) - 3.648cos(2B) - 9.228sin(2B)$
$t_{slr} = t_{std} + 4(M_{std} - L_{loc}) + E$

6.4.2 Deriving the Equation for Santa Time

Knowing that Santa time always happens on the 25^{th} of December at Midnight solar time, we can simplify a lot of the equations
DST = Offset to account for Saving time, either 0 or 1
Z = Local Time Zone

$t_{std} = DST + \frac{15Z - L_{loc}}{15} - \frac{E}{60}$

6.4.3 Example

Let's take Vierzon, France as an example.
$L_{loc} = 2.0698$, $DST = 1$, $Z = 0$, Therefore:

$t_{std} = DST + \frac{15Z - L_{loc}}{15} - \frac{E}{60}$

$t_{std} = 0 + \frac{15*1 - 2.0698}{15} - \frac{0.3829280015475218}{60}$

$t_{std} = 0.85563119997$
This means, that Santa will arrive precisely 0.85563119997 hours after Midnight, or exactly at 00:51:20

6.5 Miscellaneous

All 4 groups decided on making the other 3 group's websites available on each other's website, we will therefore have another page be accessible trough a button on the bottom left of the website where users will be able to see the team and the 3 other websites.

We will also have to add a cookie warning per the RGPD to inform users of our use of cookies

Finally, we will add a small *Donate* button in the lower right where users will be able to donate to the UNICEF.

7 Back End

The back end will be split into 2 parts, the first part, the HTTP(S) servers responsible for serving the website will be hosted on Google Cloud to take advantage of Kubernetes. This part consist of a Public Static IP that will be accessible trough https://giftcountdown.algosup.com/ this will then redirect the request trough Kubernetes to one of the pods running in parallel.

The Second part of the back end will be hosted on Azure, this part consists of a static public IP accessible trough https://noel.gq . The request are then handled by a Proxy Server written in Go that redirects to one of our databases

This architecture takes advantage of the free credits given by both Azure (200€ for 1 month) and Google Cloud (300€ for 3 months) and bypass some of the respective quotas imposed on us due to using only free accounts.

7.1 GKE

To go from a single IP to a network of servers running in parallel we will be using the Google Kubernetes Engine (GKE), GKE is able to automatically deploy multiples instances of a single application and have them run in parallel, by monitoring how busy each server is, GKE is able to redirect incoming requests to an avalaible server, in the case of an instance of the application going down, GKE is able to detect the failure and reboot the instance back to a working state.

On top of that, in the case of extreme stress on the current instances, GKE is able to spin up new instances of the applications, however, this is a paid feature and will therefore not use it for this project.

7.2 Docker

We are using the Docker technology to allow for portability between any architectures, each Container should be made as small as possible by using compiled go with the website embedded and building the image from scratch. This allows us to reduce the size of the container to under 7MB, allowing the container to be started, or rebooted in case of a crash, in less than a few seconds.

7.3 API Proxy

Our 2 databases are both hidden behind a single proxy server to unify the APIs, furthermore, this way of doing things allows us to only need a single certificate to use the HTTPS protocol for both DBs.

The Proxy server should receive a query, and depending on the parameters received, route the request to the appropriate DB and then send back the response.

Database	Path	Use Case	Example Query
Photon	/api?q=<*query*>	Used to recieve a list of likely locations and and addresses from a plain text query	/api?q=Vierzon
Geo2Tz	/tz/<*latitude*> /<*longitude*>	Used to get the time-zone from coordinates	/tz/2.063/47.227

Table 2: API usage

7.4 Photon

There is a lot of different possibilities to chose from to handle Geocoding, the first possibility is using a Public API to handle geocoding for us, however there are 2 different problems:

Price, most APIs are either not free, or are offering a free version that is either throttling the speed at which we can make request, or will simply refuse to respond past X requests.

The second one is availability, by using a public API, we rely on the assumption that the service will be up 100% of the time, and any internal problem would be out of our reach, being exposed to such a risk runs contrary of the Object of the project.

We will therefore be using our own DB, Nominatim was our first choice, Nominatim uses the full Open Street Map data to geocode, however, the DB is far too heavy to be deployed on Azure.

We will then use Photon, Photon is a Java Based Database that uses a smaller subset of the OSM dataset, making it only 60GB of data, which is small enough to be deployed on a Free Azure Account. Photon handles a plain text query that the user entered, and will parse it to a proper SQL request to try and find the address the user most likely meant, along with a lot of useful information such as the coordinates of that address. However, the database is slow enough to be a bottleneck in case of peak usage, and is not easily scalable, so the first step in going further with the project would be implementing Nominatim using a Premium Azure Account.

7.5 Geo2Tz

The second database will be using is called Geo2Tz, this DB is used to find the timezone of a point on Earth using its coordinates, Geo2Tz is written entirely in Go and fits inside a Docker Container, making it easily scalable if needed. It is however fast enough to not be a bottle neck at the moment.

8 Monitoring

We will monitor the incoming traffic 2 different ways, the first being the Official Google Cloud API and the second being a small server hosted and azure used to count the incoming traffic by origin

9 Security

We are using the HTTPS protocol to send and receive traffic to and from the Databases and servers, so all the information is encrypted and safe. All the cookies are containing non-sensitive information so there is no risks there either

10 Privacy

We are not saving any personal information such as queries or geolocalisation, and the use of cookies is disclaimed when entering the website

Glossary

A

Azure Online Microsoft platform to rent servers or services on the Cloud. 10

C

Countdown Time remaining until the local Solar Time for Christmas. 3

D

Database A database is a place where we store information, in our case, we store our information about Addresses for Geocoding in Azure. 3

Docker Service used to . 3

G

Geocoding Process used to get the coordinates (latitude & longitude) of an address. 3, 10

P

Promoting . 3

S

Scaling . 3

Solar Time Time System based on the position of the Sun instead of the Greenwich Meridian and time zones. 3, 10