

# Visualizing private plane travel

**Kulturwissenschaften**

17.01.2026

Michael Kaup & David Kirchner

1. Motivation und Kontext
2. Vergleichbare Projekte
3. Systemarchitektur
4. Unity Anwendung
5. Interaktion
6. Herausforderungen
7. Demo
8. Ausblick + Fazit

# **1 Motivation und Kontext**

- Flugverkehr im Kontext des Klimawandels
- Diskussion: Privatpersonen vs. Privatflüge von Vermögenden
- Schwierige Datenlage und fehlende Vergleichbarkeit
  - Unterschiedliche Zeiträume und Metriken in Statistiken
  - Kein klares Bild über tatsächliche Umweltauswirkungen

- Live-Tracking und Visualisierung von Privatflugdaten
- 3D-Darstellung auf einem virtuellen Globus
- Nutzung öffentlich verfügbarer ADS-B Daten
- Fokus auf Flugzeugtypen typisch für Privatflüge
  - Ursprünglich: Tracking prominenter Personen (Musk, Bezos, Swift)
  - Anpassung: Typen wie Gulfstream, Cessna Citation

## **2 Vergleichbare Projekte**

- Automatisiertes Flight-Tracking System von Jack Sweeney
- Postet die Position von Elon Musks Privatjet auf Twitter
- Akkumuliert Flugzeit, Kosten und CO<sub>2</sub>-Verbrauch
- Kontroverse um Datenschutz vs. öffentliches Interesse

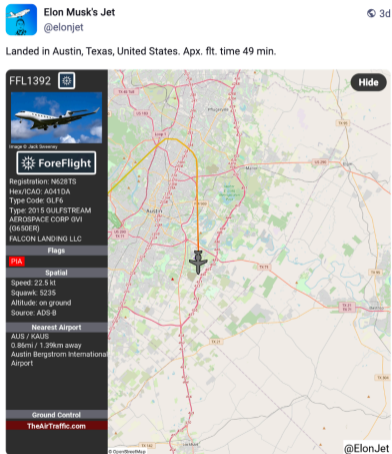


Figure 1: ElonJet Twitter Feed

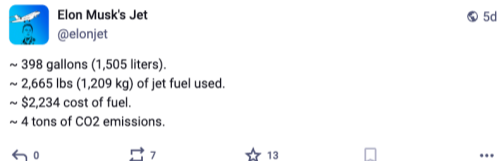


Figure 2: CO<sub>2</sub>-Verbrauch Visualisierung

- Visualisierung aller Privatjet-Flüge von Tech-Milliardären
  - Bill Gates, Jeff Bezos, Mark Zuckerberg, Elon Musk
  - Zeitraum: Juli 2021 – Juli 2022
- Datenquelle: ADS-B Exchange
- Pfeile gewichtet nach kumulativen CO<sub>2</sub>-Emissionen
- Künstlerische Kritik an Ressourcenverschwendung

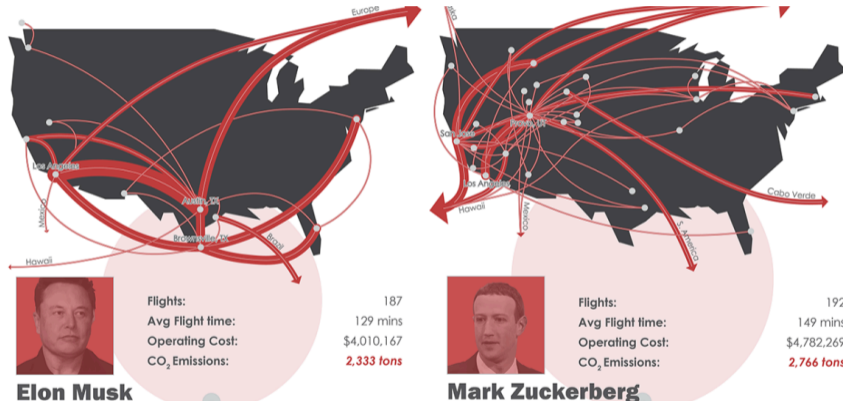


Figure 3: Architects of the Apocalypse

## **3 Systemarchitektur**

## Backend (Django + Celery)

- Regelmäßige Datenabfrage von `airplanes.live` API
- Speicherung in Postgres-Datenbank
- REST-API für Unity-Client

## Frontend (Unity)

- 3D-Globus-Visualisierung
- Interaktive Flugzeugauswahl
- Echtzeit-Datenvisualisierung

