Ski-Service Backend Management

A screenshot of a computer program

Description automatically generated

Von Tunahan, Yannick und Felipe

[1. Informieren 2](#_Toc307118567)

[1.1 Anforderungen analysieren 3](#_Toc1509369853)

[1.2 Projektstruktur definieren 3](#_Toc906417134)

[1.3 Recherche und Analyse 3](#_Toc778207727)

[2. Planen 4](#_Toc1826005550)

[2.1 Architektur und Technologien 4](#_Toc2111599015)

[2.2 Zeit- und Aufgabenplan 5](#_Toc1932225495)

[2.3 Rollenverteilung 6](#_Toc1825767885)

[3. Entscheiden 7](#_Toc1111017108)

[3.1 Technologiewahl 7](#_Toc1473469144)

[3.2 Sicherheitsmaßnahmen 7](#_Toc63748017)

[4. Realisieren 7](#_Toc532745010)

[4.1 Backend 8](#_Toc380709067)

[4.2 Frontend 8](#_Toc235068506)

[4.3 Tests und Dokumentation 8](#_Toc1052009572)

[5. Kontrollieren 9](#_Toc1928670135)

[5.1 Testberichte 9](#_Toc542814746)

[5.2 Einhaltung der Anforderungen 9](#_Toc1832046951)

[6. Auswerten 9](#_Toc2061467269)

[6.1 Feedbackanalyse 10](#_Toc1678299033)

[6.2 Lessons Learned 10](#_Toc240411078)

[6.3 Weiterentwicklungsmöglichkeiten 10](#_Toc1155538237)

[Fazit 10](#_Toc819268601)

# 1. Informieren

Das Projekt hatte zum Ziel, ein Backend-System für die Verwaltung von Ski-Service-Aufträgen zu entwickeln. Dabei standen Benutzerfreundlichkeit, Datensicherheit und die Integration effizienter Prozesse im Fokus.

## 1.1 Anforderungen analysieren

Das System sollte folgende Hauptfunktionen erfüllen:

* Authentifizierung von Mitarbeitern über Login (Benutzername und Passwort).
* Verwaltung von Kundenaufträgen, einschließlich Statusänderungen und Filteroptionen.
* Datenbankdesign gemäß der 3. Normalform.
* Integration von API-Dokumentation (Swagger) und Sicherheitsmechanismen.

## 1.2 Projektstruktur definieren

Das Projekt wurde gemäß der IPERKA-Methode in klaren Schritten umgesetzt, mit einer Aufteilung der Aufgaben entsprechend den Stärken der Teammitglieder:

* **Backend:** Entwicklung der API und Datenbankintegration.
* **Frontend:** Gestaltung eines Interfaces zur Auftragsverwaltung.
* **Dokumentation und Tests:** Erstellung der Dokumentation, Zeitplanung und Qualitätssicherung.

