**BlindSpot backend Dokumentation**

WICHTIG:

* **Nichts ist perfekt - bei Fehler soll man sich bei dem zuständigem Backend-Ingenieur beraten**
* **Beim Lesen des Dokuments bitte das Hirn einschalten (etwas was ihr nicht versteht sollt ihr nochmal lesen)**

FORMAT:

REQUEST TYPE: Endpoint-Path; (PARAMETERS...); RETURNS: RETURNVALUE

POST: ServerPath/users/register; (string userName, string password);   
RETURNS: Status Code 200 (OK) / Status Code 409 (User already exists / NOT OK)

Bespiel via Axios.js

axios.post(serverUrl + 'users/register', {userName: 'Adam', password: 'Test'}, (err, res) => {  
 // Callback Code  
})

Beispiel via C# (APICalls Klasse)

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung

POST: ServerPath/users/login; (string userName, string password);  
RETURNS: Status Code 200 + JWT Token / Status code 401 (negativ)

Axios bsp

axios.post(serverUrl + 'users/login', {userName: 'Adam', password: 'Test'}, {headers: {"Content-Type": "application/json"}})

JWT BENÖTIGT:

POST: ServerPath/containers/create; (STRING storageName, Array (Schema bei adam fragen) storageUnits);

RETURNS: 200 OK / 500 NEGATIV

const storageUnitsForTest = [  
 {  
 xCord: 0,  
 yCord: 0  
 },  
 {  
 xCord: 1,  
 yCord: 0  
 },  
 {  
 xCord: 2,  
 yCord: 0  
 }  
]

axios.post(serverUrl + 'users/login', {userName: 'Adam', password: 'Test'}, {headers: {"Content-Type": "application/json"}})  
.then(r => {  
 axios.post(serverUrl + 'containers/create', {  
 storageName: 'Schrank A',  
 storageUnits: storageUnitsForTest  
 }, {headers: {Authorization: `Bearer ${r.data.jwtToken}`}})  
 .then((r) => {  
 ***console***.log(r);  
 });  
   
});

PRODUCT CREATE

axios.post(serverUrl + 'products/create', {storageUnitId: 2, productName: 'Banane', lrAlignment: 1, fbAlignment: 0}, {headers: {Authorization: `Bearer ${r.data.jwtToken}`}})