# Intro

Partyholic is an Android app that allows the user to capture a photo with their camera or choose one from their gallery, with the photo being of a poster of a party or a festival. Then, Partyholic extracts the time, date, and location of the event from the poster and uses it to search on Google Maps and show the destination directly on a Google Maps windows inside the app screen and zoom in onto it, while leaving the user the option to go back and forth between their current location and the event location (the destination). It also calculates the time remaining until the event and shows it on top of the screen. This means, the user has just one job, which is to take a photo or choose one from their gallery. There is no need to type any text with your fingers. Instead, you, as someone who is wandering the streets looking for the next party, can enjoy your time with less disruption.

# Assumptions

We assume that the poster contains at least:

* the address of the event, which we assume, consists of (in the following order):
  + the name of a street
  + the number of a building
  + a ZIP code (German: PLZ) which is a five-digit number (in Germany)
  + name of the town/city/village/etc.…
* the time of the event.
* the date of the event.

Also, we assume that the operating system of the phone is Android and that the phone has a camera with a reasonable image capturing resolution and a working internet connection.

An important thing is, that the user will have to give permission to Partyholic to use their GPS and camera sensors and their internet connection and access the device storage as well.

In order for Google Maps service to work, the user will also have to be signed in into their Google account.

# Structure

Since our app involve image processing and optical character recognition (OCR) it is more efficient to do it with server-client architecture because the mobile phone processor is not as powerful as a computer processor.

Partyholic is two parts:

* frontend: the part which the user accesses directly by installing the .apk file on their device.
* backend: the part which the software installed on the device accesses via internet and is where the image processing happens, which in turn communicates with APIs.

The Android app has 6 activities (screens):

|  |  |
| --- | --- |
| Initial screen that asks the user for the previously mentioned permissions. |  |
| Main screen with Partyholic logo and description and a dialog and two buttons: Camera and Gallery. |  |
| Confirmation screen which views the selected/captured image one last time and asks the user to confirm they want to use it or not. |  |
| Loading screen which shows up once the user decides to use the image, since that is the most critical part and since it takes time the most. The loading screen serves two purposes:   * Indicates to the user that they have to wait a few seconds. * Prevents the user from issuing more orders while an order is already being processed. |  |
| Result screen which has a panel with a text being the time remaining until the event, or in case of failure to find time/date, an apology message, and a Google Maps window that shows the destination and the current location of the user. |  |
| Apology screen which shows up in case the address of the event could not be recognized due to the low quality of the image, or it not containing an address at all, or because the address couldn’t be found (for example, because it does not exist in the real word). |  |

# Techniques Used

## Programming languages:

* frontend : Java
* backend: Python

## Frameworks:

* Heroku: cloud platform for hosting the backend part.
* Android Studio: for the frontend
* Flask: for the backend

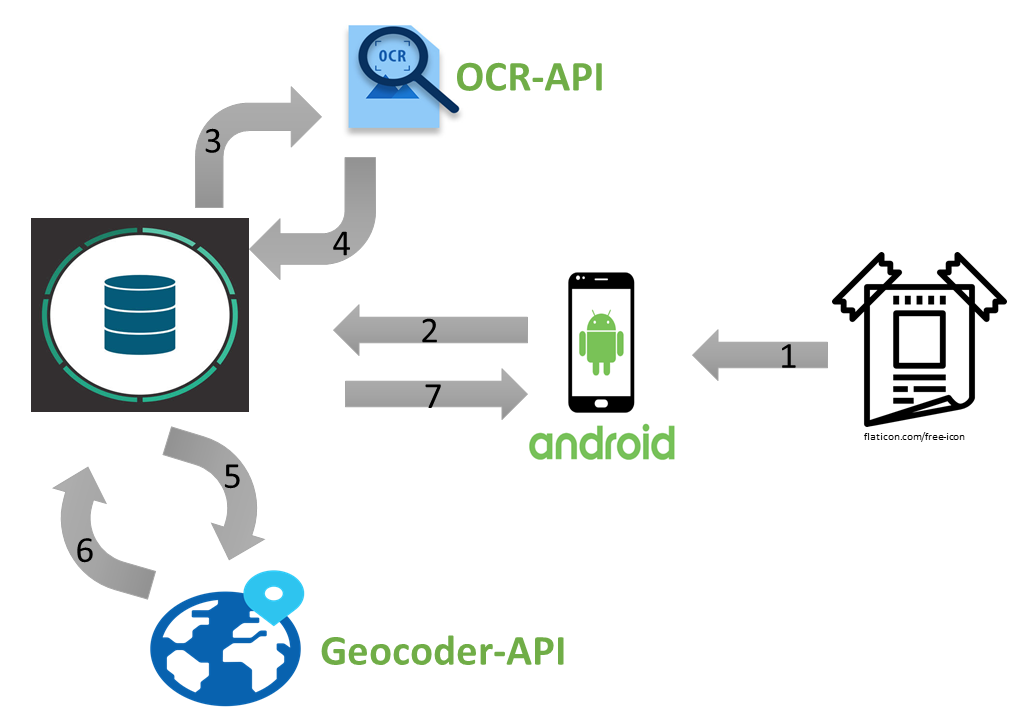
We chose the Python-based framework (Flask) to build our backend since it is easy to deploy on Heroku.

