BACKEND - WEB API

Für die Jetstream-Service AG

Dokumentation zur Projektarbeit
ICT Modul 295

Abgabedatum: Ottenbach, 21.08.2023

Prüfungsbewerber:

Mahir Gönen

Weihermatt 1
8913 Ottenbach

21.11.2023

Inhalt

1 Versionsverzeichnis

Version	Autor	Datum	Änderung
1.0	Mahir Gönen	13.11.2023	Erstellung des Dokuments
1.1	Mahir Gönen	18.11.2023	Ergänzung des Dokuments
1.2	Mahir Gönen	21.11.2023	Rechtschreibung und Grammatik Kontrolle

2 Executive Summary

Jetstream-Service, ein spezialisiertes KMU für Skiservice-Arbeiten, steht vor der Herausforderung, ihre veraltete Webpräsenz zu modernisieren und zu digitalisieren. Ziel dieses Projekts ist die Entwicklung eines robusten und effizienten Backends für die Online-Auftragsverwaltung im Bereich Skiservice.

Das Backend-System soll die bestehende Online-Anmeldung für Serviceaufträge ergänzen und optimieren. Es wird eine umfassende Web-API mit Authentifizierung implementiert, die es ermöglicht, Serviceaufträge zu verwalten, zu mutieren und den Status zu überwachen. Die Anwendung wird eine sichere und benutzerfreundliche Schnittstelle bieten, die den Mitarbeitern den Zugriff auf und die Bearbeitung von Aufträgen ermöglicht.

Die Datenbank, die mit MS-SQL realisiert wird, wird nach modernen Prinzipien des Datenbankdesigns gestaltet. Dabei kommt die Methode Code First zum Einsatz, um eine effiziente und flexible Datenverwaltung zu gewährleisten.

Die Realisierung des Projekts erfolgt unter Einhaltung der vorgegebenen Fristen und Milestones. Eine gründliche Testphase, einschließlich der Verwendung von Unit-Tests und Postman, stellt sicher, dass die Anwendung stabil und fehlerfrei funktioniert.

Dieses Backend-Projekt ist ein wesentlicher Schritt in der digitalen Transformation von Jetstream-Service und wird die Effizienz der internen Prozesse erheblich steigern. Es bietet nicht nur eine verbesserte Benutzererfahrung für die Kunden, sondern auch ein optimiertes Managementwerkzeug für die Mitarbeiter. Dieses Executive Summary vermittelt Stakeholdern und Entscheidungsträgern einen klaren Überblick über die Ziele und den Umfang des Backend-Entwicklungsprojekts.

3 Ausgangslage

Bei Jetstream-Service wird derzeit ein Backend-System auf Basis von Node.js verwendet, dass die Verwaltung von Serviceaufträgen über eine JSON-basierte Datenhaltung ermöglicht. Diese Struktur weist jedoch erhebliche Einschränkungen auf: Die Daten sind nur in einem JSON-Format zugänglich und gespeichert, was die Flexibilität und Skalierbarkeit der Datenverwaltung begrenzt. Zudem fehlt eine Authentifizierung, was in Anbetracht der aktuellen Sicherheitsstandards und Datenschutzanforderungen als veraltet betrachtet werden muss. Diese Einschränkungen führen zu einer komplizierten Handhabung der eingegangenen Bestellungen für die Mitarbeiter.

Die Notwendigkeit eines modernisierten und effizienteren Backends ist daher zwingend. Das geplante neue Backend-System soll eine direkte Anbindung an die Webseite ermöglichen, wodurch die Mitarbeiter nach dem Login in einen geschützten Bereich gelangen und alle Serviceaufträge effizient verwalten können. Hierbei wird eine verbesserte Verwaltung durch Authentifizierungsschritte beim Ändern und Löschen eines Auftrages eingeführt. Insbesondere beim Löschen von Aufträgen wird ein Ansatz verfolgt, bei dem Aufträge nicht vollständig gelöscht, sondern lediglich als storniert markiert werden. Dies verhindert, dass Mitarbeiter unnötige Daten erhalten und ermöglicht es gleichzeitig, die stornierten Daten weiterhin in der Datenbank zu speichern. Diese Massnahme trägt dazu bei, dass nur relevante und aktuelle Daten für die tägliche Arbeit abgerufen werden, während stornierte Informationen für historische oder analytische Zwecke erhalten bleiben.

Die Implementierung des neuen Backends ist somit ein wesentlicher Teil der digitalen Transformation von Jetstream-Service, der darauf abzielt, die Effizienz und Sicherheit der internen Prozesse zu steigern und gleichzeitig eine verbesserte Benutzererfahrung für die Mitarbeiter zu bieten.

4 Anforderungen

Das Backend-System für das Projekt "Jetstreamski-Service Management" muss eine Reihe von spezifischen Anforderungen erfüllen, um eine effiziente und sichere Verwaltung der Skiservice-Aufträge zu gewährleisten:

- **Login-Funktionalität:** Implementierung eines Login-Dialogs mit Passwort für den autorisierten Zugang der Mitarbeiter. Nur autorisierte Mitarbeiter dürfen Datenänderungen vornehmen.
- **Datenbankmanagement:** In der Datenbank müssen Informationen des Serviceauftrags und die Login-Daten der Mitarbeiter verwaltet werden. Die Struktur der Datenbank muss in der 3. Normalform (3NF) sein, inklusive referenzieller Integrität.
- **Serviceaufträge Management:** Erfassung und Abrufbarkeit von Serviceaufträgen. Mitarbeiter können eine Statusänderung eines Auftrages vornehmen (Offen, In Arbeit, Abgeschlossen) und Aufträge löschen (z.B. bei Stornierungen).
- **API-Endpoints Protokollierung:** Die aufgerufenen API-Endpoints müssen zur Fehlerlokalisierung protokolliert sein, entweder in einer Datenbank oder in einer Protokolldatei.
- **Sicherheitsaspekte:** Für die Web-API Applikation muss ein eigener Datenbankbenutzerzugang mit eingeschränkter Berechtigung (nur DML-Befehle) zur Verfügung gestellt werden.
- **Dokumentation:** Das Web-API muss vollständig nach Open-API (Swagger) dokumentiert sein.
- **Projektmanagement:** Das gesamte Projektmanagement muss nach der IPERKA-Methode dokumentiert sein und das Softwareprojekt ist über ein Git-Repository zu verwalten.
- **Technologische Bedingungen:** Als Datenbanksystem ist MS-SQL zu verwenden und Postman soll als Web-API Test-Tool eingesetzt werden. Der Datenbankzugriff hat über einen OR-Mapper zu erfolgen.
- **Optionale Erweiterungen:** Es gibt zusätzliche optionale Anforderungen wie die Möglichkeit für Mitarbeiter, Kommentare zu Aufträgen hinzuzufügen, Aufträge mit sämtlichen Datenfeldern zu ändern, und ein automatisches Sperrsystem für das Login nach drei aufeinanderfolgenden Falschanmeldungen.

5 Zeitplanung

Das Projekt wird in mehreren Phasen durchgeführt und es wurden folgender Zeitplan festgelegt:

Projektphase	Geplante Zeit
Informieren	2h
Planung, Entwurf, Entscheidung	4h
Realisierung	15h
Abnahme	4h
Dokumentation	10h
Gesamt	35h

Tabelle 1: Grobe Planung

Dieses Projekt findet innerhalb von zwei Wochen statt.

Ein Projektstrukturplan (PSP) ist ein essenzielles Werkzeug in der Projektmanagement-Praxis. Er dient dazu, die Struktur und den Umfang eines Projekts zu definieren und zu visualisieren. Der PSP zerlegt das gesamte Projekt in kleinere, handhabbare Teile oder Arbeitspakete. Dies erleichterten die Planung, Überwachung und Steuerung des Projekts.

