ipso! Bildung seit 1876

PROJEKT MODUL 165

Azin Ildas

Titelbild 1: MongoDb.

2. FEBRUAR 2024

IPSO BILDUNG AG
IBZ BASEL

Abbildungsverzeichnis

Quellenverzeichnis

Versionsverzeichnis

Version	Datum	Änderung
1	25.01.2024	Projektdokumentation Erstelldatum
1.1	25.01.2023	Erste Einträge in die Dokumentation
1.3	02.02.2024	Letzter Eintrag in die Dokumentation

ipso! Bildung seit 1876

Inhalt

1 Einleitung

1.1 Was war der Auftrag?

Die Firma Jetstream-Service hat in den letzten Jahren in eine digitale Auftragsanmeldung und Auftragsverwaltung investiert. Aufgrund guter Auftragslage und geplanter Neueröffnungen an verschiedenen Standorten soll die bestehende relationale Datenbank durch ein NoSQL Datenbanksystem ersetzt werden.

1.2 Rollen

1.2.1 Der Auftraggeber

Der Auftraggeber für das Projekt ist die Firma Jetstream Service.

1.2.2 Der Auftragnehmer

Der Auftragnehmer welches die Anforderungen von dem Auftraggeber entgegennimmt ist Azin Ildas.

2 Index – Strukturen

Indexstrukturen sind spezielle Datenstrukturen in Datenbanken, die das schnelle Suchen und Abrufen von Datensätzen erleichtern, indem sie wie ein Inhaltsverzeichnis funktionieren, das direkten Zugriff auf Informationen ermöglicht, ohne jede Zeile durchsuchen zu müssen.

Code	Beschreibung
use M165Azin	Zur Datenbank M165Azin wechseln.
show collections	Alle Sammlungen innerhalb der Datenbank auflisten.
db.registrations.find()	Gibt alle Dokumente innerhalb der "registrations" Collection zurück.
db.userInfos.find()	Gibt alle Dokumente in der "userInfos" Collection zurück.
db.userSessions.find()	Gibt alle Dokumente in der "userSessions" Collection zurück.
db.registrations.find({ service: "Kleiner Service" })	Um Bestellungen zu finden mit dem Service «Kleiner Service".
db.registrations.find().sort({ finishDate: 1 })	Welche Bestellungen als erstes abgeschlossen werden, also je weiter oben desto weniger Zeit bis zum Abschluss.
db.registrations.find().sort({ finishDate: -1 })	Welche Bestellungen als letztes abgeschlossen werden, also je weiter oben desto mehr Zeit bis zum Abschluss.
db.userSessions.find().sort({ ID: 1 })	Eine Bestellung anhand der ObjectId löschen.
db.registrations.deleteOne({ _id: ObjectId('dieGewünschteObjectId') })	

3 Informieren

3.1 Was soll getan werden?

Das Projekt für Jetstream-Service umfasst die Umsetzung von:

- Entwicklung einer NoSQL-Datenbankstruktur.
- Überführung von Daten aus der relationalen in die NoSQL-Datenbank.
- Anpassung des WebAPI-Projekts an die NoSQL-Datenbank.
- Erstellung eines umfassenden Testplans und Durchführung der Tests.
- Entwicklung des gesamten Backend-Teils entsprechend den Anforderungen.
- Gewährleistung der reibungslosen Zusammenarbeit von Backend und bestehendem Frontend.

4 Planen

4.1 Welche Lösungswege gibt es und wie kann ich vorgehen?

In der Planungsphase werde ich die optimale NoSQL-Datenbank für das Jetstream-Service-Projekt auswählen, indem ich Optionen wie MongoDB oder Neo4j in Betracht ziehe. Ziel ist es, eine passende Lösung zu finden, die den Anforderungen der neuen Standorte entspricht.

