Ionic Workshop – Tutor Notes

Version für Ionic 8 / Angular 19 mit standalone components

Erkläre die Grundlagen von Ionic (Präsentation)
Zeige fertige App und erkläre, was wir machen

Projekt anlegen: *ionic start:*

- App Creation Wizard: **no**,
- Framework: **Angular**,
- Template **blank**,
- ngModules or **standalone**,
- create a free Ionic Account: **no**

ionic serve vorstellen und grundlegende Struktur einer Ionic-App zeigen.
Eine Kleinigkeit ändern – Automatic Reload zeigen

in src/app neuen Ordner pages anlegen und home-Verz. reinverschieben:
zeigen, dass die App jetzt nicht mehr läuft -> app-routing.module.ts anpassen (sollte automatisch passieren – aber speichern).

ionic g page pages/QuestionList (beachte erzeugtes File!!!!)
ionic g page pages/Question
ionic g page pages/Quiz
ionic g service services/data

zeigen/überprüfen, dass app-routing.module.ts angepasst wurde.
Ändere:

```
{ path: 'question/:id', loadChildren: './pages/question/question.module….},
```

URL manuell eingeben und zeigen, dass Seite aufgerufen wird.

Chrome-Inspect Mode und Handyansicht zeigen.

Business Logik: DataService und Interfaces

Lege Homepage mit dem ersten Button an – zeige, dass Seite im Browser neu geladen wird.

Erstelle services/Question.ts zeige export
Erstelle services /Quiz.ts zeige import
Lege in DataService Testdaten an:

services/Question.ts:

```
export interface Question {
    id: string;
    title: string;
    a1: string;
    a2: string;
    a3: string;
    a4: string;
    correct: number;
}
```

services/Quiz.ts:

```
import { Question } from './question';
export interface Quiz {
    id: string;
    quizName: string;
    questions: Question[];
}
```

Data.Service.ts

```
…
export class DataService {

  //initialize the currentQuiz property with a new quiz object
  public currentQuiz: Quiz = {id: '', quizName: 'newQuiz', questions: []};

  constructor() {
    //add a question to the currentQuiz property
    this.currentQuiz.questions.push({
      id: '1',
      title: 'What is the capital of France?',
      a1: 'New York',
      a2: 'London',
      a3: 'Paris',
      a4: 'Dublin',
      correct: 3
    });
   }
}
```

Homepage anlegen:

Erkläre

- MVC in Ionic/Angular
- Konzept eines Singleton und Umsetzung als Injectable in Angular
- Ionic-Komponenten in html, müssen in ts importiert werden – siehe gelbe Markierung (VS Code: Angular Essentials Plugin will do that for you automatically)

Home.page.ts:

```
import { Component, inject } from '@angular/core';
import { Router } from '@angular/router';
import { IonHeader, IonToolbar, IonTitle, IonContent, IonButton } from
'@ionic/angular/standalone';
import { DataService } from 'src/app/services/data.service';

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
  imports: [IonHeader, IonToolbar, IonTitle, IonContent, IonButton],
})
export class HomePage {
  public data = inject(DataService); //inject the DataService into the data
property - new alternative to using the constructor
  private router = inject(Router); //inject the Router into the router property -
new alternative to using the constructor

  constructor() {}

  startTest() { this.router.navigate(["/quiz"]); }
  showList() { this.router.navigate(["/question-list"]); }
}
```

Home.page.html:

```
<ion-content [fullscreen]="true">
  <p>Anzahl der Fragen: {{data.currentQuiz.questions.length}}</p>

  <ion-button shape="round"
              color="primary"
              expand="block"
              fill="outline"
              (click)="showList()">
    Fragen bearbeiten
  </ion-button>
  <ion-button shape="round" color="primary" expand="block"
              fill="outline" (click)="startTest()">
    Test starten
  </ion-button>

</ion-content>
```

QuestionList und Question anlegen (ohne Speichern)

Erklären: Back-Button und ion-list (Ionic Components), *ngFor (Angular)

