Adaptation des interfaces à l'environnement

Eau Pour Tous
Water For Everyone

BootStrap- Pure Css - Ionic 2

Abdelkader Ibrahim - Ben Aouicha Zeyneb - Sahli Helmi

Sommaire

Introduction

EAU POUR TOUS est un site web et une application mobile de sensibilisation et de charité pour les zones pauvre en eau. Avec notre projet, On doit sensibiliser ses internautes du besoin immédiat en eau potable dans ces zones et leur proposera ensuite de faire un don afin d'aider à créer la différence, Aussi Visualiser le témoignage des habitants de ces zones.
Notre site doit alors s'adapter aux différents supports de visualisation :

- L'application pourra être visitée sur des postes d'ordinateurs standard où elle doit afficher des articles/photos et vidéo sur le sujet avec l'option « faire un don ».
- L'application doit être adaptée aux terminaux mobiles utilisés surtout durant des évènements occasionnels comme « La journée mondiale de la terre » ou « La journée mondiale de l'eau » ici le site doit mettre en relief l'entré : « Faire un don ».

Notre projet utilise plusieurs adaptation : une adaptation à la conception d'où faciliter la vie au développeur et aussi à l'exécution dont le but de faciliter la vie au utilisateurs de notre application. Dans ce rapport on va vous présenter un tutoriel sur les trois technologies utilisés dans notre projet, les avantages, les limites et aussi l'adaptation visée.

Ionic 2 : Tutorial

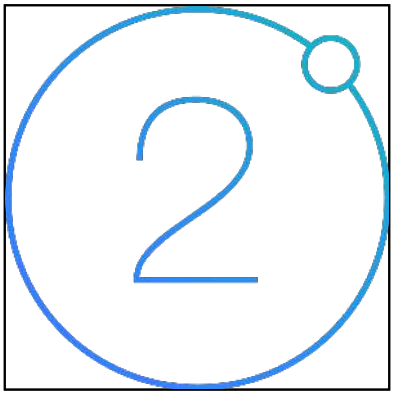


figure 1 : Logo Ionic 2

Présentation de la technologie

IONIC est une Framework permettant de construire des applications hybrides. Les applications hybrides sont des applications mobiles web fonctionnant dans un navigateur à l'intérieur d'une application native. L'objectif principal de cette Framework est le fait de pouvoir développer des applications cross- plateforme basées sur des langages web comme HTML5, CC3 et Javascript, dans le but d'être capable d'utiliser certaines fonctionnalités du mobile, comme les capteurs et les actionneurs, sans avoir besoin de connaître les langages natifs de toutes les plateformes ciblées. Pour implémenter une application hybride, Ionic utilise des outils comme Apache Cordova pour effectuer la communication entre l'application web et la plateforme native.

Il existe actuellement 2 versions de cette Framework. Tout d'abord, il existe IONIC 1, fonctionnant sous Angularjs pour effectuer la gestion de la logique métier de l'application. De même, il existe une deuxième version appelée IONIC 2.

Nous avons décidé de présenter, à travers ce tutoriel, l'utilisation de la technologie IONIC 2 fonctionnant sous Angular 2. De ce fait, nous serons amenés à découvrir et à utiliser les web components. De même, étant donné que notre objectif est de réaliser une application à destination de terminaux mobiles, nous montrerons aussi comment réaliser l'accès aux capteurs et actionneurs de l'appareil cible grâce à IONIC 2.

Ionic 2 a des meilleurs performances que Ionic, mais ces derniers reste faible par rapport à un développement sur du natif comme Android et Ios, ou sur du cross-platform comme Xamarin.

Réalisation

Pour commencer, il faut créer un projet ionic 2. Il y a peu de prérequis, il faut juste avoir une version de Node.js supérieur à la version 6. Pour cela, il suffit de se rendre sur le site nodejs.org pour télécharger la dernière version stable. Cette installation vous permet d'utiliser le gestionnaire de paquet npm. Ainsi, vous pourrez utiliser la commande suivante pour télécharger et installer ionic et cordova.

$ npm install -g ionic cordova

$ ionic start EauPourTous blank --type ionic-angular

La deuxième commande vous permet de créer le projet dans le répertoire courant.

Le *ionic-angular* est nécessaire pour faire un projet sous ionic 2.

Ces deux commandes permettent de créer les 3 fichiers nécessaires à un composant ou à une page :

- le contrôleur typescript,
- le template html,
- et le fichier scss pour le style du template.

Dans notre Application, On a 5 pages de Base d'où chaque page contient trois fichiers de base :le contrôleur typescript, le template html et le fichier de style du template le .scss.