4.2 Kleine Arbeitspakete Planen

Arbeitspaket 1.	Sql Datenbank zu MongoDb importieren.
Arbeitspaket 2.	Code mit der MongoDb verbinden.
Arbeitspaket 3.	Weitere Projektanforderungen erfüllen.
Arbeitspaket 4.	Dokumentation schreiben.
Arbeitspaket 5.	Projekt präsentieren.

5 Entscheiden

5.1 Für welches vorgehen entscheiden ich mich?

In der Entscheidungsphase habe ich mich für die Nutzung von MongoDB als NoSQL-Datenbank für das Jetstream-Service-Projekt entschieden. MongoDB bietet eine flexible und skalierbare Datenbankstruktur, die den Anforderungen des Projekts, besonders in Bezug auf Datenverteilung und Skalierung, entspricht.

6 Realisieren

6.1 Bestehende SQL Datenbank zu MongoDb realisieren

6.1.1 Backup der Datenbank

Schritt 1. Ein Backup der bestehenden Datenbank erstellen, indem im Microsoft SQL Server Management Studio auf die gewünschte Datenbank geklickt wird, dann auf „Tasks" und anschliessend auf „Backup". Das Backup ist wichtig, um im Falle eines Problems bei der Migration darauf zurückgreifen zu können.

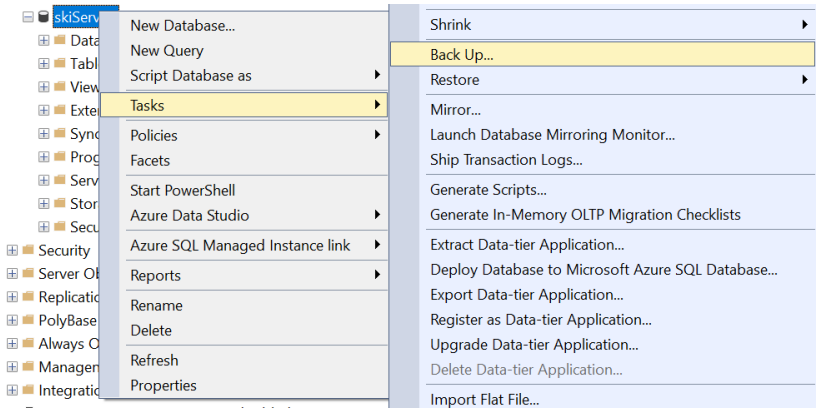

Abbildung 1: Ansicht zum Backupmenü.

Schritt 2. Den Backup an einem sicheren Ort speichern.

6.1.2 Alte Datenbank exportieren

Schritt 1. Im Microsoft SQL Server Management Studio mit der rechten Maustaste auf die gewünschte Datenbank klicken, dann auf „Tasks" und auf „Export Data" klicken.

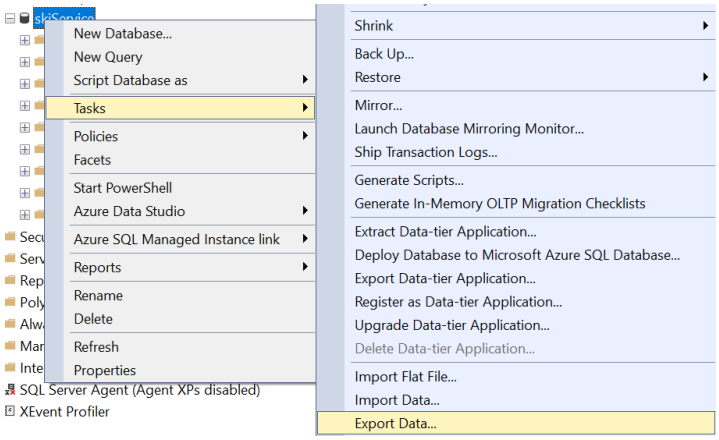

Abbildung 2: Ansicht um Daten zu Exportieren.

