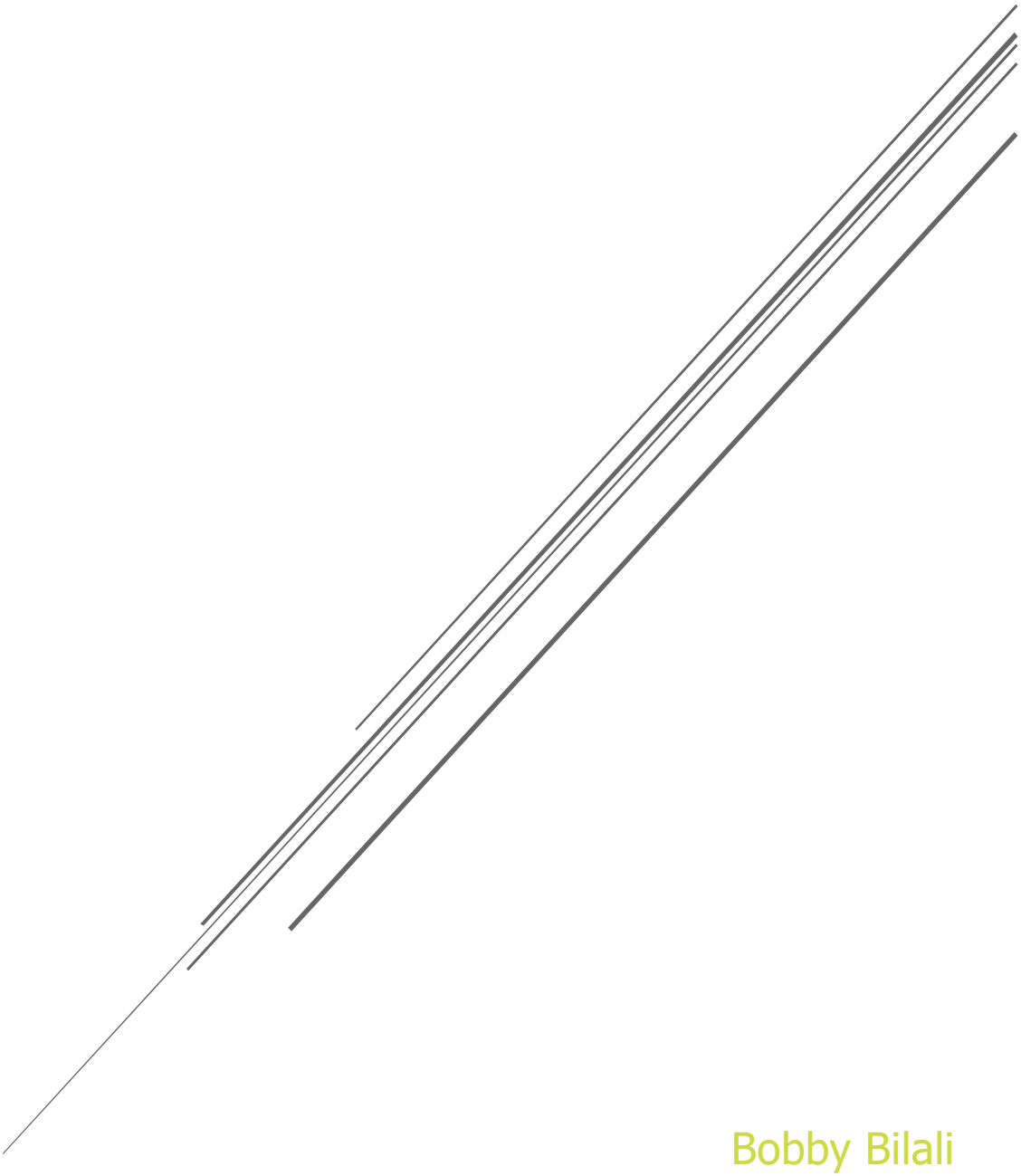
ipso! Bildung seit 1876

MODUL 165

Projektdokumentation



Bobby Bilali

JETSTREAM API NOSQL BACKEND

ipso! Bildung seit 1876

Inhaltsverzeichnis

1 Vesrsionsverzeichnis

Version	Autor	Datum	Änderung
1.0	Bobby Bilali	24.01.2024	Erstellung des Dokuments
1.1	Bobby Bilali	26.01.2024	Erstellung der Planung
1.2	Bobby Bilali	30.01.2024	Grobe Fertigstellung des Dokuments
1.3	Bobby Bilali	02.02.2024	Rechtschreibprüfung und Textkorrekturen

Tabelle 1: Versionsverzeichnis

2 Executive Summary: "Ski-Service Management"

Jetstream-Service, ein KMU im Ski-Service, plant die Expansion an neuen Standorten aufgrund steigender Auftragszahlen. Um den wachsenden Anforderungen gerecht zu werden und Lizenzkosten zu senken, wird das Backend-System auf ein NoSQL-Datenbanksystem migriert. Dieses Projekt umfasst die Datenbankmigration, Anwendungsentwicklung und Integrationstests, um eine effiziente Auftragsverwaltung sicherzustellen und die Expansion zu unterstützen.

3 Einleitung

Dieses Dokument fungiert als grundlegende Projektdokumentation für das NoSQL-Migrationsprojekt des Backend-Systems der Jetstream-Service Webseite. Das Hauptziel dieses Projekts ist die Umstellung der technischen Infrastruktur, um den Anforderungen der neuen Webseite gerecht zu werden. Ein wesentlicher Schwerpunkt liegt auf der Migration zu einem NoSQL-Datenbanksystem, um die Verwaltung von Serviceaufträgen zu optimieren und eine reibungslose Integration in die bestehende Webplattform sicherzustellen.

4 Ausgangslage

Die Firma Jetstream-Service, ein KMU im Bereich der Wintersaison, hat in den letzten Jahren erhebliche Investitionen in die Digitalisierung ihrer Skiservicearbeiten getätigt. Dies umfasst eine durchgängige digitale Auftragsanmeldung und -verwaltung, bestehend aus einer datenbankbasierten Web-Anmeldung und Auftragsverwaltung. Angesichts einer florierenden Auftragslage hat sich die Geschäftsführung entschieden, das Unternehmen durch die Eröffnung neuer Standorte zu diversifizieren.

Die bisher verwendete relationale Datenbank kann den wachsenden Anforderungen an Datenverteilung und Skalierbarkeit nicht mehr gerecht werden. Um diesen Herausforderungen zu begegnen und gleichzeitig Lizenzkosten einzusparen, plant das Unternehmen die Migration des Backend-Systems auf ein NoSQL-Datenbanksystem.

