Händelse- och väderinsikts app

Streamlit

Arina Godman

EC Utbildning

Projekt i Data Science

202410

Abstract

This project focuses on the development of an interactive web application designed to provide users with a comprehensive overview of public safety in Sweden and real-time weather conditions. The data was extracted from two APIs: Polisens API, which includes events in Sweden reposted av Police, and the SMHI(Swedish Meteorological and Hydrological Institute) API, which provides weather information. Extracted data was processed and partially stored in Google BigQuery for further analysis. The application, deployed using Streamlit, consists of three interactive pages: Events, Analysis, and Weather, offering users insights into police-reported incidents and current weather and driving conditions across various Swedish cities.

The project was developed using an Agile approach, which facilitated efficient collaboration through regular meetings, rapid iterations, and a focus on tasks that were both engaging and productive for the team. The successful outcome of this project is attributed to the teamwork of myself, Arina Godman, alongside my colleagues Shriya Walia and Filip Östlund. I am grateful for the collaborative efforts that contributed to the completion of this project.

Förkortningar och Begrepp

SMHI – Sveriges meteorologiska och hydrologiska institut

API – applikationsprogrammeringsgränssnitt

ETL – Extract, transform, load

Innehållsförteckning

1 Inledning

Sverige har länge betraktats som ett tryggt land att bo i, men under de senaste åren har rapporterna om explosioner, skjutningar och ökande gängkriminalitet blivit allt vanligare. Denna utveckling har lett till ett ökat behov av verktyg som kan ge en överblick över offentliga säkerhetsdata och händelser. I takt med detta har även trafiksäkerhet blivit en viktig aspekt för många människor, där realtidsinformation om väder och körförhållanden kan ha stor påverkan på vardagliga beslut.

Syftet med detta arbete är att undersöka om det är möjligt att bygga en interaktiv applikation som utnyttjar offentliga data från Polisens och SMHI API för att visualisera händelser och väderinformation i realtid. Applikationen ska även ge användaren möjlighet att filtrera fram områden med flest rapporterade incidenter och presentera aktuell information om väg- och körförhållanden.

För att uppfylla syftet kommer följande frågeställningar att besvaras:

1. Går det att integrera offentliga API för händelser rapporterade av Polisen och väderdata i en användarvänlig applikation som visar information i realtid?
2. Hur kan denna applikation bidra till att förbättra medvetenheten om säkerhet och vägförhållanden i Sverige?

Bakgrunden till arbetet motiveras av det ökande behovet av att analysera och visualisera offentliga säkerhetsdata, i en tid där både kriminalitet och väderförhållanden kan påverka människors liv och beslut. Detta gäller särskilt för de som söker bostad eller är beroende av biltransporter på svenska vägar.

2 Teori

2.1 Data visualisering

Datavisualisering är en viktig komponent inom modern informationshantering, där komplexa dataset omvandlas till visuella representationer som underlättar förståelsen för användaren. En av de mest effektiva formerna av datavisualisering är kartbaserade visualiseringar, som gör det möjligt att lokalisera händelser geografiskt och få en tydlig överblick över olika regioner och deras tillstånd.

2.1.1 Folium

Folium är ett bibliotek som primärt används för att skapa interaktiva kartor genom att integrera Python-kod med Leaflet.js. Detta gör det enkelt att bygga kartbaserade visualiseringar, vilket är särskilt värdefullt inom geografisk dataanalys. Folium stöder olika bakgrundskartor och lager, vilket gör det möjligt för användare att snabbt identifiera mönster och trender i data (Folium Documentation, 2024).

2.1.2 Pyecharts

Pyecharts är ett kraftfullt bibliotek för att skapa interaktiva grafer och diagram i Python. Pyecharts möjliggör smidig visualisering av tidsserier och kategoriska data, vilket hjälper användarna att identifiera utvecklingar och förändringar över tid.

2.2 API-teknologi

Ett API (Application Programming Interface) är en uppsättning definierade metoder som tillåter olika system att kommunicera och utbyta data med varandra. Genom att använda API kan externa dataflöden integreras i en applikation i realtid, utan att behöva lagra hela datasetet lokalt. API möjliggör också kontinuerlig uppdatering och synkronisering av data från flera källor.