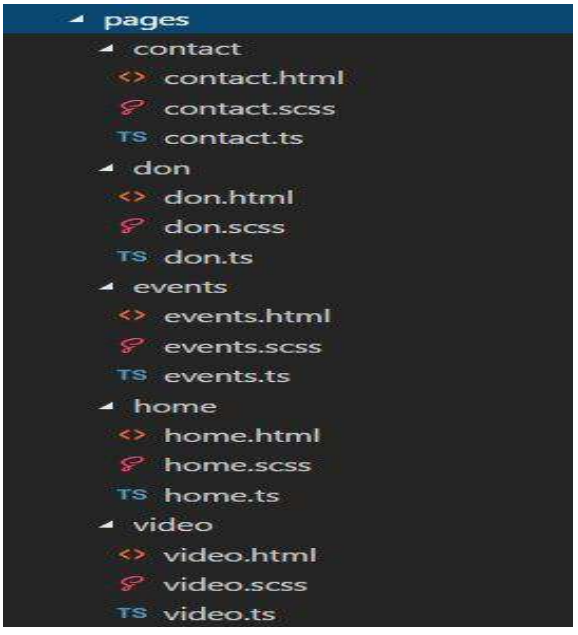


figure 2 : La Structure des pages

Pour notre application, il faut créer les pages :

- vidéo qui contiendra les vidéos de témoignages,
- home qui représente la page d'accueil de notre application,
- events qui permet à l'utilisateur de consulter les futures événements,
- don qui permet à notre visiteur de faire un don pour notre Association,
- et contact qui permettra de nous contacter,

ainsi que les composants

- eventList qui construira une liste des événements,
- videoList qui construira une liste des vidéos de témoignage,
- paysList pour les pays qui souffrent encore de manque de l'eau.

Commençons maintenant par éditer *page/home/home.ts*. On a fait un petit menu en haut du page en utilisant la notion des buttons segments sous ionic 2 Ou on a mis les trois parties de base de notre Application (Faire le don, Les vidéos de témoignage et les événements à venir) en utilisant l'élément <ion-segment>.

figure 3 : page d'accueil

Dans le corps de la page on s'intéresse à créer une liste des pays qui souffrent du manque de l'eau en donnant des images avec des descriptions pour chaque cas comme ceci :

figure 4 : page de pays.

Pour expliquer rapidement la syntaxe, (click) et (tap) permettent de gérer les événements de clic ou d'appuis tactiles sur cette balise. {{..}} permet d'appeler des variables utilisées dans le contrôleur. Enfin, *ngFor permet de faire une boucle *for* directement sur une balise et son contenu, et *ngIf permet d'afficher ou non une balise et son contenu.

La fonction *"goto"* n'est pas native, il ne s'agit que du raccourci de :

this._navCtrl.push({});

On obtient *_navCtrl* en ajoutant un argument dans le constructeur du component :

constructor(platform: Platform, _navCtrl: NavController) {

Ce service permet d'avoir facilement une pile des pages consultée pour permettre un retour arrière automatique dans chaque solution native.

Il faut utiliser au maximum les balises de ionic 2. Sinon, on peut perdre la compatibilité et adaptation responsive produite par ionic. Par contre, on perd des libertés de personnalisation sur ces balises.

On page maintenant à la page qui nous permet de lire les vidéos de témoignage une par une, avec la possibilité de regarder la vidéo en plein écran.

figure 5 : page de vidéos de témoignage.

Tout d'abord, pour accéder à la lecture de vidéo sous Ionic 2, une installation des plugin sera obligatoire d'où on a utilisé ces deux commandes:

$ ionic cordova plugin add https://github.com/moust/cordova-plugin-videoplayer.git
$ npm install --save @ionic-native/video-player

Dans cette partie on a utilisé le <ion-card> qui contient le frame de vidéo de Youtube, Ionic 2 ne donne pas l'accès à mettre des video sur cette forme classique :

- https://www.youtube.com/watch?v=gGGeorOV7Jw

Donc la il faut changer implicitement **watch?=v** par **embed** pour que la vidéo soit intégré dans la vue et être visualisé par notre utilisateur. d'où le lien devient comme ça :

- https://www.youtube.com/embed/gGGeorOV7Jw

En passant pour la vue qui intéressent une grande pourcentage de notre application : c'est la page des événements à venir comme le jour de la terre ,de l'eau et la nuit de l'eau avec une description bien détaillée.

```
<ion-card *ngFor="let item of upload">
  <ion-item color="primary">
    <ion-avatar item-left>
      <img src="{{item.imgl}}">
    </ion-avatar>
    <h2>{{item.name}}</h2>
    <p>{{item.dateEvent}}</p>
  </ion-item>
  <img src="{{item.imgs}}">
  <ion-card-content>
    <ion-row padding>
        <ion-item>
          <button ion-button icon-left clear-small>
            <ion-icon name="link"></ion-icon>
            <div>{{item.link}}</div>
          </button>
        </ion-item>
```

figure 6 : events.html

Dans ce bout de code, on an a fait une liste des événement qu'on charge dynamiquement depuis le fichier.ts d'où chaque item présente un détails comme le lien, le lieu, les nombres de participants par événements. et pour rendre sera adaptable pour les autres plateformes on a ajuster ca avec une petite manipulation de style sur le template

```
ion-card.card{
    margin:0px;
    width:100%;
    }
```

figure 7 :event.scss

le même principe de les événements est utilisé pour la rubrique contact mais juste pour une seul ion-card dont le but est d'afficher des détails pour contacter le développeur de cette application.

figure 8 : la page de contact

En finissant avec la page qui permet à l'utilisateur de faire le don pour notre association

figure 9 : page faire un don

Au bas de cette page, On a ajouté un footer qui va être interprété avec la meme interpretation sur tous les plateformes.

