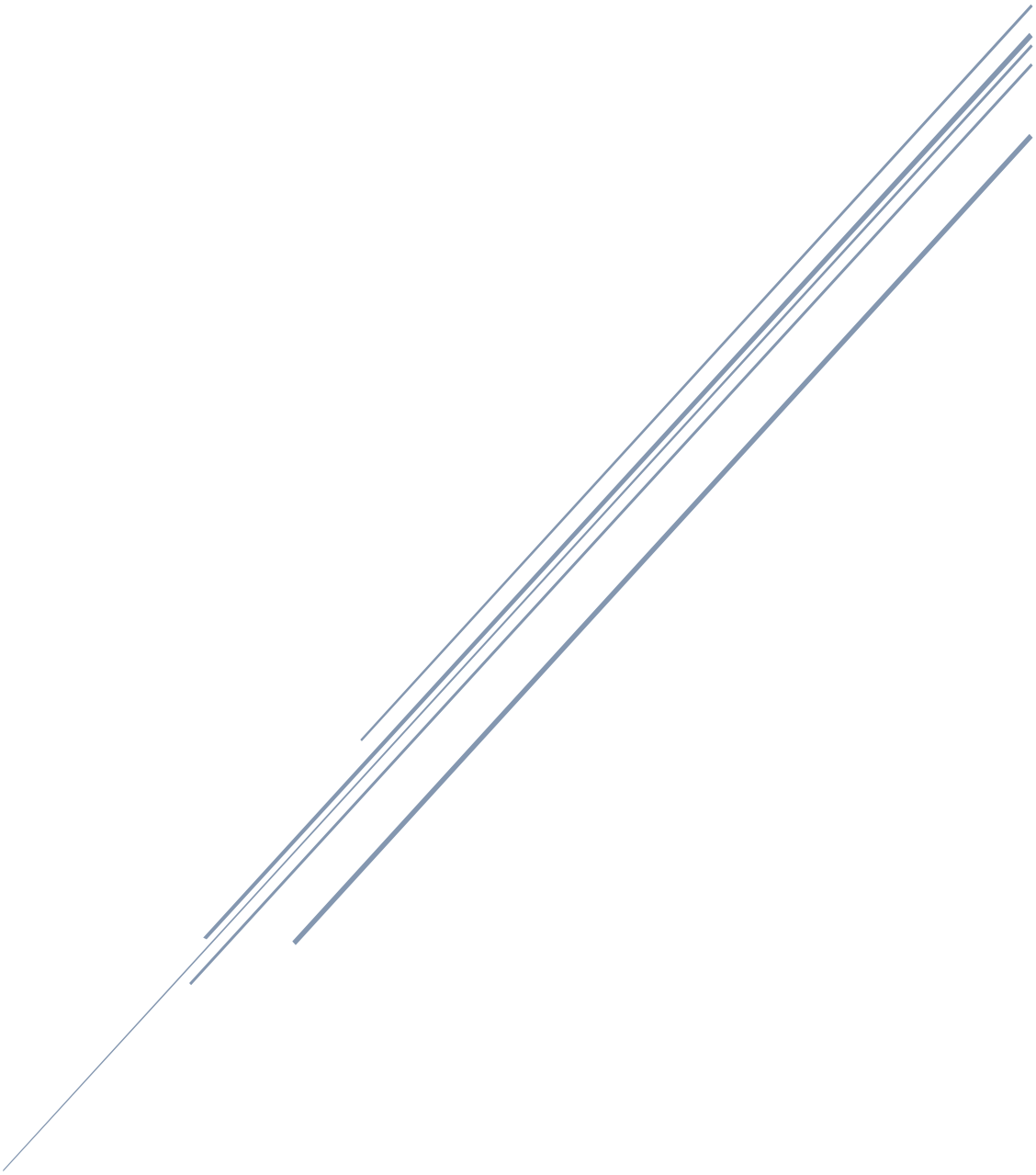
PROJEKTDOKUMENTATION PRAXISARBEIT MODUL 165

Ilia Kalygin



Inhaltsverzeichnis

Informieren

Ausgangssituation

Die Firma Jetstream-Service führt als KMU in der Wintersaison Skiservicearbeiten durch und hat in den letzten Jahren grosse Investitionen in eine durchgängige digitale Auftragsanmeldung und Verwaltung, bestehend aus einer datenbankbasierender Web-Anmeldung und Auftragsverwaltung getätigt. Aufgrund guter Auftragslage hat sich die Geschäftsführung für eine Diversifizierung mit Neueröffnungen an verschiedenen Standorten entschieden.

