دانشگاه اصفهان
دانشکده مهندسی کامپیوتر

پروژه پایانی: سامانه بلادرنگ تاکسی‌های تلفنی | **استاد:** دکتر نعمت‌بخش

درس: داده‌های حجیم | **دستیاران:** فاطمه ابراهیمی، پریسا لطیفی، امیر سرتیپی

نام و نام خانوادگی: سید عمید اسدالهی مجد | **شماره دانشجویی:** ۴۰۰۳۶۱۴۰۰۴

آدرس گیت: https://github.com/amidmajd/bigdata-final-project

مقدمه

در این پروژه گام‌های شش و هشت از پروژه پایانی این درس انجام گرفت. همچنین گام اول نیز انجام شد تا بتوان رفتار واقعی به‌هنگام دریافت داده از کافکا[1]، ذخیره در ردیس[2] و دریافت در API را شبیه‌سازی نمود. در انجام این بخش‌ها از تکنولوژی‌های مختلفی از جمله داکر، کافکا، ردیس، FastAPI و ... استفاده گردید. تمام کتاب‌خانه‌های استفاده شده در فایل requirements.txt قرار دارند که می‌توان با دستور pip install -r requirements.txt آنها را نصب نمود.

گام اول

داکر و راه‌اندازی کافکا

برای راه اندازی یک کلاستر شامل کافکا از داکر استفاده شد. در فایل docker-compose.yml تنظیمات مربوطه قرار دارند که با دستور docker-compose up -d می‌توان محفظه[3]های مورد نیاز را راه اندازی نمود. این بخش شامل سه محفظه است. محفظه اول مربوط به zookeeper است که وظیفه همگام‌سازی نودها، پیام‌ها و تاپیک‌ها را بر عهده دارد. محفظه دوم مربوط به kafka است که محفظه اصلی می‌باشد. محفظه سوم مربوط به kafka-ui است که یک رابط گرافیکی را در اختیار کاربر در آدرس http://127.0.0.1:8080 قرار می‌دهد. تنظیمات خاصی در داکر انجام‌شده تا این سه محفظه به درستی درکنار هم کار کنند. کافکا در پورت 9092 در localhost در دسترس است.

