Projekt Dokumentation

Modul 165

Maximilian Läpple

Inhaltsverzeichnis

1. Informieren

Die Jetstream-Service GmbH, ein KMU im Bereich Ski-Service, steht vor der Herausforderung, ihre wachsende Kundenbasis und die Diversifizierung an neuen Standorten effizient zu managen. Die bisherige relationale Datenbanklösung stößt an ihre Grenzen, insbesondere was die Skalierbarkeit und Datenverteilung betrifft. Vor diesem Hintergrund zielt das Projekt darauf ab, die bestehende Datenbankinfrastruktur auf ein flexibles und skalierbares NoSQL-Datenbanksystem zu migrieren. Dieser Schritt ist entscheidend, um den gestiegenen Anforderungen gerecht zu werden und gleichzeitig Kosten zu optimieren. Das Projekt umfasst die Neugestaltung des Datenbankdesigns, die Migration von bestehenden Daten, die Anpassung der WebAPI und die Sicherstellung der Funktionsfähigkeit durch umfassende Tests. Durch die Implementierung eines modernen NoSQL-Systems sollen die Verwaltung von Serviceaufträgen optimiert, die Kundenzufriedenheit gesteigert und die interne Effizienz verbessert werden.

Anforderungen

Für das Projekt "Ski-Service Auftragsverwaltung" sind folgende Anforderungen definiert:

Datenmigration

Sichere Übertragung bestehender Daten aus dem relationalen System in das neue NoSQL-Datenbanksystem ohne Datenverlust.

Datenintegrität und-sicherheit

Gewährleistung von Datenintegrität und die Implementierung von Sicherheitsmaßnahmen zum Schutz sensibler Kunden- und Geschäftsdaten.

Skalierbarkeit und Performance

Das neue System muss leicht skalierbar sein, um mit dem Wachstum des Unternehmens Schritt halten zu können und eine hohe Abfrageleistung zu bieten.

Benutzerfreundlichkeit: Die WebAPI und die zugehörigen Anwendungen müssen benutzerfreundlich sein, um eine effiziente Verwaltung von Serviceaufträgen zu ermöglichen.

Erwarteter Nutzen:

Durch die Umsetzung des Projekts erwartet die Jetstream-Service GmbH folgenden Nutzen:

Verbesserte Effizienz

Durch die Automatisierung und Optimierung der Auftragsverwaltung wird eine höhere Effizienz bei der Bearbeitung von Kundenanfragen erreicht.

Höhere Kundenzufriedenheit

Schnellere Bearbeitungszeiten und verbesserte Servicequalität führen zu einer gesteigerten Kundenzufriedenheit.

Kosteneinsparungen

Die erhöhte Effizienz und die Skalierbarkeit des neuen Systems tragen zur Reduzierung der Betriebskosten bei.

Zukunftssicherheit: Die Implementierung eines modernen und flexiblen Datenbanksystems sichert das Unternehmen gegen zukünftige Herausforderungen ab und bietet Raum für Innovation.

Planen

Konvertierung von SQL in NoSQL

Meine jetzige Datenbank ist in MSSQL, diese werde ich als erstes konvertieren, indem ich mir meine 4 Tabellen anschaue und sie einfach konvertiere

Tabellen

- dbo.Benutzer
- dbo.Bestellungen
- dbo.Mitarbeiter
- dbo.Status

Diese 4 Tabelle werden ich in Json umwandeln dabei werden die Zeilen und die Spalten einfach kopiert.

Als erstes werden die Vorhandenen FK und PK gelöscht, so das danach nur noch die Tabellen ohne Verbindungen existieren, das erleichtert die umwandelung

Fehler Behandlung

Für meine Fehler Behandlung habe ich mich bei PUT und POST für ein Datenbank zugriff entschieden die wird bei den 2 Endpoints eingesetzt.

Die schaut ob bei dem Service, Status oder Mitarbeiter ein gültiger wert eingeben wird sonst wird eine Exception geworfen, um zu verhindern das der code abstürzt.

Endpoints

Beim Planen der Web API-Endpoints für das "Ski-Service Auftragsverwaltung"-Projekt sollten RESTful Prinzipien angewendet werden. Entwirf intuitive Endpoints für Aktionen wie Anmeldung, Auftragserstellung und Statusupdates, z.B. /api/benutzer für Benutzerverwaltung und /api/bestellungen für Aufträge. Konzentriere dich auf die Verwendung der Methoden GET, POST, PUT, und DELETE zur Manipulation der Daten und stelle sicher, dass Validierung und Authentifizierung implementiert sind, um die Sicherheit zu gewährleisten.

Beispiele

Bestellung

- GET
- POST
- GET by id
- PUT by id
- Delete by id (nur wenn angemeldet nutzbar)

Login

- GET (Mit Jwt verschlüsselt)

Zeit Anspruch vor dem Projekt

Informieren	Soll Zeit
Planen	2h
Entscheiden	0.5h
Realisieren	12h
Kontrollieren	3h
Auswerten	1h

Entscheiden

Arbeitsumgebung

Programme

- Visual Studio
- Postmen
- MongoDB

Sprachen

- C#
- Mongo Shell
- Json
- JavaScript

NuGet Pakete

- Microsoft.AspNetCore.Authentication.JwtBearer Version 7.0.15
- Microsoft.AspNetCore.OpenApi
- MongoDB.Bson
- MongoDB.Driver
- Swashbuckle.AspNetCore

Festlegung der Datenmigrationsstrategie

Bei der Planung der Datenmigration von SQL zu NoSQL wird eine genaue Analyse der aktuellen SQL-Strukturen vorgenommen. Basierend darauf entwickeln wir eine Strategie, die eine nahtlose Überführung unter Beibehaltung der Datenintegrität gewährleistet. Die Anpassung an NoSQL-spezifische Vorteile wie Skalierbarkeit und flexible Datenabfragen ist dabei zentral.

