ipso! Bildung seit 1876

# JetStream backend

By david Reichert

ipso! Bildung seit 1876

# Inhaltsverzeichnis

# 1 Versionsverzeichnis

| Version | Autor | Datum | Änderung |
|---|---|---|---|
| 1.0.0 | David Reichert | 12.11.2023 | Basis Erstellung des Dokuments + erste ein träge |
| 1.1.0 | David Reichert | 15-19.11.2023 | Weiterführung der Dokumentation |
| 2.2.0 | David Reichert | 20-21.11.2023 | Fertigstellung und Rechtschreibung überprüfen |

# 2 Ausgangslage

Die Firma Jetstream-Service, ein kleines bis mittleres Unternehmen (KMU), spezialisiert sich während der Wintersaison auf Skiservicearbeiten. Im Rahmen ihrer Digitalisierungsbestrebungen plant das Unternehmen, die Verwaltung dieser Ski-Serviceaufträge vollständig über eine webbasierte Anwendung, die auch eine Datenbankanbindung hat, abzuwickeln. Ziel ist es, den bereits vorhandenen Online-Anmeldeprozess beizubehalten und ihn um zusätzliche Funktionen für ein effizientes Auftragsmanagement zu erweitern.

In der geschäftigen Hauptsaison arbeiten bis zu zehn Mitarbeiter an der Durchführung dieser Servicearbeiten. Für diese Mitarbeiter soll ein sicherer, passwortgeschützter Zugang zu einem System eingerichtet werden, über das sie Zugriff auf die anstehenden Aufträge erhalten. Sie sollen in der Lage sein, diese Aufträge nicht nur zu übernehmen, sondern auch deren Status zu aktualisieren, um eine effiziente und reibungslose Abwicklung zu gewährleisten. Dadurch wird der Arbeitsprozess nicht nur digitalisiert, sondern auch vereinfacht und effizienter gestaltet.

# 3 Anforderungen

## 3.1 Notwendige

Das Auftragsmanagementsystem der Firma Jetstream-Service, welches sich auf die Verwaltung von Skiservicearbeiten konzentriert, muss spezifische Funktionen bieten, um die Arbeitsabläufe zu optimieren:

1. Login-Funktionalität: Mitarbeiter müssen sich mit einem Benutzernamen und Passwort anmelden können. Dies gewährleistet Sicherheit und personalisierten Zugriff auf die Systemdaten.

2. Anzeige anstehender Serviceaufträge: Das System sollte eine übersichtliche Liste aller anstehenden Serviceaufträge bieten. So können Mitarbeiter auf einen Blick sehen, welche Arbeiten anstehen.

3. Mutation bestehender Serviceaufträge: Mitarbeiter müssen in der Lage sein, den Status der Serviceaufträge zu ändern. Hierfür stehen verschiedene Stati zur Verfügung:

   3.1. Offen: Der Auftrag wurde noch nicht begonnen.
   3.2. In Arbeit: Der Auftrag wird derzeit bearbeitet.
   3.3. Abgeschlossen: Der Auftrag wurde fertiggestellt.

4. Löschen von Aufträgen: Es muss möglich sein, Aufträge zu löschen, beispielsweise bei einer Stornierung durch den Kunden.

Zusätzlich müssen die Informationen zur Online-Anmeldung, die bereits implementiert ist, bei Bedarf erweitert werden, um folgende Details zu erfassen:

- Kundenname

- E-Mail-Adresse
- Telefonnummer
- Priorität des Auftrags
- Gewählte Dienstleistung: Hierbei ist zu beachten, dass pro Serviceauftrag nur eine Dienstleistung zugeordnet werden kann.
-

Das Dienstleistungsangebot der Firma umfasst:

- Kleiner Service: Grundlegende Wartungsarbeiten.
- Großer Service: Umfassende Überholung und Wartung.
- Rennski-Service: Spezialisierte Serviceleistungen für Rennskier.
- Bindung montieren und einstellen: Anpassung und Einstellung der Skibindungen.
- Fell zuschneiden: Zurechtschneiden von Skifellen.
- Heisswachsen: Anwendung eines Wachsverfahrens zur Optimierung der Skigleiteigenschaften.

