ipso! Bildung seit 1876

9. DEZEMBER 2024

Modul 295
JetStream Ski-Service
Backend API
Projektarbeit

ANTHONY FRANZ & NESIM ABDELAZIZ
IPSO BILDUNG AG
Elisabethenanlage 9, 4051 Basel

ipso! Bildung seit 1876

Inhaltsverzeichnis

1 Versionsverlauf

2 Informieren

2.1 Ausgangslage

Die Firma Jetstream-Service, ein KMU, führt während der Wintersaison Skiservicearbeiten durch. Um die internen Abläufe zu digitalisieren, soll eine neue Anwendung entwickelt werden, die die Verwaltung der Serviceaufträge vollständig unterstützt. Der Fokus liegt auf der Backend-Entwicklung, während die bereits existierende Online-Anmeldung bestehen bleibt. Ziel ist es, ein effizienteres Auftragsmanagementsystem zu schaffen, das den Anforderungen der Mitarbeiter und der Geschäftsleitung entspricht.

2.2 Ziel des Projekts

- Entwicklung eines Web-API-Backends mit Authentifizierung.

- Erstellung eines Datenbankdesigns in der 3. Normalform (Code-First oder Database-First).
- Sicherstellen, dass alle Anforderungen an das Backend und die Datenbank vollständig umgesetzt werden.
- Implementierung eines Testplans mit automatisierten Tests (z. B. Unit-Tests) und manuellen Tests (Postman).
- Bereitstellung einer vollständigen OpenAPI-Dokumentation.

2.3 Rahmenbedingungen

- Datenbanksystem: MySQL oder MS SQL.
- Zugriff auf die Datenbank über einen OR-Mapper (z. B. Entity Framework oder Sequelize).
- Testwerkzeuge: Postman für API-Tests.
- Projektmanagement nach der IPERKA-Methode.
- Implementierung in einem Git-Repository mit sauberer Versionierung.

2.4 Anforderungen (Funktional Nicht-Funktional)

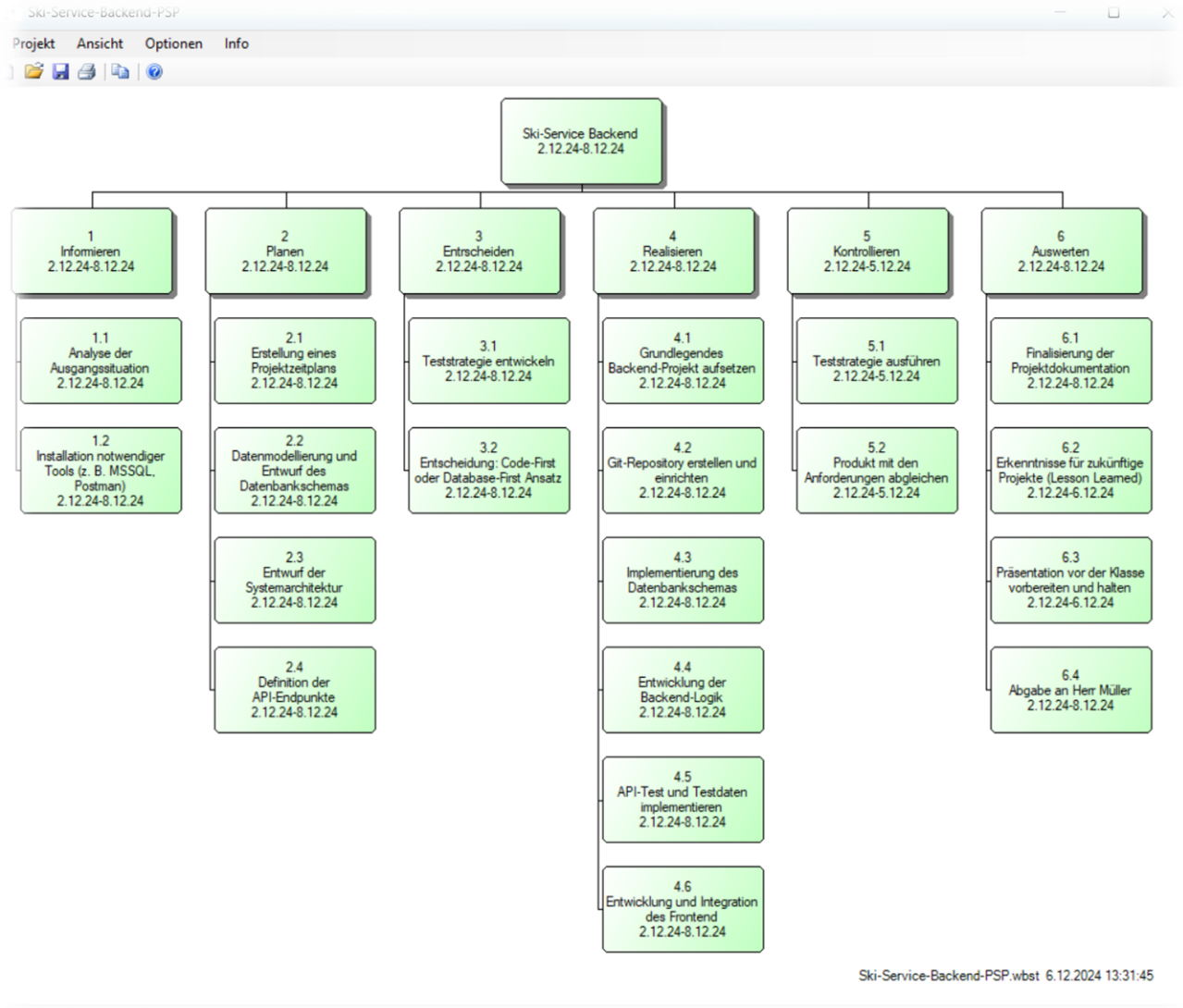
Funktionale Anforderungen:

