Cloud-Based Currency Converter

**with Geolocation Service**

Computer Science Concepts for Data Scientists

**Group 11**

**Maha Hazimé-Zayour**

**Esin Handenur Isik**

**Sabrina Rigo**

**Table of Contents**

[**Information**](#_heading=h.flhfmrxeqe8z) **I**

[**About the App**](#_heading=h.fe1c79ibwbpk) **1**

[**Motivation & Usage**](#_heading=h.1pt56zh9cm7k) **1**

[**Data Sources**](#_heading=h.c58wifa12338) **1**

[**Process and Outcome**](#_heading=h.nvziv7p9eyqa) **1**

[**Process Summary**](#_heading=h.2et92p0) **1**

[**The Draft of the Interface**](#_heading=h.9nbx5o93dr3t) **2**

[**App Functionalities**](#_heading=h.coy5tsxxaf3) **2**

[**Code Building on Thunkable**](#_heading=h.lwwsidwwoaoy) **2**

[Interface: Final Version](#_heading=h.gdq4f5dgbh9) 3

[**Problems & Solutions**](#_heading=h.gx7ehcahzqzf) **4**

[**Reflection and Outlook**](#_heading=h.4d34og8) **4**

[**Interface Pictures**](#_heading=h.v1973mq4tzmo) **5**

[**Appendix**](#_heading=h.u7368yq4vhi8) **6**

[**Interface Draft 1**](#_heading=h.se7y0ntej1ga) **6**

[**Interface Draft 2**](#_heading=h.zfwajamir9w2) **7**

# Information

The link of the thunkable project is the following: <https://x.thunkable.com/copy/fc96bbe491852da85368a8c55e6482f1>

Disclaimer: During the course of developing the application, we ran into multiple complications regarding the live-testing on mobile devices. Upon research, it was found out that certain coding blocks on thunkable can cause the app not to work on various mobile devices (especially coding blocks associated with the Google Maps function). While it appeared to only affect Android phones, continuous testing revealed that it can affect some iOS devices as well. However, the last tests made have proven the app to be fully functional on an Android, and an iOS operated mobile phone. Nevertheless, we recommend doing a live-test on two different phones since the exact cause of this error remains unclear.

# 

# About the App

## Motivation & Usage

With the “Currency Calculator” we wanted to create an app in which you can convert whichever currency you like with real-time exchange rates. If a person is traveling around, it is possible to search for the currency at the current location with the help of a location sensor. The benefit behind this idea is to make it possible for a user to convert currencies even if the currency at the current location is not known.

## Data Sources

The data used in this application is retrieved from Exchange Rates Data API and the OpenCageData API. A list[[1]](#footnote-0) of all supported currencies were extracted from the Converter API and put in a list which is again used to verify the input currencies inside the app. Also, geolocation data requested by the user is utilized for operating the OpenCageData API and the map view.

# Process and Outcome

## Process Summary

In the planning phase of the project, a discussion between the team members about different mobile Application ideas was carried out. After the discussion, the decision of developing a currency calculator mobile application was made. It was agreed that the app should access and visualize cloud-based data.

During the design phase, a user interface was developed and tested on different devices by the team members, to ensure that the project was moving in the right direction. After that new features were added, and problems were fixed. This looping phase was kept on repeating until the App met all the requirements.

## The Draft of the Interface

In one draft[[2]](#footnote-1), we started with two screens. Screen 1 included the calculator where the user could choose if they want to retrieve the “from” or the “to” currency. The second screen in that draft included a map showing where the current location of a person is with a marker. This was the backup plan in case we would not be able to implement the OpenCageData API.

In the other draft[[3]](#footnote-2), we visualized our idea with three main screens:

- home screen

- search by location screen

- the currency converter screen

In the end we decided to create the app with these three screens.

## App Functionalities

We used the OpenCageData API to get the currency ISO code at the current location. Therefore, the API key and the parameter q was required. Since we used the reverse geocoding “q” is equal to latitude + longitude in decimal format. From the reverse geocoding response format we then determined where the ISO-Code is located so that the right part of the JSON response is displayed in the end.

The Exchange Rates Data API is an API that requires 3 query parameters: “amount”, “from” and “to” which are included in the request URL[[4]](#footnote-3). The API request is structured to have a header included which communicates the API key. The response data contains only the result which is the conversion amount.

