**Erstellung eines webbasierten, interaktiven taxiway-instruction generators**

vorgelegt von

**Valentin ADlgasser**

**Projektarbeit**

zur Erlangung der Berufsreifeprüfung

im Fachbereich

**Informationsmanagement und Medientechnik**

BFI-Salzburg

Salzburg, am 13. Mai 2025

vorgelegt bei

Mag. Wolfgang Lehner

Eidesstattliche Erklärung

*Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Die vorliegende Arbeit hat in dieser oder einer ähnlichen Form noch keiner anderen Prüfungsbehörde vorgelegen.*

*Salzburg, 13. Mai 2025 Unterschrift*

*Valentin ADLGASSER*

Kurzbeschreibung

Im Rahmen dieser Projektarbeit wurde eine interaktive Webanwendung entwickelt, die das Generieren von Taxi-Instruktionen für alle sechs zivilen Flughäfen Österreichs ermöglicht. Ziel war es, ein benutzerfreundliches Tool zu schaffen, das sowohl zur Schulung als auch zur Veranschaulichung der Abläufe am Boden eines Flughafens dient.

Die Webanwendung wurde mit den Programmiersprachen HTML, CSS und JavaScript umgesetzt. Für die Darstellung der Flughafenlayouts und die Platzierung der Marker wurde die JavaScript-Bibliothek Leaflet.js verwendet. Die Anwendung ist responsiv gestaltet und kann sowohl auf Desktopgeräten als auch auf mobilen Endgeräten verwendet werden.

Nutzerinnen und Nutzer haben die Möglichkeit, auf einer interaktiven Karte einzelne Flughafenbereiche wie Abstellpositionen, Start- und Landebahnen, Exits sowie Holdingpoints auszuwählen. Auf Basis der ausgewählten Elemente generiert das System automatisch eine standardisierte Taxi-Instruktion in Textform. Diese Funktion kann insbesondere für Ausbildungszwecke, etwa im Rahmen von Flugfunk- oder Lotsenausbildungen, von Nutzen sein.

Die Projektarbeit verbindet theoretisches Wissen mit praktischer Umsetzung. Im Verlauf des Projekts konnte ich meine Kenntnisse in der Webentwicklung deutlich erweitern. Insbesondere im Umgang mit geografischen Datenformaten wie GeoJSON, die Arbeit mit interaktiven Karten sowie die Umsetzung von responsivem Design waren zentrale Lernbereiche. Die Anwendung ist modular aufgebaut und kann bei Bedarf um weitere Flughäfen und Daten ergänzt werden.

Inhaltsverzeichnis

[Eidesstattliche Erklärung 2](#_Toc198061720)

[Kurzbeschreibung 3](#_Toc198061721)

[Inhaltsverzeichnis 4](#_Toc198061722)

[1 Einleitung 5](#_Toc198061723)

[2 Die Planungsphase 6](#_Toc198061724)

[2.1 Zieldefinition 6](#_Toc198061725)

[2.2 Grobes Designkonzept 7](#_Toc198061726)

[3 Technische Umsetzung 9](#_Toc198061727)

[3.1 Entwicklungsumgebung: Visual Studio Code 9](#_Toc198061728)

[3.2 Strukturierung mit HTML 10](#_Toc198061729)

[3.3 Interaktivität mit JavaScript 10](#_Toc198061730)

[3.4 Gestaltung mit CSS 11](#_Toc198061731)

[4 Aufbau der Seiten 13](#_Toc198061732)

[4.1 Navigationsleiste 13](#_Toc198061733)

[4.2 Startseite (Generatorseite) 14](#_Toc198061734)

[4.3 Informationsseite 15](#_Toc198061735)

[4.4 Kontaktseite 16](#_Toc198061736)

[4.5 Impressum 17](#_Toc198061737)

[5 Sonstige verwendete Technologien und Programme 18](#_Toc198061738)

[5.1 Leaflet.js – Interaktive Kartendarstellung 18](#_Toc198061739)

[5.2 GIMP – Bildbearbeitung 18](#_Toc198061740)

[5.3 Moongoose.exe – Lokaler Webserver 19](#_Toc198061741)

[6 Herausforderungen & gewonnene Erkenntnisse 20](#_Toc198061742)

[Abbildungsverzeichnis 22](#_Toc198061743)

[Literarturverzeichnis 23](#_Toc198061744)

[Projektantrag 24](#_Toc198061745)

1. Einleitung

Für mein Abschlussprojekt im Rahmen der Berufsreifeprüfung habe ich mich entschieden, ein Thema aus meinem beruflichen Umfeld aufzugreifen. Ich bin als Apron Controller am Flughafen Salzburg tätig und beschäftige mich täglich mit dem sicheren und effizienten Ablauf von Flugbewegungen am Boden. Dabei sind Taxi-Instruktionen ein zentraler Bestandteil meiner Arbeit. Durch meine tägliche Praxis sowie den Erwerb des dafür notwendigen Funksprechzeugnisses entstand die Idee, ein technisches Hilfsmittel zu entwickeln, das diesen Aspekt der Luftfahrt digital unterstützt.

Ziel der Anwendung ist es, eine digitale Unterstützung bereitzustellen, die den Aufbau und die Struktur von Flughäfen visuell darstellt und gleichzeitig automatisierte Taxi-Instruktionen erstellt. Diese Kombination aus interaktiver Karte und textbasierter Ausgabe soll das Verständnis für Abläufe am Boden fördern und vor allem Einsteigerinnen und Einsteigern im Bereich der Luftfahrt eine Hilfestellung bieten.

Durch die Umsetzung dieses Tools konnte ich meine technischen Fähigkeiten in der Webentwicklung erweitern und gleichzeitig ein Thema aufgreifen, das mich sowohl beruflich als auch persönlich interessiert. Die Anwendung verbindet meine Erfahrung aus der Luftfahrpraxis mit modernen Technologien und biete praxisorientierte Lösungen für einen konkreten Anwendungsfall.