En cliquant sur le bouton faire un don ca va déclencher une alert qui va demander à l'utilisateur de choisir avec combien d'euros va aider les gens en manque de l'eau :

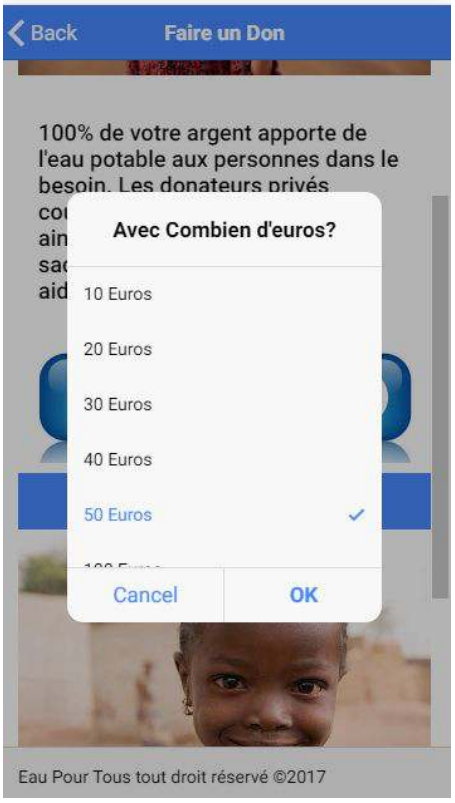


figure 10 : Alert 1 faire un don.

Après que l'utilisateur fait son choix, une autre alert qui s'affiche et une légère vibration va se produire comme retour à l'utilisateur.

Pour accéder à la vibration de notre device physique, ces deux commandes seront obligatoire pour le bon fonctionnement de la vibration dans notre application.

$ ionic cordova plugin add cordova-plugin-vibration

$ npm install --save @ionic-native/vibration

Déploiement et exécution

Pour exécuter facilement notre application, on peut utiliser la commande suivante :

$ ionic serve

Ainsi NodeJs fait tourner notre application en local, et met à jour à chaque changement.

Pour avoir un test plus réaliste de l'application, on peut très facilement tester sur un téléphone android. Pour cela, il suffit d'avoir bien défini son $ANDROID_HOME, puis d'utiliser la commande :

$ ionic cordova run android --device

Grâce à la commande ionic run android, qui permet de tester notre application sur un device Android relié en usb à l'ordinateur, en tant qu'application native. L'avantage de cette manière de tester est qu'on peut utiliser les capteurs intégrées au smartphone. Cependant, ce processus de déploiement sur device connecté prend beaucoup de temps (1min environ), contrairement au test sur navigateur qui ne prend que quelques secondes à recharger automatiquement après une modification du code.

Outils de développement et de débogage

Pour développer cette application, j'ai utilisé Visual Studio Code comme éditeur de code Cross-plateforme et open Source.

Visual Studio Code est parfait pour ce projet car cet éditeur est dédié au code Cross Plateforme d'où il peut interpréter tous les types de fichier qui nous intéressent dans ce projet.

En ce qui concerne le débogage, plusieurs possibilités s'offrent à nous. Nous pouvons utiliser un navigateur web ou alors effectuer le lancement de l'application directement sur une cible Android.

Dans notre cas, nous avons utilisé le navigateur Google Chrome pour simuler notre application. Google Chrome permet de simuler différents types de mobiles et tablettes. De même, il nous permet d'effectuer des changements d'orientation, ce qui est bien pour tester l'adaptation écran.

Capacité d'adaptation

Je n’ai malheureusement pas pu intégrer l'événement de Shake lors le scénario était de secouer le téléphone pour faire le don , même si j’ai réalisé ça sur Android Native auparavant dans le projet des techniques d'interactions.

Sinon ca n’etait pas trop difficile de manipuler la caméra ou le capteur de position sous ionic 2 mais il n’y a pas de scénario pour inclure ces fonctionnalités natives à mon application. Par contre, j’ai séparé au maximum les Web Components, pour pouvoir les réutiliser.

En parlant de l’adaptation, Ionic 2 nous permettre de réaliser deux types d’adaptation.

