# Lösungsentwurf – Modernisierung der Software für den Flugplatz Seitenstetten

## Einleitung / Zielsetzung

Der Flugplatz Seitenstetten verwendet derzeit eine veraltete Softwarelösung für die Verwaltung von Piloten, Flugzeugen, Reservierungen und Kostenstellen. Ziel der Diplomarbeit ist es, diese Software durch eine moderne, benutzerfreundliche und erweiterbare Web-Applikation zu ersetzen.

Die neue Lösung soll:

* Eine einfach zu bedienende Oberfläche für Piloten und Administratoren bieten
* Eine sichere Benutzerverwaltung mit Authentifizierung enthalten
* In Echtzeit synchronisierten Daten anzeigen
* Zukunftssicher, wartbar und erweiterbar sein

## Rahmenbedingungen

* Zielgruppe: Piloten und Administratoren des Flugplatzes Seitenstetten
* Systemanforderungen:
  + Datenbankanbindung für Piloten-, Flugzeug- und Reservierungsdaten
  + Authentifizierung mit Benutzer und Rollenverwaltung
  + Einsatz auf einem Server mit .NET-Umgebung
* Nicht-Ziele: Entwicklung einer mobilen App, Offline-Unterstützung (nur Webbrowser erforderlich)

## Lösungsansatz

Nach Analyse verschiedener Frameworks (ASP.NET Web App vs Blazor) wurde Blazor als technologische Basis gewählt.

Begründung:

* Einheitliche Programmiersprache (C# für Frontend und Backend)
* Geringerer Einarbeitungsaufwand (bestehendes C#-Wissen nutzbar)
* Echtzeit-Synchronisation über SignalR (Wichtig für Reservierungen und Verwaltungsaufgaben)
* Einfaches Hosting auf Servern mit .NET-Umgebung
* Gute Dokumentation und große Community

Für die Benutzeroberfläche wird MudBlazor verwendet, ein UI-Framework auf Basis von Material Design. Dies ermöglicht ein modernes und konsistentes Erscheinungsbild.

## Blazor

Die geplante Anwendung basiert auf einer 3-Schichten-Architektur:

1. Präsentationsschicht

* Blazor-Komponenten (UI)
* Umsetzung mit MudBlazor (Formulare, Dialoge, Tabellen, FIlter)

2. Logikschicht

* Geschäftslogik in C#
* Rollenverteilung (Pilot / Admin)
* Validierung von Eingaben und Prozessen

3. Datenzugriffsschicht

* Entity Framework Core als ORM
* SQL-Datenbank (Maria DB)
* Definition von Datenmodellen (Pilot, Flugzeug, Reservierung, Kostenstelle, User)

Systemübersicht

* Benutzer meldet sich über Login-Seite an 🡪 System prüft Rolle (Pilot/Admin).
* Abhängig von der Rolle wird das entsprechende Dashboard geladen
* Änderungen (z.B. neue Reservierung, Bearbeitung von Flugzeugdaten) werden in Echtzeit mit allen Clients synchronisiert

## Ablauf

**Pilot-Workflow**

1. Pilot loggt sich ein
2. Pilot öffnet die Reservierungsseite
3. Pilot wählt Flugzeug und Zeitfenster
4. System prüft Verfügbarkeit
5. Reservierung wird gespeichert und ist sofort für Admins sichtbar

**Admin Workflow**

1. Admin loggt sich ein
2. Admin hat Zugriff auf Benutzer-, Flugzeug-, Kostenstellen-, und Reservierungsverwaltung
3. Admin kann Benutzer hinzufügen, Flugzeuge verwalten oder Reservierungen einsehen/ändern
4. Änderungen werden in Echtzeit im System sichtbar

**Wart Workflow**

1. Wart loggt sich ein
2. Wart hat öffnet Start-/Landeliste
3. Wart kann den Flugverkehr protokollieren, z.B.:

* Start, Landung, Zwischenlandung
* Schleppflug, Wartungsflug, Schulungsflug usw.
* Uhrzeiten von Abgeben und Landen

1. Wart kann Berichte einsehen und Rechnungen erstellen (z.B. für Landegebühren und Treibstoffverbrauch).
2. Alle Änderungen werden gespeichert und sind in Echtzeit für Admins und ggf. Piloten sichtbar

## Technologien & Werkzeuge

* Framework: Blazor Server (.Net 8)
* UI: MudBlazor (Material Design)
* Datenbank: Maria DB
* ORM: Entity Framework Core
* Authentifizierung: ASP.NET Core Identity
* Entwicklungsumgebung: (Visual Studio)
* Design Entwürfe: Figma

## Beispielhafte Szenarien

* Pilot: „Ich möchte einen Flug für Samstag reservieren.“
  + Pilot loggt sich ein 🡪 Reservierung anlegen 🡪 DB-Eintrag 🡪 Admin sieht Reservierung sofort.
* Admin „Ich möchte ein neues Flugzeug hinzufügen.“
  + Admin öffnet die Flugzeugverwaltung 🡪 Formular ausfüllen 🡪 Flugzeug wird gespeichert 🡪 steht sofort zur Auswahl für Reservierungen bereit.
* Wart „Ich möchte einen Flugverkehr protokollieren.“
  + Wart loggt sich ein 🡪Start-/Landeliste öffnen 🡪 Er trägt Nummer, Kennzeichen, Typ, Pilot, Startzeit (Einflug/Ausflug), Dauer, Flugart und Bemerkung ein 🡪 Daten sind sofort im System verfügbar.

## Bewertung & Alternativen

* ASP.NET MVC / Razor Pages: Sehr performant, aber höhere Komplexität durch C#, Razor und JavaScript.
* Blazor WebAssembly: Offlinefähig und entlastet den Server jedoch längere Ladezeiten und komplexeres Hosting.
* Blazor Server (gewählt):
  + Einheitliche Sprache (C#)
  + Echtzeitfähigkeit
  + Einfaches Hosting
  + Konsistentes, modernes UI mit MudBlazor

Die Entscheidung für Blazor Server ist daher optimal für die Anforderungen des Projekts.

## Fazit & Nächste Schritte

Der Lösungsentwurf definiert eine klare, moderne und skalierbare Architektur für die neue Verwaltungssoftware am Flugplatz Seitenstetten.

Nächste Schritte:

* Datenmodellierung (ER-Diagramm, Tabellen, Relationen)
* Implementierung der Benutzer und Rollenverwaltung
* Umsetzung der Pilotenseiten (Reservierungen, Flüge, Profil)
* Umsetzung der Adminseiten (Benutzer, Flugzeuge, Kostenstellen, Reservierungen)
* Testphase mit Piloten und Admins
* Finales Deployment und Übergabe