DÉVELOPPEMENT DES APPLICATIONS MOBILES HYBRIDES MULTIPLATEFORMES

CH 3 - Templates et Customisation

FORMATEUR : MAHAMANE SALISSOU YAHAYA : ysalissou@gmail.com

(TechnoLAB -ISTA)

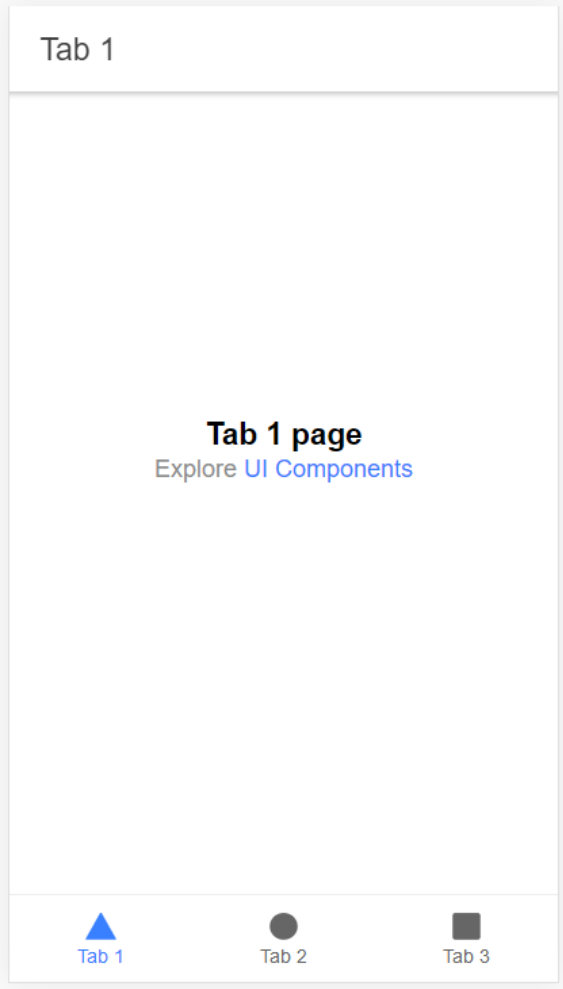
4ième année TEL

OCTOBRE 2021

Dans le chapitre précédent nous avons installé Ionic et ses dépendances. Nous avons également pu créer notre application GUIDE D'ORIENTATION et nous nous sommes amusés à le modifier tant bien que mal.

https://enseignementsuperieur.gouv.bj/doc/GUIDE%20D'ORIENTATION%202021-2022.pdf

Dans ce chapitre, nous allons apprendre à customiser un peu plus notre application mobile et à créer de nouvelles pages.



Customisation

OCTOBRE 2020

Attributs de style

Ionic met à disposition un ensemble d'attributs qui peuvent être utilisés sur n'importe quel élément pour de modifier du texte, le centrer ou encore gérer les marges. À la différence de Bootstrap où on fait usage de classes css (row, col,...), ici on utilisera plutôt des attributs.

```
<ion-content padding>
  <h2 class="ion-text-center">Réchercher un étudiant</h2>
</ion-content>
```

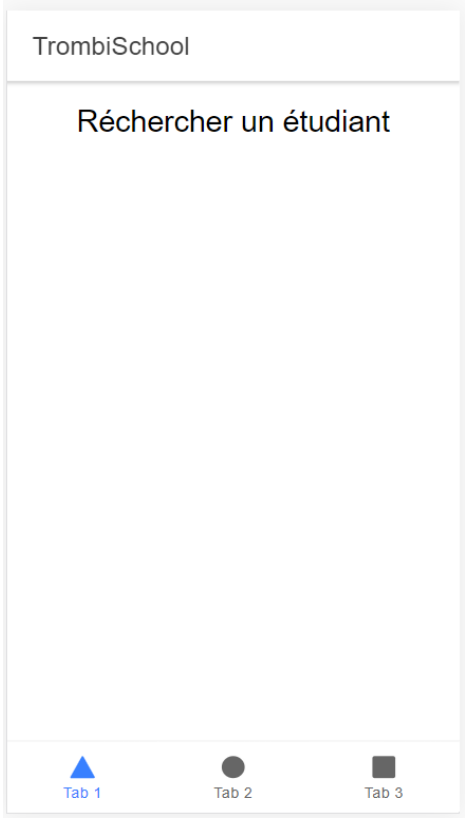
Class	Style Rule	Description
`.ion-text-left`	`text-align: left`	The inline contents are aligned to the left edge of the line box.
`.ion-text-right`	`text-align: right`	The inline contents are aligned to the right edge of the line box.
`.ion-text-start`	`text-align: start`	The same as `text-left` if direction is left-to-right and `text-right` if direction is right-to-left.
`.ion-text-end`	`text-align: end`	The same as `text-right` if direction is left-to-right and `text-left` if direction is right-to-left.
`.ion-text-center`	`text-align: center`	The inline contents are centered within the line box.