Das Teilprojekt konzentriert sich hauptsächlich auf den Backend-Bereich und umfasst die folgenden Aufträge, die gemäss IPERKA durchgeführt werden:

- **Datenbankdesign und Implementierung (NoSQL):** Entwicklung einer geeigneten Datenbankstruktur unter Verwendung eines NoSQL-Datenbanksystems.
- **Datenmigration (SQL ➔ NoSQL):** Sichere Übertragung von bestehenden Daten aus dem relationalen SQL-System in das neue NoSQL-Datenbanksystem.
- **Migration WebAPI Projekt:** Anpassung und Migration des bestehenden WebAPI-Projekts, um die Kommunikation mit der neuen Datenbank zu ermöglichen.
- **Testprojekt / Testplan:** Entwicklung eines umfassenden Testplans und Durchführung von Tests, um sicherzustellen, dass die Anwendung ordnungsgemäss funktioniert.

- **Realisierung der kompletten Anwendung:** Umsetzung der gesamten Anwendung gemäss den definierten Anforderungen, um eine effiziente Auftragsverwaltung zu gewährleisten.
- **Durchführung Integrationstest:** Test der Integration zwischen Backend und Frontend, um sicherzustellen, dass die Lösung nahtlos funktioniert und mit vorhandenen Frontend-Lösungen kompatibel ist.

Die erfolgreiche Umsetzung dieses Teilprojekts wird Jetstream-Service in die Lage versetzen, sein Auftragsmanagement effizienter zu gestalten und gleichzeitig Lizenzkosten zu reduzieren. Dies legt die Grundlage für eine reibungslose Expansion des Unternehmens und steigert die Wettbewerbsfähigkeit in der Branche.

5 Anforderungen

Für das Auftragsmanagement sind folgende Kernfunktionen erforderlich:

- **Authentifizierung:** Sicheres Einloggen mit Benutzername und Passwort.
- **Auftragsübersicht:** Anzeigen von aktuellen Serviceaufträgen in einer Liste.
- **Auftragsstatus:** Möglichkeit zur Aktualisierung des Status von Serviceaufträgen (Offen, In Arbeit, Abgeschlossen).
- **Auftragsverwaltung:** Löschen von Aufträgen bei Bedarf, z.B. bei Stornierung.

Des Weiteren müssen die Informationen der Online-Anmeldung, die bereits realisiert wurden, bei Bedarf um zusätzliche Details ergänzt werden können:

- **Kundeninformationen:** Name, E-Mail, Telefonnummer.
- **Priorisierung:** Einstufung der Dringlichkeit des Auftrags.
- **Dienstleistungen:** Zuordnung einer der angebotenen Dienstleistungen zu jedem Serviceauftrag, wobei jede Dienstleistung aus einer Liste von Angeboten (Kleiner Service, Grosser Service, Rennski-Service, Bindung montieren und einstellen, Fell zuschneiden, Heisswachsen) ausgewählt werden kann.

Diese Anforderungen bilden die Grundlage für die Entwicklung des Backend und stellen sicher, dass das System den operativen Bedürfnissen des Unternehmens gerecht wird.

5.1 Zusammenfassung der 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

Abbildung 1: Zusammenfassung Anforderungen

5.2 Zusätzliche Anforderungen

Nr.	Beschreibung
AO1	Automatisiertes Backup-Konzept durchgeführt u. implementiert.
AO2	Komplexe Schema Validierungen umgesetzt (Referenzen, enum, min, max. usw)
AO3	Datenmigrationsskripte zu den RDBMS nach NoSQL realisiert
AO4	Komplexes Datenmodell mit mehr als 6 Grundtypen (Collection / Labels) implementiert
AO5	Komplexe statistische Auswertungsabfragen realisiert

Abbildung 2: Zusätzliche Anforderungen

6 Informieren

In dieser Phase liegt der Fokus auf der umfassenden Informationsbeschaffung, die für die Entwicklung des Backend-Systems für Jetstream-Service in der NoSQL-Umgebung von entscheidender Bedeutung ist. Hierzu gehört die gründliche Analyse der technischen Anforderungen, die Erkundung der erforderlichen Systemarchitektur und die Bewertung von Sicherheitsaspekten.

Das Hauptziel besteht darin, ein tiefes Verständnis für die technischen Herausforderungen und die spezifischen Bedürfnisse des NoSQL-Migrationsprojekts zu entwickeln. Dies bildet die Grundlage für eine erfolgreiche Umstellung der Datenbankstruktur und gewährleistet, dass die zukünftige NoSQL-Infrastruktur den operativen Anforderungen von Jetstream-Service gerecht wird.

6.1 Vorgabe/Anforderungen

Im Rahmen des NoSQL-Projekts ergeben sich folgende Kernelemente und Hauptziele:

Umstellung auf eine stabile und sichere Datenbankstruktur zur Verwaltung von Kundeninformationen und Serviceaufträgen.

Schaffung einer effizienten Web-API, die eine nahtlose Kommunikation zwischen dem Frontend und der Datenbank gewährleistet.

Integration von Authentifizierungs- und Sicherheitsmassnahmen, um die Vertraulichkeit sensibler Kundendaten zu gewährleisten.

Die Backend-Umstellung zielt darauf ab, eine leistungsfähige und skalierbare Architektur zu schaffen, die den spezifischen Anforderungen von Jetstream-Service entspricht. Darüber hinaus wird das Backend auf die nahtlose Integration in bestehende Systeme sowie auf zukünftige Erweiterungen vorbereitet.

6.2 Zusätzliche Anforderungen

Es müssen mindestens zwei der oberen zusätzliche Anforderungen umgesetzt werden:

- Automatisiertes Backup-Konzept durchgeführt u. implementiert.
- Komplexes Datenmodell mit mehr als 6 Grundtypen (Collection / Labels) implementiert.

7 Planen

In der Planungsphase geht es darum, die in der Informationsphase gesammelten Daten und Erkenntnisse in einen strukturierten und umsetzbaren Plan zu überführen. Diese Phase ist entscheidend, um die Grundlage für die erfolgreiche Realisierung des Projekts zu legen.

7.1 Zeitplanung

Das Projekt findet innerhalb von 10 Tagen statt und wird in mehreren Phasen durchgeführt

Projektphase	Geplante Zeit
Informieren	4h
Planung, Entwurf, Entscheidung	10h
Realisierung	15h
Abnahme	3h
Dokumentation	10h
Gesamt	42h

Tabelle 2: Grobe Planung

7.2 Projektplanung

Ein Projektplan ist ein unverzichtbares Instrument im Projektmanagement, das für die Definition und Visualisierung der Struktur und des Umfangs eines Projekts verwendet wird. Durch den Plan wird das gesamte Projekt in überschaubare Abschnitte oder Arbeitspakete unterteilt, was die Planung, Überwachung und Steuerung vereinfacht.

