Ski-Service .nET-APP

Projektdokumentation

Fokko Vos & Robin Ruf

IBZ 2023

Inhaltsverzeichnis

[1 Einleitung 3](#_Toc152455321)

[2 Informieren 4](#_Toc152455322)

[2.1 Ausgangssituation 4](#_Toc152455323)

[2.2 Anforderungsanalyse 4](#_Toc152455324)

[2.2.1 Funktionale Anforderungen 4](#_Toc152455325)

[2.2.2 Nicht-Funktionale Anforderungen 4](#_Toc152455326)

[2.3 Technische Anforderungen 5](#_Toc152455327)

[3 Planen 5](#_Toc152455328)

[3.1 Zeitplan 5](#_Toc152455329)

[3.2 Systemarchitekturentwurf 6](#_Toc152455330)

[3.3 Mockups 7](#_Toc152455331)

[Entscheiden 12](#_Toc152455332)

[3.4 Technologie und Testgeräte 12](#_Toc152455333)

[3.5 Test-Strategie 12](#_Toc152455334)

[4 Realisieren 13](#_Toc152455335)

[5 Kontrollieren 13](#_Toc152455336)

[6 Auswerten 13](#_Toc152455337)

[7 Anhänge 14](#_Toc152455338)

[7.1 Quellen 14](#_Toc152455339)

[7.2 Abbildungsverzeichnis 14](#_Toc152455340)

[7.3 Tabellenverzeichnis 14](#_Toc152455341)

Versionsverlauf

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Datum | Autor | Bemerkung |
| 1.0 | 28.09.2023 | Fokko Vos & Robin Ruf | Initial Version |
| 1.1 | 03.12.2023 | Fokko Vos & Robin Ruf | Vollenden der Planung |
| 1.2 | 10.12.2023 | Fokko Vos & Robin Ruf | Vollenden der Entscheidung |
| 1.3 | 30.12.2023 | Fokko Vos & Robin Ruf | Vollenden der Realisierung |

1 - Versionsverlauf

# Einleitung

Willkommen zur Dokumentation des Projekts „Ski-Service.NET App“ der Firma Jetstream-Service. In einer Welt, in der digitale Technologien zunehmend an Bedeutung gewinnen, ist die Anpassung und Optimierung von Geschäftsprozessen durch innovative Lösungen ein Schlüsselfaktor für den Unternehmenserfolg. Die Ski-Service.NET App ist eine Antwort auf die Notwendigkeit, den Workflow und die Datenverwaltung im Bereich der Ski-Service-Aufträge effizienter zu gestalten.

Diese Dokumentation bietet einen umfassenden Einblick in das Projekt, das darauf abzielt, eine benutzerfreundliche und für unterschiedliche Endgeräte optimierte Anwendung zu entwickeln. Unser Ziel ist es, eine intuitive Benutzeroberfläche zu schaffen, die den spezifischen Anforderungen und Arbeitsbedingungen einer Skiservice-Werkstatt gerecht wird.

In dieser Dokumentation werden alle Phasen des Projekts abgedeckt, von der Konzeption bis zur technischen Umsetzung, einschließlich der Überlegungen zu den verwendeten Technologien und der Integration bestehender Systeme. Sie dient als Leitfaden und Informationsquelle für alle Projektbeteiligten und Stakeholder und gibt einen klaren Überblick über die Ziele, Strategien und erwarteten Ergebnisse des Projekts.

# Informieren

## Ausgangssituation

**Unternehmen**  
Jetstream-Service, ein Unternehmen, das sich auf Skiservicearbeiten spezialisiert hat.

**Aktueller Stand**  
Das Unternehmen hat neue Touchscreen-fähige Hardware angeschafft (Tablets, Surface) und stellt diese den Mitarbeitern für die Datenpflege der Ski-Service-Aufträge zur Verfügung.

**Ziel**  
Die einfache, intuitive und aufgabenangemessene Bedienung der Benutzeroberfläche, unter Berücksichtigung der Arbeit mit Handschuhen.

**Integration**  
Erweiterung der existierenden Datenbasis zur Online-Anmeldung und der Service-App mit der neuen GUI-Lösung für Tablets und Handys.