-Adaptation à l’environnement avec prise en compte de la mobilité

-Un retour interactif pour l’utilisateur après le don : l’utilisateur va recevoir une vibration après qu’il fasse le don..

-Affichage des vidéos de témoignage : accès au plugin natif des videos player.

-Adaptation au dispositif

-Ajustement de l’affichage par rapport à la taille de l’écran du dispositif cible téléphonique en ne faisant varier que le type de dispositif sur lequel l’application est utilisée

Avantages

- Plus de performance et de fluidité dans la gestion des gestes et événements.
- Facile et rapide à construire.
- Interface adaptable aux plusieurs plateformes (sans duplication de code) .
- Composants modulaires,facilement imbricables.
- Accès à beaucoup de capteurs

Inconveignants

- Créer des applications avec des technologies front-end peut rendre l’apparence similaire à une page web classique.
- Contrôle des composants (Moins souple que dans un développement natif).
- Difficulté pour l’Accès aux capteurs très spécifiques.
- Performances globales (Performances des animations).
- Debuggage.

Bootstrap : Tutorial

figure 11 : Logo Bootstrap.

Présentation de la technologie

Bootstrap est une collection d'outils utile à la création du design 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 GitHub. En effet, l'intercompatibilité des éléments supportés par Bootstrap rend le design et la réalisation des sites web une magnifique expérience pour tous les développeurs qui le maîtrisent. Bootstrap est aussi un Framework open-source et il s'améliore d'une manière permanente: de nouvelles fonctionnalités absolument utiles ont été ajoutées comme l'approche Mobile First complémentaire au Responsive Design et qui est conçu afin de bâtir une structure et un web design dédié d'abord pour les smartphones et les tablettes plutôt que pour les ordinateurs.

Une des principales forces de ce Framework c'est aussi sa structure en grille: Il s'agit tout simplement d'un découpage en cellules de mêmes dimensions. On peut alors décider d'organiser du contenu en utilisant, pour chaque élément, une ou plusieurs cellules.

La grille Bootstrap comporte par défaut 12 colonnes. Cependant, il faut savoir qu'elle est flexible et extensible dans la mesure où l'on peut réduire le nombre de colonnes ou en insérer de nouvelles dans chaque colonne existante.

Utilisation du Framework

D'après le site officiel de Bootstrap, on peut l'utiliser via cinq méthodes :

- Utilisation des liens CDN Bootstrap fournies par le MaxCDN
- En récupérant les fichiers archivés à partir du site officiel de Bootstrap
- l'Installer avec Bower
- l'Installer avec NPM
- l'Installer avec Composer

Outils de développement

Pour commencer à développer avec Bootstrap on n'aura besoin que d'un simple éditeur de code comme Notepad++ ou Visual Studio Code et d'un navigateur web comme Google Chrome Descktop et Mobile.

Réalisation

1ére étape : On crée une squelette de page web classique et on inclut la balise contenant le Viewport, les liens CDN Bootstrap (CSS et Js) nécessaires pour le Framework et des liens utiles pour la police.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <!-- Theme Made By www.w3schools.com - No Copyright -->
  <title>Water Poverty Website</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <link href="https://fonts.googleapis.com/css?family=Montserrat" rel="stylesheet" type="text/css">
  <link href="https://fonts.googleapis.com/css?family=Lato" rel="stylesheet" type="text/css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
```

Figure 12 : Les liens nécessaires

2éme étape : On ajoute le menu responsive en spécifiant son nom de classe « navbar » et sa position par « navbar-default » et « navbar-fixed-top »

```
<body id="myPage" data-spy="scroll" data-target=".navbar" data-offset="60">
<nav class="navbar navbar-default navbar-fixed-top">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#myNavbar">
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#myPage">Water For Everyone</a>
    </div>
    <div class="collapse navbar-collapse" id="myNavbar">
      <ul class="nav navbar-nav navbar-right">
        <li><a href="#about">ABOUT</a></li>
        <li><a href="#projects">PROJECTS</a></li>
        <li><a href="#events">Events</a></li>
        <li><a href="#donate">DONATE</a></li>
        <li><a href="#contact">CONTACT</a></li>
      </ul>
    </div>
  </div>
</nav>
```

Figure 13 : Création du menu

Ensuite, afin d'implémenter le « Slider » on aura besoins de ces noms de classes « carousel », « slide » et « carousel-inner » ensuite « carousel-control » pour contrôler le « Slider » comme c'est décrit ci-dessous :

```
<div id="mySlider" class="carousel slide" data-ride="carousel">
  <!-- Indicators -->
  <ol class="carousel-indicators">
    <li data-target="#mySlider" data-slide-to="0" class="active"></li>
    <li data-target="#mySlider" data-slide-to="1"></li>
    <li data-target="#mySlider" data-slide-to="2"></li>

  </ol>

  <!-- Wrapper for slides -->
  <div class="carousel-inner" role="listbox">···
  </div>

  <!-- Left and right controls -->
  <a class="left carousel-control" href="#mySlider" role="button" data-slide="prev">···
  </a>
  <a class="right carousel-control" href="#mySlider" role="button" data-slide="next">
    <span class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
```

Figure 14 : Création du slider

3éme étape : On ajoute deux conteneurs du type "container-fluid" pour les rubrique "About Our Project" et "Our values" et ayant la composition suivante pour chacun d'eux :

- Première « div » pour le logo ayant comme taille 2 colonnes "xs" afin de supporter les petits écrans. faut spécifier cela avec la classe « col-xs-2 »
- Deuxième « div » pour le texte ayant comme taille 10 colonnes "xs" .