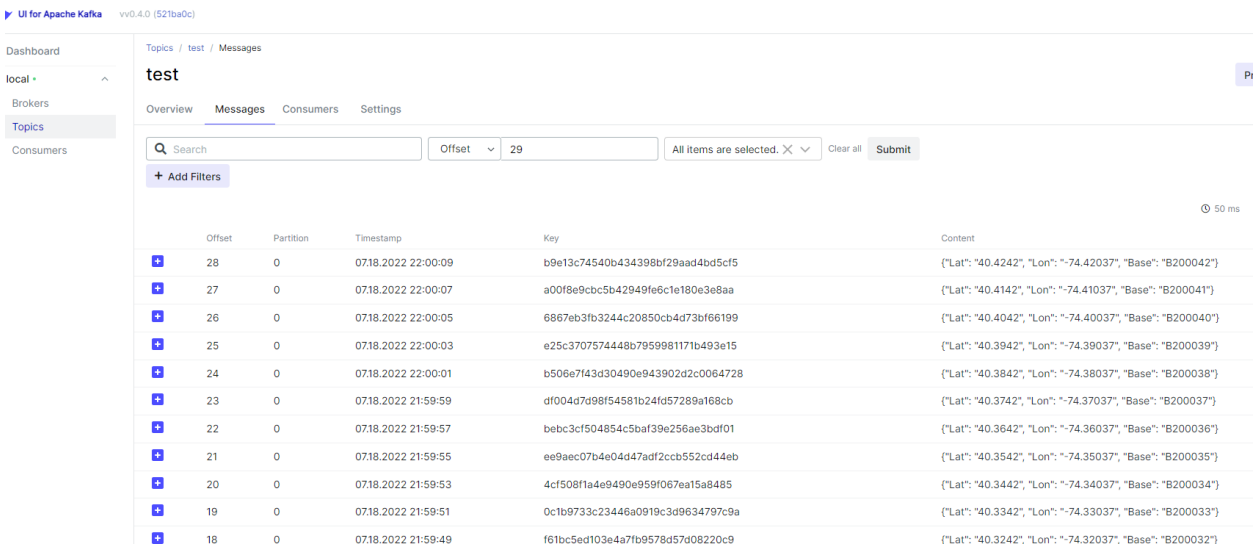
ارسال داده (پیام) به کافکا

با استفاده از زبان برنامه‌نویسی پایتون و کتابخانه kafka می‌توان با کافکا ارتباط برقرار نمود. فایل send_data_to_kafka.py در دایرکتوری src مربوط به این بخش است. پس از ساخت یک نمونه تولیدکنند[4] در برنامه، داده‌ها به کافکا ارسال می‌شوند. داده‌ها با یک کلید uuid ساخته‌شده و تایم‌استمپ[5] آن‌ها نیز برابر با ستون Date/Time داده‌ها قرار میگیرد. تابع send_data_to_kafka این وظیفه را برعهده دارد. در این فایل دو تابع اصلی قرار دارند.

---

[1] Kafka
[2] Redis
[3] Container
[4] Producer
[5] Timestamp

تابع send_real_data_to_kafka داده‌های پروژه را از فایل csv خوانده و با استفاده از تابع send_data_to_kafka به کافکا ارسال می‌نماید. همچنین در این برنامه تابعی با عنوان send_fake_data_to_kafka وجود دارد که داده‌های فیک تولید می‌نماید و به‌صورت لحظه‌ای (با فاصله ۲ ثانیه) به کافکا ارسال می‌نماید که برای بررسی بلادرنگ بودن سیستم مورد استفاده قرار میگیرد.



نمونه داده‌های (پیام) ارسال‌شده به کافکا

گام ششم

داکر و راه‌اندازی Redis

برای راه‌اندازی Redis از فایل redis_compose.yml استفاده می‌شود که با دستور docker-compose -f redis_compose.yml up -d می‌توان راه اندازی محفظه را انجام داد. پس از اینکار دو محفظه ایجاد می‌شوند. محفظه اول مربوط به ردیس است و محفظه دوم رابط کاربری را در آدرس https://localhost:8001 در اختیار کاربر قرار می‌دهد.

دریافت داده از کافکا و ذخیره در Redis

فایل from_kafka_into_redis.py مسئول این بخش است. در این فایل پس از ساخت یک نمونه مصرف‌کننده[6] و اتصال به سرور ردیس داده‌ها از کافکا دریافت می‌شوند. برای دریافت پیام‌های ارسال شده به کافکا به تاپیک مورد نظر subscribe می‌کنیم و سپس به ازای هر پیام قبلی یا ورود پیام جدید ۳ کار انجام می‌شود. به ازای هر روز و ساعت یک کلید به فرمت d/h ساخته می‌شود. به طور مثال 25/16 یعنی روز ۲۵ام ماه و بازه زمانی از ساعت ۱۶ الی ۱۷. این کلید برای شمارش تعداد سفر در هر روز و ساعت استفاده می‌شود. همچنین کلید دیگری به ازای هر پایم با فرمت d/h/details ساخته می‌شود که لیستی شامل اطلاعات تکمیلی‌تر برای سفرهای هر ساعت و روز می‌باشد. در نهایت کلید دگیری با عنوان 1000_latest_trips ساخته می‌شود که شامل لیستی از ۱۰۰۰ سفر آخر می‌باشد. همچنین برای کلیدهای d/h مقدار

