ADAPTATION DE L'INTERFACE A L'ENVIRONNEMENT POLYTECH CLASSROOM

Réalisé par :
Meriem Chebaane
Ameni Haouel
Insaf Jebara
Nidhal Fliss

Rapport Final —— Master 2 IAM

Table des matières

Table de figures

1. Introduction générale

De nos jours, l'amélioration de la technologie des nouveaux Framework existants est accompagnée par une panoplie de changements affectant la production d'Interfaces Homme Machine (IHM) multi cibles. En effet, une IHM multi cibles a plusieurs capacités d'adaptation tout en respectant l'utilisabilité. Une cible se définit par le triplet (utilisateur, plate-forme, environnement).

C'est alors dans le but de bien étudier ce domaine et comparer les technologies existantes qu'on a créé une application de cours en ligne. Chaque visiteur peut à la fois être un lecteur ou un rédacteur. Les cours peuvent être réalisés aussi bien par des membres ou par l'équipe du site. Initialement orienté autour des tutoriels de programmation informatique, le projet est basé sur l'adaptation des interfaces à l'environnement.

Après une étude approfondie des contextes d'usage des technologies disponibles, nous entamons dans ce rapport la présentation du travail réalisé. Tout d'abord, nous allons présenter l'environnement de travail que nous avons utilisé pour chaque technologie. Ensuite, nous allons présenter la méthode de développement, les pratiques adoptées pour appliquer les divers méthodes et techniques d'adaptation pour passer ensuite à l'étape de test dont nous allons présenter quelques interfaces de l'application. Enfin, nous terminons par une conclusion comparative.

2. Tutoriel Ionic

Introduction

Ionic est un framework open source qui permet de développer des applications hybrides, c'est-à-dire des applications web pouvant être exécutées sur différentes plateformes à savoir Android, iOS et windows phone à travers des webview. Ces applications ne sont pas développées en langage natif mais plutôt en faisant appel à Angular JS pour le front-end qui se base essentiellement sur les langages web HTML5, CSS3 et JavaScript. Pour la partie construction en natif, Ionic se base sur Apache Cordova qui permet de déployer l'application sur la plateforme adéquate.

Le choix de travailler avec Ionic vient du fait qu'il utilise des langages web et garde presque le même design qu'une application native avec un plus, c'est qu'il est plus facile et plus rapide à développer et nous fait gagner du temps quant au développement sur les autres plateformes. Ionic permet notamment d'accéder aux capteurs du smartphone tel que la caméra ce qui lui donne un point bonus par rapport aux autres framework de développement web.

Avant de commencer

Avant de se plonger dans le développement il faudra effectuer quelques installations pour faire fonctionner l'outil.

Ionic s'appuie sur la plateforme Node JS et plus précisément NPM (Node Package Manager) pour installer les nouveaux modules développés par la communauté et pour gérer les dépendances entre les modules donc n'oubliez pas de l'installer en premier lieu.

Figure 1 : installation environnement - Ionic

Cette commande permet à la fois d'installer globalement Ionic et cordova. Elle nous donnera par la suite accès à la commande ionic.

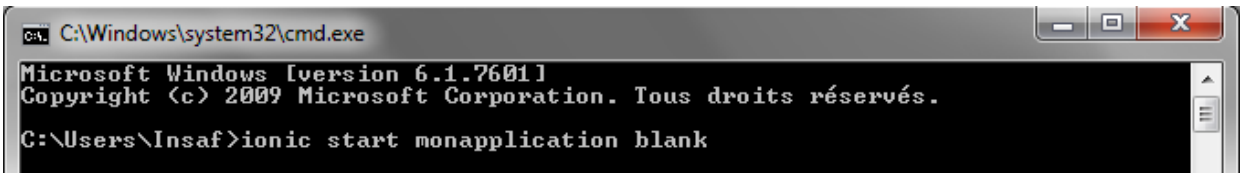


Figure 2 : Création de projet - Ionic

Cette commande permet de créer un projet, intitulé monapplication.

Le paramètre blank c'est pour dire que c'est une application à partir du scratch.

Structure du projet

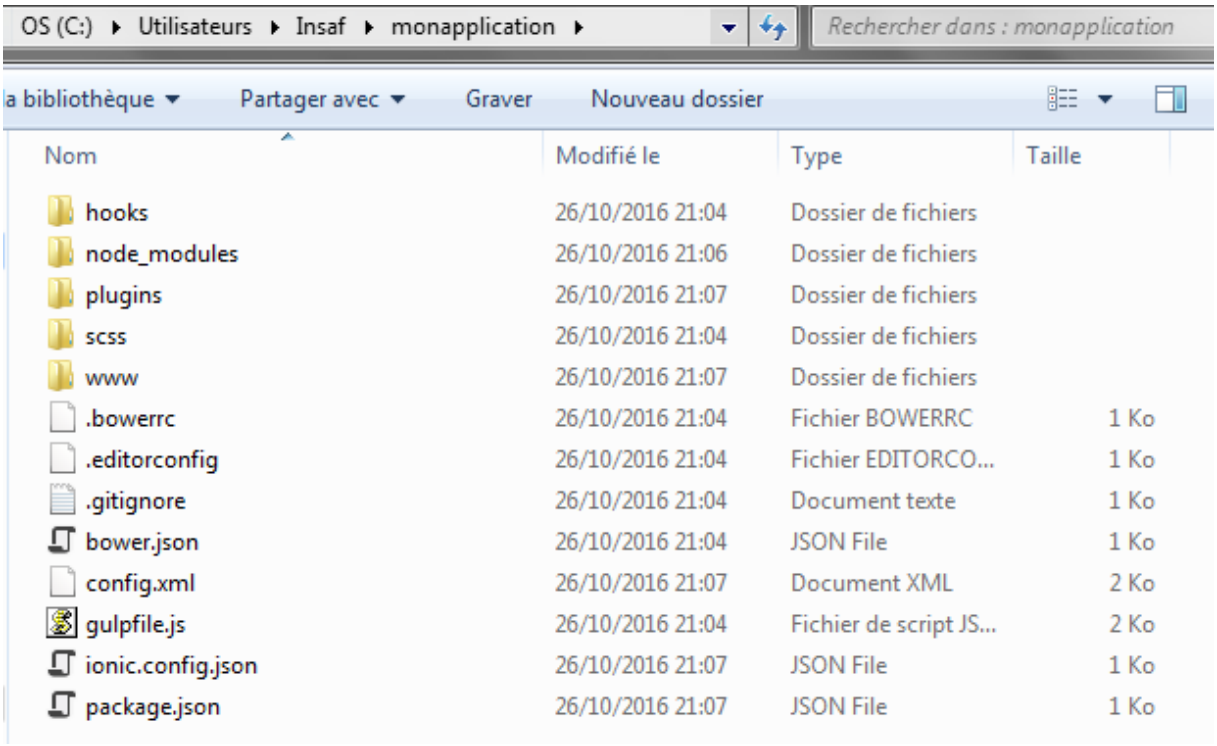


Figure 3 : Structure du projet - Ionic

Ceci est le contenu du dossier de note application créée.

Le www :

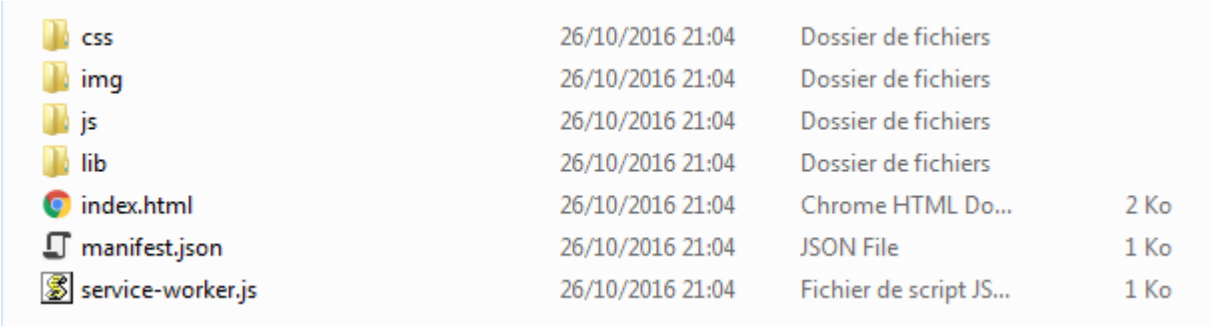


Figure 4 : Contenu du dossier www - Ionic

Plugins : les modules installés

Et des fichiers de configuration comme par exemple le bower.json pour gérer les dépendances ou gulpfile.js qui s'occupe de gérer les tâches à réaliser, leurs options, leurs sources et destination ; en d'autres mots c'est le chef d'orchestre.

