NOSQL ANBINDUNG

Für die Jetstream-Service AG

Dokumentation zur Projektarbeit
ICT Modul 165

Abgabedatum: Basel, 09.01.2024

Prüfungsbewerber:

Mahir Gönen

09.02.2024

Inhalt

1 Versionsverzeichnis

Version	Autor	Datum	Änderung
1.0	Mahir Gönen	06.02.2024	Erstellung des Dokuments
1.1	Mahir Gönen	06.02.2024	Ergänzung des Dokuments
1.2	Mahir Gönen	07.02.2024	Rechtschreibung und Grammatik Kontrolle
1.3	Mahir Gönen	09.02.2024	Fertigstellung des Dokuments

2 Executive Summary

Das Projekt zur Migration des Backend-Systems von Jetstream-Service auf MongoDB war eine strategische Initiative, die darauf abzielte, die Datenverwaltungsinfrastruktur des Unternehmens zu modernisieren und dessen operative Effizienz zu steigern. Angesichts der wachsenden Anforderungen und der Notwendigkeit, die Skalierbarkeit und Flexibilität des Systems zu verbessern, wurde die Entscheidung getroffen, von einer relationalen SQL-Datenbank zu einer NoSQL-Lösung überzugehen.

Die Migration umfasste mehrere Schlüsselaspekte: die sorgfältige Planung und Analyse der bestehenden Datenstrukturen, die Entwicklung eines robusten Migrationsprozesses, die Aktualisierung des Backends und die Implementierung von Sicherheits- und Zugangskontrollmechanismen. Durch den Einsatz von spezialisierten Skripten wurde die Datenmigration effizient durchgeführt, während gleichzeitig die Integrität und Sicherheit der Daten gewährleistet wurde.

Ein wesentliches Element des Projekts war die Entwicklung und Anwendung von Skripten in JavaScript und PowerShell, um die Datenbankinitialisierung, Zugangskontrollen und das Backup-Management zu automatisieren. Diese Skripte spielten eine entscheidende Rolle bei der Vereinfachung der Systemwartung und der Gewährleistung hoher Sicherheitsstandards.

Die Realisierung des Projekts führte zu signifikanten Verbesserungen in der Leistungsfähigkeit und Effizienz des Backend-Systems. Die Migration zu MongoDB bot nicht nur verbesserte Performance und Skalierbarkeit, sondern ermöglichte auch eine kosteneffizientere Verwaltung der Datenbankressourcen. Darüber hinaus trug die erfolgreiche Umsetzung des Projekts zur Steigerung der Benutzerzufriedenheit und zur Optimierung der täglichen Betriebsabläufe bei.

Zusammenfassend hat die Migration des Backend-Systems zu MongoDB Jetstream-Service ermöglicht, seine technologische Basis zu stärken und sich für zukünftiges Wachstum zu positionieren. Die Erfahrungen und Erkenntnisse aus diesem Projekt dienen als wertvolle Grundlage für zukünftige Initiativen und Innovationen.

3 Ausgangslage

Die Firma Jetstream-Service, ein KMU, das sich auf Skiservicearbeiten spezialisiert hat, stand vor einer Herausforderung: Die bisherige IT-Infrastruktur, insbesondere die relationale Datenbank, konnte mit den steigenden Anforderungen des Unternehmens nicht mehr mithalten. In den letzten Jahren hatte das Unternehmen erheblich in eine digitale Auftragsanmeldung und -verwaltung investiert, um den Serviceprozess zu optimieren und die Kundenzufriedenheit zu steigern. Diese Systeme basierten auf einer datenbankgestützten Web-Anmeldung und Auftragsverwaltung, die jedoch an ihre Grenzen stiess.

Die Hauptziele der Migration waren klar definiert: Die Ablösung der SQL-Datenbank durch ein flexibleres und skalierbareres NoSQL-Datenbanksystem, um die Datenverteilung und Skalierung effektiver zu gestalten. MongoDB wurde als die geeignete Technologie ausgewählt, um diesen Anforderungen gerecht zu werden. Die Entscheidung fiel aufgrund von MongoDBs Fähigkeit, grosse Mengen unstrukturierter Daten effizient zu verwalten, was für die Verarbeitung der vielfältigen Serviceaufträge und Kundendaten von Jetstream-Service essentiell war.

Die Ausgangssituation war geprägt von der Notwendigkeit, nicht nur die technologische Basis zu erneuern, sondern auch die Betriebskosten zu senken. Die Lizenzkosten für die bisherige relationale Datenbank stellten eine erhebliche finanzielle Belastung dar. Zudem war die Skalierung der alten Systeme kompliziert und kostspielig, was die Expansionspläne des Unternehmens, neue Standorte zu eröffnen und das Dienstleistungsangebot zu diversifizieren, erheblich behinderte.

Um diesen Herausforderungen zu begegnen, wurde ein umfassender Migrationsplan entwickelt, der nicht nur die technische Umsetzung, sondern auch strategische Überlegungen berücksichtigte. Die Migration zu MongoDB ist somit ein zentraler Baustein in der digitalen Transformation von Jetstream-Service. Sie ermöglicht es dem Unternehmen, agiler auf Marktveränderungen zu reagieren, die Betriebseffizienz zu steigern und eine solide Basis für zukünftiges Wachstum zu schaffen.

4 Anforderungen

Die erfolgreiche Migration der Jetstream-Service-Datenbankinfrastruktur von einer relationalen SQL-Datenbank zu einem NoSQL-System wie MongoDB stellte eine zentrale Säule in der digitalen Transformationsstrategie des Unternehmens dar. Um diesen Übergang nahtlos und effizient zu gestalten, wurden spezifische Anforderungen definiert, die sowohl technische als auch betriebliche Aspekte des Projekts abdecken.

Zunächst war es entscheidend, eine vollständige Datenmigration von der bestehenden relationalen Datenbank zu MongoDB zu gewährleisten. Dies erforderte eine sorgfältige Planung und Durchführung, um Datenintegrität und -konsistenz ohne Verlust oder Korruption sicherzustellen. Darüber hinaus musste ein Benutzerkonzept implementiert werden, das mindestens zwei Benutzeranmeldungen mit verschiedenen Berechtigungsstufen unterstützt, um eine differenzierte Zugriffskontrolle auf die Datenbank zu ermöglichen.

Ein weiterer wichtiger Punkt war die Bereitstellung eines eingeschränkten Datenbankbenutzerzugangs für die Web-API-Applikation, der lediglich DML (Data Manipulation Language) Operationen erlaubt. Dies dient der Sicherheit und Integrität der Datenbank, indem es unautorisierte oder potenziell schädliche Zugriffe verhindert.

Um die Leistungsfähigkeit und Effizienz der neuen Datenbankstruktur zu maximieren, war die Implementierung eines Schemas für Datenkonsistenz sowie die Einrichtung von Datenbankindizes für die schnelle Ausführung von Suchabfragen erforderlich. Zudem mussten Backup- und Restore-Möglichkeiten entwickelt werden, die durch Skript-Dateien unterstützt werden, um Datenverluste in Notfallsituationen zu verhindern und eine hohe Verfügbarkeit der Daten zu gewährleisten.

Die Migration des Web-API-Projekts (CRUD) auf das NoSQL-Datenbanksystem und die vollständige Dokumentation des Datenmodells, einschliesslich grafischer Darstellungen, waren ebenso essentielle Bestandteile des Projekts. Die Einrichtung eines einfachen Testprojekts in Postman unterstützte die Validierung und das Debugging der API-Endpunkte. Abschliessend wurde die Bedeutung einer effektiven Projektverwaltung mittels eines Git-Repositories und einer detaillierten Dokumentation des gesamten Projektverlaufs nach der IPERKA-Methode hervorgehoben.