Add Icons used in the application globally in
/main.ts:

```
import { addIcons } from 'ionicons';
import { addCircle, checkmark } from 'ionicons/icons';

// register icons globally
addIcons({
  addCircle,checkmark
})
```

question-list.page.html

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>Questions</ion-title>
    <ion-buttons slot="start">
      <ion-back-button defaultHref="home"></ion-back-button>
    </ion-buttons>
    <ion-buttons slot="end">
      <ion-button icon-only (click)="show('0')">
        <ion-icon name="add-circle"></ion-icon>
      </ion-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true">
  <ion-header collapse="condense">
    <ion-toolbar>
      <ion-title size="large">Questions</ion-title>
    </ion-toolbar>
  </ion-header>
  <ion-list>
    <!-- //define the id of the collection or use "track $index" for static
collections-->
    @for (q of data.currentQuiz.questions; track q.id) {
      <ion-item-sliding>
        <ion-item (click)="show(q.id)">
          <ion-label>{{q.title}}</ion-label>
        </ion-item>
        <ion-item-options side="end">
          <ion-item-option color="danger" (click)="deleteQuestion(q)">
              Delete
          </ion-item-option>
        </ion-item-options>
      </ion-item-sliding>
    }
  </ion-list>
</ion-content>
```

Die grau hinterlegten Teile sollten in einem 2. Step gezeigt werden.

Question-list.page.ts

Um nicht jedes Element von Ionic einzelnd zu importieren, kann man auch das IonicModule importieren:

```ts
import { Component, inject, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import {IonicModule} from '@ionic/angular';
import { DataService } from 'src/app/services/data.service';
import { Question } from 'src/app/services/question';
import { Router } from '@angular/router';

@Component({
  selector: 'app-question-list',
  templateUrl: './question-list.page.html',
  styleUrls: ['./question-list.page.scss'],
  standalone: true,
  imports: [IonicModule]
})
export class QuestionListPage implements OnInit {

  private router = inject(Router);
  public data = inject(DataService);

  constructor() { }

  ngOnInit() {
  }
  show(qid: string) {
    this.router.navigate(["/question", qid]);
  }
  deleteQuestion(q: Question) {
    //this.data.deleteQuestion(q);
  }
}
```

```ts
//  this.data.deleteQuestion(q); // ist noch nicht in data.service.ts implementiert
```

Question.page.html:

```
...
<ion-content>
  <ion-list>
    <ion-item>
      <ion-label position="fixed">Fragen-ID: </ion-label>
      <ion-label>{{question.id}}</ion-label>
    </ion-item>
    <ion-item>
      <ion-input label="Title:" labelPlacement="fixed" type="text"
[(ngModel)]="question.title"></ion-input>
    </ion-item>
    <ion-item>
      <ion-input label="Antwort 1:" labelPlacement="fixed" type="text"
[(ngModel)]="question.a1"></ion-input>
      <ion-button icon-only (click)="setCorrect(1)">
          <ion-icon name="checkmark"></ion-icon>
      </ion-button>
    </ion-item>
    <ion-item>
…
    <ion-item>
      <ion-label position="fixed">richtig: </ion-label>
      <ion-label position="fixed">{{question.correct}}</ion-label>
    </ion-item>
  </ion-list>
</ion-content>
```

QuestionPage.ts:
Auch hier importiere ich das IonicModule und das FormsModule (sonst funktioniert [(ngModel)] im Template nicht.

