# **Verslag: Client-Serverapplicatie – Analyse van Los Angeles Arrestatiegegevens (2020 - heden) Warre Snaet**

## **1. Analyse van de Dataset**

Voor deze opdracht heb ik gekozen om de dataset **"Los Angeles Arrestation Data 2020 to present"** te analyseren en verwerken in mijn applicatie. Deze dataset vond ik bijzonder interessant, omdat ze inzicht biedt in de aard, tijdstippen en spreiding van arrestaties in een grootstedelijk gebied als Los Angeles.

### **Leeftijdsanalyse**

Bij een eerste verkenning van de data ben ik nagegaan welke leeftijden het vaakst voorkomen bij arrestaties. Daaruit blijkt dat de piek ligt bij personen eind de 20 en begin de 30. Dit resultaat is logisch te verklaren: jongvolwassenen zijn vaak actiever en nemen mogelijk meer risico’s dan ouderen.

### **Tijdsanalyse**

Vervolgens onderzocht ik het aantal arrestaties per dag en per uur. We zien dat er in het weekend doorgaans minder arrestaties plaatsvinden. Door de week valt op dat de meeste arrestaties plaatsvinden rond 20:00 uur 's avonds.

### **Arrestaties per maand en in de tijd**

Bij analyse van het aantal arrestaties per maand, viel het op dat sinds het begin van 2020 (de start van de coronapandemie) het aantal arrestaties opvallend gedaald is. Ook is er elk jaar een lichte daling te zien in december, vermoedelijk door de feestdagen.

### **Meest voorkomende aanklachten ("Charges")**

De meest voorkomende "charge" in de dataset is **"Miscellaneous Other Violations"**, maar omdat deze niet erg specifiek is, heb ik ook gekeken naar de tweede meest voorkomende aanklacht: **"Aggravated Assault"** (zware mishandeling).

## **2. Preprocessing**

Voorafgaand aan de visualisaties en zoekopdrachten heb ik een aantal preprocessing-stappen uitgevoerd:

1. **Controleren op ontbrekende waarden** in de dataset.
2. **Converteren van datums** naar correcte Date objecten.
3. **Detectie van outliers** in de leeftijdskolom (bv. geen personen van 150 jaar oud).
4. **Omzetten van coördinaten** naar het gestandaardiseerde GeoJSON-formaat.

Al deze stappen zijn met succes uitgevoerd.

## **3. Visualisaties (Plots)**

Ik had vier visualisaties gepland; drie hiervan zijn volledig verwerkt in de preprocessingfase. De tweede plot (de geografische heatmap) werd geïntegreerd als live query in de clientapplicatie.

### **Plot 1 – Aantal arrestaties per geslacht**

Een boxplot toont het verschil tussen mannelijke en vrouwelijke arrestaties. Mannen zijn verantwoordelijk voor ongeveer **240.000** arrestaties, terwijl vrouwen er slechts **60.000** tellen — een viervoudig verschil. Bij het uitsplitsen naar type misdrijf blijkt dat **vrouwen vaker gearresteerd worden voor prostitutie** dan mannen.

### **Plot 2 – Geografische hotspots**

Deze plot werd niet opgenomen in de preprocessing, maar is verwerkt als een interactieve query in de applicatie. Hierin wordt een heatmap getoond van geografische hotspots van arrestaties binnen Los Angeles.

### **Plot 3 – Gestapelde staafdiagram: Demografie vs. misdrijven**

Ik deelde leeftijden op in volgende categorieën:  
 ['10-17', '18-25', '26-35', '36-45', '46-55', '56-65', '66+'].  
 Deze combineerde ik met descent codes en type overtreding. Hieruit blijkt dat **Hispanics in de leeftijdscategorie 26-35 het vaakst gearresteerd worden**. De volledige top drie wordt ingenomen door deze groep.

### **Plot 4 – Violin plot: Leeftijd vs. misdrijf**

Voor deze visualisatie koos ik de **top 5 charges** en onderzocht ik hoe de leeftijden daarbij verdeeld zijn. De piek ligt telkens rond de leeftijd van 30, wat consistent is met eerdere analyses. Hieruit blijkt dat de **leeftijdsverdeling weinig varieert tussen verschillende charges**.