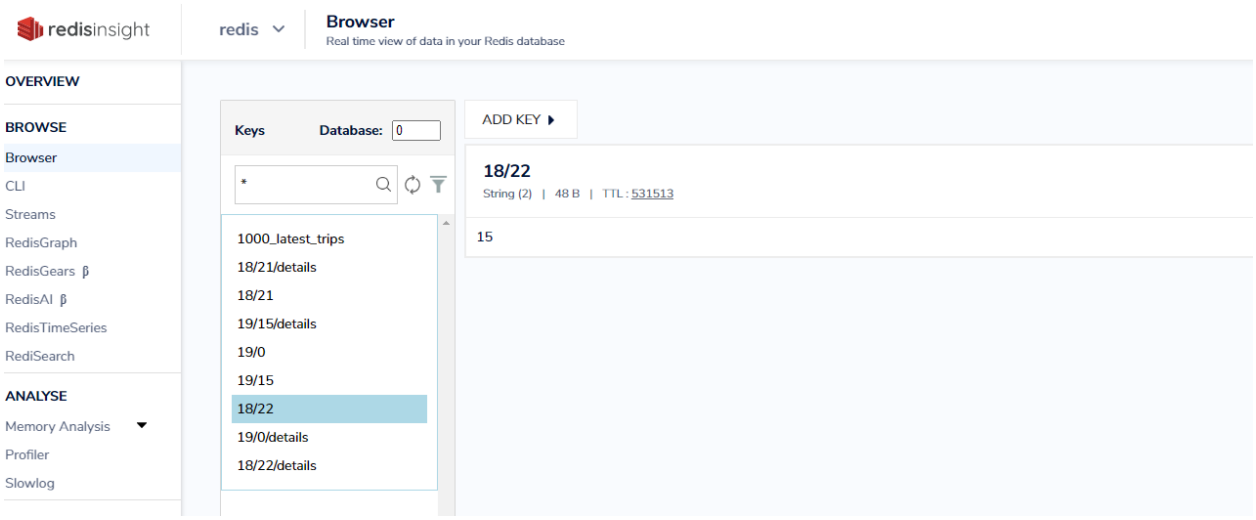
[6] Consumer

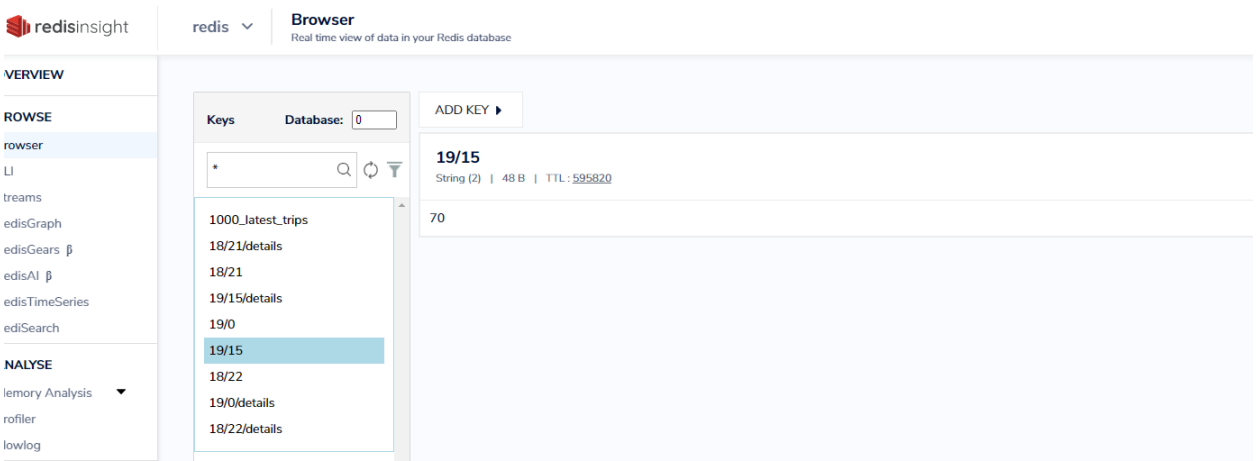
ttl یا زمان expire شدن یک هفته‌ای تنظیم می‌شود. برای کلیدهای d/h/details مقدار ttl ۷ ساعت درنظر گرفته می‌شود تا ردیس دچار شلوغی بیش از حد نگردد.