Der PSP ermöglicht es, den Projektumfang klar zu definieren und die benötigten Ressourcen für jede Phase festzulegen.

Phase	**Startdatum**	**Enddatum**	**StdSoll**	**StdIst**
Informieren	24.01.2024	25.01.2024		
Anforderungen	24.01.2024	25.01.2024	1h	1h
NoSQL-Datenbanken Migration	24.01.2024	25.01.2024	1h	1h
Backendmigration	24.01.2024	25.01.2024	1h	1h
WebAPI Migration	24.01.2024	25.01.2024	1h	1h
Planen	26.01.2024	27.01.2024		
Web API Migration	26.01.2024	27.01.2024	0.5h	0.5h
Datenbankmodell	26.01.2024	27.01.2024	0.5h	0.5h
Datenbank Migration	26.01.2024	27.01.2024	1h	1h
Backup und Restore	26.01.2024	27.01.2024	1h	1h
Testing	26.01.2024	27.01.2024	1h	1h
Entscheiden	28.01.2024	29.01.2024		
Welche Tools einsetzen	28.01.2024	29.01.2024	0.5h	0.5h
Welche Zusatz Anforderungen	28.01.2024	29.01.2024	0.5h	0.5h
Anpassung Web API	28.01.2024	29.01.2024	1h	1h
Datenbankstruktur Umsetzung	28.01.2024	29.01.2024	1h	1h
Benutzerkonzept Umsetzung	28.01.2024	29.01.2024	0.5h	0.5h
Schema Erweiterungen Umsetzung	28.01.2024	29.01.2024	1h	1h
Index-Strukturen Umsetzung	28.01.2024	29.01.2024	0.5h	0.5h
Backup und Restore Umsetzung	28.01.2024	29.01.2024	0.5h	0.5h
Testing	28.01.2024	29.01.2024	0.5h	0.5h
Realisieren	30.01.2024	01.02.2024		
Datenbankmodell	30.01.2024	31.02.2024	2h	5h
Datenbank mit DatabaseFirst	30.01.2024	31.02.2024	3h	5h
Backend-Migration	30.01.2024	31.02.2024	3h	8h
Zusatzanforderungen	31.01.2024	01.02.2024	3h	2h
Skripte für Migration	31.01.2024	01.01.2024	4h	5h
Kontrollieren	01.02.2024	02.02.2024		
Web API Postman	01.02.2024	02.02.2024	1h	2h
HTTP-Methoden	01.02.2024	02.02.2024	0.5h	1h
Zusatzaufgaben	01.02.2024	02.02.2024	1h	2h
xUnit Test Projekt	01.02.2024	02.02.2024	0.5h	1h
Auswerten	02.02.2024	02.02.2024		
Fazit	02.02.2024	02.02.2024	5h	5h
Dokumentation Fertigstellen	02.02.2024	02.02.2024	5h	5h
		Total:	42h	55h

Tabelle 3: Planung und Phasen

7.3 Zu verwendende Tools

Für die Entwicklung und Verwaltung des Projekts werden verschiedene Tools verwendet:

GitHub: Dient als zentrales Repository für den Quellcode, ermöglichte Versionskontrolle.
Visual Studio und Visual Studio Code: Diese integrierten Entwicklungsumgebungen (IDEs) werden für die Programmierung und das Debugging des Codes eingesetzt.
MongoDB: Wird verwendet, um die Datenbanken zu verwalten, zu konfigurieren und zu testen.
SwaggerUI: Dient zur Dokumentation und zum Testen der RESTful APIs.
Postman: Wird eingesetzt, um die Entwicklung und das Testen der APIs zu erleichtern.

7.4 Systemarchitektur

Die Systemarchitektur des JetstreamApi-Projekts ist klar in drei Schichten aufgeteilt, die die Kommunikation und den Datenfluss zwischen dem Client-Browser, dem Backend-Server und der Datenbank verdeutlichen.

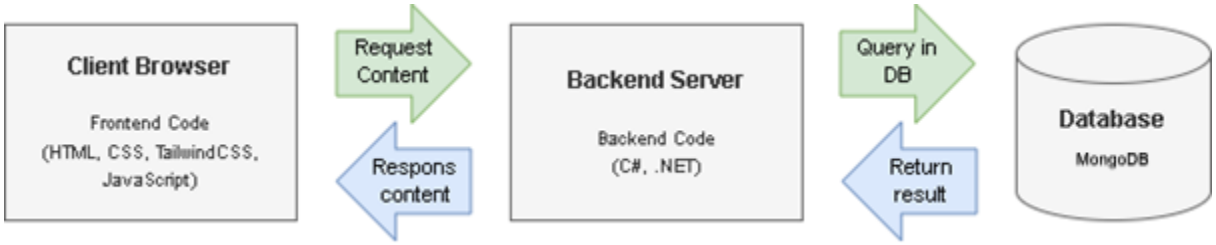


Abbildung 3: Systemarchitektur

- **Client-Browser:** Nutzer interagieren mit der Anwendung über einen Webbrowser, in dem der Frontend-Code ausgeführt wird. Dieser besteht aus HTML, CSS und JavaScript, wobei auch Frameworks wie TailwindCSS zum Einsatz kommen.
- **Backend-Server:** Wenn Inhalte angefordert werden, verarbeitet der Backend-Server diese Anfragen. Er ist mit C# und .NET programmiert und stellt die Verbindung zur Datenbank her, um erforderliche Daten abzufragen oder zu manipulieren.
- **Datenbank:** Die MongoDB NoSQL-Datenbank speichert und verwaltet alle erforderlichen Daten. Sie empfängt Abfragen vom Backend-Server, führt diese aus und gibt die Ergebnisse zurück, die dann an den Client-Browser gesendet werden können.

Diese strukturierte Architektur ermöglicht eine klare Trennung der Verantwortlichkeiten und erleichtert die Wartung und Skalierung der Anwendung in MongoDB.

7.4.1 Datenbankstruktur

Die Datenbank ist so konzipiert, dass sie effizient Kundendaten und Serviceaufträge verwalten kann. Es wird ein relationales Datenbankschema verwendet, das die Integrität und Sicherheit der Daten gewährleistet.

