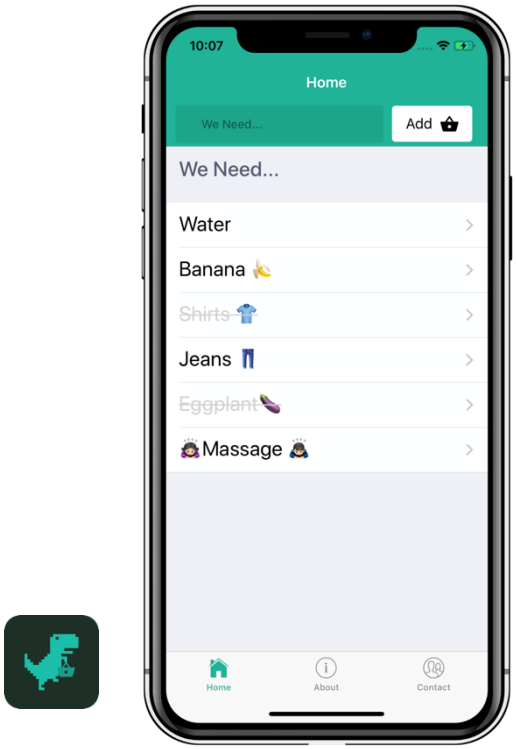
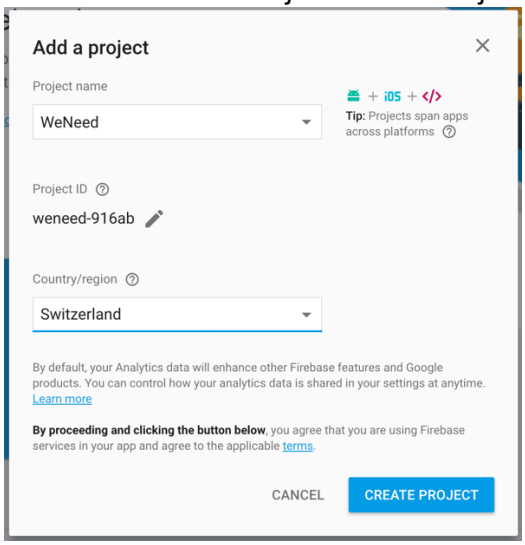
Aufgabe 3

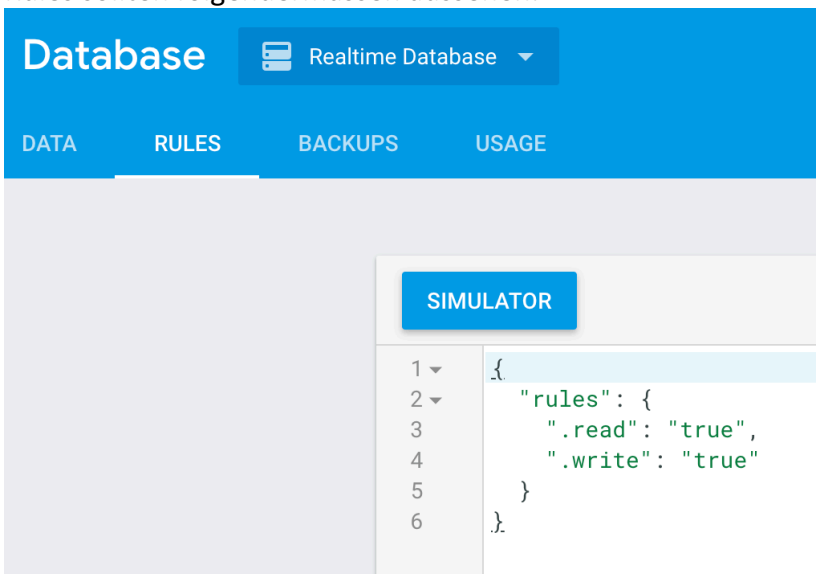
Wir erstellen eine App **WeNeed**, mit welcher wir Dinge die wir brauchen speichern können. Zum persistieren der Daten verwenden wir Firebase welches eine Realtime Database anbietet.



1. Zuerst verlassen wir unsere Code Editor und erstellen uns einen Account bei Firebase. Gehe dazu auf:
    a. https://firebase.google.com/
    b. Gehe zur Konsole -> ZUR KONSOLE
    c. Erstelle ein neues Projekt -> Add Project
    d. 

    e. Im Seitenmenu wähle DEVELOP -> Database -> Realtime Database aus

f. Unter dem Tab RULES können wir die Authentifizierungsregeln festlegen. Wir setzten sie für jetzt Read und Write auf ohne Authentifizierung. Du kannst diese später wieder ändern wenn du am Login für deine App arbeitest. Die Rules sollten folgendermassen aussehen:



g. Als nächstes aktiviere die Authentifizierungsmodus damit anonymer Zugriff erlaubt ist, indem du im Sidemenu auf Authentication gehst und dann auf SIGN-IN METHOD klickst. Klicke auf Anonymous in der Liste und setze es den Status auf **Enabled.**

h. Nun brauchen wir noch die Config Infors zu unserer neu erstellten Firebase DB damit wir aus unserer Daten lesen und schreiben können. Die Config Infos findest du auf dieser Authentication Seite oben rechts unter **WEB SETUP.** Kopiere den Code ab apiKey bis und mit messagingSenderId. Wir brauchen Code im übernächsten Schritt.