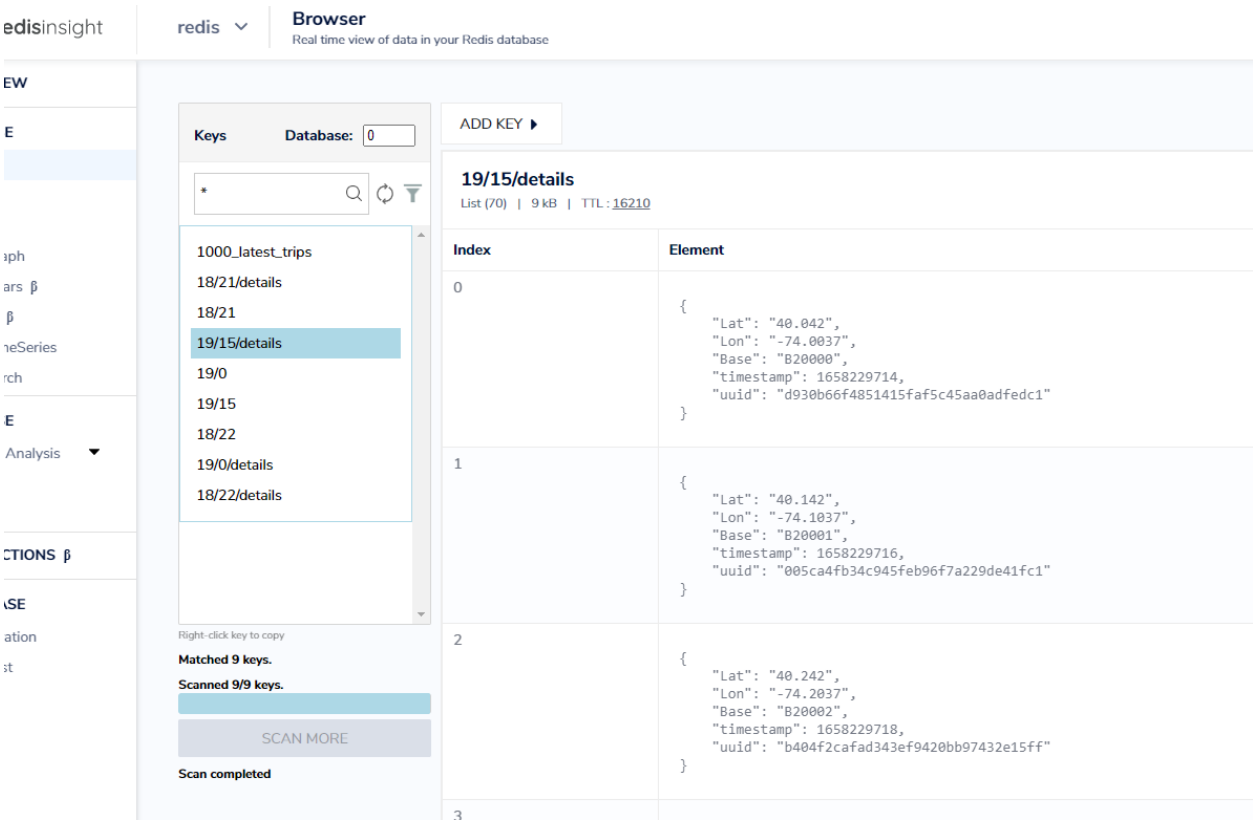
کارهای انجام شده با دریافت هر پیام (هر سفر):

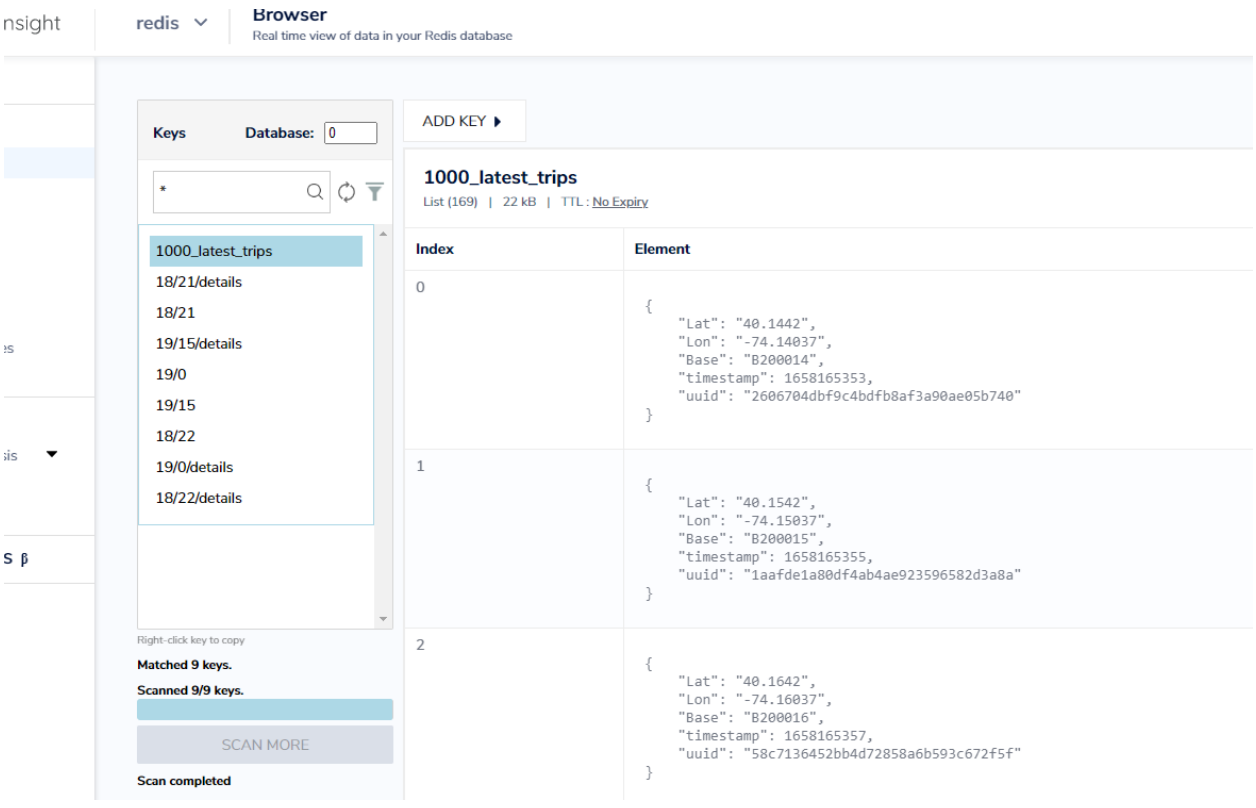
1. افزایش یک واحدی کلید d/h یا ساخت کلید با مقدار اولیه صفر در صورت عدم وجود و همچنین تنظیم ttl
2. ذخیره اطلاعات (push) جزئیتر هر سفر (پیام) در کلیدی با نام d/h/details که یک لیست است و همچنین تنظیم ttl
3. ذخیره اطلاعات (push) جزئ سفر (پیام) در لیستی با کلید 1000_latest_trips و بررسی جهت جلوگیری از تجاوز تعداد سفرها در این لیست از ۱۰۰۰ تا (فقط اطلاعات ۱۰۰۰ سفر آخر نگه‌داری می‌شوند). جدیدترین سفر در انتهای این لیست قرار می‌گیرد