- **Django Python-Framework**

- REST-API für Flugdaten
- Django-Rest-Framework

- **Celery Beat Task Scheduler**

- Automatisierte API-Abfragen
- Alle 5 Minuten
- 1 Sekunde Pause pro  
Flugzeugtyp

- **Datenvalidierung**

- Tail-Number-Validierung
- Höhenangaben-  
Normalisierung

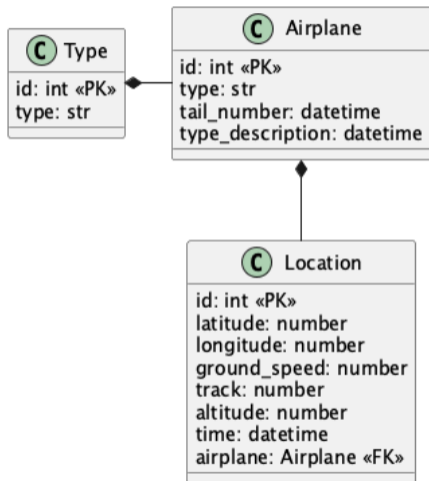


Figure 4: Relationales Datenmodell

## Django administration

### Site administration

#### API

#### Airplanes

[+ Add](#) [Change](#)

#### Locations

[+ Add](#) [Change](#)

#### Types

[+ Add](#) [Change](#)

Figure 5: Admin Übersicht

#### ZK-PHN

Type: Tail number: Type description: 

#### LOCATIONS

LATITUDE	LONGITUDE	TIME	GROUND SPEED	TRACK	ALTITUDE	DELETE?				
2025-11-27 18:37:55.862096+00:00										
<input type="text" value="-37.033195"/>	<input type="text" value="174.873872"/>	Date: 2025-11-27 <a href="#">Today</a>	<input type="text" value="-1.0"/>	<input type="text" value="-1.0"/>	<input type="text" value="-1.0"/>	<input type="checkbox"/>				
		Time: 18:37:55 <a href="#">Now</a>								
Note: You are 1 hour ahead of server time.										
2025-11-27 18:42:54.580583+00:00										
<input type="text" value="-37.027475"/>	<input type="text" value="174.878785"/>	Date: 2025-11-27 <a href="#">Today</a>	<input type="text" value="14.2"/>	<input type="text" value="-1.0"/>	<input type="text" value="-1.0"/>	<input type="checkbox"/>				
		Time: 18:42:54 <a href="#">Now</a>								
Note: You are 1 hour ahead of server time.										
2025-11-27 18:47:55.834924+00:00										
<input type="text" value="-37.224060"/>	<input type="text" value="174.877028"/>	Date: 2025-11-27 <a href="#">Today</a>	<input type="text" value="249.4"/>	<input type="text" value="165.13"/>	<input type="text" value="8975.0"/>	<input type="checkbox"/>				
		Time: 18:47:55 <a href="#">Now</a>								
Note: You are 1 hour ahead of server time.										

Figure 6: Flugzeug-Details

**Vollständiges API-Schema verfügbar unter:**

<https://flights.davidkirchner.de/swagger>

**Endpoints:**

- `/api/airplanes/` - Alle Flugzeuge
- `/api/airplanes/{tail}/` - Spezifisches Flugzeug
- `/api/airplanes/popular/` - Populäre Flugzeuge
- `/api/locations/` - Alle Positionsdaten

## 4 Unity Anwendung

## AirplaneMapper

- Verwaltung der Flugzeug-Objekte
- Marker-Lifecycle-Management
- Integration mit CoordinateFetcher

## PlayerController

- Kamera-Steuerung (WASD + Maus)
- Zwei Modi: UI-Modus und Bewegungsmodus
- Konfigurierbare Bewegungsparameter

## Hauptkomponenten (ii)

19 / 40

### ClickPlaneManager

- Flugzeugauswahl per Raycast
- Anzeige von Flugdetails im UI
- Visual Feedback (Outline)

1. **CoordinateFetcher** ruft Django-API ab
2. **JSON-Daten** werden in 3D-Koordinaten umgewandelt
  - Latitude/Longitude → 3D-Position auf Globus
  - Altitude → Radialer Offset
3. **Flugzeug-Prefabs** werden instanziiert
4. **Coroutine** spielt Route zeitkomprimiert ab
5. **Marker** werden in Intervallen gesetzt (FIFO)
6. **Rotation** wird an Globus-Normale angepasst

