



A crowdsourced map for checking supermarket wait times worldwide

A 2020 project to help people during the Covid19 global lockdown

WhoAmI

Hi everyone from Florence, Italy!
I'm a Full Stack Engineer @ Growens
and co-founder of Schrodinger Hat
Podcast

I'm in love with Python and open
source but, especially, with pizza!

You can find me everywhere as
@thejoin95 and on mikilombardi.com



TABLE OF CONTENTS

01

Intro &
Overview

02

Architecture
& Cloud

03

APIs &
Redis

04

Costs &
Conclusions



01

Intro & Overview

As you may know in 2020



That pretty much sums it up

Everyone started to take everything
from the pharmacies and supermarket



The issue

In the days following the global lockdown, supermarkets, pharmacies and parapharmacies were attacked by people in search of basic goods and necessities.

Immediately after the announcement of the lockdown, many supermarkets struggled to manage the influx of people, forcing them to gather outside the supermarket, standing in line and waiting to enter the building.

The context

The longer you stay out and in contact with other people, the more likely you are to contract the virus. Especially in closed and unventilated environments.

The project that I am presenting to you was created primarily for personal use and then became for public use worldwide. The project was not for profit but only to help others and stay safe.

<https://github.com/TheJoin95/covid19-market-waiting-times>

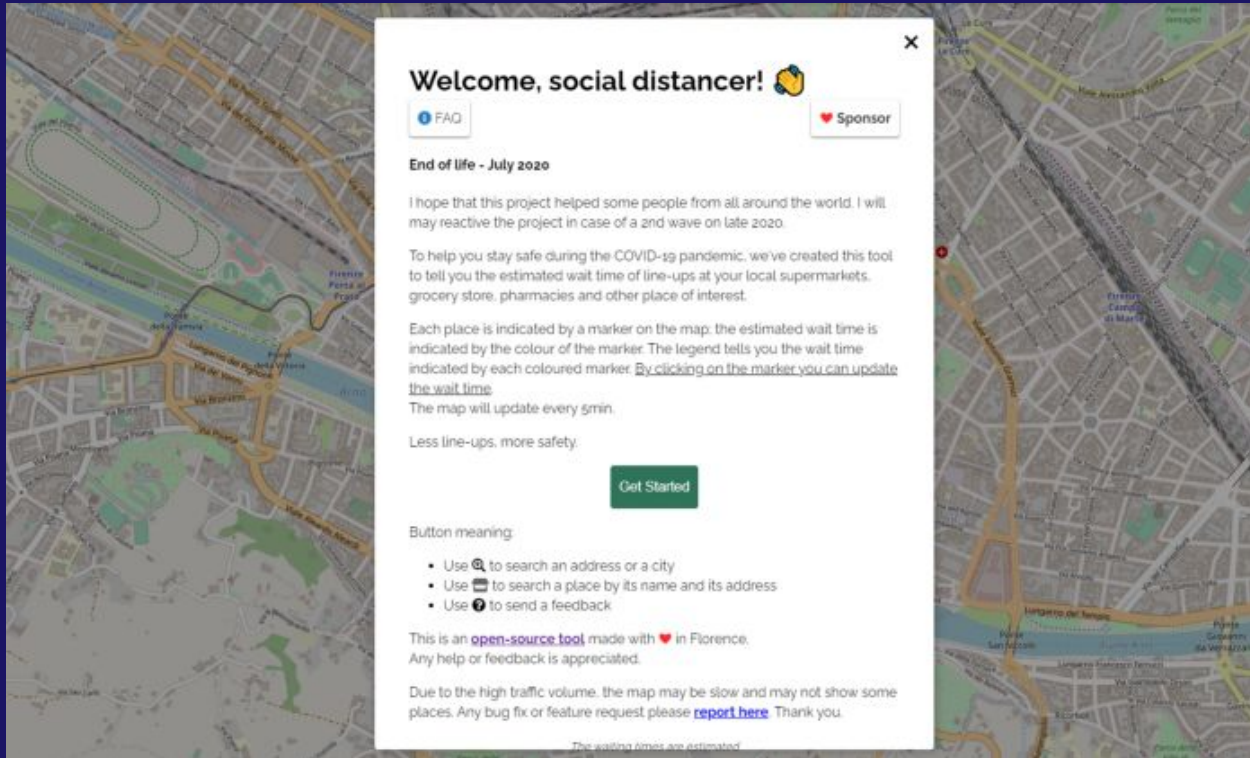
The idea

The gathering of people in line in front of shopping malls and pharmacies made me think. Why not create some sort of tracking based on some data? Kind of how it works for traffic.