نمونه‌ای از داده‌های ایجاد شده پس به صورت زنده:






redisinsight
redis
Browser
Real time view of data in your Redis database
Keys
Database: 0
*
1000_latest_trips
18/21/details
18/21
19/15/details
19/0
19/15
18/22
19/0/details
18/22/details
Right-click key to copy
Matched 9 keys.
Scanned 9/9 keys.
SCAN MORE
Scan completed
ADD KEY
19/15/details
List (70) | 9 kB | TTL : 16210
Index
Element
0
{
"Lat": "40.042",
"Lon": "-74.0037",
"Base": "B20000",
"timestamp": 1658229714,
"uuid": "d930b66f4851415faf5c45aa0adfedc1"
}
1
{
"Lat": "40.142",
"Lon": "-74.1037",
"Base": "B20001",
"timestamp": 1658229716,
"uuid": "005ca4fb34c945feb96f7a229de41fc1"
}
2
{
"Lat": "40.242",
"Lon": "-74.2037",
"Base": "B20002",
"timestamp": 1658229718,
"uuid": "b404f2cafad343ef9420bb97432e15ff"
}
3


redis
Browser
Real time view of data in your Redis database
Keys
Database: 0
*
1000_latest_trips
18/21/details
18/21
19/15/details
19/0
19/15
18/22
19/0/details
18/22/details
Right-click key to copy
Matched 9 keys.
Scanned 9/9 keys.
SCAN MORE
Scan completed
ADD KEY
1000_latest_trips
List (169) | 22 kB | TTL : No Expiry
Index
Element
0
{
"Lat": "40.1442",
"Lon": "-74.14037",
"Base": "B200014",
"timestamp": 1658165353,
"uuid": "2606704dbf9c4bdfb8af3a90ae05b740"
}
1
{
"Lat": "40.1542",
"Lon": "-74.15037",
"Base": "B200015",
"timestamp": 1658165355,
"uuid": "1aafde1a80df4ab4ae923596582d3a8a"
}
2
{
"Lat": "40.1642",
"Lon": "-74.16037",
"Base": "B200016",
"timestamp": 1658165357,
"uuid": "58c7136452bb4d72858a6b593c672f5f"
}

گام هشتم

در این گام یک backend و frontend مجزا طراحی گردید.