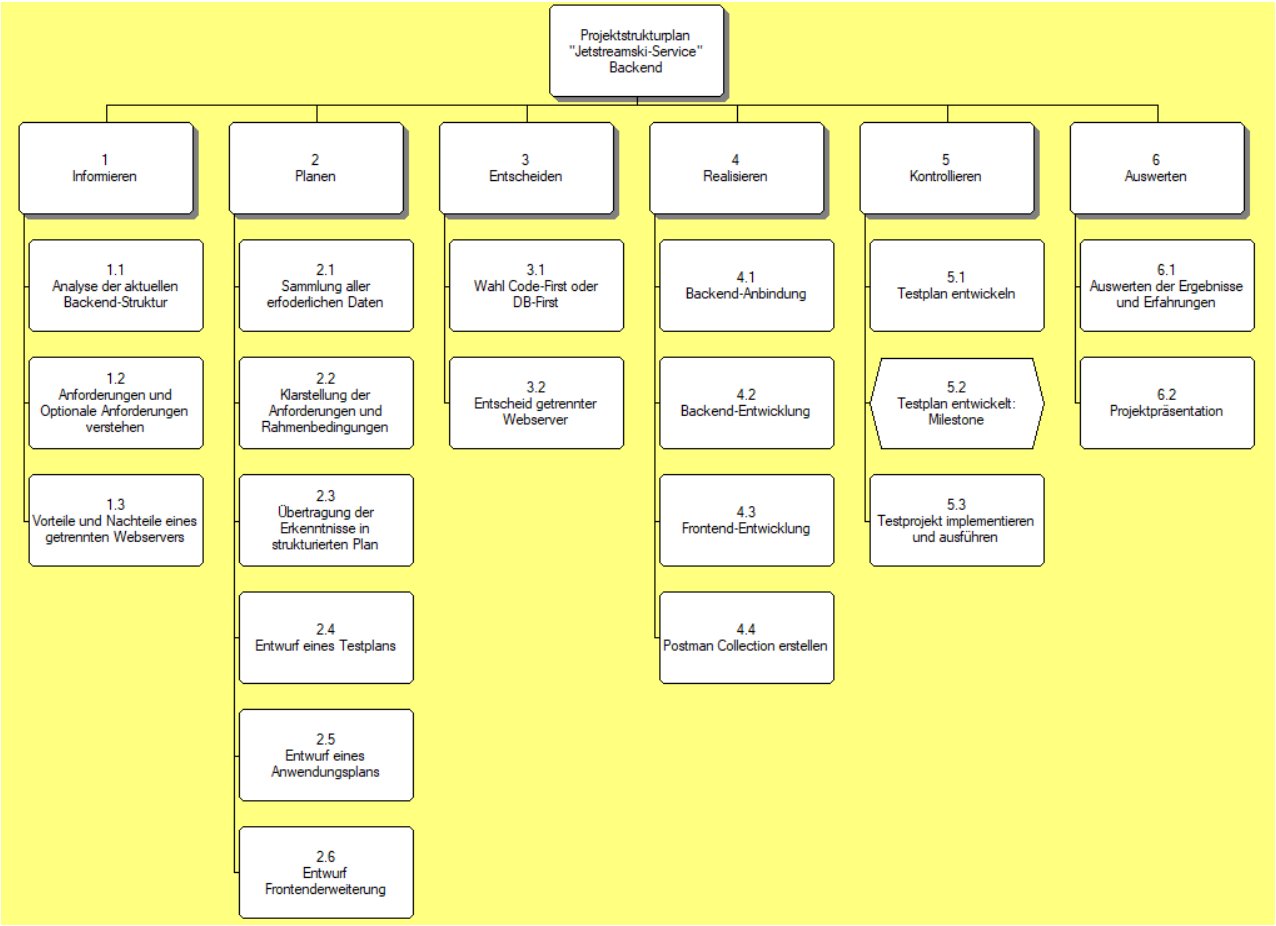
In Bezug auf die Ressourcenplanung ist der PSP besonders wertvoll, da er hilft, die benötigten Ressourcen für jedes Arbeitspaket zu identifizieren und zuzuweisen. Dies umfasst sowohl materielle Ressourcen als auch personelle Ressourcen.

Durch die Erstellung eines PSP kann folgendes entnommen werden:

- Den Umfang des Projekts klar definieren und verstehen.
- Die benötigten Ressourcen für jedes Arbeitspaket identifizieren und planen.
- Die Zuweisung von Ressourcen und die Fortschrittsverfolgung für jedes Arbeitspaket überwachen.
- Potenzielle Risiken und Probleme identifizieren und entsprechende Massnahmen ergreifen.

Insgesamt trägt der PSP dazu bei, das Projekt effizient und erfolgreich zu managen und sicherzustellen, dass alle Ressourcen optimal genutzt werden.

5.1 Projektstrukturplan



1 Projektstrukturplan

6 Informieren

In dieser Phase des IPERKA-Modells liegt der Schwerpunkt auf der gründlichen Informationsbeschaffung. Ziel ist es, ein detailliertes Verständnis für die spezifischen Bedürfnisse, Anforderungen und Herausforderungen bei der Entwicklung des Backend-Systems für Jetstream-Service zu erlangen. Diese Phase beinhaltet die Analyse der gegenwärtigen Backend-Struktur, das Erkennen der Bedürfnisse der Nutzer und Mitarbeiter, das Verständnis für die technischen Anforderungen sowie die Berücksichtigung aller weiteren Faktoren, die für den erfolgreichen Abschluss des Backend-Entwicklungsprojekts entscheidend sind.

6.1 Einleitung

Dieses Dokument dient als umfassende Projektdokumentation für die Entwicklung und Implementierung eines neuen Backend-Systems für Jetstream-Service. Im Gegensatz zur rein optischen Modernisierung der Webpräsenz, fokussiert dieses Projekt auf die technische Erneuerung und funktionale Erweiterung des bestehenden Backend-Systems. Dies umfasst insbesondere die Implementierung einer effizienten, sicheren und benutzerfreundlichen Schnittstelle für die Verwaltung von Serviceaufträgen sowie die Integration moderner Sicherheits- und Datenverwaltungsstandards, um den Mitarbeitern eine optimierte und effiziente Nutzung zu ermöglichen.

6.2 Vorgabe/Anforderungen

Die Hauptziele des Backend-Projekts umfassen die Entwicklung und Implementierung von Schlüsselelementen, die für ein effizientes und sicheres Auftragsmanagement erforderlich sind:

- Login-Funktionalität: Ein sicherer Login-Dialog mit Passwort ist erforderlich, um den Mitarbeitern autorisierten Zugang zu den Daten zu gewähren.
- Mitarbeiter können Serviceaufträge einsehen, den Status ändern (Offen, In Arbeit, Abgeschlossen) und bei Bedarf Aufträge löschen (z.B. Stornierungen).
- Protokollierung: API-Endpoints müssen für die Fehlerlokalisierung protokolliert werden, entweder in einer Datenbank oder in einer Protokolldatei.

Die Dokumentation folgt dem IPERKA-Modell und ist in entsprechende Abschnitte unterteilt, die die Informationsbeschaffung, Planung, Entscheidung, Realisierung, Kontrolle und Auswertung des Projekts abdecken. Zudem enthält sie ein Versionsverzeichnis, um die Aktualität und Relevanz der bereitgestellten Informationen sicherzustellen.

6.3 Zusätzliche Anforderung

Im Laufe der Informationsphase wurden neben den grundlegenden Anforderungen weitere spezifische Bedürfnisse und Wünsche identifiziert, die das Projekt in einer optimierten Form prägen könnten. Diese zusätzlichen Anforderungen sollen sicherstellen, dass das Backend in ihrer Funktion nicht nur den grundlegenden Bedürfnissen, sondern auch spezielleren Ansprüchen gerecht wird.

6.3.1 Kommentarfunktion für Aufträge

Um die Kommunikation und Dokumentation zu verbessern, soll die Möglichkeit für Mitarbeiter geschaffen werden, zu jedem Auftrag individuelle Kommentare oder Notizen hinzuzufügen. Diese Funktion erleichtert den Informationsaustausch und bietet eine Plattform für interne Anmerkungen und Details, die für die Bearbeitung des Auftrags relevant sind.

6.3.2 Flexibilität bei der Auftragsbearbeitung

Ein wichtiger Aspekt ist die Fähigkeit, sämtliche Datenfelder eines Auftrags bei Bedarf ändern zu können. Diese Flexibilität ermöglicht es den Mitarbeitern, auf Änderungen in den Kundenanforderungen oder interne Updates reagieren zu können, was die Effizienz und Kundenzufriedenheit erhöht.

6.3.3 Sicherheitsmechanismus beim Login

Zur Erhöhung der Sicherheit wird ein Mechanismus implementiert, der das Login eines Mitarbeiters nach drei aufeinanderfolgenden Fehlanmeldungen automatisch sperrt. Diese Funktion dient dem Schutz sensibler Daten und verringert das Risiko unautorisierten Zugriffs.