Diese Funktionen und Dienstleistungen sorgen zusammen für ein effizientes und kundenfreundliches Management der Skiserviceaufträge.

# 4 Informieren

Informationsquellen
Die für dieses Projekt relevanten Informationen wurden aus einer Vielzahl von Quellen bezogen:

1. Unterrichtssitzungen: Während der Unterrichtseinheiten wurden grundlegende Konzepte und Methoden des Software-Engineerings sowie Best Practices für die Entwicklung von Web-Anwendungen und Datenbanksystemen erörtert. Diese Inhalte sind essentiell für das Verständnis der technischen Aspekte des Projekts.

2. OneNote des Dozenten: Das vom Dozenten bereitgestellte OneNote enthielt detaillierte Anleitungen, Beispiele und zusätzliche Ressourcen. Diese Informationen waren insbesondere hilfreich für die tiefere Eintauchung in spezifische Themen wie Authentifizierungssysteme, Datenbankdesign und API-Entwicklung.

3. Projekt-Dokument: Das Dokument "PA-SkiService-Management" lieferte eine umfassende Projektbeschreibung, darunter die Ausgangssituation des Unternehmens, die technischen Anforderungen für das System, die gewünschten Funktionen und Dienstleistungen sowie die Rahmenbedingungen für die Implementierung.

## 4.1 Technische Anforderungen

Die technischen Spezifikationen des Projekts umfassen:

- Entwicklung eines Login-Systems: Sicherer Zugriff für Mitarbeiter mittels Benutzername und Passwort.

- Listendarstellung anstehender Serviceaufträge: Ein System, das es den Mitarbeitern ermöglicht, anstehende Aufträge effizient einzusehen und zu verwalten.

- Mutation bestehender Serviceaufträge: Möglichkeit zur Aktualisierung des Status von Aufträgen (Offen, In Arbeit, Abgeschlossen) sowie deren Löschung.

- Integration der Online-Anmeldung: Ergänzung von Kundeninformationen und Dienstleistungsauswahl in das bestehende System.

Erwartete Nutzervorteile
Die Digitalisierung des Ski-Service-Management soll folgende Vorteile bringen:

- Verbesserte Effizienz: Schnellere Bearbeitung von Serviceaufträgen durch digitale Verwaltung.
- Höhere Transparenz: Klare Übersicht über den Status aller Aufträge für die Mitarbeiter.
- Benutzerfreundlichkeit: Einfacher Zugang und Bedienbarkeit des Systems für alle Beteiligten.

Durch die Kombination der Informationen aus dem Unterricht, dem OneNote des Dozenten und dem Projekt-Dokument wurde ein solides Verständnis für die Anforderungen und Ziele des Projekts entwickelt, welches als Basis für die weiteren Schritte im IPERKA-Prozess dient.

# 5 Planung

## 5.1 1. Meilensteine definieren

### 5.1.1 a. Erstellung der Modellklassen und des Kontexts

- **Relevante Dateien:**
  - **Models/ServiceAuftrag.cs**, **Models/Mitarbeiter.cs**, etc.: Definition der Datenmodelle.
  - **Models/SkiServiceContext.cs**: Kontextklasse für die Datenbankverbindung.

- **Ziele:**
  - Entwurf und Implementierung der Datenmodelle.
  - Einrichtung des Datenbankkontexts.

### 5.1.2 b. Datenbankmigration

- **Relevante Dateien:**
  - **Migrations/**: Migrationsskripte für die Datenbank.
- **Ziele:**
  - Durchführung der Datenbankmigration.
  - Sicherstellung der korrekten Datenbankstruktur.

### 5.1.3 c. Erstellung von Controllern und Serviceklassen

- **Relevante Dateien:**
  - **Controllers/ServiceAuftragController.cs**, **Controllers/mitarbeiterController.cs**, etc.
  - **Service/ServiceAuftragService.cs**, **Service/mitarbeiterService.cs**, etc.
- **Ziele:**
  - Implementierung der Geschäftslogik in den Serviceklassen.
  - Erstellung der Controller für die API-Endpunkte.