Une liste beaucoup plus exhaustive se trouve dans la documentation, qui est extrêmement bien faite
https://ionicframework.com/docs/theming/css-utilities/

On va pouvoir utiliser ces attributs directement dans nos pages. Centrons par exemple le h2 de la page d'accueil et justifions le contenu du texte qui le suit :

src/app\tab1\tab1.page.html

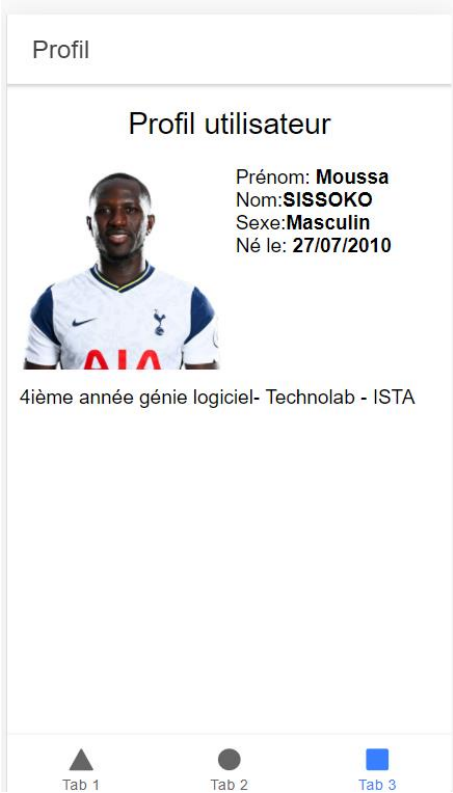
```
<ion-header>
  <ion-toolbar>
    <ion-title>
      TrombiSchool
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content padding>
  <h2 class="ion-text-center">Réchercher un étudiant</h2>
</ion-content>
```



Grille CSS responsive

Ionic propose également un système de grille css pour permettre une meilleur gestion de blocs de contenus. Il est assez similaire dans sa syntaxe à celui que propose **Bootstrap**.

```
<ion-content padding>
  <h2 class="ion-text-center">Profil utilisateur</h2>
  <ion-grid>
    <ion-row>
      <ion-col col-lg-1>
        <img src="assets/imgs/sissoko.png" />
      </ion-col>
      <ion-col>
        Prénom: <strong>Moussa</strong>
        <br>Nom:<strong>SISSOKO</strong>
        <br>Sexe:<strong>Masculin</strong>
        <br>Né le: <strong>27/07/2010</strong>
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col>
        4ième année génie logiciel- Technolab - ISTA
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
```



Pour plus de détails, merci de consulter la documentation correspondante :
https://ionicframework.com/docs/theming/responsive-grid/

Utilisation de SASS

Ionic est construit sur Sass (Syntactically Awesome Stylesheets), un langage de génération de feuilles de style, robuste et facile à prendre en main.

En fait si vous savez déjà définir une feuille de style, ce langage ne vous choquera pas trop. Grâce à cette technologie embarquée dans Ionic, nous allons non seulement pouvoir définir des styles génériques pour notre application, qui pourront être utilisé à plusieurs endroits différents, mais nous pourrons également changer les styles par défaut des attributs et composants Ionic.