1. Die Planungsphase

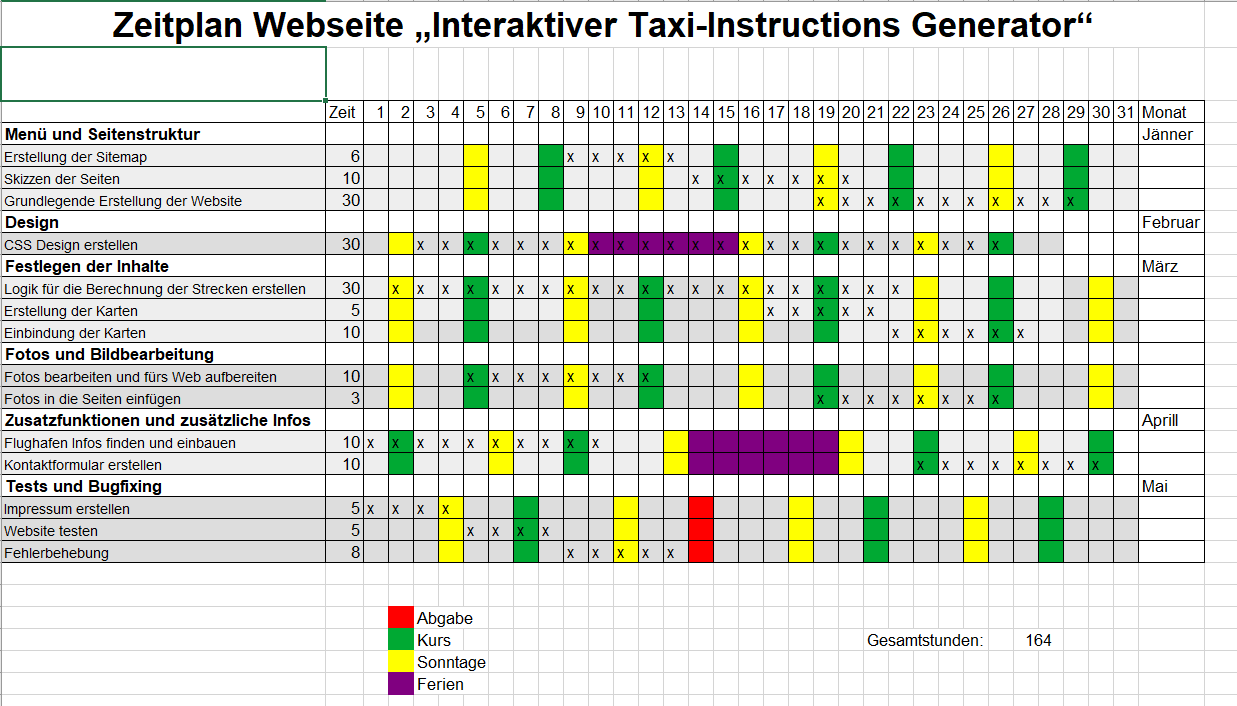
Zu Beginn der Projektarbeit wurde ein detaillierter Zeitplan erstellt, um den Arbeitsaufwand übersichtlich zu strukturieren und die einzelnen Entwicklungsschritte zeitlich realistisch einzuordnen. Dieser Zeitplan wurde in Form einer Excel-Datei angelegt und in mehrere Phasen unterteilt.

Abbildung 1: Zeitplan

* 1. Zieldefinition

Das Hauptziel bestand darin, eine Webanwendung zu entwickeln, die für alle sechs zivilen Flughäfen Österreichs Taxi-Instruktionen auf Basis visueller Kartendaten generiert. Die Anwendung sollte sowohl auf Desktopgeräten als auch auf mobilen Endgeräten problemlos nutzbar sein. Einfache Bedienung, klare Struktur und visuelles Feedback standen im Vordergrund. Zusätzlich sollte es einfach sein, weiter Flughäfen in Zukunft hinzuzufügen. Außerdem sollten Infos zu jedem Flughafen vorhanden sein.

* 1. Grobes Designkonzept

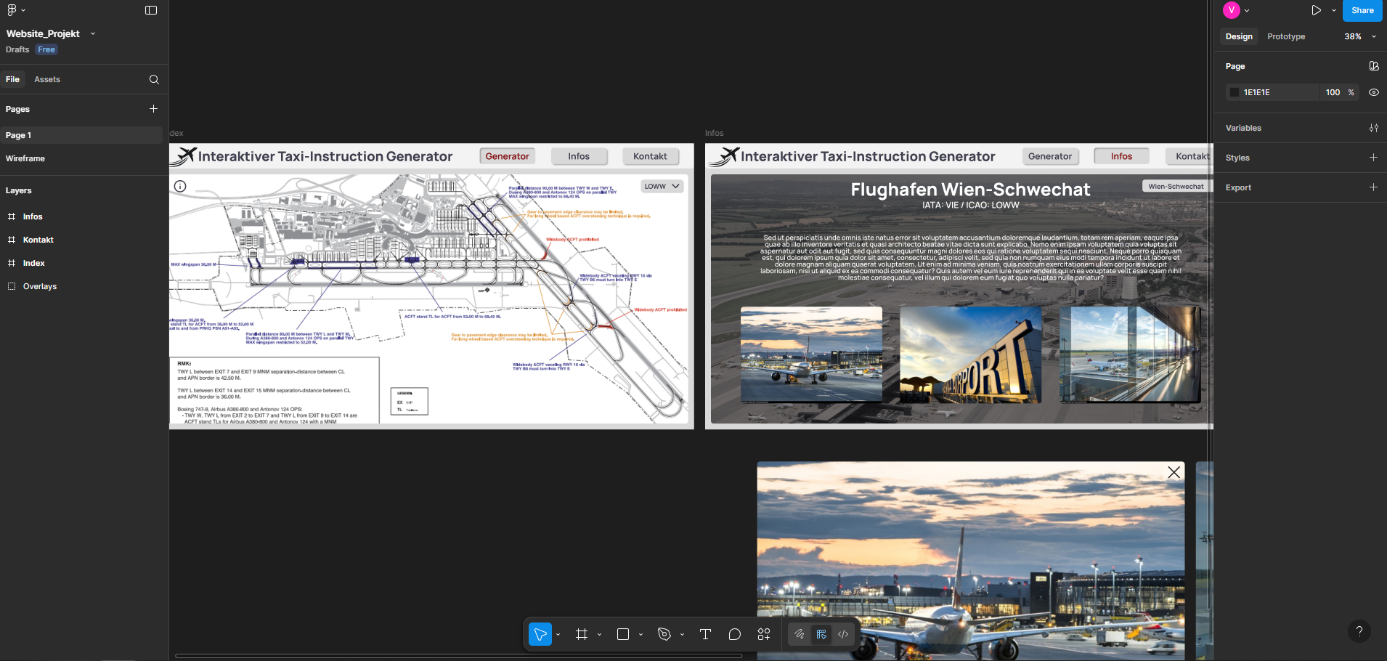
Für die Erstellung des ersten Designkonzepts wurde das Online-Tool „Figma“ verwendet. Dieses Werkzeug bietet die Möglichkeit, Benutzeroberflächen webbasiert zu entwerfen und darzustellen. Ein wesentlicher Vorteil von Figma liegt darin, dass es kostenlos nutzbar ist und keine lokale Installation erfordert. Zudem war mir das Tool aus früheren Anwendungen bereits bekannt, was den Einstieg in die Designarbeit erleichterte. 