سمت سرور (Backend)

برای طراحی بک‌اند[7] و APIها از نوع Restful از فریم‌ورک[8] FastAPI و زبان پایتون استفاده شد. فایل‌های api.py و api_routes.py مربوط به این بخش هستند. فایل api.py هسته اصلی بک‌اند است و FastAPI را راه‌اندازی می‌نماید. در این بخش route یا مسیرهای API که در فایل api_routes.py قرار دارند به هسته اصلی اضافه می‌شوند. همچنین تنظیمات مربوط به CORS که مسئول بررسی دسترسی به بک‌اند است طوری تنظیم می‌شود که از همه‌ی آدرس‌ها بتوان دسترسی داشت (به دلیل محلی بودن پروژه و اجرای محلی خطر امنیتی وجود ندارد ولی در سرور اصلی باید دقیق تنظیم شود از چه آدرسی می‌شود به بک‌اند دستری داشت).

بخش Swagger بک‌اند در آدرس http://127.0.0.1:9090/docs قابل دسترسی است. برای اجرای پروژه‌های FastAPI نیاز است تا از یک WSGI[9] استفاده شود که به‌طور خلاصه یک ورودی از دنیای بیرون به پردازه بک‌اند است. سرور بک‌اند را می‌توان با دستور زیر در مسیر اصلی (root) پروژه و در پورت 9090 از localhost اجرا نمود:

```
uvicorn src.api:app --host localhost --port 9090
```

مسیرها (Routes)

فایل api_routes.py برای هر API و مسیر مربوطه یک تابع تعریف می‌شود که در ابتدای آن مسیر آن نیز تعریف می‌شود. تمام مسیرهای فعلی API با عبارت trip شروع می‌شوند بنابراین در فایل api.py این مسئله تنظیم می‌شود.

تابع get_required_redis_keys

این تابع با دریافت بازه زمانی کلیدهایی که در ردیس نیاز به بازخوانی دارند برمیگرداند. به صورت پیشفرض یک ساعت اخیر به عنوان بازه درنظر گرفته میشود. همچنین پارامتر details مشخص میکند آیا کلیدها از نوع d/h باشند یا d/h/details. لیستی از کلیدهای مورد نیاز به عنوان خروجی برگردانده میشوند. مثلا برای start=18/5 و end=18/8 کلیدهای 18/5، 18/6، 18/7، 18/8 برگردانده می‌شوند. این تابع از تابع کمکی calc_keys_for_period استفاده می‌نماید که مسئول محاسبه کلیدها بر اساس روز و ساعت شروع و پایان مورد نظر است.

مسیر trips/count/total

این مسیر برای دریافت تعداد سفرهای انجام شده استفاده می‌شود. بطور پیشفرض یک ساعت اخیر را از ردیس می‌خواند و محاسبه کرده و ارسال می‌کند. همچنین با دادن مقدار start و end به عنوان query parameter میتوان بازه زمانی برای شمارش تعداد سفر مشخص نمود. این مقادیر باید به فرمت d/h باشند. این عملیات با استفاده از تابع get_required_redis_keys انجام میشود.

---

[7] Backend
[8] Framework
[9] Web Server Gateway Interface

مسیر trips/count

این مسیر مشابه مسیر قبلی است ولی بهصورت پیشفرض ۶ ساعت اخیر را بررسی میکند و تعداد سفرهای این بازه را برمیگرداند. همچنین می‌توان بازه زمانی به آن داد و یا شمارش را بر اساس یک محل یا مقدار base خاص انجام داد. محاسبات کلید نیز مانند مسیر (api) قبلی با استفاده از تابع get_required_redis_keys انجام می‌شود. مقادیر location و base جهت این امر باید به عنوان پارامتر جستجو داده شوند. همچنین امکان فیلتر بر اساس مکان و بازه زمانی دلخواه نیز فراهم است.