Die zusätzlichen, optionalen Anforderungen, wie die Implementierung eines automatisierten Backup-Konzepts und die Umsetzung komplexer Schema-Validierungen, boten die Möglichkeit, die Robustheit und Flexibilität der Datenbankinfrastruktur weiter zu erhöhen. Diese optionalen Erweiterungen sollten dazu beitragen, die Datenbanklösung nicht nur den aktuellen, sondern auch zukünftigen Anforderungen anzupassen und somit eine langfristige Skalierbarkeit und Leistungsfähigkeit zu gewährleisten.

Diese umfassenden Anforderungen bildeten die Grundlage für ein erfolgreiches Migrationsprojekt, das darauf abzielte, Jetstream-Service mit einer modernen, effizienten und skalierbaren Datenbanklösung auszustatten, die den Weg für weiteres Wachstum und Innovation ebnet.

5 Zeitplanung

Das Projekt wird in mehreren Phasen durchgeführt und es wurden folgender Zeitplan festgelegt (totale Stunden vom Entwickler):

Projektphase	Geplante Zeit
Informieren	3h
Planung, Entwurf, Entscheidung	5h
Realisierung	25h
Abnahme	2h
Dokumentation	10h
Gesamt	45h

Tabelle 1: Grobe Planung

Dieses Projekt findet innerhalb von knapp zwei Wochen statt.

6 Informieren

In dieser Phase des IPERKA-Modells liegt der Schwerpunkt auf der gründlichen Informationsbeschaffung. Ziel ist es, ein detailliertes Verständnis für die spezifischen Bedürfnisse, Anforderungen und Herausforderungen bei der Migration für Jetstream-Service zu erlangen. Diese Phase beinhaltet die Analyse für das Erkennen der Bedürfnisse der Nutzer und Mitarbeiter, das Verständnis für die technischen Anforderungen sowie die Berücksichtigung aller weiteren Faktoren, die für den erfolgreichen Abschluss der Applikation entscheidend sind.

6.1 Einleitung

In der Einleitung für das Projekt "NoSQL-Migration für Jetstream-Service" wird der Kontext und die Relevanz des Vorhabens detailliert dargelegt. Angesichts der wachsenden Anforderungen an Skalierbarkeit, Flexibilität und Effizienz in der Datenverwaltung hat Jetstream-Service die Entscheidung getroffen, seine bestehende relationale Datenbankinfrastruktur zu überdenken. Die Migration zu MongoDB, einem führenden NoSQL-Datenbanksystem, markiert einen strategischen Schritt, um diesen Herausforderungen zu begegnen und die digitale Transformation des Unternehmens voranzutreiben. Dieses Projekt zielt darauf ab, die Effizienz im Backend und in der Datenverarbeitung zu steigern, indem eine leistungsfähigere und kosteneffizientere Lösung implementiert wird. Die Anpassung des Backends auf die neue Datenbankarchitektur und die Gewährleistung einer nahtlosen Integration mit bestehenden Frontend-Lösungen stehen im Mittelpunkt der Bemühungen. Die Einleitung betont zudem die Bedeutung der

Modernisierung und Digitalisierung für Jetstream-Service und stellt den Bezug zur IPERKA-Methodik her, um die systematische und zielgerichtete Herangehensweise des Projekts zu verdeutlichen.

6.2 Vorgabe/Anforderungen

Die Hauptziele der Datenbankmigration von SQL zu MongoDB für Jetstream-Service umfassen mehrere Schlüsselaspekte, die für die Realisierung eines effizienten, skalierbaren und sicheren Systems notwendig sind:

- **Migration und Datenintegrität:** Die vollständige und sichere Übertragung aller bestehenden Daten von der relationalen SQL-Datenbank zu MongoDB, unter Wahrung der Datenintegrität und -konsistenz.
- **Benutzerzugriff und -verwaltung:** Implementierung eines Benutzerkonzepts mit mindestens zwei Anmeldungsstufen mit verschiedenen Berechtigungen, um differenzierte Zugriffskontrollen zu ermöglichen und die Sicherheit zu erhöhen.
- **Effizienz und Performance:** Entwicklung und Implementierung von Datenbankindizes und Schemata für Datenkonsistenz, um schnelle Ausführungen von Suchabfragen und Transaktionen zu gewährleisten.
- **Backup und Wiederherstellung:** Einrichtung von Backup- und Restore-Prozessen mittels Skripten, um die Datenbestände gegen Verlust oder Beschädigung zu schützen.
- **Dokumentation und Management:** Konsequente Anwendung des IPERKA-Modells für das gesamte Projektmanagement, einschliesslich einer detaillierten Dokumentation der Datenmigration, der neuen Datenbankstruktur und der Anpassungen an der Web-API. Zudem soll ein Versionsverzeichnis die Übersicht und Nachvollziehbarkeit aller Projektphasen gewährleisten.

Diese Anforderungen bilden das Fundament für die Umstellung auf ein NoSQL-Datenbanksystem, welches die zukünftige Expansion und Flexibilität von Jetstream-Service unterstützt und eine moderne, effiziente Lösung für das Auftragsmanagement bietet.

6.3 Zusätzliche Anforderung

Im Laufe der Informationsphase wurden neben den grundlegenden Anforderungen weitere spezifische Bedürfnisse und Wünsche identifiziert, die das Projekt in einer optimierten Form prägen könnten.

6.3.1 Automatisiertes Backup-Konzept durchgeführt und implementiert

Ein optionales Element der Sicherheitsstrategie des Projekts war die Implementierung eines automatisierten Backup-Konzepts für die MongoDB-Datenbank. Dieses Konzept wurde mithilfe eines PowerShell-Skripts realisiert, welches so konfiguriert ist, dass es automatisch tägliche Backups durchführt. Diese Backups sind entscheidend, um die Integrität und Verfügbarkeit der Daten zu gewährleisten, insbesondere in einem dynamischen Umfeld, das durch regelmässige Updates und Skalierungsmassnahmen gekennzeichnet ist.

Das Skript wurde so gestaltet, dass es im Windows "Aufgabenplaner" auf dem Server, auf dem das Backend läuft, eingerichtet werden kann. Es sorgt für die automatische Erstellung von Backups der MongoDB-Datenbank, die eine schnelle Wiederherstellung im Falle eines

Datenverlusts oder einer Beschädigung ermöglicht. Das Backup-Skript und das entsprechende Aufgabenplaner-Skript sind im GitHub-Repository des Projekts verfügbar.

Bei der Erstinitialisierung des Backup-Systems müssen die Pfade des Backupskripts und des Aufgabenplaner-Skripts an die spezifische Serverumgebung angepasst werden. Diese Anpassung umfasst die Konfiguration der Speicherorte, an denen die Backups gespeichert werden sollen, sowie die Planung der Backup-Zeitpläne, um den Betrieb nicht zu stören. Die sorgfältige Konfiguration und regelmässige Überprüfung des Backup-Prozesses sind essenziell, um die Sicherheit und Zuverlässigkeit der Datenbankinfrastruktur zu gewährleisten.

Durch dieses automatisierte Backup-Konzept wird eine robuste Sicherheitsmassnahme implementiert, die den Schutz kritischer Unternehmensdaten in der MongoDB-Datenbank sicherstellt. Die Dokumentation und Konfigurationsdateien bieten eine klare Anleitung für die Einrichtung und Wartung des Backup-Systems, was eine zuverlässige Datenwiederherstellung und Betriebskontinuität im Falle unvorhergesehener Ereignisse gewährleistet.