2.2.1 Polisens API[1]

Detta API tillhandahåller realtidsdata om polisiära händelser i Sverige, såsom brott, olyckor och andra incidenter. Genom att göra API-anrop till Polisens API kunde applikationen hämta uppdaterad information om händelser och visa dem på kartan.

API ger användare tillgång till 500 senaste händelser som uppdateras oregelbundet men i realtid. Incidenter i denna API är inledande rapporter som sedan kan ligga till grund för en notis på Polisens webbsida.

2.2.2 SMHI API för Meteorologisk Analys (MESAN)[2]

SMHI erbjuder flera olika offentliga API, och i detta projekt användes specifikt MESAN-API, som ger detaljerade meteorologiska data. MESAN innehåller viktiga parametrar som temperatur, vindbyar, nederbörd och sikt. Denna information används för att bedöma körförhållanden i realtid för olika städer i Sverige, vilket är viktigt för att ge användarna en uppdaterad bild av trafiksäkerheten baserat på aktuella väderförhållanden.

2.3 Agil-projektledning

Agil projektledning är en flexibel och iterativ arbetsmetod som ofta används inom mjukvaruutveckling. Agila metoder, såsom Scrum eller Kanban, bygger på att dela upp arbetet i korta iterationer eller "sprintar", där målen för varje sprint är tydligt definierade och resultaten utvärderas löpande. Detta gör det möjligt att snabbt anpassa projektet efter nya krav och förbättra

[1] API över polisens händelser | Polismyndigheten
[2] SMHI Open Data API Docs - Meteorological Analysis MESAN

arbetsprocessen kontinuerligt. Denna samarbetsform främjar innovation och säkerställer att alla bidrar till projektets framgång (Agile Alliance, n.d.).

2.4 Google BigQuery

Google BigQuery är en kraftfull och skalbar datalagrings- och analyslösning som ingår i Google Cloud Platform. BigQuery är utformad för att hantera och analysera stora datamängder, vilket gör den idealisk för företag och organisationer som arbetar med big data. Genom att använda en kolumnbaserad datalagringsteknik erbjuder BigQuery snabbare fråga- och analysprestanda jämfört med traditionella, raderbaserade databaser, vilket gör det möjligt att hantera data mer effektivt och kostnadseffektivt i storskaliga miljöer (Google Cloud, n.d.).

En av de mest framstående egenskaperna hos BigQuery är dess förmåga att köra SQL-liknande frågor på stora datamängder i realtid. Användare kan formulera komplexa frågor och få snabba svar, vilket underlättar både datavisualisering och beslutsfattande i realtid. Tjänsten är dessutom helt hanterad, vilket innebär att användarna inte behöver oroa sig för infrastrukturen eller underhållet, utan kan fokusera helt på dataanalysen.

En begränsning med BigQuery är dock avsaknaden av stöd för primary key constraints eller andra liknande unika nyckelbegränsningar. Detta innebär att användare måste hantera duplicerade data manuellt, vilket kan kräva ytterligare arbete för att säkerställa datakvalitet och korrekthet i dataset som regelbundet uppdateras eller är under kontinuerlig bearbetning.

2.5 Streamlit

Streamlit är ett öppen källkod-bibliotek i Python som förenklar skapandet av interaktiva applikationer. Verktyget möjliggör att dataforskare enkelt kan skapa och distribuera datavisualiseringar och analysapplikationer, vilket är användbart för snabb prototypframtagning (Streamlit, 2023).

3 Metod

3.1 3.1 Projektöversikt

I detta kapitel presenteras de metoder och tekniker som användes för att utveckla den interaktiva applikationen för att visualisera offentliga säkerhetsdata och väderinformation. Arbetet följde en agil metodik, vilket möjliggjorde kontinuerlig feedback och anpassning av projektet under utvecklingsprocessen.

3.2 Agilt arbetssätt

Projektet genomfördes enligt en agil arbetsmetod, där teamet regelbundet hade möten två gånger i veckan för att diskutera framsteg, identifiera eventuella hinder, justera prioriteringar och formulera nya mål. Denna iterativa process möjliggjorde en snabb anpassning till förändringar och en effektiv problemlösning under projektets gång. Genom att arbeta agilt kunde projektet hållas på rätt spår och utvecklingen genomföras på ett strukturerat och effektivt sätt. Varje teammedlem bidrog inom sina intresseområden, vilket främjade motivation och effektivitet.

