Leon Egli

Doku Modul 295

Von Leon Egli

Inhalt

[Lerndokumentation: Ski-Service Management System 2](#_Toc190705047)

[Informieren 2](#_Toc190705048)

[1.1 Problemstellung 2](#_Toc190705049)

[1.3 Allgemeine Anforderungen 3](#_Toc190705050)

[2. Planen 4](#_Toc190705051)

[2.1 Projektstruktur & Technologien 4](#_Toc190705052)

[2.2 Zeitplanung 4](#_Toc190705053)

[3. Entscheiden 5](#_Toc190705054)

[3.1 Datenbankdesign 5](#_Toc190705055)

[3.2 Warum Database First? 5](#_Toc190705056)

[4. Realisieren 6](#_Toc190705057)

[4.1 Implementierung der API 6](#_Toc190705058)

[4.2 Fehlersuche & Debugging 6](#_Toc190705059)

[5. Kontrollieren 7](#_Toc190705060)

[5.1 Testfälle (Postman) 7](#_Toc190705061)

[6. Auswerten 8](#_Toc190705062)

[6.1 Erkenntnisse 8](#_Toc190705063)

[6.2 Lessons Learned 8](#_Toc190705064)

[6.3 Nächste Schritte 8](#_Toc190705065)

[7. Fazit 8](#_Toc190705066)

# Lerndokumentation: Ski-Service Management System

## Informieren

### 1.1 Problemstellung

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.

Das Teilprojekt umfasst ausschliesslich den Backendteil und umfasst folgende Aufträge, welche nach IPERKA durchzuführen sind:

* Web-API Projekt mit Authentifikation (Backend) inkl. OpenAPI Dokumentation
* Datenbankdesign und Implementierung (Code First, oder Database First)
* Testprojekt / Testplan (Unit-Test)
* Realisierung der kompletten Anwendung, gemäss den Anforderungen
* Durchführung der Tests (Postman) Bemerkung:
* Bei datenlesenden Operationen (z.B. Auftragsliste usw.) ist keine Authentifikation erforderlich.
* Änderungen von Auftragsdaten ist nur über eine Authentifikation des Mitarbeiters möglich.

### 1.3 Allgemeine Anforderungen

Das Auftragsmanagement muss folgende Funktionen zur Verfügung stellen:

* Login mit Benutzername und Passwort
* Anstehende Serviceaufträge anzeigen (Liste)
* Bestehende Serviceaufträge mutieren. Dazu stehen folgende Stati zu Verfügung: Offen, InArbeit und abgeschlossen
* Aufträge löschen (ggf. bei Stornierung)

Die Informationen zur Online-Anmeldung, welche bereits realisiert wurde, müssen ggf. bei Bedarf wie folgt ergänzt werden.

* Kundenname
* E-Mail
* Telefon
* Priorität
* Dienstleistung (Angebot), siehe nachfolgende Auflistung. Pro Serviceauftrag kann immer nur eine Dienstleistung zugeordnet werden.

Die Firma bietet folgende Dienstleistungen (Angebot) an:

* Kleiner Service
* Grosser Service
* Rennski-Service
* Bindung montieren und einstellen
* Fell zuschneiden
* Heisswachsen

## 2. Planen

### 2.1 Projektstruktur & Technologien

* **Backend**: ASP.NET Core Web API
* **Datenbank**: MySQL (Database First mit Entity Framework Core)
* **Authentifizierung**: JWT-Token
* **Testumgebung**: Postman für API-Tests
* **Codeverwaltung**: GitHub

### 2.2 Zeitplanung

|  |  |
| --- | --- |
| **Aufgabe** | **Geschätzte Dauer** |
| Datenbankdesign | 2h |
| API-Entwicklung | 4h |
| Authentifizierung & Rollenverwaltung | 2h |
| API-Dokumentation (Swagger) | 1h |
| Testplanung & Umsetzung | 3h |
| Fehlerbehebung & Optimierung | 2h |
|  |  |

## 3. Entscheiden

### 3.1 Datenbankdesign

Die Datenbank wurde basierend auf den Anforderungen in folgende Tabellen unterteilt:

* Users: Speichert die Mitarbeiterdaten.
* UserRoles: Weist den Benutzern Rollen zu (Admin oder Mitarbeiter).
* Orders: Enthält die Aufträge mit Kundeninformationen.
* Service: Definiert die angebotenen Dienstleistungen.

### 3.2 Warum Database First?

* Die bestehende Struktur konnte direkt als Grundlage verwendet werden.
* Einfachere Verwaltung der Datenbank.

## 4. Realisieren

### 4.1 Implementierung der API

* Erstellung der Datenbank mit MySQL.
* Implementierung von CRUD-Funktionen für Bestellungen.
* Authentifizierung mit JWT:
  + Login-Endpoint erstellt.
  + Token-Generierung mit Benutzername & Passwort.
  + Berechtigungen über Middleware gesteuert.
* API-Dokumentation mit Swagger umgesetzt.
* Logging für Fehleranalyse hinzugefügt.

### 4.2 Fehlersuche & Debugging

* Problem: Unauthorized Access beim Löschen von Bestellungen.
  + Ursache: User.Identity.Name war null, da keine korrekte Authentifizierung stattfand.
  + Lösung: Sicherstellen, dass der Token im Authorization-Header übergeben wird.

## 5. Kontrollieren

### 5.1 Testfälle (Postman)

|  |  |  |
| --- | --- | --- |
| Testfall | Erwartetes Ergebnis | Status |
| Login mit gültigen Daten | Erfolgreiche Authentifizierung & JWT-Token erhalten | ✅ |
| Login mit falschen Daten | Fehlermeldung "Unauthorized" | ✅ |
| Bestellung erstellen | Bestellung wird gespeichert & ID zurückgegeben | ✅ |
| Bestellung ohne Authentifizierung löschen | Fehlermeldung "Unauthorized" | ✅ |
| Bestellung mit Admin-Login löschen | Bestellung wird entfernt | ✅ |

## 6. Auswerten

### 6.1 Erkenntnisse

* **Fehleranalyse ist essenziell:** Ein falsches Rollenmanagement kann zu unerwarteten Zugriffsbeschränkungen führen.
* JWT-Authentifizierung erfordert korrekte Middleware-Konfiguration.
* Swagger erleichtert die API-Dokumentation erheblich.

### 6.2 Lessons Learned

* Bessere Planung der Authentifizierung von Anfang an.
* Mehr Tests während der Entwicklung durchführen.
* Fehlerlogging von Anfang an integrieren.

### 6.3 Nächste Schritte

* Implementierung eines **Rate-Limits** zur Absicherung gegen Brute-Force-Angriffe.
* **Optimierung der Fehlerbehandlung**, um dem Benutzer bessere Feedback-Meldungen zu geben.

## 7. Fazit

Die Umsetzung des Backends für das Ski-Service Management war erfolgreich. Die API bietet eine strukturierte Verwaltung der Bestellungen und eine sichere Authentifizierung für Mitarbeiter. Durch die Implementierung von JWT und eine detaillierte Fehlerprotokollierung konnten Sicherheitslücken minimiert werden. In Zukunft sollten noch bessere Mechanismen zur Benutzerverwaltung und Performance-Optimierungen implementiert werden.