## External APIs:

* Google Maps: it is used inside Partyholic in a Fragment.
* Nominatim (OpenStreetMap): a free geocoding API finds coordinates of addresses.
* ocr.space: optical character recognition for extracting the text from the image.

In order to achieve our goals we need some steps to take :

# How Partyholic works (in chronical order):



## capturing image (or choosing from gallery):

## receiving the image from the mobile app

after capturing an image with the app or exploring an old one , the image will be sent to the server as a stream of bytes whereas the server will receive these stream of bytes and write it in the home directory and give this image a unique name to distinguish it from other requests since the server can receive more than one request in the same time from different users .

When the image is no more needed after sending the response to the app it get deleted so that the home directory is not filled with unnecessary files and the storage will not run out of capacity.

## applying the OCR on the received image

The Process of extracting the text of an image called optical character recognition (OCR)

and it is an complex process which involve neural network and machine learning.

There is an open source engine to apply OCR on an image called tesseract and we try to use it first

but it appears that this engine is developed to be applied on a scanned book documents and it gives in that case very accurate and reliable results.

But in our project the data we need to process are colorful images and after applying the tesseract on some test cases we did not get an accurate results even after many experiments with reprocessing as filtering , Re-scaling , Blurring, and image Thresholding.

Because of that we choice to do this step by a third party API which is ocr.space and

they offer a free API-KEY but some limitation

One of this limitations is the file size , which should be less than 1 MB and therefore we compress the image so that it will be always under this limitation

## Detecting the related information from the text

We did this step with regular expression (Regex) , which is the manual way or (rolled based way) in Natural language processing (NLP) to detect a specific pattern in a given text.

As we are aiming the German user , we specify our app just for the german language

We define some patterns to detect the time of the party, which could be as followed

(um 20:00 uhr ) or (um 18 uhr)

As will as for the date

For example :

(am 12.04.2021) or (am 12-januar-2021)

And also for detecting the addresses , which were not doable just with regular expression

because it is not possible to detects all possible addresses without trained data and machine learning

So we just tried to catch the most address patterns and it seems that the word “Straße” appeared on the most street name’s in Germany depending on this website :

<http://interactive.zeit.de/german-streetnames/>

and it also help us to find other common patterns .

## Resolving the coordinate of the address

In order to show the location of the party on user’s google maps we need a geocoder , which is an API to resolve the coordinate of an address

We try first to use the google-api-geocoder but google does not let us to use this service before adding an Credit card and therefore we switch to an open geocoder it called “Nominatim” and it does not required an api-key to use it.

## Sending the response to Front-End

The last step is adding all extracted information which is the address ,date ,time ,coordinate to a JSON file and send it to the front-end

In case of the image does not have one of the mentioned information or the server could not extract it this part will filled in the JSON file with “Unknown”

# Conclusion

Now we have vision of how to build an application in all its stages, starting from discussing the idea of ​​the project and the problems that we may face us within a working group and how to find solutions and search in different places sites. We learned how to design the graphic user interface, build the backend and link the user interface to the appropriate API, in addition to testing the program to find out and troubleshoot problems.