6.3.4 Umgang mit gelöschten Aufträgen

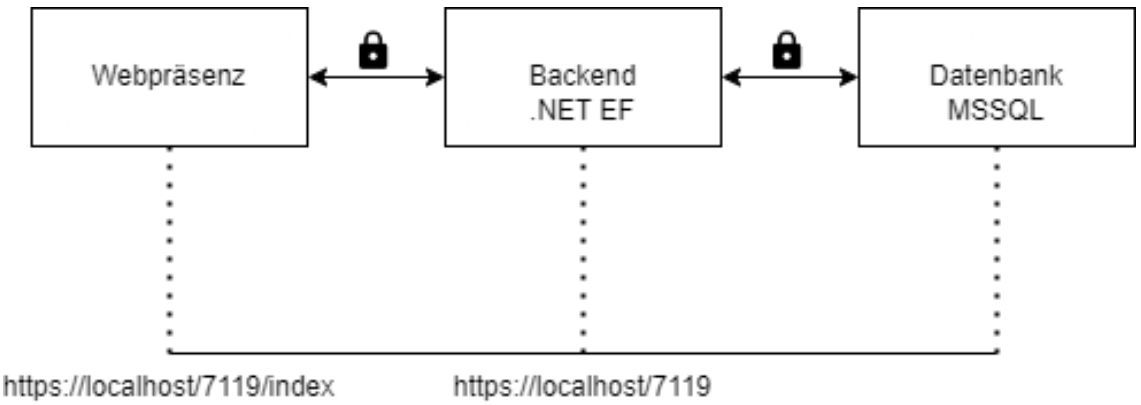
Anstatt gelöschte Aufträge komplett aus der Datenbank zu entfernen, werden diese lediglich als gelöscht markiert. Dieser Ansatz sorgt dafür, dass alle Daten für zukünftige Analysen und Berichte erhalten bleiben, während die aktuelle Datenansicht für die Mitarbeiter übersichtlich und relevant bleibt.

7 Planen

Die Planungsphase stellt einen entscheidenden Abschnitt dar. Nachdem in der Informationsphase alle erforderlichen Daten gesammelt und ein klares Bild der Anforderungen und Rahmenbedingungen geschaffen wurde, gilt es nun, diese Erkenntnisse in einen strukturierten und realisierbaren Plan zu übertragen.

7.1 Entwurf eines Anwendungsdesign

Im Rahmen der Aktualisierung des Backends der Firma Jetstreamski-Service liegt ein besonderer Fokus auf die Anbindung des Backends. Dazu wurde ein Anwendungsdesign erstellt, um die Backendverbindung einfacher darzustellen.



2 *Anwendungsdesign*

7.2 Organisation

Um die Architektur und das Zusammenspiel der Komponenten des Backend-Systems optimal zu gestalten, wurde eine sorgfältige Auswahl an Technologien und Methoden getroffen.

7.2.1 Technologieauswahl

Für die Entwicklung des Backend-Teils des Systems wurde ASP.NET mit dem .NET Entity Framework (EF) gewählt. Diese Technologie bietet ein robustes und etabliertes Framework für die Erstellung von skalierbaren und sicheren Webanwendungen. Die Entscheidung für ASP.NET basiert auf seiner Leistungsfähigkeit, seiner umfangreichen Funktionalität und seiner nahtlosen Integration mit dem .NET EF, welches als OR-Mapper für die Datenbankinteraktion dient.

Die Gestaltung der Datenbank erfolgt nach dem Code-First-Ansatz, der es ermöglicht, Datenbankstrukturen direkt aus den im Code definierten Entitätsklassen zu generieren. Dies wird durch den Einsatz von Migrationsbefehlen wie Add-Migration in der .NET EF Command-Line-Interface erreicht. Dieser Ansatz fördert die Agilität und Flexibilität der Entwicklung, da Änderungen am Datenmodell einfach durchgeführt und auf die Datenbank übertragen werden können.

Als Datenbankmanagementsystem wurde MSSQL ausgewählt. MSSQL ist bekannt für seine Zuverlässigkeit, Sicherheit und Kompatibilität mit dem .NET-Ökosystem, was die Entwicklung und Wartung der Anwendung erleichtert.

Für die Authentifizierung wurde entschieden, JWT (JSON Web Tokens) Bearer Token zu verwenden. Diese Methode gilt als ein sicherer Standard für die Zustandsverwaltung zwischen Client und Server, bei der die Identität des Benutzers durch Token bestätigt wird, die über sichere HTTPS-Verbindungen übertragen werden.

7.2.1.1 Anwendungen und NuGet-Pakete

Alle verwendeten Anwendungen und NuGet-Pakete werden hier aufgelistet und beschrieben. Um Komplikationen zu verhindern sind die Versionierungen und vollständigen Beschreibung auf Seite dokumentiert.

Die WebAPI läuft unter dem Framework ASP.NET7.0

Folgende NuGet-Pakete werden benötigt für das Backend:

- Microsoft.AspNetCore.Authentication.JwtBearer
- Microsoft.AspNetCore.OpenApi
- Microsoft.EntityFrameworkCore
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.EntityFrameworkCore.Tools
- Microsoft.IdentityModel.Tokens
- Serilog.AspNetCore
- Swashbuckle.AspNetCore
- System.IdentityModel.Tokens.Jwt

..

Da, dass Testprojekt ein eigenständiges xUnit-Projekt ist kommen folgende NuGet-Pakete für das Testprojekt dazu:

- FluentAssertions
- Microsoft.NET.Test.Sdk
- Moq
- Xunit
- xunit.runner.visualstudio
- coverlet.collector

7.2.2 GitHub

Mithilfe von GitHub wurde das komplette Projekt durch Versionsverläufe dokumentiert. Durch ein Git-Repository kann der Entwickler in der Realisierungsphase die komplette Programmierarbeit einfacher und unabhängig des Standortes bewältigen. Durch die Commits, die der Entwickler bei allen Änderungen macht, bringt dies eine grosse Transparenz in das Projekt zwischen dem Projektteam und Kunden bei.

7.2.3 Postman

Postman ist als integraler Bestandteil der Backend-Entwicklung ausgewählt worden. Dieses leistungsstarke Tool wird für das Testen der Web-APIs verwendet, die im Rahmen des Projekts entwickelt werden. Postman bietet mehrere Vorteile, die es zu einem unverzichtbaren Werkzeug im Entwicklungsprozess machen:

- **Einfache Handhabung:** Mit einer benutzerfreundlichen Oberfläche ermöglicht Postman das schnelle Erstellen und Senden von HTTP-Anfragen sowie das Betrachten von Antworten, was die API-Entwicklung und das Debugging vereinfacht.
- **Umfangreiche Testmöglichkeiten:** Postman erlaubt es, automatisierte Tests für verschiedene Szenarien zu schreiben und auszuführen, um die Robustheit und Zuverlässigkeit der API sicherzustellen.
- **Environment- und Variable-Management:** Durch die Verwendung von Environments und Variablen kann Postman verschiedene Entwicklungsumgebungen simulieren und erleichtert somit das Testen unter verschiedenen Bedingungen.
- **Dokumentation:** Postman unterstützt das Erstellen von API-Dokumentationen.

7.2.4 Trello-Board

Durch den Einsatz von Kanban-Online wird die Organisation weiter gefördert um allen Anforderungen, die in dieser Phase bestimmt werden, gerecht zu werden. Es gibt verschiedene Abteilungen, die farblich gekennzeichnet werden.

7.3 Planung des Backends

Die Backend-Planung für Jetstream-Service zielt darauf ab, ein robustes, skalierbares und sicheres System zu entwickeln, das als zuverlässige Basis für die Verwaltung von Online-Serviceaufträgen dient. Die Architektur soll künftige Integrationen erleichtern und auf ASP.NET Core basieren, wobei das Entity Framework für die Datenbankinteraktionen eingesetzt wird.

Architektur-Design:

Strukturierte Verzeichnisse: Das Backend wird in verschiedene Verzeichnisse aufgeteilt, die eine klare Trennung der Verantwortlichkeiten ermöglichen.

Services: Diese Komponenten sind für die direkte Interaktion mit der Datenbank zuständig und abstrahieren die Geschäftslogik von den Controllern.

Controllerklassen: Sie handhaben die eingehenden API-Anfragen und sind für die Implementierung von Authentifizierungsmechanismen verantwortlich.

DTOs (Data Transfer Objects): DTO-Klassen werden verwendet, um eine klare Trennung zwischen der API-Schnittstellenschicht und dem internen Datenmodell zu schaffen.

Entwicklungspraktiken:

Validierung: Die Validierung von Eingaben wird serverseitig mit Hilfe von Regular Expressions durchgeführt, um die Korrektheit und Sicherheit der Daten zu gewährleisten.

Frontend-Integration: Die Frontend-Ressourcen werden im "wwwroot"-Verzeichnis des Backends bereitgestellt, um eine enge Integration und schnelle Auslieferung zu ermöglichen.

Datenbankkonzeption:

Code-First Ansatz: Die Datenbank wird über Code-First-Migrationen aufgebaut, wobei die Modelle in die DbContext-Klasse eingebunden werden, um die MSSQL-Datenbank zu generieren.

Konfiguration: Sämtliche Verbindungszeichenfolgen (ConnectionStrings) werden zentral in der appsettings.json-Datei gespeichert, um eine sichere und einheitliche Konfigurationsverwaltung zu gewährleisten.