Routing

Ionic s'appuie fortement sur le UI-Router pour la navigation qui est utilisé pour naviguer entre les états d'affichage – par état d'affichage on veut dire des écrans composés sous forme de fichiers de modèle.

Chaque state représente une nouvelle vue dans l'application.

Ci-dessous une vue globale du fichier app.js

```
// Ionic Starter App

// angular.module is a global place for creating, registering and retrieving Angular modules
// 'starter' is the name of this angular module example (also set in a <body> attribute in index.html)
// the 2nd parameter is an array of 'requires'
// 'starter.controllers' is found in controllers.js
angular.module('starter', ['ionic', 'starter.controllers'])
angular.module('starter', ['ionic', 'starter.controllers', 'starter.services'])

.run(function($ionicPlatform) {
  $ionicPlatform.ready(function() {
    // Hide the accessory bar by default (remove this to show the accessory bar above the keyboard
    // for form inputs)
    if (window.cordova && window.cordova.plugins.Keyboard) {
      cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);
      cordova.plugins.Keyboard.disableScroll(true);

    }
    if (window.StatusBar) {
      // org.apache.cordova.statusbar required
      StatusBar.styleDefault();
    }
  });
})

.config(function($stateProvider, $urlRouterProvider) {
  $stateProvider

    .state('app', {
    url: '/app',    -> url auquel on peut accéder à travers les href
    abstract: true,
    templateUrl: 'templates/menu.html',  -> lien à la vue en html
    controller: 'AppCtrl'    -> controlleur pour cette vue
  })

  .state('app.courses', {
    url: '/courses',
    views: {
      'menuContent': {
        templateUrl: 'templates/courses.html'
      }
    }
  })
```

Figure 5 : app.js - Ionic

Le UI-router permet d'injecter une template à travers la balise <ion-nav-view> qui est contenue dans le fichier index.html comme montré ci-dessous.

```
    <!-- your app's js -->
    <script src="js/app.js"></script>
    <script src="js/controllers.js"></script>
    <script src="js/services.js"></script>
  </head>
  <body ng-app="starter">
      <ion-nav-view></ion-nav-view>
  </body>
</html>
```

Figure 6 : index.html - Ionic

Remarque : Tout fichier *.js créé sous le répertoire js est déclaré dans le fichier index.html

Contrôleurs

Les contrôleurs permettent de définir et d'implémenter les fonctions utilisées dans les vues.

Ci-dessous un exemple du contrôleur agissant sur la partie login. La fonction submit étant définie dans le « ng-click » du bouton « Log in »

```
.controller('LoginCtrl', function($scope, $state) {
    $scope.submit = function(){
        var $user = $scope.username;
        var $pwd = $scope.password;
        if ($scope.username == 'insaf' && $scope.password == 'insaf')
        {
            $state.go('app.upload');
            $scope.closeLogin();
        } else {
            alert("Wrong credentials !");
        }
    }
})
```

Figure 7 : Controller.js - Ionic

LoginCtrl est défini en tant que « ng-controller » dans le fichier login.html (page de vue login)

$State.go permet d'aller vers le state (la page) « app.upload »

Services

Pour accéder à la caméra du smartphone, nous sommes menés à définir des services qui nous permettront de profiter de ses fonctions comme prendre des photos ou filmer des vidéos.

Pour notre projet, nous avons besoin de charger des cours dans l'application donc ça sera soit à travers une capture directe soit à partir des dossiers du mobile.

```
angular.module('starter.services', [])

.factory('Camera', function($q) {

   return {
      getVideo: function(options) {
         var q = $q.defer();

         navigator.camera.getVideo(function(result) {
            q.resolve(result);
         }, function(err) {
            q.reject(err);
         }, options);

         return q.promise;
      }
   }

var deferred = $q.defer();
var promise = deferred.promise;

promise.success = function(fn) {
    promise.then(fn);
    return promise;
}
promise.error = function(fn) {
    promise.then(null, fn);
    return promise;
}
});
```

Figure 8 : Services.js - Ionic

Première interface

Dans cette première interface nous allons ajouter un side menu, un bouton de login à droite, le logo de notre application à gauche et une petite conclusion sur les membres développeurs du projet.

Figure 9 : 1ere interface mobile - Ionic

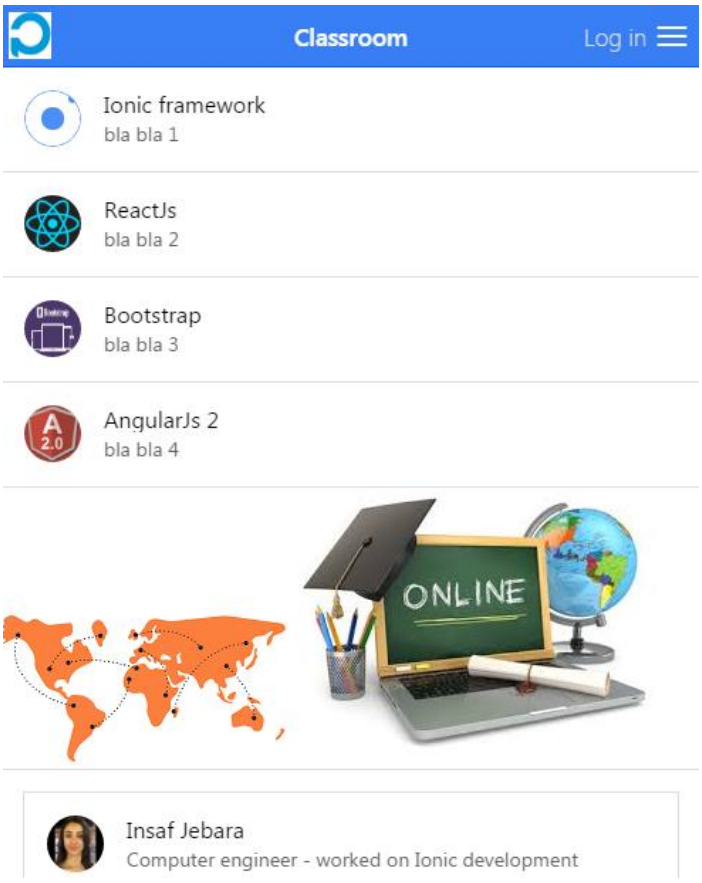


Figure 10 : 1ere interface tablette - Ionic

Pour aboutir à cette interface, nous avons commencé par mettre en place notre template.

<ion-side-menus> nous a permis de créer la barre de navigation.

<ion-side-menu-content> a permis d'ajouter l'icône à gauche, le bouton Log in et le bouton de menu à droite dans le <ion-nav-bar>

<ion-side-menu side="right"> permet de spécifier le contenu du side-menu, dans notre cas nous avons une liste Home (cette page d'accueil) et Courses (Liste des cours) définie par <ion-list> et <ion-item>

Le contenu de la page est défini dans une <ion-view>. La partie supérieure, présentant les technologies utilisées, est développées à partir de balises simples de lien <a> et <img class="item item-avatar"> pour les images. La partie inférieure, celle de la représentation des développeurs de l'application, est développées à partir d'une liste : <ion-list> et des <ion-avatar> pour chaque item.

```
<ion-side-menus enable-menu-with-back-views="false">
        <ion-side-menu-content>
            <ion-nav-bar class="bar-positive">
                .
                .
                .
            <ion-nav-view name="menuContent">
            </ion-nav-view>
        </ion-side-menu-content>
            <ion-side-menu side="right">
                <ion-header-bar class="bar-pos
                    <h3 class="title">Menu</h3
                </ion-header-bar>
                <ion-content class="right-menu'
                    <ion-list>
                        <ion-item menu-close h
                        <ion-item menu-close h
                    </ion-list>
                </ion-content>
            </ion-side-menu>
    </ion-side-menus>
```

Figure 11 : menu.html - Ionic

Style

Bien que Ionic soit orienté web, il permet de personnaliser les interfaces de l'application pour qu'elles ne ressemblent pas à ce qui est proposé par défaut. Pour ce faire, il est possible d'utiliser SASS en plus du CSS pour le coté design.

