

Développement d'une application To do list

Avec Ionic et Firebase

Réalisé par :

IMANE BARIGOU

OLAYA EL YOUSSEFI

YOUSRA LABROUKI

FIRDAOUS CHAKIR

Encadre par :

SARRA ROUBI

Lien drive vers la vidéo présentative et descriptive du projet (démonstration):

https://drive.google.com/file/d/1KPl_ZXZhuJwd_0KGF-d7x--bZAyspnn5/view?usp=share_link

I. Introduction





1) Contexte et objectifs de l'application

Contexte :

Dans notre monde actuel, il est courant de jongler entre différentes tâches et responsabilités, qu'il s'agisse de travailler, d'étudier ou de gérer des tâches personnelles. Cependant, il peut être difficile de tout garder en tête et de suivre les choses à faire. De plus, les listes de tâches papier peuvent être faciles à perdre ou à oublier. Pour résoudre ce problème, nous avons décidé de créer une application de liste de tâches pour aider les gens à gérer leurs tâches plus facilement et efficacement.

Objectifs :

L'objectif de notre application de liste de tâches est de fournir une solution simple et pratique pour aider les utilisateurs à gérer leurs tâches quotidiennes. Nous avons pour objectif de créer une application qui :

-  Permet aux utilisateurs de créer et de modifier des listes de tâches rapidement et facilement
-  Fournit une interface intuitive et conviviale pour les utilisateurs
-  Offre des fonctionnalités avancées telles que la possibilité de définir des rappels pour les tâches, d'organiser les tâches par ordre de priorité, de catégoriser les tâches, etc.
-  Synchronise les listes de tâches sur plusieurs appareils pour que les utilisateurs puissent y accéder de n'importe où et à tout moment
- Permet aux utilisateurs de partager leurs listes de tâches avec d'autres personnes, telles que des amis, des collègues ou des membres de la famille, pour une meilleure collaboration et coordination

En créant cette application, nous espérons aider les gens à être plus organisés, productifs et efficaces dans leur vie quotidienne.

II. Conception de l'application

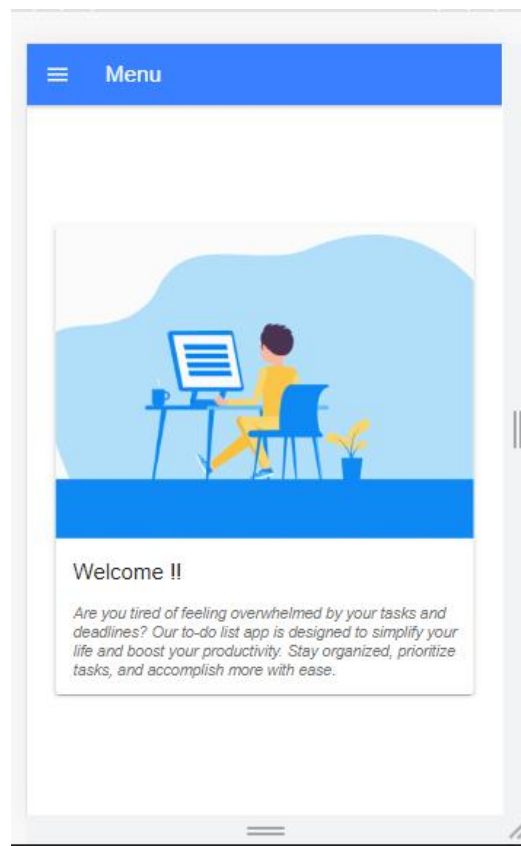
Description de l'architecture de l'application

- **Interface d'accueil :**

Cette interface est souvent la première chose que les utilisateurs voient lorsqu'ils ouvrent l'application contient une barre de menu en haut peut être utilisée pour naviguer entre les différentes parties.