Sicherheit und Initialisierung:

Erstinstallation: Die anfängliche Konfiguration der Datenbank erfolgt mit einem Systemadministrator-Konto unter Verwendung der Windows-Authentifizierung.

SQL-Login: Nach der Ersteinrichtung wird ein spezifischer SQL-Login erstellt und die Datenbankverbindung über einen neuen ConnectionString konfiguriert.

Dokumentation und Teststrategie:

Statuscodes: Alle HTTP-Statuscodes werden genau dokumentiert, um eine transparente Kommunikation über den API-Status zu ermöglichen.

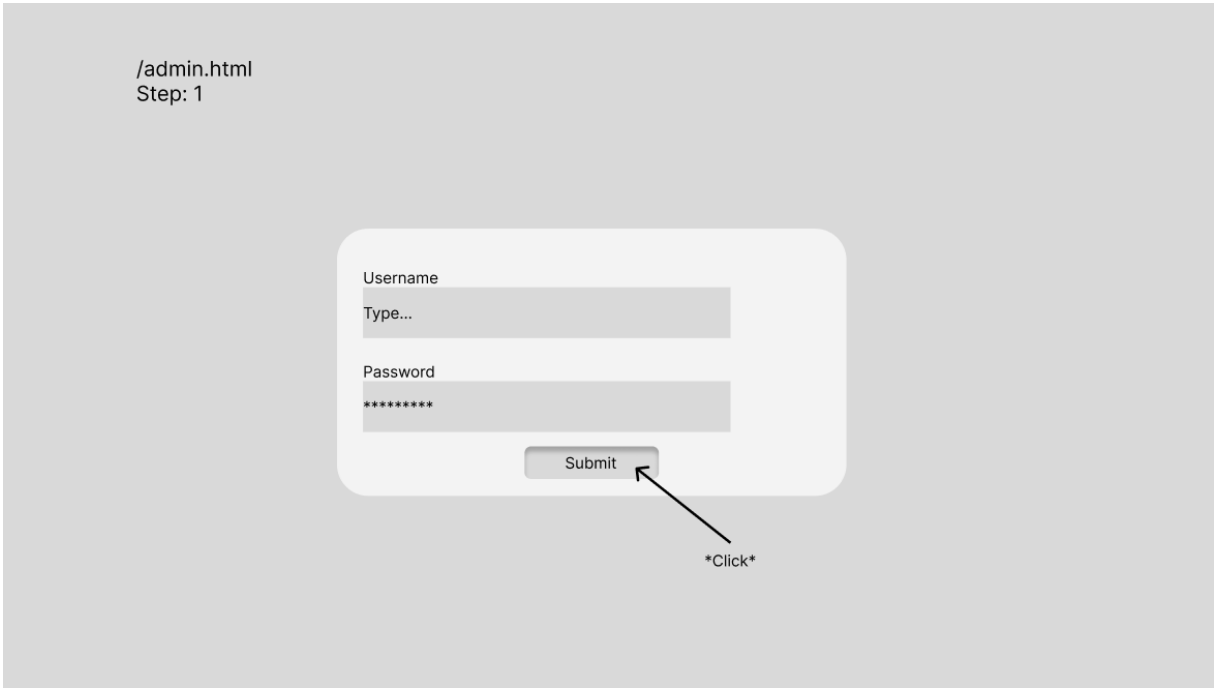
Code-Kommentierung: Funktionen, Methoden und Klassen werden mit XML-Summary-Kommentaren vollständig dokumentiert, um die Wartbarkeit und Verständlichkeit des Codes zu unterstützen.

Testplanung: Es wird ein Testplan entwickelt, der die Rahmenbedingungen für die Durchführung der Unit-Tests festlegt, um die Funktionalität und Stabilität des Backends sicherzustellen.

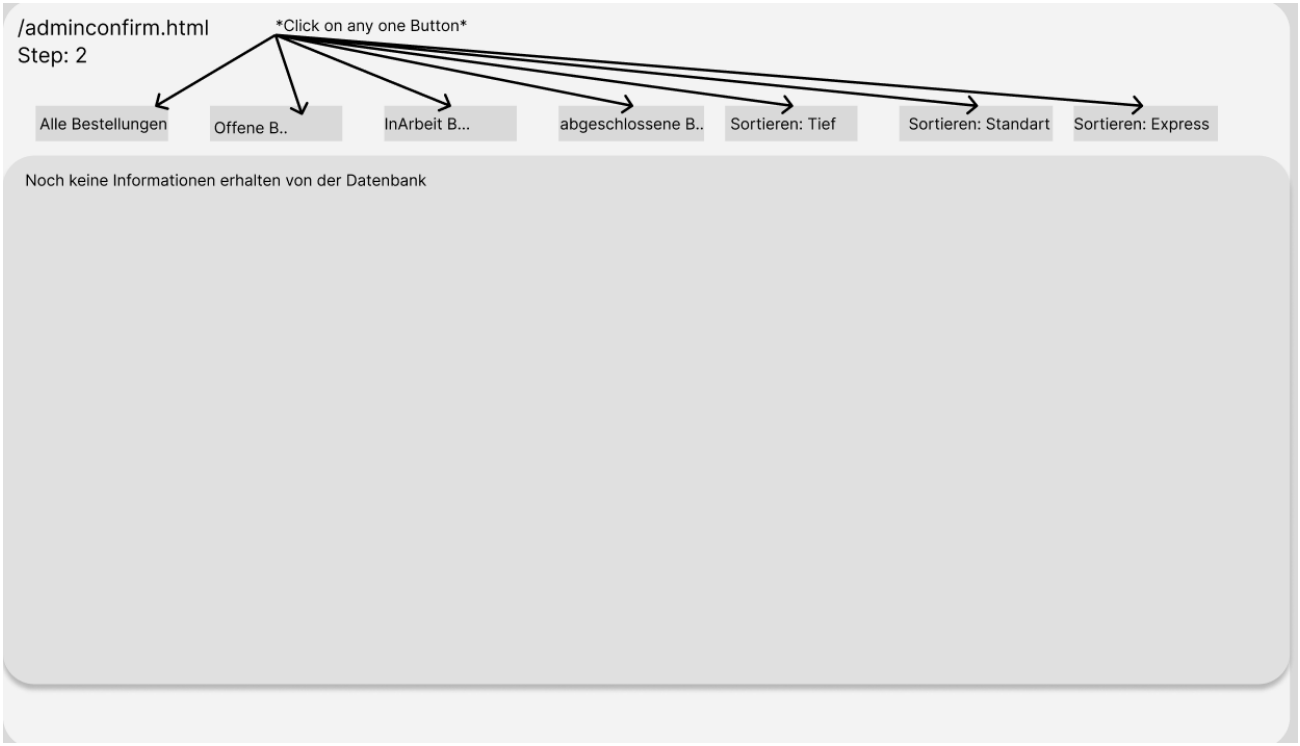
Diese strategische Planung garantiert ein geordnetes und effizientes Backend, das die betrieblichen Anforderungen von Jetstream-Service erfüllt und gleichzeitig eine hohe Codequalität und Systemintegrität aufrechterhält.

7.4 Entwurf für das Frontend Adminpanel

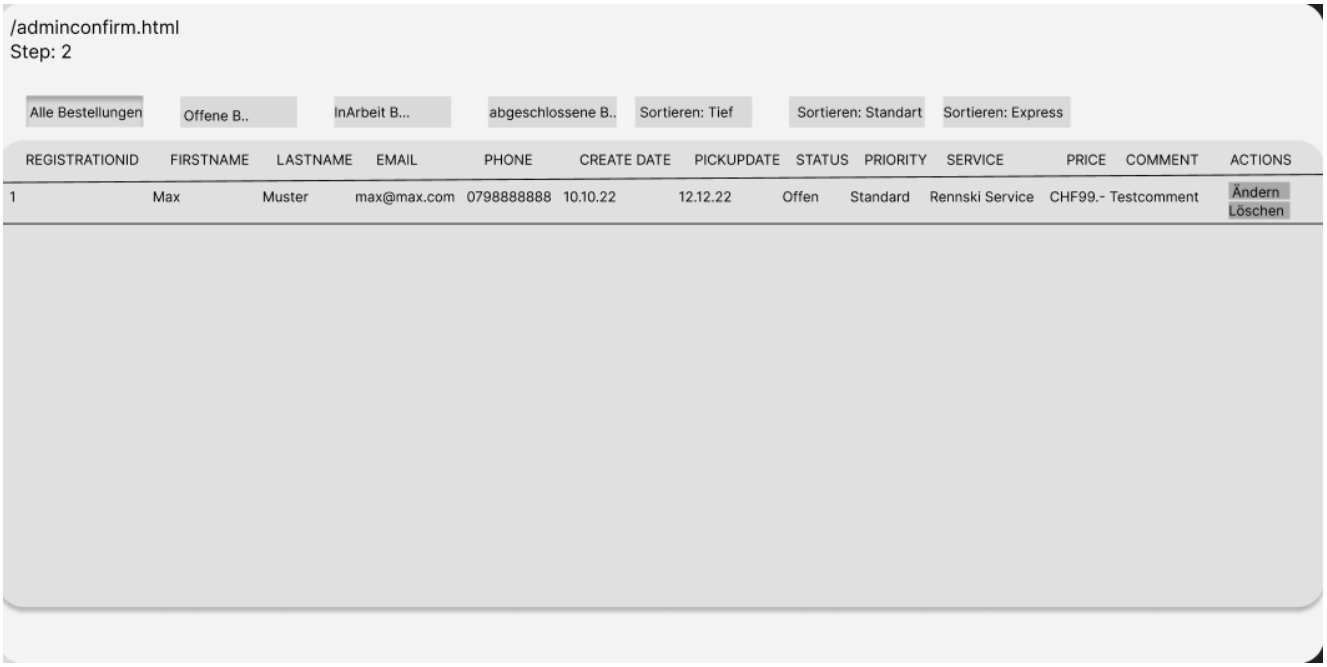
Da, dass Projekt auch eine Erweiterung des Frontends benötigt musste auch dies geplant werden. Das Frontend wurde somit mithilfe von Wireframe skizziert. Da die Onlineanfrage vom letzten Projekt stand, gab es nur das Login Formular und das Adminpanel zu skizzieren.