```
import { Component, inject, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import {IonicModule} from '@ionic/angular';
import { Question } from 'src/app/services/question';
import { ActivatedRoute } from '@angular/router';
import { DataService } from 'src/app/services/data.service';

@Component({
  selector: 'app-question',
  templateUrl: './question.page.html',
  styleUrls: ['./question.page.scss'],
  standalone: true,
  imports: [IonicModule, FormsModule]
})
export class QuestionPage implements OnInit {
  public question!: Question;
  private data = inject(DataService);
  private route = inject(ActivatedRoute);
```

```
  constructor() {
  }

  ngOnInit() {
    let questionId = this.route.snapshot.paramMap.get('id') || "new";
    if (questionId === "new") {
      this.question = this.data.getNewQuestion();
    } else {
      this.question = this.data.getQuestion(questionId) ||
this.data.getNewQuestion();
    }
  }

  setCorrect(i: number) {
    this.question.correct = i;
  }

  ionViewWillLeave() {
    if (this.question.id === '0' && this.question.title.length>2)
      this.data.addQuestion(this.question);
    this.data.saveQuiz();
  }

}
```

Vgl. https://ionicframework.com/docs/angular/lifecycle

Data.service.ts:

Installiere die uuid Library:

```
npm install uuid
npm i --save-dev @types/uuid
```

```
import { v4 as uuidv4 } from 'uuid';

getQuestion(questionId: string): Question | undefined {
    return this.currentQuiz.questions.find(q => q.id === questionId);
  }
  getNewQuestion(): Question {
    return {id: '0', title: '', a1: '', a2: '', a3: '', a4: '', correct: 1};
  }

  deleteQuestion(q: Question) {
    this.currentQuiz.questions = this.currentQuiz.questions.filter(question =>
question.id !== q.id);
    this.saveQuiz();
  }

  addQuestion(q: Question) {
    // add id by using uuid4
    q.id = uuidv4();
    this.currentQuiz.questions.push(q);
    this.saveQuiz();
  }

  saveQuiz(){
    Preferences.set({
      key: 'quiz',
      value: JSON.stringify(this.currentQuiz)
    }).then(() => {
      console.log('Quiz saved');
    }).catch((error) => {
      console.error('Error saving quiz', error);
    });
  }

  async loadQuiz(){
    const someObject = await Preferences.get({ key: 'quiz' });
    if (someObject.value) {
      this.currentQuiz = JSON.parse(someObject.value);
    }
  }
```

Daten lokal speichern und laden

Dokumentation zeigen:
https://ionicframework.com/docs/native/preferences

Capacitor aktivieren:
Ionic integrations list (was ist schon installiert?)
Ionic integrations enable capacitor (wenn nicht schon aktiv)
`npm install @capacitor/preferences`
und wenn man schon ein WWW-Verzeichnis hat (ionic build), dann:
`npx cap sync`

Preferences in data.service einbauen:

```
import { Preferences } from '@capacitor/preferences';
  constructor() {
    ...
    this.loadQuiz ();
  }
```

```
//add saving adata:
  public saveQuiz() {
    Preferences.set({
      key: "currentQuiz",
      value: JSON.stringify(this.currentQuiz)
    }).then(()=>{
      console.log("gespeichert");
    }).catch((reason)=>{
      console.log(reason);
    });
  }

  public async loadQuiz() {
    try {
      const obj = await Preferences.get({key: "currentQuiz"});
      if (obj.value) {
      this.currentQuiz = <Quiz> JSON.parse(obj.value);
      console.log(obj);
      console.log("loaded");
      } else {
        console.log("not found");
      }

    } catch(reason) {
      console.log(reason);
    }
  }
```

Und alle save() Aufrufe auskommentieren:

- In data.service.addQuestion und …deleteQuestion
- In question.page.ts ionViewWillLeave

RESTFull HTTP-Webservices aufrufen:

Dokumentation: https://angular.dev/guide/http

app.modulte.ts ändern:

```
import { provideHttpClient } from '@angular/common/http';

...
providers: [{ provide: RouteReuseStrategy,
                useClass: IonicRouteStrategy,
              },
              provideHttpClient()],
```

Data.service:

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

```
constructor(private http: HttpClient) {
```

```
  loadJSON() {
    this.http.get<Quiz>("/assets/data.json").subscribe((loadedData: Quiz)=>{
      if (loadedData !=null) {
        this.currentQuiz = loadedData;
        console.log(this.currentQuiz);
      } else {
        console.log("loaded data is Null");
      }
    });
  }
```

Data.json erstellen:
In dataserive.load() folgendes einbauen:

```
console.log(JSON.stringify(this.currentQuiz));
```

mit Hilfe des JSON-Validators schön machen: https://jsonformatter.curiousconcept.com/
und in assets/data.json speichern.

Und einen Button in der Homepage zum Laden anlegen.

Versuch vom Server zu laden: `https://schmiedl.co.at/json/data.json`
Im Browser möglich, aber als Webservice-Request -> CORS-Fehler!

CORS-aktivierte Variante: `https://schmiedl.co.at/json_cors/data.json`

Google Firebase einbauen

Firebase aufrufen: https://firebase.google.com/
Zur Konsole – neues Projekt anlegen
Database > neue Cloud Firestore DB anlegen > Testmodus
Neue Collection "Quizzes" anlegen + ein sinnloses Dokument (wird später gelöscht)

Öffne https://github.com/angular/angularfire2

Installiere angular2firebase: `npm install firebase @angular/fire --save`
Überprüfe, ob das Package in der package.json drin ist.

Gehe nach Installationsanleitung auf angular2firebase vor:

- Open `/src/environments/environment.ts` and add your Firebase configuration
- Kopiere die Daten aus firebase->Projekteinstellungen->neue Webapplikation
  - 
- `/src/app/app.module.ts` ändern:

```
import { AngularFireModule } from '@angular/fire';
import { AngularFirestoreModule } from '@angular/fire/firestore';
import { AngularFireAuthModule } from '@angular/fire/auth';
import { environment } from '../environments/environment';
…
imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule,
HttpClientModule, IonicStorageModule.forRoot(),
AngularFireModule.initializeApp(environment.firebase),
AngularFirestoreModule,
AngularFireAuthModule],
```

Data.Service: AngularFireStore injecten:

```
import { AngularFirestore } from '@angular/fire/firestore';
…
  constructor(private storage: Storage, private http: HttpClient,
              private db: AngularFirestore) {…
```

Aktuell gibt es einen Bug beim Ausführen:
Hier ist die Lösung: https://github.com/angular/angularfire2/issues/1993

App.module.ts:

```
import { FirestoreSettingsToken } from '@angular/fire/firestore';

...

{ provide: FirestoreSettingsToken, useValue: {} }
```

Data.service: connect einbauen und im constructor aufrufen:

```
  constructor(private storage: Storage, private http: HttpClient,
        private db: AngularFirestore) {
    this.load();
    this.connectToFirebase();
  }

  private quizAFSCollection: AngularFirestoreCollection<Quiz>;
  private quizObservable: Observable<Quiz[]>

  connectToFirebase() {
    this.quizAFSCollection = this.db.collection("quizzes");
    this.quizObservable = this.quizAFSCollection.valueChanges();
    this.quizObservable.subscribe( (data: any) => {
        console.log(data);
    }); //die gelben dann wieder auskommentieren!!!
  }

//*** Ende Firebase */
```

Wenn man jetzt in die DB neue Werte eingibt, werden sie sofort geloggt.

Quizzes speichern – und Button in Homepage einbauen:

```
saveQuizToFirebase() {
    let id: string;
    if (this.currentQuiz.id == "") {
      id = this.db.createId();
      this.currentQuiz.id = id;
    }
    this.quizAFSCollection.doc(this.currentQuiz.id).set(this.currentQuiz);
  }
```

Homepage:

```
  <ion-item>
    <ion-label position="floating">Quizzname</ion-label>
    <ion-input type="text" [(ngModel)]="data.currentQuiz.quizName"></ion-input>
  </ion-item>
  <ion-button shape="round" color="primary" expand="block"
```

```
            fill="outline" (click)="saveQuizzToFirebase()">
    Quizz in Firebase speichern
  </ion-button>
```

Jetzt einmal aufrufen um das Quizz in Firebase zu speichern.
Check Firebase und lösche die Dummyeinträge.

Quizzes von Firestore anzeigen:

Erzeuge neue Page quiz-list und lege einen Button zum Aufruf in der Homepage an.
ionic g page pages/QuizzList

Quiz-List.page.html

