

# Dokumentation

## Modul 295

Fach: Modul 295

Dozent: Lukas Müller

Autoren: Nelo Nisslé, Muhammet Topal

Datum: 13.12.24

Standort: Basel

## Inhaltsverzeichnis

Einführung .....	4
Informieren .....	5
Ausgangslage: .....	5
Ressourcen: .....	5
Ziel Auswertung .....	6
Planen .....	8
Projektstruktur: .....	8
Skizzen .....	9
Entscheiden .....	10
Warum Bootstrap? .....	10
Warum Logger? .....	10
Technologieentscheidung .....	10
Realisieren .....	11
Code Der Webseite .....	11
Hilfs Mittel bei der Umsetzung .....	11
Verbindung mit JavaScript: .....	12
Backend Code: .....	12
Database und Migration: .....	13
Sicherheit (JWT) .....	14
Controller im Backend: .....	14
Kontrollieren .....	15
Testen mit Postman: .....	15
Frontend Test: .....	17
Vollständigkeit liste: .....	18
Auswerten .....	19
Anleitung für admin page: .....	19
Swagger Dokumentation: .....	20
Lessons Learned: .....	21

Verbesserungen in der Zukunft .....	21
Fazit: .....	22
Referenzen s Quellen.....	23
Quellen:.....	23

## Einführung

Hier werden wir sie kurz einführen in unsere Projekt Arbeit.

In unserem Projekt geht es darum eine Backend für unsere Interaktive Webseite zu erstellen die mit unserem selbst erstellten backend kommunizieren muss um Daten auszutauschen die wir dann mit den CRUD Operationen behandeln konnten. Das Backend muss in C# geschrieben sein da wir diese Sprache auch noch in Zukunft verwenden werden und sie am besten zu dieser Aufgabe gepasst hat. Um das Ganze so schnell wie möglich in die Realität umzusetzen haben wir uns nach IPERKA Orientiert. Außerdem mussten wir uns an den Vorgegeben Zielen Orientieren und diese Nach Dringlichkeit auswerten. Zu dem hatten wir uns auch eine grobe Skizzen Planung erstellt für unsere Webseite.

## Informieren

In dem Ersten Schritt «Informieren» haben wir uns vor allem auf die Information Beschaffung und auf die Zielsetzung Fokussiert. Damit wir in den späteren teilen eine festes ziel haben auf das wir uns voll konzentrieren können.

### Ausgangslage:

Das Ist die Ausgangslage die wir auf GitHub vorgegeben bekommen haben.

Die Firma Jetstream-Service führt als KMU in der Wintersaison Skiservicearbeiten durch und will im Zuge der Digitalisierung die interne Verwaltung der Ski-Service Aufträge komplett über eine Web- und Datenbank basierte Anwendung abwickeln. Die bereits existierende Online-Anmeldung soll bestehen bleiben und mit den erforderlichen Funktionen für das Auftragsmanagement erweitert werden. In der Hauptsaison sind bis zu 10 Mitarbeiter mit der Durchführung der Serverarbeiten beschäftigt. Diese sollen einen autorisierten passwortgeschützten Zugang zu den anstehenden Aufträgen erhalten und diese zur Abarbeitung übernehmen und ändern können.

### Ressourcen:

Wir haben Verschiedenste Tools und andere Anwendungen verwendet um unser Projekt umzusetzen.

- Visual Studio Code: Für die Entwicklung der Webseite und des Backends.
- Bootstrap: Zur Umsetzung eines responsiven Webdesigns und für vorgefertigte UI-Komponenten.
- C#: Für die Umsetzung des backends.
- Postman: Für das Testen der REST-API und die Überprüfung der Datenübertragung.
- GitHub: Zur Versionskontrolle und Zusammenarbeit im Projekt.

Wir haben uns Hier Auch noch Gedanken gemacht bezüglich Verwendung von Hilfs mitteln und sind zum Entschluss gekommen das wir möglichst ohne Hilfsmittel arbeiten wollen aber wenn es Dazu kommen sollte das wir welche benötigen brauchen wir diese dann auch.

## Ziel Auswertung

Hier haben wir erstmals die ziele und auch gleich in einer weiteren Tabelle die ziel Auswertung und Priorität.

Hier die Anforderungen an unsere Webseite:

### **Nr. Beschreibung**

A1	Login Dialog mit Passwort für den autorisierten Zugang der Mitarbeiter (Datenänderungen).
A2	In der Datenbank müssen die Informationen des Serviceauftrags und die Login Daten der Mitarbeiter verwaltet sein.
A3	Erfasste Serviceaufträge abrufbar sein.
A4	Die erfassten Serviceaufträge müssen selektiv nach Priorität abrufbar sein.
A5	Mitarbeiter können eine Statusänderung eines Auftrages vornehmen.
A6	Mitarbeiter können Aufträge löschen (z.B. bei Stornierungen)
A7	Die aufgerufenen API-Endpoints müssen zwecks Fehlerlokalisierung protokolliert sein (DB oder Protokolldatei).
A8	Datenbankstruktur muss normalisiert in der 3.NF sein inkl. referenzieller Integrität
A9	Für die Web-API Applikation muss ein eigener Datenbankbenutzerzugang mit eingeschränkter Berechtigung (DML) zur Verfügung gestellt werden (Benutzer root bzw. sa ist verboten).
A10	Das Web-API muss vollständig nach Open-API (Swagger) dokumentiert sein.
A11	Das Softwareprojekt ist über ein Git-Repository zu verwalten.
A12	Ganzes Projektmanagement muss nach IPERKA dokumentiert sein

Das sind Die Vorgegeben Anforderungen die an unsere Webseite gerichtet sind. Jetzt werde ich diese Ziele Noch in einer weiteren Tabelle nach Ihrer Priorität auswerten so das wir ein möglichst effizientes Vorgehens Modell haben.