I arbetsfördelningen ansvarade jag, Arina Godman, för projektets idé och utveckling av ETL-pipelinen för Polisens data samt skapandet av Streamlit-sidan för Händelser. Filip Östlund ansvarade för hanteringen av GitHub, utveckling av analysdelen i Streamlit samt korrigering av mindre buggar. Shriya Walia var ansvarig för utformningen av Streamlit-sidans användargränssnitt samt för Väder-sidan.

3.3 Teknisk Infrastruktur

3.3.1 Val av verktyg och teknologier

För att bygga applikationen användes följande verktyg och teknologier:

- **Python**: Programmeringsspråket för backend-utveckling, vilket möjliggjorde datahantering och API-integration. I projektet användes Python version 3.10.9 då det var mest kompatibel med alla bibliotek som användes (t.ex. Folium).
- **Streamlit**: Ett ramverk för att bygga webbapplikationer, vilket användes för att skapa det interaktiva användargränssnittet.
- **Google BigQuery**: En kraftfull databaslösning för lagring och analys av stora datamängder, vilket användes för att lagra data från Polisens API. Big Query blev vald då det är ett bra verktyg för lagring av stora mängder av data och ger alla i arbetsgruppen möjlighet att använda samma data utan fördröjningar (om man jämför med lokala databas).
- **Folium**: Ett Python-bibliotek som användes för att skapa interaktiva kartor som visualiserar händelser på en geografisk karta. Folium blev vald vid arbete på detta projekt då det är ett verktyg som är mycket enkel att utveckla.
- **Pyecharts**: Användes för att generera interaktiva diagram och visualiseringar av analyserade data. Pyecharts har bra kompatibilitet med Streamlit.

3.3.2 API-integration

Två offentliga API användes för att hämta data:

- **Polisens API**: Användes för att extrahera information om polisiära händelser i realtid. Detta inkluderade uppgifter om händelsetyper, platser (namn, longituder, latituder), sammanfattning och tidpunkter för incidenter.

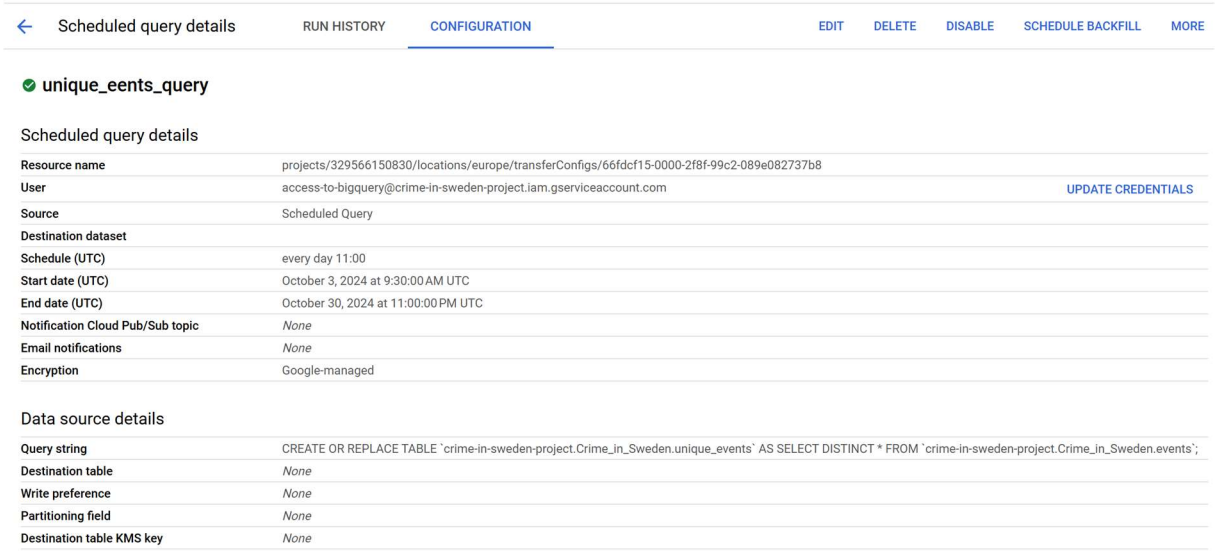
- **SMHI API**: Användes för att få meteorologiska data, inklusive temperatur, vindförhållanden och nederbörd. Specifikt användes API för Meteorologisk Analys (MESAN) som erbjuder detaljerade väderparametrar.