### 5.1.4 d. Frontend-Integration

- **Relevante Dateien:**
  - **wwwroot/js/**, **wwwroot/html/**, etc.: Frontend-Dateien.
- **Ziele:**
  - Anbindung des Frontends an die Backend-API.
  - Implementierung der Benutzeroberfläche.

### 5.1.5 e. Implementierung des Login-Systems mit JWT

- **Relevante Dateien:**
  - **Controllers/AuthController.cs**
  - **wwwroot/js/login.js**

- **Ziele:**
  - Einrichtung der Authentifizierung mit JWT.
  - Integration des Login-Systems in das Frontend und Backend.

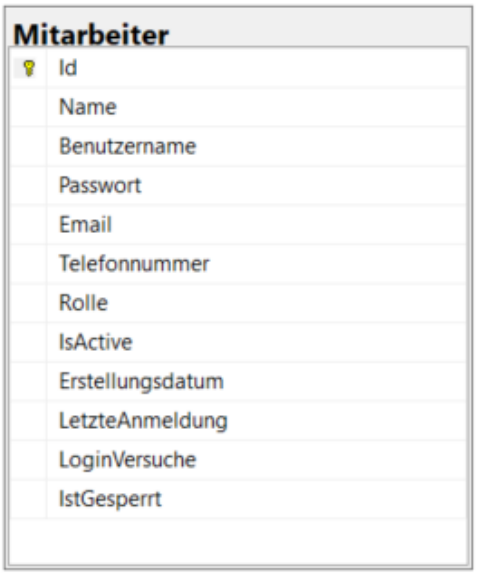
### 5.1.6 Einrichtung von HTTPS und Sicherheitsfeatures

- **Relevante Dateien:**
  - Konfigurationsdateien wie **appsettings.json**
- **Ziele:**
  - Konfiguration von HTTPS.
  - Implementierung zusätzlicher Sicherheitsmaßnahmen.