## Anforderungsanalyse

### Funktionale Anforderungen

* Sicheres Login-System für Mitarbeiter mit Benutzername und Passwort.
* Verwaltung von Serviceaufträgen
  + Anzeige aller anstehenden Ski-Serviceaufträge in einer Liste.
  + Möglichkeit zur Filterung und Suche innerhalb der Serviceauftragsliste.
  + Fähigkeit zur Mutation (Statusänderung, Notizen hinzufügen, Löschen) bestehender Serviceaufträge.
  + Unterschiedliche Statusoptionen für Aufträge: Offen, In-Arbeit, Abgeschlossen.
  + Zuordnung und Verwaltung von Prioritäten bei Serviceaufträgen.
  + Ergänzung von Kundeninformationen (Name, E-Mail, Telefon) bei Bedarf.
  + Eindeutige Zuordnung einer Dienstleistung pro Serviceauftrag aus einer vorgegebenen Liste (z.B. Kleiner Service, Großer Service, Rennski-Service).

### Nicht-Funktionale Anforderungen

* Starke Authentifizierung und Autorisierung, um den Zugang zu sensiblen Daten zu sichern.
* Schnelle Verarbeitung von Anfragen, um eine effiziente Abwicklung der Serviceaufträge zu gewährleisten.
* Einfache, intuitive und aufgabenangemessene Benutzeroberfläche, die auch mit Handschuhen bedienbar ist.
* Stabile und fehlerresistente Anwendungsleistung, insbesondere während der Hochsaison.

## Technische Anforderungen

**UX und UI Design**  
Erstellung von Mockups, die eine einfache und aufgabenangemessene Bedienung ermöglichen, unter besonderer Berücksichtigung der Bedienbarkeit mit Handschuhen.

**Integration und Kompatibilität**  
Sicherstellung, dass das neue User Interface reibungslos mit dem bereits existierenden Backend-System integriert wird. Dies beinhaltet die Anbindung an die bestehenden Web-APIs und Datenstrukturen. Überprüfung und ggf. Anpassung der Schnittstellen, um eine effiziente Kommunikation zwischen dem neuen Frontend und dem existierenden Backend zu gewährleisten.

**Test und Qualitätssicherung**  
Durchführung umfassender Tests, um die Funktionalität und Leistung der neuen UI-Komponenten sowie deren Integration mit dem Backend zu überprüfen.

# Planen

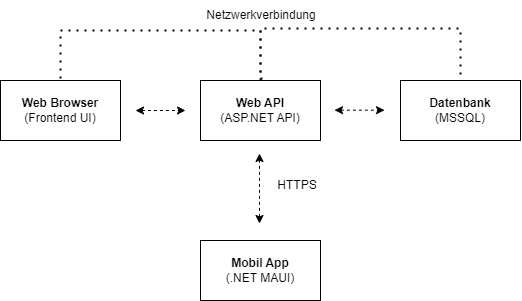
## Zeitplan

Der Aufwand in Stunden bezieht sich auf die von allen Projektmitgliedern gemeinsam aufgewendete Zeit.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nr.** | **Beschreibung** | **SOLL-Zeit (h)** | **IST-Zeit (h)** |
| **1** | **Informieren** | **6** |  |
| 1.1 | Situationsanalyse | 2 | 2 |
| 1.2 | Erfassung der Nutzerbedürfnisse | 3 | 2 |
| 1.3 | Erfassung der Schulischen anforderungen | 1 | 1 |
| **2** | **Planen** | **12.5** |  |
| 2.1 | Erstellung eines Zeitplans | 1.5 | 1.5 |
| 2.2 | Entwurf der Systemarchitektur | 1 | 1 |
| 2.3 | Mockups für die Benutzeroberfläche | 10 | 9 |
| **3** | **Entscheiden** | **2.5** |  |
| 3.1 | Technologienwahl | 0.5 | 1 |
| 3.2 | Test-Strategie | 2 | 2 |
| **4** | **Realisieren** | **24.5** |  |
| 4.1 | Git-Repository Einrichten | 0.5 | 0.5 |
| 4.2 | Verbindung zum Backend aufbauen | 1 |  |
| 4.3 | Entwicklung der Benutzeroberfläche | 15 |  |
| 4.4 | Entwicklung der nötigen Tests | 8 |  |
| **5** | **Kontrollieren** | **3** |  |
| 5.1 | Test-Strategie ausführen | 1 |  |
| 5.2 | Anforderungen mit dem Produkt abgleichen | 2 |  |
| **6** | **Auswerten** | **6** |  |
| 6.1 | Finalisierung der Dokumentation | 2 |  |
| 6.2 | Lessons-Learned Identifizieren | 2 |  |
| 6.3 | Präsentation vorbereiten | 2 |  |
| **Gesamt** | | **54.5** |  |

