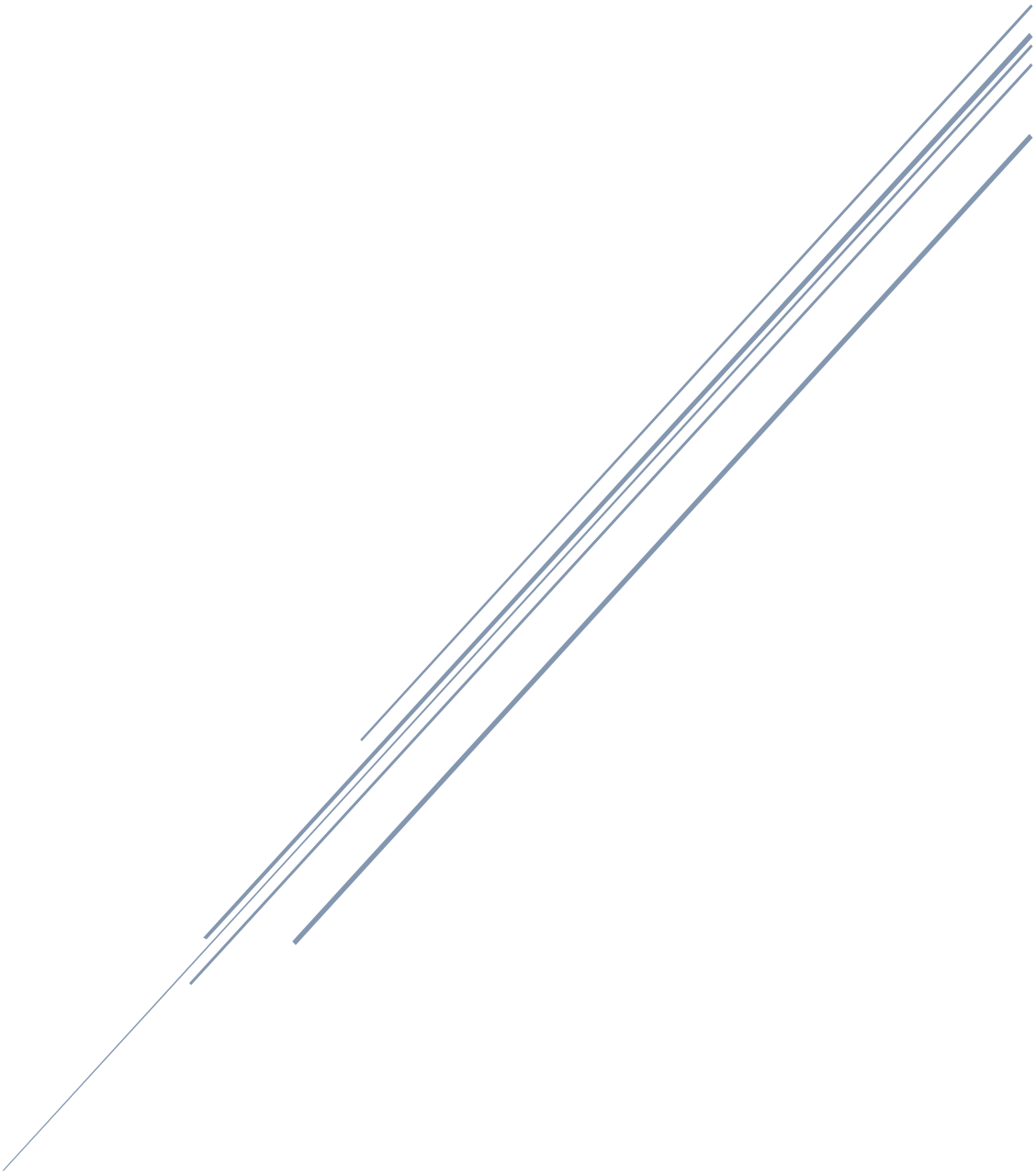
PROJEKTDOKUMENTATION PROJEKTARBEIT MODUL 295

Ilia Kalygin



Inhaltsverzeichnis

Informieren

Ausgangssituation

Die Firma Jetstream-Service führt als KMU in der Wintersaison Skiservicearbeiten durch und will im Zuge der Digitalisierung die interne Verwaltung der Ski-Service Aufträge komplett über eine Web- und Datenbank basierte Anwendung abwickeln. Die bereits existierende Online-Anmeldung soll bestehen bleiben und mit den erforderlichen Funktionen für das Auftragsmanagement erweitert werden. In der Hauptsaison sind bis zu 10 Mitarbeiter mit der Durchführung der Serverarbeiten beschäftigt. Diese sollen einen autorisierten passwortgeschützten Zugang zu den anstehenden Aufträgen erhalten und diese zur Abarbeitung übernehmen und ändern können.