Le message de bienvenue qui s'affiche sous la barre de menu qui contient un texte d'accueil pour les utilisateurs de l'application

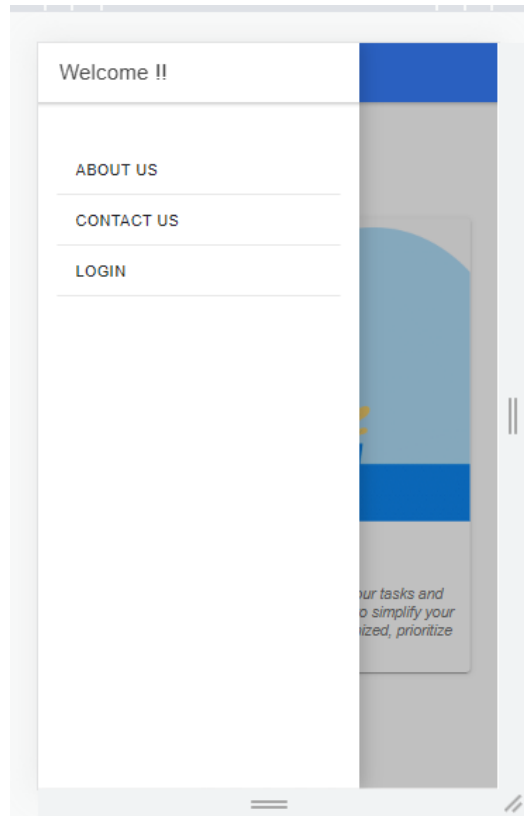
Voilà à quoi ça ressemble cette interface :



- **L'interface menu :**

L'interface menu est accessible depuis l'écran d'accueil de l'application. Elle est composée de trois boutons : "ABOUT US", "CONTACT US" et "LOGIN". Chaque bouton est associé à une page spécifique de l'application.

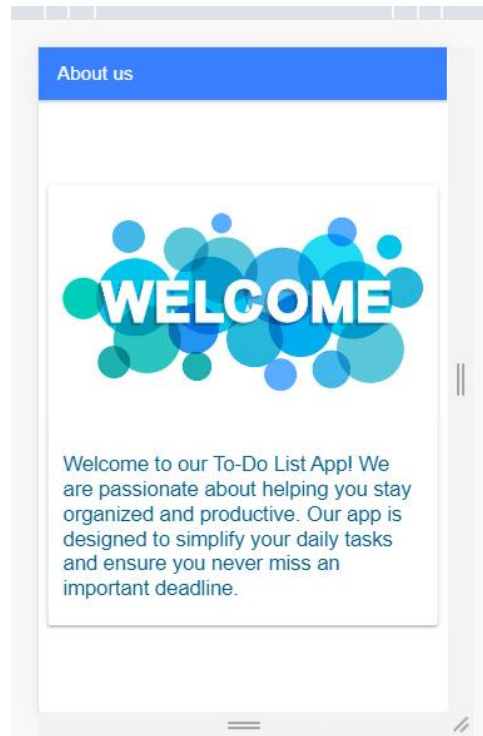
Voilà à quoi ça ressemble cette interface :



- **Interface "ABOUT US" :**

Cette interface est accessible en cliquant sur le bouton "À propos de nous" de la page d'accueil de l'application. Elle fournit aux utilisateurs une description en générale de l'application.

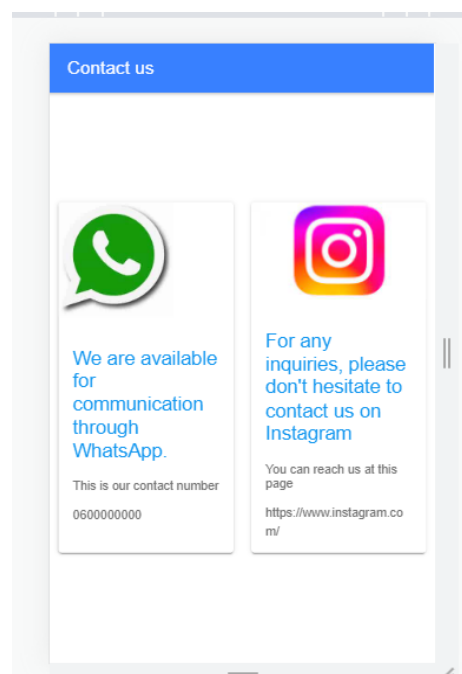
Voilà à quoi ça ressemble cette interface :



