Datenbank und Service-Provider

Inhalt:

1. Datenbank anlegen
2. http-Modul für Datenbank-Verbindung
3. Service-Provider erstellen

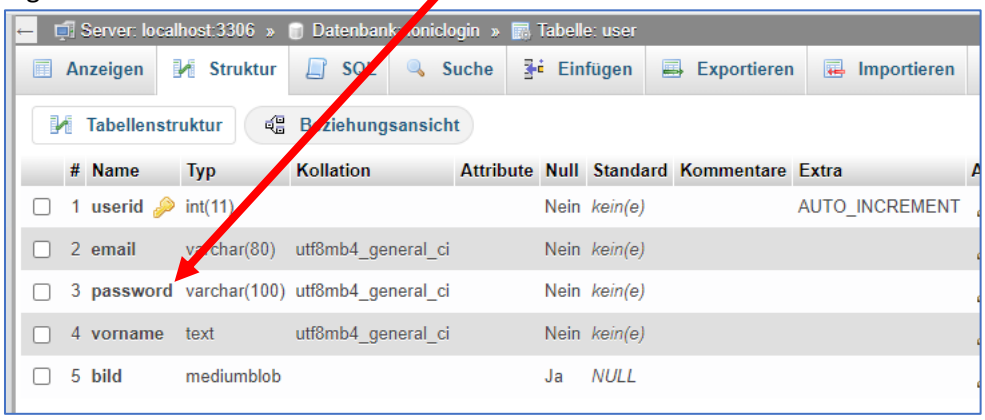
1)Datenbank anlegen

Starte Xampp, öffne „phpMyAdmin“ und lege eine neue Datenbank an mit dem Namen „ioniclogin“.

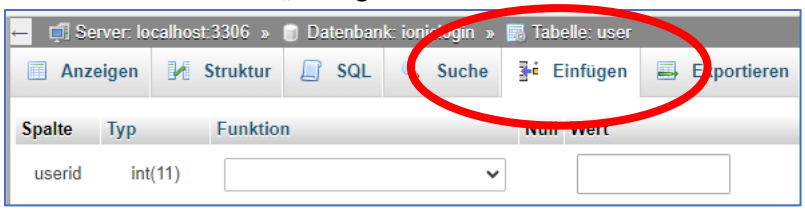
Besser ist es, das Element „Passwort“ in der Datenbank mit hartem „t“ zu schreiben, weil in der späteren API es zwar auch mit weichem „d“ funktioniert, aber die SQL Darstellung sinnvoller ist, wenn es „passwort“ heißt.

Hier ist das leider noch nicht so berücksichtigt.

Ergebnis:

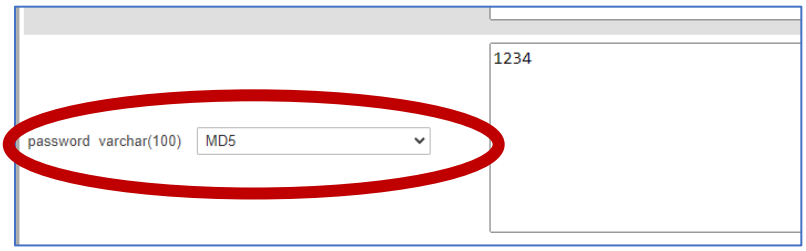


Erstelle zwei User:
Nutze dazu den Button „Einfügen“:



Beachte: Bei „password“ gib vorne im Auswahlfeld die Verschlüsselungsvariante „md5“ ein, weil in der Datenbank später auch das Login mit dieser Variante erstellt werden wird. Würde man hier keine übereinstimmende Variante einstellen, könnte das Passwort nicht ausgelesen werden – da beim

Login auch diese Auslesung mit „md5“ erfolgt.



Ergebnis:

	userid	email	password	vorname	bild
chen	1	alex@hak.at	81dc9bdb52d04dc20036dbd8313ed055	Alexander	[BLOB - 21,7 KiB]
chen	2	fabian@hak.at	81dc9bdb52d04dc20036dbd8313ed055	Fabian	*NULL*

Tipp: Vielleicht ist es am Beginn besser, das MD5 wegzulassen und das Passwort im Klartext zu schreiben, weil das spätere Umwandeln in der API mittels „password_verify()“ manchmal zickt.

2)http-Modul für Datenbank-Verbindung

Theorie:
Das HttpClient Modul benötigt man, um http **POST, GET, PUT und DELETE requests** machen zu können. Dadurch wird es möglich Daten zum API (Verbindung zur Datenbank) zu senden bzw. zu erhalten.