Anforderungen

Das Projekt zur Digitalisierung der Ski-Service-Auftragsverwaltung bei Jetstream-Service umfasst die Entwicklung einer web- und datenbankbasierten Anwendung, die sich nahtlos in die bestehende Online-Anmeldung integriert. Die Schlüsselanforderungen des Projekts sind:

1. Web-API mit Authentifizierung: Entwicklung einer sicheren Web-API, die eine Authentifizierung für die Mitarbeiter ermöglicht. Dies ist entscheidend für den Zugang zu und die Bearbeitung von Serviceaufträgen.

2. Datenbankdesign und Implementierung: Konzeption und Umsetzung einer effizienten Datenbankstruktur. Die Implementierung kann entweder mit einem "Code First"- oder "Database First"-Ansatz erfolgen.

3. Integration in bestehende Systeme: Die neue Anwendung muss mit der bereits existierenden Online-Anmeldung für Kunden kompatibel sein und zusätzliche Funktionalitäten für das Auftragsmanagement bieten.

4. Mitarbeiterzugang: Bis zu 10 Mitarbeiter sollten in der Hauptsaison Zugriff auf die Anwendung haben, um Aufträge zu übernehmen, zu bearbeiten und zu aktualisieren.

5. Testprojekt / Testplan: Entwicklung eines umfassenden Testplans, einschliesslich Unit-Tests, um die Zuverlässigkeit und Funktionalität der Anwendung sicherzustellen.

6. Realisierung der Anwendung: Vollständige Umsetzung der Anwendung gemäss den spezifischen Anforderungen des Unternehmens, einschliesslich der erforderlichen Benutzerinterfaces und Backend-Logik

7. Funktionen für Auftragsmanagement: Die Anwendung muss Funktionen für das Anzeigen, Bearbeiten und Löschen von Serviceaufträgen bereitstellen. Hierzu gehören verschiedene Statusoptionen wie "Offen", "In Arbeit" und "Abgeschlossen".

8. Erweiterung der Kundeninformationen: Die Online-Anmeldung muss um zusätzliche Felder wie Kundenname, E-Mail, Telefon, Priorität und die spezifische Dienstleistung erweitert werden.

9. Sicherheitsaspekte: Sicherstellung der Datensicherheit und des Datenschutzes, insbesondere bei der Authentifizierung und Übertragung sensibler Informationen.

10. Benutzerfreundlichkeit und Zugänglichkeit: Die Anwendung sollte intuitiv und leicht bedienbar sein, um eine effiziente Nutzung durch die Mitarbeiter zu gewährleisten.

Planen

Projektplan

1. Projektstart und Vorbereitung
2. Anforderungsanalyse
3. Planung und Konzeption
4. Design und Entwicklung
5. Testing Phase 1
6. Weiterentwicklung und Optimierung
7. Testing Phase 2 und erstellen einer Benutzeranleitung
8. Finalisierung
9. Präsentation des Ergebnisses

Welche Software und Werkzeuge stehen zur Verfügung?

Entwicklungs- und Codierungs-Tools

- **Visual Studio**: Eine leistungsstarke integrierte Entwicklungsumgebung (IDE) für .NET- und C#-Entwicklung, ideal für die Backend-Entwicklung und für komplexe Programmieraufgaben.
- **Visual Studio Code**: Ein vielseitiger und leichtgewichtiger Code-Editor, der sich gut für Frontend-Entwicklung, kleinere Backend-Arbeiten und die Skripterstellung eignet.
- **HTML/CSS/JS für Frontend**: Standard-Webtechnologien zur Gestaltung der Benutzeroberfläche, die in Kombination mit Visual Studio Code eine effiziente Frontend-Entwicklung ermöglichen.

