Lernatelier: Projektdokumentation

Vontobel

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Version** | **Arbeit** | **Autor** |
| 16.12.2022 | 0.1.0 | Informieren, Planen | Vontobel |
| 23.12.2022 | 0.2.0 | Informieren, Planen, Entscheiden | Vontobel |
| 13.01.2023 | 0.3.0 | Realisieren | Vontobel |
| 20.01.2023 | 0.4.0 | Realisieren | Vontobel |
| 27.01.2023 | 1.0.0 | Kontrollieren, Auswerten | Vontobel |

# Informieren

## Ihr Projekt

Ich erstelle ein Backend für eine CRUD-Applikation, in der ich eine Funktion für ein Login erstelle, für das Löschen von Daten, für das Erstellen von Daten, für das Verändern von Daten und das Lesen von Daten. Die Daten sind hierbei Schuhe von der Marke Nike

## Quellen

https://www.youtube.com/watch?v=mbsmsi7l3r4

<https://www.youtube.com/watch?v=7Q17ubqLfaM>

<https://www.youtube.com/watch?v=Ud5xKCYQTjM>

<https://www.youtube.com/watch?v=VShtPwEkDD0>

<https://www.youtube.com/watch?v=SccSCuHhOw0>

## Anforderungen

|  |  |  |  |
| --- | --- | --- | --- |
| **Nummer** | **Muss / Kann?** | **Funktional? Qualität? Rand?** | **Beschreibung** |
| 1 | Muss | Funktional | Backend kann zu einer GET request alle Daten zurücksenden |
| 2 | Muss | Funktional | Backend kann zu einer GET request bestimmte Daten zurücksenden |
| 3 | Muss | Funktional | Zu allen Daten wird zusätzlich noch eine ID erstellt, mit der man sie identifizieren kann |
| 4 | Muss | Funktional | Backend kann zu einer PUT request bestimmte Daten ändern, wenn ein Nutzer eingeloggt ist |
| 5 | Muss | Funktional | Backend kann zu einer POST request bestimmte Daten erstellen, wenn ein Nutzer eingeloggt ist |
| 6 | Muss | Funktional | Backend kann zu einer DELETE request bestimmte Daten löschen, wenn ein Nutzer eingeloggt ist |
| 7 | Muss | Funktional | Nutzer kann sich einloggen |
| 8 | Kann | Qualität | Nutzer kann sich registrieren |
| 9 | Kann | Rand | Frontend zum Backend, welches die Funktionen benutzt |

## 1.4 Diagramme

## 1.5 Testfälle

|  |  |  |  |
| --- | --- | --- | --- |
| **Nummer** | **Voraussetzung** | **Eingabe** | **Erwartete Ausgabe** |
| 1.1 | GET Funktion ist erstellt | GET http://localhost:5000/allShoes  Mit Hilfe des REST Client wird die request gemacht | Alle Daten werden ausgegeben |
| 2.1 | GET Funktion ist erstellt und Daten besitzen jeweils eine ID | GET http://localhost:5000/shoe/477472a6-722f-4845-831f-b2b4056943a5 | Die Daten zu der angefragten ID werden ausgegeben |
| 3.1 | POST Funktion um Daten zu erstellen funktioniert | POST http://localhost:5000/addShoe  Und Bearer Token und Daten zum Posten | Zusätzlich zu den geposteten Daten sollte noch eine ID erstellt werden |
| 4.1 | PUT Funktion um Daten zu updaten ist erstellt | PUT http://localhost:5000/changeShoe/477472a6-722f-4845-831f-b2b4056943a5  Und Bearer Token und Daten zum Updaten | Daten werden übernommen und gespeichert |
| 5.1 | POST Funktion um Daten zu erstellen ist erstellt | POST http://localhost:5000/addShoe  Und Bearer Token und Daten zum Posten | Daten werden erzeugt und |
| 6.1 | DELETE Funktion um Daten zu löschen ist erstellt | DELETE http://localhost:5000/deleteShoe/477472a6-722f-4845-831f-b2b4056943a5  Und Bearer Token | Daten zu der gegebenen ID werden gelöscht |
| 7.1 | POST Funktion um sich einzuloggen ist erstellt und ein Login existiert | POST http://localhost:5000/login  Und Logindaten | Nutzer wird eingeloggt und erhält einen Token |
| 8.1 | POST Funktion um sich zu registrieren ist erstellt | POST http://localhost:5000/register  Und gewünschte Logindaten | Nutzerkonto wird erstellt und kann benutzt werden, um sich einzuloggen |
| 9.1 | Alle Funktionen des Backend sind erstellt und funktioniert | Frontend Applikation mit Fetch Funktionen | Daten können benutzt werden im Frontend |