## **4. Bespreking van de Gekozen Zoekopdrachten**

### **Query 1 – *Arrestaties per Gebied en Tijdsperiode***

* **Parameters**: Gebied, start- en einddatum, leeftijdsbereik
* **Metadata**: Gebieden in dropdown, datums automatisch ingesteld (volledige tijdsperiode)

### **Query 2 – *Trend van Specifieke Overtreding over Tijd***

* **Parameters**: Arrestatietype, tijdsgranulariteit (dagelijks tot jaarlijks)
* **Metadata**: Arrestatietypes in dropdown, tijdseenheid instelbaar

### **Query 3 – *Demografische Analyse van Arrestaties***

* **Parameters**: Geslacht, descent code, charge group, arrestatietype
* **Metadata**: Alles voorzien via dropdown of selectiemenu

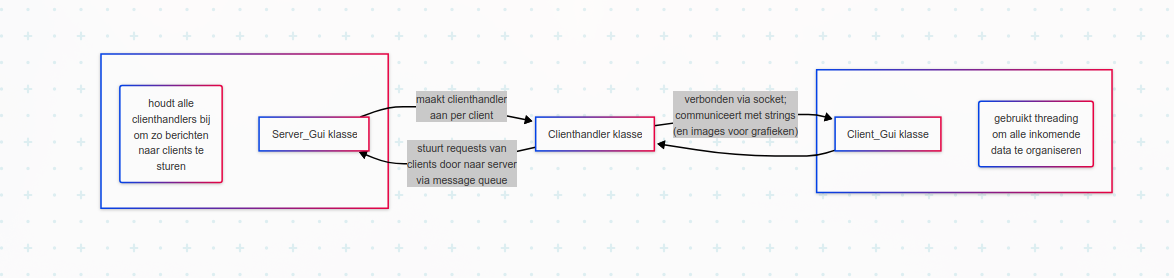
Weergave: barplot

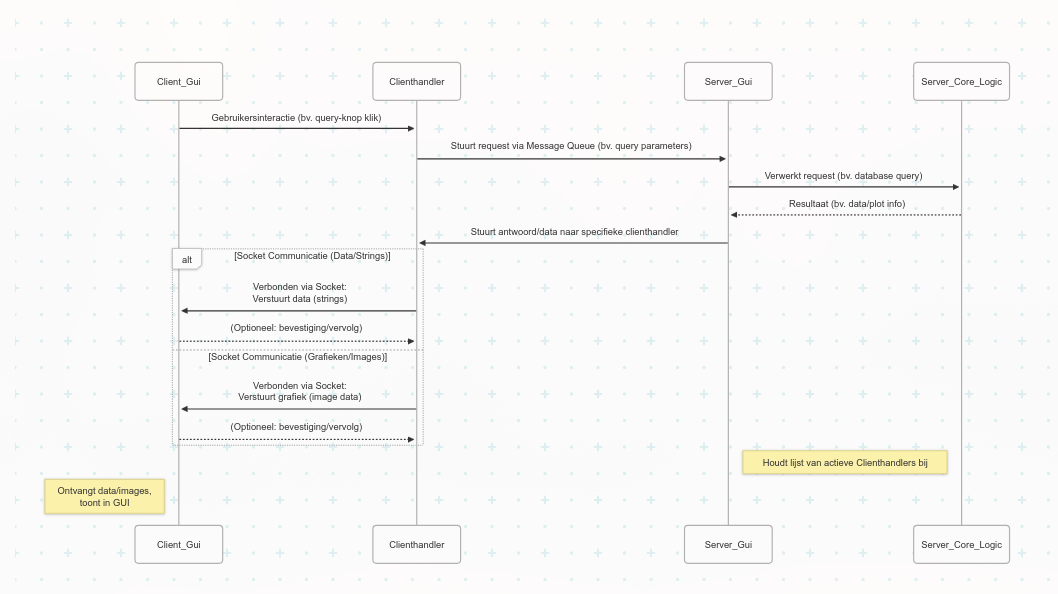
### **Query 4 – *Geografische Hotspots van Arrestaties***