<b>NR. Auswertung nach Priorität.</b>	
A1	Hoch
A2	Hoch
A3	Mittel
A4	Hoch
A5	Hoch
Ac	Mittel
A7	Hoch
A8	Mittel
AS	Hoch
A10	Hoch
A11	Mittel
A12	Hoch

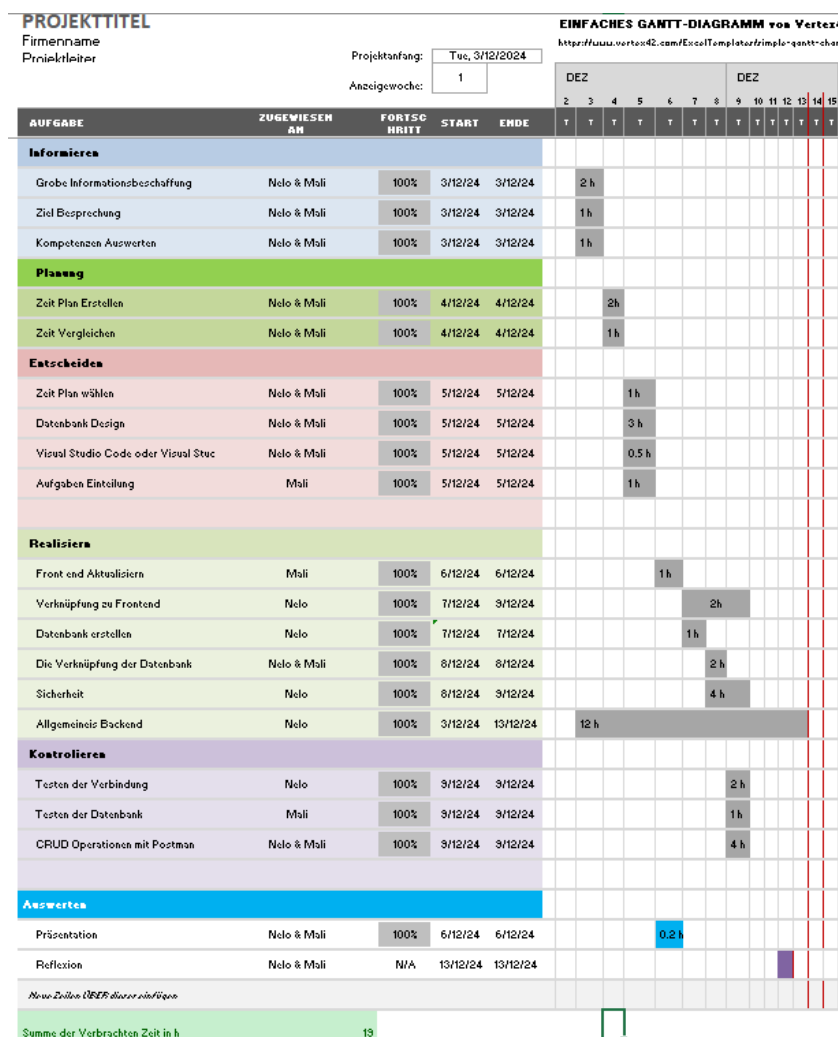
## Planen

In der Planung haben wir vor allem punkte gemacht wie die Zeitplanung, Aufgaben Aufteilung und erste Blaupausen unserer API-Dokumentation . Das ist für uns essentiell da es uns eine Grundlage gibt für die effiziente Erarbeitung unseres Projekts.

### Projektstruktur:

Hier haben wir einen Zeitplan Entworfen und unsere Aufgaben Einteilung erstellt das wir so schnell wie möglich Arbeiten und beide Ihre stärken aufbringen und anwenden können.

Hier ist ein Bild unsers Zeitplanes den wir erstellt haben und nach dem wir vorgegangen sind:



Das ist unser Grober Zeitplan den wir Entwickelt haben. Wie man unschwer erkennen kann ist er sehr grob da wir unsere Arbeit eigentlich beide auf verschiedenste zeiten eingeteilt haben und die aufgaben auch sehr Individuell erarbeitet haben.