```
<!-- Container (About Section) -->
<div id="about" class="container-fluid">
  <div class="row">
    <div class=" col-xs-2">
      <span class="glyphicon glyphicon-globe logo"></span>
    </div>
    <div class=" col-xs-10">···
    </div>

  </div>
</div>
```

Figure 15 : Création du conteneur pour la section "About"

4éme étape : Cette étape consistera d'implémenter les 4 parties suivantes :

- Une vidéo en spécifiant le nom de classe : « embed-responsive »
- Un portfolio pour les évènements en spécifiant les noms des classes : « slideanim » et « thumbnail »
- Rubrique pour les témoignages du type « Carousel Slide »
- Une carte géographiques avec des marqueurs représentant les zones avec des besoins critiques en eaux potables « GoogleMap »

```
  <div class="embed-responsive embed-responsive-16by9 col-xs-12">
    <iframe class="embed-responsive-item" src="https://player.vimeo.com/video/14176808?autoplay=1&loop=1&playlist=BCHhwxvQqxg" frameborder="0" allowfullscreen></iframe>
  </div>
  <br><br>

</div>
<div class="myflex" >
<!-- Container (Portfolio Section) -->
<div id="events" class=" c container-fluid text-center bg-grey">
  <h2>Events</h2><br>
  <h4>This year events :</h4>
  <div class="row text-center slideanim">
    <div class="col-sm-4">
      <div class="thumbnail">···
      </div>
    </div>
    <div class="col-sm-4">···
    </div>
    <div class="col-sm-4">···
    </div>
  </div><br>

  <h2>Testimony</h2>
  <div id="myCarousel" class="carousel slide text-center" data-ride="carousel">···
  </div>
</div>

<div id="googleMap" class="a" style="height:400px;width:100%;"></div>
```

Figure 16 : Implémentation du vidéo, évènements, témoignages et carte géographique

5éme étape : Finalement on aura besoin d'implémenter :

- Un bouton et une fenêtre du type « modal » pour effectuer les dons
- Une zone de contact

```
<div id="donate" class="b container-fluid">
  <div class="text-center">
    <h2>donate</h2>
    <h4>Choose a donating amount that works for you</h4>
    <button type="button" class="btn btn-primary btn-lg "data-toggle="modal" data-target="#myModal">Donate Now</button>
    <!-- Modal -->
    <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel">
      <div class="modal-dialog" role="document">
        <div class="modal-content">
          <div class="modal-header">
            <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span>
            <h4 class="modal-title" id="myModalLabel">Donate Form :</h4>
          </div>
          <div class="modal-body">···
  </div>
</div>
</div>
```

Figure 17 : Implémentation de la partie : « Donate »