1. Login mit Benutzername und Passwort für Mitarbeiter (authentifizierte Bereiche).
2. Auftragsliste für anstehende Serviceaufträge anzeigen.
3. Statusänderung von Aufträgen durch Mitarbeiter.
4. Löschen von Aufträgen bei Stornierungen.
5. Protokollierung der API-Aufrufe zur Fehlerdiagnose.
6. Selektives Abrufen von Serviceaufträgen basierend auf Priorität.

Nicht-Funktionale Anforderungen:

1. Die API muss konsistent und RESTful gestaltet sein.
2. Hohe Performance und Zuverlässigkeit für die Hauptsaison (bis zu 10 Benutzer gleichzeitig).
3. Datensicherheit durch eingeschränkte Zugriffe und geschützte Authentifizierung.

3 Planen



4 Entscheiden

4.1 Technologische Entscheidungen

- **Datenbankmodellierung**: Der Code-First-Ansatz wurde gewählt, um die Datenbankstruktur direkt aus dem Code zu generieren. Dies bietet Flexibilität und nahtlose Integration in Entity Framework Core.
- **Authentifizierungsmethode**: JSON Web Token (JWT) gewährleistet eine sichere und skalierbare Authentifizierung.
- **Programmiersprache**: C# mit .NET Core wurde aufgrund der guten Integration in OR-Mapper und der hohen Performance gewählt.
- **Teststrategie**: Kombination aus Unit-Tests (für die Geschäftslogik) und Postman-Tests (für API-Endpunkte).

4.2 Begründung

Der Code-First-Ansatz minimiert die Komplexität der Datenbankentwicklung, während JWT eine moderne und sichere Lösung für Authentifizierung darstellt. Die Auswahl von C# und .NET Core bietet hohe Effizienz und bewährte Tools für professionelle Backend-Entwicklung.