- **Interface "Nous contacter" :**

Cette interface est accessible en cliquant sur le bouton "CONTACT US" de la page d'accueil de l'application. Elle fournit aux utilisateurs les informations nécessaires pour contacter nous, notamment une adresse du compte instagram, un numéro de téléphone. Cette interface peut pour permettre aux utilisateurs de soumettre des demandes ou des questions.

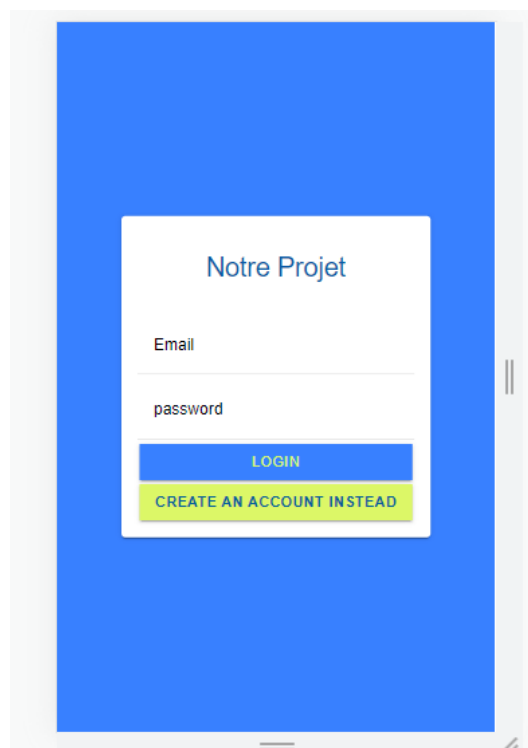
Voilà à quoi ça ressemble cette interface :



- **Interface de connexion :**

Cette interface est accessible en cliquant sur le bouton "Se connecter". Elle permet aux utilisateurs de se connecter à l'application en entrant leur email et leur mot de passe. Cette interface

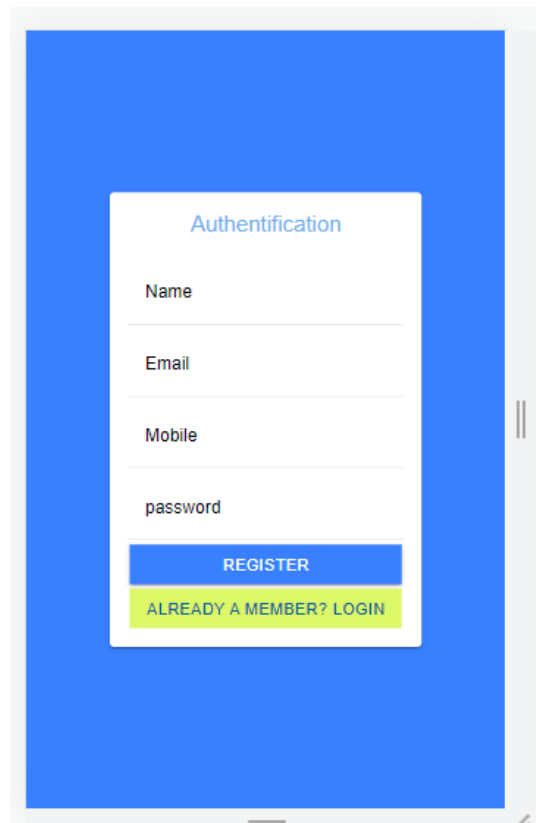
Voilà à quoi ça ressemble cette interface :



Le bouton "LOGIN" est destiné aux utilisateurs qui ont déjà un compte sur l'application.