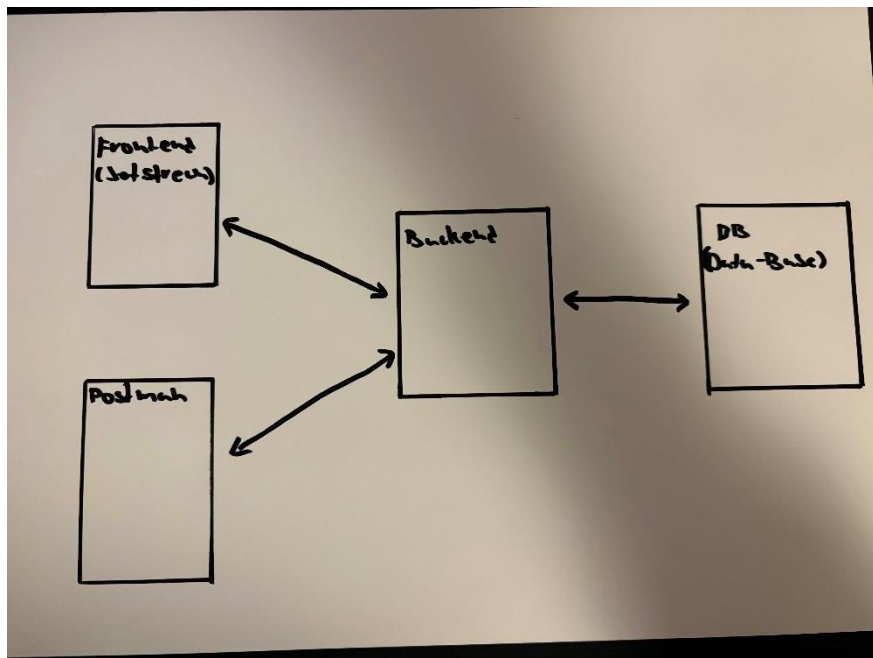


Und Auch haben wir uns hier noch eingeteilt wer welche aufgaben übernimmt so das wir am einfachsten arbeiten konnten auch wen wir gerade nicht mit einander sprechen konnten. So waren wir nicht nur extrem effizient sondern auch extrem Flexibel. Diese Aufgaben Einteilung sah wie Folgend aus:

Aufgabe	Aufteilung
<b>Backend</b>	<b>Nelo</b>
<b>Kommunikation mit dem Server</b>	<b>Nelo</b>
Datenbank Design	<b>Nelo/ Mali</b>
Backend CRUD Operationen	<b>Nelo</b>
<b>Dokumentation</b>	<b>Nelo/ Mali</b>
<b>Präsentation</b>	<b>Nelo/ Mali</b>
<b>Testing</b>	<b>Nelo/ Mali</b>
<b>Optimierung des Forntends</b>	<b>Mali</b>

## Skizzen

Hier Haben wir uns erstmalig Gedanken darüber gemacht wie wir unsere API End Points machen, wie wir sie Verbinden und wie wir das ganze anschließen können sowohl an das Backend aber auch an das Frontend



Dies ist die Skizze die wir erstellt haben für unsere API Kommunikation.

## Entscheiden

Hier sind wir auf die Punkte eingegangen Warum wir wider Bootstrap Genommen haben für die weitem Elemente der Webseite, Serio Logger, Code First (Für Datenbank) und vieles weitere

### Warum Bootstrap?

Da Bootstrap Schon Komplette fertige Elemente geliefert hat die wir nur kopieren mussten und den JavaScript code anpassen mussten und noch ein paar Sachen im HTML umbenenne und dann lief das ganze auch schon. Vor allem für die Admin Page war das extrem wichtig

### Warum Logger?

Da das Essentielle ist für unsere Arbeit mit dem Backend war ohne das konnten wir Nicht gut Debuggen und unsere logs ansehen und so Fehler Meldungen berücksichtigen.

## Technologieentscheidung

Hier haben wir festgelegt welche Tools und welche Applikationen wir verwendet haben.

Das waren die einfachsten und schnellsten Entscheidungen da wir diese schon beherrscht haben.

Diese Technologischen Entscheidungen waren folgende:

- Arbeit Umgebung: Visual Studio Code da wir extrem viel schon damit gemacht haben und uns dementsprechend sehr gut damit auskannten.
- Testing Programm: Postman weil wir damit schon gearbeitet habe und es mit abstand die umfassendste Umgebung war um unsere Überprüfungen zu machen.
- C#: Die einfachste und beste Sprache für so ein Backend meiner Meinung nach.
- Bootstrap: Für uns am besten passendes Framework.
- JavaScript: mussten wir für den Backend verwenden und haben wir in dem Modul angeschaut.

## Realisieren

Hier werden wir auf den Teil der Realisation eingehen aber nicht auf alles ganz so genau da das meiste wie HTML eigentlich selbsterklärend sind. Darum gehen wir aber genauer auf den teil der Server Verbindung ein und erklären den so gut wie möglich.

## Code Der Webseite

Den Code Der Neuen Elemente in der Webseite werde ich nicht erklären da sie selbsterklärend sind und wir sie schon im Modul Zuvor behandelt haben.

## Hilfs Mittel bei der Umsetzung

Wir haben Hilfsmittel verwendet auch wenn es Nicht ganz so viel waren, waren es doch schon einige. Wir haben viel auf <https://stackoverflow.com/> Nachgeschaut und auch auf weitem Portalen. Für die neuen Frontend Elemente haben wir einfach auf Bootstrap geschaut und die von da genommen.