API calls, die das http-Client-Modul nutzen sind asychron. Dabei wird mit modernen JavaScript API-Elementen gearbeitet, nämlich

- Promises
- Observables

Promises
Ein „promise“ kann 3 Zustände haben, nämlich pending (warten), fulfilled und rejected.

Observables
ist der neuere Standard ab Ionic5 und kann mehr Features als promises.

Registriere HttpClientModule in AppModule

Öffne den Ordner in VisualStudioCode.

Öffne bzw. schreibe im geöffneten internen Terminal:

Es sollen 2 Features nachinstalliert werden.

- Um http requests machen zu können:

  npm install rxjs

  npm install –save rxjs-compat

Öffne die „app.module.ts“

Um mit diesem http-Modul arbeiten zu können muss man es importieren.

app.module.ts - erstelle den Import von HttpClientModule UND vom Storage:

import { HttpClientModule } from '@angular/common/http';

```
8   import { AppRoutingModule } from './app-routing.module';
9   import { HttpClientModule } from '@angular/common/http';
10
11  @NgModule({
```

Füge dazu auch in den „imports“ es nach einem Beistrich ein:

```
13    imports: [BrowserModule, IonicModule.forRoot(),
14      AppRoutingModule, HttpClientModule],
```

3)Service Provider erstellen

Um an einer zentralen guten Stelle die Ausgabe zu organisieren, sollte dies immer in einem vorhandenen Service durchgeführt werden.

Erstelle im integrierten Terminal einen Provider mit dem CMD-Befehl.
Es gibt dafür **2 Varianten:** entweder

1. lässt man IONIC für sich arbeiten

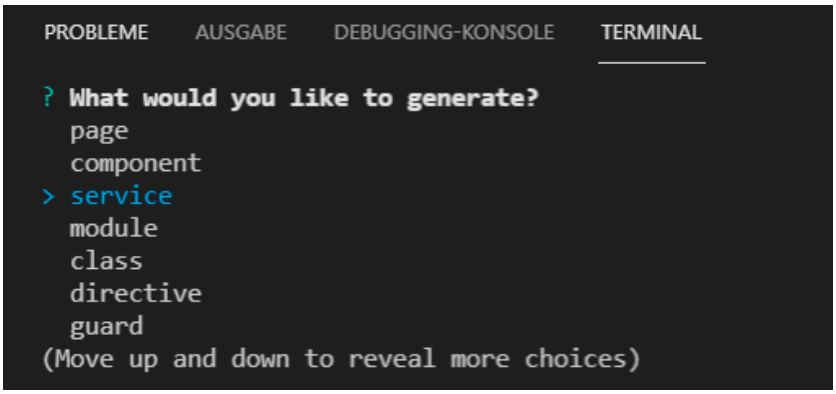
   gib im Terminal ein:

```
D:\25ionic\ersteapp> ionic g service api
```

2. ODER es werden die Auswahlmöglichkeiten angeboten:

   ionic g

Auswahl mit Pfeiltasten auf „service“ stellen



oder selbst eingeben:

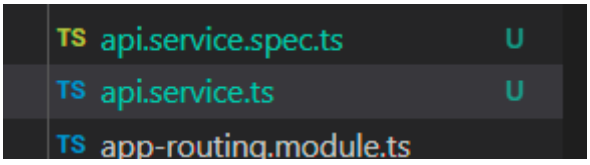
1. ionic generate service

```
PS D:\ionic_start\erstesApp> ionic generate service
```

Dann wird nach dem Namen gefragt. Gib ein

api

Dieser Name wird automatisch folgendermaßen umgesetzt und in die 2 neue Dateien übernommen:



- app.module.ts

Dieses Service muss man NICHT in app.modules.ts importieren!

Erstelle http Service mit RxJS Observables

Öffne „api.service.ts“.

Schreibe in die Zeile 2 und 3 folgenden IMPORT:

import {HttpClient, HttpHeaders} from '@angular/common/http';

import { map } from 'rxjs/operators';

```
src > app > TS api.service.ts > ...
  1   import { HttpClient } from '@angular/common/http';
  2   import { Injectable } from '@angular/core';
  3   import { map } from 'rxjs/operators';
  4
```

Dann in den Constructor:

```
 9    constructor(
10      public http: HttpClient
11    ) { }
```

Man könnte auch hier wieder zuerst die Methode im constructor schreiben und sich automatisiert den Import vorschlagen lassen und dann annehmen, um den Import in Zeile 2 nicht selbst schreiben zu müssen.