Une installation est demandée pour ceci :

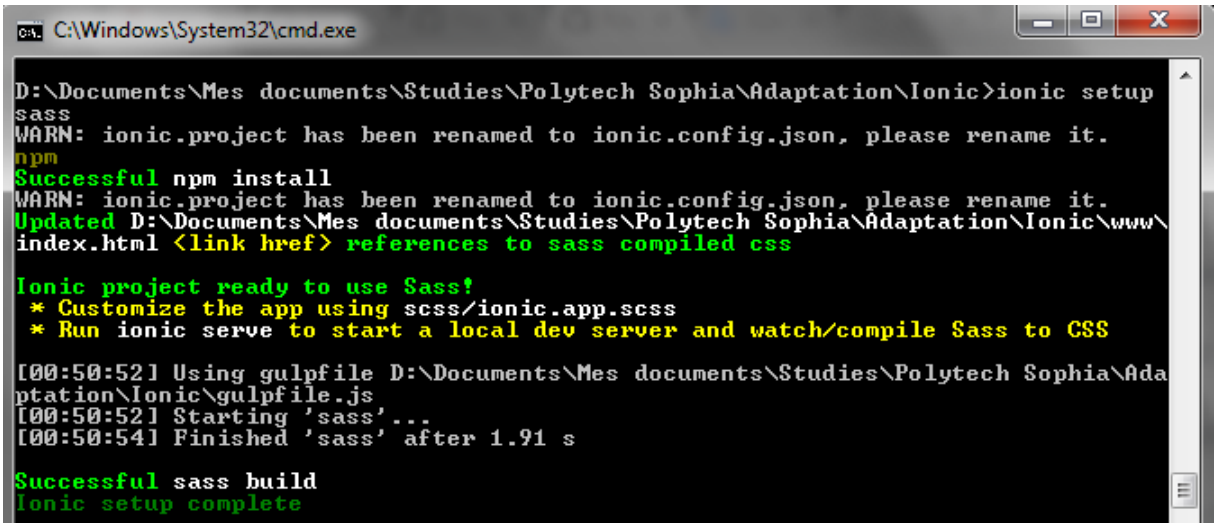


Figure 12 : Installation SASS – Ionic

Plugins installés

Pour cette application, nous avons installé principalement les plugins relatifs à l'accès à la caméra :

$ionic plugin add cordova-plugin-camera

$bower install ngCordova
$cordova plugin add org.apache.cordova.file-transfer

Il est temps de visualiser l'application

Nous avons fait le tour des principales fonctionnalités de l'Ionic ainsi qu'un tutoriel global sur les vues, modèles et contrôleurs. Il ne nous reste plus qu'à compiler et lancer l'application.

Nous avons déjà mentionné que Ionic est multiplateformes donc pour le compiler, nous devons avoir les plugins installés.

Vous avez le choix, soit de faire des installations par invite de commande sinon vous pouvez utiliser l'outil Ionic Lab qui facilite l'installation de plugins pour l'application et effectue le « serve » en background.

Invite de commande :

Pour android :

$ionic platform add android

$ionic build android → Pour avoir le .apk qui sera localisé sous platforms/android/build/outputs/apk dans le répertoire du projet

$ionic serve → sur browser

$ionic run android → sur un terminal mobile

Pour iOS:

$ionic platform add ios

$ionic build ios

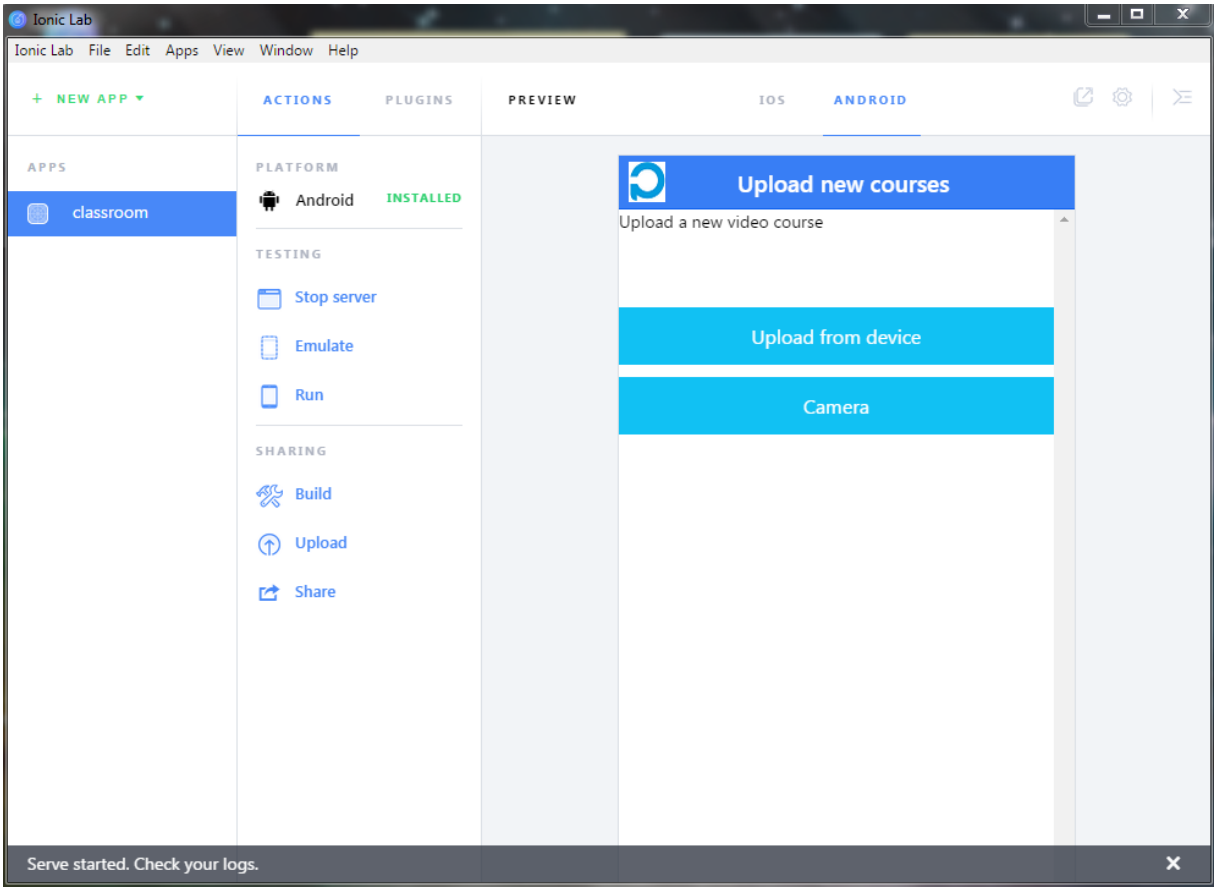
$ionic emulate ios

Pour Windows Phone 10:

$ionic platform add windows@https://aka.ms/cordova-win10

$ionic run windows

Ionic Lab



Test Ionic

Pour tester l'adaptation de l'application aux contraintes imposées au début du projet, nous avons essayé d'installer l'application sur plusieurs dispositifs de taille et résolution différentes pour voir le comportement de l'application.

3. Tutoriel Bootstrap

Bootstrap est une collection d'outils utile à la création de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres ´éléments interactifs, ainsi que des extensions JavaScript en option. C'est l'un des projets les plus populaires sur la plateforme de gestion de développement Git Hub. Les développeurs qui utilisent Bootstrap pour la création de leur site web choisissent les éléments qu'ils veulent utiliser avec la certitude qu'ils ne seront pas incompatibles entre eux. En fait, c'est comme un puzzle. Sauf que dans ce puzzle, chaque pièce s'imbrique parfaitement dans les autres, quelle qu'elle soit. Et grâce à la magie de l'open-source, Bootstrap s'améliore en permanence : de nouvelles fonctions absolument géniales ont été ajoutées comme le 100% mobile responsive ou la très large sélection de plugins jQuery.

Une des principales forces de ce framework c'est sa structure en grille. C'est cette structure qui permet l'adaptation entre plusieurs périphériques : PC, tablettes, smartphone ...

Installation du Framework

On peut installer Bootsrap de deux manières :

- En récupérant les fichiers archivés à partir du site officiel de bootsrap
- En utilisant les packages bower : sous la racine du projet :
  bower install --save boostrap

Dans mon cas j'ai choisi la première manière mais j'ai dû télécharger jQuery et l'ajouter aux sources bootsrap ce qui n'est pas nécessaire en utilisant la deuxième méthode.