Schritt 2. Bei Data Source „Microsoft OLE DB Provider for SQL Server" auswählen:

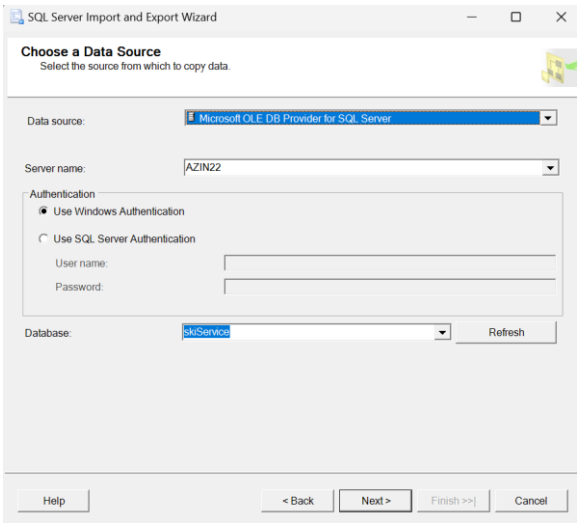

Abbildung 3: Ansicht im Data Source.

Schritt 3. Nachdem wir auf „Next" geklickt haben wählen wir beim zweiten mal im Data Source „Flat File Destination" um den Import als .txt Datei zu speichern:

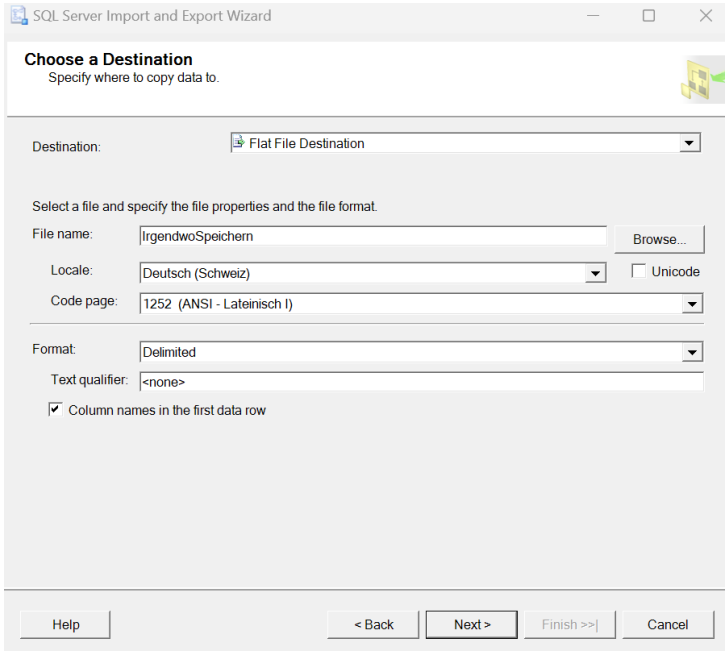

Abbildung 4: Ansicht in der «Flat File Destination» Umgebung.

Schritt 4. Nachdem wir auf „Next" geklickt haben wählen wir bei „Source table or view" die Tabelle die wir Importieren möchten und beenden den Vorgang indem wir auf „Next" klicken:

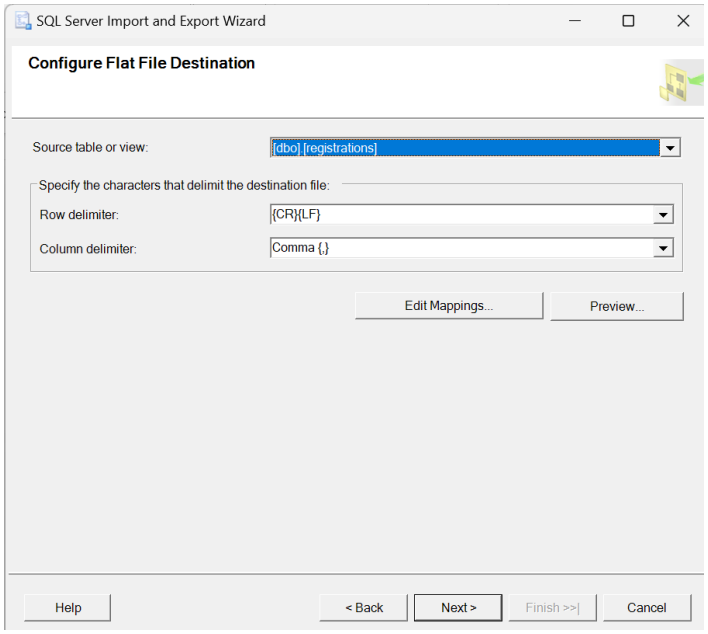

Abbildung 5: Ansicht um den Flat File Destination zu konfigurieren.

Schritt 5. Den ganzen Vorgang für die anderen Tabellen in der Datenbank wiederholen.

6.1.3 Die Imports zur MongoDb hinzufügen

Schritt 1. Im cmd in das Verzeichnis wechseln, in dem die importierten Daten gespeichert sind. Dazu verwenden wir den Befehl cd .., um ein Verzeichnis nach oben zu navigieren, und geben anschliessend cd Pfad/Zum/Verzeichnis/ ein, um in das gewünschte Verzeichnis zu gelangen, in dem sich die importierten Daten befinden.

Schritt 2. Wir geben im cmd den folgenden code: „mongoimport --uri mongodb://localhost:27017/M165Azin --collection meineKollektion --file /pfad/zur/meinedaten.txt --jsonArray" dieser Pfad muss noch an die korrekten infos angepasst werden.

Schritt 3. Diesen Vorgang für alle Imports durchführen.

6.1.4 Die Imports in der MongoDb austesten

Schritt 1. Mit db.kollektionName.find() sehen wir dann den Inhalt in der gewünschten Kollektion. Die Kollektion, die wir dann sehen möchten, sollte anstelle von „kollektionName" stehen.

Schritt 2. Diesen Vorgang für alle Imports testen.

6.2 Meine Web Apis's anhand meiner MongoDb anpassen

Die Anpassungen der bestehenden WebAPIs begannen mit einer Überarbeitung des Codes, um ihn den neuen Anforderungen der MongoDB anzupassen. Nachdem die Codeänderungen implementiert waren, führte ich gründliche Tests durch, um sicherzustellen, dass die APIs wie vorher auf der neuen Umgebung funktionieren.

Abbildung 6: Meine Web Api's.

6.3 Mein Postman anhand von meinen Web Api's anpassen

Ich habe meine Postman - Kollektion so angepasst, dass sie auf meinen Web API's basiert, sodass sie genau die gleichen Aktionen und Abfragen wie die API's selbst durchführen. Diese Anpassung ermöglicht eine effiziente und direkte Interaktion mit den API Funktionen zur Überprüfung und zum Testen.