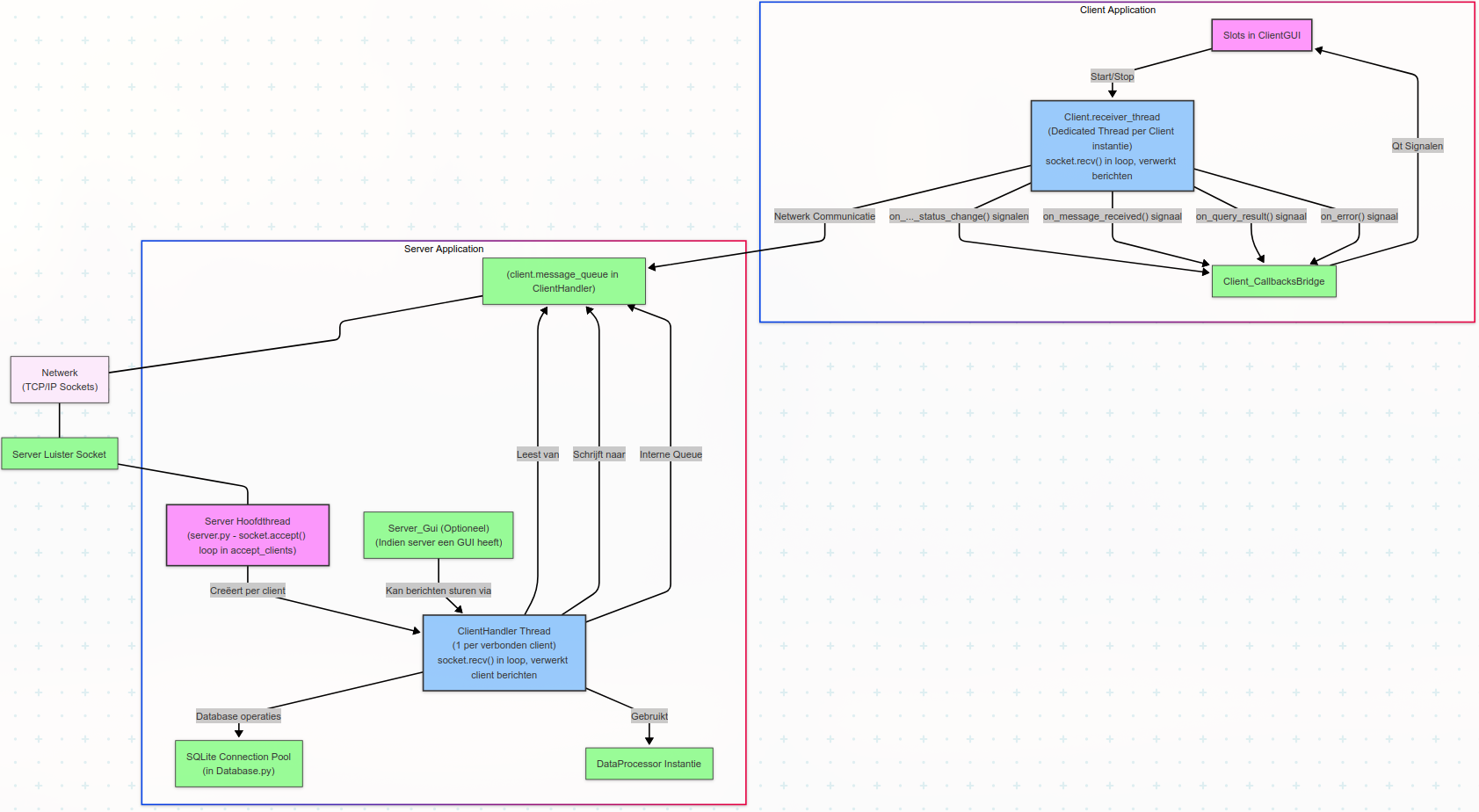
* **Parameters**: Centrumpunt, radius, start/einddatum, arrestatietype
* **Metadata**: Alle gegevens beschikbaar via dropdown/selectie
* **Weergave**: Kaart van Los Angeles met markers op hotspots

## **5. Architectuurschema – Opbouw van de Applicatie**

De applicatie is opgebouwd als een typische **client-serverarchitectuur**:

****





## **6. Vergelijking Projectopdracht vs. Gerealiseerd Project**

Alle gevraagde functionaliteiten werden met succes geïmplementeerd.