Die bis anhin eingesetzte relationale Datenbank genügt den damit verbundenen Ansprüchen an Datenverteilung und Skalierung nicht mehr. Um einerseits den neuen Anforderungen gerecht zu werden sowie anderseits Lizenzkosten einzusparen, soll im Backend der Anwendung die Datenbank auf ein NoSQL Datenbanksystem migriert werden.

Anforderungen

Nr.	Beschreibung
A1	Datenbasis aus relationaler Datenbank vollständig nach NoSQL migriert
A2	Benutzerkonzept mit min. 2 Benutzeranmeldungen mit verschiedenen Berechtigungsstufen implementiert.
A3	Für die Web-API Applikation muss ein eigener Datenbankbenutzerzugang mit eingeschränkter Berechtigung (DML) zur Verfügung gestellt werden
A4	Schema für Datenkonsistenz implementiert
A5	Datenbank Indexe für schnelle Ausführung von Suchabfragen implementiert
A6	Backup und Restore Möglichkeiten umgesetzt (Skript-Dateien)
A7	Vollständige Datenbankmigration mittels Skript-Dateien realisiert
A8	Das Web-API Projekt (CRUD) komplett auf NoSQL Datenbanksystem migriert
A9	Datenmodell vollständig dokumentiert, inkl. Grafik zum Datenmodell
A10	Einfaches Testprojekt in Postman erstellt.
A11	Das Softwareprojekt ist über ein Git-Repository zu verwalten.
A12	Ganzes Projektmanagement muss nach IPERKA dokumentiert sein

Planen

Projektplan

Aktivität	Zeit in h
Projektstart und Vorbereitung	2
Anforderungsanalyse	1
Planung und Konzeption	2
Design und Entwicklung	5
Weiterentwicklung und Optimierung	3
Erstellen einer Benutzeranleitung	1
Finalisierung	2
Dokumentation	3
Total:	**19**

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.

Datenbank-Management und Entwicklung

- **MongoDB Shell**: Eine interaktive JavaScript-Shell für MongoDB, die für die Verwaltung der Datenbank und das Ausführen von Datenbankabfragen verwendet wird. Sie ist ideal für die Arbeit mit MongoDB-Datenbanken und kann für das direkte Ausführen von Befehlen, das Testen von Abfragen und für administrative Aufgaben genutzt werden.

- **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

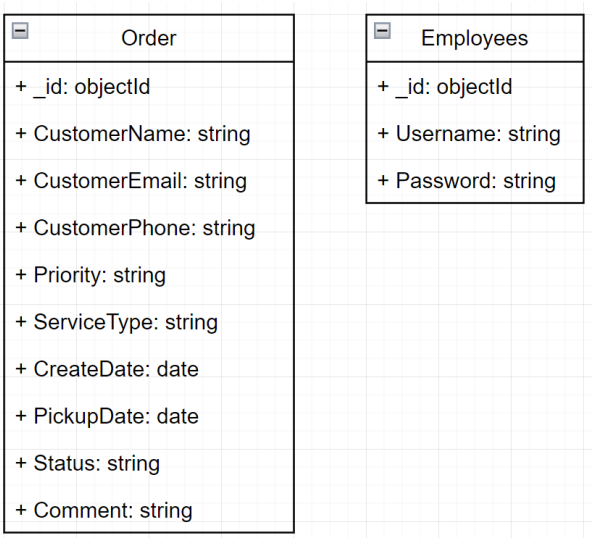
MongoDB oder Neo4j?