Abbildung 2: Figma

Im Rahmen der Designphase wurden drei Hauptseiten konzipiert:

* Die Startseite (oder Generatorseite), auf der ein direkter Einstieg in die Kartenfunktion angeboten wird,

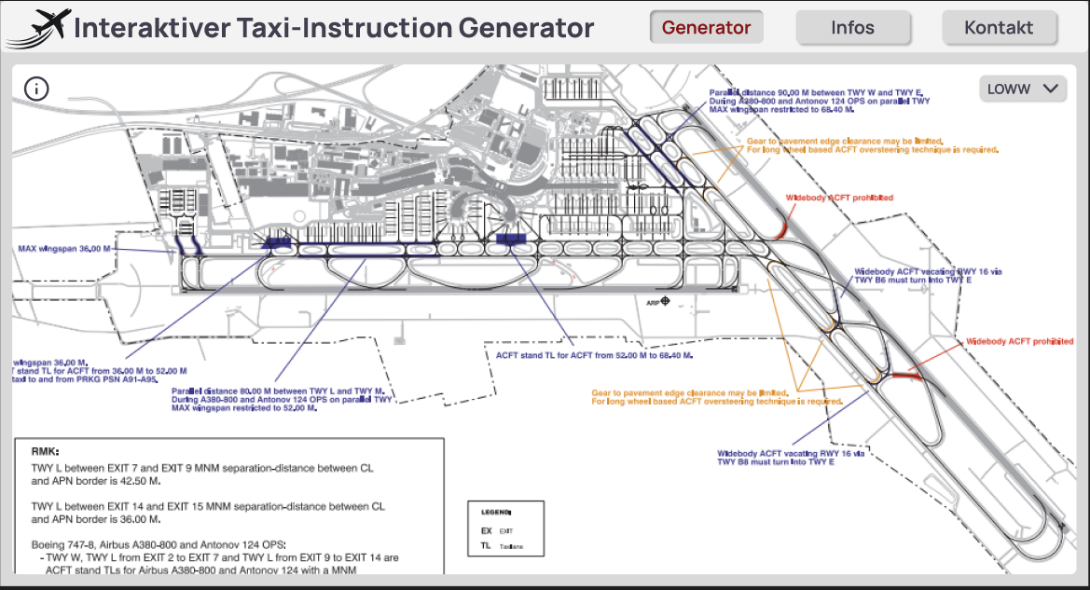


Abbildung 3: Startseite

* Die Informationsseite, auf der für jeden Flughafen spezifische Inhalte und Fotos vorbereitet werden, 

Abbildung 4: Infoseite

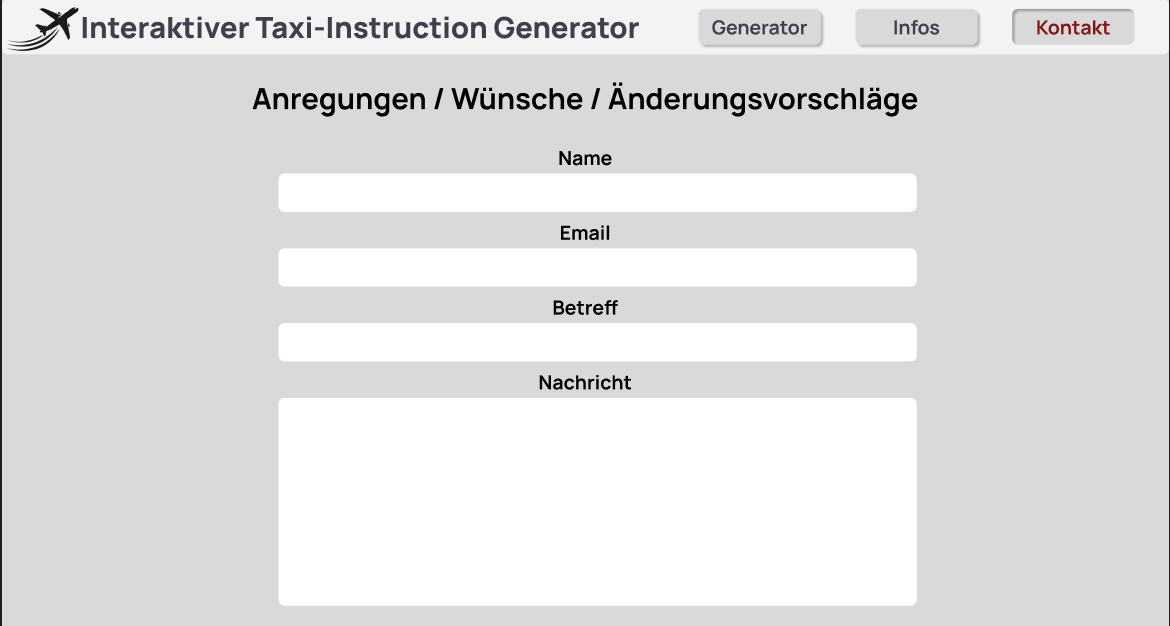
* Sowie die Kontaktseite, die ein einfaches Kontaktformular zur Verfügung stellt. 

Abbildung 5: Kontaktseite

Diese Entwürfe orientierten sich von Beginn an am geplanten Funktionsumfang und gaben einen ersten Eindruck vom zukünftigen Erscheinungsbild der Webanwendung. Die Designs wurden im weiteren Projektverlauf mehrfach angepasst und weiterentwickelt. Änderungen ergaben sich insbesondere aus technischen Umsetzungsfragen und aus der laufenden Verbesserung der Benutzerfreundlichkeit.

Die Indexseite wurde nicht extra designed, da sich dessen Aufbau an der Kontaktseite orientieren sollte.

1. Technische Umsetzung

Nach Abschluss der Planungsphase wurde mit der praktischen Umsetzung der Webanwendung begonnen. Die Technische Umsetzung erfolgte auf Basis von HTML, CSS und Javascript. Darüber hinaus kam die externe Bibliothek Leaflet.js zur Kartendarstellung sowie eigenen JavaScript-Dateien für die Datenstruktur der Flughäfen zum Einsatz.

