Dokumentation:  
Projekt OpenSky SouthTyrol

Inhaltsverzeichnis

[**Anforderungsanalyse** 1](#_Toc130129130)

[**Architektur der Anwendung** 3](#_Toc130129131)

[**Frontend** 3](#_Toc130129132)

[**Backend** 5](#_Toc130129133)

[**Hosting** 5](#_Toc130129134)

[**Links zum Source-Code** 5](#_Toc130129135)

# **Anforderungsanalyse**

1. **Sich selbst bzw. eigene Firma vorstellen**OpenSky SouthTyrol ist eine Web-Anwendung, welche den Usern ermöglicht, sich kostenlos über den Flugverkehr in Südtirol zu informieren. Dabei können die User verschiedenste Informationen aus geplanten und gerade stattfindenden Flügen einholen.
2. **Was ist meine Zielgruppe?**  
   Zur Zielgruppe gehören alle Menschen, welche sich für den Flugverkehr in und um Südtirol interessieren. Dabei spricht die Anwendung keine spezifische Altersgruppe an. Auch muss der Benutzer kein Experte sein. Damit wird eine breite Zielgruppe angesprochen. Die Webapp wurde für Desktop, als auch für Mobile-Geräte konzipiert und ist somit überall einwandfrei verwendbar (responsive Design).
3. **Welcher Inhalt soll präsentiert werden? (Art der Website)**

Die Website soll informativen Inhalt präsentieren.

1. **Wie ist die Struktur der Website?**

Die Website kann grundsätzlich in zwei Teile gegliedert werden. Der erste Teil hat ausschließlich die Aufgabe den User über das Produkt zu informieren und ihn davon zu begeistern. Hier werden eine Feature-Liste, ein Impressum und eine Registrierungs- bzw. Login-Funktion realisiert. Der zweite Teil der Website implementiert die eigentlichen Funktionalitäten der Anwendung. Nach einem erfolgreichen Login kann der User, innerhalb des geschützten Kundenbereichs, alle Features nutzen und Einstellungen an seinem Account vornehmen.

1. **Funktionalität der Website (Use Cases)**

* Ein User kann sich auf der Homepage über das Tool informieren
  + Eine Seite präsentiert das Produkt und die darin enthaltenen Features
  + Ein Impressum zeigt relevante Informationen über die Websitebetreiber und die Datenverarbeitung an
* Ein User kann ein neues Nutzerkonto anlegen (registrieren)
  + Username, E-Mail und Passwort müssen angegeben werden
* Ein User kann sich mit seinem Nutzerkonto authentifizieren (anmelden)
  + Username und Passwort müssen angegeben werden
* Ein eingeloggter User kann sein Passwort abändern
  + Das alte und das neue Passwort müssen angegeben werden
* Ein eingeloggter User kann sich über geplante Flüge ab bzw. zum Flughafen Bozen informieren
  + Hier kann der User auch auf bereits stattgefundene Flüge zurückgreifen
  + Durch das Anklicken eines Links, wird der User zur Skyalps-Webseite weitergeleitet, wo weitere Informationen zum geplanten Flug vorliegen
* Ein eingeloggter User kann auf einer Karte in Echtzeit Flugzeuge beobachten, welche gerade über Südtirol fliegen
  + Ein Tooltip liefert beim hovern weitere Informationen zum Flug
  + Ein Click auf das Icon öffnet eine Detail-Ansicht zum jeweiligen Flug mit weiterführenden Informationen
* Ein eingeloggter User kann in einer Tabelle in Echtzeit Flugzeuge beobachten, welche in jenem Moment über Südtirol fliegen
  + Durch das Anklicken des Links auf der Flug-Nummer, wird der User zur Detailansicht weitergeleitet, welche weiterführende Informationen zum Flug enthält.
* Ein eingeloggter User kann sich ausloggen

1. **Soll die Website für versch. Sprachgruppen zugänglich sein?**  
   Die Website soll auf einfach gehaltenem Englisch angezeigt werden, um eine große Zielgruppe ansprechen zu können. In der Anwendung wird grundsätzlich sehr wenig Text dargestellt, meist sind nur klare Zahlen und Fakten zu lesen. Eine Ausnahme ist hierbei das Impressum, wo ein recht komplexer Text zu finden ist, der allerdings auch rechtliche Relevanz hat. Deshalb haben wir uns dazu entschieden diesen Teil auch in deutscher Sprache anzubieten. Es sind keine weiteren Übersetzungen für spezifische Sprachgruppen geplant, da der Mehraufwand vermutlich nur zu einer vernachlässigbaren Zielgruppenerweiterung führen würde.
2. **Wie wird die Aktualität der Website gewährleistet?**

Die Web-Anwendung holt sich all ihre Daten von verschiedenen APIs. Somit wird die Aktualität automatisch gewährleistet.