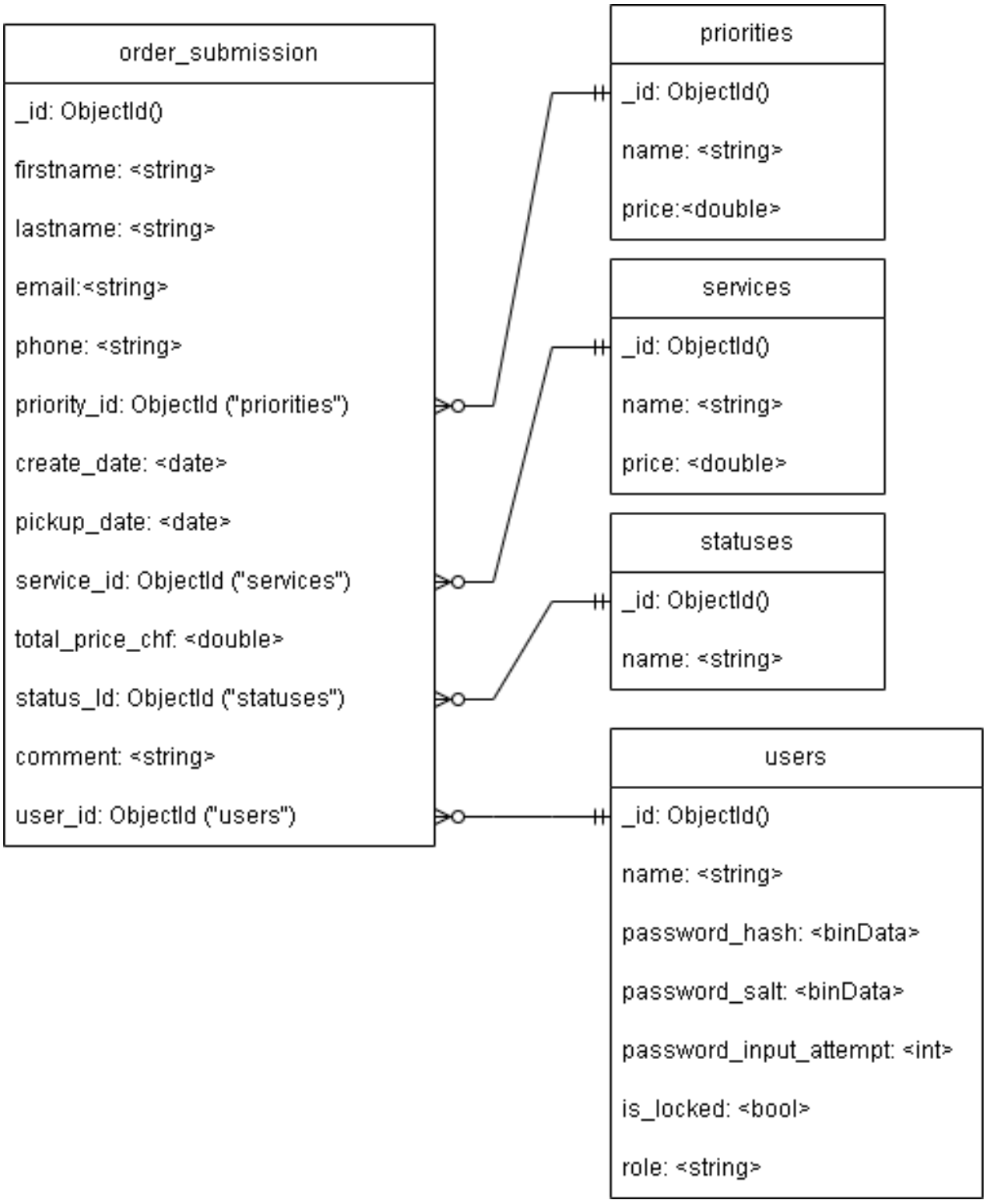


Abbildung 4: Datenbankdiagramm

7.4.2 Web-API

Die Web-API ist das Herzstück des Backends und ermöglicht eine nahtlose Kommunikation zwischen dem Frontend und der Datenbank. Sie ist auf Effizienz und Sicherheit ausgelegt, um eine schnelle Verarbeitung von Anfragen und den Schutz sensibler Kundendaten zu gewährleisten.

7.5 Technologien

Die Auswahl der Technologien für die Backend-Entwicklung erfolgt mit dem Ziel, ein robustes und skalierbares System zu schaffen, das die spezifischen Anforderungen des JetstreamApi-Projekts erfüllt. Die gewählten Technologien umfassen:

- **Datenbanktechnologie: MongoDB NoSQL**
  Für die Datenverwaltung wird MongoDB als NoSQL-Datenbankmanagementsystem verwendet. Es bietet Flexibilität bei der Speicherung und Abfrage von Daten, hohe Skalierbarkeit sowie eine einfache Integration in moderne Anwendungen.
- **Serverseitige Programmierung: .NET Core**
  Als Plattform für die serverseitige Programmierung wird .NET Core verwendet. .NET Core ist besonders geeignet für die Entwicklung von Webanwendungen, da es eine hohe Performance, Sicherheit und Flexibilität bei der Integration verschiedener Dienste bietet.

7.6 Validierung und Sicherheit

Ein wesentlicher Aspekt der Backend-Entwicklung ist die Implementierung von Validierungs- und Sicherheitsmechanismen, insbesondere für die Verarbeitung von Kundeninformationen. Dies umfasst:

- **Validierung:** Strenge Überprüfung der Eingabedaten, um die Korrektheit und Vollständigkeit der Kundeninformationen zu gewährleisten.
- **Sicherheit:** Umsetzung von Sicherheitsmassnahmen wie Authentifizierung, Autorisierung und Datenverschlüsselung, um den Schutz der Kundendaten zu gewährleisten.
- **Backup:** Umsetzung von Manuellen und Automatisierten Backupskripts die entweder vom User manuell ausgeführt werden oder wenn ein automatisiertes Backup erwünscht ist wird ein Windows Task erstellt der immer um 4:00 Uhr morgens das backup.ps1 Skript ausführt.

8 Entscheiden

In der Entscheidungsphase des Backend-Entwicklungsprozesses des JetstreamApi-Projekts werden strategische Entscheidungen getroffen, um die Ziele des Projekts zu erreichen und eine effiziente, sichere und benutzerfreundliche Anwendung zu gewährleisten. Die Entscheidungen werden auf Basis der gesammelten Informationen aus der Ausgangslage und den definierten Anforderungen getroffen und umfassten folgende Schlüsselbereiche:

Technologie-Stack:

- **Wahl der Datenbanktechnologie:** MongoDB NoSQL wurde aufgrund der Anforderungen und seiner Leistungsfähigkeit, Zuverlässigkeit und Skalierbarkeit als Datenbanktechnologie ausgewählt.
- **Serverseitige Programmiersprache:** .NET Core wurde als serverseitige Plattform ausgewählt, da es eine Anforderung ist und eine hohe Performance, Sicherheit und Kompatibilität mit verschiedenen Systemen und Diensten bietet.