* 1. Entwicklungsumgebung: Visual Studio Code

Für die Umsetzung des Projekts wurde der Quellcode-Editor „Visual Studio Code“ (VS Code) verwendet. Dieses plattformübergreifende Tool wird von Microsoft entwickelt und bietet umfangreiche Funktionen wie Syntaxhervorhebung, intelligente Codevervollständigung (IntelliSense), integrierte Git-Unterstützung und eine Vielzahl von Erweiterungen. VS Code ist kostenlos verfügbar und eignet sich sowohl für Einsteiger als auch für erfahrene Entwickler.

„Visual Studio Code ist ein einfacher, aber leistungsstarker Quellcode-Editor, der auf Ihrem Desktop ausgeführt wird und für Windows, macOS und Linux verfügbar ist. Die Lösung bietet integrierte Unterstützung für JavaScript, TypeScript und Node.js und verfügt über ein umfangreiches Ökosystem von Erweiterungen für andere Sprachen und Runtimes (z. B. C++, C#, Java, Python, PHP, Go, .NET).“   
(„Visualstudio“, 2025)

Die Entscheidung für VS Code fiel aufgrund seiner Benutzerfreundlichkeit, der umfangreichen Erweiterungsmöglichkeiten und der bereits vorhandenen Erfahrung im Umgang mit diesem Editor

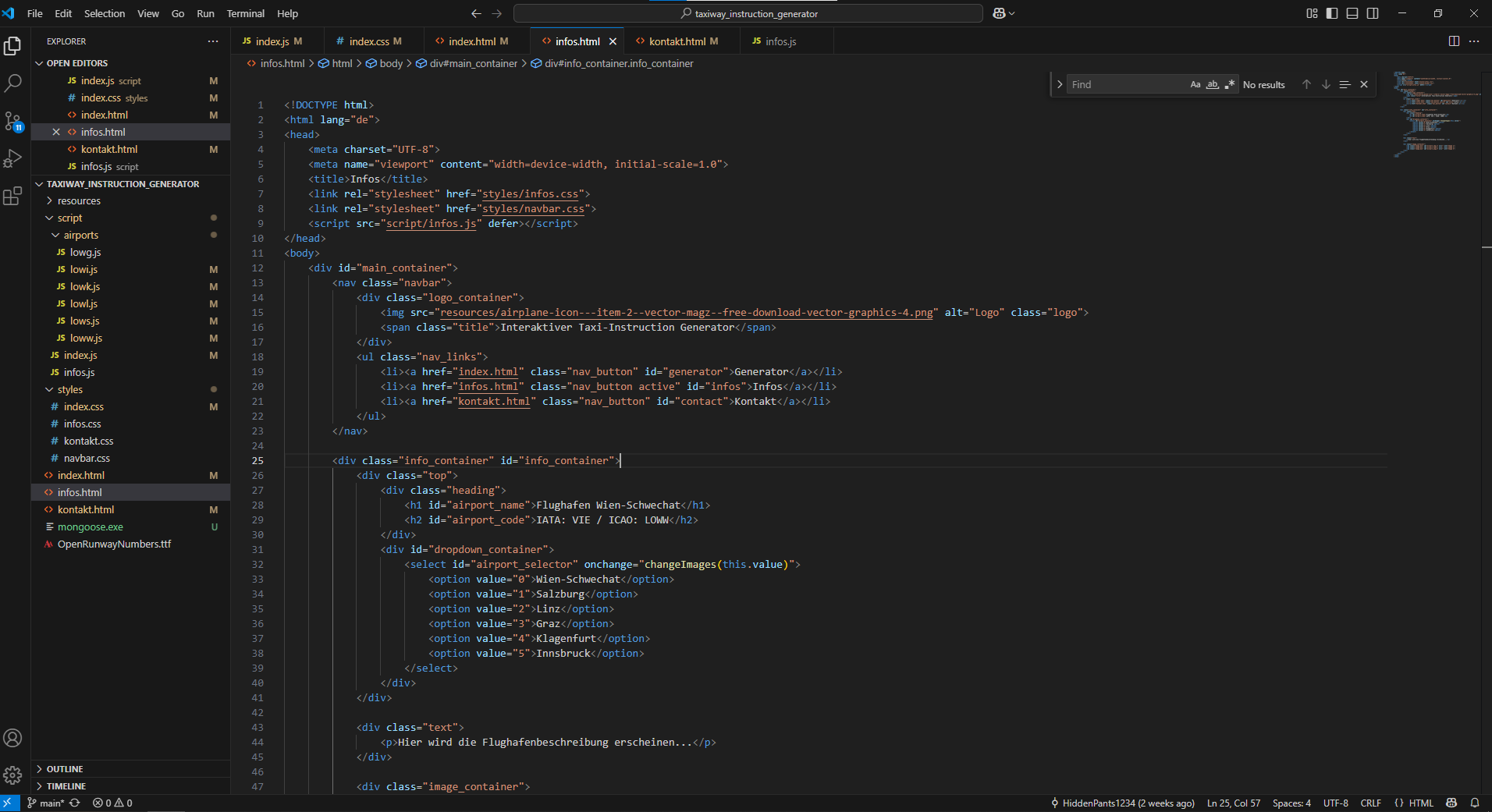


Abbildung 6: Visual Studio

* 1. Strukturierung mit HTML

Die Hypertext Markup Language (HTML, englisch für Hypertext-Auszeichnungssprache) ist eine textbasierte Auszeichnungssprache zur Strukturierung elektronischer Dokumente wie Texte mit Hyperlinks, Bildern und anderen Inhalten.  
(„Wiki/Hypertext Markup Language“, 2025)

HTML bildet also die Grundlage jeder Webseite – so auch bei diesem Projekt. Es wurde verwendet, um die Seitenstruktur der Anwendung festzulegen.

Die HTML-Dateien definieren dabei alle sichtbaren Elemente wie Überschriften, Textabschnitte, Auswahlfelder, Buttons und Container für die Kartenansicht. Eine klare und semantische Struktur war wichtig, um die Anwendung übersichtlich, logisch aufgebaut und technisch sauber zu gestalten.

Beispielhaft sieht ein Teil der Startseite folgendermaßen aus:

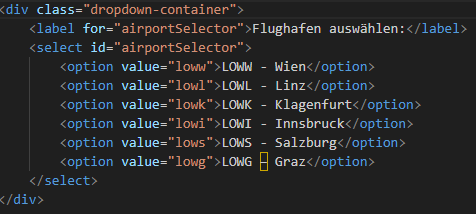


Abbildung 7: Dropdown

Diese Dropdown-Auswahl wird später über JavaScript mit Funktionalität verbunden.

Durch die saubere Trennung von Inhalt (HTML), Aussehen (CSS) und Verhalten (JavaScript) entstand ein modularer Aufbau, der das Projekt gut wartbar und erweiterbar macht.