```
<!-- Container (Contact Section) -->
<div id="contact" class="container-fluid bg-grey">
  <h2 class="text-center">CONTACT</h2>
  <div class="row">
    <div class="col-xs-5">
      <p>Contact us and we'll get back to you within 24 hours.</p>
      <p><span class="glyphicon glyphicon-map-marker"></span> France, Nice</p>
      <p><span class="glyphicon glyphicon-phone"></span> +33 1515151515</p>
      <p><span class="glyphicon glyphicon-envelope"></span> myemail@something.com</p>
    </div>
    <div class="col-xs-7 slideanim">
      <div class="row">
        <div class="col-xs-6 form-group">
          <input class="form-control" id="name" name="name" placeholder="Name" type="text" required>
        </div>
        <div class="col-xs-6 form-group">
          <input class="form-control" id="email" name="email" placeholder="Email" type="email" required>
        </div>
      </div>
      <textarea class="form-control" id="comments" name="comments" placeholder="Comment" rows="5"></textarea><br>
      <div class="row">
        <div class="col-xs-12 form-group">
          <button class="btn btn-default pull-right" type="submit">Send</button>
        </div>
      </div>
    </div>
  </div>
</div>
```

Figure 18 : Implémentation de la partie : « Contact »

A ce stade le résultat obtenu sera semblable à celui aux captures d'écran suivantes :

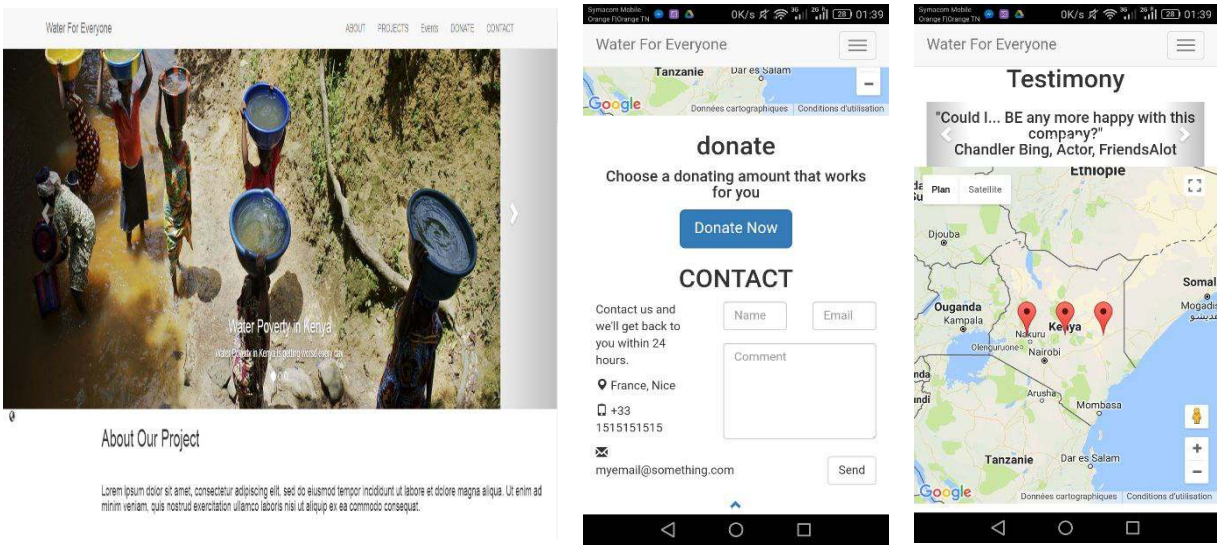


Figure 19 : Visualization du résultat sur un ordinateur et un Smartphone Android

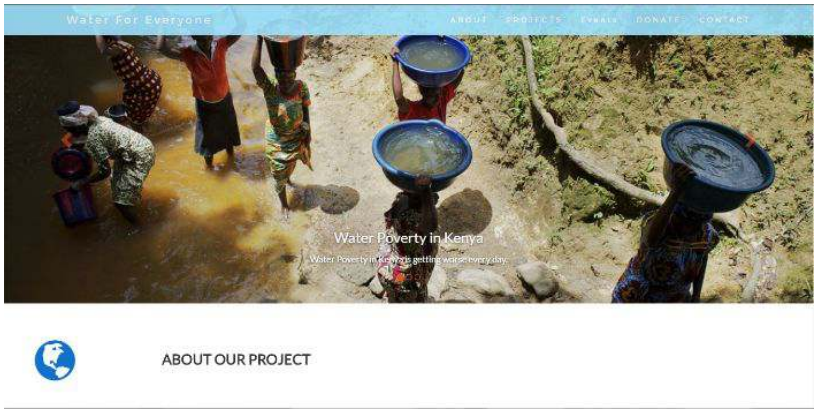
En effet on remarque que si on se limite uniquement au Bootstrap on aura un résultat décevant surtout avec les slides de la classes « Carousel » qui n'occupent pas la totalité du largeur de l'écran d'où la nécessité de faire des corrections en utilisant du CSS et des « Media Queries »

Le code suivant illustre comment utiliser les « Media Queries » afin de corriger les problèmes liés au « Carousel » :

```
@media screen and (min-width: 768px) {
  .logo {
      font-size: 70px;
  }

}
.carousel-caption h3 {
    color: #fff !important;
}
.carousel-inner img {
  width: 100%; /* Set width to 100% */
  margin: auto;

@media (max-width: 600px) {
  #mySlider img {
  margin-top: 50px;

  }
  .carousel-caption {
    display: none; /* Hide the carousel text when the screen is less than 600 pixels wide */
  }
}
```

Figure 20 : Implémentation des « Media Queries »

Le résultat obtenu après l'ajout du CSS et des « Media Queries » :



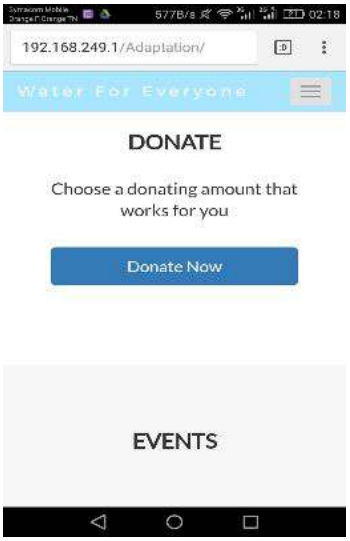


Figure 21 : Visualization du résultat après modifications sur un ordinateur et un Smartphone Android

Capacité d'adaptation

Afin d'avoir une meilleure adaptation aux écrans mobiles j'ai identifié les limites suivantes du Bootstrap que j'ai essayé de les corriger :

Problème 1 : J'ai remarqué que la zone de la superposition du « Navbar » avec le « Slider » n'a pas les même proportions sur l'écran de l'ordinateur que sur le Smartphone. En effet presque la moitié du slider est cachée par le menu sur Smartphone.

Solution : Créer une marge fixe imbriquée dans une Media Querry et qui couvre la barre du menu.