6.3.2 Komplexe Schema Validierung umgesetzt

Im Rahmen der Migration des Backend-Systems auf MongoDB wurde ein automatisiertes Backup-Konzept entwickelt und implementiert, um die Datenintegrität und Verfügbarkeit zu gewährleisten. Die Automatisierung des Backup-Prozesses erfolgte durch die Erstellung eines PowerShell-Skripts, das in der Windows-Aufgabenplanung eingesetzt wird. Dieses Skript ermöglicht es, tägliche Backups automatisch durchzuführen, und trägt somit zur Minimierung von Datenverlust-Risiken bei.

Das PowerShell-Skript, zusammen mit der Konfigurationsanleitung, ist im GitHub-Repository des Projekts verfügbar. Bei der Erstinitialisierung des Backup-Systems müssen spezifische Pfade für das Backup- und das Aufgabenplaner-Skript angepasst werden, um eine korrekte Funktion in der Serverumgebung zu gewährleisten. Diese Anpassungen sind essenziell, um sicherzustellen, dass die Backups ordnungsgemäss in der vorgesehenen Speicherlocation abgelegt werden und im Falle eines Systemausfalls oder Datenverlusts schnell wiederhergestellt werden können.

Zur weiteren Optimierung des Backup-Konzepts wurden Tests durchgeführt, um die Zuverlässigkeit und Effizienz der automatisierten Backups zu verifizieren. Diese Tests umfassten sowohl die Überprüfung der Backup-Dateien auf Vollständigkeit und Integrität als auch die Simulation von Wiederherstellungsprozessen, um die minimale Downtime im Ernstfall zu garantieren.

Das implementierte Backup-Konzept ist ein wesentlicher Bestandteil der neuen Backend-Architektur und unterstreicht das Engagement des Projekts für Sicherheit und Beständigkeit der Daten. Durch die erfolgreiche Umsetzung dieses Konzepts wird eine solide Grundlage für die zukünftige Skalierbarkeit und Erweiterung des Systems gelegt, wobei die Daten des Unternehmens geschützt und jederzeit verfügbar gehalten werden.

7 Planen

Die Planungsphase stellt einen entscheidenden Abschnitt dar. Nachdem in der Informationsphase alle erforderlichen Daten gesammelt und ein klares Bild der Anforderungen und Rahmenbedingungen geschaffen wurde, gilt es nun, diese Erkenntnisse in einen strukturierten und realisierbaren Plan zu übertragen.

7.1 Entwurf eines Anwendungsdesign

Im Rahmen der Entwicklung für die Firma Jetstreamski-Service liegt ein besonderer Fokus auf die Anbindung des Backends an die Datenbank. Dazu wurde ein Anwendungsdesign erstellt, um die Backendverbindung einfacher darzustellen.

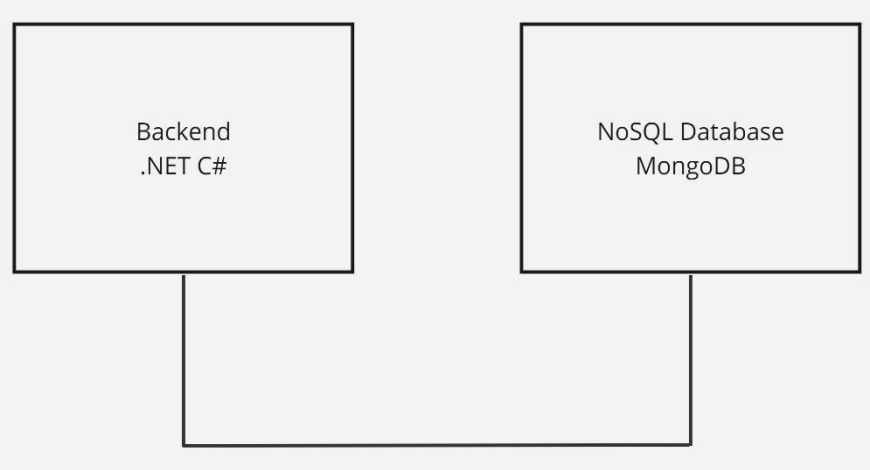

Abbildung 1 Anwendungsdesign

7.2 Datenmodell

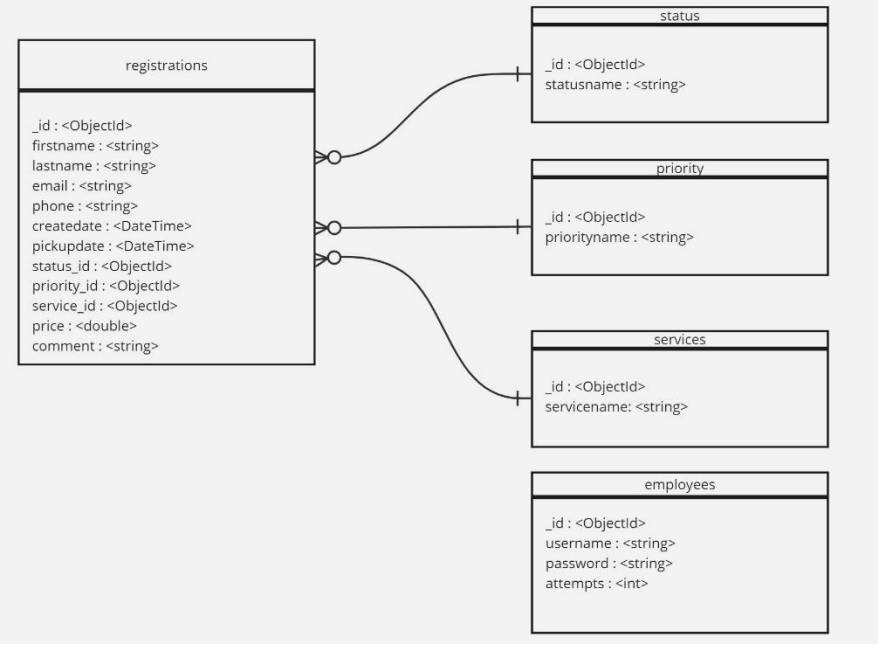

Abbildung 4 Datenmodell

Abbildung 5 Postman Testprojekt ResultateAbbildung 6 Datenmodell

7.3 Organisation

Um die Architektur und das Zusammenspiel der Komponenten des Backend-Systems optimal zu gestalten, wurde eine sorgfältige Auswahl an Technologien und Methoden getroffen.

7.3.1 Technologieauswahl

Die Wahl der Technologien für das neue Backend-System wurde von mehreren Schlüsselfaktoren geleitet, darunter Skalierbarkeit, Performance, Sicherheit und die Fähigkeit zur nahtlosen Integration in bestehende Systeme. Nach eingehender Bewertung verschiedener Optionen fiel die Entscheidung auf MongoDB als NoSQL-Datenbanklösung. MongoDB bietet eine hohe Leistungsfähigkeit bei der Verarbeitung grosser Datenmengen und erlaubt eine flexible Datenmodellierung, was für die dynamischen Anforderungen von Jetstream-Service essentiell ist.

Für die Entwicklung der Web-APIs wurde C# ausgewählt, eine leistungsstarke und vielseitige Programmiersprache, die sich durch ihre Sicherheitsfeatures, die umfangreiche Standardbibliothek und die Unterstützung für moderne Softwareentwicklungspraktiken auszeichnet. C# in Verbindung mit dem .NET Framework ermöglicht eine effiziente Entwicklung von robusten und skalierbaren Webdiensten.