3 Login-Skizze



5 Admin-Panel Skizze



4 Admin-Panel Skizze (Zustand nach einem Klick)

7.5 Entwurf für Validierungstests

Die Qualität der Daten, die durch die Online-Anmeldung für Serviceaufträge erfasst werden, ist entscheidend für das erfolgreiche Management der Aufträge sowie für spätere Marketing-Aktivitäten.

Validierungstests für den Namen

- Leerzeichen: Der Name sollte keine führenden oder nachfolgenden Leerzeichen haben. Ein Trim-Verfahren wird angewandt, um unerwünschte Leerzeichen zu entfernen.
- Zeichentyp: Nur alphabetische Zeichen sind erlaubt. Sonderzeichen oder Zahlen sind nicht zugelassen.

Validierungstests für die Telefonnummer

- Format: Die Telefonnummer muss dem lokalen Standard entsprechen, zum Beispiel +41 79 123 45 67 für die Schweiz.
- Zeichentyp: Nur Zahlen und spezielle Zeichen wie "+" oder "-" sind erlaubt.
- Länge: Die Telefonnummer muss eine minimale Länge von 7 und eine maximale Länge von 15 Zeichen haben.

Validierungstests für die E-Mail-Adresse

Format: Die E-Mail-Adresse muss dem standardmässigen Format entsprechen, z.B. "beispiel@email.com".

Zeichentyp: Nur standardmässige Zeichen für E-Mail-Adressen sind erlaubt, einschliesslich Alphabet, Zahlen, und spezielle Zeichen wie ".", "@", "_", etc.

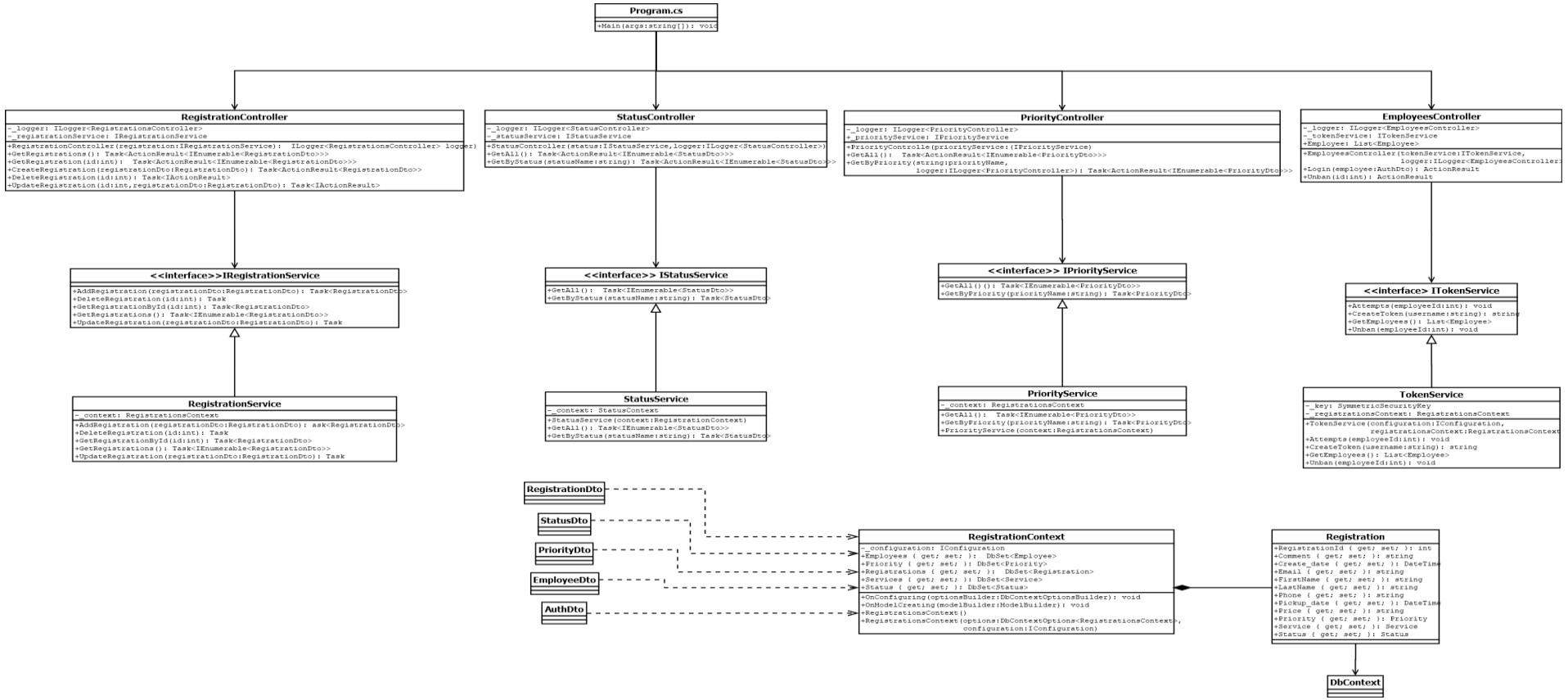
Technische Umsetzung

Frontend: Im Frontend wird JavaScript eingesetzt, um eine Erstvalidierung der Eingabefelder durchzuführen. Diese Validierung erfolgt direkt im Browser des Benutzers, um die Benutzererfahrung zu verbessern und die Last auf dem Backend zu reduzieren.

Darüber hinaus wurde eine zusätzliche Validierungsschicht im Backend implementiert, die mit ASP.NET realisiert wird. Diese dient als zweite Verteidigungslinie, um sicherzustellen, dass keine ungültigen Daten die Datenbank erreichen. Diese doppelte Überprüfung ist entscheidend, um die Datenqualität zu sichern und die Robustheit des Systems zu gewährleisten.

7.6 UML-Klassendiagram

Die Planung des UML-Klassendiagramms für das Backend-System von Jetstream-Service ist ein kritischer Schritt, der dazu beiträgt, die Architektur der Anwendung präzise zu definieren. Ein UML-Klassendiagramm bietet eine visuelle Darstellung der Klassen, ihrer Attribute und Methoden sowie der Beziehungen zwischen ihne



6 UML-Klassendiagram

7.7 Entwurf eines Testplans für xUnit-Tests

Der Testplan für die xUnit-Tests zielt darauf ab, die Funktionalität und Zuverlässigkeit des Jetstream-Skiservice-API zu gewährleisten. Folgende Testkategorien werden dabei berücksichtigt:

EmployeeControllerTests: Diese Tests überprüfen die Funktionalität der Endpunkte, die für die Verwaltung von Mitarbeiterdaten zuständig sind. Getestet wird, ob Mitarbeiterdaten korrekt abgerufen, hinzugefügt, aktualisiert und gelöscht werden können.

PriorityControllerTests: Diese Tests stellen sicher, dass die Priorisierung von Serviceaufträgen wie erwartet funktioniert. Es wird überprüft, ob die API korrekt auf Anfragen reagiert, die sich auf die Priorität von Aufträgen beziehen, einschließlich deren Erstellung, Abfrage und Aktualisierung.

RegistrationControllerTests: Hierbei wird die Handhabung von Serviceauftragsanmeldungen getestet. Dazu gehören Validierungen der Eingabedaten und die korrekte Erfassung und Speicherung von Serviceanmeldungen.