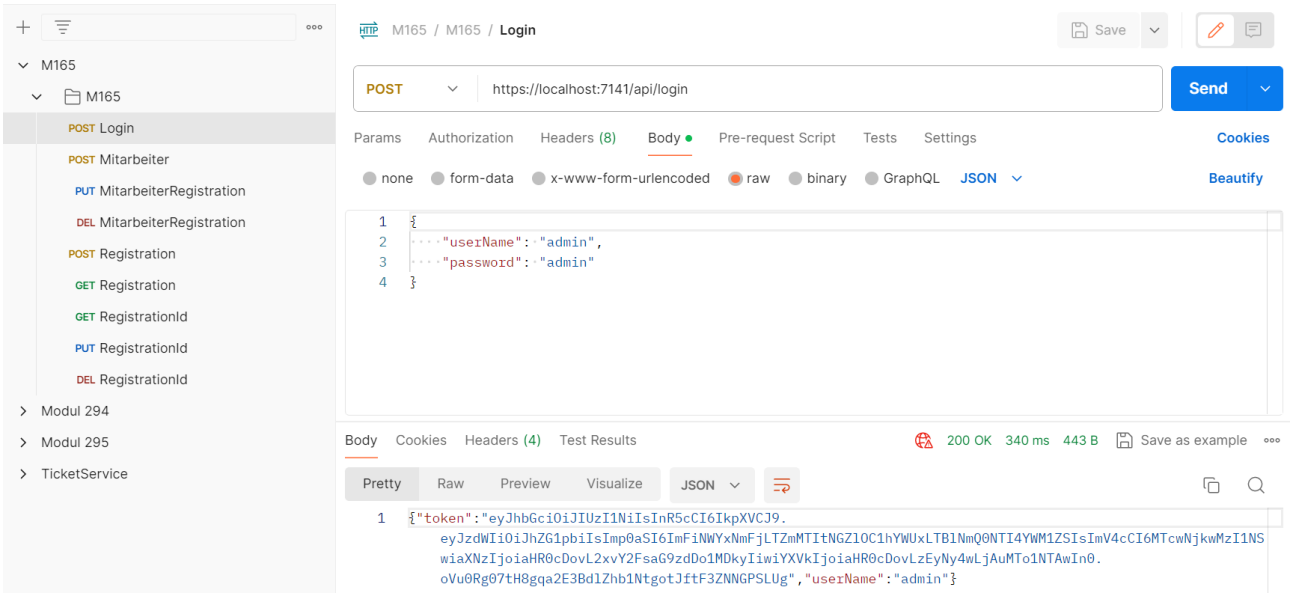

Abbildung 7: Ansicht im Postman.

6.4 Datenmodell

Mein Modell besteht aus drei Sammlungen: registrations, userInfos und userSessions.

Registrations: Hier werden Registrierungen mit Details wie Name, Kontakt, Service Priorität und Service Zeitraum gespeichert. Jeder Eintrag hat eine ObjectId und eine ID.

UserInfos: Diese Sammlung enthält Benutzernamen und Passwörter zur Authentifizierung.

UserSessions: Hier werden Sitzungsschlüssel, meist JWTs, und Verweise auf userInfos zur Session-Verwaltung gespeichert.

Die userSessions referenzieren auf userInfos, um Benutzerkonten und Sitzungen zu verknüpfen.

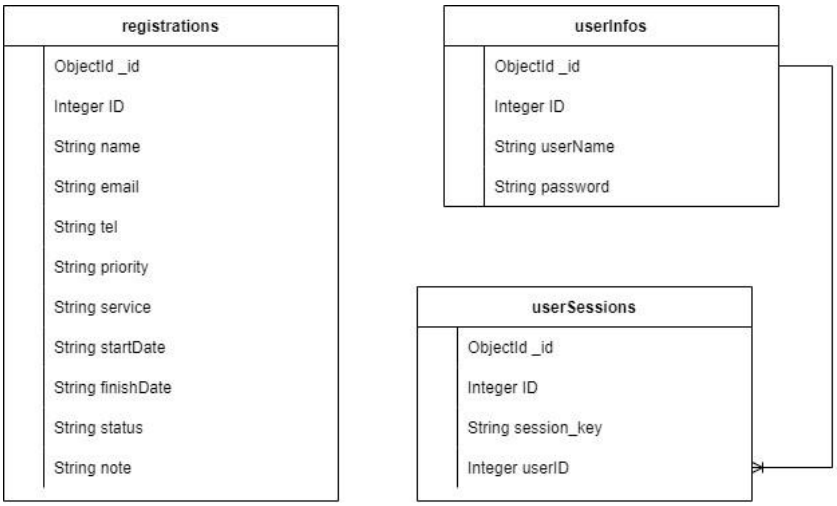

Abbildung 8: Mein Datenmodell.

6.5 Benutzerkonzept

Mein Benutzerkonzept erklärt:

Login: Der Prozess beginnt mit einer Login-Aktion. Hierbei kann sich nur der Admin anmelden.