Die Entscheidung für diese Technologien wurde auch durch die Notwendigkeit einer hohen Kompatibilität mit verschiedenen Plattformen und Systemen beeinflusst, sowie durch die Verfügbarkeit von Entwicklungsressourcen und Community-Unterstützung. Die Technologieauswahl spiegelt das Bestreben wider, eine zukunftssichere Lösung zu implementieren, die eine kontinuierliche Verbesserung und Anpassung an sich ändernde Geschäftsbedingungen ermöglicht.

Die Integration dieser Technologien bildet das Fundament für ein leistungsfähiges, skalierbares und sicher betreibbares Backend-System. Durch die strategische Auswahl von MongoDB und C# wurde eine solide Basis geschaffen, die die Effizienz der Skiservice-Arbeiten verbessert und die Verwaltungsaufgaben vereinfacht, während gleichzeitig die Grundlage für zukünftige Erweiterungen und Verbesserungen gelegt wird.

7.3.1.1 Anwendungen und NuGet-Pakete

Alle verwendeten Anwendungen und NuGet-Pakete werden hier aufgelistet und beschrieben. Um Komplikationen zu verhindern sind die Versionierungen und vollständigen Beschreibung auf hier dokumentiert.

Die WebAPI läuft unter dem Framework WPF.NET8.0

Folgende NuGet-Pakete werden benötigt für das Backend:

- Automapper.Extensions.Microsoft.DependencyInjection
- Microsoft.AspNetCore.Authentication.JwtBearer
- Microsoft.AspNetCore.OpenApi
- Microsoft.IdentityModel.Tokens
- MongoDB.Bson
- MongoDB.Driver

- MongoDB.Driver.Core
- Serilog.AspNetCore
- Swashbuckle.AspNetCore
- System.IdentityModel.Tokens.Jwt

Da, dass Testprojekt ein eigenständiges xUnit-Projekt ist kommen folgende NuGet-Pakete für das Testprojekt dazu:

- Microsoft.NET.Test.Sdk
- Moq
- Xunit
- xunit.runner.visualstudio
- coverlet.collector

7.3.2 GitHub

Mithilfe von GitHub wurde das komplette Projekt durch Versionsverläufe dokumentiert. Durch ein Git-Repository kann der Entwickler in der Realisierungsphase die komplette Programmierarbeit einfacher und unabhängig des Standortes bewältigen. Durch die Commits, die der Entwickler bei allen Änderungen macht, bringt dies eine grosse Transparenz in das Projekt zwischen dem Projektteam und Kunden bei.

7.4 Planung des Schemas

Die Konzeption der Datenbankstruktur begann mit der Entwicklung eines umfassenden Schemas, das nicht nur die aktuellen Datenanforderungen abbildet, sondern auch genügend Flexibilität für zukünftige Erweiterungen bietet. MongoDB's schemalose Natur ermöglichte es eine dynamische Struktur zu erstellen, in der Dokumente unterschiedliche Felder und Datenstrukturen enthalten können, was eine effizientere Datenabfrage und -speicherung ermöglicht.

Ein wesentlicher Aspekt der Datenbankplanung war die Implementierung von Indizes. Indizes sind entscheidend für die Leistungsfähigkeit der Datenbank, da sie die Effizienz von Abfrageoperationen erheblich steigern können. Durch die gezielte Indexierung der am häufigsten abgefragten Datenfelder konnte das Team sicherstellen, dass Abfragen schnell und effizient bearbeitet werden, was zu einer verbesserten Nutzererfahrung führt.

Um die Integrität und Konsistenz der Daten zu gewährleisten, wurden zudem Validierungsregeln festgelegt. Diese Regeln stellen sicher, dass nur Daten, die den definierten Anforderungen entsprechen, in die Datenbank eingefügt oder aktualisiert werden können. Die Validierung auf Datenbankebene ist ein kritischer Schritt, um Datenfehler und Inkonsistenzen zu minimieren, die die Anwendungsleistung beeinträchtigen könnten.

7.5 Planung des Indexes

Die Indexplanung begann mit einer Analyse der Abfrageanforderungen der Anwendung. Indem das Team die Abfragemuster und die am häufigsten genutzten Suchkriterien identifizierte, konnte es gezielt Indizes für die entsprechenden Felder erstellen. Diese strategische Vorgehensweise gewährleistete, dass die Indizes die Abfragen unterstützen, die für die Geschäftslogik und die Benutzerinteraktionen am wichtigsten sind.

Bei der Implementierung der Indizes achtete das Team auch auf die Balance zwischen der Beschleunigung der Leseoperationen und den Auswirkungen auf Schreiboperationen. Jeder Index, der hinzugefügt wird, kann das Einfügen, Aktualisieren und Löschen von Dokumenten in der Datenbank verlangsamen, da der Index ebenfalls aktualisiert werden muss. Daher war es wichtig, eine sorgfältige Auswahl zu treffen, um eine optimale Leistung zu erzielen, ohne die Systemeffizienz negativ zu beeinflussen.

7.6 Planung der Seed-Daten

Die direkte Einführung der Seed-Daten ins Backend war entscheidend, um eine schnelle und effiziente Initialisierung des Systems zu gewährleisten. Diese Vorgehensweise ermöglichte es, die Datenbank mit allen notwendigen Datensätzen zu bevorraten, die für die Funktionalität des Systems erforderlich sind, einschliesslich Benutzerkonten, Serviceaufträge und Produktkataloge.

- **Sicherstellung der Erstinitialisierung:** Durch das direkte Einführen der Seed-Daten in das Backend wurde sichergestellt, dass das System von Beginn an mit einer realistischen Datenbasis arbeitet. Dies ist besonders wichtig für die Entwicklungsumgebung, da es Entwicklern und Testern ermöglicht, Features und Funktionen unter realitätsnahen Bedingungen zu evaluieren.

- **Effizienz und Zeitersparnis:** Die Verwendung von Seed-Daten beschleunigt den Entwicklungsprozess erheblich, da manuelle Eingaben oder das Warten auf die Akkumulation von Live-Daten entfallen. Entwickler können sofort mit einem voll funktionsfähigen System arbeiten, was die Zeit bis zur Markteinführung verkürzt.
- **Konsistenz und Wiederholbarkeit:** Ein weiterer Vorteil der Planung und Einführung von Seed-Daten ist die Gewährleistung von Konsistenz über verschiedene Entwicklungs-, Test- und Produktionsumgebungen hinweg. Dies erleichtert die Fehlersuche und die Qualitätskontrolle, da das Verhalten des Systems unter kontrollierten Bedingungen getestet wird.
- **Grundlage für Tests und Demonstrationen:** Seed-Daten bieten eine solide Basis für die Durchführung von Funktionstests, Lasttests und Demonstrationen für Stakeholder. Sie tragen dazu bei, das Vertrauen in die Stabilität und Leistungsfähigkeit des Systems zu stärken, indem sie zeigen, wie das System unter Einsatzbedingungen funktioniert.

Die Planung und Einführung von Seed-Daten war ein kritischer Schritt im Entwicklungsprozess des Backend-Systems von Jetstream-Service. Durch diesen Ansatz konnte das Team effizient eine robuste und realistische Testumgebung schaffen, die für die erfolgreiche Implementierung und den Betrieb des Systems unerlässlich ist.

7.7 Planung der Migration SQL zu MongoDB

Die Migration von der bestehenden SQL-Datenbank zu MongoDB markiert einen entscheidenden Wendepunkt in der technologischen Evolution von Jetstream-Service. Angesichts der Anforderung, das alte Datenbanksystem vollständig zu ersetzen, wurde ein sorgfältiger und systematischer Ansatz für die Migration gewählt, der direkt in die C#-Anwendung bei der Datenbankinitialisierung integriert wurde.