In der Entscheidungsphase unseres Projekts zur Ski-Service-Auftragsverwaltung standen wir vor der Wahl zwischen der dokumentenorientierten NoSQL-Datenbank MongoDB und der graphenbasierten Datenbank Neo4j. Nach gründlicher Analyse der spezifischen Projektanforderungen, insbesondere der Notwendigkeit, komplexe Beziehungen und Verbindungen effizient zu verwalten, haben wir uns für MongoDB entschieden. Ausschlaggebend waren hierbei die Stärken von MongoDB in Bezug auf Flexibilität, Skalierbarkeit und die gut etablierte Community. Zudem bietet MongoDB robuste Unterstützung für horizontales Skalieren und eine hohe Leistungsfähigkeit bei Abfragen und Datenmanipulation, was es zu einer sicheren Wahl für unsere Anforderungen macht.

Realisieren

Datenmodell

Die Orders-Kollektion speichert alle relevanten Informationen zu den Serviceaufträgen. Jeder Datensatz in dieser Kollektion umfasst:



- CustomerName: Name des Kunden, der den Serviceauftrag erteilt hat.
- CustomerEmail: E-Mail-Adresse des Kunden für Kommunikationszwecke.
- CustomerPhone: Telefonnummer des Kunden.
- Priority: Priorität des Auftrags, um dringende Fälle zu kennzeichnen.
- ServiceType: Spezifischer Typ des angeforderten Services.
- CreateDate: Datum, an dem der Auftrag erstellt wurde.
- PickupDate: Geplantes Abholdatum für den Servicegegenstand.
- Status: Aktueller Status des Auftrags (z.B. offen, in Bearbeitung, abgeschlossen).
- Comment: Zusätzliche Anmerkungen oder spezielle Anforderungen.

Für die Orders-Kollektion haben wir ein JSON Schema implementiert, das die Datenintegrität gewährleistet, indem es die erforderlichen Felder und deren Datentypen definiert. Zusätzlich wurde ein Index auf dem CustomerName-Feld erstellt, um die Abfrageeffizienz zu verbessern.

Die Employees-Kollektion enthält Informationen zu den Mitarbeitern, die das System nutzen. Jeder Mitarbeiterdatensatz beinhaltet:

- Username: Eindeutiger Benutzername für den Systemzugang.
- Password: Passwort für den Benutzerzugang, gespeichert in einer sicheren, verschlüsselten Form.

Auch hier wurde ein JSON Schema zur Validierung der Datensätze verwendet, und es wurde ein Index auf dem Username-Feld angelegt, um schnelle Zugriffe und Authentifizierungsprozesse zu ermöglichen.

Datenbank erstellen

Im Realisierungsabschnitt der Projektarbeit zur Ski-Service-Auftragsverwaltung wurde die Datenbankstruktur in MongoDB mit Einsatz von JSON Schema für die Validierung der Datenstrukturen in den Collections Orders und Employees implementiert. JSON Schema ermöglicht es, genaue Anforderungen an die Daten in den Dokumenten beider Collections zu definieren, einschliesslich erforderlicher Felder, Datentypen und Einschränkungen für Feldwerte. Dies stellt sicher, dass die eingefügten oder aktualisierten Daten den festgelegten Regeln entsprechen, was die Datenintegrität und -qualität im gesamten System verbessert.

```
-- Datenbank erstellen
use SkiServiceManagement;

-- Collection Employees erstellen
db.createCollection("Employees", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["Username", "Password"],
      properties: {
        _id: {
          bsonType: "objectId"
        },
        Username: {
          bsonType: "string",
          maxLength: 50
        },
        Password: {
          bsonType: "string",
          maxLength: 50
        }
      },
      additionalProperties: false
    }
  },
  validationLevel: "strict",
  validationAction: "error"
});

-- Test Datensatz in Employees einfügen
db.Employees.insertMany([
  { "Username": "admin", "Password": "admin" }
]);
```

Pogrammierung WEB API

Im Realisierungsabschnitt der Projektarbeit wurde die Programmierung der Web API fokussiert. Dabei dienten Beispiele aus dem Unterricht und ein spezifisches Tutorial von Microsoft (https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-mongo-app) als Grundlage. Diese Ressourcen boten einen umfassenden Einblick in die Erstellung von Web APIs mit ASP.NET Core und die Integration mit MongoDB, einschliesslich der Implementierung von Authentifizierungsmechanismen. Durch die Anwendung dieser Konzepte konnte eine robuste und sichere API entwickelt werden, die als Kernstück für die Verwaltung von Ski-Service-Aufträgen dient.