2 - Zeitplan

## Systemarchitekturentwurf



1 - Systemarchitekturentwurf

Das System, das wir für die "Ski-Service.NET App" entwerfen, ist eine Weiterentwicklung der bestehenden Drei-Schichten-Architektur und integriert neue Komponenten für erweiterte Funktionalität und Benutzerfreundlichkeit. Es folgt eine Beschreibung der aktualisierten Systemarchitektur basierend auf dem visuellen Entwurf.

**Web Browser (Frontend UI)**  
Die Präsentationsschicht, zugänglich über einen Web Browser, bleibt die primäre Schnittstelle für Benutzerinteraktionen

**Web API (ASP.NET API)**  
Die Geschäftslogikschicht wird durch eine ASP.NET API repräsentiert, die das Rückgrat unserer Anwendung bildet. Sie verarbeitet Anfragen, führt Geschäftsregeln aus und handhabt die Datenkommunikation mit der Datenbank.

**Datenbank (MSSQL)**  
Die Datenpersistenzschicht basiert weiterhin auf Microsoft SQL Server, welcher die notwendigen Daten speichert und bereitstellt. Diese Schicht ist für hohe Leistung und Transaktionsintegrität optimiert und ermöglicht schnelle Datenabfragen.

**Mobile App (.NET MAUI)**  
Als neue Komponente führen wir eine mobile Anwendung ein, die mit .NET MAUI entwickelt wird. Diese App erweitert die Zugänglichkeit der Systemfunktionen und bietet eine optimierte Benutzererfahrung für mobile Geräte.

Die Netzwerkverbindung zwischen allen Schichten gewährleistet sichere und effiziente Datenübertragungen über HTTPS, wobei JSON als Datenaustauschformat zum Einsatz kommt. Dieses Format unterstützt eine hohe Interoperabilität und erleichtert das Parsen der Daten.

Diese erweiterte Architektur erhält die bewährten Prinzipien der klaren Schichtentrennung und fügt gleichzeitig neue Elemente hinzu, um die Anwendung modern, flexibel und zukunftssicher zu machen. Sie berücksichtigt sowohl die aktuellen Anforderungen als auch die Skalierbarkeit für zukünftige Erweiterungen und ist somit ein solides Fundament für die fortlaufende Entwicklung unserer Projektziele.

## Mockups

Die Texte welche in den Mockups verwendet werden, dienen nur zur Orientierung und stehen noch nicht endgültig fest.

|  |
| --- |
| 2 - Mockup - Anmeldung |
| 3 - Mockup - Dashboard |
| 4 - Mockup - Menu Open |
| 5 - Mockup - Liste  6 - Mockup - Nutzer liste |
|  |
| 7 - Mockup - Auftrag Informationen  8 - Mockup - Auftrag bearbeiten |
|  |
| 9 - Mockup - Stornieren Dialog  10 - Mockup – Logout Dialog |
|  |

# Entscheiden

## Technologie und Testgeräte

Die technologische Basis für das Projekt "Ski-Service.NET App" bildet die Entscheidung für die Nutzung von .NET MAUI und .NET 8, wodurch wir die Vorteile der neuesten Entwicklungen in der Softwaretechnologie voll ausschöpfen. .NET MAUI ermöglicht uns eine plattformübergreifende Entwicklung, während .NET 8 uns Zugriff auf die neuesten Sprachfeatures, Sicherheitsupdates und Performance-Optimierungen bietet.