La définition ou la redéfinition de style css dynamique se fait depuis le fichier ***src/theme/variables.scss*** :

```
/** Ionic CSS Variables **/
:root {
  /** primary **/
  --ion-color-primary: #3880ff;
  --ion-color-primary-rgb: 56, 128, 255;
  --ion-color-primary-contrast: #ffffff;
  --ion-color-primary-contrast-rgb: 255, 255, 255;
  --ion-color-primary-shade: #3171e0;
  --ion-color-primary-tint: #4c8dff;

  /** secondary **/
  --ion-color-secondary: #3dc2ff;
  --ion-color-secondary-rgb: 61, 194, 255;
  --ion-color-secondary-contrast: #ffffff;
  --ion-color-secondary-contrast-rgb: 255, 255, 255;
  --ion-color-secondary-shade: #36abe0;
  --ion-color-secondary-tint: #50c8ff;

  /** tertiary **/
  --ion-color-tertiary: #5260ff;
  --ion-color-tertiary-rgb: 82, 96, 255;
```

Vous pouvez ici effectuer des changements sur les valeurs par défaut des thèmes primaire, secondaire, ...Et ils s'appliqueront automatiquement à l'ensemble de vos composants.

```
<ion-header>
  <ion-toolbar color="primary">      You, seconds ago • Uncomm
    <ion-title>
      TrombiSchool
    </ion-title>
  </ion-toolbar>
</ion-header>
```

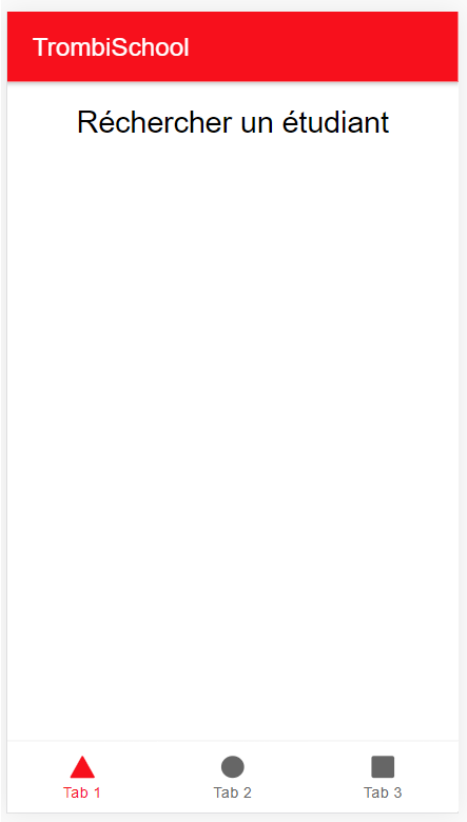
Vous pouvez également utiliser l'outil de génération de thème d'Ionic accessible ici: https://ionicframework.com/docs/theming/color-generator

Utilisation de SASS

Ici la barre de navigation aura comme couleur de fond (background) celle définie dans le fichier de variables scss et comme couleur de texte du blanc. Si vous voulez autre chose que du blanc, disons du jaune, vous devriez modifier votre style comme ceci :

src/theme/variables.scss

```
:root {
  /** primary **/
  --ion-color-primary: #f7101c;
  --ion-color-primary-rgb: 56, 128, 255;
  --ion-color-primary-contrast: #ffffff;
  --ion-color-primary-contrast-rgb: 255, 255, 255;
  --ion-color-primary-shade: #3171e0;
  --ion-color-primary-tint: #4c8dff;
```



Utilisation de SASS

Il est également possible d'appeler des variables définis dans ce fichier **src/theme/variables.scss** directement dans nos fichiers scss. Changeons par exemple la couleur du bouton présent dans l'onglet Profil :

src\app\tab3\tab3.page.scss

```
ion-content {
    ion-button{
        --background:var(--ion-color-secondary);
    }
}
```

qui est l'équivalent css de :

```
ion-content {
    ion-button{
        --background: #3dc2ff;
    }
}
```

Templates et création de nouvelles pages

OCTOBRE 2020

Racine de toutes les pages

Considérons le fichier **src/app/app.componment.html**, c'est à partir de ce fichier que seront "**générées**" toutes les autres pages.