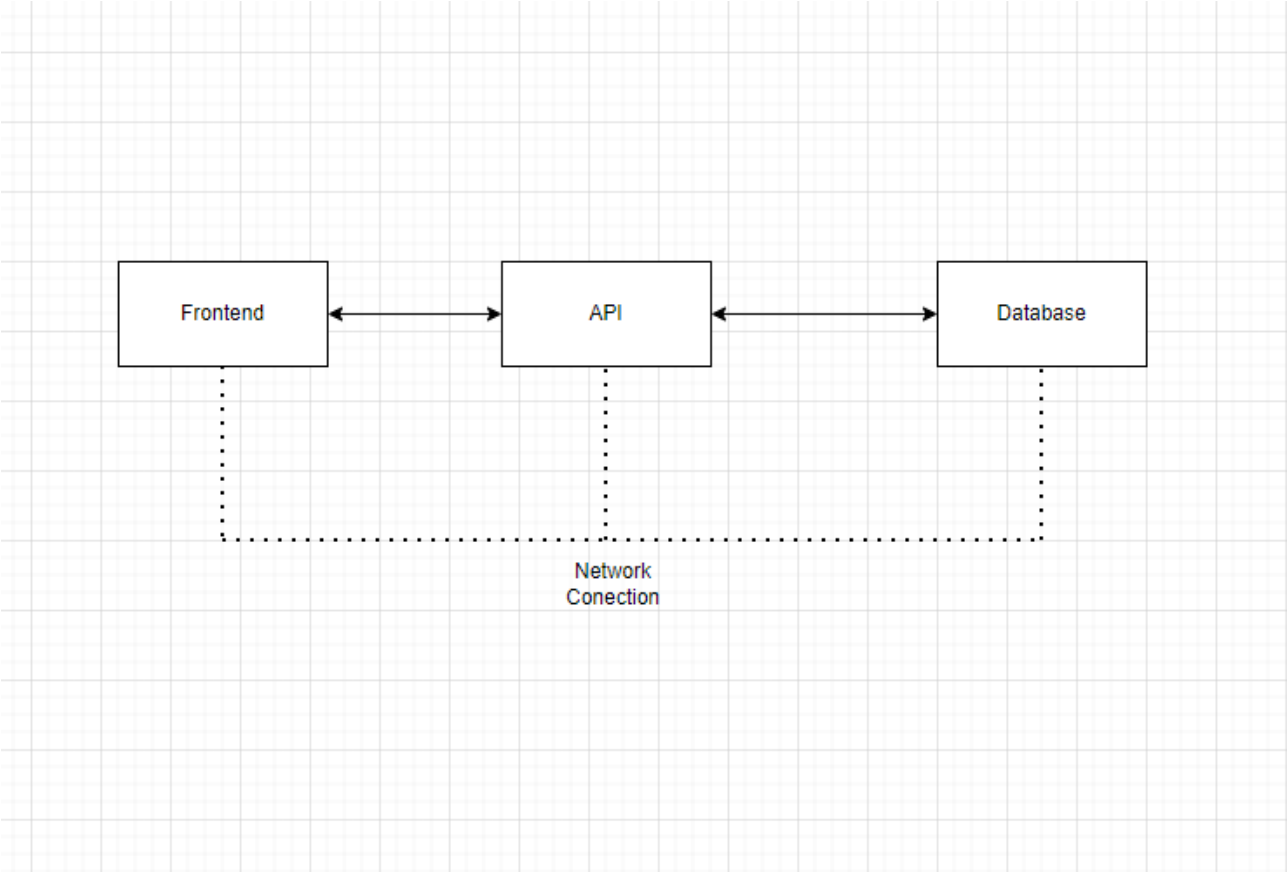
## 5.2 Ressourcenplanung

- **Entwickler:** Zuweisung von Aufgaben basierend auf Fachkenntnissen und neu gelernten Kenntnissen (Backend, Frontend, Sicherheit).
- **Testwerkzeuge:** Planung von Swagger oder Postman Tests.
- **Serverkapazitäten:** Bereitstellung von Entwicklungsservern und Produktionsumgebungen.

## 5.3 Datenbank Diagramm

### 5.4 System Architektur




## 6 Entscheidungen treffen

### 6.1 a. Technologieauswahl

#### 6.1.1 Datenbankansatz: Code First

Sie haben sich für den "Code First"-Ansatz entschieden, was bedeutet, dass Sie die Datenbankstruktur direkt aus dem Code heraus generieren. Dies bietet Flexibilität und eine enge Kopplung zwischen der Datenbank und Ihrem Code.

#### 6.1.2 Testwerkzeug: Postman

Für das Testen der API haben Sie Postman gewählt. Dies ermöglicht eine einfache Authentifizierung und das Testen verschiedener Endpunkte Ihrer API.

#### 6.1.3 Entwicklungsumgebung: Visual Studio

Als Entwicklungsumgebung dient Visual Studio.

# 7 Realisierung des Projekts "JetStreamBackend"

## 7.1 Initialisierung und Projektstruktur

Das Projekt beginnt mit der Einrichtung eines Git-Repositorys, um eine Versionskontrolle und eine kollaborative Arbeitsumgebung zu ermöglichen.
In Visual Studio wird die Grundstruktur des Projekts erstellt, einschließlich der notwendigen Konfigurationsdateien und Verzeichnisse für Modelle, Controller, Services und Views.
Eine README.md-Datei wird angelegt, um eine Übersicht über das Projekt, seine Ziele und die verwendeten Technologien zu bieten.

## 7.2 Datenmodellierung und Datenbankintegration

Unter Verwendung des Code First-Ansatzes werden die Datenmodelle für die Kernfunktionen der Anwendung entwickelt. Dies umfasst Klassen wie ServiceAuftrag und Mitarbeiter, die die Geschäftslogik und die Datenstruktur definieren.
Der SkiServiceContext wird als zentraler Datenbankkontext implementiert, der die Verbindung zur Datenbank und die Konfiguration der Entity Framework Core-Features handhabt.
Durch die Durchführung von Datenbankmigrationen wird die Datenbankstruktur basierend auf den definierten Modellen erstellt. Dies ermöglicht eine agile Entwicklung und Anpassung der Datenbank an sich ändernde Anforderungen.

