2.1.2024

Author

Dokumentation: Ski-Service API

Modul 165 – NoSQL-Datenbanken einsetzen

Inhaltsverzeichnis

[Tabellenverzeichnis 1](#_Toc126430368)

[Abkürzungsverzeichnis (Chronologisch) 1](#_Toc126430369)

[Einleitung 2](#_Toc126430370)

[Verwendet Tools: 2](#_Toc126430371)

[Informieren 2](#_Toc126430372)

[Projektumfeld 2](#_Toc126430373)

[Projektorganisation 3](#_Toc126430374)

[Projektziele 3](#_Toc126430375)

[Planen 3](#_Toc126430376)

[Datenbank Möglichkeiten 4](#_Toc126430377)

[SQL-Daten in NoSQL Datenbank Migrieren 4](#_Toc126430378)

[Entscheiden 4](#_Toc126430379)

[Realisieren 5](#_Toc126430380)

[Datenbank 5](#_Toc126430381)

[API 6](#_Toc126430382)

[Backup & Restore 9](#_Toc126430383)

[Kontrollieren 11](#_Toc126430384)

[Auswerten 11](#_Toc126430385)

[Fazit 11](#_Toc126430386)

[Testprotokoll 12](#_Toc126430387)

[Abbildungsverzeichnis **Fehler! Textmarke nicht definiert.**](#_Toc126430388)

[Quellenverzeichnis: **Fehler! Textmarke nicht definiert.**](#_Toc126430389)

# Tabellenverzeichnis

[Tabelle 1: Grober Zeitplan der einzelnen Phasen 4](#_Toc126433977)

[Tabelle 2: Testprotokoll 12](#_Toc126433978)

# Abkürzungsverzeichnis (Chronologisch)

API Application Programming Interface

SQL Structured Query Language

NoSQL Not (only) SQL

VS Visual Studio

JSON JavaScript Object Notation

JWT JSON Web Tokens

# Einleitung

Das Projekt «Ski-Service API» zielte darauf ab, ein Backend zu erstellen. Dieses Projekt umfasst folgenden Punkte:

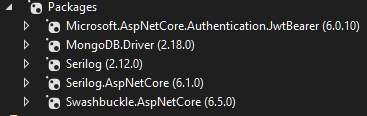
* Datenbankdesign und Implementierung (NoSQL)
* Datenmigration (SQL ➔ NoSQL)
* Migration WebAPI Projekt
* Testprojekt / Testplan
* Realisierung der kompletten Anwendung, gemäss den Anforderungen
* Durchführung Integrationstest mit bestehender Frontend Lösung.

Das Projekt wurde von me realisiert und dokumentiert. Die Dokumentation, sowie die Umsetzung des Ski-Service API basieren auf dem IPERKA-Modell. Das Vorgehen wurde phasenweise dokumentiert.

## Verwendet Tools:

Als IDE wurde Visual Studio 2022 (VS) 17.4.4 genutzt. Zusätzlich wurden mehrere NuGets von VS verwendet:

Abbildung 1: Nuget Liste



Für die Datenbank wurde MongoDB mit allen Tools, die MongoDB zur Verfügung stellt, verwendet. Für die Dokumentation der API Swagger und um diese zu testen, Postman.

## Informieren

Als Erstes las ich den Arbeitsauftrag durch und besprachen bestehende mit meinen Klassenkameraden. Nachdem die Unklarheiten geklärt waren, inspizierten ich die im Arbeitsauftrag beschriebenen Schritte zur Umsetzung des Projektes. Danach überlegte ich mir Möglichkeiten, um diese umzusetzen. Weiter habe ich mich über verschiedene Tools informiert, die ich für die Umsetzung des Projekts benutzen kann. Zusätzlich setzten ich mich mit den Bewertungskriterien auseinander. Anschliessend habe ich mich im Internet nach Möglichkeiten gesucht, wie ich dieses Projekt umsetzen könnte.