3.4 Datahantering

3.4.1 Datainsamling och bearbetning

Datainsamlingen inleddes med att anropa de valda API för att hämta nödvändig information. För Polisens API utvecklades en ETL-pipeline (Extract-Transform-Load) som möjliggjorde en effektiv inhämtning, transformation och lagring av data i BigQuery-databasen. Denna pipeline inkluderade också automatisering av dataextraktionen med hjälp av Windows Task Scheduler, vilket görs dagligen, samt loggning för att säkerställa att processen genomfördes framgångsrikt.

Eftersom BigQuery inte har en unik nyckelrestriktion (unique-key constraint), var det inte möjligt att lagra endast unika händelser rapporterade av Polisen. För att lösa detta problem skapades en separat tabell som innehöll enbart unika händelser. Detta genomfördes med hjälp av en SQL-query (SELECT DISTINCT), som också automatiserades med hjälp av Scheduled Query för att säkerställa kontinuerlig uppdatering av datan.



Figur 1. Scheduled Query i Google BigQuery

Data från SMHI-API hämtades direkt till Streamlit, där den transformerades och presenterades för användaren i ett lättförståeligt format. Denna metod möjliggjorde en sömlös integration av realtidsväderinformation i applikationen.

3.4.2 Datavisualisering

Data visualisering är en central del av applikationen och genomfördes med hjälp av verktygen Folium och Pyecharts. Folium används för kartbaserade visualiseringar, vilket gör det möjligt att representera polisiära händelser geografiskt. Genom att använda Folium kan användarna interaktivt utforska kartan och se var olika incidenter har inträffat, vilket ger en tydlig visuell representation av säkerheten i olika områden.

För att ytterligare förbättra användarupplevelsen och möjliggöra djupare insikter användes Pyecharts för att skapa olika typer av diagram och grafer. Dessa visualiseringar ger användarna en bättre förståelse för datamönster och trender, vilket är avgörande för att analysera händelser och väderförhållanden.

4 Resultat och Diskussion

4.1 Application

Den interaktiva applikationen byggdes enligt projektets ursprungliga syfte och funktionalitet, vilket var att tillhandahålla en översikt av allmän säkerhet och väderförhållanden i Sverige. Genom att använda data från Polisens API och SMHI API, samt lagring och hantering i Google BigQuery, har applikationen lyckats integrera data i realtid på ett användarvänligt sätt. Nedan presenteras resultat för varje sida i applikationen, följt av en diskussion om prestanda och användarupplevelse.

Själva applikation är på länk: https://policeapi.streamlit.app/

4.1.1 Intro-Sidan

Intro-sidan är den första sidan som användaren möter och ger en kort introduktion till applikationens innehåll. Här får användaren en överblick av projektet och hur informationen är strukturerad. Sidan fungerar som en guide som hjälper användaren att förstå hur de kan navigera mellan Events-, Analysis- och Weather-sidorna för att få den information de söker.

Figur 2. Intro-sidan (Streamlit)