Architektur-Entscheidungen:

- **Entscheidung für eine modulare Architektur:** Um Flexibilität und Skalierbarkeit zu gewährleisten, wurde eine modulare Architektur gewählt, die die unabhängige Entwicklung und Wartung von Systemkomponenten ermöglicht.

- **API-Strategie:** Die Entscheidung fiel auf die Entwicklung von RESTful-APIs, um eine standardisierte und plattformunabhängige Kommunikation zwischen Frontend und Backend sicherzustellen.

Sicherheitskonzept:

- **Authentifizierungsmechanismen:** Starke Authentifizierungsverfahren wurden implementiert, um sicherzustellen, dass nur berechtigte Nutzer Zugriff auf die Systemfunktionen erhalten.
- **Datenverschlüsselung:** Es wurde entschieden, sensible Daten zu verschlüsseln, um Datenschutz und Compliance-Anforderungen zu erfüllen.

Auswahl der Entwicklungswerkzeuge:

- **Entwicklungsumgebungen:** Visual Studio und Visual Studio Code wurden aufgrund ihrer umfangreichen Funktionen und Unterstützung für .NET Core als primäre Entwicklungswerkzeuge gewählt.
- **API-Design und Testwerkzeuge:** SwaggerUI wird für das API-Design und Postman für das Testen der APIs ausgewählt, um eine effiziente Entwicklung und Wartung der API-Schnittstellen zu gewährleisten.
- **Versionskontrolle:** GitHub wird als zentrales Tool für Versionskontrolle verwendet.

Diese Entscheidungen bilden die Basis für die weitere Umsetzung und Entwicklung des Backend-Systems und sorgen dafür, dass die Anwendung den technischen und geschäftlichen Anforderungen gerecht wird.

9 Realisieren

Die Realisierungsphase setzt die geplanten und entschiedenen Architektur- und Anforderungsdetails in eine funktionierende Lösung um. Während dieser Phase wird das Backend-System entwickelt und die zugehörigen Tests erstellt. Die erzielten Schlüsselergebnisse umfassen:

9.1 Backend-Entwicklung

Die Backend-Entwicklung ist eine zentrale Säule des Gesamtsystems. Ziel ist es, eine robuste und sichere Server-Umgebung zu schaffen, die eine reibungslose Verarbeitung und Verwaltung der Serviceanfragen ermöglicht.

Aufbau der Backend-Infrastruktur:

- Einrichtung, Konfiguration und Vorbereitung der Datenbank auf MongoDB NoSQL Server.
- Implementierung der Geschäftslogik in .NET Core und Entwicklung der notwendigen API-Endpunkte unter Berücksichtigung der Sicherheitsanforderungen.

Datenbankdesign und -implementierung:

- Ursprünglich CodeFirst-Ansatz mit Entity Framework Core zur Schemaerstellung.
- Migration zu MongoDB mit DatabaseFirst-Ansatz für flexiblere Datenverwaltung.
- Erstellung und Verwaltung von Datenbankschema, Indizes und Daten mittels PowerShell-Skripten.

Projektstruktur:

Organisation des Codes in einer modularen Struktur zur Förderung der Wartbarkeit und Skalierbarkeit.

- **Controllers:** '**OrderSubmissionController**' und '**UsersController**' behandeln HTTP-Anfragen.
- **Data:** Beinhaltet Klassen und Konfigurationen für die Datenbankzugriffsschicht, einschliesslich des Datenbankkontexts zur Verwaltung der Datenbankdaten. Hier werden auch Seeds definiert, um Benutzer und Dummy-Serviceanfragen in der Datenbank zu erstellen.

- **DTOs:** Spezifizieren die Struktur für den Datenaustausch zwischen API und Clients.
- **Interfaces:** Definiert Verträge für die Services und Repository-Klassen, um die Abstraktion und lose Kopplung im Code zu fördern, was das Testen und die Wartung erleichtert.
- **Middleware:** Für zentrales Fehlermanagement und andere Middleware-Funktionen.
- **Models:** Definieren die Datenbankentitäten und Geschäftsobjekte.
- **Services:** Beinhalten die Geschäftslogik und den Datenzugriff.

Die Struktur fördert die Wiederverwendbarkeit von Code, erleichtert Tests und ermöglicht ein effizientes Teamwork.

API-Entwicklung:

- Entwicklung von RESTful APIs mit .NET Core unter Einhaltung der OpenAPI-Spezifikationen.
- Sicherstellung der API-Kompatibilität mit dem Frontend und anderen Diensten, die auf das Backend zugreifen.

9.2 Sicherheitsimplementierung

Ein umfassendes Sicherheitskonzept wird implementiert, um die Integrität und Schutz der Daten zu gewährleisten:

- Einsatz von HTTPS und Authentifizierungstoken zur Sicherstellung der sicheren Kommunikation.
- Verschlüsselungsverfahren zum Schutz sensibler Daten.

9.3 Testverfahren

Eine Teststrategie wird etabliert, um die Zuverlässigkeit des Backends zu gewährleisten:

- Unit-Tests für die Komponentenprüfung und Integrationstests zur Überprüfung der Systeminteraktionen.
- Nutzung von Postman für End-to-End-Tests und zur Sicherstellung des korrekten Datenflusses.

9.4 Dokumentation und Wartbarkeit

Die Dokumentation und Wartung des Systems werden mit folgenden Werkzeugen unterstützt:

- SwaggerUI zur detaillierten Dokumentation der API-Endpunkte.
- GitHub zur Versionskontrolle.

Die Realisierung des Backend-Systems wird iterativ durchgeführt, wobei kontinuierliches Feedback aus dem Testing und von den Stakeholdern zur stetigen Verbesserung und Anpassung des Systems beigetragen haben.

10 Kontrollieren

In der Phase werden umfangreiche Tests durchgeführt, um die Funktionalität, Sicherheit und Stabilität der entwickelten Lösung zu gewährleisten. Verschiedene Validierungstests werden etabliert, um sicherzustellen, dass die Anforderungen erfüllt werden und entsprechend den Spezifikationen funktionieren:

10.1 Validierungstests durchführen

Die Tests wurden mit xUnit und Postman durchgeführt. Der xUnit-Test konzentrierte sich ausschliesslich auf die Funktionalität des User-Logins. xUnit sollte für umfangreichere Tests genutzt werden, darauf wurde aus Zeitgründen jedoch verzichtet.