## Projektumfeld

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

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

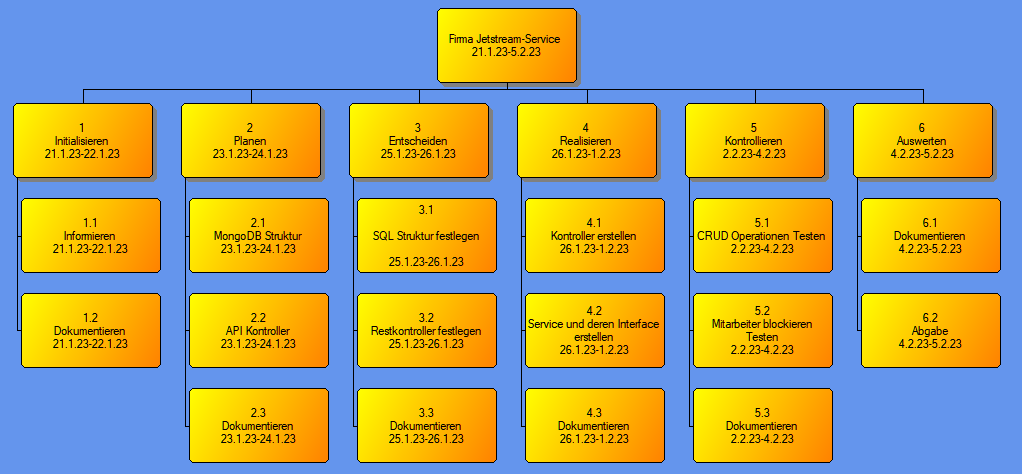
### Projektziele

Das Projekt wird mit dem Ziel durchgeführt, dass die Firma Jetstream-Service einen einfacheren Umgang mit ihren Kundendaten haben. Die Skalierungsmöglichkeit wird durch die Verwendung von MongoDB gewährleistet. So wird auch garantiert, dass die neu eröffneten Filialen kein Problem mit der Datenbank haben werden.

## Planen

Die Projektplanung werde ich nach dem IPERKA Modell machen. Der Grund dafür ist, dass ich schon viel mit diesem Modell gearbeitet habe und weil es für mich die beste Methode ist, um ein Projekt zu planen.

Abbildung 3: Projektplanung



**Ablauf- und Terminplanung**

1. Arbeitspakete erstellen

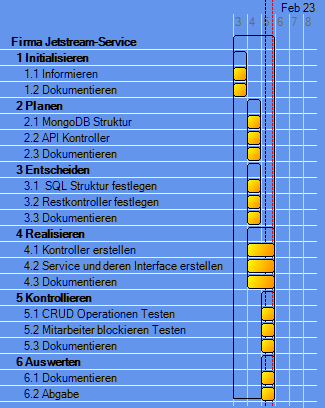
2. Reihenfolge der Arbeitspakete erstellen

3. Vorgangsdauer und Aufwand pro Arbeitspaket definieren.

4. Gesamtlänge und Gesamtaufwand des ganzen Projektes schätzen.

Nachdem ich diese Arbeitstakte erstellt, deren Reihenfolge und Aufwand geschätzt habe, entsteht folgende Planung:

Abbildung 4:GANTT für das Projekt



|  |  |
| --- | --- |
| Projektphase | Geplante Zeit |
| Informieren | **2 h** |
| Planen | **1 h** |
| Entscheiden | **0.5 h** |
| Realisieren | **4 h** |
| Kontrollieren | **1 h** |
| Auswerten | **4 h** |
| Gesamt | **12.5 h** |

Tabelle 1: Grober Zeitplan der einzelnen Phasen

### Datenbank Möglichkeiten

MongoDB ist eine NoSQL-Datenbank, die auf Dokumentenorientierung basiert. Das bedeutet, dass Daten in Form von Dokumenten gespeichert werden. Hierdurch ist es möglich, Daten einfach und flexibel zu modellieren, ohne dass es einer vorherigen Festlegung eines Schemas bedarf. Ausserdem bietet MongoDB eine hohe Skalierbarkeit.

Im Gegensatz dazu ist Neo4j eine Graphen-Datenbank, die auf der Verwendung von Knoten und Beziehungen zur Modellierung von Daten basiert. Es ist hervorragend für die Verwaltung von Beziehungen und Netzwerken geeignet, aber kann Schwierigkeiten haben, bei der Verarbeitung von Datenmengen oder bei der Skalierung auf mehrere Knoten.

### SQL-Daten in NoSQL Datenbank migrieren

Da ich schon eine laufende API habe, welches mit einer SQL-Datenbank verbunden ist, konnte ich so planen, dass ich die Daten einfach über die API abrufe, sie kopiere und anschliessend in die NoSQL-Datenbank einzufügen.