* 1. Interaktivität mit JavaScript

JavaScript ist eine Programmiersprache, die Entwickler verwenden, um interaktive Webseiten zu erstellen. Von der Aktualisierung von Social Media Feeds bis hin zur Anzeige von Animationen und interaktiven Karten können JavaScript-Funktionen die Benutzerfreundlichkeit einer Website verbessern.  
(„What is/Javascript“, 2025)

JavaScript ist die zentralste Programmiersprache in diesem Projekt, da sie sämtliche interaktive Funktionen steuert – insbesondere auf der Generator-Seite. Dort ermöglicht JavaScript das Laden der Kartendaten, das Platzieren der Marker, das Erfassen von Benutzereingaben und das automatische Generieren einer standardisierten Taxi-Instruktion.

Ein Auszug aus der Logik zur Marker Interaktion: 

Abbildung 8: JavaScript

Durch diese dynamische Steuerung entsteht ein geführter Ablauf, bei dem Nutzende Schritt für Schritt Rollwegpositionen auswählen und am Ende eine automatisch generierte Anweisung erhalten, z.B.:  
*„Taxi from Main Apron via Exit 12 and Holdingpoint A6 to Runway 29“*

* 1. Gestaltung mit CSS

Cascading Style Sheets (CSS) ist eine Programmiersprache, die es Ihnen ermöglicht, das Design von elektronischen Dokumenten zu bestimmen. Anhand einfacher Anweisungen – dargestellt in übersichtlichen Quellcodes – lassen sich so Webseiten-Elemente wie Layout, Farbe und Typografie nach Belieben anpassen.  
(„Websites/Webdesign/Was ist CSS“, 2025)

CSS wurde eingesetzt, um der Webanwendung ein einheitliches Erscheinungsbild zu verleihen. Ziel war eine klare, schlichte Gestaltung mit intuitiver Bedienbarkeit – auch auf mobilen Endgeräten. Dies wurde unter anderem durch Flexbox-Layouts und Media Queries erreicht.

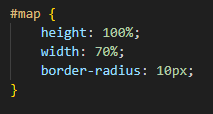
Ein Beispiel aus dem Styling der interaktiven Kartenelemente: 

Abbildung 9: CSS

Diese Anweisung sorgt dafür, dass die Karte auf Desktopgeräten großflächig angezeigt wird, mit abgerundeten Ecken für ein modernes Design. Auch die Navigationsleiste, Buttons und Formulare wurden optisch mit CSS angepasst.

Dank CSS konnte die Anwendung responsiv gestaltet werden, was bedeutet, dass sie sowohl auf großen Bildschirmen als auch auf Smartphones problemlos nutzbar ist.

1. Aufbau der Seiten

Die Webanwendung besteht aus mehreren eigenständigen HTML-Seiten, die über eine gemeinsame Navigationsleiste verbunden sind. Jede Seite erfüllt eine bestimmte Funktion im Gesamtsystem. Der Quellcode wurde so aufgebaut, dass eine klare Trennung von Inhalt, Design und Funktionalität gegeben ist.

* 1. Navigationsleiste

Die Navigationsleiste befindet sich am oberen Rand jeder Seite und ermöglicht den Wechsel zwischen „Generator“, „Info“ und „Kontakt“. Sie ist in einer eigenen CSS-Datei gestaltet und in allen HTML-Seiten eingebunden.

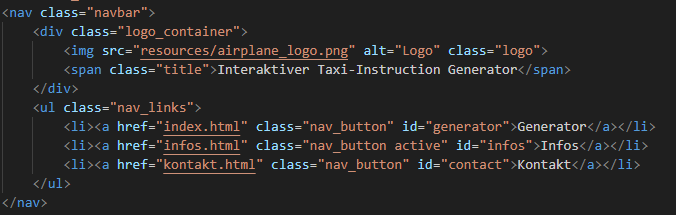
Im HTML wird die Navbar als <nav>-Element mit einer Logo-Sektion und einem Navigationsmenü umgesetzt: 

Abbildung 10: Navbar

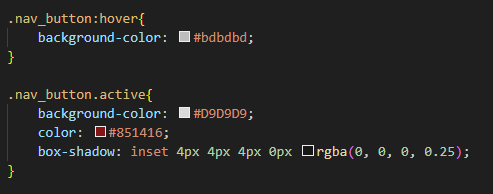
Das zugehörige CSS legt Farben, Abstände und Hover-Effekte fest. Über die Klasse „.active“ wir hervorgehoben, auf welcher Seite man sich gerade befindet. 

Abbildung 11: Navbar CSS

* 1. Startseite (Generatorseite)

Die Startseite enthält die zentrale Funktionalität der Anwendung – die interaktive Kartendarstellung mit Taxi-Instruktions-Generator. Der HTML-Aufbau besteht aus einem Dropdown zur Flughafenauswahl, einem Bereich zur Anzeige der gewählten Elemente sowie dem Kartencontainer. 

Abbildung 12: Startseite HTML

Wichtige JavaScript-Funktionen im Hintergrund:

* Laden des korrekten Kartenausschnitts und Bildes je nach gewähltem Flughafen.
* Platzieren interaktiver Marker für Positions-, Exit-, Holding- und Runwaypunkte.
* Schrittweises Durchführen der Auswahl und automatisches Erzeugen der Instruktion

Der folgende Codeausschnitt initialisiert eine Leaflet-Karte mit einem festgelegten Zoom- und Koordinatensystem.

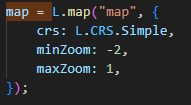


Abbildung 13: Startseite JS

Das CSS kümmert sich um das Layout, Flex-Verteilung zwischen Seitenbereich und Karte sowie das Popup-Design für die generierten Instruktionen.

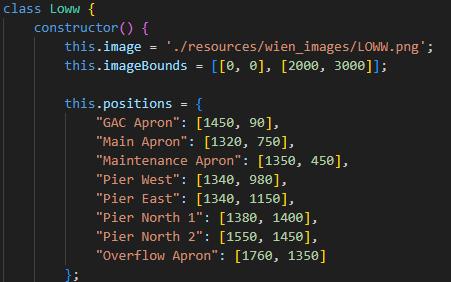
Eine Besonderheit in der JavaScript-Struktur der Startseite ist die Einbindung von jeweils einer eigenen JavaScript-Datei pro Flughafen. Diese Dateien enthalten die notwendigen Datenstrukturen für:

Abbildung 14: LOWW.js

* Abstellpositionen
* Exits
* Holding Points
* Runways
* Sowie das zugehörige Hintergrundbild und die Bildgrenze für das Leaflet.