StatusControllerTests: Diese Tests validieren die Statusverwaltung der Serviceaufträge. Dabei wird geprüft, ob die Statusänderungen wie geplant reflektiert werden und ob alle zugehörigen Operationen wie das Abrufen des aktuellen Status korrekt ausgeführt werden.

Für jeden Testfall sind folgende Schritte definiert:

- Initialisierung: Einrichtung der Testumgebung, einschließlich Mocking von Abhängigkeiten.
- Ausführung: Durchführung der Testaktionen und Überprüfung der API-Antworten.
- Überprüfung: Bestätigung, dass das erwartete Verhalten mit dem tatsächlichen Ergebnis übereinstimmt.

Dieser Testplan gewährleistet eine umfassende Abdeckung der kritischen Funktionalitäten des Backend-Systems und stellt sicher, dass alle Komponenten wie vorgesehen arbeiten.

Dieser Entwurf dient in der Planungsphase als allgemein. Es wird in der Realisierungsphase ein vollständiger Testplan entwickelt.

8 Entscheiden

8.1 Code-First Datenbankerstellung

Nach sorgfältiger Abwägung verschiedener Ansätze für die Datenbankentwicklung wurde entschieden, dass der Code-First-Ansatz am besten für die Bedürfnisse von Jetstream-Service geeignet ist. Die Entscheidung basiert auf den folgenden Überlegungen:

- Agilität und Flexibilität: Code-First ermöglicht eine agile Entwicklung, bei der Änderungen am Datenmodell schnell umgesetzt und in die Datenbank überführt werden können.
- Versionierung und Quellcode-Integration: Das Datenbankschema kann als Teil des Quellcodes versioniert werden, wodurch eine bessere Nachvollziehbarkeit und Kollaboration im Entwicklungsteam gewährleistet ist.
- Fokus auf das Objektmodell: Dieser Ansatz erlaubt es Entwicklern, sich auf das Domain-Modell zu konzentrieren, anstatt sich mit Datenbankschemata zu beschäftigen, was eine objektorientierte Herangehensweise fördert.
- Migrationen: Entity Framework bietet ein leistungsfähiges Migrations-Tooling, dass die Verwaltung von Datenbankschemata erleichtert und die automatische Anwendung von Updates auf die Datenbank ermöglicht.

Die Implementierung von Code-First steht im Einklang mit den Prinzipien der modernen Softwareentwicklung und unterstützt das Ziel, eine skalierbare und wartbare Backend-Lösung zu schaffen. Durch die Verwendung dieses Ansatzes wird sichergestellt, dass das Backend-System von Jetstream-Service auf zukünftige Anforderungen und Erweiterungen vorbereitet ist.

8.2 wwwroot statt eigener Webserver

Nach eingehender Überlegung wurde entschieden, die Webanwendung direkt im "wwwroot"-Verzeichnis des ASP.NET Core-Projekts zu hosten, anstatt einen separaten Webserver einzurichten. Diese Entscheidung basiert auf mehreren Überlegungen:

Integrierte Lösung: Die Verwendung des "wwwroot"-Verzeichnisses ermöglicht eine nahtlose Integration der Webressourcen in die Backend-Architektur. Dadurch wird die Bereitstellung und Wartung der gesamten Anwendung vereinfacht.

Reduzierte Komplexität: Durch das Vermeiden eines separaten Webservers wird die Systemarchitektur vereinfacht, was die Konfiguration, das Deployment und das Monitoring der Anwendung erleichtert.

Verbesserte Performance: Die direkte Einbettung der Webressourcen in das Backend führt zu einer schnelleren Ladegeschwindigkeit und reduzierten Latenzzeiten, da die statischen Inhalte direkt vom ASP.NET Core-Server ausgeliefert werden.

Sicherheitsaspekte: Das Hosting der Webanwendung innerhalb des ASP.NET Core-Projekts bietet eine höhere Kontrolle über die Sicherheit, da die Sicherheitsfunktionen von ASP.NET Core direkt auf die Webressourcen angewendet werden können.

Die Entscheidung für das "wwwroot"-Verzeichnis steht im Einklang mit dem Ziel, eine effiziente, wartbare und sichere Webanwendung zu entwickeln, die reibungslos mit dem Backend-System interagiert.

9 Realisieren

Dieser Abschnitt dokumentiert die Umsetzungsphase des Projekts, in der die geplanten Konzepte und Entscheidungen in die Praxis umgesetzt werden.

9.1 Backend-Entwicklung

Die Backend-Entwicklung für Jetstreamski-Service umfasste mehrere Schlüsselaspekte, um eine robuste, effiziente und sichere Anwendung zu gewährleisten:

Architektur und Design: Basierend auf der vorherigen Planungsphase wurde eine saubere und modulare Architektur implementiert, die auf ASP.NET Core mit Entity Framework basiert. Die Struktur des Backends wurde in verschiedene Schichten unterteilt, einschließlich Datenzugriffsschicht, Geschäftslogikschicht und API-Schicht.

Datenbankintegration: Die Datenbank wurde mit dem Code-First-Ansatz unter Verwendung des Entity Framework Core erstellt. Dabei wurden die Modelle und Beziehungen definiert, die den Anforderungen des Jetstream-Service entsprechen. Migrationsverfahren wurden angewendet, um das Datenbankschema zu verwalten und zu aktualisieren.

RESTful API-Entwicklung: Es wurden mehrere Endpunkte implementiert, die CRUD-Operationen (Create, Read, Update, Delete) für die verschiedenen Entitäten wie Serviceaufträge und Kundeninformationen ermöglichen. Die API-Endpunkte wurden sorgfältig entworfen, um eine intuitive und konsistente Nutzung zu gewährleisten.

Sicherheit: Großen Wert wurde auf die Sicherheit gelegt. Dies umfasste die Implementierung von Authentifizierung und Autorisierung mittels JWT (JSON Web Tokens) und die Absicherung der API-Endpunkte gegen unautorisierten Zugriff.

Fehlerbehandlung und Logging: Um die Zuverlässigkeit und Wartbarkeit des Backends zu gewährleisten, wurden umfassende Fehlerbehandlungsmechanismen und ein detailliertes Logging-System implementiert.

Testen: Die Entwicklung wurde von umfangreichen Unit- und Integrationstests begleitet, um sicherzustellen, dass jede Komponente wie erwartet funktioniert und die gesamte Anwendung stabil und fehlerfrei läuft.

9.2 Frontend-Erweiterung

In diesem Projekt wurde das Frontend maßgeblich für die Bedienung durch Mitarbeiter erweitert, insbesondere durch die Einführung eines neuen Adminpanels. Diese Neugestaltung umfasste die Implementierung erweiterter Interaktionsmöglichkeiten für die Verwaltung aller Serviceaufträge.

Adminpanel-Design: Das Design des Adminpanels wurde bewusst schlicht und einfach gehalten, um eine klare und intuitive Benutzerführung zu gewährleisten. Die Mitarbeiter haben dadurch die Möglichkeit, effizient und ohne unnötige Komplexität auf die Verwaltungsfunktionen zuzugreifen.

Sicherheitsfunktion: Eine bedeutende Ergänzung stellt die adminshutdown.html-Seite dar. Diese Seite wird Mitarbeitern angezeigt, wenn das Passwort dreimal hintereinander falsch eingegeben wurde. Sie informiert darüber, dass der Zugang zum Konto vorübergehend gesperrt ist. Diese Funktion dient der Sicherheit des Systems und schützt vor unberechtigten Zugriffsversuchen.

9.3 Backend-Anbindung

Ein weiterer wichtiger Aspekt des Projekts ist die Integration der Online-Anmeldung in das neue Backend. Die Anbindung erfolgt über eine REST-API, was eine einfache und effiziente Kommunikation zwischen Frontend und Backend ermöglicht. Bei der Online-Anmeldung werden verschiedene Datenfelder wie Kundenname, E-Mail, Telefonnummer, Priorität und gewählte Dienstleistung erfasst. Ausserdem wird bei erfolgreichem Login der Bearer Token im localstorage des Browsers gespeichert und kann somit für weitere Operationen, die eine Authentifikation benötigen benutzt werden.