1. Planen

|  |  |  |  |
| --- | --- | --- | --- |
| **Nummer** | **Frist** | **Beschreibung** | **Zeit (geplant)** |
| 1.1 | 13.01.2023 | Es soll ein GET request erstellt werden, womit jeder Schuh aus der Liste zurückgesendet wird. | 30min |
| 2.1 | 13.01.2023 | Es soll ein GET request erstellt werden, womit ein bestimmter Schuh zurückgesendet wird. (Schuh soll über ID gefunden werden) | 30min |
| 3.1 | 13.01.2023 | Wenn ein Schuh erstellt wird, sollte dieser zusätzlich noch eine ID bekommen, womit er identifiziert werden kann. | 15min |
| 4.1 | 13.01.2023 | Mit einem PUT request sollte es möglich sein Daten von Schuhen zu verändern. | 60min |
| 5.1 | 13.01.2023 | Mit einem POST request sollte es möglich sein einen neuen Schuh zu erstellen. | 60min |
| 6.1 | 13.01.2023 | Jeder Schuh sollte der Nutzer löschen können per DELETE request. | 45min |
| 7.1 | 20.01.2023 | Jeder Nutzer soll die Möglichkeit haben sich einzuloggen und sollte dadurch einen Token bekommen. | 120min |
| 8.1 | 20.01.2023 | Jeder Nutzer soll die Möglichkeit haben sich zu registrieren mit einem Nutzernamen und Passwort. | 90min |
| 9.1 | 20.01.2023 | Zu dem erstellten Backend soll eine Frontend Applikation erstellt werden, in der man alle Funktionen der obigen Anforderungen nutzen kann. | übrige Zeit |

1. Entscheiden
2. Realisieren

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nummer** | **Frist** | **Beschreibung** | **Zeit (geplant)** | **Zeit (effektiv)** |
| 1.1 | 13.01.2023 | Es soll ein GET request erstellt werden, womit jeder Schuh aus der Liste zurückgesendet wird. | 30min | 30min |
| 2.1 | 13.01.2023 | Es soll ein GET request erstellt werden, womit ein bestimmter Schuh zurückgesendet wird. (Schuh soll über ID gefunden werden) | 30min | 30min |
| 3.1 | 20.01.2023 | Wenn ein Schuh erstellt wird, sollte dieser zusätzlich noch eine ID bekommen, womit er identifiziert werden kann. | 15min | 15min |
| 4.1 | 20.01.2023 | Mit einem PUT request sollte es möglich sein Daten von Schuhen zu verändern. | 60min | 80min |
| 5.1 | 20.01.2023 | Mit einem POST request sollte es möglich sein einen neuen Schuh zu erstellen. | 60min | 80min |
| 6.1 | 20.01.2023 | Jeder Schuh sollte der Nutzer löschen können per DELETE request. | 45min | 60min |
| 7.1 | 13.01.2023 | Jeder Nutzer soll die Möglichkeit haben sich einzuloggen und sollte dadurch einen Token bekommen. | 120min | 180min |
| 8.1 | 20.01.2023 | Jeder Nutzer soll die Möglichkeit haben sich zu registrieren mit einem Nutzernamen und Passwort. | 90min | 60min |
| 9.1 | 20.01.2023 | Zu dem erstellten Backend soll eine Frontend Applikation erstellt werden, in der man alle Funktionen der obigen Anforderungen nutzen kann. | übrige Zeit | 180min  (180min wurden nicht für das reine Frontend gebraucht, sondern auch noch für Änderungen am Backend) |

1. Kontrollieren

## Testprotokoll

|  |  |  |  |
| --- | --- | --- | --- |
| **Nummer** | **Datum** | **Resultat** | **Durchgeführt** |
| 1.1 | 27.01.2023 | **OK** | Vontobel |
| 2.1 | 27.01.2023 | **OK** | Vontobel |
| 3.1 | 27.01.2023 | **OK (im Frontend konnte die Funktion "authenticateToken" nicht verwendet werden.)** | Vontobel |
| 4.1 | 27.01.2023 | **OK (im Frontend konnte die Funktion "authenticateToken" nicht verwendet werden.)** | Vontobel |
| 5.1 | 27.01.2023 | **OK (im Frontend konnte die Funktion "authenticateToken" nicht verwendet werden.)** | Vontobel |
| 6.1 | 27.01.2023 | **OK (im Frontend konnte die Funktion "authenticateToken" nicht verwendet werden.)** | Vontobel |
| 7.1 | 27.01.2023 | **OK** | Vontobel |
| 8.1 | 27.01.2023 | **OK** | Vontobel |
| 9.1 | 27.01.2023 | **OK (im Frontend konnte die Funktion "authenticateToken" nicht verwendet werden.)** | Vontobel |

1. Auswerten

Fazit: Die Tests zuerst in Der IDE sind alle ohne Probleme gelungen, doch im Frontend gab es ein Problem mit der Funktion "authenticateToken", welche dafür sorgen sollte, dass Daten nur verändert, gelöscht oder hinzugefügt werden können, wenn ein Token vorhanden ist. Da dies nicht funktionierte habe ich eine andere Auswegs Lösung gefunden nämlich mit der "getToken" Funktion im Frontend die für das gleiche Sorgt. Die Applikation funktioniert jetzt trotzdem zu 100%.