## Entscheiden

Aufgrund der Anforderungen des Projekts, bei dem es darum geht, grosse Datenmengen flexibel zu speichern und skalierbar zu sein, habe ich mich für MongoDB entschieden. Bei der Struktur der Dokumente habe ich mich für die simpelste Möglichkeit entschieden. So wird für jede Bestellung ein eigenes Dokument erstellt. Ich bin davon ausgegangen, dass die wenigsten Kunden mehrere Bestellungen machen werden und somit sich die Redundanz in Grenzen hält.

Für das Backup und Restore habe ich mich für die Tools "mongoimport" und "mongorestore" von MongoDB entschieden. Um die Tools aufrufen zu können, habe ich mich für das Benutzen von PowerShell entschieden.

Ausserdem habe ich mich dafür entschieden, dass ich den Grossteil der Logik in den Service-Klassen machen werde, um es übersichtlicher und auch einheitlicher zu halten. Für die Sicherheit habe ich mich gegen einen API-Key und für das JSON Web Token entschieden.

## Realisieren

### Datenbank

In dieser Phase begann ich mit dem eigentlichen Programmieren. Die Struktur der Datenbank ist wie Folgt:

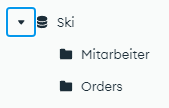


Abbildung 5: Datenbank mit zwei Collections



Abbildung 6: Index für die Collection Orders



Abbildung 7: Schema für die Collection "Orders"

Für die Administration der Datenbank habe ich drei User erstellt.



Abbildung 8: Autorisation aktivieren

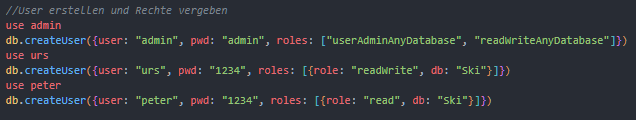


Abbildung 9: MongoDB User-erstellen

Um die Autorisation zu aktivieren, musste ich die Konfiguration von MongoDB anpassen. Diese findet man im MongoDB Installationsordner unter " MongoDB\Server\6.0\bin\mongod.cfg".

### API

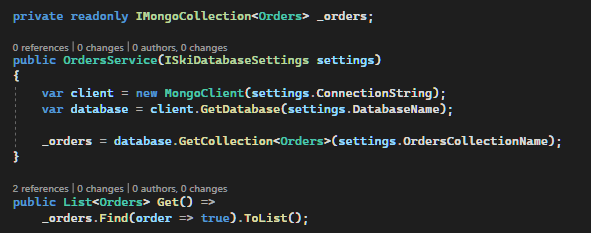
Der nächste Schritt war das Erstellen der Kontroller Klassen. Jeweils eine für die Registrationen und die Mitarbeiter. Anschliessen habe ich Interfaces der Services und die einzelnen Services erstellt und deren Logik geschrieben. Hier ein Beispiel eines Kontrollers:



Abbildung 10: Kontroller für Orders

Und die dazu passende Methode in der Service-Klasse, die im Kontroller aufgerufen wird:

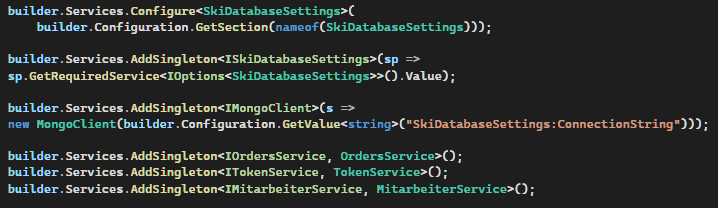
Abbildung 11: Methode, um alle Registrationen abzurufen.



In dem Konstruktor des Kontrollers gebe ich über Dependency Injection das Interface der dazugehörigen Service-Klasse mit, um eine Verbindung der beiden Klassen herzustellen. Im Kontroller rufe ich dann die Methode Get() in der Klasse OrdersService auf. In dieser Methode wird eine Verbindung zu der Datenbank hergestellt. Dieses Spiel habe ich für alle CRUD-Methoden wiederholt.

Damit die erstellten Services geladen werden, müssen wir diese in program.cs zuerst erstellen. Dies sieht wie folgt aus.