src > app > app.component.html > ion-app

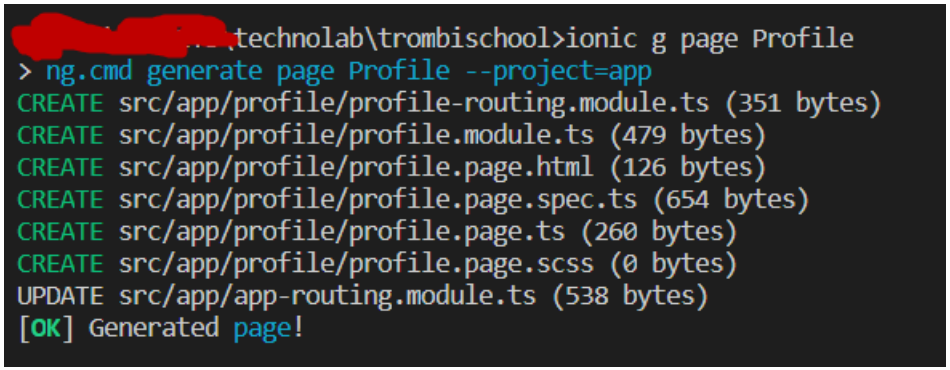
```
1   <ion-app>
2     <ion-router-outlet></ion-router-outlet>
3   </ion-app>
4
```

On y trouve le tag **ion-router-outlet** qui composant de routage. C'est dans cet élément que seront encapsuler les composants de l'application en fonction de l'url. Toute la logique de routage (notion que l'on abordera plus tard) est gérée dans le fichier **src/app/app-routing.module.ts**.

```
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@angular/router';

const routes: Routes = [
  {
    path: '',
    loadChildren: () => import('./tabs/tabs.module').then(m => m.TabsPageModule)
  }
];
@NgModule({
  imports: [
    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
  ],
  exports: [RouterModule]
})
export class AppRoutingModule {}
```

Création d'une nouvelle page

Pour créer une nouvelle page, il vous suffit de saisir la commande **ionic g page LeNomDeLaPage** :



"**g**" pour "**generate**".

Dans cet exemple, j'ai créé une nouvelle page qui va nous permettre d'afficher un profil utilisateur. Cette commande m'a automatiquement générer un certain nombre de fichiers dont le triplet : fichier .**ts + fichier .html + fichier .scss.**

Création d'une page de consultation des étudiants

Pour créer notre page, il vous saisir saisir la commande **ionic g page etudiants** :

src/app/etudiants/etudiants.page.ts

```
import { Component, OnInit } from '@angular/core';

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

  etudiants: { id: number, nom: string, prenom: string, sexe: string, telephone: string, email: string, formation: string }[] = [
    { "id": 1, "nom": "DIARRA", "prenom": "Fanta", "sexe": "F", "telephone": "78905467", "email": "diarra@gmail.com", "formation":
    { "id": 2, "nom": "SISSOKO", "prenom": "Kadidia", "sexe": "F", "telephone": "78905467", "email": "sissoko@gmail.com", "formatio
    { "id": 3, "nom": "DRAME", "prenom": "Harouna", "sexe": "M", "telephone": "78905467", "email": "drame@gmail.com", "formation":
  ];
  constructor() { }

  getEtudiantById(id) {
    return this.etudiants.filter(e => e.id = id);
  }
  ngOnInit() {
  }

}
```

Nous avons simplement défini une variable nommée etudiants, de type tableau et contenant, comme vous vous en doutez, des étudiants.

Création d'une page de consultation des étudiants

Puis dans le fichier html, apportons les modifications suivantes dans le composant ion-content:
src/app/etudiants/etudiants.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>Liste des étudiants</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-card *ngFor="let e of etudiants">
    <ion-card-header>
      <ion-card-title>{{e.nom}} {{e.prenom}}</ion-card-title>
    </ion-card-header>
    <ion-card-content>
      <p><strong>{{e.formation}}</strong></p>
      <p>Téléphone: {{e.telephone}}</p>
      <p>Email: {{e.email}}</p>
    </ion-card-content>
  </ion-card>