Geheimer Token: Nach dem Login erhält der Admin einen geheimen Token. Dieser Token ist ein JSON Web Token, der verwendet wird, um sicherzustellen, dass Anfragen an die API von einem autorisierten Benutzer kommen.

Post / Put / Delete Mitarbeiter: Mit dem erhaltenen Token kann der Admin dann Aktionen wie Post (Erstellen), Put (Aktualisieren) und Delete (Löschen) auf der Mitarbeiter-API durchführen. Diese Aktionen ermöglichen es dem Admin, Kundendatensätze zu verwalten.

Mitarbeiter und Admin: Sowohl Mitarbeiter als auch Admin können dann Get (Abrufen), Post (Erstellen), Put (Aktualisieren) und Delete (Löschen) Operationen auf der Kunden-API durchführen.

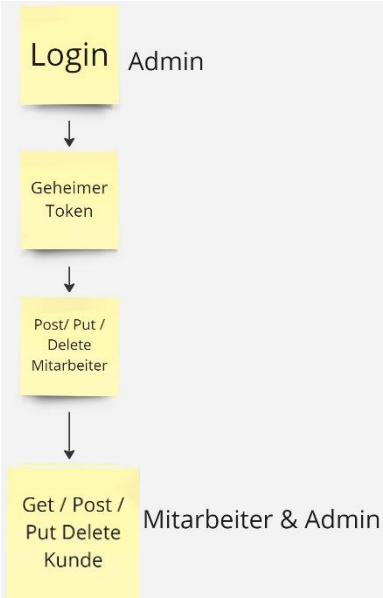

Abbildung 9: Mein Benutzerkonzept.

7 Kontrollieren

7.1 Ist der Auftrag fachgerecht und auftragsgerecht ausgeführt?

Die Ausführung des Auftrags war im Grossen und Ganzen sehr in Ordnung. Die fachgerechte und auftragsgerechte Umsetzung war gut erkennbar, jedoch gäbe es kleine Verbesserungsmöglichkeiten. Die erbrachte Arbeit erfüllte die gestellten Anforderungen, obwohl es noch Raum zur Optimierung geben könnte.

8 Auswerten

8.1 Wie war es, was muss nächstes Mal besser gemacht werden?

Im Allgemeinen hat mir das Projekt sehr Spass gemacht. Ich musste viel Zeit und Aufwand investieren und bin mit dem Ergebnis zufrieden. Durch manche Fehler habe ich viel Wichtiges gelernt.

8.2 Zeitmanagement

8.2.1 SOLL

- Geplant war im Grossen und Ganzen insgesamt 35 Stunden.

Beschreibung	Verantwortlich	Plan Arbeit d
Projekt 165		**35.00**
Informieren		**1.00**
Projektbeschreibung		1.00
Planen		**1.00**
Arbeitspakete Planen		1.00
Entscheiden		**1.00**
Funktionen Entscheiden		1.00
Realisieren		**25.00**
Arbeitspakete Realisieren		25.00
Kontrollieren		**6.00**
Testing		2.00
Dokumentation nachführen		4.00
Auswerten		**1.00**
Ergebniss auswerten		1.00

Abbildung 10: Zeitmanagement – SOLL.

8.2.2 IST

- Aufgrund von vielen Hindernissen habe ich mehr als die geschätzte Zeit gebraucht:

Beschreibung	Verantwortlich	Plan Arbeit d
Projekt 165		**42.00**
Informieren		**1.00**
Projektbeschreibung		1.00
Planen		**1.00**
Arbeitspakete Planen		1.00
Entscheiden		**1.00**
Funktionen Entscheiden		1.00
Realisieren		**32.00**
Arbeitspakete Realisieren		32.00
Kontrollieren		**6.00**
Testing		2.00
Dokumentation nachführen		4.00
Auswerten		**1.00**
Ergebniss auswerten		1.00

Abbildung 11: Zeitmanagement - IST.