Bei Problemen haben wir einfach gegoogelt und so meist eine sehr gute und schnelle Lösung bekommen. Natürlich haben wir aber auch ein paar mal ChatGPT verwendet für eine Hilfestellung die wir sonst nicht so schnell kapiert hätten. Da es auch gelogen wäre hätten wir gesagt das wir kein ChatGPT verwendet haben ist es klar das wir es angeben

## Verbindung mit JavaScript:

Da wir die Verbindung schon in der alten Dokumentation vom Modul 294 und auch in der Präsentation erklärt haben werde ich den Code Nicht komplett erklären sondern Nur einzelne schritte. Der Code ist auch mit Kommentaren erklärt im Programm und ist relativ simple Und Die Logik auch gleich ist wie die bei dem Modul294. Nur er Code Der „Adminpage“ ist ein Bisschen Komplexer aber da das eher so ein Zusatz war erkläre ich den auch nicht genauer, er ist ebenfalls im Code Mit Kommentaren erklärt.

Wichtiger Teil Kurz erklärt:

```
try {  
  const response = await fetch("http://localhost:5013/api/Users/Login", {  
    method: "POST",  
    headers: { "Content-Type": "application/json" },  
    body: JSON.stringify(data)  
  });
```

Hier sieht man die fetch Funktion für Login.

```
const response = await fetch(http://localhost:5013/api/Users/Login
```

Diese Zeile ist Speziell Wichtig da wir sie auch beim Register Verwenden und da einfach nur anstatt `/api/Users/Login`

Das in `/api/Users/Register` Register Um ändern müssen.

Das Wollte ich noch hier erwähnen da es relativ spannend ist und auch wichtig.

## Backend Code:

Wir werden ihnen Nicht den kompletten Code Erklären das wäre viel zu viel. Da der Code auch schon bei der Präsentation ausführlich erklärt wurde und der Code auch Kommentiert ist er Relativ selbst erklärend. Dennoch werde ich auch wichtige Punkte eingehen und diese Kurz erklären.

## Database und Migration:

Da ich hier nicht alle Datenbanken erklären kann werde ich einfach eine nehmen und mit der zeigen wie es in etwa abläuft.

Hier ist ein Beispiel der User Datenbank in C# Code:

```
namespace Modul295PraxisArbeit.Models
{
    public class User
    {
        public int UserId { get; set; }
        public string? Username { get; set; }
        public string? PasswordHash { get; set; }
        public string? Role { get; set; }
    }
}
```

Kurze Erklärung des Codes:

Das Modell User definiert eine einfache, aber wesentliche Datenstruktur für die Verwaltung von Benutzerinformationen in einer Anwendung. Es bietet Platz für:

1. Eine eindeutige Benutzer-ID (UserId).
2. Einen Benutzernamen (Username).
3. Einen Passwort-Hash (PasswordHash) zur sicheren Speicherung von Passwörtern.
4. Eine Rolle (Role), um Benutzerrechte zu definieren.

Das ist ein Schema und so sieht das dann in der Regel aus und Mann kann sich das Oft so Vorstellen wie hier nur das es wen es Grössere Datenbanken sind Natürlich deutlich Mehr Code ist.

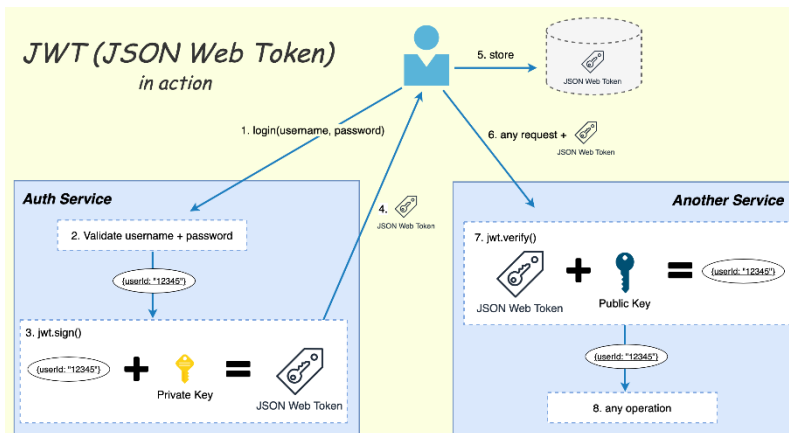
Um diese dann zu erstellen brauchten wir eine Migration diese sind 2 befehle die so Aussehen:

```
dotnet ef migrations add InitialCreate
dotnet ef database update
```

Diese Befehle erstellen eine Migration der Datenbanken die Migration heisst dann InitialCreate.

## Sicherheit (JWT)

Wir haben Sicherheit gewährleistet mit den JWT (JSON Web Token) die wir benutzt haben das Daten wie Passwörter und Username Verschlüsselt Kommuniziert werden und so Unleserlich gemacht werden. Ich werde ihnen Kurz eine Grafik einblenden die das ganze vereinfacht.



Wie man Hier auf der Grafik erkennt sieht man wie die Kommunikation mit dem JWT Funktioniert und warum es so sicher ist.