```
@media (max-width: 600px) {
  #mySlider img {
  margin-top: 50px;

  }
```

Figure 22 : Solution du problème 1

Problème 2 : La configuration de la vidéo du « iframe » permettant de la faire lancée automatiquement ne fonctionne pas sur les terminaux mobiles.

Solution : Aucune solution possible (même en utilisant du JavaScript et l'API officielle)

Problème 3 : La carte géographique de « Google Map » n'est pas responsive et n'affiche pas la totalité des marqueurs sur le Smartphone.

Solution : Modifier la fonction javascript de l'API GoogleMap pour qu'elle prend en compte la taille du dispositif pour fixer le niveau du zoom.

```
<script>
function myMap() {
var width = $(window).width();
var customizedZoom ;
if( width<800){
  customizedZoom = 5 ;
}
else{
  customizedZoom = 8 ;
}
var myCenter = new google.maps.LatLng(0.0236,37.9062);
var myCenter2 = new google.maps.LatLng(0.0238,39.9062);
var myCenter5= new google.maps.LatLng(0.0256,35.9062);

var mapProp = {center:myCenter, zoom:customizedZoom, scrollwheel:false, draggable:false, mapTypeId:google.maps.MapTypeId.ROADMAP};
var map = new google.maps.Map(document.getElementById("googleMap"),mapProp);
var marker = new google.maps.Marker({position:myCenter});

var marker2 = new google.maps.Marker({position:myCenter2});

var marker5 = new google.maps.Marker({position:myCenter5});

marker.setMap(map);

marker2.setMap(map);

marker5.setMap(map);


}
```

Figure 23 : Solution du problème 3

Problème 4 : Bootstrap ne permet pas de changer l'ordre des lignes de ses éléments de layout (lignes, colonnes ...). En effet l'une des premières contraintes de notre application était de faire monter la balise « Donate » sur les écrans mobiles afin de la mettre en valeur lors des événements de charité.

Solution : Utiliser les « FlexBox » et les « Media Queries » du CSS3 qui représente un modèle de boite flexible fait pour réordonner la disposition d'une page.

```
.myflex{
  display: flex;
  flex-direction: column;
}
@media screen and (max-width: 768px) {
  .col-xs-4 {···
  }
  .btn-lg {
      width: 100%;
      margin-bottom: 35px;
  }
  .logo {···
  }
  .a{order: 3;}
  .b{order: 1;}
  .c{order: 2;}