Sicherheitskonzepte

Im Projekt wurde JWT zur Absicherung verwendet, indem es für Authentifizierung und Autorisierung eingesetzt wird. Es ermöglicht die sichere Übermittlung von Informationen zwischen Nutzern und Server, in dem Fall wurde es für die Anmeldung benutzt

Backup- und Wiederherstellungsstrategien

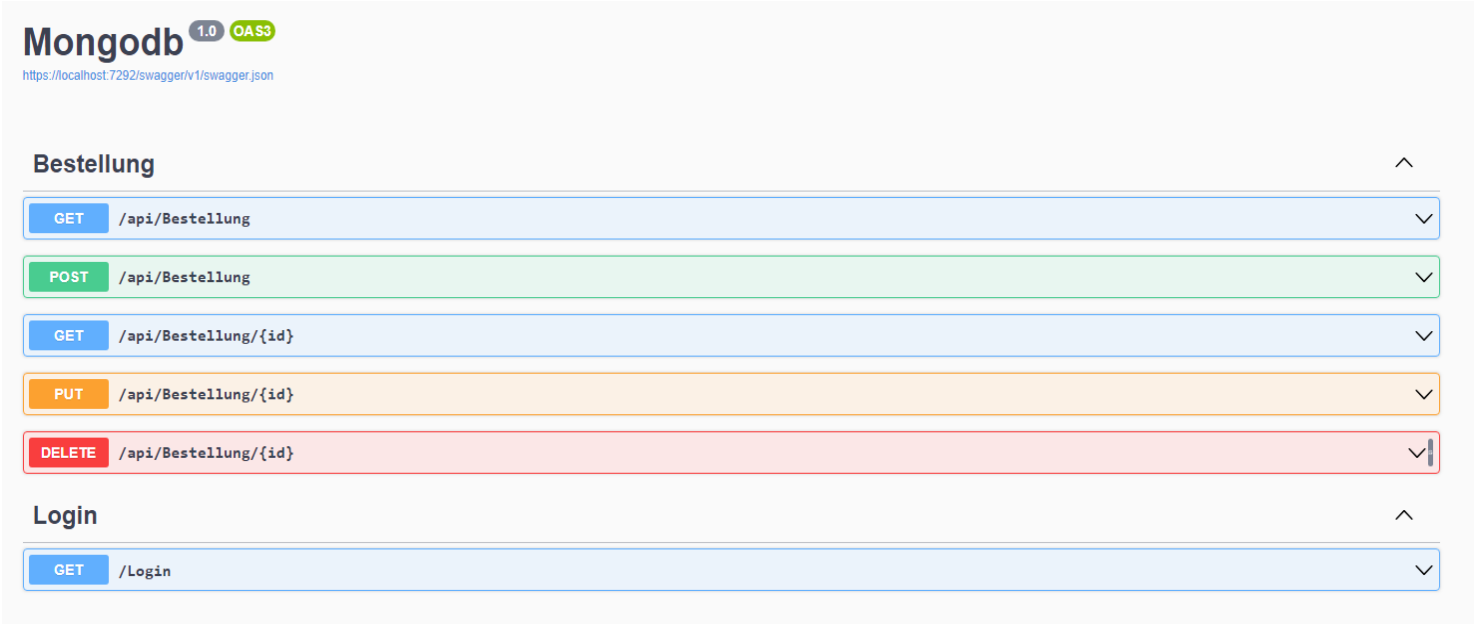
Hier habe ich in GitHub ein paar Kommands ins Readme geschrieben damit Backups erstellt werden können und bei Fehlern wieder auf eine sichere Version wiederhergestellt werden kann.
Dies sollte man so oft es geht machen oder bei grossen Änderungen.

Realisieren

Anpassung der WebApi

Bei der Anpassung der WebAPI für das Projekt "Ski-Service Auftragsverwaltung" wurde die Schnittstelle so überarbeitet, dass sie effektiv mit der neuen NoSQL-Datenbank kommuniziert. Dies umfasste die Aktualisierung von Endpunkten, um die dokumentenorientierte Datenstruktur zu unterstützen, und die Optimierung von Abfragen für verbesserte Leistung. Sicherheitsmechanismen wie Authentifizierung und Autorisierung wurden ebenfalls verstärkt, um die Integrität und den Schutz der Daten zu gewährleisten. Die Anpassung ermöglichte eine reibungslose Integration mit der Frontend-Anwendung und unterstützte die effiziente Verwaltung von Ski-Serviceaufträgen.