Für die Entwicklungsumgebung haben wir uns auf Visual Studio festgelegt. Diese IDE stellt uns alle notwendigen Werkzeuge und Emulatoren zur Verfügung, um eine effiziente Entwicklung und ein akkurates Testing zu ermöglichen. Die Emulatoren in Visual Studio erlauben es uns, ein breites Spektrum an Geräten zu simulieren, um so die Kompatibilität und Benutzerfreundlichkeit unserer App sicherzustellen.

Speziell für die Testzwecke haben wir uns entschieden, das "Tablet 420 DPI 8in" als Standard-Testgerät zu verwenden. Dieses Gerät, mit einem Arbeitsspeicher von 4 GB und einer Auflösung von 2200 x 2480 bei 420 DPI, gewährleistet, dass wir die App unter realistischen Bedingungen entwickeln und testen können. Diese Spezifikationen stellen sicher, dass die App performant läuft und die Benutzererfahrung optimiert ist, insbesondere in Bezug auf die Darstellung und Handhabung der Benutzeroberfläche auf Geräten mit hoher Pixel- und Punktdichte (DPI).

## Test-Strategie

Für die "Ski-Service.NET App" wird eine mehrschichtige Teststrategie verfolgt, die sicherstellt, dass sowohl die einzelnen Komponenten als auch die Anwendung als Ganzes die festgelegten Anforderungen und Qualitätsstandards erfüllen. Die Entscheidung für unsere Teststrategie beinhaltet folgende Kernpunkte

**Unit Tests**  
Werden verwendet, um die grundlegenden Bausteine der Anwendung zu prüfen.

**Integrationstests**  
Stellen das korrekte Zusammenspiel der einzelnen Module und Dienste sicher.

**End-to-End-Tests**  
Simulieren den vollständigen Ablauf von wichtigen Benutzerinteraktionen mit der Anwendung.

Zusätzlich zu diesen automatisierten Testverfahren werden manuelle **Usability Tests** während der gesamten Entwicklung durchgeführt. Diese Tests sind entscheidend, um ein benutzerzentriertes Design zu gewährleisten und um sicherzustellen, dass die Anwendung nicht nur funktioniert, sondern auch effizient und angenehm zu bedienen ist. Die Usability Tests werden in verschiedenen Phasen der Entwicklung durchgeführt, um kontinuierliches Feedback in den Entwicklungsprozess einfließen zu lassen und eine hohe Gebrauchstauglichkeit zu sichern.

# Realisieren

## Maui Basisprojekt erstellen

Wir haben in Visual Studio ein .NET Maui Projekt erstellt. Dieses bietet uns eine gute Basisstruktur und damit einen einfachen Einstieg in unser Projekt.

**Grundstruktur des Maui-Projekts**

Die Erstellung des Maui-Projekts beinhaltete die Implementierung einer grundlegenden Struktur und Architektur für die App. Wir haben eine Lösungsdatei (SkiServiceApp.sln) erstellt und verschiedene XAML-Dateien für die Benutzeroberfläche, wie App.xaml, AppLogin.xaml und AppShell.xaml, hinzugefügt. Diese Dateien legen das visuelle Grundgerüst und die Navigationselemente der Anwendung fest.

**Backend-Kommunikation und Authentifizierung**

Ein zentraler Bestandteil des Projekts war die Einrichtung einer effizienten Backend-Kommunikation. Wir implementierten einen BaseAPIService, um alle Kommunikationen mit dem Backend zu handhaben. Zusätzlich wurde eine Login-Funktionalität integriert, die für die Nutzung der App erforderlich ist. Diese Schritte gewährleisten eine sichere und effiziente Benutzerauthentifizierung und -autorisation.

**Implementierung der JWT-Authentifizierung**

In unserem Ski-Service-App-Projekt haben wir eine sichere und effiziente Benutzerauthentifizierung und -autorisation durch die Implementierung von JWT (JSON Web Tokens) realisiert.

**Docker-Unterstützung**

Ein weiterer wichtiger Aspekt unseres Projekts war die Integration der Docker-Unterstützung. Durch die Bereitstellung von Dockerfiles und Skripten ermöglichten wir die Containerisierung der APIs und des Backends. Diese Integration vereinfacht die Inbetriebnahme der App. Die APIs und das Backend werden über Docker-Compose gestartet und dann ist die App direkt über einen Emulator einsatzbereit und funktionsfähig.