9.4 Authentifikation Bearer Token

Im Rahmen dieses Projekts wurde ein Bearer-Token-Authentifizierungssystem eingeführt, das für den Zugriff auf bestimmte, sicherheitsrelevante HTTP-Anfragen verwendet wird. Diese Methode stellt sicher, dass nur autorisierte Nutzer Zugang zu sensiblen Daten und Funktionen haben. Die Hauptmerkmale des implementierten Bearer-Token-Systems sind:

- **Token-Signatur:** Jeder Bearer-Token enthält eine Signatur, die die Authentizität und Integrität des Tokens sicherstellt. Diese Signatur kann bei Bedarf geändert werden, um die Sicherheit zu erhöhen und auf sich verändernde Anforderungen zu reagieren.
- **Nutzung des Usernames**: Für die Generierung des Tokens wird der Username des Nutzers herangezogen. Dies ermöglicht eine personalisierte und sichere Authentifikation, indem der Token direkt mit dem individuellen Nutzerkonto verknüpft wird.
- **Gültigkeitsdauer:** Um die Sicherheit weiter zu erhöhen, ist jeder Token nur für die Dauer eines Tages gültig. Nach Ablauf dieses Zeitraums muss der Nutzer einen neuen Token anfordern. Diese Maßnahme schützt vor Langzeit-Missbrauch von eventuell kompromittiertem Token.

Die Einführung der Bearer-Token-Authentifizierung ist ein zentraler Baustein für die Sicherheit der Anwendung. Sie gewährleistet, dass nur berechtigte Nutzer Zugriff auf wichtige Funktionen haben und trägt somit entscheidend zur Integrität und Vertraulichkeit der Benutzerdaten und -operationen bei.

10 Kontrollieren

Die Kontrollphase ist entscheidend, um sicherzustellen, dass das entwickelte System den Anforderungen und Erwartungen entspricht. Ein wichtiger Bestandteil dieser Phase ist die Implementierung eines Testprojekts.

10.1 Testplan

Dieser Testplan ist für das Testprojekt entwickelt geworden.

1. Testziele

Sicherstellen, dass alle Controller-Funktionalitäten korrekt arbeiten.

Überprüfung der Integration zwischen Controllern, Services und Datenmodellen.

2. Testumfang

Tests für EmployeeController, PriorityController, RegistrationController und StatusController.

3. Teststrategie und -ansatz

Unit-Tests für individuelle Funktionen in jedem Controller.

Mocking von Services und Datenmodellen mit Moq.

Verwendung von FluentAssertions für Testergebnisse.

4. Testumgebung

Die Tests werden in einer standardisierten Entwicklungsumgebung unter .NET durchgeführt.

5. Testdaten

Erstellung von Testdaten für Mitarbeiter, Prioritäten, Registrierungen und Status.

6. Testfälle und -szenarien

Jede Testdatei (EmployeeControllerTests.cs, etc.) enthält spezifische Testfälle für die entsprechenden Controller-Funktionen.

7. Risikomanagement

Identifikation potenzieller Risiken bei der Testdurchführung und Entwicklung von Strategien zur Risikominderung.

8. Kriterien für den Abschluss der Tests

Alle Testfälle müssen erfolgreich durchlaufen werden.

Keine kritischen Fehler dürfen in den Systemkomponenten verbleiben.

10.2 Testprojekt implementieren

Für Jetstream-Service wurde ein umfassendes Testprojekt implementiert, das auf den festgelegten Testplan abgestimmt ist. Die Implementierung des Testprojekts umfasste folgende Schlüsselelemente:

Testziele: Hauptziel war die Sicherstellung der korrekten Funktionsweise aller Controller. Dazu gehörte die Überprüfung der Integration zwischen Controllern, Services und Datenmodellen, um eine nahtlose Funktionalität der Anwendung zu garantieren.

Testumfang: Das Testprojekt konzentrierte sich auf die Tests für den EmployeeController, PriorityController, RegistrationController und StatusController. Jeder dieser Controller wurde auf korrekte Funktionalität und Integration überprüft.

Teststrategie und -ansatz: Für die Tests wurden Unit-Tests für jede Funktion innerhalb der Controller verwendet. Zur Simulation der Service- und Datenmodellinteraktionen wurde die Mocking-Bibliothek Moq eingesetzt. Die Überprüfung der Testergebnisse erfolgte mittels FluentAssertions, um präzise und verständliche Testausgaben zu ermöglichen.

Testumgebung: Die Tests wurden in einer standardisierten .NET-Entwicklungsumgebung durchgeführt, um konsistente und zuverlässige Ergebnisse zu gewährleisten.

Testdaten: Es wurden spezifische Testdatensätze für die verschiedenen Entitäten wie Mitarbeiter, Prioritäten, Registrierungen und Status erstellt, um realistische Testbedingungen zu schaffen.

Testfälle und -szenarien: Für jeden Controller wurden in den jeweiligen Testdateien (EmployeeControllerTests.cs usw.) spezifische Testfälle definiert, die die einzelnen Funktionalitäten abdecken.

Risikomanagement: Mögliche Risiken im Testprozess wurden identifiziert, und es wurden Strategien entwickelt, um diese Risiken zu minimieren. Es wurden kritische Fehler dadurch ausgeschlossen.

Kriterien für den Abschluss der Tests: Das Testprojekt wurde als erfolgreich abgeschlossen betrachtet, sobald alle Testfälle erfolgreich durchlaufen wurden und keine kritischen Fehler in den Systemkomponenten verblieben. Erfolgreiche Testfälle siehe hier.

Das Testprojekt hat sich als wichtig erwiesen, um die Zuverlässigkeit und Stabilität der Anwendung zu gewährleisten. Es trug entscheidend dazu bei, das Vertrauen in die Funktionalität der einzelnen Komponenten zu stärken und die Gesamtqualität des Systems zu sichern.

11 Auswerten

Die Auswertungsphase des Projekts bei Jetstream-Service bot eine entscheidende Gelegenheit, die gesammelten Ergebnisse und Erfahrungen umfassend zu analysieren. Diese Phase war von zentraler Bedeutung, da sie nicht nur die Leistung des entwickelten Systems bewertete, sondern auch wertvolle Einsichten für zukünftige Projekte lieferte. Durch die detaillierte Untersuchung der Projektergebnisse konnten sowohl die erreichten Erfolge als auch die Bereiche, in denen Verbesserungen möglich sind, identifiziert werden. Diese Bewertung war besonders relevant, um den Gesamterfolg des Projekts zu messen und zu verstehen, inwiefern die entwickelten Lösungen den Anforderungen und Erwartungen entsprachen.

Ein wichtiger Aspekt in dieser Phase war, dass das Backend der Anwendung noch nicht offiziell gelauncht wurde. Daher konzentrierte sich die Sammlung von Feedback vornehmlich auf die Rückmeldungen der Mitarbeiter und auf die Ergebnisse aus den durchgeführten Tests. Diese Feedbacks waren besonders aufschlussreich, da sie direkte Einblicke in die Nutzererfahrung und die technische Leistungsfähigkeit des Systems boten.

Insgesamt bot die Auswertungsphase tiefe Einblicke in die Stärken und Schwächen des Projekts. Sie lieferte essenzielle Erkenntnisse, die als Basis für zukünftige Innovationen und Verbesserungen bei Jetstream-Service dienen. Diese Phase war somit ein entscheidender Schritt, um die Qualität und Effizienz des Projekts kontinuierlich zu steigern und die Weichen für zukünftige Erfolge zu stellen.

12 Fazit

Das Projekt war eine gute Challange für den Entwickler und es war eine tolle Erfahrung für Jetstreamski-Service zuvor die Webseite neuzugestalten und anschliessend das komplette Backend zu entwickeln.

12.1 Persönliches Fazit - Mahir Gönen

Das Projekt bei Jetstream-Ski-Service zählt für mich zu den tollsten Erfahrungen meiner schulischen Laufbahn. Es hat mir nicht nur ermöglicht, mein Wissen und meine Fähigkeiten auf die Probe zu stellen, sondern auch mich für die Backend-Entwicklung zu sehen. Die Arbeit an diesem Projekt war außerordentlich bereichernd und hat mir viel Freude bereitet.

Die Kombination aus Unterrichtseinheiten und der Möglichkeit, selbstständig zu arbeiten, war ideal für mein Lernen und meine Entwicklung. Die tiefe Auseinandersetzung mit der Struktur und Funktionsweise eines Backends war besonders faszinierend. Es war eine aufschlussreiche Erfahrung, die verschiedenen Aspekte des Backend-Designs und der Implementierung zu verstehen und anzuwenden. Ich bin stolz auf das, was ich erreicht habe, und fühle mich bestärkt in meinem Verständnis und meiner Kompetenz in der Softwareentwicklung. Das Projekt hat mir nicht nur neue Fertigkeiten vermittelt, sondern auch mein Vertrauen in meine eigenen Fähigkeiten gestärkt.

Die Tatsache, dass ich das Ergebnis meiner Arbeit auf GitHub veröffentlichen und mit der Öffentlichkeit teilen kann, ist für mich ein besonderer Höhepunkt. Es ist nicht nur ein Zeugnis meines technischen Könnens, sondern auch ein Beleg für meine Hingabe und Leidenschaft für die Softwareentwicklung.