##### 1.3 Recherche und Analyse

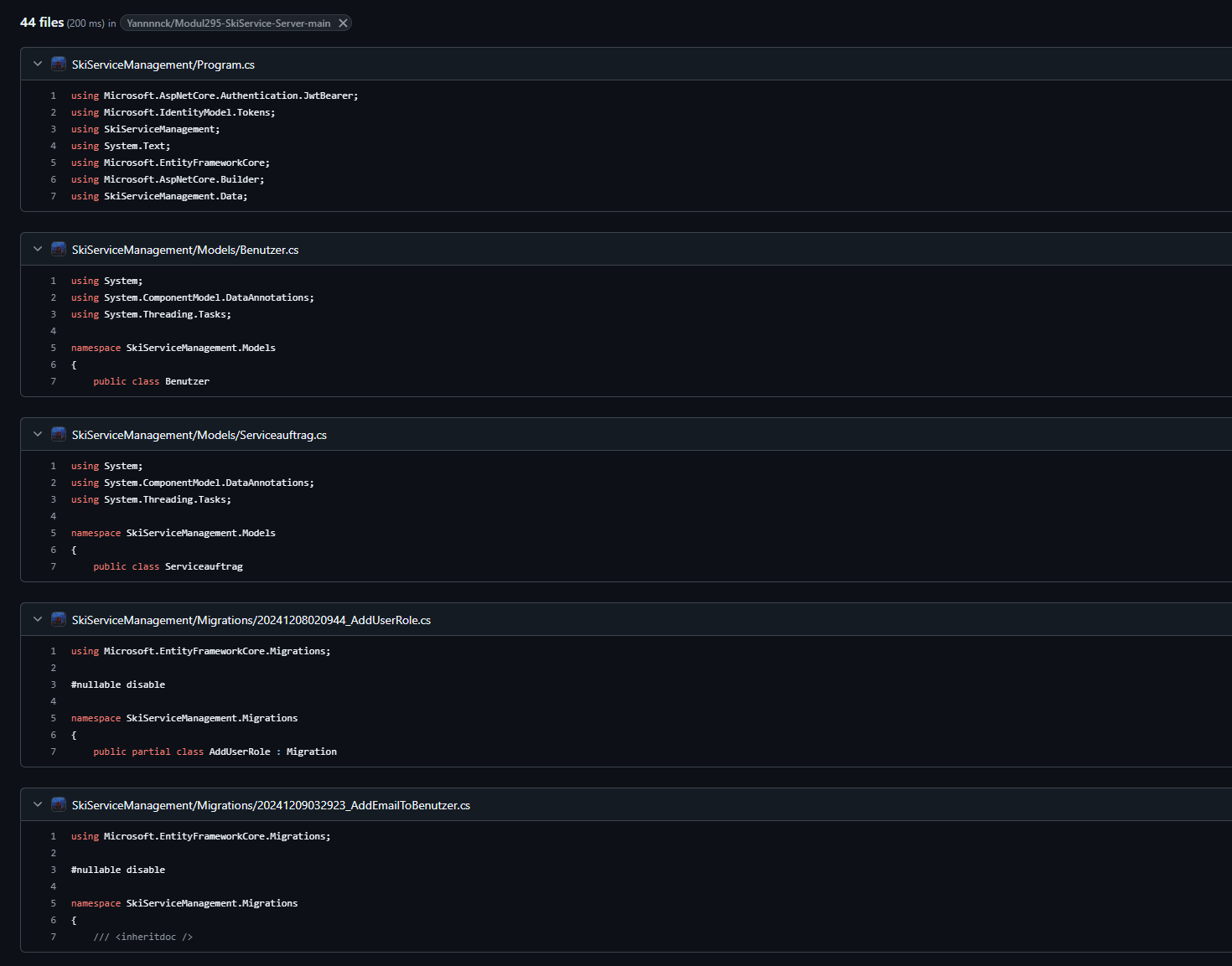
Es wurden bestehende Technologien evaluiert:

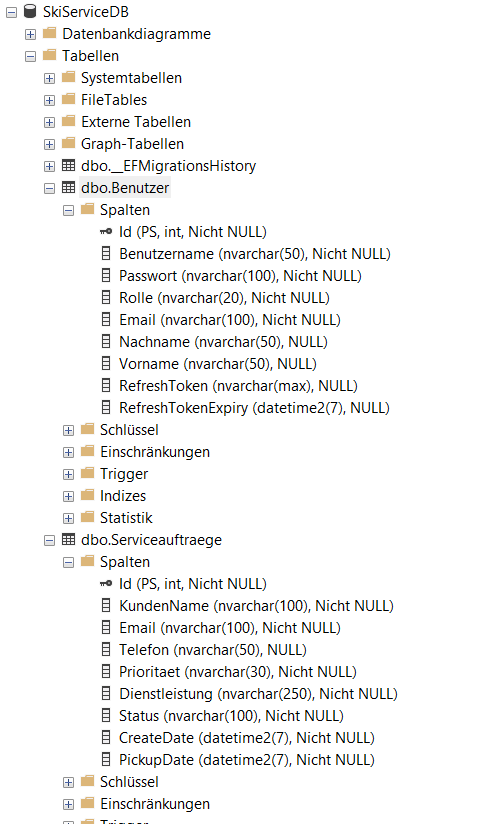
* **Backend:** ASP.NET Core für die API-Entwicklung.
* **Datenbank:** MS-SQL mit Entity Framework als OR-Mapper.
* **Testtools:** Postman für API-Tests und Unit-Tests zur Code-Validierung.
* **Sicherheit:** Passwort-Hashing und eingeschränkte Datenbankzugriffe.

# 2. Planen

## 2.1 Architektur und Technologien

Für die technische Umsetzung wurde folgende Architektur definiert:

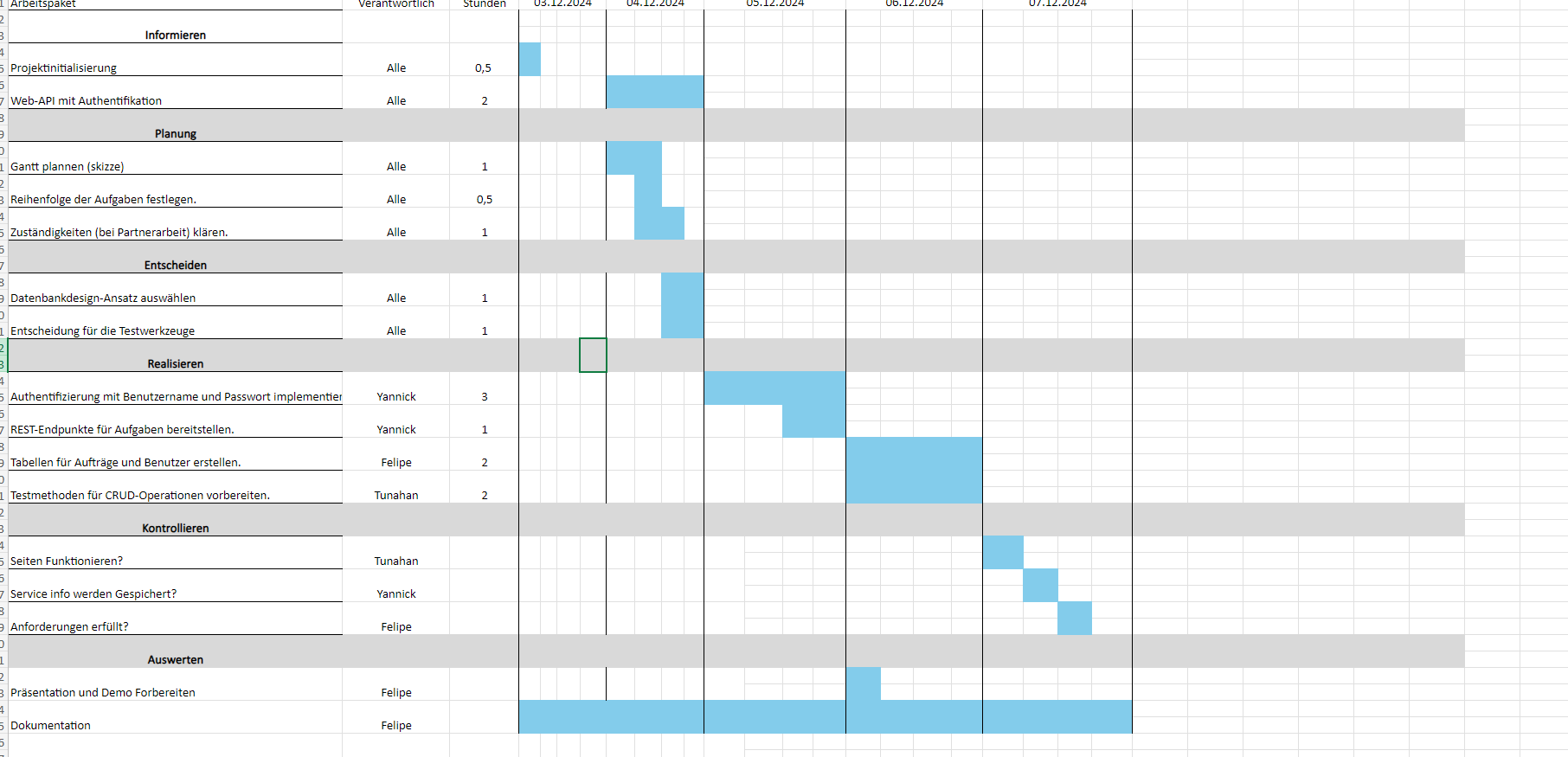
* **Backend:** RESTful API mit ASP.NET Core, implementiert in C#.  
  

**Datenbank:** Normalisierte Tabellenstruktur (3. NF) mit MS-SQL

* **Frontend:** Einfaches, benutzerfreundliches Interface in React.js.

## 2.2 Zeit- und Aufgabenplan

Ein Gantt-Diagramm wurde erstellt, um Aufgaben und Zeitrahmen festzulegen:

* **Phase 1 ( 3-4. Dezember):** Analyse und Planung.
* **Phase 2 ( 4. Dezember):** Entwicklung von Backend und Datenbank.
* **Phase 3 ( 5.Dezember):** Frontend-Integration und API-Verknüpfung.
* **Phase 4 ( 7.Dezember):** Tests, Fehlerbehebung und Dokumentation.

## 2.3 Rollenverteilung

* **Yannick:** Backend-Entwicklung und Datenbankintegration.
* **Tunahan:** Frontend-Design und API-Integration.
* **Felipe:** Dokumentation, Tests und Präsentation.

# 3. Entscheiden

## 3.1 Technologiewahl

* **Backend:** ASP.NET Core für schnelle Entwicklung und einfache Integration.
* **Datenbank:** MS-SQL für Performance und Stabilität.
* **Frontend:** React.js für flexible und interaktive Benutzeroberflächen.
* **Versionskontrolle:** GitHub zur Zusammenarbeit und Nachverfolgbarkeit.

## 3.2 Sicherheitsmaßnahmen

* **Passwort-Hashing:** Sicherstellung durch Bcrypt.
* **Datenbankzugriff:** Einschränkungen auf Benutzer mit minimalen Rechten.
* **Eingabevalidierung:** Um XSS- und SQL-Injection-Angriffe zu vermeiden.