Structure du projet

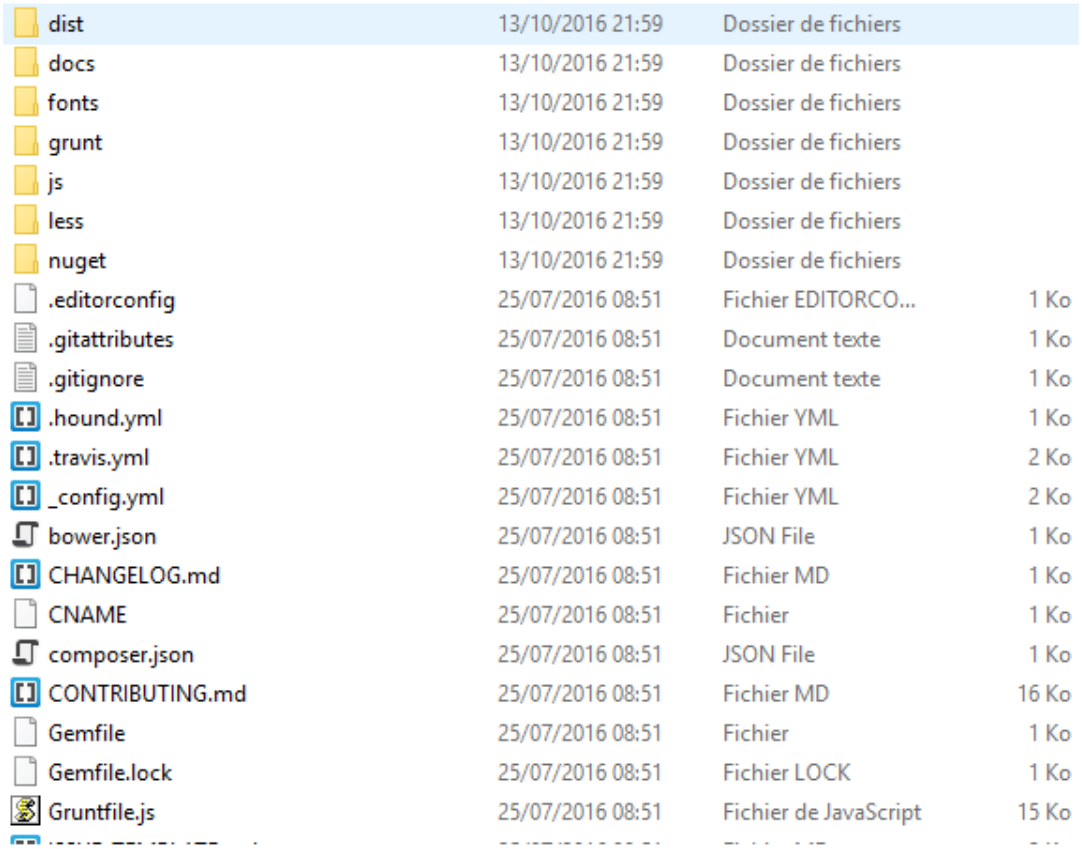


Figure 13:Structure du projet

Utilisation du Framework

Inclure le fichier Css de bootsrap dans la balise head de chaque page développée.

```
<link href="css/bootstrap.min.css" type="text/css" rel="stylesheet">
```

Figure 14: Référencer Bootsrap dans la page html

Ajouter entre les balises body juste avant la balise de fermeture l'import de script suivant permettant JQuery ainsi que les composants javascript de bootsrap de fonctionner :

```
<script src="js/jquery.min.js" type="text/javascript"></script>

<script src="js/bootstrap.min.js" type="text/javascript"></script>
```

Figure 15: Insertion du Jquery dans la page HTML

Les librairies sont maintenant disponibles donc on peut commencer à développer.

Implémentation du menu de navigation

L'implémentation d'un menu de navigation se fait très rapidement et suit quasiment la même démarche pour beaucoup d'applications web. Juste après la balise ouvrante <body>, on utilise le tag pour définir notre menu de navigation. Bootstrap étant un framework intuitif, la classe CSS pour construire une barre de navigation s'appellent navbar. Il faut aussi savoir que souvent en Bootstrap, pour mettre en forme un composant, on indique sa classe générale avant d'indiquer sa classe particulière. Nous avons utilisé la classe générale btn, pour Donc pour la barre de navigation, la balise sera de la forme. Nous allons donc utiliser la barre de navigation par défaut de Bootstrap.

Sous la balise nav de menu de navigation on constate la présence d'une div avec la classe container afin de garantir que les composant soit responsive avec une taille fixe. Cette classe permet d'avoir une page plus agronomique en séparant les marges du logo et les liens des navigations.

```
<div class="loader-backdrop">
    <div class="loader">
        <div class="ball-1"></div>
        <div class="ball-2"></div>
    </div>
</div>

<header>
    <nav class="navbar navbar-default">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#main-navigation" aria-expanded="false">
                    <span class="sr-only">Toggle Menu</span>
                    <span>Menu</span>
                </button>
                <a class="navbar-brand" href="index-2.html"><img src="images/Logo%20Univ.%20Nice%20Sophia-Antipolis.jpg" alt=""></a> <!-- Replace with your Logo -->
            </div>
```

Figure 16: Code de la barre de navigation

On constate que le header a pour classe navbar-header , que nous avons en dessus une base button avec trois attributs :

- class="navbar-toggle collapsed" pour basculer en mode navigation
- data-toggle="collapse pour plier / déplier
- data-target="#main-navigation" qui possède la fonctionnalité d'un lien href afin de rédériger en cliquant vers la div du menu principal.

Dans la balise button on a mis la balise   ayant pour classe « sr-only » , c'est une classe destinée aux Screan Reader.

Dernièrement la balise nav-brand pour le logo inséré dans l'entête du site web.

L'entête est terminé on peut passer maintenant au développement des liens du menu de navigation. Les liens seront entre les balis et ayant pour id= « main-navigation »

```
<div class="collapse navbar-collapse" id="main-navigation">
    <ul class="nav navbar-nav navbar-right">
        <li class="active"><a href="index-2.html">Accueil</a></li>

        <li><a href="#">Pages</a>
            <ul class="sub-menu">
                <li><a href="#"> Cours <i class="fa fa-angle-right pull-right"></i></a>
                    <ul class="sub-menu">
                        <li><a href="course-list.html">Liste des Cours</a></li>
                        <li><a href="course-add.html">Ajouter un Cours</a></li>
                    </ul>
                </li>
            </ul>
        </li>
        <li><a href="#">Contact</a>
            <ul class="sub-menu">
                <li><a href="contact.html">Nous Contacter</a></li>
            </ul>
        </li>
    </ul>
</div>
```

Figure 17: Code du Menu et l'insertion du logos

Cette div a comme classe navbar-collapase pour mentionner qu'elle peut être pliée et dépliée.

En dessus on trouve une liste non ordonnée <ul> afin de contenir la liste de menu principale ayant la classe navbar-right pour être en extrême droite dans le cas de notre site sinon il se positionneront par défaut à gauche.

La classe sub-menu afin d'ajouter les colonnes du menu défilant en cliquant dessus pour enfin insérer les différentes parties de menus avec la balise

Maintenant le menu est terminé on peut le tester en l'affichant sur le navigateur et on obtient le résultat suivant :

Figure 18: Aperçu de la barre de navigation (navigateur)

Le menu vu à travers l'émulateur de firefox :

Figure 19: Aperçu du la barre de navigation (émulateur Firefox)

Implémentation de l'interface d'identification

```
<div class="mb-100">
    <div class="container">
        <div class="row">
            <div class="col-sm-5 mt-60">
                <div class="newsletter">
                    <h4 class="newsletter-heading">S'enregistrer</h4>
                    <form>
                        <div class="form-group">
                            <input class="form-control" placeholder="Nom" id="nd">
                        </div>
                        <div class="form-group">
                            <input class="form-control" placeholder="Email">
                        </div>
                        <div class="form-group">
                            <button type="submit" class="btn btn-primary btn-block" onclick="cliquer()">Enregistrer</button>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
```

Figure 20: Code de l'interface identification

On constate la réutilisation de la mémé classe container afin de garantir la séparation entre les différents composants puis l'utilisation de la classe News letter afin d'avoir la forme d'une enveloppe comme on voit si dessus :