Insgesamt war dieses Projekt eine unglaublich lohnende Erfahrung, die mir nicht nur tiefe Einblicke in die technischen Aspekte der Softwareentwicklung gegeben hat, sondern auch meine Liebe zur Programmierung und zum Problemlösen weiter gefestigt hat.

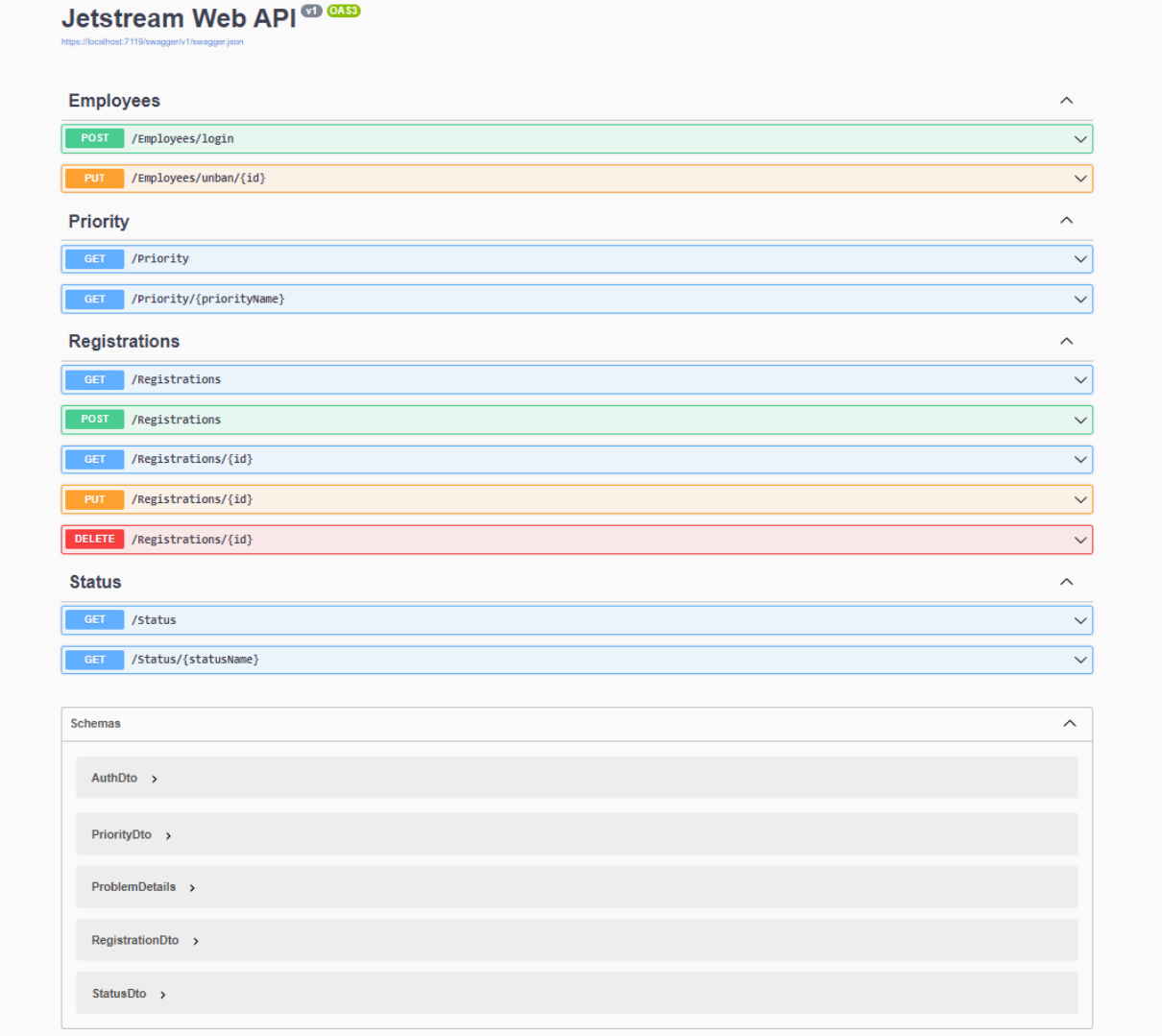
Anhänge

I. Glossar

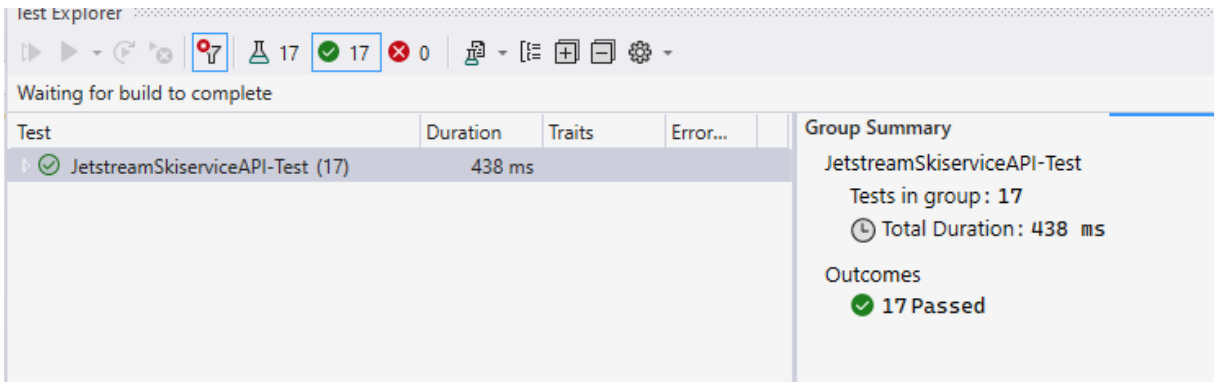
BEGRIFF	BESCHREIBUNG / ERKLÄRUNG
Backend	Das Backend bezeichnet den serverseitigen Teil einer Anwendung, der für die Datenverarbeitung, Logik und Interaktion mit Datenbanken zuständig ist. Es arbeitet im Hintergrund und ist für den Endnutzer nicht direkt sichtbar.
Code-first	Beim Code-First wird die Datenbank vom Code erstellt. Der Code bzw. die komplette Datenbank wird nicht durch selbst geschriebenen SQL-Querys erstellt, sondern durch ein Framework und dem Code migriert.
Frontend	Frontend bezieht sich auf die Benutzeroberfläche und alle Komponenten, die der Endbenutzer direkt in einem Webprojekt oder einer Softwareanwendung erlebt.
JSON	JSON steht für JavaScript Object Notation und ist ein leichtgewichtiges Datenformat, das für den Austausch von Daten zwischen einem Server und einer Webanwendung verwendet wird.
JSON Bearer Token	Ein JSON Bearer Token ist ein Authentifizierungs-Token, das in einem standardisierten JSON-Format vorliegt und zur Absicherung von API-Anfragen verwendet wird. Es wird im Rahmen des Bearer-Authentifizierungsschemas im HTTP-Header übertragen und ermöglicht den sicheren Zugriff auf geschützte Ressourcen, indem es die Identität des Benutzers überprüft.
Regex	Regex steht für Regular Expression und ist ein Muster, das für die Übereinstimmung mit einer Zeichenfolge innerhalb eines Texts verwendet wird.
REST-API	REST-API steht für "Representational State Transfer Application Programming Interface". Es ist eine Schnittstelle, die den Datenaustausch zwischen verschiedenen Systemen über HTTP ermöglicht, oft im JSON- oder XML-Format.

Tabelle 2 Glossar

II. Weitere Dokumente



7 SWAGGER-UI



III. Versionsverzechnis der Anwendung/Pakete

Anwendung/Paket	Version
.NET WebAPI	.NET7.0
Microsoft.AspNetCore.Authentication.JwtBearer	7.0.14
Microsoft.AspNetCore.OpenApi	7.0.13
Microsoft.EntityFrameworkCore	7.0.13
Microsoft.EntityFrameworkCore.SqlServer	7.0.13
Microsoft.EntityFrameworkCore.Tools	7.0.13
Microsoft.IdentityModel.Tokens	7.0.3
Serilog.AspNetCore	7.0.0
Swashbuckle.AspNetCore	6.5.0
System.IdentityModel.Tokens.Jwt	7.0.3
FluentAssertions	6.12.0
Microsoft.NET.Test.Sdk	17.6.0
Moq	4.20.69
Xunit	2.6.1
xunit.runner.visualstudio	2.4.5
coverlet.collector	3.2.0

13 Quellenverzeichnis

OneNote

Microsoft WebAPI: https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-8.0&tabs=visual-studio , https://dotnet.microsoft.com/en-us/apps/aspnet/apis

14 Tabellenverzeichnis

15 Abbildungsverzeichnis