```
…
<ion-content>
  <ion-list>
    <ion-item *ngFor="let quiz of data.quizObservable | async">
      <ion-label (click)="load(quiz)">{{quiz.quizName}}</ion-label>
      <ion-button (click)="delete(quiz)">
        <ion-icon slot="icon-only" name="trash"></ion-icon>
      </ion-button>
    </ion-item>
  </ion-list>
</ion-content>
```

Quiz-List.page.ts:

```
import { Component, OnInit } from '@angular/core';
import { Quiz } from 'src/app/interfaces/Quiz';
import { DataService } from 'src/app/services/data.service';
import { ToastController } from '@ionic/angular';

@Component({
  selector: 'app-quizz-list',
  templateUrl: './quizz-list.page.html',
  styleUrls: ['./quizz-list.page.scss'],
})
export class QuizzListPage implements OnInit {

  constructor(private data: DataService, private toast: ToastController,
              private navCtrl: NavController) { }

  ngOnInit() {}

  load(q: Quiz) {
    this.data.currentQuiz = q;
    this.data.save();
    this.navCtrl.navigateBack("/");
  }

  async delete(q: Quiz) {
    await this.data.deleteQuizFromFirebase(q);
    const t = await this.toast.create({
```

```
      message: "Quiz wurde gelöscht",
      duration: 3000
    });
    t.present();
  }
}

Data.Service:
  deleteQuizFromFirebase(quiz: Quiz): Promise<void> {
    return this.quizAFSCollection.doc(quiz.id).delete();
  }
```

Firebase Hosting:

```
Install the Firebase CLI: npm install -g firebase-tools
firebase login
firebase list
```

Ionic Projekt einmal builden:

```
Ionic build
```

Initialisiere das Projekt (im Projektverzeichnis)

```
firebase init
```

public director: www
single-page app: yes
overwite: No

```
firebase deploy
```

Am Device zum Laufen bringen: Capacitor einbauen

https://capacitor.ionicframework.com/
https://capacitor.ionicframework.com/docs/getting-started/with-ionic

```
//npm install --save @capacitor/core @capacitor/cli
// ersetzt durch ionic integrations enable capacitor
ionic build
```

```
Es wird ein neues Verzeichnis angelegt: www
```

```
Einmal machen:
```

- `npx cap init //legt capacitor.config.ts an. Bundle-Identifier NUR KLEINBUCHSTABEN`
- `npm install @capacitor/ios`
- `npm install @capacitor/android`
- `npx cap add ios`
- `npx cap add android`