Swagger Darstellung



Datenmigration

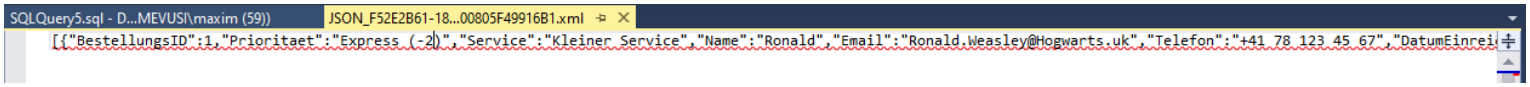
Hier habe ich ein einfachen Query in SQL benutzt, um die Daten in Json zu konvertieren dies war für mich die einfache Methode, um die Daten nach MongoDB zu senden

Query für Konvertierung in Json

```
SELECT *
FROM [SkiService].[dbo].[TabellenName]
FOR JSON AUTO;
```

Dies habe ich für alle 4 Tabellen gemacht, wie in der Planung erwähnt.

Der Output sah so aus



Dies habe ich dann also gut geordnete Json Datei umgewandelt und in MangoDb in Atlas importiert, dies war meiner Meinung nach die einfachste Methode.

Integrationstests

Das Testen habe ich sehr früh im Projekt angefangen. Das Einfachste, was ich zuerst gemacht habe, ist die Verbindung mit einem einfachen GET von meinen Bestellungen zu machen, um nicht alle Endpoints zu machen und nachher debuggen.

Das hatte viel länger gebraucht und war auch eine gute Methode und den Fortschritt erkennen können, dabei konnte ich auch einfach ein commit machen um alles in GitHub Dokumentieren zu können.

Danach habe ich dann alle Endpoints einzeln implementiert und auch immer wieder getestet mithilfe von Postmen, dort werden wieder aber im Detail in Kontrollieren wieder zurückkomme.

Kontrollieren

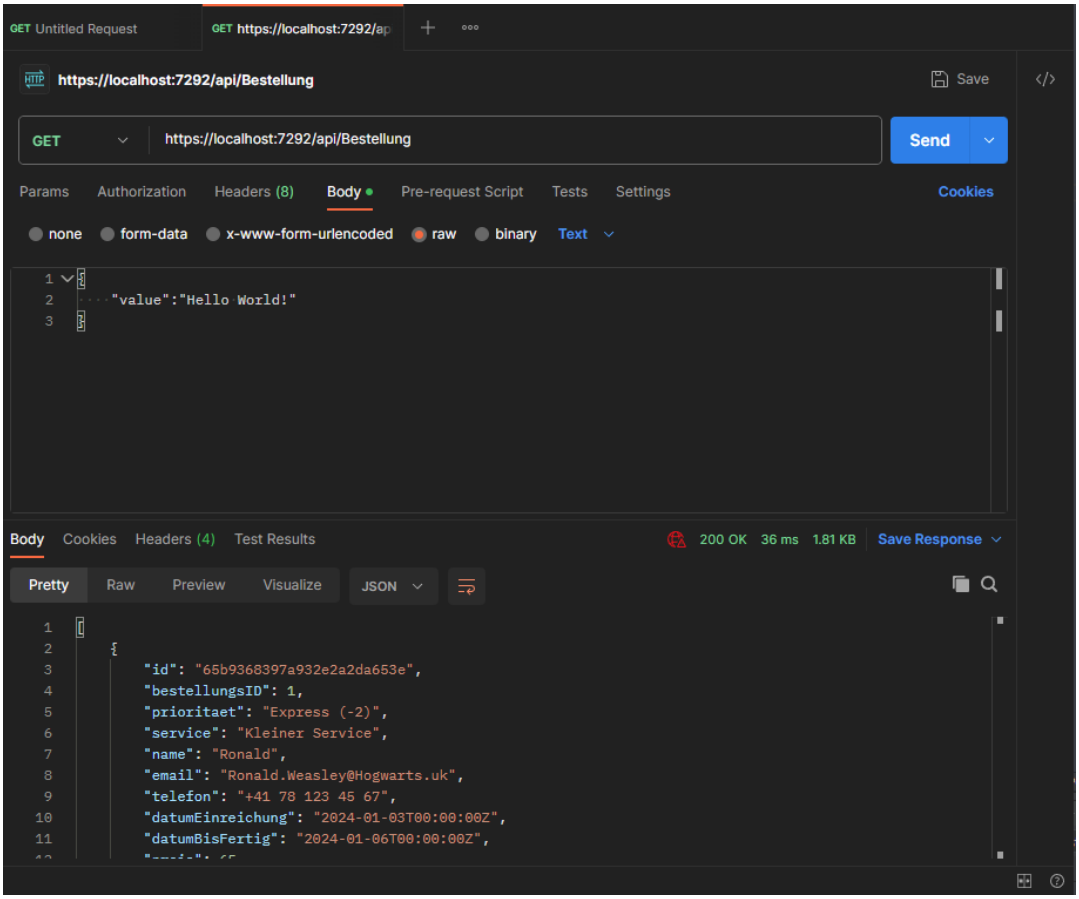
Funktionalitätstests

Im Rahmen der "Kontrollieren"-Phase wurden Funktionalitätstests durchgeführt, indem alle WebAPI-Endpoints systematisch in Postman getestet wurden. Diese Tests bestätigten die korrekte Funktion der Schnittstellen nach der Migration auf NoSQL und der Anpassung der WebAPI. Jeder Endpoints wurde auf korrekte Antwortcodes (200), erwartete Ausgaben und die Handhabung von Fehlerfällen geprüft. Diese umfassenden Tests stellten sicher, dass die API die erwarteten Anforderungen erfüllt und die Integration mit der Frontend-Anwendung reibungslos funktioniert. Diese Test wurden mit Postmen gemacht, um eine gute Antwort mit möglichst vielen Detail zu bekommen.