1. **Wie wird die Website veröffentlicht?**

Das Frontend wird mithilfe eines Docker-Containers gehostet. Im Container sorgt der nginx-Webserver dafür, dass die Website auf dem Port 443, für alle über https erreichbar ist und ordnungsgemäß angezeigt wird. Das Python-Backend wird wiederum in einem eigenen Docker-Container gehostet.

1. **Gibt es Konkurrenz bzw. vergleichbare Websites?**

Es gibt einige Webseiten, welche Informationen über Flüge in der ganzen Welt anbieten. Allerdings sind diese Anwendungen meist teilweise oder gar vollständig kostenpflichtig. Zudem konzentriert sich unsere Anwendung auf den Flugverkehr in Südtirol und führt speziell für den Bereich verschiedenste Datenquellen zusammen. Zusammenfassend kann man also sagen, dass die Webapp einen „unique-selling-point“ aufweist.

1. **Budget und Deadline**

Um die Website zu hosten, wurden 2,19€ für das Mieten eines Servers ausgegeben, wobei wir uns den Server mit der Gruppe von Hell und Jäger geteilt haben. Deadline des Projekts ist Mittwoch, der 22.03.2023.

# **Architektur der Anwendung**

Die Anwendung kann grundsätzlich in 3 Schichten gegliedert werden:

1. Das Frontend: implementiert die Anwendungs- bzw. Darstellungsschicht. Hier erfolgt die Darstellung der Daten aus den APIs und die Interaktion mit dem User.
2. Das Python-Backend: stellt dem Frontend Methoden zur Authentifizierung und Benutzerverwaltung an.
3. Daten-APIs: die verschiedenen APIs liefern dem Frontend Daten, welche dort visualisiert werden. Dabei wird konkret auf folgende APIs zugegriffen:

* Über eine REST-API des OpenDataHubs wird auf die Daten von SkyAlps (Betreiber des Bozner Flughafens) zugegriffen, um Informationen über geplante oder bereits durchgeführte Flüge zum bzw. ab Flughafen Bozen zu bekommen
* Über ein Websocket des OpenDataHubs werden der Anwendung Live-Daten zu Flugzeugen geliefert, welche sich gerade über Südtirol befinden
* Zusätzlich dazu werden 3 REST-APIs vom Dienst AirLabs konsultiert, um weiterführende Informationen zum Flug, zu den Flughäfen und Airlines anhand der Flugnummer anzeigen zu können.
* Um für die Detail-Ansicht eines Fluges ein passendes Bild zum jeweiligen Flugzeug anzeigen zu können, wird die API der Suchmaschine Qwant angefragt, welche uns für einen Suchbegriff mehrere Bilder zurückliefert und wird dann das erste Ergebnis dem User anzeigen.
* Schließlich greift die Anwendung auch auf den Karten-Service von openstreetmap.org zu, mit dessen Hilfe wir die Flugdaten passend visualisieren können.

# **Frontend**

Um das Frontend der Website zu realisieren, haben wir uns für das Framework Angular entschieden. Das Framework ermöglicht modernes Web-Development mit Typescript und kümmert sich zugleich um die Browserkompatibilität und um die Effizienz der Website. Wir haben uns hierbei für die neueste Angular Version 15.1.2 mit Node.js 18.12.1 und dem Node Package Manager (npm) 9.3.1 entschieden. Nachfolgend betrachten wir zwei besonders interessante Aspekte des realisierten Frontends.