</ion-content>
```

Pour l'instant on affiche simplement dans la vue html (src/app/etudiants/etudiants.page.html), les étudiants que nous avons définie dans le controleur de la page (src/app/etudiants/etudiants.page.ts).

Mais nous souhaitons aussi pouvoir cliquer sur un étudiant pour en afficher les détails. Pour cela, nous allons créer une nouvelle page que nous nommerons simplement Etudiant, qui permettra d'effectuer ces actions.

Création d'une page de consultation des détails d'un étudiant

Depuis votre invite de commandes, faites ceci :
$ ionic g page etudiant

Notre page a correctement été créé, et la page de routage a automatiquement été mise à jour. Modifier le fichier de routage comme ceci :

src/app/app-routing.module.ts

```
const routes: Routes = [
  {
    path: '',
    loadChildren: () => import('./tabs/tabs.module').then(m => m.TabsPageModule)
  },
  {
    path: 'profile',
    loadChildren: () => import('./profile/profile.module').then(m => m.ProfilePageModule)
  },
  {
    path: 'etudiants',
    loadChildren: () => import('./etudiants/etudiants.module').then(m => m.EtudiantsPageModule)
  },
  {
    path: 'etudiants/:id',
    loadChildren: () => import('./etudiant/etudiant.module').then(m => m.EtudiantPageModule)
  }
];
@NgModule({
  imports: [
    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
  ]
```

Création d'une page de consultation des détails d'un étudiant

Dans cette nouvelle configuration, on dit simplement que :

L'url http://localhost:8100/etudiants nous redirigera vers la page des etudiants
L'url http://localhost:8100/etudiants /UnID renverra vers la page d'un étudiant dont l'identifiant est UnID.

Maintenant que nous avons créé notre page etudiant, nous allons pouvoir faire en sorte qu'un clic sur un étudiant sur la liste nous renvoie vers le détail de celui-ci.

Pour cela, modifions à nouveau le fichier src/app/etudiants/ etudiants.page.html en ajoutant les **directives [routerLink]** et routerDirection comme ceci :

src/app/etudiants/etudiants.page.html

```
<ion-content>
  <ion-card *ngFor="let e of etudiants" [routerLink]="'/etudiants/' + e.id" routerDirection="forward">
    <ion-card-header>
      <ion-card-title>{{e.nom}} {{e.prenom}}</ion-card-title>
    </ion-card-header>
    <ion-card-content>
      <p><strong>{{e.formation}}</strong></p>
      <p>Téléphone: {{e.telephone}}</p>
      <p>Email: {{e.email}}</p>
    </ion-card-content>
  </ion-card>
```

Nous avons donc ajouter un lien de routage grâce à la directive **routerLink** avec la valeur **/etudiants/ID**. Voyez cela comme l'équivalent d'un href. La directive **routerDirection** permet de jouer sur l'animation du changement de page.

Création d'une page de consultation des détails d'un étudiant

Editons ensuite notre page etudiant, pour afficher les détails en fonction d'un identifiant :

src/app/etudiant/etudiant.page.ts

On importe tout d'abord la class **ActivatedRoute**, qui va nous permettre de récupérer l'Identifiant de l'étudiant à partir de l'url.

```
// On importe cette classe
import { ActivatedRoute } from '@angular/router';
```

On déclare ensuite d'abord une interface qui va nous permettre de caractériser notre etudiant : un objet possédant un identifiant (id) un nom, du prenom etc...

```
// Cette interface permet de caractériser un objet etudiant
interface Etudiant {
  id: number;
  nom: string;
  prenom: string;
  sexe: string;
  telephone: string;
  email: string;
  formation: string;
}
```

Création d'une page de consultation des détails d'un étudiant

Dans la classe EtudiantPage, on redéclare à nouveau la variables notes contenant un tableau de notes (le même que celui en page liste des étudiants). Plus tard, lorsque nous aborderons les notions de services et de persistance de données, nous n'aurons plus besoin doublonner cette variable.

On déclare également un objet etudiant du type de l'interface Etudiant.

```
export class EtudiantPage implements OnInit {

  etudiants: { id: number, nom: string, prenom: string, sexe: string, te
    { "id": 1, "nom": "DIARRA", "prenom": "Fanta", "sexe": "F", "telephon
    { "id": 2, "nom": "SISSOKO", "prenom": "Kadidia", "sexe": "F", "telep
    { "id": 3, "nom": "DRAME", "prenom": "Harouna", "sexe": "M", "telepho
  ];
  etudiant: Etudiant;

  constructor() { }
```

Dans le construteur de la classe, on initialise l'étudiant à vide. Cette déclaration aura surtout du sens lorsque l'on ira chercher des données de manière asynchrone (via un API ou depuis la base de données du téléphone).