## Code Building on Thunkable

The screen 1 includes two code blocks that get activated by clicking on either the button “CurrencyCalculatorButton” or “SearchByLocationButton”.

For screen 2 a code block was created which leads back to the first screen when clicking on the “HomeScreenButton1” button. By tapping the “SearchLocationButton” button the location sensor gets started and we get the latitude and longitude needed for the OpenCageData API. With this function the currency ISO-code is retrieved and the map to display the current location is initiated. We included the zoom function so that it shows the right section of the map and a block to show a marker on the map. Another block checks if the entered currency “to” is equal to “from” and would in that case show “choose other”. If the input is not in our data source table it changes to “invalid”. The search button[[5]](#footnote-4) can be clicked and some rows will get invisible to make space for the list with all the currencies and uses the list viewer function. This function then saves the first column of the data source table to the variable “Currency Name List” and this then is set to be the viewer content. The variable “list to search” is also set to that content. With the smart search block it is possible it iterates over the “List to search” variable with every entered letter and filters it. The code blocks when an item from the list is clicked on, restores the hidden rows and initiates the “List Selector to Currency Code” function. With this function a loop is created which goes over the currency name list to receive the ISO-Code. This code is then included in the wanted currency input field. The currency switch block makes it possible to store the values and rearrange it so that the codes will have switched places. The last block is used to set the parameter and header used for the Exchange Rates Data API so that it converts and displays the amount to the wanted currency.

Screen 3 uses the same code blocks as screen 2. With the difference that the first input field also has a smart filter search list and there is no map or location sensor.

## Interface: Final Version

The final product of this project is an application that converts currencies based on real-time data. The characteristics are based on 2 essential features and are presented in a 3 screen layout:

The home screen (1) has 2 buttons which direct to a currency converter with a smart search filter or to a converter with the location feature.

The currency converter on Screen 2 integrates the live location after receiving the user's approval. The app interface then shows the statement: “The currency at your location: [Location])” and a map on the bottom of the screen that points to the current location. For the second currency, the user can enter the currency code (i.e., CHF) or simply press the search button to see and filter options. The smart search filter contains currencies in descriptive form (i.e., “Swiss franc”) and starts filtering the list per entered letter in the search bar. Once a selection is made, its respective ISO code will be pasted into the input bar. A switch button on this screen allows both currencies on screen to be switched out. When “convert” is pressed, the API delivers the conversion rate and the result shows on the screen as: [Entered amount] [initial currency] is equal to [result] [wanted currency].

Screen 3 has a similar layout and enables converting currency by entering by entering/selecting both. In this case, there are two lists with a smart search filter available. Also, both screen 2 and 3 have the previously mentioned validity checker.

# Problems & Solutions

In this subsection, the most challenges and solutions faced during this mobile App development are presented.

Thunkable is not designed for group App development. Thus, working on the project together was a challenge that was solved by creating a common Gmail account to allow access to all team members to the project.

Furthermore, working with Thunkable blocks was tricky. Retrieving data with multiple columns with info of one element in one column required a rowID. This was solved by creating a loop iteration. In addition, the blocks were doubled for nearly identical elements. Therefore, various functions were created to avoid big code sizes.

The different mobile platform was not only a challenge regarding User Interface but also for testing. App testing does not work on all laptops because of geolocation technicalities. Thus, a test was needed against different Android and iOS versions to ensure that the App works, and the layout appears as required. The solution was to agree to be satisfied with a slightly distorted app view if all screen elements are visible and in a sensible order.

Dealing with APIs was demanding. The PositionStack API wouldn’t work because encryption was not suitable for Thunkable. This was solved by using OpenCageData API which required studying JSON responses further for the extraction of the wanted result from within the response structure.

# Reflection and Outlook

We managed to implement what we wanted in the app. However, we realized when working on it that some ideas were not necessary or that it can be done differently. It helped us a lot to understand how with the APIs real-time data can be used.

However, with more times we would have implemented some more things. We could have increased the input/search options within the smart filter (i.e., possible to search by country name) or we would have made the interface design more user friendly and appealing. To make it a better tourist/travel app we could have included some restaurant recommendations and tourist attractions as markers in the map.