**Das Authentifizierungs-System**  
Es gibt eine Großzahl an Angular-Frameworks, welche ein Authentifizierungssystem realisieren. Allerdings haben wir uns dazu entschieden die Authentifizierungsmanagement selbst zu implementieren.  
Wenn der User auf den Registrieren-Button klickt, werden die Eingaben validiert und diese an den AuthenticationService weitergegeben. Der Service erstellt mit diesen Daten ein API-Call. War das Registrieren nicht erfolgreich (also Responsecode != 200), so wird der Backend-Fehler vom error.interceptor.ts abgefangen, gehandelt oder an den Benutzer direkt ausgegeben. Bei einer erfolgreichen Registrierung ruft das SignUp-Component mit denselben Daten die login()-Funktion des Authentication-Service auf. Der Service erstellt mit diesen Daten ein API-Call auf dem Login-Endpoint. War diese Operation erfolgreich, so wird aus der Backend-Response ein User-Objekt zusammengebaut und im lokalen Speicher des Benutzers abgelegt. Damit kann jederzeit ohne Cookies verifiziert werden, dass der User korrekt eingeloggt ist. Bei einem potenziellen Logout wird dieses User-Objekt aus dem lokalen Speicher wieder entfernt. Nach dem erfolgreichen Login navigiert das SignUp-Component automatisch zur Hauptseite des geschützten Kundenbereichs.  
Damit ein nicht eingeloggter User nicht auf die geschützten Seiten zugreifen kann, wurde der auth.guard.ts implementiert. Der AuthGuard wird vom app-routing-module verwendet, um zu überprüfen, ob der aktuelle User die gewünschte Seite aufrufen darf oder nicht. Dabei wird das CanActive-Interface implementiert, um bestimmte Routen für den User zu blockieren oder zu aktivieren. Der AuthGuard überprüft über den AuthenticationService, ob der User eingeloggt ist oder nicht. Wenn ja, so wird der User auf die gewünschte Route weitergeleitet, falls nein, so findet sich der User auf der Login-Seite wieder. Im letzten Fall wird die angeforderte Route zwischengespeichert, sodass diese nach einem erfolgreichen Login automatisch aufgerufen werden kann.  
Die Passwörter werden vom Frontend nicht gehasht, da die Daten über https automatisch mit SSL verschlüsselt werden. Das Backend speichert dann die Passwörter in der Datenbank in gehashter Form, um maximale Sicherheit zu gewährleisten.

**Live-Karte mit Leaflet**Für die Darstellung von Daten auf Karten, haben wir das Karten-Framework Leaflet verwendet. Leaflet ist eine Open-Source-Bibliothek für interaktive Karten in HTML5. Leaflet bietet dabei eine einfache Schnittstelle zum Erstellen von Karten mit benutzerdefinierten Markern, Popups, Interaktionsmöglichkeiten wie z.B. Zoom, sowie Zugriff auf verschiedene Kartenanbieter wie OpenStreetMap, Mapbox oder Google Maps an. Wir nutzen den Karten-Anbieter OpenStreetMap, da man auf dessen API recht schnell und ohne API-Key zugreifen kann.  
Zunächst wird die Leaflet-Karte in der ngOnInit()-Methode der MapComponent-Klasse initialisiert. Eine Karte wird erstellt und ein Kartenlayer hinzugefügt. Danach wird eine Verbindung zum WebSocket-Server hergestellt, um Informationen über Flugzeuge zu empfangen, die auf der Karte angezeigt werden sollen.

Wenn eine WebSocket-Nachricht empfangen wird, wird überprüft, ob das empfangene Flugzeug bereits auf der Karte vorhanden ist oder nicht. Wenn es bereits vorhanden ist, werden seine Koordinaten aktualisiert und der dazugehörige Marker auf der Karte ebenfalls aktualisiert. Andernfalls wird ein neues Flugzeug zur planes-Liste hinzugefügt und ein neuer Marker auf der Karte erstellt.  
Um einen Marker zu erstellen, wird ein neues L.marker()-Objekt mit den Koordinaten des Flugzeugs erstellt und dann mit der Leaflet-Karte this.map verbunden. Der icon-Parameter des L.marker()-Objekts wird auf ein neues L.icon()-Objekt gesetzt, welches das Bild des Flugzeugsymbols enthält. Das rotationAngle-Attribut wird ebenfalls gesetzt, um die Ausrichtung des Symbols entsprechend dem Kurs des Flugzeugs zu drehen.  
Um den Marker auf der Karte anzuzeigen, wird er mit der Methode addTo(this.map) hinzugefügt. Ein Tooltipp wird ebenfalls hinzugefügt, um zusätzliche Informationen wie Flugzeugname, Geschwindigkeit und Höhe anzuzeigen.  
Auf ähnlicher Weise, wurde die Karte der Flugroute in der Flug-Detail-Ansicht implementiert.

# **Backend**

# **Hosting**

# **Links zum Source-Code**

**Frontend**<https://github.com/tfobz-informatik/opendatahub-kofler-morandell/>

**Backend**  
<https://github.com/maxkofler/opensky-st-api>