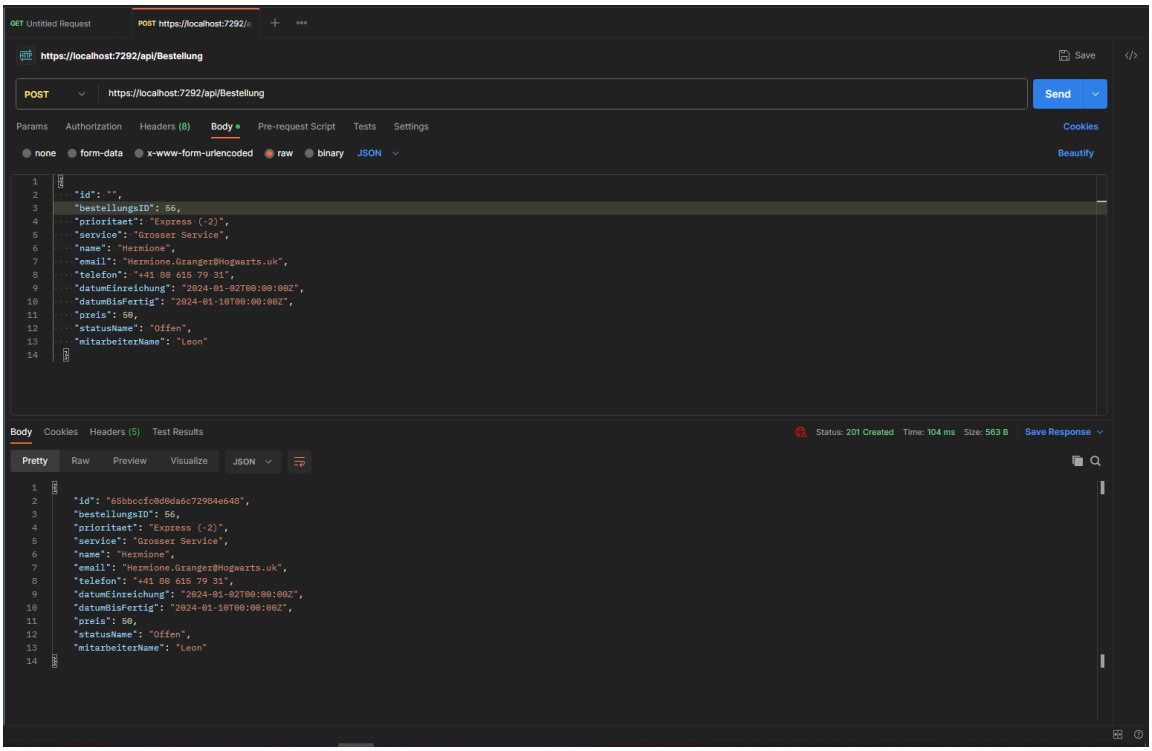
Test (Bilder)

Hier werden alle Bilder zu dem Test gezeigt, sowie auch falsche angaben da eine Fehlerbehandlung bei Mitarbeiter, Service und Status eingebaut wurde