5 Realisieren

Die Umsetzung des Projekts "JetStreamServiceAPI" erfolgte in mehreren iterativen Phasen, wobei die definierten Anforderungen systematisch abgearbeitet wurden:

5.1 Entwicklungsprozess

1. **Projektsetup**:
   - Initialisierung des Git-Repositories zur Versionskontrolle.
   - Anlegen der Projektstruktur mit einer klaren Trennung von Modulen (z. B. Models, Controllers, Services).
   - Integration von Entity Framework Core und Installation relevanter NuGet-Pakete wie Microsoft.EntityFrameworkCore.Sqlite.
   - Implementierung einer Basisklasse für Datenbankkontext (DbContext) und Einrichten der Verbindungszeichenfolge.
2. **Modellierung der Datenbank**:
   - Erstellung der Datenmodelle wie CreateServiceOrderModel mittels des Code-First-Ansatzes.
   - Implementierung von Datenbankmigrationen und Generierung der SQLite-Datenbank.
3. **Backend-Implementierung**:
   - Entwicklung der API-Endpunkte für CRUD-Operationen.
   - Implementierung der JWT-Authentifizierung mit Endpunkten für Login und Registrierung.
   - Validierung der Benutzereingaben mittels Data Annotations.
4. **Frontend-Integration**:
   - Bereitstellung einer einfachen HTML-Oberfläche für Kunden (Impressum, Formular).
   - Einbindung von Bootstrap für ein responsives Design.
5. **Testen und Debugging**:
   - Unit-Tests für die Geschäftslogik, insbesondere Validierungen und Service-Methoden.
   - API-Tests mit Postman, um die Funktionalität und Sicherheit der Endpunkte sicherzustellen.
6. **Deployment**:
   - Lokale Bereitstellung der API mit Kestrel.
   - Optional: Vorbereitung für den Einsatz auf einem Cloud-Service (z. B. Azure).

6 Dokumentation

- Erstellung von Entwicklerdokumentationen mit einer Übersicht über API-Endpunkte.
- Erstellung von Benutzerdokumentationen für die Frontend-Nutzung.

7 Kontrollieren

Die Qualitätskontrolle des Projekts wurde durch kontinuierliche Tests und Feedback-Schleifen sichergestellt.

8 Testprozess

8.1 Unit-Tests:

- Testabdeckung für die Validierungsmethoden im CreateServiceOrderModel.
- Test der JWT-Authentifizierung auf korrekte Token-Ausgabe und -Prüfung.

8.2 API-Tests mit Postman:

- Test der Endpunkte für Kundenbestellungen (Erstellen, Lesen, Aktualisieren, Löschen).
- Test der Authentifizierungs- und Autorisierungsmechanismen.

8.3 Integrationstests:

- Prüfung der Datenübertragung zwischen Frontend, API und Datenbank.
- Sicherstellung der Konsistenz der Daten in der SQLite-Datenbank.

9 Fehleranalyse

- Identifizierte Probleme bei der Telefonnummernvalidierung wurden durch Überarbeitung der regulären Ausdrücke behoben.
- Fehlerhafte Token-Ablaufzeiten wurden durch Anpassung der JWT-Konfiguration korrigiert.

10 Auswerten

10.1 Projektziele

Das Ziel, eine API zur Verwaltung von Kundenaufträgen mit einer sicheren Authentifizierung und einfacher Benutzeroberfläche zu entwickeln, wurde erreicht. Die wichtigsten Features wurden erfolgreich umgesetzt:

- JWT-Authentifizierung
- CRUD-Operationen für Serviceaufträge
- Responsives Frontend

10.2 Herausforderungen

- Initiale Schwierigkeiten bei der Datenvalidierung (z. B. reguläre Ausdrücke).
- Abstimmung zwischen Frontend und Backend.
- Sicherstellung der korrekten Token-Generierung und Überprüfung.

10.3 Erkenntnisse

- Der Code-First-Ansatz mit Entity Framework Core erleichterte die Datenbankmodellierung erheblich.
- Postman war ein wertvolles Werkzeug zur Überprüfung der API-Funktionalität.
- Bootstrap ermöglichte eine schnelle und ansprechende Gestaltung der Benutzeroberfläche.