Auch Noch Wichtig ist zu erwähnen das wir die Passwörter von den Leuten Verschlüsselt haben so das sie unleserlich Gemacht werden so Sieht ein Verschlüsseltes Passwort aus:

`$2a$11$4UaEsOMkTpb3W54IRdu9..Y7LOfA.C9FdwPz0V.5tWWbiQOBgAx6G`

Wir haben das mit **BCrypt** gemacht da es am einfachsten war zu Implementieren Natürlich gab es Noch Sichere Varianten die sind aber zu Komplex und würden den Ramen Sprengen

## Controller im Backend:

In den Controller Spreche ich die ganzen CRUD Operationen an für die User und die ServiceOrders Dort sind die Methoden drin wo ich zB. Post deklariere und die andern Operationen.

Ich habe diese Funktionen in meiner Präsentation ausführlich erklärt und werde Darum hier in der Dokumentation nicht weiter drauf eingehen da es am einfachsten ist wen man den Code Dazu Anschaut.

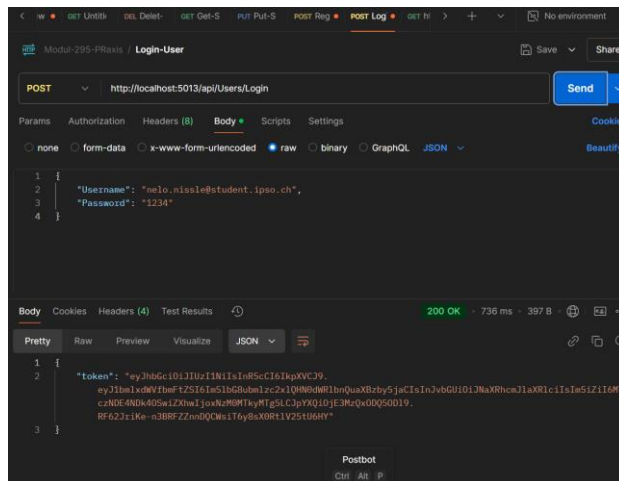
## Kontrollieren

Bei dem Kapitel Kontrollieren sind wir vor allem auf unsere Kontrollen mit Postman eingegangen da wir unsere Testes so gut wie nur über Postmans gemacht haben und mit der «REST» Extension in Visual Studio Code gemacht haben. Mehr hatten wir Bei den Tests auch gar nicht gemacht da diese Tests Alle Erfolgreich waren und wir dem entsprechen auch kein Test Diagramm erstellen mussten da es Direkt Funktioniert hat.

### Testen mit Postman:

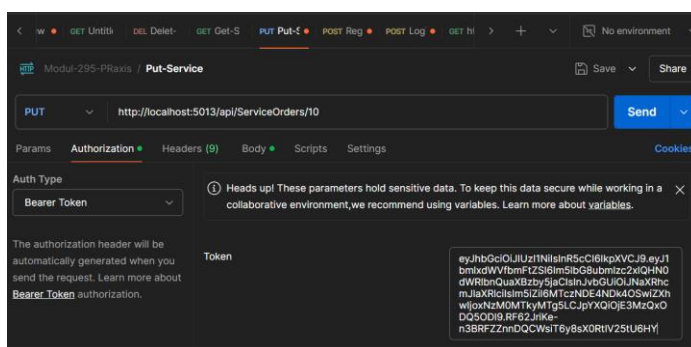
Die Tests in Postman haben wir Durchgeführt mit dem wir die URL geprüft haben. Ich werde Hier Deshalb einen Ausführlichen Test zeigen in dem ich Schritt für schritt erkläre was ich mache.

#### Schritt 1:



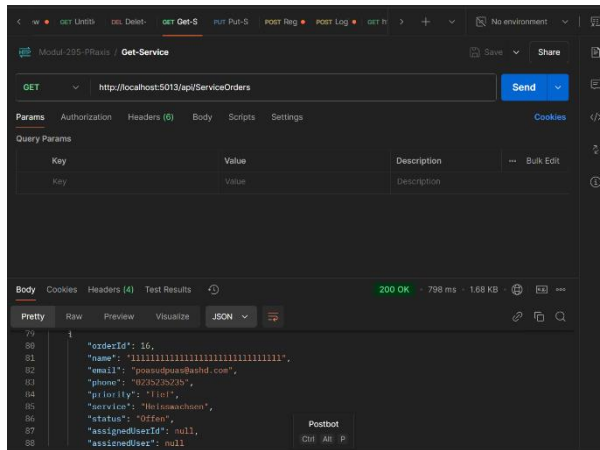
Hier Logge ich mich in ein Bestehendes Admin Konto ein und Kriege dann einen JWT den ich brauche um die Service Orders zu Bearbeiten.

#### Schritt 2:



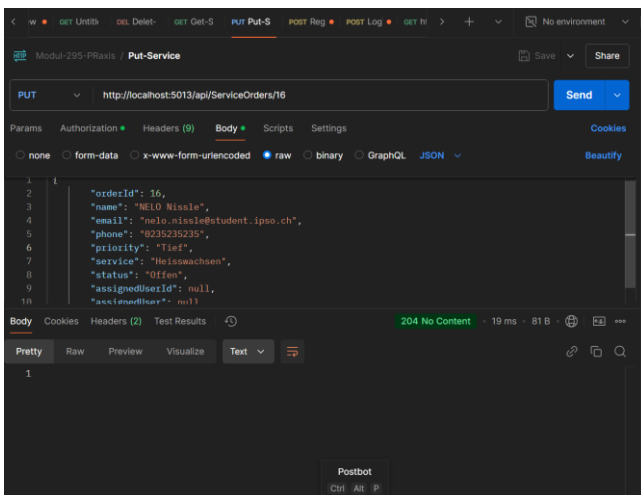
Ich Kopiere den JWT den ich beim Login erhalten habe und Füge ihn in der Collection als Bearer Token so Funktioniert er Für Alle CRUD Operationen.