So the goal was to create a **map** with the various points of interest, such as: food, bakers, pharmacies, supermarkets, liquor stores etc. where the waiting times retrieved by some APIs were shown giving the user the possibility to add a temporal feedback for the store in which he was located.

Crowdsourcing was then fundamental for the success of all this, since API's data is based on the time, the popularity of the place and how many devices are nearby. User feedback was therefore crucial.

The map



Welcome, social distancer! 🙌

[FAQ](#) [Sponsor](#)

End of life - July 2020

I hope that this project helped some people from all around the world. I will may reactive the project in case of a 2nd wave on late 2020.

To help you stay safe during the COVID-19 pandemic, we've created this tool to tell you the estimated wait time of line-ups at your local supermarkets, grocery store, pharmacies and other place of interest.

Each place is indicated by a marker on the map, the estimated wait time is indicated by the colour of the marker. The legend tells you the wait time indicated by each coloured marker. [By clicking on the marker you can update the wait time.](#)

The map will update every 5min.

Less line-ups, more safety.

[Get Started](#)

Button meaning:

- Use 📍 to search an address or a city
- Use 🏠 to search a place by its name and its address
- Use 🗣️ to send a feedback

This is an [open-source tool](#) made with ❤️ in Florence. Any help or feedback is appreciated.

Due to the high traffic volume, the map may be slow and may not show some places. Any bug fix or feature request please [report here](#). Thank you.

The waiting times are estimated.

From local to global

The project stayed online from late April to mid June, when the emergency disappeared





02

Architecture & Cloud



Flask

Technology



redis

Python3.7

Flask

uWSGI + Nginx

Netlify

Redis

PWA

uWSGI



netlify

NGINX

PWA

Personal use & Beta

Hosted on a RBPi 3B 1GB+32GB 32 Bit in port forwarding & dynamic dns

- Frontend
- Redis server
- Nginx
- uWSGI
- Flask APIs
- ... everything!

Beta testers: friends, mums, friends of friend, grandmums, uncles, colleagues, local priest in an area 30x30km



Going Worldwide

- 1 VPS on OVH.cloud (2GB+30GB)
- 3 VPS Digital Ocean 1x2GB+20GB (ny) & 2x1GB+10GB (uk, sf), CDN

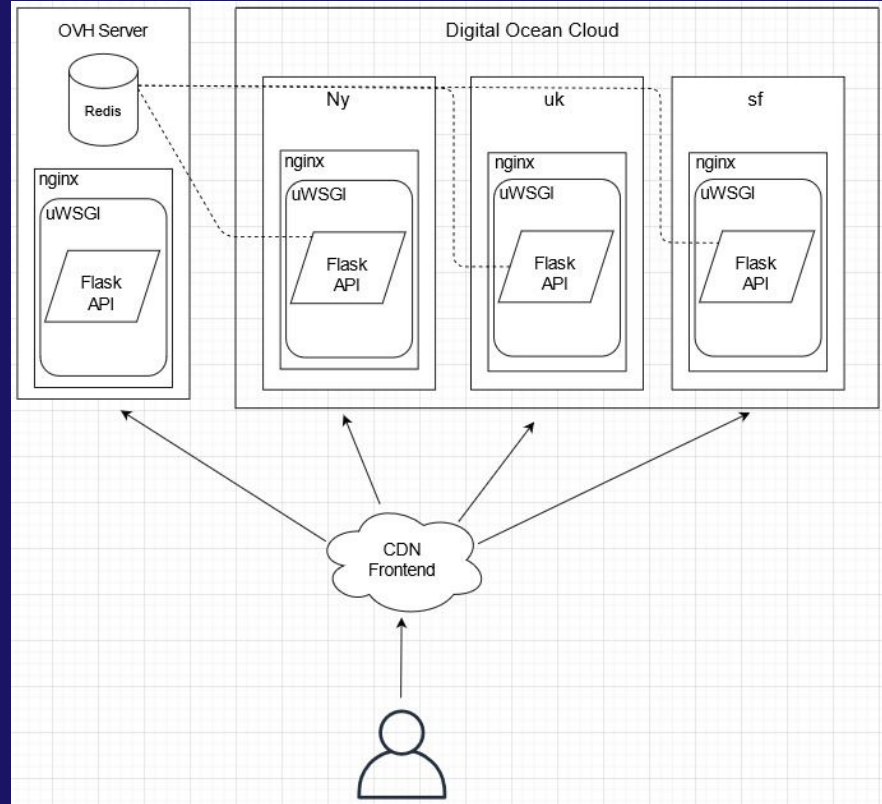
Why I decided to go in cloud:

- + More than 100 contemporary users
- + More than 20k pageviews per day
- + Performance
- + Scaling
- + Availability
- + Future

The “future” was: around 1600 contemporary users & 90-100k pageviews day from all over the world especially from North America & Europe

Schema & Management

- + Centralized Redis
- + 4 APIs instance
- + API rotation in Frontend
- + Up/Down status script for each instance
- + uWSGI web server + Flask layer
- + North America + Europe covered





03

APIs & Redis

APIs

A recap of what these endpoints were used for:

- + Get geocoding data based on Here & Google API
- + Get the estimated waiting time for a place id
- + Get the place by its name
- + Get the place by geocoding
- + Crowdsourcing logic via feedback
- + Logging data such as trace error, message infos. No log user activity

Endpoints

- + /geocode
- + /places/get-waiting-times
- + /places/get-by-name
- + /places/explore
- + /places/browse
- + /feedback
- + /logger

Multithreading via uWSGI

```
@uwsgidecorators.postfork
@uwsgidecorators.thread
def initThread():
    while (threading.active_count() <= 400):
        t = threading.Thread(target=worker_fulldetail)
        t.daemon = True
        t.start()
```

Geospatial in Redis

In order to get the places with their wait times we add the place in a Redis db via GEOADD

By retrieving the information via latitude and longitude we can use GEORADIUS

Having the information saved and encoded geographically is a really good advantage in terms of performance

On every request, we write on Redis db with a TTL

```
def getPlaceInRadius (lat, lng, distance=15):  
    # ...  
    places = r.georadius(  
        "places",  
        lng,  
        lat,  
        distance,  
        unit="km",  
        withdist=False,  
        count=150,  
        sort="ASC"  
    )  
    # ...  
  
def geoAddKeyRedis (key, lat, lng):  
    # ...  
    try:  
        r.geoadd('places', lng, lat, key)  
    # ...
```



04

Costs & Conclusions

Costs: from RBPi to Cloud

Apart from having a RBPi 3B, the cost would be around 30\$.

I reused also my former 2020 domain *thejoin.tech*, creating a subdomain *covid19-waiting-time*.*

I spent about \$ 25 per month for cloud usage, the total is around \$ 75 per 3 months (ovh+digital ocean).

The free inbound/outbound traffic was 10GB per month, so it wasn't an issue.

In the end, having also the frontend on Cloud it allowed me to satisfy (in the whole life cycle) more than 1Mln of users and more than 2.1Mln of sessions in 3 months of life with more than 1.5K contemporary user

Conclusion

I'm really pleased to had the chance of helping people all over the world.

- + Thanks, to the Open Source
- + Big thumb up to Daniel Dafoe for helping me on the WebIAM & Design
- + Thanks, to my favorites "beta testers", my friends & family

The code is still available on my Github, the website is offline.

Stay safe,
Miki



CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik.

Q&A