4.1.2 Händelser-sidan

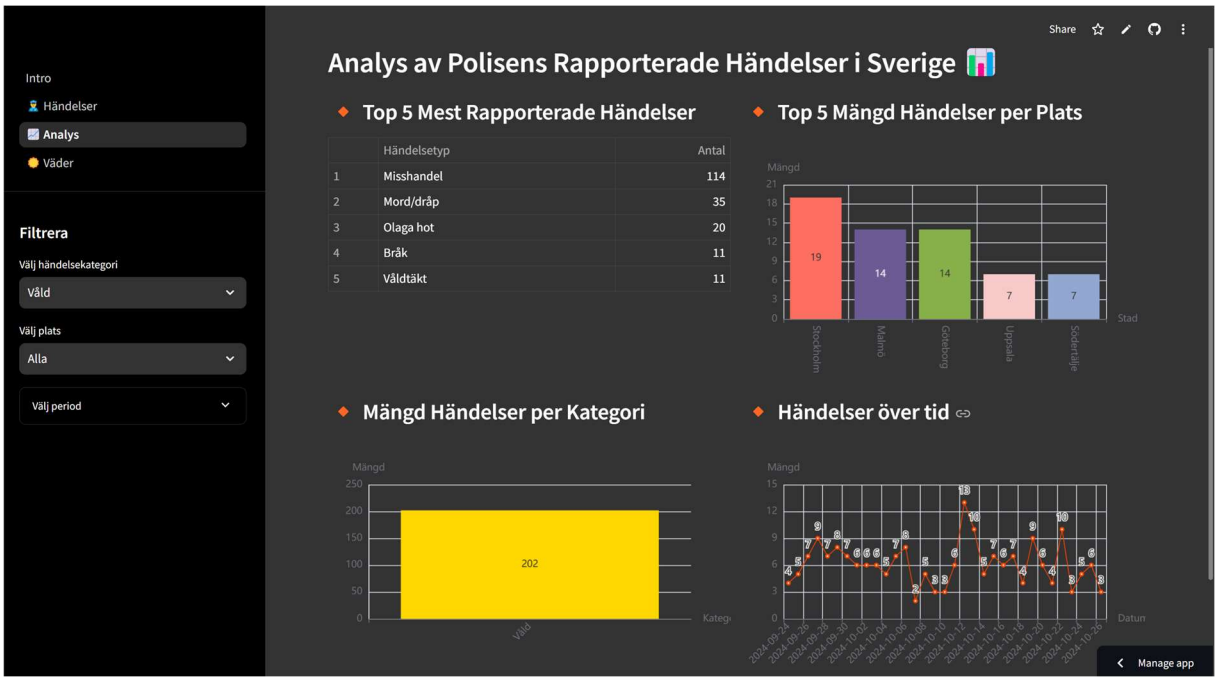
Här kan användare se en kartbaserad visualisering av incidenter rapporterade av polisen i olika svenska städer. Med hjälp av filter kan användare anpassa vyn baserat på plats, kategori av händelse (varje kategori har även sin egen ikon) och tidsperiod. Visualiseringar av incidenter implementerades genom Folium, vilket möjliggjorde en interaktiv karta med möjligheter att "hovera" över städer för incidentantal, samt att klicka på städer för att se detaljer i en sidovy.

Figur 3. Händelser-sidan (Filtrerad för Skottlossning, i sidofältet ser man beskrivning av händelser)

4.1.3 Analys-sidan

Analysis-sidan fokuserade på att ge användare insikter om mönster och trender i polisiära händelser. Genom Pyecharts visualiserades data med grafer som presenterade exempelvis fördelning av olika händelsetyper, förändringar över tid, och incidenter i specifika områden.

På denna sida finns samma filtrar som på Händelser-sidan.



Figur 4. Analys-sidan (Filtrerad för kategori "Våld")