## Schritt 3:



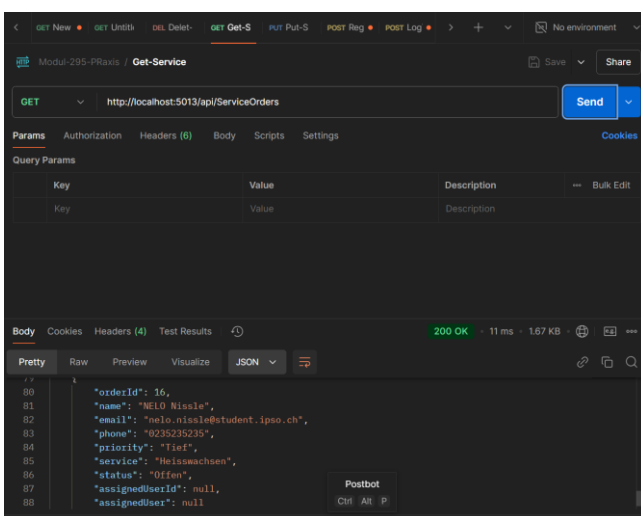
Ich sende eine Get anfrage an ServiceOrders um die Gewünschte anfrage zu suchen die ich bearbeiten will.

## Schritt 4:



Wie man sieht steht 204 No Content das heisst es lief gut. Ich habe hier meinen Namen geändert und die email. Um das zu überprüfen mache ich erneut eine Get anfrage und kann das mit dem ersten Bild ( der ersten Get anfrage überprüfen ob es noch gleich ist)

## Schritt 5:



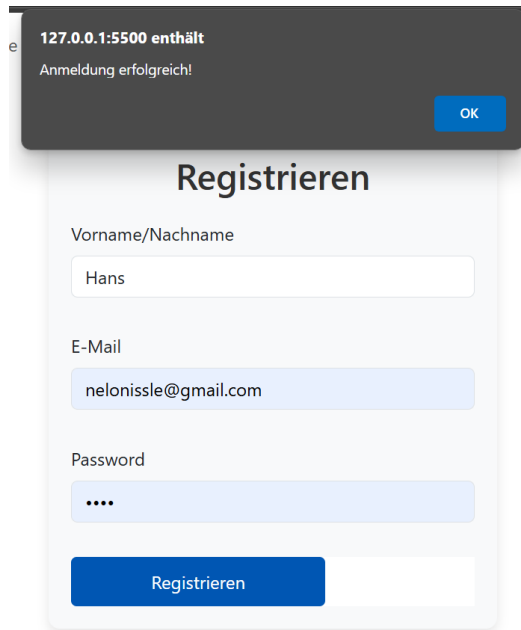
Wie man sieht hat sich der Name verändert und somit kann ich sagen das der Test positiv gelaufen ist. Das gleiche habe ich auch so mit den anderen Funktionen getestet.



## Frontend Test:

Der Frontend Test habe ich sehr ähnlich gemacht wie der Postman Test Ich zeige es mit dem Register und wider in schritten.

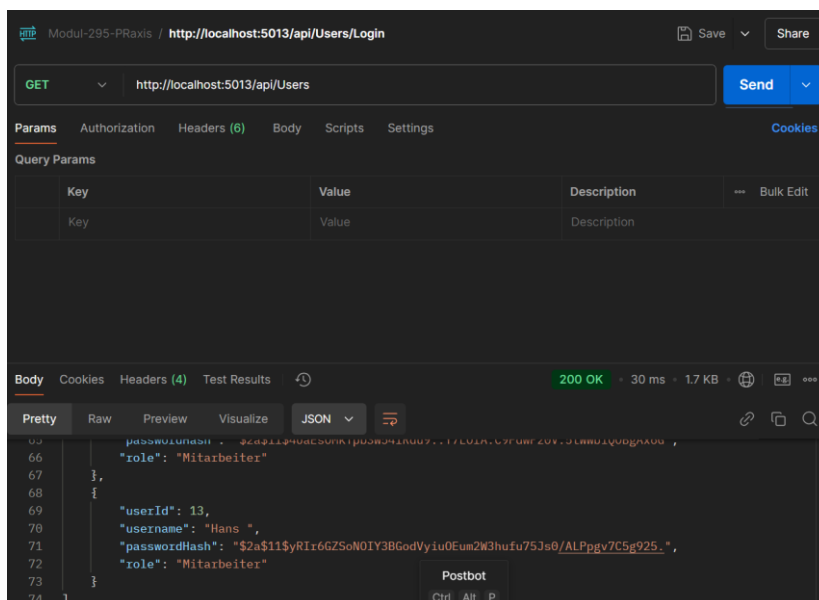
### Schritt 1:



The screenshot shows a web application interface. At the top, a dark grey notification box displays the text "127.0.0.1:5500 enthält" and "Anmeldung erfolgreich!" with an "OK" button. Below this is a form titled "Registrieren". The form contains three input fields: "Vorname/Nachname" with the value "Hans", "E-Mail" with the value "nelonissle@gmail.com", and "Password" with masked characters "....". A blue "Registrieren" button is at the bottom of the form.