## 7.3 Backend-Entwicklung und API-Implementierung

Die Geschäftslogik des Projekts wird in Serviceklassen wie ServiceAuftragService und mitarbeiterService implementiert. Diese Klassen kapseln die Geschäftsregeln und Interaktionen mit der Datenbank.
Controller wie ServiceAuftragController und mitarbeiterController werden entwickelt, um die API-Endpunkte zu definieren. Diese Controller verarbeiten Anfragen von Clients, rufen die entsprechenden Services auf und geben die Ergebnisse zurück.
Für das Testen der API-Endpunkte wird Postman verwendet. Dies ermöglicht eine einfache Authentifizierung und das Testen verschiedener Szenarien, um sicherzustellen, dass die API wie erwartet funktioniert.

## 7.4 Frontend-Entwicklung und Integration

Das Frontend der Anwendung wird unter Verwendung von HTML, CSS und JavaScript entwickelt. Die Struktur und das Design der Benutzeroberfläche werden erstellt, um eine benutzerfreundliche und ansprechende Erfahrung zu bieten.
JavaScript wird verwendet, um die Interaktion mit der Backend-API zu ermöglichen. Dies umfasst das Senden von Anfragen an die API und das Empfangen von Antworten, die dann im Frontend dargestellt werden.
Die Integration des Frontends mit dem Backend ist ein kritischer Schritt, um eine voll funktionsfähige Anwendung zu schaffen. Dies beinhaltet die Verbindung der Benutzeroberfläche mit den entsprechenden API-Endpunkten und die Sicherstellung, dass Daten korrekt zwischen Frontend und Backend ausgetauscht werden.

## 7.5 Implementierung des Authentifizierungssystems

Ein zentrales Feature des Projekts ist das Authentifizierungssystem, das unter Verwendung von JSON Web Tokens (JWT) implementiert wird. Dies bietet eine sichere Methode zur Verwaltung von Benutzersitzungen und zum Schutz von sensiblen Bereichen der Anwendung.
Der AuthController wird entwickelt, um die Logik für die Benutzeranmeldung, Token-Erzeugung und -Validierung zu handhaben.
Die Integration des Authentifizierungssystems in das Frontend und Backend stellt sicher, dass nur authentifizierte Benutzer Zugriff auf bestimmte Funktionen und Daten haben.

## 7.6 Sicherheitsmaßnahmen und HTTPS-Konfiguration

Um die Sicherheit der Anwendung zu gewährleisten, werden verschiedene Sicherheitsbest Practices implementiert. Dazu gehören die Verschlüsselung von Benutzerdaten, die sichere Handhabung von Authentifizierungstokens und die Implementierung von Sicherheitsmechanismen gegen gängige Angriffsvektoren wie SQL-Injection und Cross-Site Scripting (XSS).
HTTPS wird konfiguriert, um eine verschlüsselte Verbindung zwischen dem Client und dem Server zu gewährleisten. Dies ist entscheidend, um die Integrität und Vertraulichkeit der übertragenen Daten zu schützen.

### 7.7 Tests, Überprüfung und Deployment

Umfassende Tests werden durchgeführt, um die Funktionalität und Zuverlässigkeit der Anwendung zu gewährleisten. Dies umfasst Unit-Tests für einzelne Komponenten und Integrationstests, um das Zusammenspiel der verschiedenen Teile der Anwendung zu überprüfen.
Eine gründliche Überprüfung der Anwendung wird durchgeführt, um sicherzustellen, dass alle Anforderungen erfüllt sind und die Anwendung bereit für den Einsatz ist.
Das Deployment der Anwendung auf einem Produktions-Server markiert den Abschluss der Entwicklungsphase. End-to-End-Tests in der Produktionsumgebung stellen sicher, dass die Anwendung unter realen Bedingungen wie er

## 8 Kontrolle und Qualitätssicherung mit Swagger und Postman

### 8.1 Verwendung von Swagger

### 8.2 API-Dokumentation und Interaktive Tests:

Swagger wird eingesetzt, um eine automatisierte Dokumentation Ihrer API-Endpunkte zu erstellen. Diese Dokumentation bietet eine klare Übersicht über die verfügbaren API-Methoden, ihre Parameter und die erwarteten Antwortformate.
Über die Swagger UI können Sie interaktive Tests durchführen. Sie können API-Anfragen direkt in der Benutzeroberfläche ausführen und die Antworten überprüfen. Dies erleichtert das Testen und Debuggen der API während der Entwicklung.