Le bouton "S'inscrire" est destiné aux utilisateurs qui ne sont pas encore inscrits à l'application. Ce bouton redirige les utilisateurs vers l'interface d'inscription où ils peuvent créer un compte pour accéder à toutes les fonctionnalités de l'application.

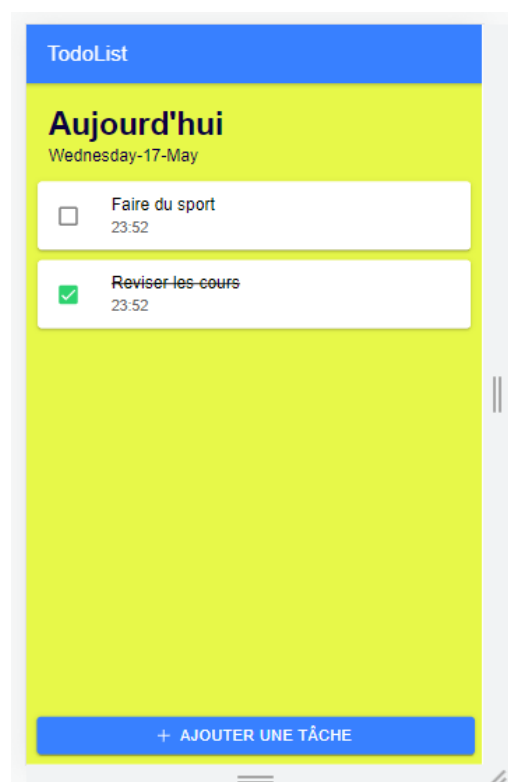
Voilà à quoi ça ressemble **l'interface d'authentification** :



- **Interface utilisateur d'accueil :**

L'interface utilisateur d'accueil est la première page que les utilisateurs voient après s'être connectés. Elle affiche les listes de tâches actuelles de l'utilisateur et permet à l'utilisateur d'en créer de nouvelles, de les modifier ou de les supprimer.

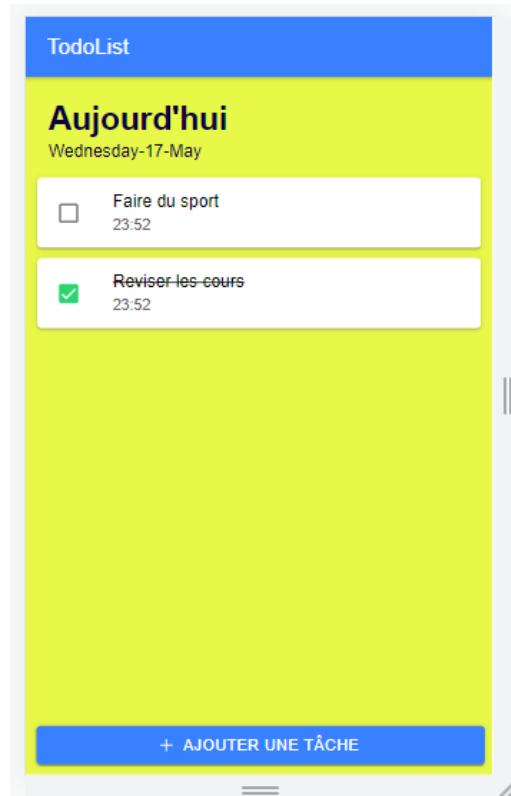
Voilà à quoi ça ressemble l'interface :