## Git-Repository aufsetzen

Für die Versionskontrolle haben wir uns für die beliebte Plattform GitHub entschieden. Wir haben ein neues Repository erstellt und einen initialen Commit gepusht. Folgende Dateien sind haben wir zusätzlich ergänzt:

**.gitignore**

Wir haben eine sorgfältig konfigurierte .gitignore-Datei hinzugefügt, um sicherzustellen, dass nur relevante Dateien und Verzeichnisse im Repository verfolgt werden. Diese Datei schließt unnötige oder vertrauliche Dateien aus, wie z.B. lokale Konfigurationsdateien und Build-Artefakte.

**Lizenz**

Die LICENSE-Datei definiert, wie andere unser Projekt verwenden dürfen. Dies ist wichtig für die Festlegung von Urheberrechten und Nutzungsbedingungen. Wir haben uns für die verbreitete Apache-Lizenz entschieden.

**README.md**

Eine zentrale Dokumentationsdatei, die README.md, wurde erstellt, um eine Übersicht über das Projekt, Installationsanweisungen, Nutzungshinweise und andere wichtige Informationen zu bieten. Der notwendige Inhalt wird zu einem späteren Zeitpunkt hinzugefügt. Vorerst wurde diese Datei nur initial gepusht.

## App Entwicklung

Die Entwicklung der Ski-Service-App war ein umfangreicher Prozess, der durch mehrere wichtige Commits gekennzeichnet war. Diese Commits spiegeln die verschiedenen Aspekte der App-Entwicklung wider, einschließlich der Implementierung des MVVM-Patterns, der Erstellung von Views, ViewModels, Models, der Unterstützung mehrerer Sprachen und der Entwicklung benutzerdefinierter Komponenten.

**Implementierung des MVVM-Patterns**

Das MVVM (Model-View-ViewModel) Pattern wurde als grundlegendes Architekturprinzip unserer App verwendet. Dieses Muster trennt die Geschäftslogik und die Benutzeroberfläche, was die Wartbarkeit und Testbarkeit der App verbessert. Die verschiedenen Commits zeigen die Entwicklung der einzelnen Komponenten dieses Musters:

**Model**  
Die Datenstrukturen und Geschäftslogik der App wurden in Model-Klassen definiert.

**View**  
Die Benutzeroberfläche der App wurde durch verschiedene XAML-Dateien realisiert, die die visuelle Darstellung der App definieren.

**ViewModel**  
Die Logik und das Verhalten der Views wurden in ViewModel-Klassen implementiert. Diese Klassen dienen als Bindeglied zwischen den Views und den Models.

**Multi-Lingual Support**

Unsere Ski-Service-App wurde mit einem umfangreichen Multi-Lingual Support entwickelt, um Benutzer weltweit anzusprechen und eine breite Nutzerbasis zu erreichen. Dieser Ansatz ermöglicht es Benutzern, die App in ihrer bevorzugten Sprache zu nutzen, was die Zugänglichkeit und Benutzerfreundlichkeit erheblich verbessert.

**Integration von Sprachressourcen**  
Die App integriert eine Vielzahl von Sprachressourcen, die in .resx-Dateien gespeichert sind. Diese Ressourcendateien enthalten lokalisierte Texte für verschiedene Elemente der Benutzeroberfläche, wie Menüs, Dialoge und Anweisungen. Die Verfügbarkeit mehrerer Sprachen, darunter Englisch, Spanisch, Französisch, Deutsch und viele andere, stellt sicher, dass die App für ein globales Publikum geeignet ist.

**Dynamische Sprachumschaltung**  
Die App unterstützt dynamische Sprachumschaltungen, was bedeutet, dass Benutzer die Sprache der App jederzeit ändern können, ohne die App neu starten zu müssen. Dies wird durch die Bindung der UI-Elemente an die Sprachressourcen erreicht, wodurch eine sofortige Aktualisierung der Texte bei Sprachwechsel ermöglicht wird.

**Entwicklung Benutzerdefinierter Komponenten**

Um die Benutzererfahrung zu verbessern und die Wiederverwendbarkeit des Codes zu fördern, wurden benutzerdefinierte Komponenten entwickelt. Diese Komponenten ermöglichen eine flexible und konsistente Darstellung von Elementen in der App.