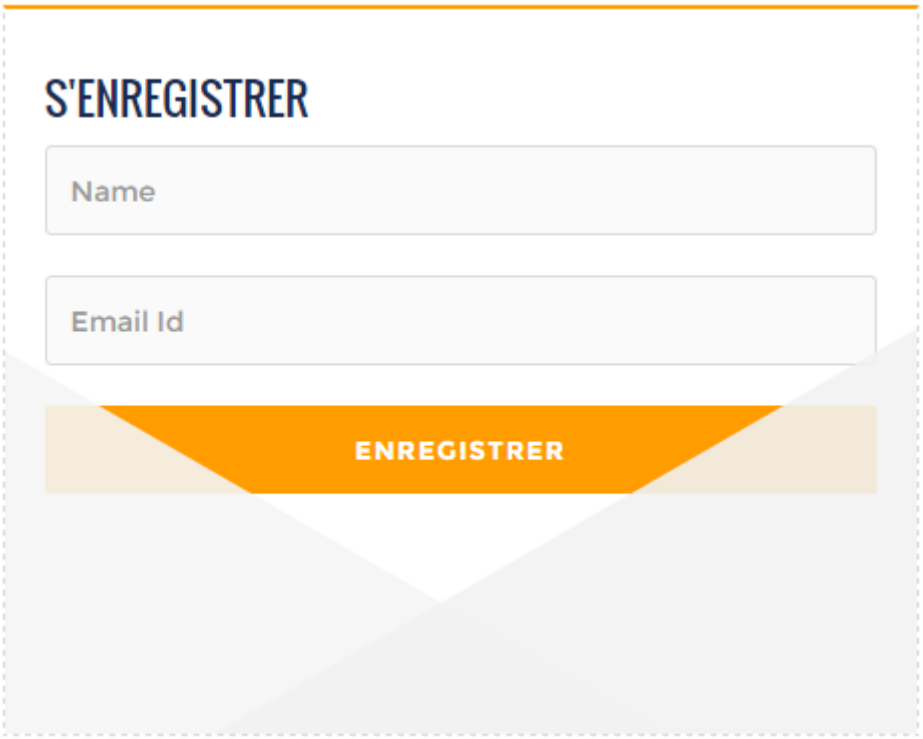


Figure 21:Apeçu de l'interface d'identification

Implémentation de la galerie de photos

On utilise pour l'implémentation de la galerie de photos la classe container-fluid afin que les photos dispose de conteneur de largeur, couvrant toute la largeur de notre fenêtre.

```
<div class="gallery">
    <div class="container-fluid">
        <div class="row">
            <div class="gallery-overlay">
                <h2 class="gallery-heading">Notre Vie à Polytech</h2>
            </div>
            <div class="col-sm-3 no-gutter">
                <div class="img-box">
                    <a href="images/gallerie1.jpg" data-gal="prettyPhoto[galleryName]" class="img-zoom"><i class="fa fa-search"></i></a>
                    <img src="images/gallerie1.jpg" alt="" class="img-responsive">
                </div>
            </div>
            <div class="col-sm-3 no-gutter">
                <div class="img-box">
                    <a href="images/gallerie2.jpg" data-gal="prettyPhoto[galleryName]" class="img-zoom"><i class="fa fa-search"></i></a>
                    <img src="images/gallerie2.jpg" alt="" class="img-responsive">
                </div>
                <div class="img-box">
                    <a href="images/gallerie3.jpg" data-gal="prettyPhoto[galleryName]" class="img-zoom"><i class="fa fa-search"></i></a>
                    <img src="images/gallerie3.jpg" alt="" class="img-responsive">
                </div>
            </div>
            <div class="col-sm-3 no-gutter">
                <div class="img-box">
                    <a href="images/gallerie4.JPG" data-gal="prettyPhoto[galleryName]" class="img-zoom"><i class="fa fa-search"></i></a>
                    <img src="images/gallerie4.JPG" alt="" class="img-responsive">
                </div>
                <div class="img-box">
                    <a href="images/galerie5.jpg" data-gal="prettyPhoto[galleryName]" class="img-zoom"><i class="fa fa-search"></i></a>
                    <img src="images/galerie5.jpg" alt="" class="img-responsive">
                </div>
            </div>
            <div class="col-sm-3 no-gutter">
                <div class="img-box">
                    <a href="images/galerie6.jpg" data-gal="prettyPhoto[galleryName]" class="img-zoom"><i class="fa fa-search"></i></a>
                    <img src="images/galerie6.jpg" alt="" class="img-responsive">
                </div>
            </div>
        </div>
    </div>
</div>
```

Figure 22: Code de la galerie des photos

En testant on s'aperçoit de la difficulté de disposition des photos

Figure 23: Aperçu de la galerie

Implémentation des tableaux

En développant l'application il était difficile de personnaliser la page d'accueil et de rendre le design plus beau donc en cliquant au sous menu mes cours j'ai essayé de faire apparaitre l'un des composants de bootsrap sans l'utilisation des différentes classes.

Cela m'a permis de voir la grande limite de bootsrap avec le grand tableau mal organisé pas trop beau à voir.

J'ai juste utilisé la classe table-bordered afin de préciser la limite de tableau et remplir la page avec plein de lignes et de colonnes (texte + image)

```
<body>
    <h1 >Mes technologies d'adaptation</h1>
    <div class="table table-bordered">
    <table  class="text-justify">
   <thead>
    <th>Nom de la technologies</th>
       <th> Definition</th>
       <th>Avantages</th>
       <th>Inconvénients</th>
       <th>Image</th>
    </thead>
        <tbody>
            <tr>
                <td>Bootsrap</td>
              <td>
```

Figure 24: Code d'implémentation du tableau

En testant le résultat était décevant, tout en blanc et noir et mal organisé :

MES TECHNOLOGIES D'ADAPTATION

NOM DE LA TECHNOLOGIES	DEFINITION	AVANTAGES	INCONVÉNIENTS	IMAGE
Bootsrap	Bootstrap est une collection d'outils utile à la création de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option. C'est l'un des projets les plus populaires sur la plate-forme de gestion de développement GitHub.Les développeurs qui utilisent Bootstrap pour la création de leur site web choisissent les éléments qu'ils veulent utiliser avec la certitude qu'ils ne seront pas incompatibles entre eux. En fait, c'est comme un puzzle. Sauf que dans ce puzzle, chaque pièce s'imbrique parfaitement dans les autres, quelle qu'elle soit. Et grâce à la magie de l'open-source, Bootstrap s'améliore en permanence : de nouvelles fonctions absolument géniales ont été ajoutées comme le 100% mobile responsive ou la très large sélection de plugins jQuery	• mobile first • inclus le support de Less et Sass • très grande collection de composants • très grosse communauté qui propose des centaines d'autres composants	• Les sites qui se ressemblent. • Code inutilisé qui alourdit les pages : Code CSS et JS par défaut qui alourdit le chargement des pages • Nécessité de modifier ou ajouter du code : Il faut ajouter du code supplémentaire afin de se rapprocher au design souhaité • Modification ou adaptation de code original du Framework peut être long : une modification impose de défaire une architecture en imposant des nouveaux critères (CSS,HTML,JS)	
Ionic	onic est un framework utilisant Apache Cordova: un m´elange d'outils et de technologies pour d´evelopper des applications mobiles hybrides rapidement et facilement. Il s'appuie sur AngularJS Cours Plasticit´e page 4 de 7 pour la partie application web du framework (d´eveloppement backend) et sur Cordova pour la partie construction des applications natives. Ce framework open source permet de d´evelopper une application d´eployable sur plusieurs syst`emes tel que Android, iOS ou Windows Phone. Ionic poss`ede son propre th`eme HTML/CSS afin de pouvoir disposer rapidement d'une application fonctionnelle. L'utilisation de Sass permet notamment de modifier le variables du th`eme afin de changer les couleurs, les polices et tout autre param`etre disponible, il met `a disposition un certain nombre d''el´ements comme le fait ´egalement bootstrap. .	• Plus de performance et de fluidit´e dans la gestion des gestes et ´ev`enements (slide, clic, acc`es aux capteurs du smartphone) • Facile et rapide `a construire • Ne co^ute pratiquement rien	Cr´eer des applications avec des technologies front-end peut rendre l'apparence similaire `a une page web classique	
	React(aussi appel´e React.js ou ReactJS) est une biblioth`eque JavaScript libre d´evelopp´ee par Facebook depuis 2013. Le but principal de cette			

Figure 25: Aperçue du tableau

La page avec l'émulateur firefox :

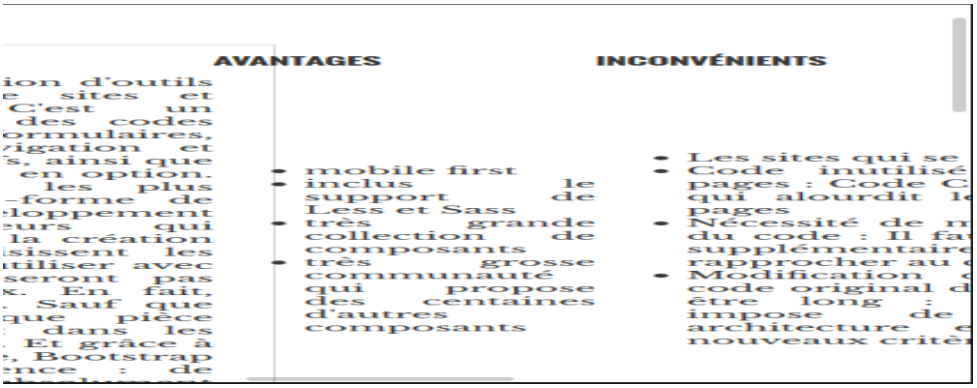


Figure 26: Aperçu tableau (émulateur Firefox)

Conclusion

Bootsrap est un langage facile à implémenter, riche de composants et possède une très grande communié mais il est difficile à personnaliser, tous les sites se ressemblent en plus lors de l'implémentation de la page sans l'utilisation de plusieurs classes c'était mal organisé, non agronomique et surtout la responsivité est mal établie.