Diese Struktur macht es möglich, dass beim Wechsel des Flughafens im Dropdown-Menü die passende Klasse geladen und auf der Karte angezeigt wird – die einzelnen Datenobjekte unterscheiden sich nur in den Koordinaten und der Anzahl der Marker.

Diese Trennung von Anwendungslogik und Datendefinition erhöht die Modularität und Wartbarkeit der Anwendung. Neue Flughäfen können durch Hinzufügen einer weiteren Datei nach demselben Muster unkompliziert ergänzt werden.

* 1. Informationsseite

Die Info-Seite zeigt hinterlegte Informationen und Bilder zu den sechs Flughäfen. Über ein Dropdown-Menü können Flughäfen ausgewählt werden. Per JavaScript wir dann der Titel, der Beschreibungstext sowie ein Satz an Bildern geladen und dargestellt.

Im HTML sind dafür eigene Platzhalter definiert:

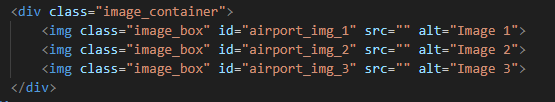


Abbildung 15: Infoseite HTML

Die JavaScript-Datei liest dynamisch aus vorbereiteten Bildordnern und Textdateien die passenden Inhalte ein. Auch der Hintergrund der Seite wir je nach Auswahl automatisch angepasst.

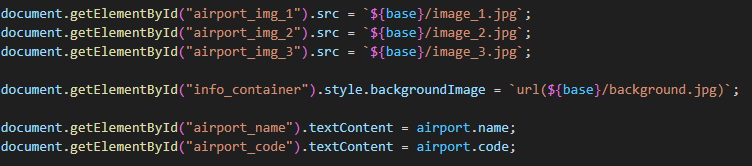


Abbildung 16: Infoseite JS

Das CSS sorgt für eine gleichmäßige Bildanzeige, ein responsives Layout und ein ansprechendes Design.

* 1. Kontaktseite

Die Kontaktseite bietet ein übersichtlich gestaltetes Formular zur Kontaktaufnahme, das es Nutzerinnen und Nutzern ermöglicht, direkt Rückmeldungen zur Webanwendung zu geben oder Fragen zu stellen.

Der Aufbau des Formulars erfolgt über klassische HTML-Formelemente:

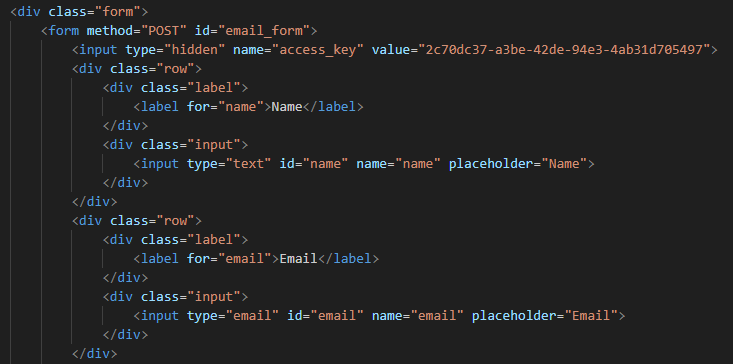


Abbildung 17: Kontaktseite HTML

Das zugehörige CSS sorgt für eine einheitliche Gestaltung der Formularfelder.

Die Funktion zur Übermittlung der Nachricht wird direkt in der HTML-Datei über ein eingebettetes JavaScript realisiert. Diese verwendet die Fetch API, um die Formulardaten als JSON an den externen Dienst „Web3Forms“ zu senden. Web3Forms übernimmt den Versand der E-Mail ohne, dass ein eigener Mailserver notwendig ist.

Ein Auszug aus dem verwendeten JavaScript:

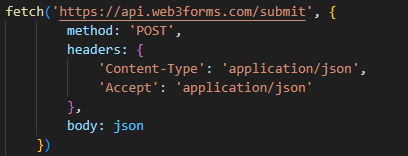


Abbildung 18: Kontaktseite JS

Das JavaScript zur Formularverarbeitung wurde nicht ausgelagert, sondern direkt in die HTML-Datei eingebettet. Der Grund dafür liegt in der Übersichtlichkeit und Einfachheit: Die Funktion ist relativ kurz, spezifisch für genau diese eine Seite und muss nicht an anderer Stelle wiederverwendet werden. Eine Trennung in eine eigene Datei hätte keinen funktionalen Vorteil gebracht.

* 1. Impressum

Wird noch gemacht.

1. Sonstige verwendete Technologien und Programme
   1. Leaflet.js – Interaktive Kartendarstellung

Die JavaScript-Bibliothek wurde verwendet, um die interaktive Darstellung der Flughafenkarten umzusetzen. Leaflet ist eine moderne Open-Source Bibliothek zur Erstellung mobiler und performanter Kartenanwendungen. In diesem Projekt wurde Leaflet genutzt, um Marker auf eine grafische Darstellung (statt auf einer georeferenzierten Weltkarte) zu platzieren und mit Benutzerinteraktionen zu verbinden.

Die Marker werden dynamisch gesetzt und ermöglichen durch Klicks eine schrittweise Auswahl der Position, Exit, Holding Point und Runway. Am Ende generiert das System automatisch eine passende Taxi-Instruktion.

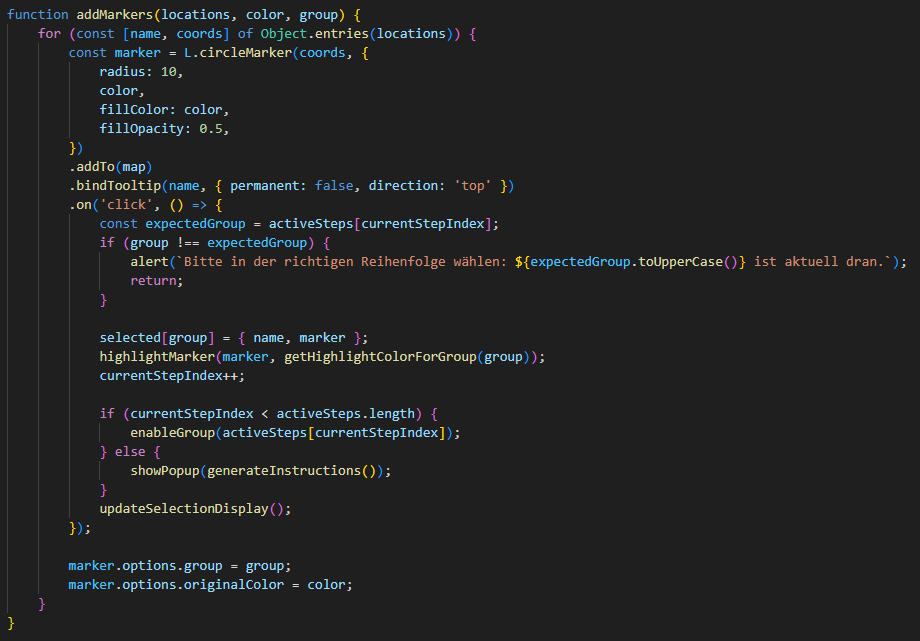


Abbildung 19: Leaflet

Leaflet wurde bewusst gewählt, da es sehr leichtgewichtig, gut dokumentiert und flexibel für benutzerdefinierte Karten wie Flughafenlayouts einsetzbar ist.