### 8.3 Verwendung von Postman

### 8.4 Manuelle und Automatisierte API-Tests:

Postman wird als leistungsfähiges Tool für das manuelle Testen der API verwendet. Sie können individuelle Anfragen an Ihre API senden, Authentifizierungsdetails einfügen und die Antworten analysieren.
Postman ermöglicht auch das Erstellen von automatisierten Testskripten. Diese Skripte können verwendet werden, um eine Reihe von Anfragen automatisch auszuführen und die

Antworten zu validieren. Dies ist besonders nützlich für Regressionstests und die Überprüfung der API-Funktionalität nach Änderungen am Code.$

# 9 Auswertung und Fazit des Projekts "JetStreamBackend"

## 9.1 Persönliche Reflexion

- **Erfahrung und Lernerfolge:**
  - Die Arbeit an dem Projekt "JetStreamBackend" war für mich eine äußerst bereichernde Erfahrung. Es bot mir die Gelegenheit, meine Fähigkeiten in der Softwareentwicklung zu vertiefen und in der Praxis anzuwenden.
  - Während des Projekts konnte ich mein Wissen in verschiedenen Bereichen erweitern, insbesondere in der Backend-Entwicklung, der API-Gestaltung und der Implementierung von Sicherheitsmaßnahmen wie JWT-basierter Authentifizierung.

## 9.2 Neue Kenntnisse und Fähigkeiten

- **Technologische Erkenntnisse:**
  - Durch die Nutzung von Tools wie Swagger und Postman habe ich wertvolle Einblicke in die effiziente Gestaltung und das Testen von APIs gewonnen. Diese Tools haben sich als unverzichtbar für die Überprüfung der Funktionalität und Leistung der entwickelten API erwiesen.
  - Die Erfahrung mit Performance-Tests und -Optimierung hat mein Verständnis für die Wichtigkeit von Systemeffizienz und Benutzererfahrung vertieft.

## 9.3 Spaß und Motivation

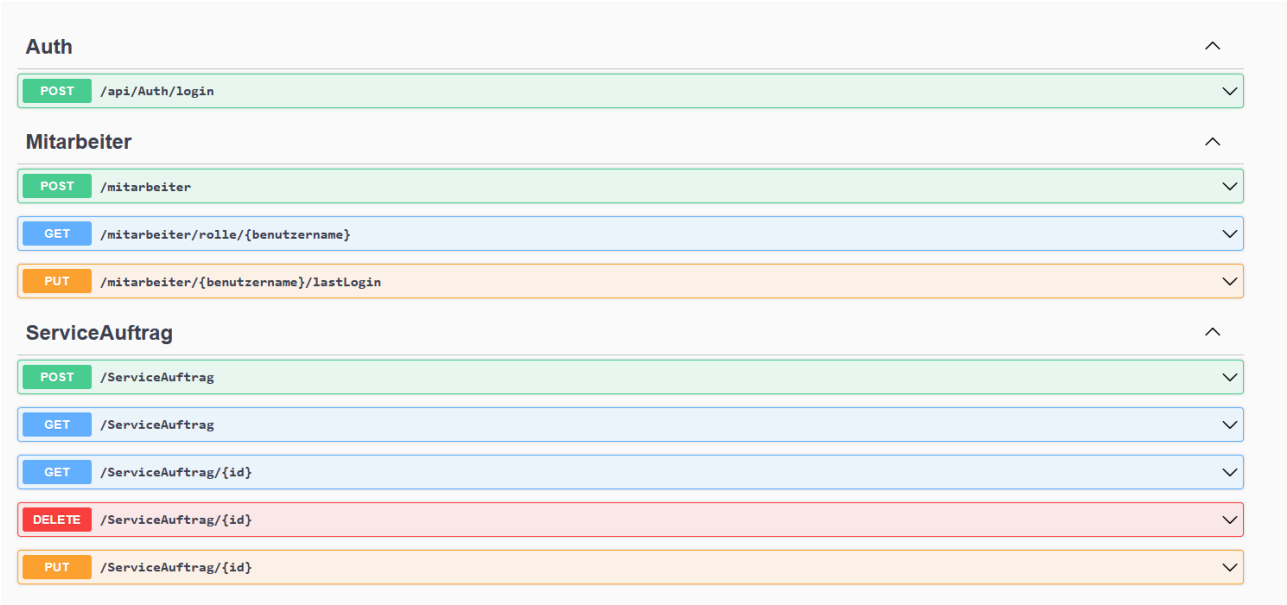
- **Persönliche Zufriedenheit:**
  - Das Projekt war nicht nur eine Lerngelegenheit, sondern auch eine Quelle großer persönlicher Zufriedenheit. Die Herausforderungen, die während der Entwicklung auftraten, waren motivierend und haben das Gefühl des Erfolgs bei der Lösung von Problemen verstärkt.
  - Die erfolgreiche Umsetzung des Projekts und das Sehen der funktionierenden Anwendung war eine sehr befriedigende Erfahrung.