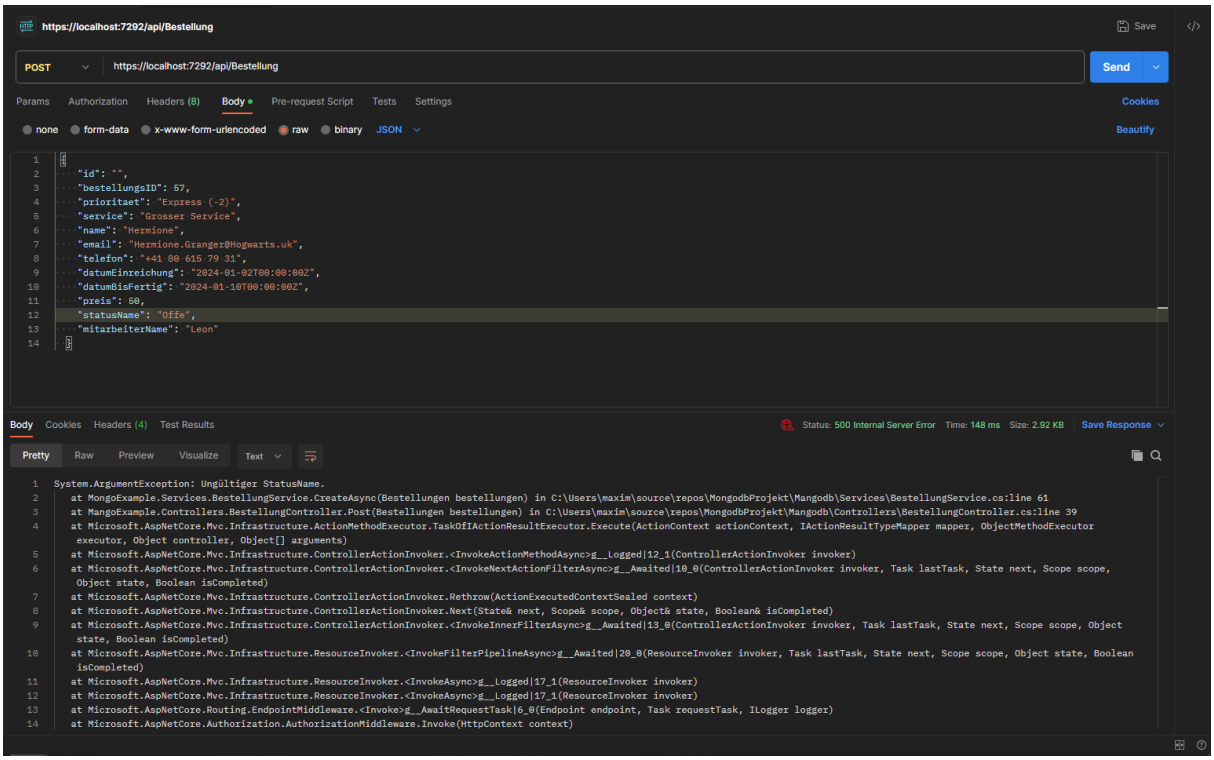
GET



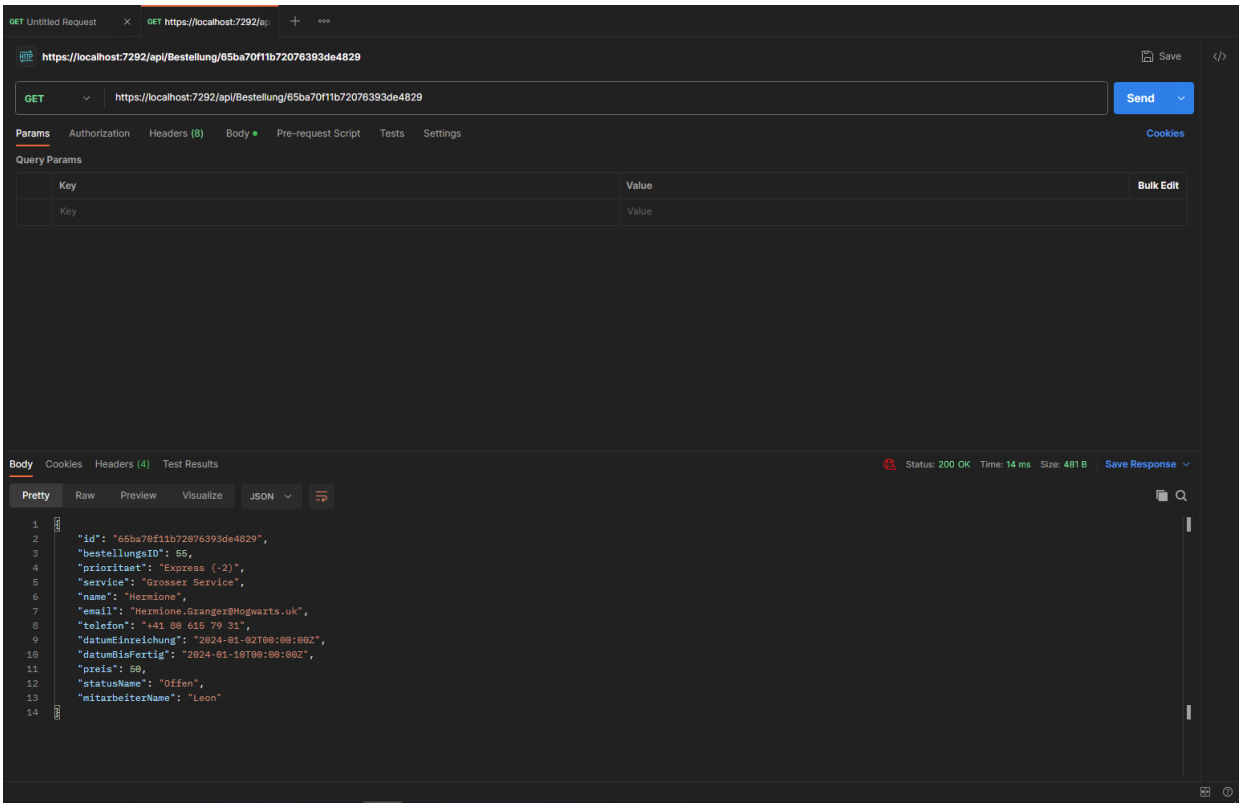
POST