* 1. GIMP – Bildbearbeitung

Für die Erstellung und Bearbeitung der Flughafengrafiken sowie der Informationsbilder auf der Info-Seite wurde das kostenlose Bildbearbeitungsprogramm GIMP (GNU Image Manipulation Program) verwendet. GIMP bietet umfangreiche Werkzeuge zur Bearbeitung, Skalierung und Optimierung von Bildern – vergleichbar mit kommerzieller Software wie Adobe Photoshop

Im Rahmen dieses Projektes wurde GIMP insbesondere für folgende Aufgaben verwendet:

* Erstellen, freistellen und Zuschneiden von Flughafenplänen
* Anpassen von Farben und Kontrast der Fotos für bessere Lesbarkeit
* Exportieren in geeignete Formate und Größen für die Webanwendung

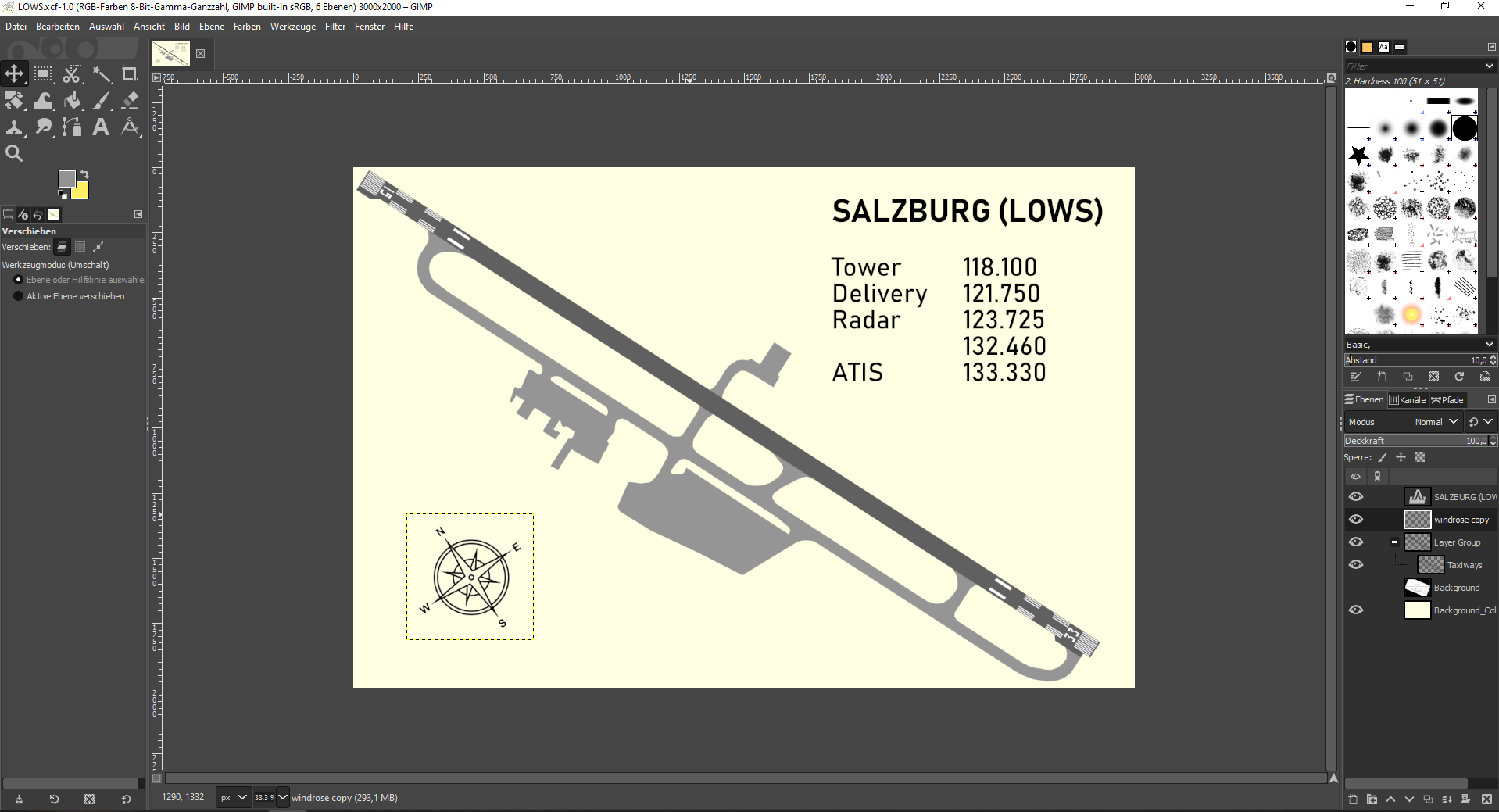


Abbildung 20: GIMP

* 1. Moongoose.exe – Lokaler Webserver

Für das Projekt wurde das kleine, portable Programm moongoose.exe verwendet, welches als lokaler Webserver dient. Dieser Webserver wurde genutzt, um die Anwendung lokal im Browser zu testen, insbesondere wegen der Ladebeschränkungen für Dateien bei lokalen HTML-Dateien. Die CORS-Richtlinien (Cross-Origin Resource Sharing) verhindern, dass Webanwendungen unkontrolliert Daten von einer anderen Domain oder Quelle nachladen.

Wenn also eine HTML-Datei im Browser geöffnet wird und darin mit JavaScript versucht wird, Daten aus Unterordnern oder anderen Quellen nachzuladen, blockiert der Browser dies standardmäßig, sofern nicht bestimmte Bedingungen erfüllt sind.