1. **Eingabevalidierung:**
   - **Namen-Validierung:** Überprüfung, dass Namen entsprechend den festgelegten Kriterien, ohne unerwünschte Leerzeichen, eingegeben werden.
   - **E-Mail-Validierung:** Einsatz von regulären Ausdrücken (Regex), um die Struktur der E-Mail-Adressen zu verifizieren.
   - **Telefonnummern-Validierung:** Sicherstellung, dass Telefonnummern in einem validen Format und entsprechend den internationalen Standards eingegeben werden.

2. **API-Kommunikation Postman Test (**Abbildung 8: Postman Collection Test**):**
   - Überprüfung der Funktionsfähigkeit und Stabilität der API-Endpunkte.
   - Testen der Datenübermittlung und -verarbeitung zwischen Frontend und Backend.
   - Validierung des korrekten Speicherns und Abfragens der Daten in der NoSQL-Datenbank.

3. **Sicherheitstests:**
   - Implementierung von Tests für Authentifizierungsverfahren, um die Sicherheit der Endpunkte zu gewährleisten.
   - Überprüfung von Verschlüsselungsprotokollen und Datensicherheit.

4. **Integrationstests:**
   - Testen der Interaktionen zwischen verschiedenen Backend-Komponenten und Diensten.
   - Überprüfung der korrekten Zusammenarbeit zwischen dem Datenbankmanagement, Geschäftslogik und API-Schicht.
5. **Benutzerakzeptanztests (UAT):**
   - Zusammenarbeit mit Stakeholdern, um die Backend-Funktionen in realen Anwendungsfällen zu testen.
   - Anpassung des Backends auf Basis des direkten Feedbacks von Endnutzern, um die Usability zu verbessern.

Die Durchführung dieser Tests und Überprüfungen war unerlässlich, um zu garantieren, dass das System reibungslos funktioniert und den Anforderungen genügt. Nur durch diese sorgfältige Kontrolle kann sichergestellt werden, dass das Backend-System bereit für den produktiven Einsatz ist.

11 Auswerten

Die Auswertungsphase dient dazu, nach Fertigstellung des Projekts eine umfassende Bewertung der verschiedenen Aspekte des entwickelten Systems vorzunehmen. Die Analyse befasst sich mit der Funktionalität, der technischen Umsetzung, der Effizienz der Entwicklung sowie den erreichten Geschäftszielen.

1. **Funktionalität und Technische Umsetzung:**
   Die im Backend implementierten Funktionen und API-Endpunkte wurden eingehend getestet, insbesondere hinsichtlich ihrer Interaktion mit dem Frontend.
   Die Datenverarbeitung und -speicherung, insbesondere die Handhabung der Serviceanfragen, wurden entsprechend den Anforderungen verifiziert.
   Die implementierten Skripte zur Erstellung der Datenbank inkl. User, Rollen, Schema, Indexes wurden erfolgreich getestet.
   Das Ausführten von Automatisierten Backups wurde mehre Tage erfolgreich getestet.
2. **Systemleistung und Sicherheit:**
   Das Backend wurde auf seine Leistungsfähigkeit hin ausgewertet, mit einem besonderen Fokus auf die Antwortzeiten durch Verwendung von Indexes und die Zuverlässigkeit unter Last.
3. **Effizienz der Entwicklung:**
   Die Entscheidungen für die verwendeten Technologien und Frameworks, wie .NET Core und Mongo Driver, sowie die Projektstruktur, wurden hinsichtlich ihrer Auswirkungen auf die Entwicklungsgeschwindigkeit und -qualität bewertet.
   Die Verwendung von Database-First Migration trug zur Effizienz der Datenbankentwicklung bei.
4. **Erreichte Geschäftsziele:**
   Die Backend-Funktionalität unterstützt nun effektiv das Auftragsmanagement und die Kundeninteraktionen, was zu einer verbesserten Betriebseffizienz führt.
   Die Datenerfassung im Backend ermöglicht es, zukünftige geschäftliche Entscheidungen auf Grundlage von soliden Datenanalysen zu treffen.
5. **Kunden- und Nutzerfeedback:**
   Rückmeldungen von Endnutzern und Stakeholdern wurden gesammelt und analysiert, um die Benutzerfreundlichkeit und Zufriedenheit mit den Backend-Funktionen zu bewerten.
   Dieses Feedback wurde als Grundlage für zukünftige Verbesserungen und Funktionsupdates herangezogen.

11.1 Fazit

Die Migration des Backend-Systems von einer SQL-basierten Architektur zu einem NoSQL-System, konkret zu MongoDB, stellte einen bedeutenden Wendepunkt in der Entwicklung des Projekts dar. Diese Umstellung war von entscheidender Bedeutung, um die Skalierbarkeits- und Flexibilitätsanforderungen des wachsenden Unternehmens Jetstream-Service zu erfüllen. Die Herausforderung bestand darin, die bestehenden relationalen Datenstrukturen in ein Schema zu überführen, das die dokumentenorientierte Natur von MongoDB vollständig ausnutzt.

Die Migration erforderte eine umfassende Analyse und Neugestaltung des Datenmodells, um die Leistungsvorteile, die MongoDB bietet, wie schnelle Datenabfragen und effiziente Datenverarbeitung, optimal zu nutzen. Ein kritischer Schritt war die Überführung der relationalen Daten in ein Format, das den Anforderungen von NoSQL-Datenbanken entspricht, einschliesslich der Anpassung von Beziehungen und der Optimierung von Abfragen für nicht-relationale Datenstrukturen.

Dieser Prozess zeigte die Bedeutung einer sorgfältigen Planung und Durchführung der Datenmigration. Durch die erfolgreiche Umstellung konnte eine erhebliche Verbesserung in Bezug auf Systemperformance und -skalierbarkeit erreicht werden, was wiederum die Handhabung und Verwaltung von Ski-Serviceaufträgen erheblich verbesserte. Die Migration zu MongoDB ermöglichte es, komplexe Abfragen effizienter zu gestalten und die Flexibilität bei der Datenverwaltung zu erhöhen, um schnell auf geschäftliche Anforderungen reagieren zu können.

Die Erfahrung aus der Migration des Backend-Systems von SQL zu NoSQL unterstrich die Notwendigkeit einer klaren Trennung von Datenverwaltungslogik und Anwendungslogik. Dies führte zu einem verbesserten Systemdesign, das nicht nur die aktuelle Leistungsfähigkeit und Effizienz steigert, sondern auch eine solide Grundlage für zukünftige Erweiterungen und Anpassungen bietet.