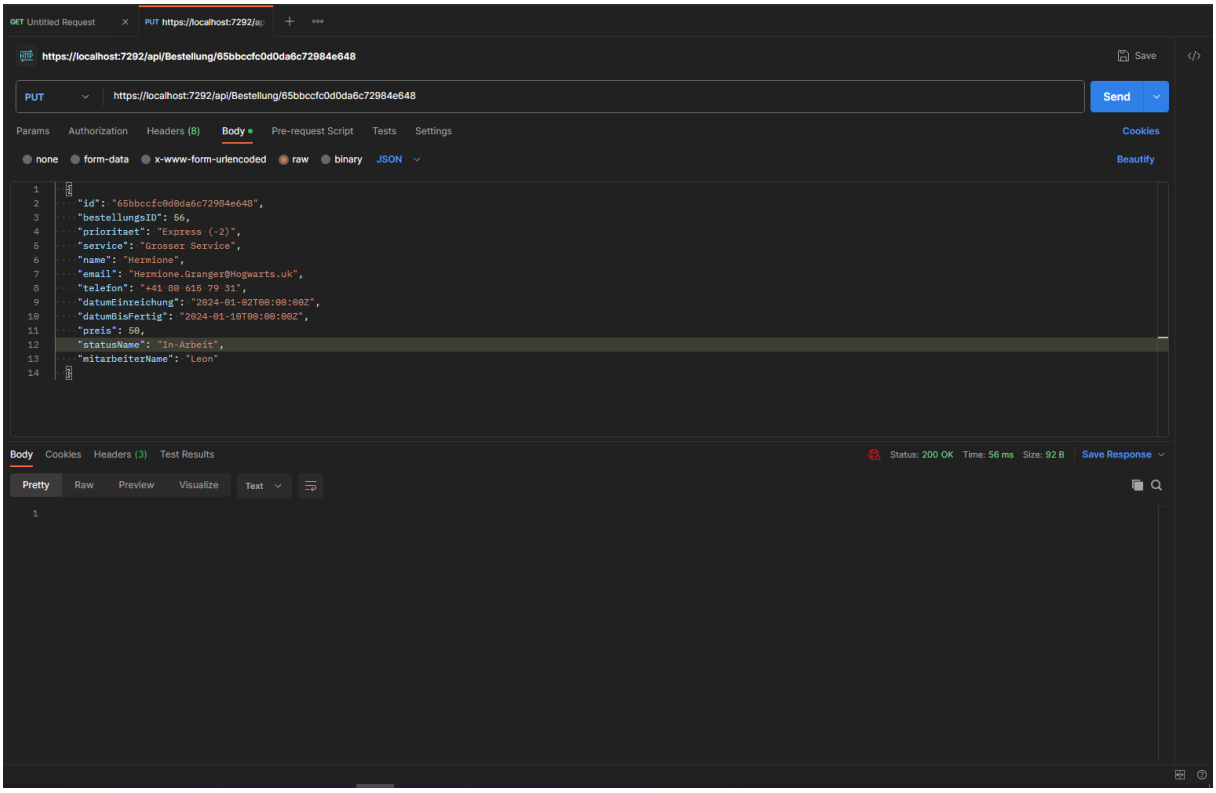
Mit falschen Angaben, ein Kommentar ist oben auch zusehen wenn etwas falsch ist