Abbildung 12: Erstellen der Services



Für die Sicherheit habe ich mich, wie bereits beschrieben, für ein JWT entschieden. Dies konnte ich als vorgefertigte Klasse übernehmen und musste es nur noch in das Projekt implementieren.Wichtig ist auch hier das Initialisieren in program.cs.

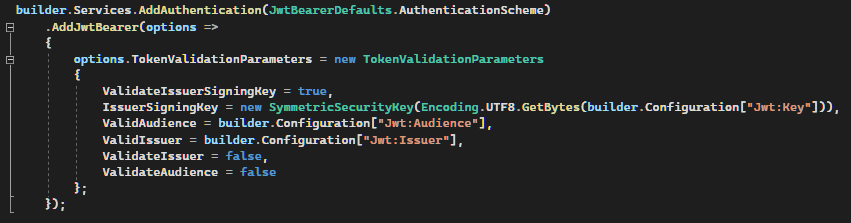


Abbildung 13: JWT-Initialisierung

Um mich mit der Datenbank verbinden und mich anmelden zu können, habe ich folgendes in das appsettings.json geschrieben. Hier habe ich das Passwort im Klartext. Ich habe mich gegen die Methode, die ich das letzte Mal verwendet habe, entschieden, da diese nicht zu meiner Zufriedenheit funktionierte.

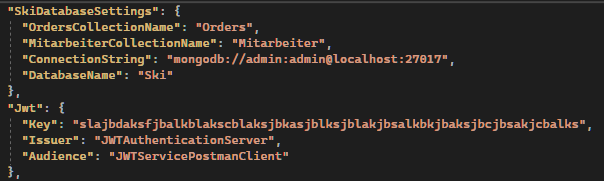


Abbildung 14: appsettings.json Beinhaltet die benötigten Informationen für die Verbindung und den JWT

Für die Anmeldung der Mitarbeiter, habe ich dieselbe Methode wie im letzten Projekt verwendet und musste diese nur anpassen, dass die Manipulationen gegen eine MongoDB geht und nicht mehr gegen eine SQL-DB.

Hier ein Überblick über das ganze API:



Abbildung 15: Klassen-Diagramm von API

### Backup & Restore

Für das Backup und Restore habe ich jeweils ein PowerShell-Skript gemacht, bei dem der User entscheiden kann, welche Datenbank wohin gespeichert werden soll oder welches Backup in welche Datenbank importiert werden soll. Diese Skripts laufen leider, nur wenn ich sie in der PowerShell ISE ausführe, wenn ich sie einfach mit der PowerShell ausführe, gelingt mir die Verbindung zu der Datenbank nicht.

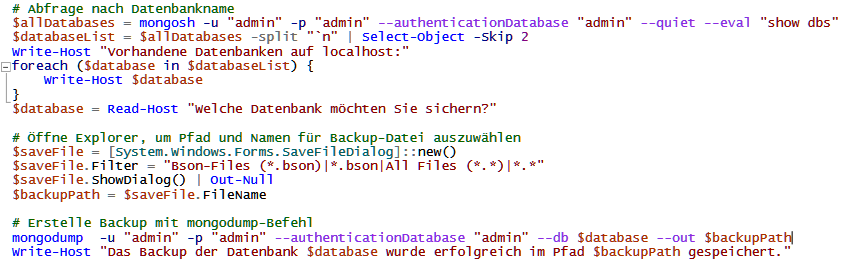


Abbildung 16: Backup-Skript für den User

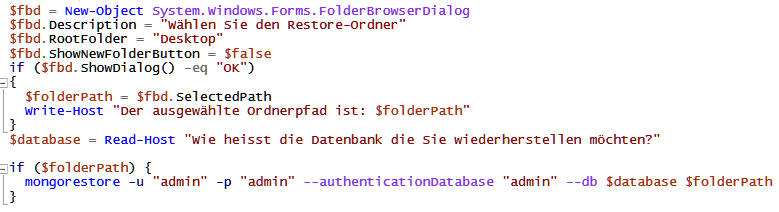


Abbildung 17: Restore-Skript für den User

Damit ich eine automatische Backuplösung machen konnte habe ich noch ein weiteres, einfacheres Skript geschrieben, welches auch auf der PowerShell läuft.