# 4. Realisieren

## 4.1 Backend

* **Authentifizierung:** JWT-basierter Login-Mechanismus.
* **API-Endpoints:**
  + POST /login: Benutzer authentifizieren.
  + GET /aufträge: Auftragsliste anzeigen.
  + PATCH /aufträge/{id}: Status ändern.
  + DELETE /aufträge/{id}: Auftrag löschen.
  + POST /aufträge: Neuer Auftrag.
* **Datenbank:** Normalisierte Tabellenstruktur gemäß der 3. NF.

## 4.2 Frontend

* **Benutzeroberfläche:** Darstellung von Aufträgen mit Filteroptionen und Statusänderungen.
* **Formulare:** Eingabe für Login und neue Aufträge.
* **API-Integration:** Kommunikation mit den Endpunkten des Backends.

## 

## 4.3 Tests und Dokumentation

* **Tests:**
  + Unit-Tests für API-Methoden.
  + Postman für manuelle und automatisierte API-Tests.
* **Protokollierung:** Alle API-Requests und -Fehler wurden in einer Logdatei gespeichert.
* **Dokumentation:** OpenAPI-Dokumentation (Swagger) für die API.

# 5. Kontrollieren

## 5.1 Testberichte

Alle Endpunkte wurden getestet, um sicherzustellen, dass sie den Anforderungen entsprechen:

* **Ergebnisse:**
  + Erfolgreiche Verarbeitung aller CRUD-Operationen.
  + Behebung von Sicherheitslücken (z. B. fehlerhafte Eingabevalidierung).

## 5.2 Einhaltung der Anforderungen

* **Obligatorische Anforderungen:** Vollständig umgesetzt.
* **Optionale Erweiterungen:** Hinzufügen von Kommentaren und personalisierten Listen.

# 6. Auswerten

## 6.1 Feedbackanalyse

Nach der Live-Demo wurde das Projekt durch das Team und externe Teilnehmer bewertet:

* **Stärken:**
  + Klare Struktur der API und Benutzeroberfläche.
  + Sicherheit durch Passwort-Hashing und eingeschränkte Zugriffsrechte.
* **Schwächen:**
  + Frontend-Funktionalitäten könnten weiter ausgebaut werden.

## 6.2 Lessons Learned

* **Was lief gut?**
  + Effektive Zusammenarbeit und klare Kommunikation im Team.
  + Anwendung der IPERKA-Methode zur strukturierten Umsetzung.
* **Was verbessern?**
  + Bessere Zeitplanung, insbesondere für die Frontend-Integration.

## 6.3 Weiterentwicklungsmöglichkeiten

* Ausbau der Funktionalitäten, z. B. durch Berichte oder Statistiken.
* Verbesserung des UI-Designs für eine noch intuitivere Bedienung.

# Fazit

Wir haben gezeigt, wie anspruchsvoll die Entwicklung eines vollständigen Backend-Systems für die Verwaltung von Ski-Service-Aufträgen sein kann. Obwohl wir große Fortschritte gemacht haben, konnten wir nicht alle geplanten Anforderungen vollständig umsetzen. Dennoch bot das Projekt uns wertvolle Lernmöglichkeiten und zeigte das Potenzial unserer Zusammenarbeit und Methodik auf.

**Erreichte Ergebnisse:**

* Wir haben eine funktionierende Grundstruktur der RESTful API und Datenbankintegration erstellt.
* Wir konnten Sicherheitsmaßnahmen wie Passwort-Hashing und Eingabevalidierung erfolgreich implementieren.
* Mit ersten Prototypen für das Frontend haben wir grundlegende Interaktionen ermöglicht.

**Herausforderungen:**

* Einige Funktionalitäten, wie eine umfassendere Frontend-Integration und zusätzliche Features, konnten wir aufgrund von Zeitmangel nicht fertigstellen.
* Unsere ursprüngliche Zeitplanung erwies sich als zu knapp bemessen, insbesondere für die Abstimmung zwischen Backend- und Frontend-Entwicklung.

**Erkenntnisse und Ausblick:**

* **Stärken:** Wir haben eine klare Struktur der API geschaffen und Sicherheitsmechanismen fundiert angewendet.
* **Verbesserungspotenziale:** Eine realistischere Zeitplanung und frühere Priorisierung der kritischen Funktionalitäten hätten uns helfen können.
* **Zukunftsmöglichkeiten:** Unser System bietet eine solide Grundlage, die wir durch eine vollständige Frontend-Integration und zusätzliche Features wie Berichte, Statistiken oder ein verbessertes UI weiter ausbauen können.

Insgesamt war das Projekt für uns ein wichtiger Schritt, um Erfahrungen in der Softwareentwicklung zu sammeln. Trotz der nicht erreichten Ziele konnten wir wichtige Erkenntnisse gewinnen, die in zukünftige Projekte einfließen werden.