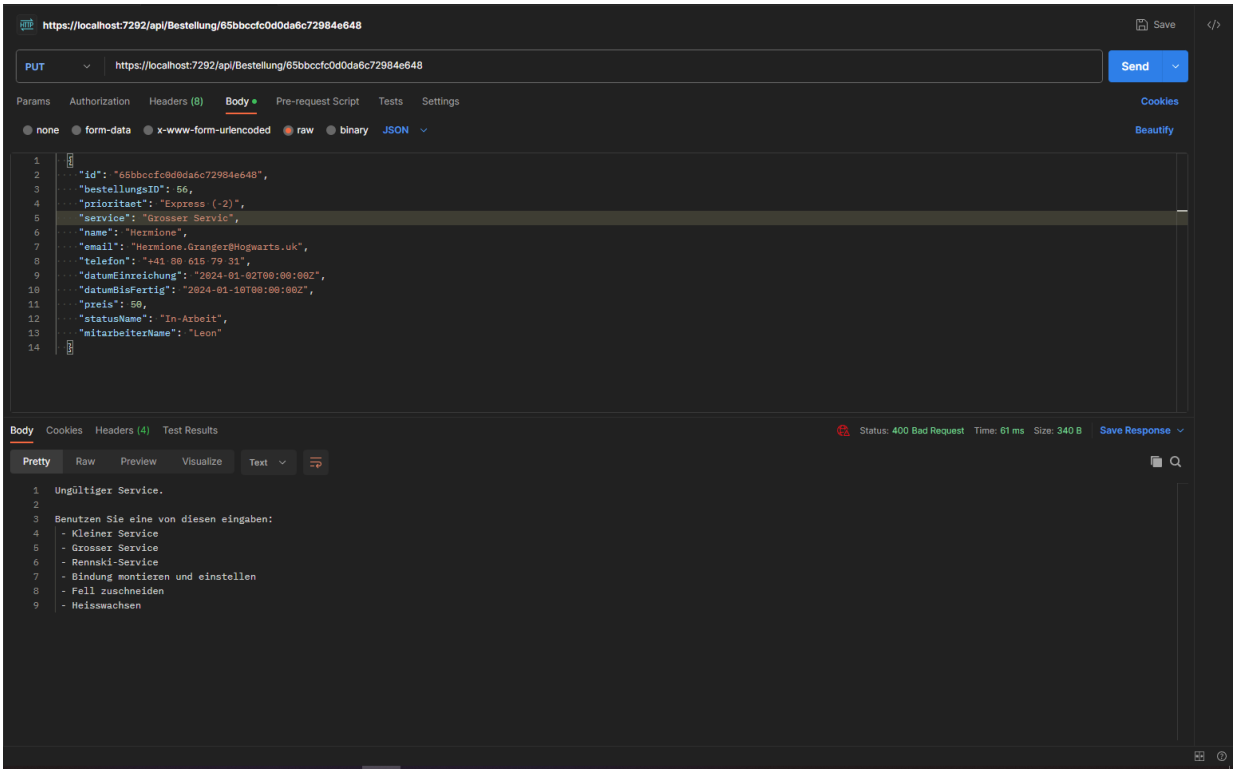
GET by id

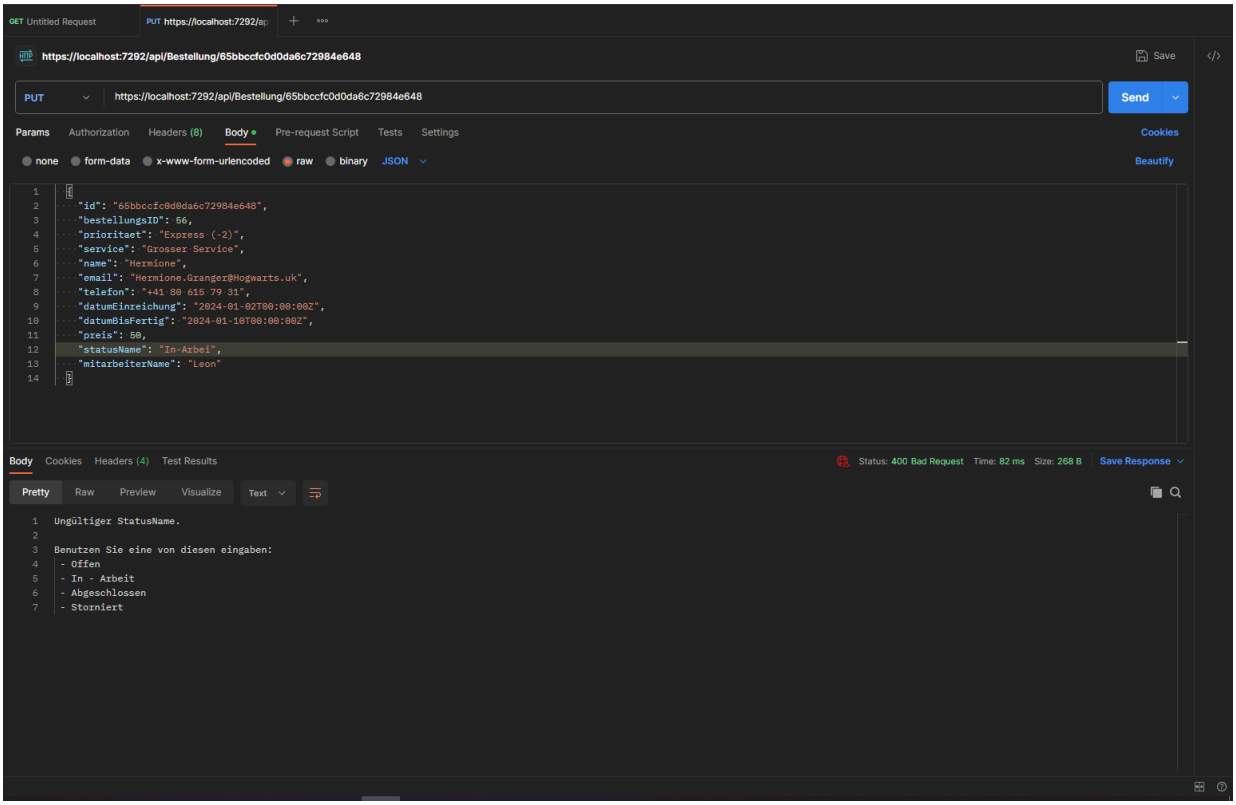

GET Untitled Request
GET https://localhost:7292/ap
https://localhost:7292/api/Bestellung/65ba70f11b72076393de4829
Save
GET
https://localhost:7292/api/Bestellung/65ba70f11b72076393de4829
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
Query Params
Key
Value
Bulk Edit
Key
Value
Body Cookies Headers (4) Test Results
Status: 200 OK Time: 14 ms Size: 481 B
Save Response
Pretty Raw Preview Visualize JSON
{
"id": "65ba70f11b72076393de4829",
"bestellungsID": 55,
"prioritaet": "Express (-2)",
"service": "Grosser Service",
"name": "Hermione",
"email": "Hermione.Granger@Hogwarts.uk",
"telefon": "+41 80 615 79 31",
"datumEinreichung": "2024-01-02T00:00:00Z",
"datumBisFertig": "2024-01-10T00:00:00Z",
"preis": 50,
"statusName": "Offen",
"mitarbeiterName": "Leon"
}