- **Vorbereitung und Analyse:** Der erste Schritt der Migrationsplanung umfasste eine detaillierte Analyse der bestehenden SQL-Datenstrukturen, um ein umfassendes Verständnis der Daten, ihrer Beziehungen und der in der alten Datenbank implementierten Geschäftslogik zu erlangen. Diese Analyse diente als Grundlage für die Entwicklung eines Migrationsplans, der die Übertragung aller relevanten Daten in das neue NoSQL-Format ohne Datenverlust oder -verfälschung sicherstellt.
- **Test und Validierung:** Nach der Entwicklung des Migrationswerkzeugs und der Durchführung der ersten Migrationsläufe wurde eine umfassende Testphase initiiert. Ziel war es, die Korrektheit der Datenübertragung zu überprüfen und sicherzustellen, dass das neue System wie erwartet funktioniert. Dies umfasste sowohl automatisierte Tests zur Überprüfung der Datenintegrität als auch manuelle Inspektionen und Funktionsprüfungen.

Die strategische Planung und sorgfältige Durchführung der Migration von SQL zu MongoDB ermöglichte es Jetstream-Service, seine Datenverwaltung auf eine moderne, leistungsfähige Plattform zu heben. Die direkte Integration der Migrationslogik in die C#-Anwendung bei der Datenbankinitialisierung gewährleistete eine effiziente und nahtlose Überführung in das neue System, legte den Grundstein für zukünftiges Wachstum und verbesserte die Systemleistung signifikant.

8 Entscheiden

Eine der Schlüsselentscheidungen in der Entwicklung des neuen Backend-Systems für Jetstream-Service betraf die Struktur des Datenmodells, insbesondere die Wahl zwischen Referenzen (Referenced Documents) und eingebetteten Dokumenten (Embedded Documents) in MongoDB. Diese Entscheidung hatte tiefgreifende Auswirkungen auf die Leistung, Skalierbarkeit und Flexibilität der Datenbank und erforderte eine sorgfältige Abwägung der Vor- und Nachteile beider Ansätze.

8.1 Entscheidung für Referenzen

Der ausschliessliche Einsatz von Referenzen statt eingebetteter Dokumente basiert auf mehreren Überlegungen:

- **Leistungssteigerung bei Abfragen:** Die Verwendung von Referenzen ermöglicht eine flexiblere Datenorganisation und optimiert die Abfrageleistung. Durch das Trennen von Daten in eigenständige Dokumente, die über Referenzen miteinander verknüpft sind, kann das System spezifische Daten effizienter abrufen, ohne umfangreiche, eingebettete Dokumentstrukturen durchsuchen zu müssen.
- **Optimale Nutzung von NoSQL-Vorteilen:** NoSQL-Datenbanksysteme, wie MongoDB, sind für ihre Fähigkeit bekannt, mit grossen Mengen von verteilten Daten effektiv umzugehen. Durch die Entscheidung für Referenzen nutzt das Projektteam die inhärente Skalierbarkeit und Flexibilität von MongoDB, um die Datenstruktur an zukünftiges Wachstum und sich ändernde Geschäftsanforderungen anzupassen.
- **Vereinfachung der Datenpflege:** Referenzen erleichtern die Wartung und Aktualisierung von Daten. Änderungen an einem Dokument, das in verschiedenen Teilen der Anwendung referenziert wird, müssen nur an einer Stelle vorgenommen werden. Dies erhöht die Datenkonsistenz und reduziert den Aufwand für die Datenpflege.
- **Skalierbarkeit und Erweiterbarkeit:** Ein Referenz-basiertes Modell unterstützt die Skalierbarkeit des Systems, indem es die Erweiterung der Datenbankstruktur ohne Beeinträchtigung der bestehenden Datenbeziehungen ermöglicht. Neue Datentypen oder Beziehungen können hinzugefügt werden, indem einfach neue Referenzen eingeführt werden, was die langfristige Flexibilität des Systems sichert.

Diese Entscheidung unterstreicht das Engagement, eine moderne, effiziente und zukunftssichere Lösung zu implementieren. Durch die konsequente Nutzung von Referenzen im Datenmodell wird sichergestellt, dass das Backend-System von Jetstream-Service die volle Leistungsfähigkeit von MongoDB ausschöpft, um eine herausragende Performance und Benutzererfahrung zu bieten.

9 Realisieren

Im Zuge der Realisierungsphase des Projekts bei Jetstream-Service stand die Umsetzung der geplanten Konzepte und Entscheidungen im Mittelpunkt. Diese Phase markierte den Übergang von der Theorie zur Praxis, wobei das Hauptaugenmerk auf der Aktualisierung und Anpassung des Backends lag, um die neuen Anforderungen und die Integration in MongoDB zu erfüllen.

9.1 Backend

Das vorhandene Backend-System, das ursprünglich für die Nutzung mit einer SQL-Datenbank konzipiert war, wurde grundlegend überarbeitet, um die Migration zu MongoDB nicht nur zu unterstützen, sondern auch um die Performance, Flexibilität und Skalierbarkeit des Systems zu optimieren.

- **Aktualisierung auf MongoDB-Treiber:** Ein zentraler Aspekt der Backend-Aktualisierung war die Implementierung des MongoDB-Treibers in die bestehende C#-Anwendung. Dieser Schritt war entscheidend, um eine nahtlose Integration und Interaktion mit der neuen NoSQL-Datenbank zu ermöglichen. Der MongoDB-Treiber bietet umfangreiche Funktionen für die Datenmanipulation, Abfrageoptimierung und Transaktionsverwaltung, die speziell auf die Bedürfnisse von dokumentenorientierten Datenbanken zugeschnitten sind.
- **Anpassung der Geschäftslogik:** Die Geschäftslogik des Backends musste entsprechend den neuen Datenstrukturen und Möglichkeiten, die MongoDB bietet, angepasst werden. Dies umfasste die Neugestaltung von Datenabfragen, die Optimierung von Datenzugriffsmustern und die Implementierung von Dokumentenbeziehungen mittels Referenzen. Ziel war es, die Effizienz zu maximieren und gleichzeitig die Konsistenz und Integrität der Daten zu gewährleisten.
- **Test und Qualitätssicherung:** Nach der Implementierung der Änderungen wurde das aktualisierte Backend-System umfassenden Tests unterzogen, um sicherzustellen, dass alle Komponenten wie erwartet funktionieren und die Migration zu MongoDB keine negativen Auswirkungen auf die Anwendungsleistung hat. Automatisierte Tests, Einheitentests und Integrationstests spielten eine wichtige Rolle in diesem Prozess, um die Zuverlässigkeit und Stabilität des Systems zu validieren.

Die Realisierungsphase war von entscheidender Bedeutung für den Erfolg des Projekts. Durch die sorgfältige Aktualisierung und Anpassung des Backends an die Anforderungen und die Nutzung des MongoDB-Treibers konnte Jetstream-Service ein leistungsstarkes, flexibles und zukunftssicheres System schaffen, das bereit ist, den Anforderungen eines wachsenden Unternehmens gerecht zu werden.

9.2 Skripts

Zugangskontrollskripte: Um eine effiziente und sichere Zugangskontrolle zu gewährleisten, wurden mehrere JavaScript-Skripte entwickelt. Diese Skripte sind für die Erstellung von Benutzerkonten im Backend zuständig, wobei zwei Haupttypen von Konten unterschieden werden: Admin und User. Jeder Kontotyp verfügt über spezifische Berechtigungen, die es ermöglichen, die Zugriffsrechte innerhalb des Systems präzise zu steuern. Zusätzlich wurde ein spezielles Zugangsskript für die API implementiert, um eine sichere Kommunikation und Interaktion mit externen Anwendungen zu ermöglichen.