- **Interface utilisateur de création de tâche :**

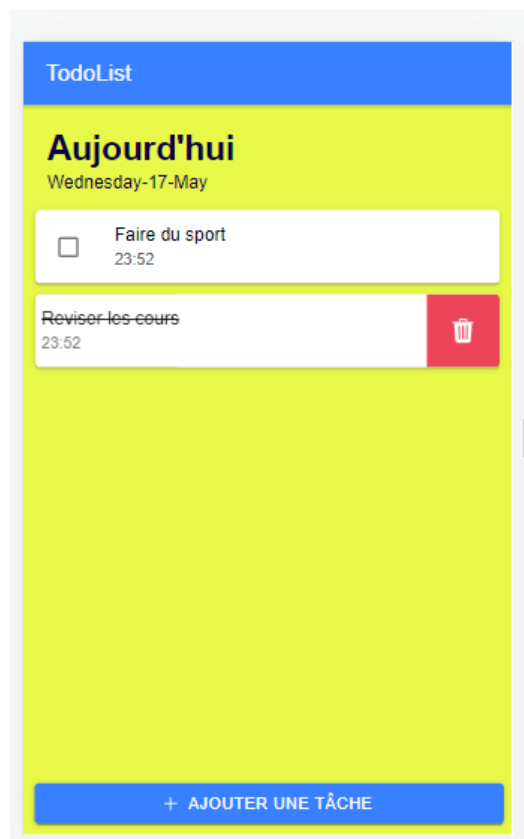
Cette interface utilisateur permet à l'utilisateur de créer une nouvelle tâche. Elle inclut des champs pour le titre de la tâche et une description facultative, ainsi qu'un bouton pour enregistrer la tâche.

Voilà à quoi ça ressemble l'interface :



- **Interface utilisateur de suppression de tâche :** Cette interface utilisateur permet à l'utilisateur de supprimer une tâche existante dans une liste de tâches. Elle demande une confirmation de l'utilisateur avant de supprimer la tâche.

Voilà à quoi ça ressemble l'interface :



III. Implémentation de l'application :

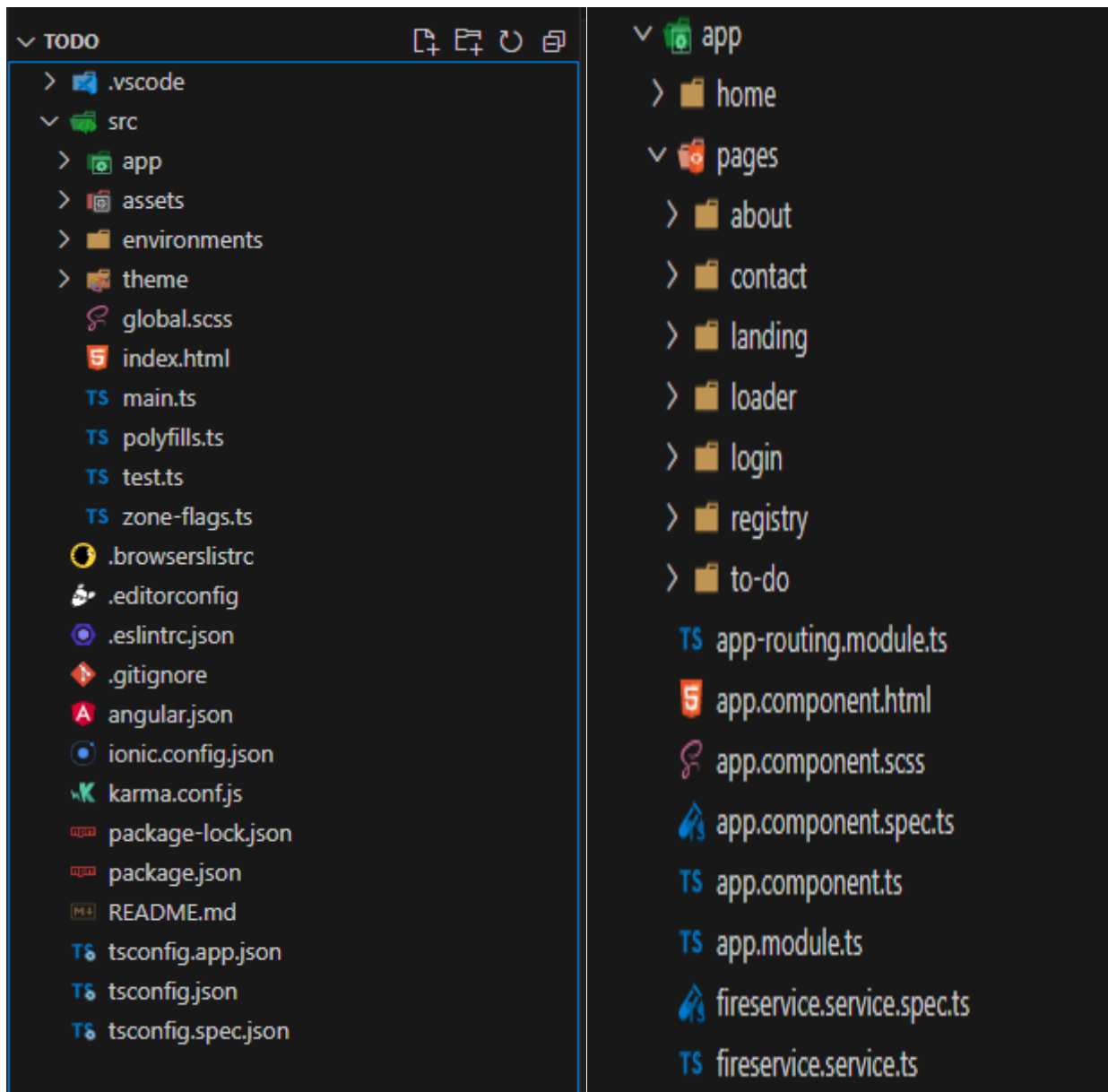
Explication des différentes étapes de développement :