```
constructor(private route: ActivatedRoute) {
  // Initialisation d'un etudiant à vide
  this.etudiant = {
    id: 0,
    nom: '',
    prenom: '',
    sexe: '',
    telephone: '',
    email: '',
    formation: ''
  };
}
```

Création d'une page de consultation des détails d'un étudiant

La méthode **ngOnInit** est appelé après l'initialisation de la page. C'est donc dans cette méthode là que l'on récupéra l'identifiant de l'étudiant, avant de récupérer tous les détails de la note grâce la méthode **getEtudiantById**.

```
ngOnInit() {
  // On récupère l'identifiant de la
  let etudId = this.route.snapshot.paramMap.get('id');
  this.etudiant = this.getEtudiantById(etudId);
}
/**
 ** Renvoie un etudiant en fonction de son identifiant
 ** @param id : identifiant de l'étudiant
 **/
getEtudiantById(id) {
  // La méthode find va rerchercher le premier dont l'identifiant est égal à id
  return this.etudiants.find(function (etudiant) {
    return etudiant.id == parseInt(id);
  });
}
```

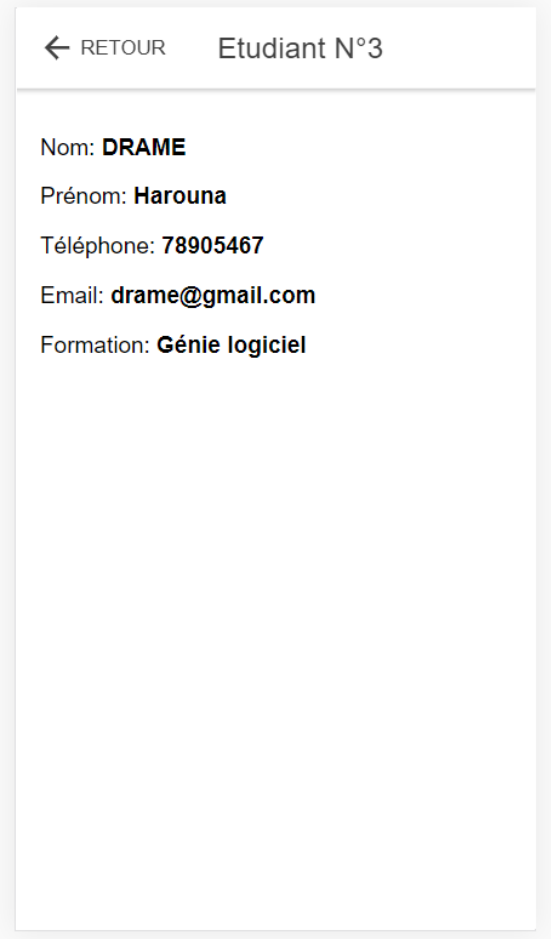
Création d'une page de consultation des détails d'un étudiant

Il ne reste plus qu'à modifier le fichier html, pour afficher les données de l'étudiant :

src/app/etudiant/etudiant.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button defaultHref="etudiants" text="Retour"></ion-back-button>
    </ion-buttons>
    <ion-title>Etudiant N°{{etudiant.id}}</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content class="ion-padding">
  <p>Nom: <strong>{{etudiant.nom}}</strong></p>
  <p>Prénom: <strong>{{etudiant.prenom}}</strong></p>
  <p>Téléphone: <strong>{{etudiant.telephone}}</strong></p>
  <p>Email: <strong>{{etudiant.email}}</strong></p>
  <p>Formation:<strong> {{etudiant.formation}}</strong></p>
</ion-content>
```

On obtient le résulta suivant :



Bravo ! Vous pouvez à présent afficher une liste d'éléments et les afficher de manière détaillée.

Dans la suite nous verrons comment habiller un peu plus notre application grâce aux nombreux composants que proposent Ionic, mais aussi créer, modifier ou supprimer nos étudiants grâce aux **Services** et à Ionic **Storage**.