4. Tutoriel React JS

Introduction

React (aussi appelé React.js ou ReactJS) est une bibliothèque JavaScript libre développée par Facebook depuis 2013. Le but principal de cette bibliothèque est de faciliter la création d'application web monopage, via la création de composants dépendant d'un état et générant une page (ou portion) HTML à chaque changement d'état.

React est une bibliothèque qui ne gère que l'interface de l'application, considéré comme la vue dans le modèle MVC. Elle peut ainsi être utilisée avec une autre bibliothèque ou un framework MVC comme AngularJS. La bibliothèque se démarque de ses concurrents par sa flexibilité et ses performances, en travaillant avec un DOM virtuel et en ne mettant à jour le rendu dans le navigateur qu'en cas de nécessité.

Même si le virtualDOM limite les interactions avec le DOM réel, il apporte aussi un gros problème de performances. À chaque modification du state, le virtualDOM doit être regénéré pour être comparé, ce qui peut rapidement devenir problématique avec une application utilisant des centaines de composants. Il faudra penser aux performances dès le début de la conception de l'application en précisant par exemple si un composant doit être recalculé. React intègre ainsi une méthode shouldComponentUpdate() qui permet d'éviter le recalcul du VirtualDOM suivant une condition précise.

Les limites de ReactJS :

- Le VirtualDOM rend parfois l'interaction avec les éléments web difficiles
- La regénération du VirtualDOM à surveiller de près, ne pas hésiter à utiliser shouldComponentUpdate()
- Plus "verbeux"que les autres frameworks

Vue globale du projet

ReactJS étant une bibliothèque et non un framework contrairement à AngularJS, il est dépendant de l'infrastructure fournie par le node package manager npm et donc il faut l'installer au préalable, puis

Pour installer l'environnement du Reactjs il faut entrer dans la ligne de commande :

npm install -g create-react-app

Pour installer le package de la création d'applications reactJS et prend en charge toutes les dépendances nécessaires pour ce fait, ainsi qu'optimiser la structure de l'application pour la production.
Pour créer un projet ReactJS, il suffit par la suite d'entrer dans la cli :

create-react-app [nom du porjet]

Et puis l'ouvrir dans un éditeur de texte à fin de commencer de travailler ou pour lancer le hello world auto généré il suffit de entrer ce qui lancera l'application sur le localhost:3000

npm start

l'environnement React étant el place, on aura aussi d'autres bibliothèques à installer pour la gestion de la plasticité de l'application ainsi que le graphisme notamment la bibliothéque **react-responsive** qui nous permettra de faire appel au mediaqueries, avec qui on pourrait modeler le comportement de l'application lors du passage d'un medium à un autre exemple d'un écran de smartphone à un écran de bureau, et assurer ainsi la plasticité de l'application pour l'installer, il suffit de se déplacer dans la racine du projet en cli, et entrer :

npm install –-save react-responsive

Dans cette application on opter vers l'utilisation des composants graphiques offerts par la bibliothéque material-ui de Google pour son adaptabilité et mouillabilité, pour l'installer il suffit d'entrer dans la cli sur la racine du projet :

npm install –-save material-design

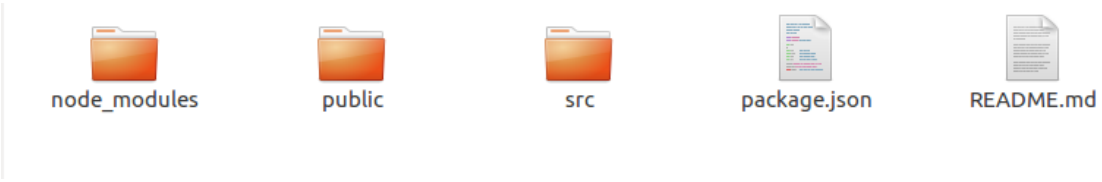


Figure 27 : structure du projet ReactJS

Le dossier node_modules contient les librairies dont dépend notre application

Figure 28 : contenu du dossier src

Le dossier src contient les ressources du projet qui définies sa structure et son comportement tels que les composants et les ressources graphiques ainsi que les routages de l'application.

Composants

React étant basé sur le concepts des composent indépendant et modulable on a rendu chaque élément de l'interface sous forme de composants

```
import React from 'react';
import {GridList, GridTile} from 'material-ui/GridList';
import IconButton from 'material-ui/IconButton';
import StarBorder from 'material-ui/svg-icons/toggle/star-border';

const styles = {
  root: {
    display: 'flex',
    flexWrap: 'wrap',
    justifyContent: 'space-around',
  },
  gridList: {
    width: 500,
    height: 450,
    overflowY: 'auto',
  },
};

const tilesData = [
  {
    img: 'images/Bootstrap.jpg',
    title: 'Boostrap',
    author: 'Ameni'

  },
  {
    img: 'images/reactJS.jpg',
    title: 'ReactJS',
```

Figure 30 : code source du composant barre de navigation

```
  {
    img: 'images/AngualrJS.jpg',
    title: 'Meriem',
    author: 'Danson67'
  },
  {
    img: 'images/Ionic.jpg',
    title: 'Ionic',
    author: 'Inaf'

  },
```

Figure 29 : coude source du la grid liste contenant les tutos

Cette barre de navigation est constituée d'une liste déroulante et d'un bouton de login qui lorsque on y clique produit une alerte que rappel que nous somme authentifier.

```
const GridListIHM = () => (
  <div style={styles.root}>
    <GridList
      cols={2}
      cellHeight={200}
      padding={1}
      style={styles.gridList}
    >
      {tilesData.map((tile) => (
        <GridTile
          key={tile.img}
          title={tile.title}
          actionIcon={<IconButton><StarBorder color="white" /></IconButton>}
          actionPosition="left"
          titlePosition="top"
          titleBackground="linear-gradient(to bottom, rgba(0,0,0,0.7) 0%,rgba(0,0,0,0.3) 70%,rgba(0,0,0,0) 100%)"
          cols={tile.featured ? 2 : 1}
          rows={tile.featured ? 2 : 1}
        >
          <img src={tile.img} />
        </GridTile>
      ))}
    </GridList>
  </div>
);

export default GridListIHM;
```

Ces deux composants constituent la première interface de l'application.

Caractéristiques du ReactJS

➢ JSX

Le JSX est un pseudo-langage d'une syntaxe proche du HTML compilé par le Javascript. Le markup et le code applicatif se composent dans un même fichier. L'avantage est que cette architecture permet d'avoir une auto complétion performante durant le développement notamment lorsque l'on renseigne une variable, un mot clé etc. C'est une différence importante avec AngularJS 2. De ce fait, ReactJS dispose d'une bonne colorisation syntaxique cohérente et complète dans la plupart des éditeurs, souvent d'ailleurs avec une auto complétion au moins partielle et la possibilité d'identifier les erreurs d'écriture au stade de la conception (sans attendre la compilation). Cela dit, AngularJS propose un parser maison pour palier à cette difficulté.

➢ Détection des erreurs

Quand on commet une erreur de syntaxe dans ReactJS, cela ne compilera pas. C'est une excellente chose. Cela signifie que vous êtes en mesure d'identifier immédiatement quelle est la ligne comportant une erreur. **Le compilateur JSX nous indique immédiatement** quand un élément n'est pas fermé par exemple ou quand une propriété est mal utilisée ou inexistante dans le contexte de la page. Cette approche fluidifie considérablement les phases de développement.

➢ Centralisé autour du Javascript

Puisque Javascript supporte de base les boucles, le JSX de ReactJS réutilise toute la puissance de Javascript directement pour ce type d'opérations, et bien d'autres, telles que les maps, les filtres etc.

Donc pour comprendre ReactJS il faut juste comprendre Javascript, contrairement aux autres environnements qui pour les comprendre il faut connaître un grand nombre de mots clé spécifiques à la technologie.

Conclusion

En conclusion ReactJS est un outil très puissant et modulable, il permet aux développeurs une plus grande liberté de choix de bibliothèques secondaires à utiliser et donc d'optimiser la performance de l'application contrairement aux autres plateformes Javascript qui imposent l'utilisation de bibliothèques et d'outils compatible avec la technologie.

5. Angular JS 2

Afin de mieux comprendre l'utilité du Framework Angular2, on commence par une petite introduction sur AngularJS qui met en valeur l'apparition du nouveau Angular.

AngularJS

AngularJS est né en 2009 dans les locaux de Google. Deux développeurs du nom de Brad Green et Shyam Seshadri commençaient sérieusement à déprimer devant leur projet appelé "Google Feedback".