Etapas de création du mini-projet :

1. Création et Configuration d'un projet Ionic

- Pour commencer nous avons créé une application de type Blank, vierge, avec une seule page Home.
- On a conçu le front avec des composantes Ionic (pages) en introduisant quelque style sur les pages.
 - about
 - contact
 - login
 - landing (about & contact && login)
 - registry
 - to-do

Arborescence du projet



Discussion :

- ✓ Le dossier **src** : C'est le dossier que nous aurons à plus manipuler pendant le développement de notre application. Dans ce dossier nous avons des sous-dossiers et fichiers.

- ✓ Le dossier **App** : Qui contient toutes les pages de notre application.
 - Chaque page est représentée par un dossier contenant des fichiers :
 - Template (.html)
 - Style (.scss)
 - Controller (.ts)
 - Test (.spec.ts)
 - Routing
 - D'autres fichiers qu'on peut ajoutés
- ✓ Le dossier **Assets** : Dans lequel nous allons mettre nos images et icones.
- ✓ Le dossier **environnement** : Qui contient des fichiers d'environnement. Surtout utiles lors du passage à la production.
- ✓ Le dossier **theme** : Qui contient le fichier de configuration du thème de notre application (couleurs de base d' Ionic)
- ✓ Le fichier **global.css** pour écrire votre css global à l'application et le fichier index.html qui est le point d'entrée de notre application.

2. **Création** d'un projet Firebase && **Récupération** du code de configuration :

- Création du projet Firebase pour pouvoir accéder aux services que propose Firebase.
- Les services utilisés sont le Database de Firebase (AngularFirestoreModule), pour stocker des informations, & AngularFireAuthModule, && AngularFireDatabaseModule
- Récupération du code de configuration qui identifie un projet Firebase et l'inclure dans le projet Ionic en local
Pour effectuer la liaison entre les deux projets (back « Firebase » && front « Ionic »)

```
export const firebaseConfig = {
  apiKey: "AIzaSyALN7rQMNUbh6VvvEHhHAs8WuMolW_ti4w",
  authDomain: "miniprojet2-e7b17.firebaseio.com",
  projectId: "miniprojet2-e7b17",
  storageBucket: "miniprojet2-e7b17.appspot.com",
  messagingSenderId: "150860262953",
  appId: "1:150860262953:web:a60597bf30ad729f7d3638",
  measurementId: "G-XJM3VDRFQD"
};
```