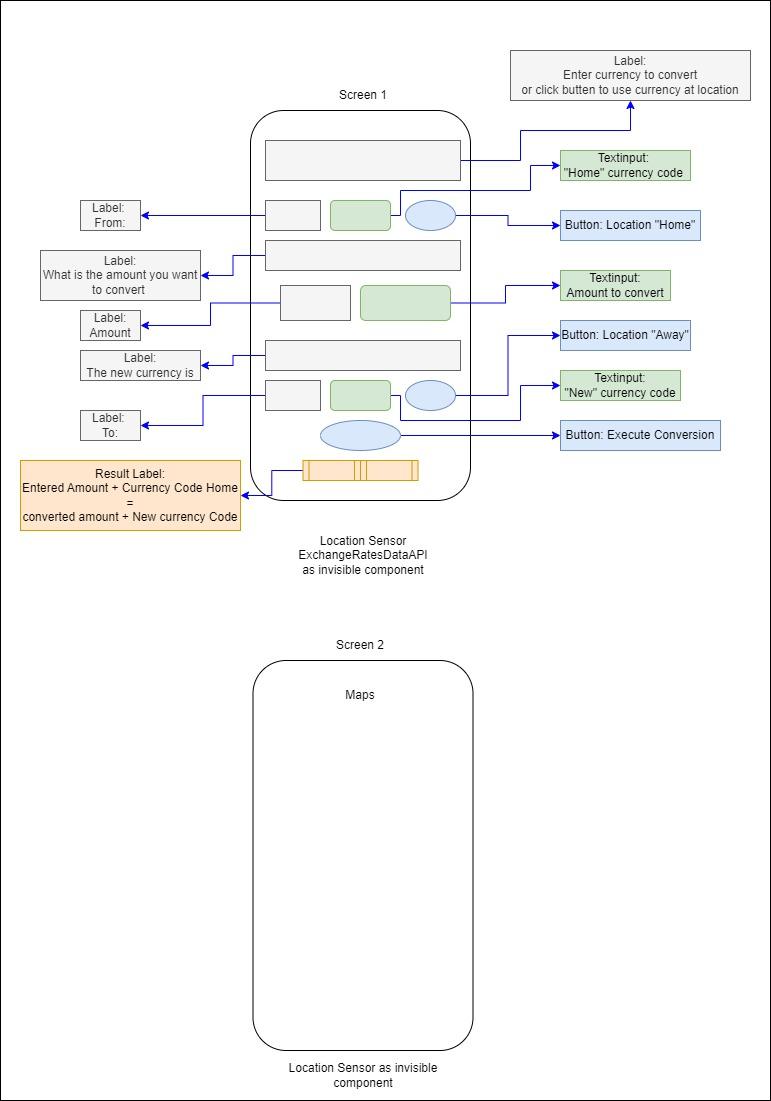
# 

# Interface Pictures

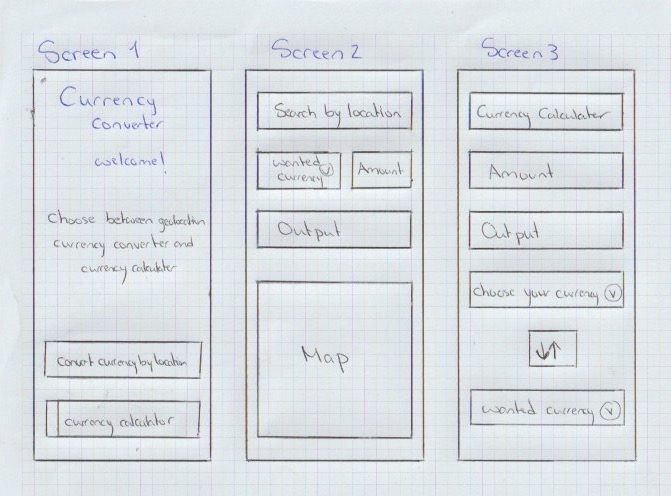
# 

# Appendix

## Interface Draft 1



## Interface Draft 2



1. Excel file: “Currency Table - Converter App” in submission folder [↑](#footnote-ref-0)
2. [Appendix “Interface Draft 1”](#_heading=h.1e5zx6nbia3z) [↑](#footnote-ref-1)
3. [Appendix “Interface Draft 2”](#_heading=h.1ogjszq5dhe4) [↑](#footnote-ref-2)
4. "https://api.apilayer.com/exchangerates\_data/convert?to={to}&from={from}&amount={amount}" [↑](#footnote-ref-3)
5. Button named on thunkable: “Screen2wantedcurrencysearchbutton” [↑](#footnote-ref-4)