Une immense frustration les envahissait au fur et à mesure que leur code grandissait. Celui-ci comptait approximativement 17 000 lignes à ce moment-là. Autant de lignes de pur front-end qui sont non testables et donc difficilement maintenables.

C'est à ce moment-là que Shyam Seshadri proposa de redévelopper entièrement la solution avec un Framework fait maison. Au bout de trois semaines, l'application ne comptait plus que 1 500 lignes de codes, parfaitement testées.

À compter de ce jour, les autres développeurs de l'équipe ont décidé de prendre en main ce Framework et de travailler avec au quotidien. Une histoire est née, l'histoire d'un Framework que l'on nommera Angular JS.

Ses limites

Bien que AngularJS vient avec beaucoup de mérites ce qui ramène à l'apparition d'une nouvelle version d'Angular, voici quelques points de préoccupation :

- Non sécurisé : Être JavaScript seul cadre, application écrite en AngularJS ne sont pas sûrs. Authentification côté serveur et l'autorisation est indispensable de maintenir une application sécurisée.
- Non dégradable : Si l'utilisateur de votre application désactive JavaScript, alors rien ne serait visible, à l'exception de la page de base.

Futur du Framework

Aujourd'hui une version 2.0 du Framework est en préparation et la philosophie de la 2.0 en dit long sur son contenu puisqu'elle sera « Mobile first ».

Cependant la refonte complète d'AngularJS risque d'être douloureuse pour les développeurs. En effet, aucune rétrocompatibilité avec la version une n'est pour l'heure à l'ordre du jour, mais cela reste possible. De plus, il va falloir que les développeurs apprennent les nouveaux concepts et langages introduits dans la version 2.0. Le temps investit dans l'apprentissage du Framework et dans le développement des applications semblent avoir un goût amer aujourd'hui.

Suite à ses annonces, la communauté AngularJS, à qui le Framework doit en grande partie son succès, a été déçue. On imagine donc assez mal AngularJS ne pas réagir afin de satisfaire sa communauté.

Cependant, la version finale est sortie à la fin de l'année 2015, ce qui laisse encore la place à des changements.

Il est possible de suivre le développement de la 2.0 sur leur nouveau site : https://angular.io.

Structure Générale

Consistes-en plusieurs librairies, dont certaines font partie du noyau et d'autres sont optionnelles.

On construit des applications Angular en écrivant des templates HTML contenant du markup Angularisé, en créant des classes de composant pour gérer ces Templates, en ajoutant la logique applicative dans des services, et en cloisonnant les composants et les services dans des modules.

Ensuite, on lance l'application en bootstrapant le module racine. Angular prend la main, en présentant le contenu de l'application dans un navigateur et en répondant aux interactions de l'utilisateur en fonction des instructions qu'on lui a fournies.

Bien entendu, en pratique, c'est un peu plus compliqué que ça. Chacun de ces concepts peut être approfondi dans d'autres recettes. Pour l'instant, on se concentre sur la vue d'ensemble.

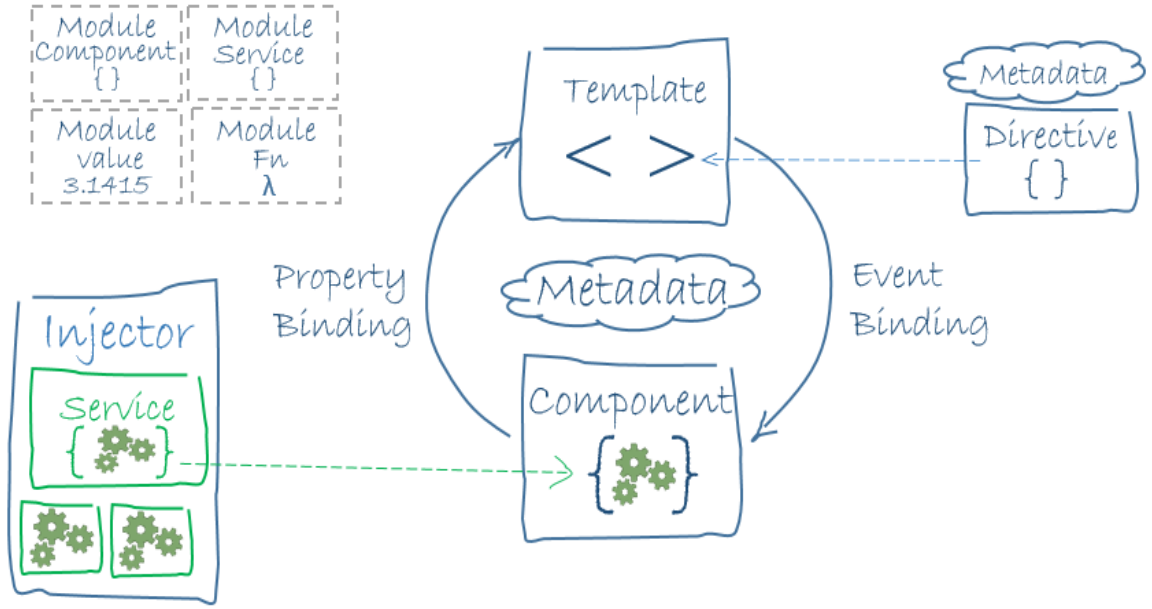


Figure 31:les huit briques d'une application mobile

Installations necessaires

- NodeJS

Il faut commencer par installer nodeJS qui sera utilisé. En effet, c'est grâce à ce dernier qu'il possible d'écrire du JavaScript côté serveur.

Node.js permet, au même titre que PHP ou Java, d'écrire des scripts qui vont vous permettre d'interagir avec les ressources de votre machine.

- VISUAL CODE

Il s'agit d'un outil qui va grandement nous simplifier la vie. Il va notamment nous permettre d'initialiser nos projets en nous créant une structure projet efficace mais également en nous installant l'ensemble des librairies qui nous seront nécessaires au cours de nos développements.

- angular-quickstart

Le projet est basé sur la librairie angular-quickstart. Il nous permet d'avoir les fichiers de configuration de base de chaque projet :

- package.json : définit les dépendances de package npm pour le projet
- tsconfig.json :définit la façon dont le compilateur tapuscrit génère JavaScript à partir des fichiers du projet
- systemjs.config.js:fournit des informations à un module loader où trouver les modules d'application, et enregistre tous les paquets nécessaires

Après avoir téléchargé cette librairie, on doit lancer la commande npm install. Afin de lancer le projet sur le browser on applique npm start

Structure générale du projet

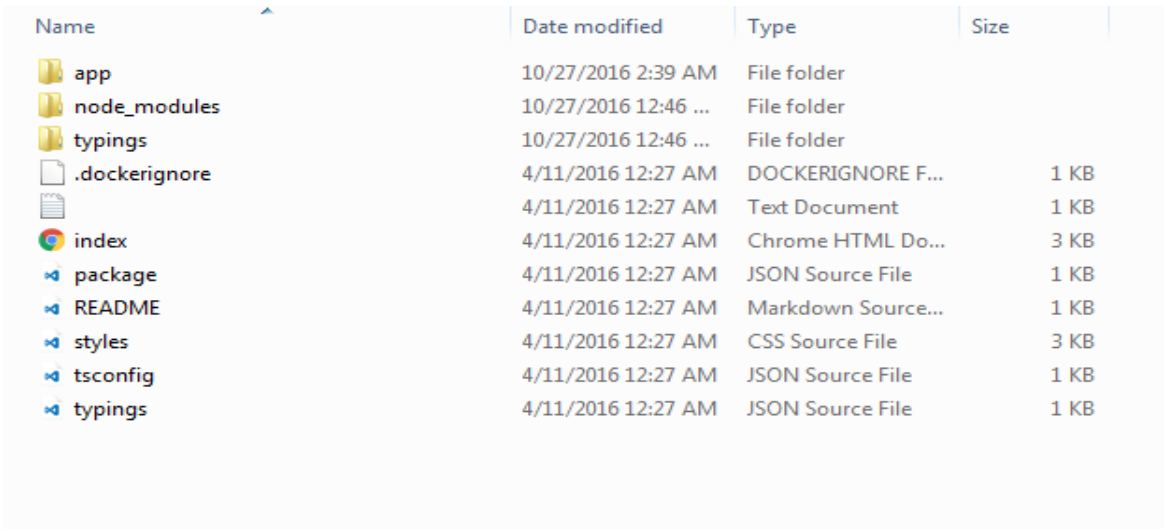


Figure 32: Structure du projet

- Composants :

Parmi les principes basiques d'Angular2 qu'on ne parle plus des conteneurs mais de composants.

Au minimum il existe un seul composant à chaque projet qui a une structure bien déterminée. Prenant l'exemple de app.component.ts qui représente le composant principale de notre application.