**CustomFlyoutItem**Die CustomFlyoutItem-Komponente ist eine maßgeschneiderte Lösung für das Navigationsmenü unserer App. Sie verbessert die Benutzerinteraktion durch eine ansprechende und intuitive Gestaltung. Die Komponente nutzt BindableProperties für Icons, Titel und Routen, um eine flexible und dynamische Navigationserfahrung zu bieten.

**CancelDialog**Der CancelDialog ist eine Dialogkomponente, die den Benutzern ermöglicht, Aktionen wie das Stornieren von Aufträgen auf interaktive Weise zu bestätigen. Diese Komponente trägt zur Benutzerfreundlichkeit bei, indem sie klare und verständliche Interaktionsoptionen bietet.

**LogoutDialog**Der LogoutDialog ist eine weitere Dialogkomponente, die für die Bestätigung des Logout-Prozesses verwendet wird. Diese Komponente stellt sicher, dass die Benutzer ihre Absicht zum Abmelden klar bestätigen können, was zur Sicherheit der App beiträgt.

**CustomListItem**Die CustomListItem-Komponente ist eine benutzerdefinierte Ansicht für Listenelemente. Sie ermöglicht eine konsistente und ansprechende Darstellung von Daten in Listenformaten, was die Lesbarkeit und Benutzererfahrung verbessert.

**OrderList**Die OrderList-Komponente ist eine spezialisierte Ansicht für die Darstellung von Auftragslisten. Sie bietet Funktionen wie das Sortieren und Filtern von Aufträgen und unterstützt interaktive Aktionen wie das Anwenden, Ändern des Status oder Stornieren von Aufträgen.

**ModifyDialog**Der ModifyDialog ist eine Dialogkomponente, die es den Benutzern ermöglicht, Änderungen an Aufträgen oder Einstellungen auf eine einfache und intuitive Weise vorzunehmen. Diese Komponente trägt zur Flexibilität und Benutzerfreundlichkeit der App bei.

**Anpassungen für Diverse Plattformen**

Unsere App wurde für eine Vielzahl von Plattformen entwickelt, darunter Android, iOS, Windows und MacOS. Um die Appkompatibilität und Benutzererfahrung auf jeder Plattform zu gewährleisten, haben wir die nötigen Anpassungen vorgenommen.

# Kontrollieren

# Auswerten

# Anhänge

## Quellen

**Im aktuellen Dokument sind keine Quellen vorhanden.**

## Abbildungsverzeichnis

[1 - Systemarchitekturentwurf 6](file:///C:\Github\ict-322-ski-service-app\files\Dokumentation.docx#_Toc152455342)

[2 - Mockup - Anmeldung 7](file:///C:\Github\ict-322-ski-service-app\files\Dokumentation.docx#_Toc152455343)

[3 - Mockup - Dashboard 8](file:///C:\Github\ict-322-ski-service-app\files\Dokumentation.docx#_Toc152455344)

[4 - Mockup - Menu Open 8](file:///C:\Github\ict-322-ski-service-app\files\Dokumentation.docx#_Toc152455345)

[5 - Mockup - Liste 9](file:///C:\Github\ict-322-ski-service-app\files\Dokumentation.docx#_Toc152455346)

[6 - Mockup - Nutzer liste 9](file:///C:\Github\ict-322-ski-service-app\files\Dokumentation.docx#_Toc152455347)

[7 - Mockup - Auftrag Informationen 10](file:///C:\Github\ict-322-ski-service-app\files\Dokumentation.docx#_Toc152455348)

[8 - Mockup - Auftrag bearbeiten 10](file:///C:\Github\ict-322-ski-service-app\files\Dokumentation.docx#_Toc152455349)

[9 - Mockup - Stornieren Dialog 11](file:///C:\Github\ict-322-ski-service-app\files\Dokumentation.docx#_Toc152455350)

[10 - Mockup – Logout Dialog 11](file:///C:\Github\ict-322-ski-service-app\files\Dokumentation.docx#_Toc152455351)

## Tabellenverzeichnis

[1 - Versionsverlauf 3](#_Toc152455352)

[2 - Zeitplan 5](#_Toc152455353)