Frontend Verbinden

Im Realisierungsabschnitt zur Verbindung des Frontends wurde das Frontend aus Modul 322 modifiziert, um die Anforderungen der Ski-Service-Auftragsverwaltung zu erfüllen. Eine wesentliche Änderung war die Anpassung der Modelklasse Order.cs, insbesondere die Modifikation des Datentyps für OrderID von int zu string. Diese Anpassung war notwendig, um die Kompatibilität mit MongoDB zu gewährleisten, da MongoDB eindeutige Identifikatoren (_id) standardmässig als Strings im Format der ObjectId speichert. Durch diese Änderung konnte eine nahtlose Integration zwischen dem Frontend und der MongoDB-basierten Backend-Datenbank erreicht werden.

Migration der Datenbasis von SQL nach NoSQL

Die Migration wurde in mehrere Schlüsselschritte unterteilt, um eine reibungslose Übertragung und Integration der Daten zu gewährleisten:

1. Extraktion der Daten aus der SQL-Datenbank: Die Daten wurden aus der bestehenden SQL-Datenbank (MySQL) unter Verwendung von SQL-Abfragen extrahiert. Die extrahierten Daten umfassten Tabellen, die relevante Informationen für das SkiServiceManagement-System enthielten.

2. Transformation der Daten in das JSON-Format: Die extrahierten Daten wurden in das JSON-Format transformiert, um die Kompatibilität mit MongoDB zu gewährleisten. Dieser Schritt wurde durch ein Python-Skript automatisiert, das jede Zeile der SQL-Daten in ein JSON-Objekt konvertierte.

3. Import der Daten in MongoDB: Die transformierten Daten wurden in die MongoDB-Datenbank importiert. Dieser Vorgang nutzte die mongoimport-Funktion von MongoDB, um die JSON-Dateien effizient in die entsprechenden Collections einzufügen.

Buckup- und Restore Scripts

Das Backup-Skript ist dafür konzipiert, eine vollständige Sicherung der Datenbank zu erstellen. Es nutzt das mongodump-Werkzeug, das Teil der MongoDB-Distributionspakete ist. Dieses Werkzeug ermöglicht es, alle Daten und Indizes der SkiServiceManagement-Datenbank in ein binäres Exportformat zu exportieren. Die Sicherung wird in einem vordefinierten Verzeichnis abgelegt, welches regelmässig auf externe Sicherungsmedien übertragen werden sollte, um Datenverlusten durch Hardwareausfälle oder ähnliches vorzubeugen.

Das Restore-Skript verwendet das mongorestore-Tool, um Daten aus einem Backup in die SkiServiceManagement-Datenbank zu importieren. Bevor das Restore durchgeführt wird, sollte sichergestellt werden, dass die Datenbank, in die die Daten wiederhergestellt werden sollen, entweder leer ist oder dass die Wiederherstellung die aktuellen Daten überschreiben darf. Dieses Skript ist besonders nützlich bei der Wiederherstellung von Daten nach einem unerwarteten Datenverlust oder beim Transfer von Daten zwischen verschiedenen Umgebungen.

Kontrollieren

Testprojekt mit 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

Es wurde eine umfassende Bewertung der umgesetzten Ski-Service-Auftragsverwaltung durchgeführt. Diese Bewertung umfasste die Analyse der erreichten Ziele im Vergleich zu den ursprünglichen Projektanforderungen, die Effektivität der implementierten Lösungen sowie die Identifizierung von Bereichen mit Potenzial für zukünftige Verbesserungen. Durch die Reflexion des gesamten Projektverlaufs konnte das Team wertvolle Einsichten gewinnen, die für die Weiterentwicklung des Projekts und die Planung zukünftiger Projekte von Bedeutung sind.