Im Rahmen dieses Projekts betraf das konkret die Funktion, bei der über JavaScript Info-Texte und Bilder zu den Flughäfen dynamisch nachgeladen werden sollen. Beim Öffnen der Anwendung über den lokalen Dateipfad funktioniert das Nachladen dieser Ressourcen nicht. Um dieses Problem zu umgehen und die Anwendung realitätsnah testen zu können, wurde die kleine Server-Software moongoose.exe eingesetzt. Durch das Starten eines lokalen Servers wird die Anwendung nicht mehr über den Dateipfad geöffnet, sondern über „http://localhost“, womit die Browserrestriktionen bezüglich CORS entfallen.

1. Herausforderungen & gewonnene Erkenntnisse

Im Zuge der Umsetzung dieses Projekts ergaben sich mehrere Herausforderungen, die sowohl technischer als auch konzeptioneller Natur waren. Diese machten die Arbeit an der Webanwendung nicht nur spannend und lehrreich, sondern führten auch zu einem spürbaren Kompetenzzuwachs in verschiedenen Bereichen der Webentwicklung.

Eine der ersten Schwierigkeiten bestand in der Auswahl und sinnvollen Strukturierung der Inhalte. Bereits in der Planungsphase musste entschieden werden, welche Flughafendaten tatsächlich dargestellt werden sollen, wie viele Auswahloptionen technisch sinnvoll sind und wie komplex die Taxi-Instruktionen aufgebaut sein dürfen, ohne die Nutzerfreundlichkeit zu beeinträchtigen. Hier zeigte sich, dass eine klare Reduktion auf das Wesentliche notwendig war, um das Ziel der Anwendung – eine intuitive und verständliche Bedienung – nicht zu gefährden.

Ein weiteres zentrales Lernfeld war der Umgang mit Leaflet.js. Während die Einbindung von Karten in Projekten mit geografischen Koordinaten gut dokumentiert ist, stellte die Verwendung statischer Flughafenbilder mit selbst definierten Koordinatensystemen eine besondere Herausforderung dar. Es war notwendig, für jeden Flughafen eigene Bildgrößen, Zoom-Stufen und Marker-Koordinaten zu definieren und diese exakt aufeinander abzustimmen. Diese technische Umsetzung erforderte viel Feinarbeit und präzises Testen.

Auch das Nachladen von Inhalten (wie Flughafenbeschreibungen und Bildern) auf der Info-Seite brachte anfangs Schwierigkeiten mit sich. Aufgrund der CORS-Richtlinien blockierte der Browser den Zugriff auf lokale Dateien. Erst durch die Integration eines lokalen Webservers (mongoose.exe) konnte dieses Problem behoben werden. Dieser Schritt war für die Funktionsfähigkeit der Anwendung entscheidend und stellte ein konkretes Praxisbeispiel für ein realitätsnahes Webproblem dar.

Im Bereich JavaScript wurde besonders deutlich, wie wichtig saubere Code-Struktur und durchdachte Logik sind. Das Erfassen der Nutzerinteraktionen – also das systematische Durchlaufen der Auswahl von Position, Exit, Holding Point und Runway – musste so gestaltet werden, dass keine unlogischen oder fehlerhaften Instruktionen entstehen. Die Entwicklung eines funktionierenden, robusten Auswahlmechanismus war zwar zeitintensiv, führte aber zu einem tiefen Verständnis für eventgesteuerte Programmierung, Fehlerbehandlung und Benutzerführung im Web.

Neben der technischen Entwicklung waren auch gestalterische Entscheidungen herausfordernd. Die Anwendung sollte modern, aber nicht überladen wirken. Durch den Einsatz von GIMP konnten Grafiken angepasst und in einheitlicher Qualität in die Anwendung integriert werden. Dies betraf sowohl die Kartenansichten als auch die Flughafenbilder auf der Informationsseite.

Rückblickend betrachtet war dieses Projekt nicht nur eine technische, sondern auch eine konzeptionelle und gestalterische Auseinandersetzung mit einem Thema, das mich persönlich und beruflich stark interessiert. Ich konnte meine Fähigkeiten im Umgang mit HTML, CSS, JavaScript und Webtools deutlich erweitern und habe ein besseres Verständnis für die Zusammenhänge zwischen Struktur, Design und Funktion in einer Webanwendung gewonnen. Besonders wertvoll war auch die Erfahrung, ein Projekt vollständig selbst zu planen, umzusetzen und dabei mit realitätsnahen Herausforderungen konfrontiert zu werden – ganz im Sinne praxisorientierten Lernens.

Abbildungsverzeichnis

[Abbildung 1: Zeitplan 6](file:///C:\Users\Vali\Downloads\Vorlage%20Projektarbeit.docx#_Toc198061700)

[Abbildung 2: Figma 7](#_Toc198061701)

[Abbildung 3: Startseite 7](#_Toc198061702)

[Abbildung 4: Infoseite 8](#_Toc198061703)

[Abbildung 5: Kontaktseite 8](#_Toc198061704)

[Abbildung 6: Visual Studio 9](#_Toc198061705)

[Abbildung 7: Dropdown 10](#_Toc198061706)

[Abbildung 8: JavaScript 11](#_Toc198061707)

[Abbildung 9: CSS 12](#_Toc198061708)

[Abbildung 10: Navbar 13](#_Toc198061709)

[Abbildung 11: Navbar CSS 13](#_Toc198061710)

[Abbildung 12: Startseite HTML 14](#_Toc198061711)

[Abbildung 13: Startseite JS 15](#_Toc198061712)

[Abbildung 14: LOWW.js 15](file:///C:\Users\Vali\Downloads\Vorlage%20Projektarbeit.docx#_Toc198061713)

[Abbildung 15: Infoseite HTML 16](#_Toc198061714)

[Abbildung 16: Infoseite JS 16](#_Toc198061715)

[Abbildung 17: Kontaktseite HTML 16](#_Toc198061716)

[Abbildung 18: Kontaktseite JS 17](#_Toc198061717)

[Abbildung 19: Leaflet 18](#_Toc198061718)

[Abbildung 20: GIMP 19](#_Toc198061719)

Literarturverzeichnis

„Visualstudio“. (11.05.2025). Visualstudio. Von   
https://visualstudio.microsoft.com/de/ abgerufen

„Wiki/Hypertext Markup Language“. (11.05.2025). Wikipedia. Von  
https://de.wikipedia.org/wiki/Hypertext\_Markup\_Language abgerufen

„What Is/Javascript“. (12.05.2025). Amazon. Von  
https://aws.amazon.com/de/what-is/javascript/ abgerufen

„Websites/Webdesign/Was ist CSS”. (12.05.2025). IONOS. Von  
https://www.ionos.at/digitalguide/websites/webdesign/was-ist-css/ abgerufen

Projektantrag