Schema- und Indexskripte: Zur Optimierung der Datenbankstruktur und -performance wurden Skripte erstellt, die Schemas und Indizes in MongoDB definieren. Diese JavaScript-Skripte spielen eine entscheidende Rolle bei der Initialisierung und Konfiguration der Datenbank, indem sie sicherstellen, dass die Daten effizient organisiert und abgefragt werden können.

Datenbankinitialisierungsskript: Ein zentrales PowerShell-Skript wurde entwickelt, um die Datenbank bei der Erstinstallation oder nach bedeutenden Änderungen automatisch zu initialisieren. Dieses Skript ruft die zuvor erwähnten JavaScript-Skripte für Zugangskontrollen, Schemas und Indizes auf und führt sie aus, um eine nahtlose und korrekte Einrichtung der Datenbank zu gewährleisten.

Backup- und Restore-Skripte: Für die Datensicherheit wurden spezielle PowerShell-Skripte (MongoBackup.ps1 und MongoRestore.ps1) erstellt. Diese ermöglichen es, regelmässige Datenbanksicherungen durchzuführen und bei Bedarf einen früheren Zustand der Datenbank wiederherzustellen.

Automatisierung der Backups: Um die regelmässigen Backups zu automatisieren, wurde das WindowsTaskplanner.ps1-Skript entwickelt. Dieses Skript nutzt die Funktionalitäten des Windows Aufgabenplaners, um das Backup.ps1-Skript entsprechend einem definierten Zeitplan auszuführen. Dadurch wird sichergestellt, dass die Daten regelmässig und ohne manuellen Eingriff gesichert werden.

Dokumentation: Ein wesentlicher Bestandteil des Skript-Pakets ist das README.txt, das detaillierte Anweisungen zur Verwendung, Konfiguration und zum Verständnis der Skripte enthält. Die Lektüre dieses Dokuments ist essentiell, um die Skripte korrekt einzusetzen und die Konfigurationen an die spezifischen Bedürfnisse und Pfade des Systems anzupassen.

Die sorgfältige Entwicklung und Implementierung dieser Skripte bildet einen wichtigen Pfeiler der Backend-Architektur von Jetstream-Service. Sie tragen nicht nur zur Sicherheit und Effizienz bei, sondern automatisieren auch kritische Prozesse, die für den reibungslosen Betrieb und die Wartung des Systems unerlässlich sind.

10 Kontrollieren

Die Kontrollphase ist entscheidend, um sicherzustellen, dass das entwickelte System den Anforderungen und Erwartungen entspricht. Ein wichtiger Bestandteil dieser Phase ist die Implementierung eines Testprojekts.

10.1 Testplan

Dieser Testplan ist für das Testprojekt entwickelt geworden.

1. Testziele

Sicherstellen, dass alle Controller-Funktionalitäten korrekt arbeiten.

Überprüfung der Integration zwischen Controllern, Services und Datenmodellen.

2. Testumfang

Tests für EmployeeController, PriorityController, RegistrationController und StatusController.

3. Teststrategie und -ansatz

Unit-Tests für individuelle Funktionen in jedem Controller.

Mocking von Services und Datenmodellen mit Moq.

Verwendung von FluentAssertions für Testergebnisse.

4. Testumgebung

Die Tests werden in einer standardisierten Entwicklungsumgebung unter .NET durchgeführt.

5. Testdaten

Erstellung von Testdaten für Mitarbeiter, Prioritäten, Registrierungen und Status.

6. Testfälle und -szenarien

Jede Testdatei (EmployeeControllerTests.cs, etc.) enthält spezifische Testfälle für die entsprechenden Controller-Funktionen.

7. Risikomanagement

Identifikation potenzieller Risiken bei der Testdurchführung und Entwicklung von Strategien zur Risikominderung.

8. Kriterien für den Abschluss der Tests

Alle Testfälle müssen erfolgreich durchlaufen werden.

Keine kritischen Fehler dürfen in den Systemkomponenten verbleiben.

10.2 Testprojekt implementieren

Für die Ski-Service.NET wurde ein umfassendes Testprojekt implementiert, das auf den festgelegten Testplan abgestimmt ist. Die Implementierung des Testprojekts umfasste folgende Schlüsselelemente:

Testziele: Hauptziel war die Sicherstellung der korrekten Funktionsweise aller Knöpfe und Dialogverhalten. Die meisten Tests sind in visueller Ansicht und deshalb wurde es von den Entwicklern durchgeführt, da ein xUnit-Test solche Sachen nicht direkt testen kann.

Testumfang: Alle Controllerklassen und Endpoints wurden im Testprojekt unter verschiedenen Szenarien getestet und auch im Postman Collection.

Teststrategie und -ansatz: Es wurde ein xUnit-Testprojekt erstellt, um HTTP-Requests zu testen.

Testumgebung: Die Tests wurden in einer standardisierten .NET-Entwicklungsumgebung durchgeführt, um konsistente und zuverlässige Ergebnisse zu gewährleisten.

Testdaten: Erstellung von Testdaten für Mitarbeiter und Aufträgen

Testfälle und -szenarien: Jede Testdatei enthält spezifische Testfälle für die entsprechenden Controller-Funktionen

Risikomanagement: Mögliche Risiken im Testprozess wurden identifiziert, und es wurden Strategien entwickelt, um diese Risiken zu minimieren. Es wurden kritische Fehler dadurch ausgeschlossen.

Kriterien für den Abschluss der Tests: Das Testprojekt wurde als erfolgreich abgeschlossen betrachtet, sobald alle Testfälle erfolgreich durchlaufen wurden und keine kritischen Fehler in den Systemkomponenten verblieben. Erfolgreiche Testfälle siehe hier.

Das Testprojekt hat sich als wichtig erwiesen, um die Zuverlässigkeit und Stabilität der Anwendung zu gewährleisten. Es trug entscheidend dazu bei, das Vertrauen in die Funktionalität der einzelnen Komponenten zu stärken und die Gesamtqualität des Systems zu sichern.

10.3 Testprojekt – Postman Collection

Erstellung der Postman Collection: Die Postman Collection wurde sorgfältig zusammengestellt, um eine vollständige Abdeckung aller API-Endpoints zu erreichen. Jeder Endpoint wurde mit spezifischen Anfragen konfiguriert, die die Bandbreite der möglichen Operationen – von einfachen Datenabfragen bis hin zu komplexen Transaktionen – repräsentieren. Diese methodische Herangehensweise ermöglichte es, das System unter verschiedenen Bedingungen zu testen und sicherzustellen, dass die API wie erwartet funktioniert.

Implementierung des Testprojekts: Innerhalb der Postman Collection wurde ein Testprojekt implementiert, das eine Reihe automatisierter Tests durchführt. Diese Tests sind darauf ausgelegt, die Funktionalität, Performance und Sicherheit der API zu überprüfen. Sie umfassen Validierungen der Rückgabewerte, Prüfungen der Fehlerbehandlung und Leistungstests unter Lastbedingungen.

Testdurchführung und -automatisierung: Die Tests wurden sowohl manuell als auch automatisiert durchgeführt, um eine kontinuierliche Überwachung der Systemleistung und -stabilität zu ermöglichen. Die Automatisierung der Testläufe über Postman ermöglichte regelmässige Überprüfungen ohne zusätzlichen manuellen Aufwand. Dies trug massgeblich zur Effizienz des Testprozesses bei und stellte eine schnelle Feedbackschleife für das Entwicklerteam sicher.