- Remarque : On n'oublie pas qu'avec Angular2 on utilise TypeScript comme langage de base.

TypeScript n'est pas imposé, mais son usage est encouragé car il apporte de nombreuses facilités de développement, comme par exemple une auto-complétion très poussée, pour peu que vous utilisiez un IDE digne de ce nom. Par ailleurs, en imposant une syntaxe plus stricte et un typage statique, TypeScript rend le code plus lisible et maintenable.

```
import {Component} from 'angular2/core';
import {RouteConfig, ROUTER_DIRECTIVES} from 'angular2/router';

import {AboutUsComponent} from './aboutus.component';
import {PricingComponent} from './pricing.component';
import {MemberDetailsComponent} from './memberdetails.component';

@RouteConfig([
  {path: '/aboutus', name:'AboutUs', component: AboutUsComponent, useAsDefault: true},
  {path: '/pricing', name:'Pricing', component: PricingComponent},
  {path: '/memberdetails/:id', name:'MemberDetails', component: MemberDetailsComponent}
  {path: '/*other', name:'Other', redirectTo: ['AboutUs']}
])

@Component({
    selector: 'my-app',
    templateUrl: 'app/app.component.html',
    directives: [ROUTER_DIRECTIVES]
})
export class AppComponent { }
```

Figure 33 : TypeScript

Chaque composant a son propre code html et son code css.

Template Url : représente le chemin (path) du code html du composant.

Routage

Avec le routage notre application est devenue beaucoup plus rapide. Il nous permet essentiellement de naviguer d'une page à une autre.

Tout d'abord il faut modifier index.html en ajoutant la commande ci-dessous :

```
<script src="node_modules/angular2/bundles/router.dev.js"></script>
```

Figure 34:Commande Routage

Maintenant, on importe ROUTER PROVIDERS dans main.ts:

```
import {bootstrap}    from 'angular2/platform/browser';
import {AppComponent} from './app.component';
import {ROUTER_PROVIDERS} from 'angular2/router';

bootstrap(AppComponent, ROUTER_PROVIDERS);
```

Figure 35:Import Routage

On ajoute RouteConfig et RouterDirectives à chaque composant où on va utiliser les routages avec la définition de chaque chemin (path) comme on a déjà vu dans l'exemple du composant.

Material Design et bootstrap

- Bootstrap

Angular nous offre la possibilité d'utilisé boostrap. Afin d'assurer ça on doit ajouter quelques lignes dans notre index.html.

```
<!-- Bootstrap -->
<link rel="stylesheet" type="text/css" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-material-design/0.3.0/css/roboto.min.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-material-design/0.3.0/css/material-fullpalette.min.c
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-material-design/0.3.0/css/ripples.min.css">
```

```
<script src='http://code.jquery.com/ui/1.10.2/jquery-ui.js'></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-material-design/0.3.0/js/material.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-material-design/0.3.0/js/ripples.min.js"></script>
```

Figure 36: Ajout du bootsrap

Bootstrap nous permet d'avoir des pages responsives et élégantes.

- Material Design

Le Material design est une bibliothèque de Angular2 qui nous permet d'ajouter les composants d'une façon simple. Par exemple, dans notre code on aura besoin de material design cards qui définit chaque cours. On aura besoin aussi d'installer à chaque fois le matériel qu'on doit utiliser avec la commande npm comme suit :

npm install @angular2-material/core @angular2-material/card

Il faut toujours télécharger le matériel core car tous les autres matériels dépendent de lui.
On ajoute aussi dans index.html :

```
  <!-- Material Design fonts -->
<link rel="stylesheet" type="text/css" href="//fonts.googleapis.com/css?family=Roboto:300,400,500,700">
<link rel="stylesheet" type="text/css" href="//fonts.googleapis.com/icon?family=Material+Icons">
```

Figure 37: Insertion du material design

Conclusion

Avec Angular2 on a eu la possibilité de :

- Augmenter les performances : l'une des plus grands reproches qui aient été faits à Angular sont ses lacunes en termes de performance. Pour pallier ce problème, les développeurs d'Angular ont décidé de se reposer davantage sur les briques natives du navigateur, comme par exemple, les prometteurs WebComponents.
- Améliorer la productivité : en affichant une syntaxe expressive basée sur la syntaxe de ES2015/TypeScript (annotations, import, types, …), plutôt que sur des surcouches, Angular 2 rompt avec une longue tradition de Framework à boiler plate

- Angular 2 s'intègre par ailleurs parfaitement bien avec les composants construits via d'autres bibliothèques comme bootstrap, ionic.

Mais à mon avis il marque beaucoup de limite :

- La documentation n'est pas encore complète
- Le TypeScript peut être déstabilisant et rend le Framework plus complexe à appréhender
- Beaucoup de concept à comprendre avant d'être efficace
- Les commandes npm je vois qu'ils sont destinés à linux plutôt que les autres systèmes d'exploitation parce que à chaque fois on aura des problèmes de dépendances à cause du npm.
- Il nous semble intéressant de souligner qu'Angular2, n'a simplement ajouté que l'aspect de composant qui existait déjà en ReactJs. On ne peut pas nier qu'il sert aussi à intégrer des nombreux outils supplémentaires comme le module HTTP ou le validateur de Formulaire mais ça reste restreint parce que la documentation pour savoir l'appliquer change très vite.

6. Comparaison

Technologies	Ionic	Angular JS 2	Bootstrap	React JS
Ionic	—	Ionic est une surcouche complémentaire pour le code Angular 2. Grâce à Ionic 2, on peut satisfaire les fonctionnalités hybrides telles que l'accès aux capteurs.	Réellement on ne peut pas comparer bootsrap et ionic car l'un est destiné mobile et l'autre web mais si on veut une application mobile native qui utilise les ressources matériels ionic est le meilleur choix et si on vent développer une application intuitive en peu de temps on a recours à bootsrap	On ne peut pas vraiement les comparé l'un est orienté vers des application web singlapage et l'autre vers des applications mobile natives et multiplatformes ce qui sont des domaines bien different avec des approches divergentes
Angular JS 2	Ionic utilise $scope et les contrôleurs. Angular 2 les a remplacés par les components et les directives	—	Bootsrap beaucoup plus facile à implémenter Parfait pour les débutants du développement front end Non structuré et organisé par rapport à angularJS2	ReactJS utilise JSX alors que Angular utilise TypeScript, bien qu'ils soient devenus compatibles ils conservent leurs particularités ReactJS detecte les erreurs lors de la compilation alors que angular les detectent lors de l'execution

Bootstrap	Ionic et Bootstrap proposent des composants prédéfinis à utiliser Ionic est orienté application mobile Accès aux capteurs du smartphone à travers Ionic à l'aide de Cordova – rendre l'application native Ionic est plus performant	-Angular 2 est un cross plateforme qui nous permet de créer une Single Page WebApp, alors que Bootstrap c'est juste une plateforme de web design ce qui le rend difficile à comparer entre les deux	—	Bootstrap est centré autour de l'HTML alors que ReactJS est centré autour le Javascript ce qui rend ReactJS bien plus puissant en termes de performance que Boostrap en depit de la puissance du JavaScript
React JS	Ionic utilise des langages web pour le développement d'application contrairement à React qui utilise des composants natifs Ionic suit le concept de « write once, run everywhere » ce qui le permet d'adapter son comportement par rapport aux plateformes contrairement à React qui ne permet d'utiliser que des composants spécifiques à chaque plateforme Ionic utilise du HTML pour ses interfaces et du CSS pour le design il est en ligne avec le modèle MCV. React présente une combinaison entre XML et Javascript qui donne le JSX. Le	-le point commun et fort au même temps entre les deux technologies c'est qu'ils sont basés sur les composants -Avec reactJS, on doit savoir quelle API qu'on va utiliser à chaque prtejet alors que c'est plus simple avec Angular 2 grâce au préconfiguration de quickstart. -Au niveau de template angular 2 est très limité par rapport à react. Les mêmes fonctionnalités avec React sont plus légères en fonction de ligne se code. Data binding bidirectionnel est plus optimisé par rapport au react .	Documentation disponible -facile à implémenter - Responsive par défaut	—

	développement peut être plus difficile pour les designers, vu qu'il ne propose pas l'ensemble des règles CSS que les navigateurs prennent généralement en charge. Pour la partie test, Ionic et React offrent une visualisation instantanée des changements dans l'application sans devoir rafraichir la page. Ionic propose des composants prédéfinis à utiliser, Dans react c'est au développeur de tout créer ou presque React permet un affichage de map natif alors que Ionic permet un affichage orienté web Taille de l'application React sur le mobile est plus important qu'avec Ionic			