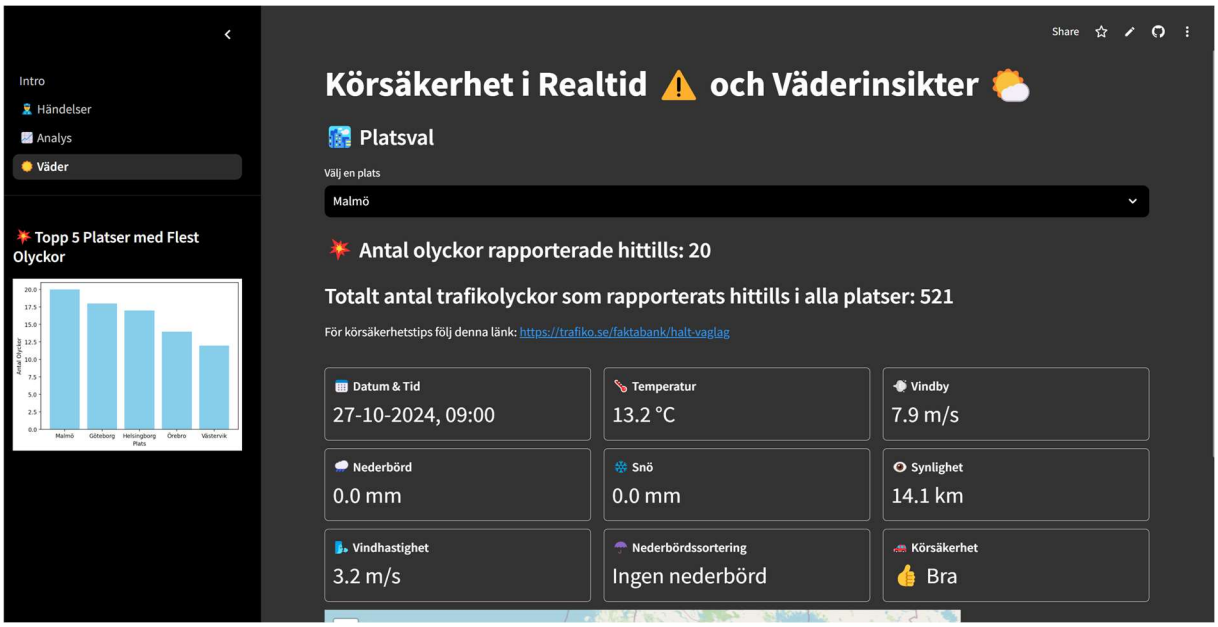
4.1.4 Väder- och Körförhållanden-sidan

Väder-sidan visade data från SMHI API i realtid, vilket inkluderade väderparametrar som temperatur, vindbyar, nederbörd och sikt. Dessa parametrar presenterades som körförhållanden för användare i

olika städer, så att användaren kunde planera sin resa mer effektivt. Genom Streamlit erbjöds en dynamisk interaktion där användare kunde välja specifika städer för att få detaljerad väderinformation.

Under översiktsinformation finns även en karta där användaren kan se var staden ligger. I sidofältet kan man se topp 5 städer med flest rapporterade trafikolyckor.

Under kartan finns möjlighet för användaren att lämna Feedback och rapportera hur är körförhållanden i hennes eller hans stad.



Figur 5. Väder och Körsäkerhet-sidan

4.2 Begränsningar och utmaningar

Den slutliga applikationen uppfyller de initiala målen och erbjuder användarna en effektiv lösning för att följa både polisiära händelser och väderförhållanden i realtid. Nedan diskuteras de viktigaste aspekterna av projektet samt några utmaningar som uppstod.

4.2.1 Streamlit-begräsningar

Applikationen ger användare möjlighet att snabbt få en överblick över aktuella polisiära incidenter och väderförhållanden. Detta är särskilt värdefullt för boende som vill bedöma tryggheten i sina områden och för bilförare som söker realtidsinformation om vägförhållanden för att kunna fatta säkra resebeslut. Under utvecklingen stötte vi dock på vissa begränsningar med Streamlit.

I projektets inledande skede försökte vi förbättra användarvänligheten genom att integrera Händelser- och Analysfunktionerna på en och samma sida, uppdelade i olika flikar (tabs). Målet var att ge användaren möjlighet att applicera filter på Händelser-fliken och se en anpassad analys direkt på Analys-fliken, baserat på samma filterinställningar. Tyvärr upptäckte vi att Streamlit renderingsfunktion inte stödjer samtidig uppdatering av information mellan flikar.

Detta ledde till att de filter som användes på Händelser-fliken inte automatiskt följde med till Analys-fliken och vice versa. Uppdateringar av information på en flik återspeglades inte heller automatiskt på den andra fliken. För att säkerställa en smidig och pålitlig användarupplevelse beslutade vi därför att separera funktionerna i två distinkta sidor: Händelser och Analys.

4.2.2 Prestanda och Utmaningar med BigQuery

BigQuery erbjöd fördelen av att kunna lagra stora mängder data, men den saknar stöd för primary key constraint, vilket komplicerade lagringen av unika händelser. Detta löstes genom en schemalagd SQL-fråga (SELECT DISTINCT) som upprättade en tabell av unika incidenter varje dag. Processen ökade lagringsbehovet, men säkerställde samtidigt att redundanta data undveks.

BigQuery databas innehåller två tabeller: ’events’ (för alla händelser) och unique_events (unika händelser som filtreras dagligen från ’events’-tabellen).