In dem Schritt registriere ich mich erstmals mit Name, email und Passwort. Und oben in der Meldung steht schon das es erfolgreich war dennoch heißt das noch gar nix und wir überprüfen ob es den Benutzer Erstellt hat.

### Schritt 2:



Wie man sehen kann hat es mich erfolgreich registriert und ich kann sehen das es den neuen Benutzer gibt und hier ist auch ein perfektes Beispiel wie die Verschlüsselung des Passworts aussieht.

Das Passwort war 1234 und Jetzt sieht es so aus.

Also der Test war Erfolgreich Natürlich habe ich viel mehr gemacht die alle auch Positiv ausgefallen sind.

### Vollständigkeit liste:

Auch haben wir noch geschaut ob wir dann alles erfüllt haben mit einer kleinen Checkliste der Ziele die auf GitHub waren das sah dann in etwa so aus:

NR	CHECK
A1	Vollständig
A2	Vollständig
A3	Vollständig
A4	Vollständig
A5	Vollständig
A6	Vollständig
A7	Vollständig
A8	Vollständig
AG	Vollständig
A10	Vollständig
A11	Vollständig
A12	Vollständig

Anhand dieser Grafik Konntet wir unsere abriet mit Gutem gewissen abgeben und so auch sehr beruhigt sein. Zu dieser liste haben wir aber auch noch zusammen die Bewertung nageschaut und uns damit Gedanken gemacht und mit Hilfe dieser Tabelle auch zu frieden sagen das wir für uns das Projekt abgeschlossen haben.

## Auswerten

Bei der Auswertung ging es Darum was wir gelehnt haben was wir verbessern könnten in Zukunft und noch ein Schluss Fazit. Auch kommt noch kurz ein Teil wie man das ganze startet und Benutzt.

### Anleitung für admin page:

Hier ist eine Kleine Anleitung wie ich den in die AdminPage komme/ wie ich meinen Token bekomme den ich dafür benötige.

#### Wichtig!!

Diese Seite bin ich erst am entwickeln und soll Noch nicht Funktionieren da sie noch nicht auf Berechtigungen angepasst ist!

#### Schritt 1:

Man muss sich mit einem Mitarbeiter Konto einloggen und Durch das bekomme ich einen Token den ich Benutzen kann um Zugang zu bekommen.

Hier ist ein Passwort und einen Benutzernamen für ein Mitarbeiter Konto das sie auch die Admin Tools Ausprobieren können.

```
{
  "Username": "nelo.nissle@student.ipso.ch",
  "Password": "1234"
}
```

Dies ist ein Login für ein Richtigen JWT Token.

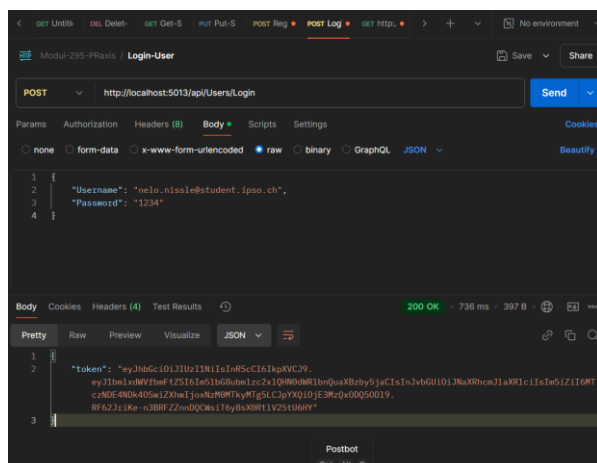
#### Schritt 2:

IM Postman diese URL Eingebn mit der Methode POST

URL: <http://localhost:5013/api/Users/Login>

Geben sie diese JSON Daten von oben im Body Feld ein und Sichern sie das ganze.

#### Schritt 3:



So Sollte das Aussehen, dann Kopieren sie den Token den Sie Unten sehen.

## Schritt 4:

## Service Orders

Enter Your Token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bmRxdWVmbmFtZSI6Im5lbG8ubmlzc2xlQHN0dWRlbnQuaXBzby5jaCIsInJvbmGUOiJNaXRhcmJlaXRlcisIm5iZiI6MTc

Kopieren sie den Token in diese Feld bei der Admin Page.

## Schritt 5:

## Service Orders

Enter Your Token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bmRxdWVmbmFtZSI6Im5lbG8ubmlzc2xlQHN0dWRlbnQuaXBzby5jaCIsInJvbmGUOiJNaXRhcmJlaXRlcisIm5iZiI6MTc

Validate Token

Token is valid.