3. Installation des package Firebases et AngularFire

- installer le plugin qui nous permet de faire communiquer Firebase et Ionic : le plugin AngularFire.
- Chacun de ces modules est utile à quelque chose de particulier dans notre application et son fonctionnement ; **AngularFireModule**: permet d'associer notre application Ionic à notre projet Firebase grâce au code de configuration,
AngularFireAuthModule: contient toutes les fonctions d'authentification, par email et mot de passe, Facebook ou Google
AngularFireDatabaseModule: donne accès aux fonctions de base de données Firebase (envoi et récupération d'informations sous forme d'objet et de liste)
- Ensuite, il nous faut déclarer ces plugins pour pouvoir les utiliser dans le reste de l'application. Pour cela, nous avons déclaré les deux plugins suivants : **AngularFireModule.initializeApp (firebaseConfig)** & **AngularFireDatabaseModule, && AngularFireAuthModule** dans le tableau imports, qui va s'exécuter au lancement de l'application

4. Commencer à **développer** (logique de l'application)



Les fonctionnalités :

- S'authentifier & Créer un nouveau compte (Registration)
- Afficher des éléments de la Todolist dans Ionic
- Ajouter des éléments à la TodoList un par un
- Cocher les cases et les synchroniser avec Firebase (opération de la suppression)

Discussion

Tout ce passe au sein des pages de notre projet

En se basant sur le modèle MVC.

Login && Registration

Soit le service« fireservice.service.ts »

```
export class FireserviceService {  
  constructor( public firestore: AngularFirestore,  
    public auth: AngularFireAuth) {  
  }  
  loginWithEmail(data:any) {  
    return this.auth.signInWithEmailAndPassword(data.email, data.password);  
  }  
  
  signup(data:any) {  
    return this.auth.createUserWithEmailAndPassword(data.email, data.password);  
  }  
}
```

Déclaration d'un objet auth (AngularFireAuth) pour pouvoir utiliser les fonctions fournies par ce module :

- **SignInWithEmailAndPassword && createUserWithEmailAndPassword**

Soit le fichier typeScript « login.page.ts »

```

public fireService: FireserviceService, private navCtrl: NavController) {
  this.ionicForm = this.formBuilder.group({
    name: ['', [Validators.required, Validators.minLength(2)]],
    email: ['', [Validators.required, Validators.pattern('[a-zA-Z0-9._-]+@[a-zA-Z0-9._-]+.[a-zA-Z.]{2,15}')]],
    password: ['', [Validators.required, Validators.minLength(2)]],
  })
}

ngOnInit() {
}

login(){
  this.fireService.loginWithEmail({email:this.email,password:this.password}).then(res=>{
    console.log(res);
    if(res.user){
      this.fireService.getDetails({uid:res.user.uid}).subscribe(res=>{
        console.log(res);
        alert('Welcome ');
      },(err: any) =>{
        console.log(err);
      });
    }
  },(err: any) =>{
    alert(err.message)
    console.log(err);
  })
  this.navCtrl.navigateForward('to-do');
}
}

```

Soit le fichier template« login.page.html »

```

<form [formGroup]="ionicForm" (ngSubmit)="submitForm()" novalidate>
  <ion-item lines="full">
    <ion-label position="floating">Email</ion-label>
    <ion-input formControlName="email" [(ngModel)]="email" type="email"></ion-input>
  </ion-item>
  <span class="error ion-padding" *ngIf="isSubmitted && errorControl['email'].errors?.['required']">
    Email is required.
  </span>
  <span class="error ion-padding" *ngIf="isSubmitted && errorControl['email'].errors?.['pattern']">
    Please provide valid email id.
  </span>
  <ion-item >
    <ion-label position="floating">password</ion-label>
    <ion-input formControlName="password" [(ngModel)]="password" type="password"></ion-input>
  </ion-item>
  <span class="error ion-padding" *ngIf="isSubmitted && errorControl['password'].errors?.['required']">
    password is required.
  </span>
  <span class="error ion-padding" *ngIf="isSubmitted && errorControl['password'].errors?.['minlength']">
    password should be min 2 chars long.
  </span>