Die Entwicklung der Postman Collection und des Testprojekts war ein wesentlicher Bestandteil des Qualitätssicherungsprozesses. Sie ermöglichte eine gründliche Überprüfung des aktualisierten Backend-Systems und trug dazu bei, das Vertrauen in die Zuverlässigkeit und Leistungsfähigkeit der neuen Lösung zu stärken. Durch die systematische Testung und Validierung konnte das Team die erfolgreiche Migration zu MongoDB bestätigen und eine solide Grundlage für den produktiven Einsatz des Systems legen.

11 Auswerten

Die Auswertungsphase des Projekts zur Migration des Backend-Systems von Jetstream-Service zu MongoDB dient als abschliessende Bewertung der durchgeführten Arbeiten und der erreichten Ziele. Diese Phase ist entscheidend, um den Erfolg des Projekts zu messen, den Mehrwert der umgesetzten Lösungen zu quantifizieren und mögliche Bereiche für zukünftige Verbesserungen zu identifizieren.

Leistungsbeurteilung: Eine umfassende Leistungsbeurteilung des neuen Systems wurde durchgeführt, um die Verbesserungen in der Geschwindigkeit, Effizienz und Skalierbarkeit zu bewerten. Die Nutzung von MongoDB hat zu signifikanten Performance-Gewinnen geführt, insbesondere in Bezug auf die Abfragegeschwindigkeit und die Flexibilität bei der Datenmanipulation. Durch die Implementierung von Indizes und die Optimierung der Datenabfragen konnte eine höhere Anwendungsperformance erzielt werden, die direkt die Benutzererfahrung verbessert.

Funktionalitätsprüfung: Die Auswertungsphase umfasste auch eine gründliche Prüfung der Funktionalität des Systems, um sicherzustellen, dass alle Anforderungen erfüllt und die gewünschten Features korrekt implementiert wurden. Dies beinhaltete Tests der Zugangskontrollen, der Datenmigrationstools sowie der Backup- und Wiederherstellungsskripte. Besonderes Augenmerk wurde auf die Sicherheitsaspekte gelegt, um die robuste Implementierung von Sicherheitsmassnahmen und die Integrität der Daten zu bestätigen.

Benutzerfeedback: Das Feedback der Endbenutzer und der Stakeholder wurde eingeholt und analysiert, um Einblicke in die Benutzerzufriedenheit und die praktische Anwendbarkeit der neuen Lösung zu gewinnen. Dieses Feedback ist von unschätzbarem Wert, um die Auswirkungen der Migration auf die täglichen Arbeitsabläufe zu verstehen und Hinweise auf eventuelle Anpassungs- oder Erweiterungsbedarfe zu erhalten.

Kosteneffizienz: Eine Bewertung der Kosteneffizienz des Projekts wurde durchgeführt, um die wirtschaftlichen Vorteile der Migration zu MongoDB zu bestimmen. Die Analyse umfasste eine Betrachtung der Einsparungen bei Lizenzkosten, der Reduktion von Wartungs- und Betriebskosten sowie der langfristigen Skalierbarkeitsvorteile. Die Ergebnisse bestätigen, dass die Entscheidung für MongoDB eine kosteneffiziente Lösung darstellt, die das Potenzial hat, die Betriebskosten weiter zu senken, während gleichzeitig die Leistungsfähigkeit und Flexibilität des Systems erhöht wird.

Identifikation von Verbesserungspotenzial: Abschliessend wurden Bereiche identifiziert, in denen zukünftige Verbesserungen möglich und wünschenswert sind. Diese Erkenntnisse bieten eine Grundlage für die kontinuierliche Optimierung des Systems und die Planung zukünftiger Entwicklungszyklen.

Die Auswertungsphase hat gezeigt, dass die Migration zu MongoDB und die damit verbundenen Aktualisierungen des Backend-Systems einen signifikanten Mehrwert für Jetstream-Service geschaffen haben. Die erzielten Ergebnisse unterstreichen die Bedeutung einer sorgfältigen Planung, Implementierung und Evaluierung in technologiegetriebenen Projekten.

12 Fazit

Die erfolgreiche Migration des Backend-Systems von Jetstream-Service von einer relationalen SQL-Datenbank zu MongoDB markiert einen signifikanten Meilenstein in der digitalen Transformation des Unternehmens. Dieses Projekt hat nicht nur die technologische Infrastruktur des Unternehmens modernisiert, sondern auch dessen Fähigkeit verbessert, auf die dynamischen Anforderungen des Marktes zu reagieren und ein nachhaltiges Wachstum zu fördern.

12.1 Persönliches Fazit – Mahir Gönen

Die Migration des Backend-Systems zu MongoDB stellte für mich persönlich ein besonders herausforderndes, jedoch äusserst erfüllendes Projekt dar. Die erfolgreiche Durchführung bestätigte nicht nur die sorgfältige Planung und Strategieentwicklung im Vorfeld, sondern bot mir auch eine unvergleichliche Lerngelegenheit. Durch die Auseinandersetzung mit den technischen Aspekten von NoSQL-Datenbanken und die Entwicklung von Automatisierungsskripten konnte ich mein Fachwissen signifikant erweitern.

Der Abschluss dieses Projekts hinterlässt ein Gefühl der Zufriedenheit und Motivation. Diese Erfahrung hat nicht nur meinen schulischen Horizont erweitert, sondern auch meine Begeisterung für zukünftige technologische Herausforderungen geweckt. Die erfolgreiche Migration ist ein Beweis für das Engagement und die harte Arbeit dient als starke Motivation für kommende Projekte.

Insgesamt betrachte ich das Projekt als einen wesentlichen Meilenstein in meiner schulischen Entwicklung, der wertvolle Einblicke in die Komplexität moderner Datenbanksysteme lieferte und meine Fähigkeiten in der Projektplanung und -durchführung vertiefte.

Anhänge

I. Glossar

BEGRIFF	BESCHREIBUNG / ERKLÄRUNG
.NET	Ein Software-Framework von Microsoft, das zur Entwicklung und Ausführung von Anwendungen dient, insbesondere für Windows-Umgebungen.
xUnit	Eine Familie von Unit-Testing-Frameworks, die für verschiedene Programmiersprachen verfügbar sind und Entwicklern beim Testen von Code helfen.
HTTP-Requests	Anfragen, die über das Hypertext Transfer Protocol gesendet werden, um Daten zwischen Clients und Servern zu übertragen.
appsettings.json	Eine Konfigurationsdatei in .NET-Anwendungen, die Einstellungen wie Datenbankverbindungen und Anwendungsparameter speichert.
relationalen SQL	Relationale SQL-Datenbanken organisieren Daten in Tabellen, wobei jede Tabelle aus Reihen und Spalten besteht. Sie nutzen die Structured Query Language (SQL) für die Erstellung, Manipulation und Abfrage der in Beziehungen stehenden Daten.
NoSQL-System	NoSQL-Systeme sind Datenbanken, die eine nicht-relationale Datenstruktur verwenden, um eine Vielzahl von Datenmodellen wie Schlüssel-Wert-Paare, Dokumente, Graphen und Spaltenfamilien zu unterstützen. Sie sind entworfen für grosse Datenmengen, hohe Skalierbarkeit und flexible Datenmodelle.
MongoDB	MongoDB ist eine dokumentenorientierte NoSQL-Datenbank, die JSON-ähnliche Dokumente mit dynamischen Schemata verwendet, um die Datenintegration in bestimmten Anwendungstypen zu vereinfachen und zu beschleunigen. Sie ist bekannt für ihre Flexibilität, Skalierbarkeit und Leistungsfähigkeit bei der Verwaltung grosser Datenmengen.
Web-API	Eine Web-API ist eine Programmierschnittstelle, die den Austausch von Daten zwischen verschiedenen Softwareanwendungen über das Internet ermöglicht. Sie definiert, wie Anwendungen miteinander kommunizieren können, oft unter Verwendung von HTTP-Protokollen für Webdienste.
CRUD	CRUD steht für Create, Read, Update, Delete - die vier grundlegenden Operationen, die in Datenbankanwendungen zur Verwaltung von Daten durchgeführt werden. Diese Operationen ermöglichen es, Daten hinzuzufügen, abzufragen, zu ändern und zu entfernen.
Backup	Ein Backup bezeichnet die Sicherung von Datenkopien auf Speichermedien oder in der Cloud, um Datenverluste bei Systemausfällen, Datenkorruption oder anderen unvorhergesehenen Ereignissen zu vermeiden. Es ermöglicht die Wiederherstellung der gesicherten Daten.