Order ID	Name	Email	Phone	Priority	Service	Assigned User ID	Status	
6	Nelo	Mali@asdasd.com	109387410395	Tief	Kleiner Service	Not Assigned	InArbeit	<a href="#">Edit</a> <a href="#">Delete</a>
9	asdaskhdashd	Mali@asdasd.com	109387410395	Standard	Grosser Service	Not Assigned	Not Set	<a href="#">Edit</a> <a href="#">Delete</a>
10	kkkkkkkkkkkk	KKKKKKKKKKKKKK@asdasd.com	109387410395	Standard	Grosser Service	Not Assigned	Not Set	<a href="#">Edit</a> <a href="#">Delete</a>
11	ashdpiahdi	poasudpuas@ashd.com	235235235	Tief	Grosser Service	Not Assigned	Not Set	<a href="#">Edit</a> <a href="#">Delete</a>
13	kkkkkkkkkkkk	poasudpuas@ashd.com	0235235235	Tief	Grosser Service	Not Assigned	Not Set	<a href="#">Edit</a> <a href="#">Delete</a>

Ist der Token Valid und Aktuell dann haben sie die Berechtigung diese Daten zu Bearbeiten.

Das war die Anleitung die sollte ihnen alle schritte gut erklären und zeigen.  
So Können sie die Adminpage Benutzen und kommen so da rein.

## Swagger Dokumentation:

Wenn sie das backend starten mit **dotnet run** können sie einfach auf die swagger Dokumentation zugreifen mit der Richtigen URL

URL: <http://localhost:5013/swagger/index.html>

So Können sie alle meine Funktionen sehen

## Lessons Learned:

Wir haben uns für diesen Punkt zusammengesetzt und wir haben beide Unterschiedliches gelehrt. Deshalb haben wir uns beide kurz ran gesetzt und entsprechend aufgeschrieben was wir gelehrt haben.

Nelo:

Nicht nur hatte ich extrem viel Spaß und Motivation für das Projekt sondern habe ich auch extrem viel neues gelehrt. Ich habe Extrem Viel gelehrt gerade was das Programmieren des Backend angeht wie die API's Funktionieren wie ich CRUD Operationen Implementiere usw. Also Extrem viel neues und extrem spannend. In dem Modul habe ich auch entdeckt das ich mich mehr für die Backend Entwicklung begeistere als für die Frontend Entwicklung.

Ich konnte mich richtig entfalten bei dem Projekt und habe es alles sehr gut verstanden.

Mali:

Im Rahmen dieser Arbeit wurde erfolgreich ein Backend realisiert, das über APIs und eine Datenbankbindung verfügt. Die Umsetzung umfasste die Entwicklung von Funktionalitäten wie Benutzer-Authentifizierung, Datenverwaltung sowie die Bereitstellung von Schnittstellen zur Interaktion mit einer Website. Durch den Einsatz moderner Technologien und Best Practices konnte ein stabiles, sicheres und skalierbares Backend geschaffen werden, das den Anforderungen des Moduls gerecht wird.

Diese Erfahrung hat nicht nur technisches Wissen in der Backend-Entwicklung erweitert, sondern auch den praktischen Umgang mit Datenbanken und API-Design vertieft. Die Arbeit bildet eine solide Grundlage für weiterführende Projekte und zeigt, wie wichtig strukturierte Planung und effiziente Implementierung in der Softwareentwicklung sind.

## Verbesserungen in der Zukunft

Wir sind beide zum Entschluss gekommen das wir in der Zukunft sicherlich besser eine so grosse Arbeit zu richtig zu planen und nicht nur «Freestyle zu arbeiten» Das war so der Hauptpunkt sonst fanden wir war alles ganz gut was unsere Arbeit anging Vor allem haben wir uns unter einander extrem gut verstanden und konnten uns gegenseitig sehr gut ergänzen gerade mit unseren verschiedenen stärk

### Fazit:

Das Projekt hat uns beiden sehr gefallen und wir konnten uns extrem gut ergänzen. Gerade in der Zusammenarbeit hatten wir richtig Spaß und mussten uns nicht irgendwie durchkämpfen sondern konnten immer zusammen etwas fragen und immer gut zusammensetzen. Auch konnten wir unsere Stärken im Team sehr gut ausbauen und so den anderen ergänzen und auch korrigieren. Die Kreativität wie wir an das Projekt angegangen sind war perfekt denn wir haben wenn wir ein Problem hatten immer den Partner der einen anderen Lösungsansatz hatte oder wusste wie man das Problem beheben konnte.

Wir haben dem entsprechend viel gelehrt gerade auch was das zusammenarbeiten angeht gerade wenn es zur Kommunikation kommt mit GitHub dem pushen/ Pullen wie man mit dem umgeht aber auch der Verständigung im Team. Das war für uns beide etwas sehr spannendes was wir so in Zukunft sicher auch noch sagen können das es uns sehr geholfen hat.

## Referenzen s Quellen

Als Referenzen und auch Quellen Material haben wir am meisten das GitHub von Modul295 von Lukas Müller verwendet. Die Genauen quellen die wir verwendet haben sind Bei Quellen in einer Tabelle aufgelistet.

### Quellen:

Das waren so unsere Haupt Quellen die wir verwendet haben.

Name	Link
<b>Bootstrap</b>	<a href="#">Bootstrap · The most popular HTML, CSS, and JS library in the world.</a>
<b>Youtube</b>	<a href="#">(221) YouTube</a>
<b>Swagger</b>	<a href="#">Swagger Editor</a>
<b>ChatGPT</b>	<a href="#">ChatGPT</a>