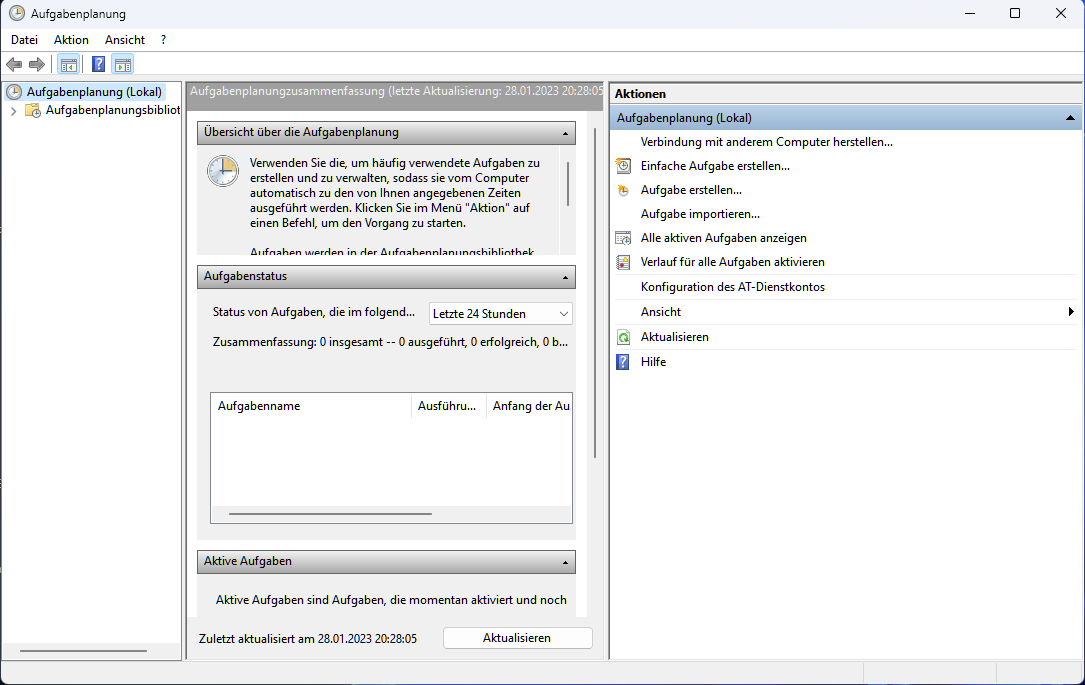


Abbildung 18:Aufgabenplanung



Abbildung 19: Backup-Skript für das automatische Backup

Um dieses Skript zu automatisieren, musste ich unter Aufgabenplanung eine neue Aufgabe erstellen.

Unter dem Reiter Trigger konnte ich einen neuen Auslöser des Skripts einstellen. Hier habe ich mich für ein tägliches um 20:33 Uhr Ausführen des Skripts entschieden.

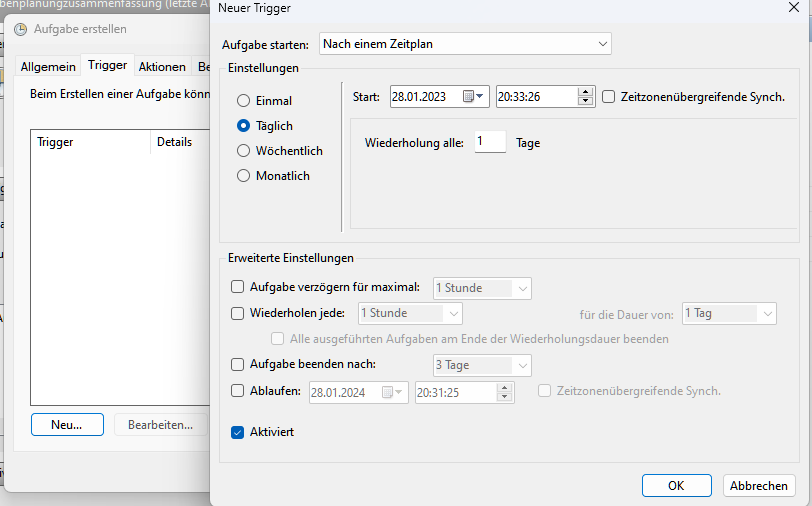


Abbildung 20: Trigger Einstellungen für die Aufgabe

Zum Schluss musste ich unter dem Reiter Aktion einstellen, was dann ausgeführt werden soll. Unter Programm habe ich die PowerShell angegeben und unter Argument den Pfad des Backup-Skriptes.