Durch die erfolgreiche Durchführung dieser Migration demonstrierte das Projekt technische Kompetenz und ein tiefes Verständnis für die Anwendung moderner Datenbanktechnologien. Es markiert einen entscheidenden Schritt in der digitalen Transformation von Jetstream-Service und trägt wesentlich zur Optimierung der Geschäftsprozesse und zur Steigerung der Kundenzufriedenheit bei.

12 Anhänge

12.1 Voraussetzungen für die Funktionalität des Backends (siehe auch readme)

Im "Script/MongoDBScripts" Ordner ist die UsersAndRoles.js -Datei in der sich der Befehl, um einen neuen Benutzer zu erstellen befindet. Es wird jeweils ein Superadmin und ein User: Johny erstellt. Der Superadmin wird erstellt damit man mit den Superadmin wenn die Authentifizierung in MongoDB aktiviert ist sich anmelden kann. Das wird alles mit create.ps1 Skript durchgeführt. Das PowerShell Skript erstellt den Superadmin, User, customRole, Schema, Collection, Index und seedet die Daten, um so die vollständige Migration durchzuführen. Der User: Johny wird benötigt um mit dem Backend Connection String dem User nur nötige Rechte wie "find", "insert", "update", "remove" zuzuweisen, das nicht auf andere Datenbanken zugegriffen werden kann, oder Collections gelöscht werden können. Der Connection String ist im appsettings.json und im Files Ordner: UserConnectionString.txt hinterlegt.

12.2 Glossar

BEGRIFF	BESCHREIBUNG / ERKLÄRUNG
API (APPLICATION PROGRAMMING INTERFACE)	Ein Satz von Routinen, Protokollen und Tools für die Erstellung von Software und Anwendungen, die eine Verbindung zwischen unterschiedlichen Softwareanwendungen ermöglichen.
AUTHENTIFIZIERUNG	Ein Prozess, der die Identität eines Benutzers verifiziert, typischerweise durch Benutzername und Passwort.
BACKEND	Der Teil der Anwendung, der auf dem Server läuft und für die Verarbeitung von Geschäftslogik, Datenbankmanagement und Server-Interaktionen zuständig ist.
CODEFIRST	Ein Ansatz bei Entity Framework, bei dem die Datenbankstruktur auf Grundlage von Code-Modellen generiert wird.
DTO (DATA TRANSFER OBJECT)	Ein Objekt, das dazu dient, Daten zwischen Subsystemen der Anwendung zu tragen, typischerweise zwischen dem Client und dem Server.
ENTITY FRAMEWORK	Ein ORM (Object-Relational Mapper), das die Datenbankzugriffsschicht einer Anwendung vereinfacht.
GANTT-DIAGRAMM	Ein Werkzeug für das Projektmanagement, das den Zeitplan von Projektaktivitäten visualisiert.
GITHUB	Eine Web-basierte Plattform für Versionskontrolle und kollaborative Softwareentwicklung.

IPERKA-MODELL	Ein Akronym für Informieren, Planen, Entscheiden, Realisieren, Kontrollieren, Auswerten - ein Prozessmodell im Projektmanagement.
JSON (JAVASCRIPT OBJECT NOTATION)	Ein leichtgewichtiges Daten-Austauschformat, das für Menschen leicht zu lesen und für Maschinen leicht zu parsen ist.
MIDDLEWARE	Software, die als Brücke zwischen einem Betriebssystem oder einer Datenbank und Anwendungen, insbesondere auf einem Netzwerk, dient.
PSP (PROJEKTSTRUKTURPLAN)	Ein dokumentiertes Werkzeug, das die erforderlichen Arbeitsschritte eines Projekts in kleinere, leichter zu handhabende Teile aufgliedert.
RESTFUL API	Ein API, das die Prinzipien der Representational State Transfer (REST)-Architektur verwendet, um Interaktionen mit Webdiensten zu ermöglichen.
SWAGGERUI	Ein Werkzeug, das die Erstellung von Dokumentationen für RESTful APIs unterstützt und eine visuelle Plattform für das Testen von API-Endpunkten bietet.
VISUAL STUDIO / VISUAL STUDIO CODE	Integrierte Entwicklungsumgebungen (IDEs) von Microsoft zur Softwareentwicklung.

Tabelle 4: Glossar

12.3 NuGet Packages Screenshot

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <InvariantGlobalization>true</InvariantGlobalization>
  </PropertyGroup>

    <ItemGroup>
        <PackageReference Include="AutoMapper.Extensions.Microsoft.DependencyInjection" Version="12.0.1" />
        <PackageReference Include="Microsoft.AspNetCore.Authentication.JwtBearer" Version="8.0.1" />
        <PackageReference Include="Microsoft.AspNetCore.OpenApi" Version="8.0.1" />
        <PackageReference Include="Microsoft.IdentityModel.Tokens" Version="7.2.0" />
        <PackageReference Include="MongoDB.Driver" Version="2.23.1" />
        <PackageReference Include="Newtonsoft.Json" Version="13.0.3" />
        <PackageReference Include="Serilog.AspNetCore" Version="8.0.0" />
        <PackageReference Include="Serilog.Settings.Configuration" Version="8.0.0" />
        <PackageReference Include="Serilog.Sinks.File" Version="5.0.0" />
        <PackageReference Include="Swashbuckle.AspNetCore" Version="6.5.0" />
        <PackageReference Include="System.IdentityModel.Tokens.Jwt" Version="7.2.0" />
    </ItemGroup>