2. Erstelle als nächstes ein neues Ionic Projekt
3. Als nächstes installieren wir Angularfire2. Angularfire macht es etwas einfacher unsere CRUD Operationen auf Firebase auszuführen. Du kannst aber auch die regulären Funktionen von Firebase verwenden oder sogar beides kombiniert. Was dir einfacher erscheint. In diesem Beispiel verwenden wir Angularfire2.
   a. npm install firebase angularfire2 --save (unter Umständen mit sudo)
   b. Füge einen Provider zu deinem Projekt hinzu damit wir alle CRUD Operationen von aus dort verwalten können.
      i. ionic g provider firebaseService
4. In app.module.ts füge vor @NgModule deine kopierte Firebase Config ein:

```
const firebaseConfig = {
  apiKey: "XXXX",
  authDomain: "XXX",
  databaseURL: "XXX",
  projectId: "XXX",
  storageBucket: "XXX",
  messagingSenderId: "XX"
};
```

5. Importiere folgende Module

```
import {HttpModule} from '@angular/http';
import {AngularFireDatabaseModule} from 'angularfire2/database';
import {AngularFireModule} from 'angularfire2';
```

a. Und füge sie den imports hinzu. Achte bei AngularFireModule darauf die Config zu initialisieren

```
    HttpModule,
    AngularFireDatabaseModule,
    AngularFireModule.initializeApp(firebaseConfig),
```

6. Als nächstes fügen wir die CRUD Operationen in firebase-service.ts um. Achte, dass alle Imports vorhanden sind:

```
import {AngularFireDatabase} from 'angularfire2/database';
import {AngularFireList} from 'angularfire2/database';
import { Observable } from 'rxjs/Observable';
```

7. Als Membervariablen verwenden wir:

```
itemsRef: AngularFireList<any>;
 items: Observable<any[]>;
```

8. Ergänze den Constructor folgendermassen, entferne HttpClient:

```
  constructor(public afd: AngularFireDatabase) {
    this.itemsRef = this.afd.list('/weneedItems/');
    this.items = this.itemsRef.snapshotChanges().map(changes => {
      return changes.map(c => ({ key: c.payload.key, ...c.payload.val() }));
    });
  }
```

9. Für die CRUD Operationen:

```
    getItems(){
      return this.items;
    }
    addItem(newName) {
      return this.itemsRef.push({ value: newName, isDone: false });
    }
    updateItem(key, newText) {
      return this.itemsRef.update(key, { value: newText });

    }

    //sets an item to done or undone
   doneItem(key, status) {
      return this.itemsRef.update(key, { isDone: status });
    }

    deleteItem(key) {
      this.itemsRef.remove(key);
    }
```

10. Wir können diese nun in unserem home.ts verwenden 👍

a. Importiere noch

```
import { FirebaseServiceProvider } from '../../../providers/firebase-
service/firebase-service';
import { Observable } from 'rxjs/Observable';
```

11. Wir speichern unsere Liste in der Variable:

```
needItems: Observable<any[]>;
```

12. Und die neuen Items welche der User eingibt in:

```
newItem: any = '';
```

13. Wir wollen auch nach jedem Eintrag in der Liste in der View ganz nach unten Scrollen, damit wir den neuen Eintrag sehen. Dazu verwenden wir ViewChild und Conten. Importiere diese:

```
import { Component, ViewChild } from '@angular/core';
import { NavController, Content } from 'ionic-angular';
```

    a. Vor dem Constructor können wir nun auf den View Content zugreifen

```
@ViewChild(Content) content: Content;
```

14. Im Constructor füge die Dependency zum FirebaseServiceProvider hinzu

```
constructor(public navCtrl: NavController, public firebaseService:
FirebaseServiceProvider,...
```