### **Serverzijde (moderator)**

* Logvenster voor belangrijke acties ✅
* Overzicht van ingelogde cliënten ✅
* Opvragen van cliëntgegevens ✅
* Per cliënt overzicht van uitgevoerde zoekopdrachten ✅
* Berichten verzenden naar cliënten ✅
* Populariteitsstatistieken van zoekopdrachten ✅
* Server afsluiten ✅

### **Cliëntzijde (gebruiker)**

* In- en uitloggen ✅
* Zoekopdrachten met parameters verzenden ✅
* Resultaten visualiseren ✅
* Meldingen van server ontvangen ✅

### **Aanvullende vereisten**

* Registratie met naam, nickname, e-mail ✅
* Online status raadpleegbaar door moderator ✅
* Gegevens van cliënten opgeslagen ✅
* Alleen server voert zoekopdrachten uit, cliënt heeft geen toegang tot data zelf ✅
* Resultaten via tabellen/grafieken ✅
* Historiek en populariteit van zoekopdrachten bijgehouden ✅
* Moderator kan berichten sturen ✅

## **7. Interessante/Moeilijke Problemen**

**Connection pool voor SQLite** Bij het gebruik van een centrale database via SQLite liep ik tegen het probleem aan dat deze standaard geen ondersteuning biedt voor meerdere gelijktijdige schrijfbewerkingen. In een client-serveromgeving waar meerdere threads of gebruikers tegelijk gegevens kunnen opvragen of opslaan, leidt dit al snel tot "database is locked" fouten. Om dit op te lossen heb ik een connection pool geïmplementeerd met een maximum van 10 actieve connecties. Deze pool zorgt ervoor dat de toegang tot de database geserialiseerd wordt, waardoor de stabiliteit van de applicatie sterk verbeterde.

**Berichtenverkeer en Queue-systeem** Tijdens het ontwikkelen van het berichtenverkeer tussen server en cliënten merkte ik dat gelijktijdige berichten soms door elkaar liepen of verkeerd werden weergegeven. Daarom heb ik een queue-systeem toegevoegd, zodat inkomende en uitgaande berichten in volgorde worden verwerkt. Daarnaast heb ik ook een fallback-mechanisme toegevoegd voor het geval de queue zou overlopen, bijvoorbeeld bij een te hoge belasting. Dit mechanisme voorkomt dat berichten verloren gaan of foutief aankomen, wat vooral belangrijk is voor meldingen van de moderator.

**Van Tkinter naar PyQt6** Oorspronkelijk was het de bedoeling om de clientinterface in Tkinter te bouwen. Echter, op mijn systeem lukte het niet om de nodige pakketten voor Tkinter correct te installeren, zelfs na meerdere pogingen en dependencies handmatig toe te voegen. Daarom ben ik overgestapt naar PyQt6, wat niet alleen stabieler bleek op mijn machine, maar me ook toeliet om een veel modernere en flexibelere GUI te bouwen. Het gebruik van layouts, event handling en styling via Qt was uiteindelijk een positieve verandering.

**Folium in combinatie met de browser voor kaartweergave** Voor de vierde query, waarbij geografische hotspots van arrestaties weergegeven worden, wilde ik een dynamische en interactieve kaart integreren. Hiervoor heb ik gebruik gemaakt van de folium-bibliotheek in combinatie met het standaard webbrowser-pakket van Python. De gegenereerde kaart wordt automatisch als een HTML-bestand weggeschreven en geopend in de standaardbrowser van de gebruiker. Deze kaarten worden lokaal opgeslagen in de temp folder van de computer. Dit wilt dus zeggen als de gebruiker zijn pc herstart dat deze ook weg zijn.

**Testen op verschillende operating softwares**Op mijn laptop draai ik een dualboot. Ik gebruik linux voor development en hou nog een windows partitie bij voor aantal games etc. Na dat mijn applicatie af was wou ik dubbelchecken als deze ook werkte op windows natuurlijk. Dus ik clone de repo dat ik online had gezet op Github. Ik merkte allemaal imports errors op, krijg functies er niet door in de gui’s. Na 2u lang proberen het probleem op te lossen, (dockerfiles, clonen van conda environment,...). Was ik vergeten de dataset toe te voegen aan mijn directory... Nog zelf mijn readme staan: “vergeet de dataset niet toetevoegen” en ja toch vergeten. Heel dom, maar ben blij dat ik het heb kunnen testen op beide operating systems!

## **8. Reflectie**

Ik ben bijzonder trots op het eindresultaat van mijn applicatie. Alle functionaliteiten zijn geïmplementeerd zoals gevraagd, en vaak zelfs uitgebreid met extra metadata of optimalisaties. Door het project aan te pakken met een duidelijk stappenplan, kon ik efficiënt en doelgericht werken. Deze gestructureerde aanpak heeft ervoor gezorgd dat ik niets over het hoofd heb gezien en het project succesvol heb kunnen afronden.