4.2.3 API-begränsningar och Realtidsprestanda

Under utvecklingen av applikationen började vi med att arbeta lokalt innan vi gjorde deployment på Streamlit. I denna tidiga fas hade användarna möjlighet att välja alla städer samtidigt på Väder-sidan, vilket gjorde det möjligt att få en överblick över körförhållandena i hela landet. Men efter att applikationen lades upp online på Streamlit blev det tydligt att dess beräkningsresurser är begränsade, och datahämtningen från SMHI-API blev mycket långsam. Detta ledde till att användarna fick vänta i över 10 minuter för att se all information.

För att förbättra prestandan och undvika långsamhet orsakad av repetitiva API-anrop, särskilt från SMHI API, beslutade vi att ta bort möjligheten för användarna att välja alla städer samtidigt. Denna förändring resulterade i en betydande ökning av hastigheten på applikationen och en därmed förbättrad användarupplevelse.

5 Slutsatser

Under utvecklingen av denna interaktiva applikation har vi framgångsrikt integrerat offentliga data från både Polisens och SMHI API för att erbjuda användarna en översikt över offentliga säkerhetsdata och väderinformation i realtid. Applikationen har visat sig vara ett värdefullt verktyg för att öka medvetenheten om säkerhet och körförhållanden i Sverige.

Vi har visat att det är möjligt att integrera dessa API och skapa en användarvänlig plattform där data från både Polisens och SMHI API presenteras i realtid. Användarna kan enkelt navigera genom olika kategorier av händelser och väderförhållanden, vilket ger dem en helhetsbild av situationen i deras närområde.

Applikationen erbjuder användarna realtidsinformation om aktuella händelser i Sverige och väderförhållanden, vilket hjälper dem att fatta informerade beslut angående sin säkerhet och resande. Genom att filtrera data kan användarna snabbt identifiera riskfyllda områden och anpassa sina resor efter aktuella vägförhållanden, vilket ökar den allmänna medvetenheten och säkerheten.

Trots de positiva resultaten finns det områden för framtida förbättringar. En ökad kapacitet för hantering av samtidiga API-anrop skulle möjliggöra mer omfattande datavisualisering utan att påverka prestandan. Dessutom skulle fler filteralternativ och anpassade vyer ytterligare kunna förbättra användarupplevelsen och ge djupare analyser av säkerhetsdata. Att inkludera fler funktioner, som historisk dataanalys och användarinteraktioner, skulle kunna göra applikationen ännu mer värdefull för användarna.

Genom att adressera dessa förbättringsmöjligheter kan vi fortsätta utveckla en applikation som inte bara informerar utan också aktivt bidrar till ökad säkerhet och medvetenhet om offentliga händelser i Sverige.

6 Självutvärdering

1. Utmaningar du haft under arbetet samt hur du hanterat dem.

Jag beskrev de tekniska utmaningar i rapporten, men det var också utmaningar med arbetssättet. Det var svårt att lita på folk :D Att ge dem möjlighet att göra sin grej utan störning. Men det visade sig snabbt att de har kommit upp med även bättre idéer och lösningar än jag själv, så jag insåg ganska snabbt att vi kan arbeta oberoende men tillsammans och producera ännu bättre projekt än bara en av oss skulle har gjort det hela.

Det var också utmaning med att arbeta med Git: att comitta och merga branches och arbeta i egna enviroments då vi behövde säkerställa att alla av oss kör koden utan problem. Vi var tydliga med varandra om vilka dokument vi jobbar med, för att undvika konflikt vid merges och delade med varandra enviroment i början, innan vi behövde deploya appen på Streamlit.

2. Vilket betyg du anser att du skall ha och varför.

Jag tycker att vi alla 3 förtjänar VG då vi har gjort jättebra jobb och har utvecklats i Agilt arbetssättet så lång det går i ett sådant projekt.

3. Något du vill lyfta fram till Antonio?

Jättespännande kurs! Tack för möjligheten och vi ses i april!

Källförteckning

Agile Alliance. (n.d.). 12 principer bakom Agile-manifestet. Hämtad 29 oktober, 2024, från https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/

BigQuery Documentation. (2024). What is BigQuery? Google Cloud. Hämtad 29 oktober, 2024, från https://cloud.google.com/bigquery

Folium Documentation. (2024). Folium — Folium 0.18.0 documentation. Hämtad 29 oktober, 2024, från https://python-visualization.github.io/folium/latest/

Streamlit. (2023). Vad är Streamlit? Hämtad 29 oktober, 2024, från https://streamlit.io/