Datenbank-Management und Entwicklung

- **SQL Express**: Eine kostenlose, leichtgewichtige und eingeschränkte Version von Microsoft SQL Server, geeignet für kleinere bis mittlere Anwendungen.
- **SQL Server Management Studio (SSMS)**: Eine umfassende Umgebung zur Verwaltung, Wartung und Entwicklung von SQL-Datenbanken, ideal zur Handhabung von SQL Express-Datenbanke

API-Entwicklung und Dokumentation

- **Swagger (OpenAPI):** Ein Tool zur Erstellung interaktiver API-Dokumentationen, das die Visualisierung und das Testen von Web-APIs vereinfacht.
- **C# Web API .NET Core für Backend**: Ein robustes Framework für die Erstellung von Web-APIs, das sich durch hohe Leistung und Plattformunabhängigkeit auszeichnet.

Testing und API-Interaktion

- **Postman**: Ein vielseitiges Tool zum Testen von APIs, das das Senden von Anfragen, das Überprüfen von Antworten und das Automatisieren von Tests ermöglicht.
- **.NET Framework**: Eine umfangreiche Softwareentwicklungsplattform von Microsoft, die für die Entwicklung von Anwendungen und Services auf .NET-Technologie verwendet wird.

Entscheiden

Codefirst oder DBfirst?

Nach sorgfältiger Überlegung und Abwägung der verfügbaren Optionen habe ich mich für den "Database First"-Ansatz für die Entwicklung der Datenbank unserer Ski-Service-Auftragsverwaltung entschieden. Diese Entscheidung basiert auf folgenden Gründen:

- Besseres Verständnis aus dem Unterricht: Während des Unterrichts und der damit verbundenen Lernphasen stellte sich heraus, dass ich ein tieferes und klareres Verständnis für den "Database First"-Ansatz entwickelt habe. Diese Kenntnisse ermöglichen es mir, die Konzepte und Techniken, die ich gelernt habe, direkt und effizient in die Praxis umzusetzen.
- Einfachere Realisierung: Meine bisherigen Erfahrungen und das erlangte Wissen befähigen mich, den "Database First"-Ansatz mit grösserer Sicherheit und Kompetenz anzuwenden. Ich fühle mich wohler bei der Gestaltung und Implementierung der Datenbankstrukturen, bevor ich mich der Entwicklung der zugehörigen Geschäftslogik und Anwendungscode widme. Diese Vorgehensweise scheint mir intuitiver und ist besser an meinen Arbeitsstil angepasst.
-

Der "Database First"-Ansatz bietet auch den Vorteil, dass die Datenbankstruktur zu Beginn des Projekts klar definiert wird. Dies erleichtert die Planung und ermöglicht ein strukturierteres Vorgehen bei der Entwicklung. Ausserdem kann ich die Datenbank mit Tools wie SQL Server Management Studio visualisieren und bearbeiten, was mir eine zusätzliche Ebene der Kontrolle und Klarheit über die Datenstrukturen und Beziehungen gibt.

Durch die Wahl dieses Ansatzes bin ich zuversichtlich, dass wir eine robuste und gut strukturierte Datenbank für unser Projekt entwickeln können, die den Anforderungen und Zielen von Jetstream-Service entspricht.

Realisieren

Datenbank erstellen

1. Prüfung der Existenz der Datenbank: Zunächst wurde überprüft, ob eine Datenbank mit dem Namen SkiServiceManagement bereits existiert. Dies geschah mit dem SQL-Befehl:
```
IF NOT EXISTS (SELECT * FROM sys.databases WHERE name =
'SkiServiceManagement')
BEGIN
    CREATE DATABASE SkiServiceManagement;
END
GO
```
Diese Abfrage stellt sicher, dass eine neue Datenbank nur dann erstellt wird, wenn sie noch nicht existiert, um Doppelungen und Konflikte zu vermeiden.
2. Verwendung der SkiServiceManagement-Datenbank: Nach der erfolgreichen Erstellung oder Bestätigung der Existenz der Datenbank wurde sie für weitere Operationen ausgewählt:
```
USE SkiServiceManagement;
GO
```
3. Erstellung der Tabelle ServiceOrders: Die zentrale Tabelle ServiceOrders wurde erstellt, um die Serviceaufträge zu verwalten. Die Tabelle wurde mit mehreren Spalten für verschiedene Auftragsdetails entworfen:
```
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'ServiceOrders') AND type in (N'U'))
BEGIN
    CREATE TABLE ServiceOrders (
        OrderID INT PRIMARY KEY IDENTITY(1,1),
        CustomerName NVARCHAR(100),
        CustomerEmail NVARCHAR(100),
        CustomerPhone NVARCHAR(20),
        Priority NVARCHAR(50),
        ServiceType NVARCHAR(100),
        CreateDate DATETIME,
        PickupDate DATETIME,
        Status NVARCHAR(50) DEFAULT 'Offen',
          Comment NVARCHAR(100) DEFAULT ''
    );
END
     GO
```