Tabelle 2 Glossar

II. Weitere Dokumente

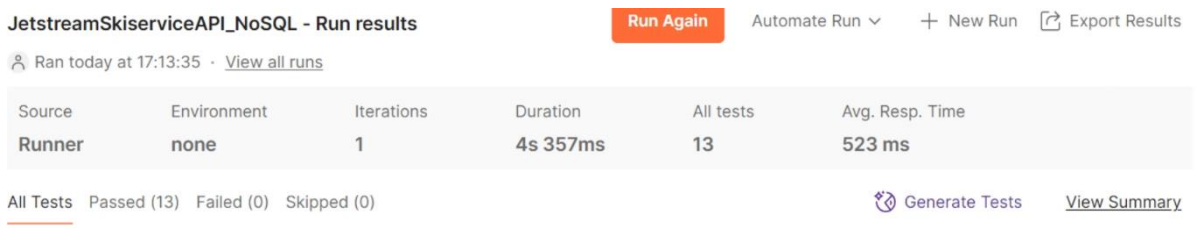

Abbildung 7 Postman Testprojekt Resultate

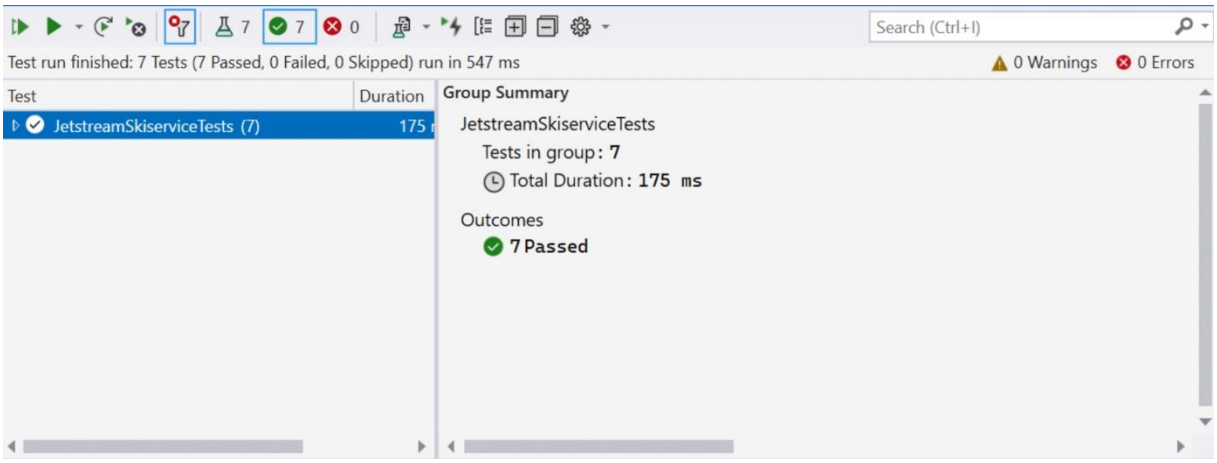

Abbildung 4 xUnit Testprojekt Resultate

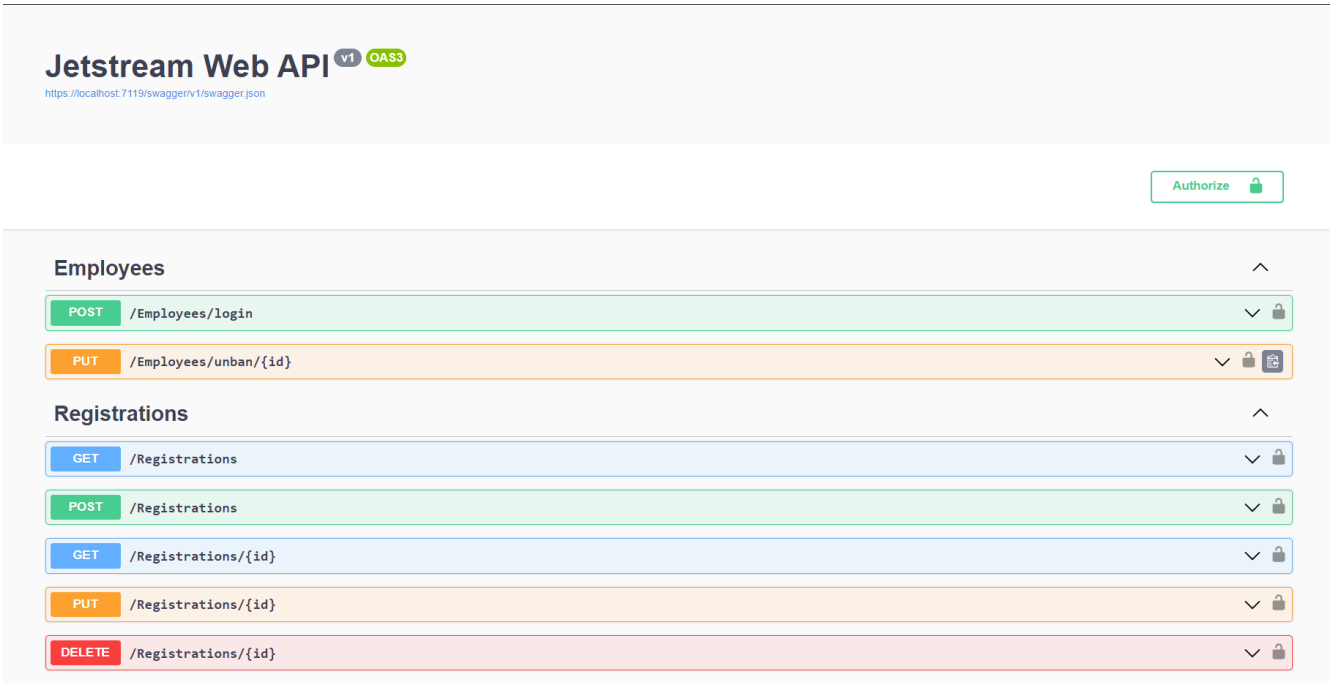

Abbildung 5 SwaggerUI

III. Versionsverzechnis der Anwendung/Pakete

Anwendung/Paket	**Version**
.NET WPF	.NET8.0
AutoMapper.Extensions.Microsoft.DependencyInjection	12.0.1
Microsoft.AspNetCore.Authentication.JwtBearer	8.0.1
Microsoft.IdentityModel.Tokens	7.2.0
MongoDB.Bson	2.23.1
MongoDB.Driver	2.23.1
MongoDB.Driver.Core	2.23.1
Serilog.AspNetCore	8.0.1
Swashbuckle.AspNetCore	6.5.0
System.IdentityModel.Tokens.Jwt	7.2.0
Microsoft.NET.Test.Sdk	17.6.0
Moq	4.20.70
xunit	2.4.2
xunit.runner.visualstudio	2.4.5

13 Quellenverzeichnis

Dokumente aus dem OneNote

https://www.mongodb.com/docs/

14 Tabellenverzeichnis

15 Abbildungsverzeichnis