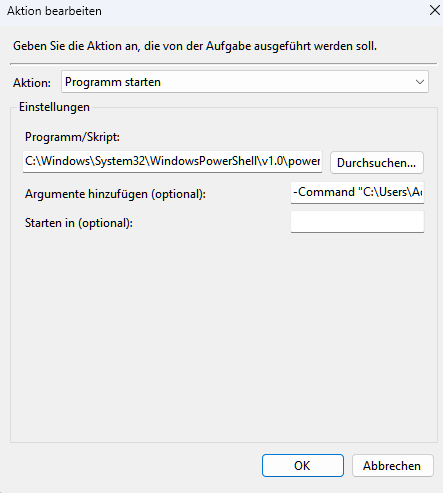


Abbildung 21: Aktion die die Aufgabe auslösen soll

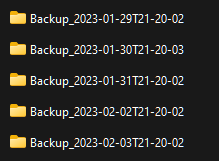


Abbildung 22: Backups die automatisch gemacht wurden

## Kontrollieren

Nachdem ich eine neue Methode implementiert habe, habe ich diese sogleich getestet. Dies mit allen CRUD-Methoden und den zusätzlichen, wie zum Beispiel das Blockieren eines Mitarbeiters oder auch das Erstellen des JWT. Zum Testen habe ich anfangs nur Swagger über den Browser und später Postman mit dem jeweiligen JWT benutzt.

Alle Skripte wurden getestet und wie bereits erwähnt, liefen die User-Skripte nur in der PowerShell ISE und ich konnte den Fehler nicht erkennen. Das Skript für das automatische Backup wurde durch die von mir erstellten Aufgabe getestet und wie auf Abb. 22 ersichtlich ist, hat dies auch funktioniert.

## Auswerten

Nach der Fertigstellung des Projektes habe ich mir Gedanken zu den positiven und negativen Momenten bei der Realisierung gemacht.

Ich hatte Anfangs grosse Probleme mit dem Management der Datenbank. Ich konnte keine Backups von einzelnen Datenbanken machen, weil ich anscheinend die Berechtigung nicht hatte, obwohl ich mich als "admin" angemeldet hatte. Erst als ich MongoDB erneut installiert und die Datenbanken und User erstellt hatte ging es ohne Probleme. Dieses Problem hatte ich nur am PC auf dem Laptop, hat es immer funktioniert.

# Fazit

Wie bei jedem Projekt, bei dem das Programmieren die Hauptsache ist, hatte ich auch bei diesem viel Spass. Leider konnte ich nicht so viel machen wie ich es gerne gemacht hätte, da wir in den letzten Wochen sehr viele andere Arbeiten zu erledigen hatten. Das Arbeiten mit NoSQL Datenbanken war für mich eine Freude, da dies viel eher meinem Verstand entspricht als SQL-Datenbanken, bei denen ich mich nicht so wohl fühle. Für mich ist das Arbeiten mit NoSQL Datenbanken, sei es MongoDB oder auch Neo4J viel einfacher als mit SQL-Datenbanken. Ich hoffe, dass ich in Zukunft noch mehr mit NoSQL-Datenbanken arbeiten werde.