## **5 Interaktion**



Figure 7: Hauptmenü

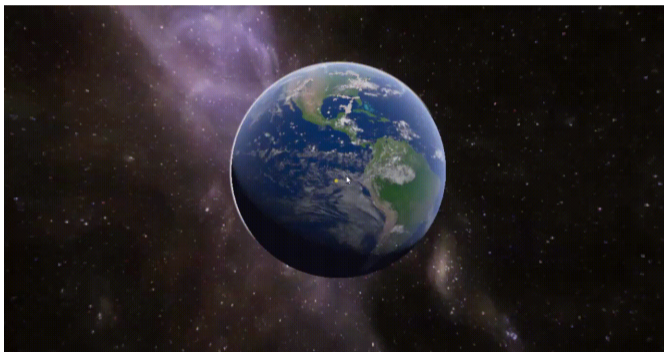


Figure 8: Kamerabewegung mit der Tastatur



Figure 9: Kamera Drehung mit der Maus

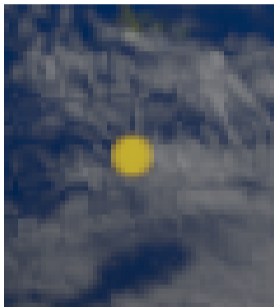


Figure 10: Cursor für die Auswahl von Flugzeugen



Figure 11: Flugzeug hervorgehoben nach Auswahl



Figure 12: HUD mit Flugzeuginformationen

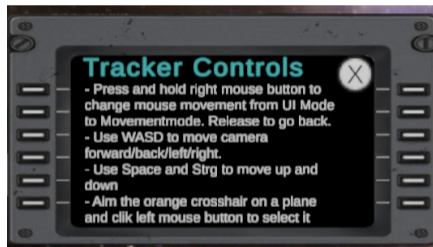


Figure 13: HUD mit Beschreibung der Controls

## **6 Herausforderungen**

## API-Ratenbegrenzung

- Künstliche Pausen (1 Sek. pro Typ)
- Moderates Abfrage-Intervall (5 Min.)
- Vermeidung von API-Sperren

## Datenbeschaffung und -bereinigung

- Kostenlose vs. kostenpflichtige Datenquellen
- Validierung der Tail-Number-Länge
- Filterung fehlerhafter Datensätze

### Zeitreiendaten

- Normalisierung der Zeitkomponente
- Alle Daten als gleichzeitig angezeigt
- Vereinfachte Analyse

## **Marker-Performance**

- Definierte Marker-Lebensdauer
- Stabile Framerate

## **Geografische Abdeckung**

- Nordamerika: Sehr hoch
- Europa: Mittel
- Asien/Globaler Süden: Gering
- Crowdsourcing-bedingte Unterschiede

## Prominente Personen

- FAA-Blockade von Registrationsdaten
- Privatjet-Owner können Daten verbergen
- Registrationsnummern nicht auffindbar

## Unsere Lösung

- Umstellung auf Flugzeugtypen
- Fokus auf typische Privatjet-Modelle
- Gulfstream, Cessna Citation, Bombardier

# Tracking-Problematik (ii)

32 / 40

## Datenverfügbarkeit

- Keine historischen Daten (kostenpflichtig)
- Eigene Datensammlung über Live-API
- `airplanes.live` als kostenlose Alternative

## 7 Demo

## 8 Ausblick

## Technische Verbesserungen

- WebSocket-Integration für Echtzeit-Streaming
- Offline-Cache und Retry-Strategien
- Verbesserte 3D-Modelle und Materialien
- UI-Overlays für Geschwindigkeit/Höhe je Marker

## Funktionale Erweiterungen

- Erweiterte Filteroptionen (Typ, Region, Zeitraum)
- Integration kommerzieller Flüge zum Vergleich

## 9 Fazit

## Erfolge

- Vollständige End-to-End-Pipeline funktionsfähig
- Robuste Backend-Architektur mit Django + Celery
- Intuitive 3D-Visualisierung in Unity
- Pragmatische Lösungen für komplexe Probleme

## Erkenntnisse

- Grenzen öffentlicher Flugdatenverfügbarkeit
- Regulatorische Hürden (FAA-Blockade)

## Projektergebnis (ii)

38 / 40

- Performance-Optimierung bei großen Datenmengen
- Wert von flexiblem Umdenken bei Hindernissen

**10 Vielen Dank für Ihre  
Aufmerksamkeit!**

**Fragen?**

API: <https://flights.davidkirchner.de/swagger>

Michael Kaup | David Kirchner