مسیر /trips/details

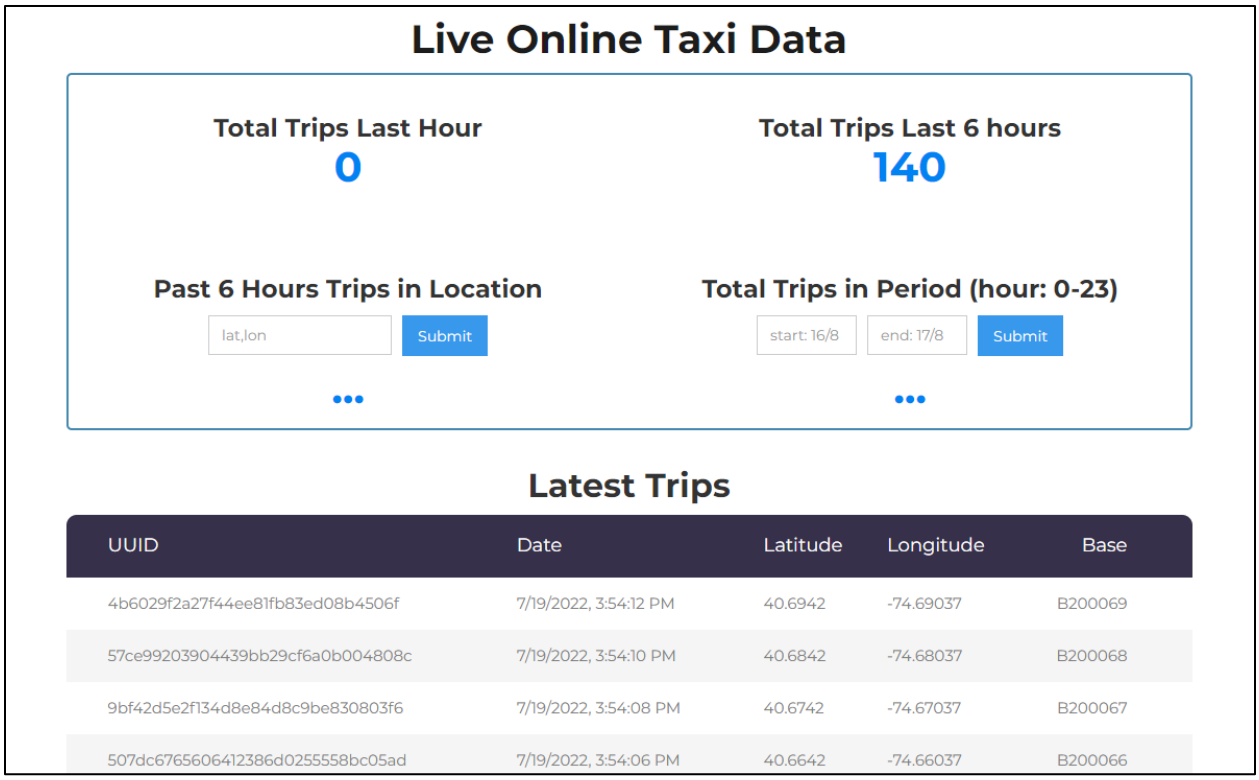
این مسیر نیز با استفاده از تابع get_required_redis_keys و استفاده از کلیدهای d/h/details اطلاعات جزئیتری از سفرها در بازه دلخواه برمیگرداند.