## 9.4 Fazit

- **Zusammenfassung und Ausblick:**
  - Insgesamt war das Projekt "JetStreamBackend" eine wertvolle und lohnende Erfahrung. Es hat nicht nur meine technischen Fähigkeiten verbessert, sondern auch mein Verständnis für den gesamten Entwicklungsprozess vertieft.
  - Die im Projekt gesammelten Erfahrungen und Kenntnisse werden eine solide Grundlage für meine zukünftigen Projekte bilden und mich in meiner Karriere als Softwareentwickler weiterbringen.

# 10 Anhänge

## 10.1 Swagger




## 10.2 NuGet Packete

**microsoft.entityframeworkcore\7.0.14**

**microsoft.entityframeworkcore.sqlserver\7.0.14**

**serilog.extensions.logging\7.0.0**

**swashbuckle.aspnetcore\6.5.0**

**serilog.sinks.file\5.0.0**

**microsoft.aspnetcore.authentication.jwtbearer\7.0.14**

**microsoft.entityframeworkcore.tools\7.0.14**

**serilog\3.1.1**

**serilog.aspnetcore\7.0.0**

| Abkürzung | Vollständige Bezeichnung | Erklärung |
|---|---|---|
| KMU | Kleines bis Mittleres Unternehmen | Bezieht sich auf Unternehmen mit einer begrenzten Anzahl an Mitarbeitern und Umsatz, kleiner als große Konzerne. |
| JWT | JSON Web Token | Ein kompakter und selbstenthaltener Weg für die sichere Übertragung von Informationen zwischen Parteien als JSON-Objekt. Wird häufig für Authentifizierungszwecke verwendet. |
| API | Application Programming Interface | Eine Schnittstelle, die es verschiedenen Softwareanwendungen ermöglicht, miteinander zu kommunizieren. |
| HTTPS | Hypertext Transfer Protocol Secure | Eine Erweiterung des HTTP-Protokolls mit zusätzlicher Sicherheit durch Verschlüsselung, häufig für sichere Internettransaktionen verwendet. |
| SQL | Structured Query Language | Eine Programmiersprache, die speziell für die Verwaltung und Abfrage von Daten in relationalen Datenbanken entwickelt wurde. |
| XSS | Cross-Site Scripting | Ein Sicherheitsrisiko bei Webanwendungen, bei dem Angreifer schädlichen Code in Webseiten einfügen können. |
| CSS | Cascading Style Sheets | Eine Stylesheet-Sprache, die zur Beschreibung des Aussehens und der Formatierung von Dokumenten verwendet wird, die in einer Markup-Sprache wie HTML geschrieben sind. |
| HTML | Hypertext Markup Language | Die Standardmarkup-Sprache für Dokumente, die im World Wide Web angezeigt werden. |
| JSON | JavaScript Object Notation | Ein kompaktes Datenformat in leicht lesbarem Text, das für den Datenaustausch zwischen Anwendungen verwendet wird. |
| UI | User Interface | Die Schnittstelle, durch die ein Benutzer mit einem digitalen Gerät oder einer Anwendung interagiert. |