```

Après avoir récupéré les informations des champs du formulaire (view), On passe au traitement dans le controller (typescript) dont on a injecté le service FireserviceService, on passe valider les champs du formulaire et utiliser les

fonctions déjà déclarer dans ce service pour s'assurer que l'utilisateur existe dans la base de données si non on déclenche un alerte pour informer l'utilisateur de l'erreur au sein des informations d'identification .

De même dans la registration faut qu'on utilise la fonction responsable de la création d'un nouveau compte déclarée ainsi dans le service.

- **Affichage et suppression des éléments de la Todolist**

Soit le fichier typescript

```
getTasks() {  
  this.afDB.list('Tasks/').snapshotChanges(['child_added', 'child_removed']).subscribe(actions => {  
    this.tasks = [];  
    actions.forEach(action => {  
      this.tasks.push({  
        key: action.key,  
        text: action.payload.exportVal().text,  
        hour: action.payload.exportVal().date.substring(11, 16),  
        checked: action.payload.exportVal().checked  
      });  
    });  
  });  
}
```

```
deleteTask(task: any) {  
  this.afDB.list('Tasks/').remove(task.key);  
}
```

Par intermédiaire de l'objet afDB déclaré de AngularFireDatabase, on a pu utiliser la fonction list pour récupérer tous les éléments de la table Tasks déclarer dans Firebase, et les stockés dans la table tasks dans notre projet Ionic

Qui va contenir les informations liées aux différentes tâches stockés dans Firebase et avec lequel on va faire un fetch des données dans la template .On a ainsi la fonctionnalité delete(Task) pour supprimer une tâche identifier par son mot clé (key)

Soit le fichier template

```
<ion-card *ngFor="let task of tasks" >
  <ion-item-sliding >
    <ion-item lines="none" >
      <ion-checkbox (ionChange)="changeCheckState(task)" color="success" [(ngModel)]= "task.checked" slot="start"></ion-checkbox>
      <ion-label>
        <h2 *ngIf="!task.checked">{{ task.text }}</h2>
        <h2 *ngIf="task.checked" style="text-decoration:line-through;">{{ task.text }}</h2>
        <p>{{ task.hour }}</p>
      </ion-label>
    </ion-item>
    <ion-item-options side="end">
      <ion-item-option color="danger" (click)="deleteTask(task)">
        <ion-icon name="trash" slot="icon-only"></ion-icon>
      </ion-item-option>
    </ion-item-options>
  </ion-item-sliding>
</ion-card>
```

- Ajout des éléments de la todoList un par un

Soit le fichier template:

```
<ion-card>
  <ion-item lines="none">
    <ion-input [(ngModel)]= "myTask" placeholder="Entrer une tâche"></ion-input>
    <ion-button (click)="addTaskToFirebase()" shape="round" slot="end">
      <ion-icon slot="icon-only" name="add"></ion-icon>
    </ion-button>
  </ion-item>
</ion-card>
```

Qui représente un formulaire (reactive forme) et le mécanisme du buinding pour envoyer les données au typescript

Soit le fichier typescript :

```
addTaskToFirebase() {
  this.afDB.list('Tasks/').push({
    text: this.myTask,
    date: new Date().toISOString(),
    checked: false
  });
  this.showForm();
}
```


Récupération du champ texte de la tâche et génération des autres informations, puis un push vers la table Tasks de la base de données, sans oublier le fait de réafficher la forme pour lister toutes les tâches existantes.

V. Conclusion :

En conclusion, cette application to-do list est un outil efficace pour aider les utilisateurs à organiser leurs tâches et à améliorer leur productivité. En utilisant une conception d'interface utilisateur intuitive et conviviale, l'application offre une expérience utilisateur agréable et facile à utiliser.