</Project>
```

Abbildung 5: NuGet Packages Screenshot

12.4 Swagger Screenshots

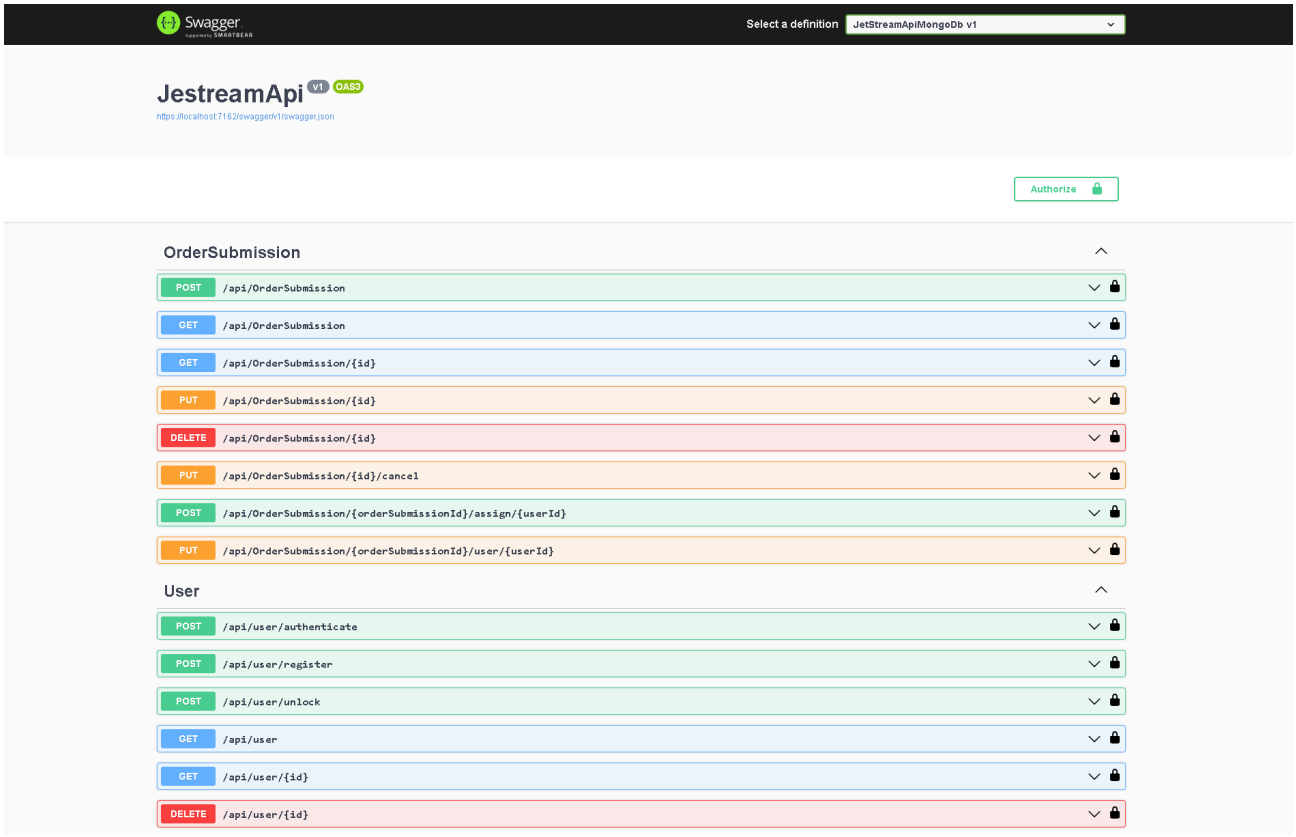


Abbildung 6: SwaggerUI - API Endpoints

Abbildung 7: SwaggerUI - Schemas

12.5 Postman Collection Test Screenshot

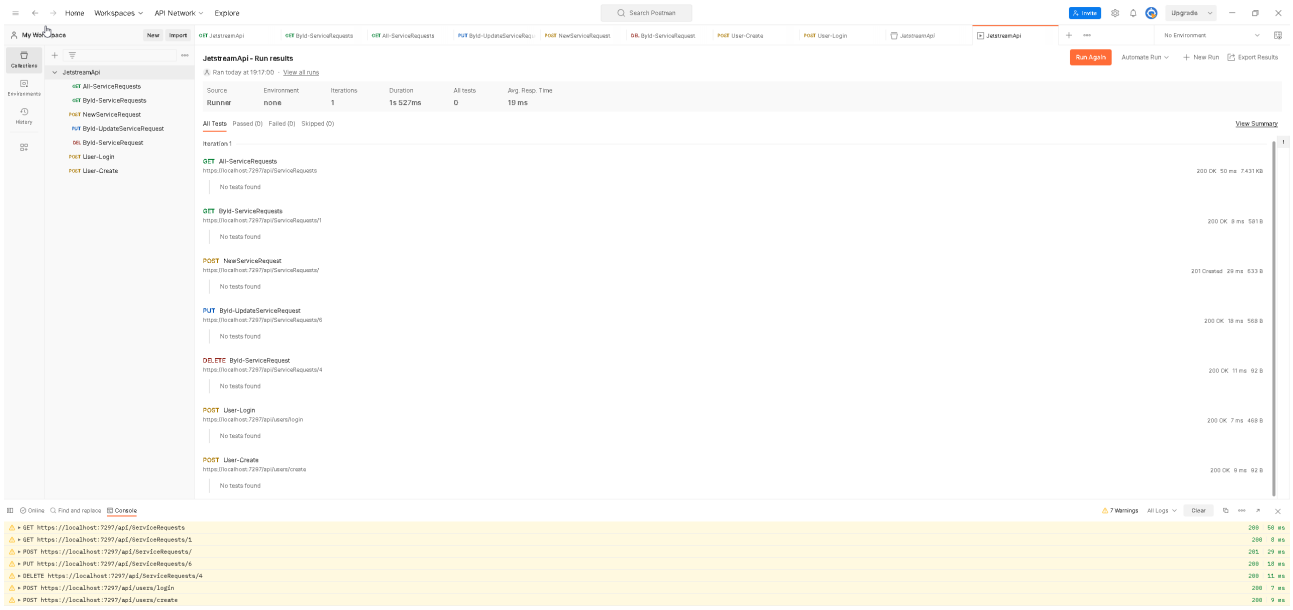


Abbildung 8: Postman Collection Test

12.6 xUnit Testergebnis Screenshot

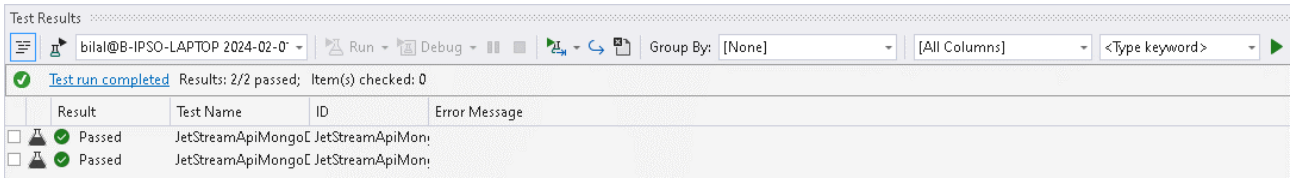


Abbildung 9: xUnit Testergebnis

12.7 Verweise

12.7.1 Quellenverzeichnis

Postman, I. (21. November 2023). *Postman API Platform*. Von Postman: https://www.postman.com abgerufen

Software, S. (21. November 2023). *Swagger UI*. Von Swagger.io: https://swagger.io/tools/swagger-ui/ abgerufen

12.7.2 Tabellenverzeichnis

12.7.3 Abbildungsverzeichnis