```
Am Beginn und nach jeder Änderung
ionic build
npx cap sync
```

```
npx cap open ios / android
```

App in den Apple Appstore bringen

```
Getestet mit MacOS 15.1, XCode 16.1
```

Schritt1: App am Simulator und iPhone zum Laufen bringen

Das Starten von XCode und Deployment auf Simulator oder iPhone hsollte funktionieren. Eventuell muss man am iPhone unter Einstellunegn->Allgemein->VPN und Geräteverwaltung die App freigeben, um die App am iPhone zum Laufen zu bekommen.

Schritt2: in den App Store bringen und mit TestFlight freigeben:

XCode: Menü->Product->Archive

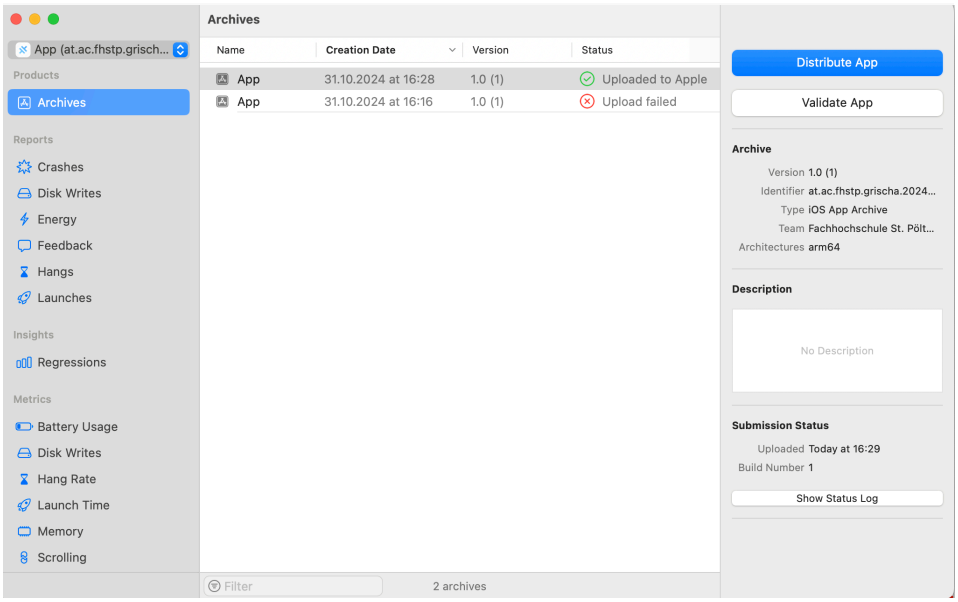
In meiner Version kam ein Buildfehler: rsync konnte irgendwelche Dateien nicht kopieren.

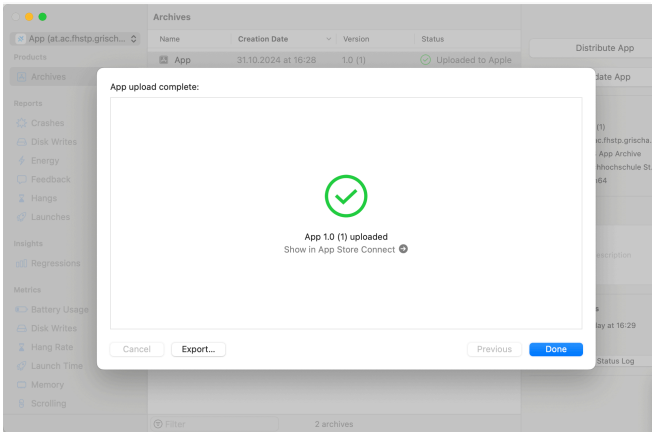
Fix:
Cocoapod updaten/installieren: kann man mit Homebrew (Befehl brew install cocoapods oder brew reinstall cocoapods installieren/reinstallieren)

Der Fehler kommt trotzdem, und kann folgendermaßen gelöst werden:
2024/10: Bug mit Cocoapods lösen:
in Xcode APPNAME/ios/App/Pods/Target Support Files/Pods-App/Pods-App-frameworks.sh:
Diese Zeile suchen: source="$(readlink "${source}")"
und so ersetzen: source="$(readlink **-f** "${source}")"

Dann sollte das Builden funktionieren und es kommt dieser Screen (Status aber noch nicht uploaded)

Distribute App - Button klicken




Archives
Name
Creation Date
Version
Status
App
31.10.2024 at 16:28
1.0 (1)
Uploaded to Apple
Distribute App
App upload complete:
App 1.0 (1) uploaded
Show in App Store Connect
Cancel
Export...
Previous
Done
Products
Archives
Reports
Crashes
Disk Writes
Energy
Feedback
Hangs
Launches
Insights
Regressions
Metrics
Battery Usage
Disk Writes
Hang Rate
Launch Time
Memory
Scrolling
Filter
2 archives

Weitere Developer-Tipps:

Embedded Javascript in den Templates mit Autocompletion: Installiere das VS Code Plugin „Angular Language Service“

Debuggen:

- Primitiv: im Code „debugger;“ einfügen und in Chrome debuggen
- Mit VS Code:
  - In der Debug-Pane eine neue Config dazufügen
  - Die URL der Config auf die URL im Chrome setzen, die geöffnet wird, wenn man ionic serve eingibt.
  - Mit „Ionic serve“, aber dann den Browser schließen und aus VS.Code debuggen starten.