15. Wir können nun die Liste im Constructor abfüllen

```
16. this.needItems = this.firebaseService.getItems();
```

17. Nun können wir Items hinzufügen, löschen, entfernen, auf erledigt setzen und editieren.

    a. Hinzufügen:

```
  addItem(){
    if(this.newItem.length === 0 || !this.newItem.trim()){
      console.log("empty");
    }else{
      this.firebaseService.addItem(this.newItem).then(()=>{
      this.newItem = "";
   //   this.keyboard.close();
      this.content.scrollToBottom();
      });
    }
  }
```

    b. Löschen:

```
  removeItem(id){
    this.firebaseService.deleteItem(id);
   }
```

    c. Auf erledigt setzen

```
   doneItem(key, status){
      this.firebaseService.doneItem(key, status);
   }
```

    d. Falls der User in der View Scrollt und die Tastatur offen ist, wollen wir diese schliessen. Dazu schreiben wir folgende Funktion (this.keyboard.close() ist noch auskommentiert, wir müssen zuerst noch das Keyboardplugin installiern. Dazu am Schluss der Übung:

```
  onScroll(event){
   // this.keyboard.close();
}
```

    e. Editieren werden wir in einer Detail Ansicht erlauben. Dazu füge eine neue Page hinzu und importiere ebenfalls deinen FirebaseServiceProvider.

```
import { FirebaseServiceProvider } from '../../providers/firebase-
service/firebase-service';
```

        i. Vergiss nicht den Service im Constructor zu laden
        ii. Die Update Funktion könnte folgendermassen aussehen:

```
  updateItem(){
    this.firebaseService.updateItem(this.selectedItem.key,
this.selectedItem.value).then(()=>{
```

```
      this.navCtrl.pop();
    });
  }
```

18. Zur View in Home.html
    a. Kleine Infos:
    b. Damit der User auch mit der „Enter“ Tase ein Item hinzufügen kann, fangen wir dies mit (keyup.enter) ab.
    c. Wenn der User die Tastatur offen hat und die View Scrollt, wollen wir die Tastatur schliessen. Wir können auf ion-conten den Scroll Event mit (ionScroll) abfangen und die nötige Funktion aufrufen um das Keyboard zu schliessen.
    d. Mit *ngIf können wir je nachdem ob der User ein Item streichen möchte zwischen den Buttons Done und Undo wechseln. Du könntest den Text auch als Expression implementieren.

```
<ion-header>
  <ion-navbar>
    <ion-title>Home</ion-title>
  </ion-navbar>

  <ion-toolbar>
      <ion-searchbar [(ngModel)]="newItem" (keyup.enter)="addItem()"
(ionCancel)="this.newItem=''" placeholder="We Need..."></ion-searchbar>
      <ion-buttons end>
        <button ion-button solid icon-right (click)="addItem()" style="margin-
right:12px; padding:21px 16px 21px 16px">Add
          <ion-icon name="md-basket"></ion-icon>
        </button>
      </ion-buttons>
  </ion-toolbar>

</ion-header>

<ion-content (ionScroll)="onScroll($event)">

  <div padding>
    <h2 style="color:#5b5b75">We Need...</h2>
  </div>

  <ion-list no-padding>
    <ion-item-sliding *ngFor="let item of needItems | async">
      <ion-item (click)="itemTapped($event, item)" detail-push>
       <h1 *ngIf="item.isDone==false">{{item.value}}</h1>
       <h1 *ngIf="item.isDone!=false" style="text-decoration: line-through;
color:rgb(211, 211, 211)">{{item.value}}</h1>
      </ion-item>
      <ion-item-options side="right">
        <button ion-button color="danger" icon-only
(click)="removeItem(item.key)"><ion-icon name="trash"></ion-icon></button>
      </ion-item-options>
      <ion-item-options (ionSwipe)="doneItem(item.key, !item.isDone)"
side="left">
```

```
        <button ion-button *ngIf="!item.isDone" color="secondary"
(click)="doneItem(item.key, !item.isDone)">Done</button>
        <button ion-button *ngIf="item.isDone" color="dark"
(click)="doneItem(item.key, !item.isDone)">Undo</button>
      </ion-item-options>
    </ion-item-sliding>
  </ion-list>

</ion-content>
```

19. Jetzt noch etwas Styling damit es dir am besten passt!
20. BONUS – Keyboard Plugin
    a. sudo npm install @ionic-native/core --save
    b. sudo ionic cordova plugin add ionic-plugin-keyboard
    c. sudo npm install --save @ionic-native/keyboard
21. Importiere in app.module.ts

```
import { Keyboard } from '@ionic-native/keyboard';
```

    a. Und füge es zur Liste der Providers:

```
  providers: [
    StatusBar,
    SplashScreen,
    Keyboard,
```

22. In home.ts kannst du nun Keyboard verwenden:

```
import { Keyboard } from '@ionic-native/keyboard';
```

    a. Ebenfalls im Constructor laden:

```
constructor(public navCtrl: NavController, public firebaseService:
FirebaseServiceProvider, private keyboard: Keyboard,
```

23. Nun kannst du die Stellen in home.ts, an welchen wir this.keyboard auskommentiert haben, verwenden.

24. Done