# Testprotokoll

Tabelle 2: Testprotokoll

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Was wir Testen | Wie wir testen | Erwartetes Ergebnis | Tatsächliches Ergebnis | Testergebnis | Wurde es gelöst | Wie wurde es gelöst |
| 1 | Login der Mitarbeiter | Falsche Angaben | Fehlermeldung bei falsch Eingabe | Fehlermeldung "Falsche Angaben" | Fehlerfrei | - |  |
| 2 | Login der Mitarbeiter | Falsches Passwort | Falsche Angaben und +1 Zähler | Zähler blieb bei Null | Fehlerhaft | Ja | Zähler in DB gespeichert |
| 3 | Login der Mitarbeiter | Richtige Angaben | JWT erhalten | JWT erhalten | Fehlerfrei | - |  |
| 4 | Mitarbeiter deblockieren | Postman Put Mitarbeiter | Zähler wird auf 0 gesetzt | Zähler wurde auf 0 gesetzt | Fehlerfrei | - |  |
| 5 | Get All Mitarbeiter | Postman Get Mitarbeiter | Liste aller Mitarbeiter | Liste mit allen Mitarbeitern aber mit Passwörtern | Fehlerhaft | Ja | Passwörter vor der Ausgabe mit \* überschrieben |
| 6 | Get Priority Registration | Postman Get Abfrage | Ganzer Datensatz nach Priorität geordnet | Datensatz unvollständig | Fehlerfrei | - | Einbauen der Model-Klasse |
| 7 | Get All Registrationen | Postman Get Abfrage | Die Ausgabe aller  Registrationen | Ausgabe aller Registrationen | Fehlerfrei | - |  |
| 8 | Post Registration | Postman Post | Eintrag in die Datenbank | Eintrag wurde in Datenbank gespeichert | Fehlerfrei | - |  |
| 9 | Get by ID  Registration | Postman Get mit ID | Einzelner Datensatz der gewünschten ID | Einzelner Datensatz der gewünschten ID | Fehlerfrei | - |  |
| 10 | Put Registration | Postman Put mit ID | Datensatz wird geändert | Datensatz wurde geändert | Fehlerfrei | - |  |
| 11 | Delete Registration | Postman Delete mit ID | Datensatz wird gelöscht | Datensatz wurde gelöscht | Fehlerfrei | - |  |
| 12 | Autorisierung | CRUD-Abfragen | Nur mit gültigem JWT Möglich | War nur mit gültigem JWT möglich | Fehlerfrei | - |  |
| 13 | Backup-Skript | Skript ausführen | Backup einer bestimmten Datenbank wird erstellt | Nur Backup von allen Datenbanken konnte erstellt werden | Fehlerhaft | Ja | Neuinstallation von MongoDB konnte das Problem beseitigen |
| 14 | Automatisches Backup | Wird die Aufgabe ausgeführt | Erstellen eines Backups zu einer bestimmten Zeit | Backup wurde erstellt | Fehlerfrei | - |  |
| 15 | Restor-Skript | Skript ausführen | Datenbank wurde wiederhergestellt | Datenbank wurde wiederhergestellt | Fehlerfrei | - |  |

# Abbildungsverzeichnis

[Abbildung 1: Nuget Liste 2](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439230)

[Abbildung 3: Projektplanung 3](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439232)

[Abbildung 4:GANTT für das Projekt 4](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439233)

[Abbildung 5: Datenbank mit zwei Collections 5](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439234)

[Abbildung 6: Index für die Collection Orders 5](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439235)

[Abbildung 7: Schema für die Collection "Orders" 5](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439236)

[Abbildung 8: Autorisation aktivieren 5](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439237)

[Abbildung 9: MongoDB User-erstellen 5](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439238)

[Abbildung 10: Kontroller für Orders 6](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439239)

[Abbildung 11: Methode, um alle Registrationen abzurufen. 6](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439240)

[Abbildung 12: Erstellen der Services 7](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439241)

[Abbildung 13: JWT-Initialisierung 7](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439242)

[Abbildung 14: appsettings.json Beinhaltet die benötigten Informationen für die Verbindung und den JWT 7](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439243)

[Abbildung 15: Klassen-Diagramm von API 8](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439244)

[Abbildung 16: Backup-Skript für den User 9](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439245)

[Abbildung 17: Restore-Skript für den User 9](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439246)

[Abbildung 18:Aufgabenplanung 9](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439247)

[Abbildung 19: Backup-Skript für das automatische Backup 9](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439248)

[Abbildung 20: Trigger Einstellungen für die Aufgabe 10](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439249)

[Abbildung 21: Aktion die die Aufgabe auslösen soll 10](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439250)

[Abbildung 22: Backups die automatisch gemacht wurden 10](https://ipsobildung-my.sharepoint.com/personal/simon_stauffer_student_ipso_ch/Documents/Schule/Modul%20165/Projektarbeit/Dokumentation/SkiServiceNoSQLDokumentation.docx#_Toc126439251)

# Quellen

https://www.youtube.com/watch?v=iWTdJ1IYGtg

https://stackoverflow.com/questions/49533659/mongodb-projection-in-c-sharp

https://stackoverflow.com/questions/10169064/mongodb-authentication-with-connection-string

https://stackoverflow.com/questions/41615574/mongodb-server-has-startup-warnings-access-control-is-not-enabled-for-the-dat

www.mongodb.com/community/forums