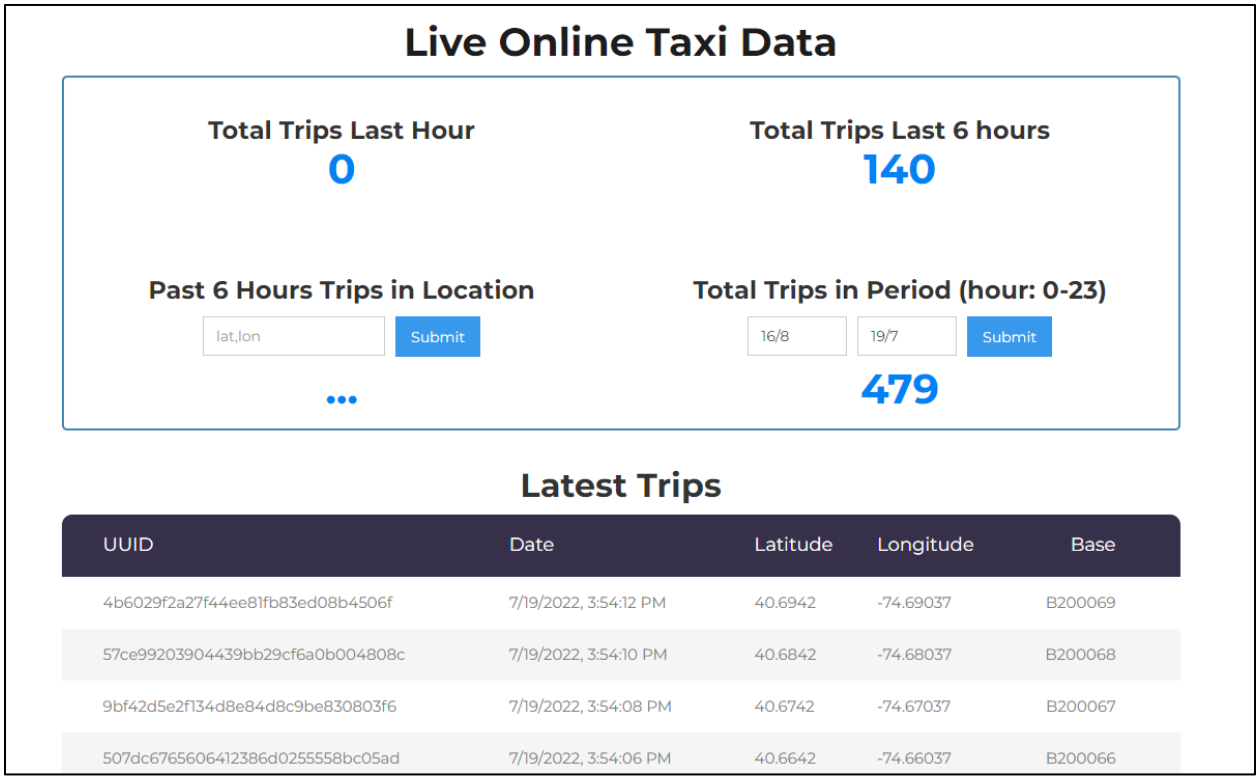
مسیر /trips/latest/1000

این مسیر با استفاده از کلید 1000_latest_trips در ردیس، ۱۰۰۰ سفر اخیر را برمیگرداند.

سمت کاربر (Frontend)

یک فایل html داریم که فایل‌های styles.css و script.js را فراخوانی میکند تا اطلاعات خواسته شده را از بک‌اند دریافت و به کاربر نمایش دهد. با استفاده از جاوااسکریپت API های مربوطه فراخوانی می‌شوند تا صفحه html به حالت پویا مقادیر خود را بروز کند. بدون نیاز به refresh کردن مقادر به صورت لحظه‌ای در صفحه بروزرسانی می‌شوند. همچنین برای بخش ۱۰۰۰ سفر اخیر فقط اختلاف لیست ۱۰۰۰تایی سفرهای قدیمی و جدید در هر لحظه مقایسه می‌شوند و فقط داده‌های جدید به جدول مربوطه در صفحه اضافه می‌شوند (بدون نیاز به refresh). همچنین با یکبار پر کردن و فشردن کلمه submit برای بخشهای نمایش‌داده شده دادهای مربوط به آن بخش نیز بهصورت پویا از آن لحظه نمایش داده می‌شود. به طور مثال با دادن شروع و پایان بازه در بخش Total Trips in Period و فشردن submit تعداد سفرهای آن بازه زمانی نیز نمایش داده می‌شود و از آن پس به صورت لحظه‌ای بروززسانی می‌شود.


Live Online Taxi Data
Total Trips Last Hour
0
Total Trips Last 6 hours
140
Past 6 Hours Trips in Location
lat,lon
Submit
Total Trips in Period (hour: 0-23)
start: 16/8
end: 17/8
Submit
Latest Trips
UUID
Date
Latitude
Longitude
Base
4b6029f2a27f44ee81fb83ed08b4506f
7/19/2022, 3:54:12 PM
40.6942
-74.69037
B200069
57ce99203904439bb29cf6a0b004808c
7/19/2022, 3:54:10 PM
40.6842
-74.68037
B200068
9bf42d5e2f134d8e84d8c9be830803f6
7/19/2022, 3:54:08 PM
40.6742
-74.67037
B200067
507dc6765606412386d0255558bc05ad
7/19/2022, 3:54:06 PM
40.6642
-74.66037
B200066


Live Online Taxi Data
Total Trips Last Hour
0
Total Trips Last 6 hours
140
Past 6 Hours Trips in Location
lat,lon
Submit
Total Trips in Period (hour: 0-23)
16/8
19/7
Submit
479
Latest Trips
UUID
Date
Latitude
Longitude
Base
4b6029f2a27f44ee81fb83ed08b4506f
7/19/2022, 3:54:12 PM
40.6942
-74.69037
B200069
57ce99203904439bb29cf6a0b004808c
7/19/2022, 3:54:10 PM
40.6842
-74.68037
B200068
9bf42d5e2f134d8e84d8c9be830803f6
7/19/2022, 3:54:08 PM
40.6742
-74.67037
B200067
507dc6765606412386d0255558bc05ad
7/19/2022, 3:54:06 PM
40.6642
-74.66037
B200066