Dieser Schritt war entscheidend, um sicherzustellen, dass alle notwendigen Informationen über jeden Serviceauftrag erfasst und effizient verwaltet werden können. Die Tabelle umfasst Spalten für Kundeninformationen, Priorität, Dienstleistungstyp, Erstellungs- und Abholdatum sowie den Status und Kommentare zu jedem Auftrag.

Pogrammierung WEB API

Bei der Programmierung der Web-API für das Ski-Service-Auftragsverwaltungssystem wurde ein strukturierter Ansatz verfolgt, der auf die effiziente Verarbeitung und Verwaltung von Serviceaufträgen abzielt. Die Kernkomponenten des Backend-Systems, die im Rahmen dieses Prozesses entwickelt wurden, umfassen den OrderController, die Konfigurationsdatei appsettings.json, sowie die Klassen Context und Order, welche die Logik und die Struktur der Datenbankinteraktionen definieren.

Der OrderController wurde implementiert, um als zentraler Knotenpunkt für die Verarbeitung von HTTP-Anfragen zu dienen. Mit Methoden zum Handhaben von GET, POST, PUT und DELETE Anfragen, bietet er die Schnittstelle, über die Benutzer mit dem System interagieren, sei es um Daten abzurufen, neue Aufträge zu erstellen, bestehende zu bearbeiten oder zu entfernen.

Die Verbindungsdetails zur Datenbank werden sicher in der appsettings.json gespeichert. Diese Abstraktion der Datenbankkonfiguration ermöglicht es, die Anbindung an die Datenbank zu zentralisieren und bei Bedarf anzupassen, ohne den Code direkt zu verändern.

Die Context-Klasse, die den DbContext von Entity Framework Core erweitert, stellt den programmatischen Zugang zu den Datenbankressourcen bereit. Innerhalb dieser Klasse wird eine Instanz von DbSet für ServiceOrders definiert, welche das Framework instruiert, wie die Daten der ServiceOrders-Tabelle verwaltet werden sollen. Dieser Ansatz unterstützt die klare Trennung zwischen der API-Logik und den Datenbankoperationen.

Die Definition der Order-Entität in der Order.cs-Datei legt die Datenstruktur fest, die die API verwenden wird. Diese Klasse spiegelt die Attribute eines Serviceauftrags wider und bindet sie an die entsprechenden Datenbankfelder, was die Grundlage für das Erstellen, Lesen, Aktualisieren und Löschen von Datensätzen bildet.

Die Programmierung der Web-API war ein methodischer Prozess, der darauf abzielte, eine robuste, wartbare und skalierbare Lösung zu schaffen, die die Anforderungen

der Ski-Service-Auftragsverwaltung effizient erfüllt. Durch die sinnvolle Nutzung von .NET Core und Entity Framework Core bietet die erstellte API eine solide Plattform für die Verwaltung von Kundenaufträgen und die Interaktion mit dem Servicepersonal.

Frontend Verbinden

Um eine vollständige Frontend-Integration mit der Backend-Web-API für das Ski-Service-Auftragsverwaltungssystem zu realisieren, wurde ein JavaScript-Script entwickelt, das den Austausch von Daten mittels GET, POST, PUT und DELETE Anfragen ermöglicht. Hier eine zusammenfassende Beschreibung des Entwicklungsprozesses:

Schritte zur Frontend-Integration der Web-API:

1. Einrichten der API-Verbindung: Zunächst wurde die Variable apiUrl mit der URL der Backend-Web-API initialisiert. Diese URL dient als Basis für alle nachfolgenden HTTP-Anfragen.
2. GET-Anfragen zum Abrufen von Daten: Mit der fetch-Funktion wurden GET-Anfragen an die API gesendet, um die Daten der Serviceaufträge abzurufen. Die ankommenden Daten wurden im Erfolgsfall in ein JSON-Format konvertiert und im Frontend verarbeitet.
3. POST-Anfragen zum Erstellen neuer Aufträge: Über ein Benutzerformular können neue Auftragsdaten eingegeben und mittels POST-Anfragen an die API gesendet werden, um neue Serviceaufträge anzulegen.
4. PUT-Anfragen zur Aktualisierung bestehender Aufträge: Für die Bearbeitung und Aktualisierung von Auftragsdaten werden PUT-Anfragen verwendet. Die entsprechenden Datenfelder im Frontend ermöglichen die Modifikation der Daten, die anschliessend an die API übermittelt werden.
5. DELETE-Anfragen zum Entfernen von Aufträgen: Um Aufträge zu löschen, werden DELETE-Anfragen eingesetzt. Im Frontend werden Schaltflächen bereitgestellt, die es dem Benutzer ermöglichen, einen Auftrag auszuwählen und dessen Löschung zu initiieren.
6. Dynamisches Rendering der UI-Komponenten: Das Script erzeugt dynamisch HTML-Elemente, die die Daten aus den API-Antworten darstellen und Felder für die Interaktion zur Verfügung stellen.
7. Fehlerbehandlung und Benutzerfeedback: Fehler bei HTTP-Anfragen werden abgefangen und dem Benutzer werden entsprechende Fehlermeldungen angezeigt.

Kontrollieren

Postman

Um die Funktionalität und Zuverlässigkeit der Web-API für das Ski-Service-Auftragsverwaltungssystem sicherzustellen, wurde Postman als wesentliches Tool für die Kontrollphase des Projekts eingesetzt. Postman ist eine Plattform für die Entwicklung von APIs, die es ermöglicht, Anfragen zu senden, Antworten zu empfangen und diese zu analysieren. In dieser Phase wurden die folgenden Schritte durchgeführt:

- API-Endpunkte testen: Jeder Endpunkt der API wurde mit Postman getestet, um sicherzustellen, dass die HTTP-Anfragen und -Antworten wie erwartet funktionieren. GET, POST, PUT und DELETE-Anfragen wurden sorgfältig geprüft, um die korrekte Logik und die erwarteten Ergebnisse zu bestätigen.

- Validierung der Daten: Die in der API zurückgegebenen Daten wurden auf ihre Korrektheit und Vollständigkeit überprüft. Dies umfasst das Überprüfen von Datentypen, Formaten und die Einhaltung des definierten Schemas.

Auswerten

Präsentation Zwischenstand

Im Verlauf des Projekts zur Digitalisierung der Ski-Service-Auftragsverwaltung wurde eine Präsentation vor der Klasse gehalten, um den aktuellen Zwischenstand darzulegen. Diese Präsentation bot die Gelegenheit, den Fortschritt zu kommunizieren, erste Erkenntnisse zu teilen und konstruktives Feedback zu erhalten. Zu diesem Zeitpunkt war das Projekt noch nicht abgeschlossen, und die folgenden Punkte wurden während der Präsentation hervorgehoben:

- Projektfortschritt: Es wurde detailliert beschrieben, welche Phasen bereits abgeschlossen waren und welche Schritte als nächstes geplant sind. Dies gab einen klaren Überblick über den bisherigen Verlauf des Projekts.

- Herausforderungen und Lösungsansätze: Bisherige Herausforderungen wurden angesprochen, zusammen mit den Strategien, die angewandt wurden, um diese zu überwinden. Dies umfasste technische, konzeptionelle und organisatorische Aspekte.

- Erste Ergebnisse und Erkenntnisse: Obwohl das Projekt noch nicht abgeschlossen war, wurden erste Ergebnisse präsentiert, die aus den bisherigen Tests und Entwicklungsarbeiten gewonnen wurden.

- Demonstration der aktuellen Funktionalität: Anhand einer Live-Demo oder Screenshots wurde die Funktionsweise der bereits implementierten Teile der Web-API und des Frontends gezeigt.

README.md Benutzerhandbuch für Github

Im Zuge des Entwicklungsprozesses der Webanwendung für die Ski-Service-Auftragsverwaltung habe ich ein umfassendes Benutzerhandbuch in Form einer README.md-Datei erstellt. Dieses Handbuch ist auf GitHub verfügbar und bietet eine detaillierte Anleitung für alle Schritte – von der Installation bis zur Ausführung der Anwendung.