PUT

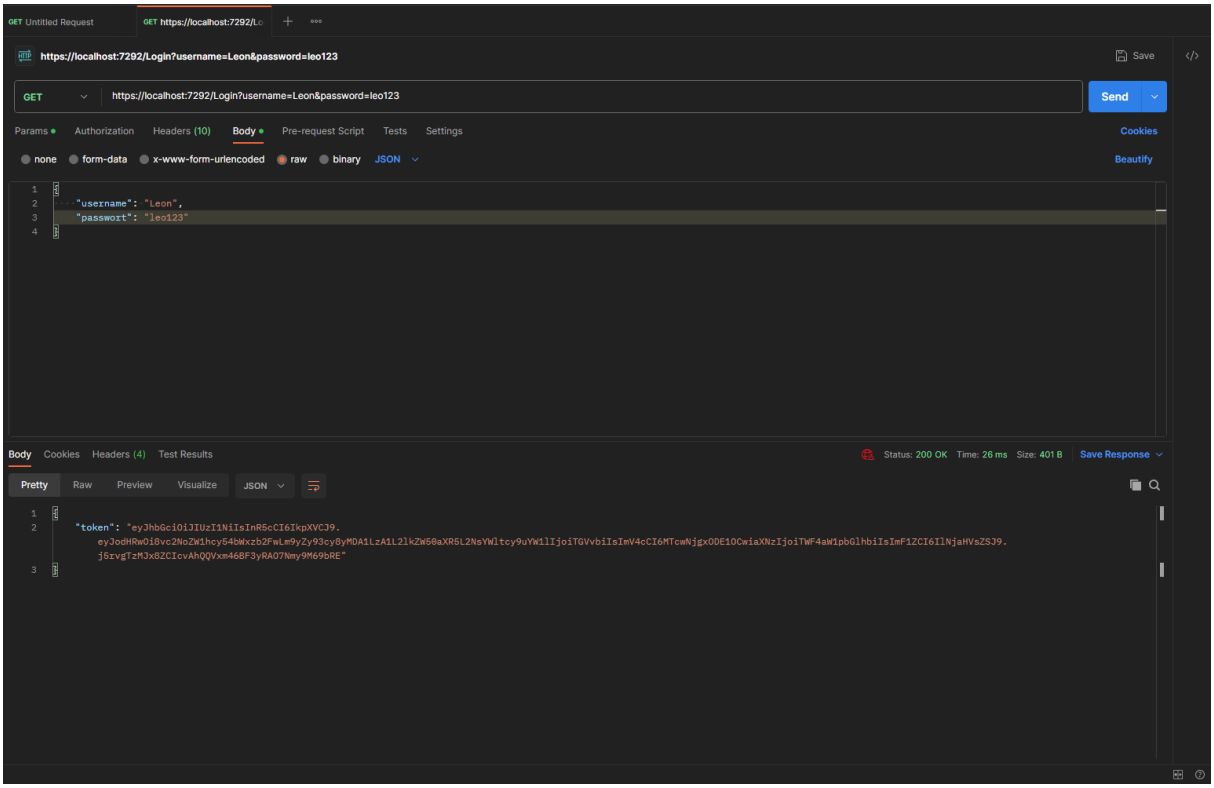

GET Untitled Request
PUT https://localhost:7292/ap
https://localhost:7292/api/Bestellung/65bbccfc0d0da6c72984e648
Save
PUT
https://localhost:7292/api/Bestellung/65bbccfc0d0da6c72984e648
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary JSON
Beautify
{
"id": "65bbccfc0d0da6c72984e648",
"bestellungsID": 56,
"prioritaet": "Express (-2)",
"service": "Grosser Service",
"name": "Hermione",
"email": "Hermione.Granger@Hogwarts.uk",
"telefon": "+41 80 615 79 31",
"datumEinreichung": "2024-01-02T00:00:00Z",
"datumBisFertig": "2024-01-10T00:00:00Z",
"preis": 50,
"statusName": "In-Arbeit",
"mitarbeiterName": "Leon"
}
Body Cookies Headers (3) Test Results
Status: 200 OK Time: 56 ms Size: 92 B
Save Response
Pretty Raw Preview Visualize Text

PUT mit falschen Angaben





GET-Login mit Jwt Token Ausgabe



JS-Skript

Der JavaScript und die nötige Datenbank zu erstellen wurden auch getestet, um mögliche Fehler auszuschliessen

Hierzu habe ich ein zweiter PC benutzt, um Konflikte mit der bestehenden Datenbank auf meinen PC zu verhindern.

Das Skript kann auf GitHub gefunden werden und ist mit PowerShell zu benutzen oder auch Mongo Shell, der kann einfach gestartet werden und die Datenbank wird einfach erstellt.

Durch die Ausführung dieses Skripts wird sichergestellt, dass die Datenbankstruktur konsistent bleibt und die Entwicklung oder Inbetriebnahme auf neuen Systemen vereinfacht wird. Es dient als effizientes Tool für die schnelle Einrichtung der Entwicklungs- oder Testumgebung.

Auswertung

Ergebnis und Lernerfahrungen

Im Rahmen des Projekts "Ski-Service Auftragsverwaltung" wurden wesentliche Ergebnisse erzielt, die die Effizienz und Benutzerfreundlichkeit des Serviceprozesses deutlich verbesserten. Durch die Migration auf eine NoSQL-Datenbank und die Anpassung der WebAPI konnten Flexibilität und Performance gesteigert werden.

Die Lernerfahrungen umfassen tiefergehende Kenntnisse in der NoSQL-Datenbanktechnologie, das Verständnis für die Bedeutung gründlicher Tests und die Erkenntnis, dass klare Kommunikation Planung sehr wichtig sowie sich gut vor einem Projekt zu Informieren und so wenig Fehler am Anfang zu machen, dies brauch andersherum viel mehr Zeit.

Jwt

Beim Umgang mit JWT (JSON Web Tokens) im Projekt lernte mir, wie wichtig es ist, sich mit Sicherheitsaspekten wie Token-Ablauf, Speicherung und Übertragung auseinanderzusetzen.

Die Implementierung ist bestimmt das Schwierigste im Projekt da ich noch wenig Erfahrungen damit hatte und auch zuerst verstehen musste, wie dies funktioniert sowie gut im Projekt implementiert werden soll.

Zeit Anspruch Nach Projekt

Informieren	Soll Zeit	Ist Zeit
Planen	2h	1.5h
Entscheiden	0.5h	0.5h
Realisieren	10h	14h
Kontrollieren	3h	3h
Auswerten	1h	1.5h