}
```

Figure 23 : Solution du problème 4

Problème 5 : La classe « Modal » qui représente une fenêtre de pop-up à un comportement étrange sur les terminaux du type Android. En effet, lorsqu'on sélectionne un champ de texte de cette fenêtre le scroll automatique ne fonctionne pas : faut commencer la saisie pour qu'il se met à jour.

Solution : Aucune solution possible (Bug officiel déclaré sur le "Wall of browser bugs" de Bootstrap)

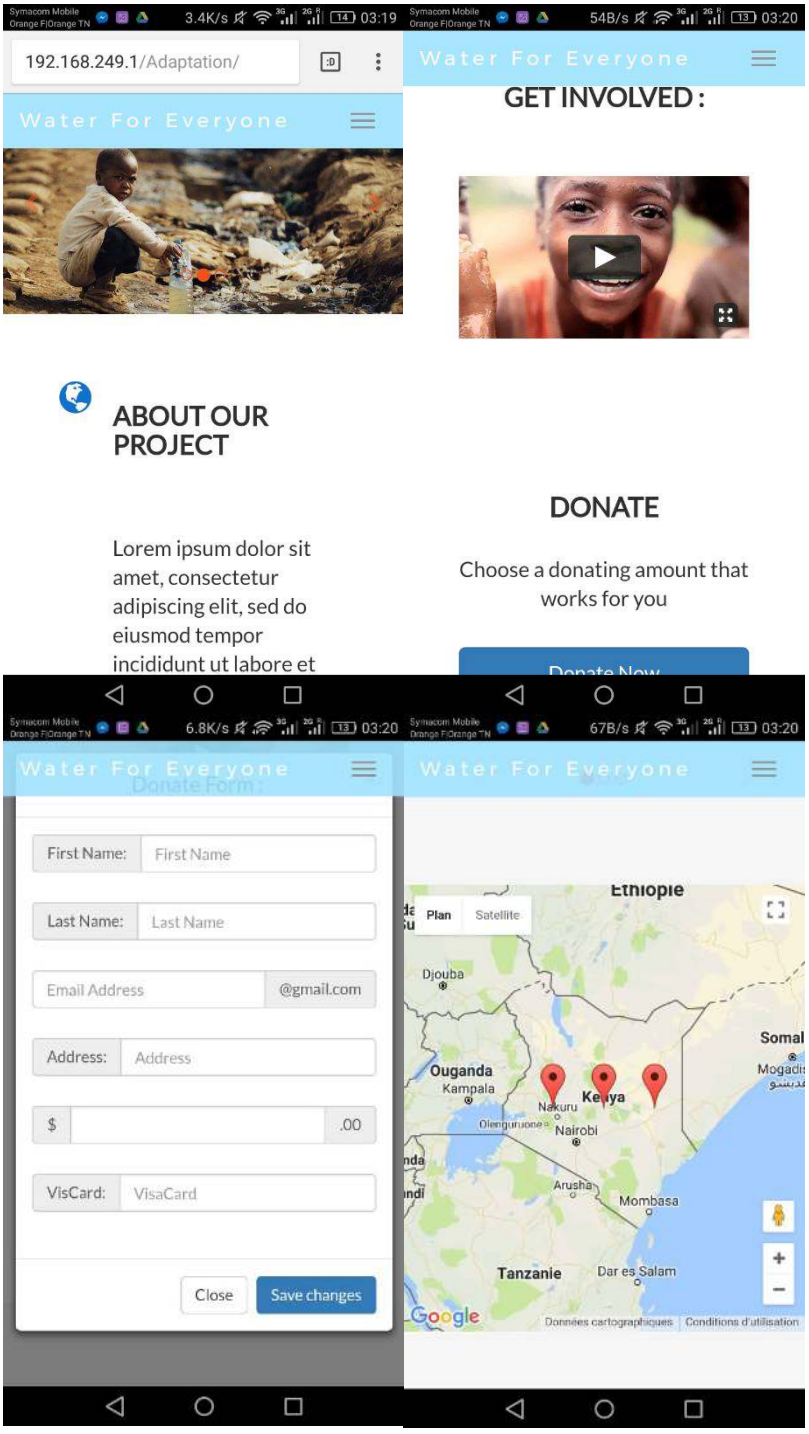


Figure 24 : Résultat final lors de la visualisation sur smartphone

Avantages:

- Facilité et Accessibilité
- Structure et méthodologie
- Documentation disponible
- Vitesse de développement accrue idéale pour le prototypage
- Framework porté vers le futur
- Grille fixe et fluide
- Personnalisable et Modulable
- Compatibilité sur les différents navigateur web

Inconvénients :

- Rendu des composants parfois limites sous IE
- L'héritage de classes peut rend l'application plus lourde
- L'utilisation de la grille posera une contrainte au niveau du design
- Avoir un site avec un rendu unique ne sera pas aussi évident

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 combinées avec du CSS et des "Media Queries" c'était mal organisé, non agronomique et surtout d'une responsivité mal établie.

Pure CSS : Tutorial

Figure 25 :Logo pure css

Définition du Pure CSS :

Pure CSS est un outil du responsive web design. Il s'agit d'insérer des styles sur un code HTML afin de définir d'une manière précise le comportement de chaque élément de la page.

Comme le **Bootstrap**, Pure CSS permet de disposer ses éléments et faire l'adaptation des sites web. Néanmoins, il n'utilise pas de JavaScript ou d'autres technologies (Contrairement à Bootstrap).

Réalisation de l'exemple avec Pure CSS :

Au début, tous les éléments ont été placés dans un fichier HTML. En ajoutant une ligne (Cf. figure 1) au code, une liaison entre le code HTML et le code CSS (style.css) a été faite pour gérer le design du page HTML. On a évité de mélanger les deux codes dans un même fichier pour assurer une meilleure visibilité.

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

Figure 26: Ligne ajoutée au code

Ensuite, en utilisant les feuilles de styles (Cf. figure 2), le contenu HTML a été classé sur une page web et le site avec un affichage en mode desktop a été obtenu (Cf. figures 3, 4 et 5).

```css
body {
    background-color: #FFF;
    font-family: 'Raleway', sans-serif;
}

header {
    background-color: #466995;
    width:99%;
    height:60px;
    position:relative;
    text-align:left;
}

li{
    color: #FFF;
    font-family: Georgia;
    font-size: 16px;
    font-weight: 300;
    text-transform: uppercase;
    display:inline;
    width:70px;
    border: 1px solid #FFF;
    padding:10px;
    border-radius:30%;
}
```

Figure 27: Feuilles de styles

Figure 28: Aperçu du site obtenu

Figure 29: Aperçu du site obtenu

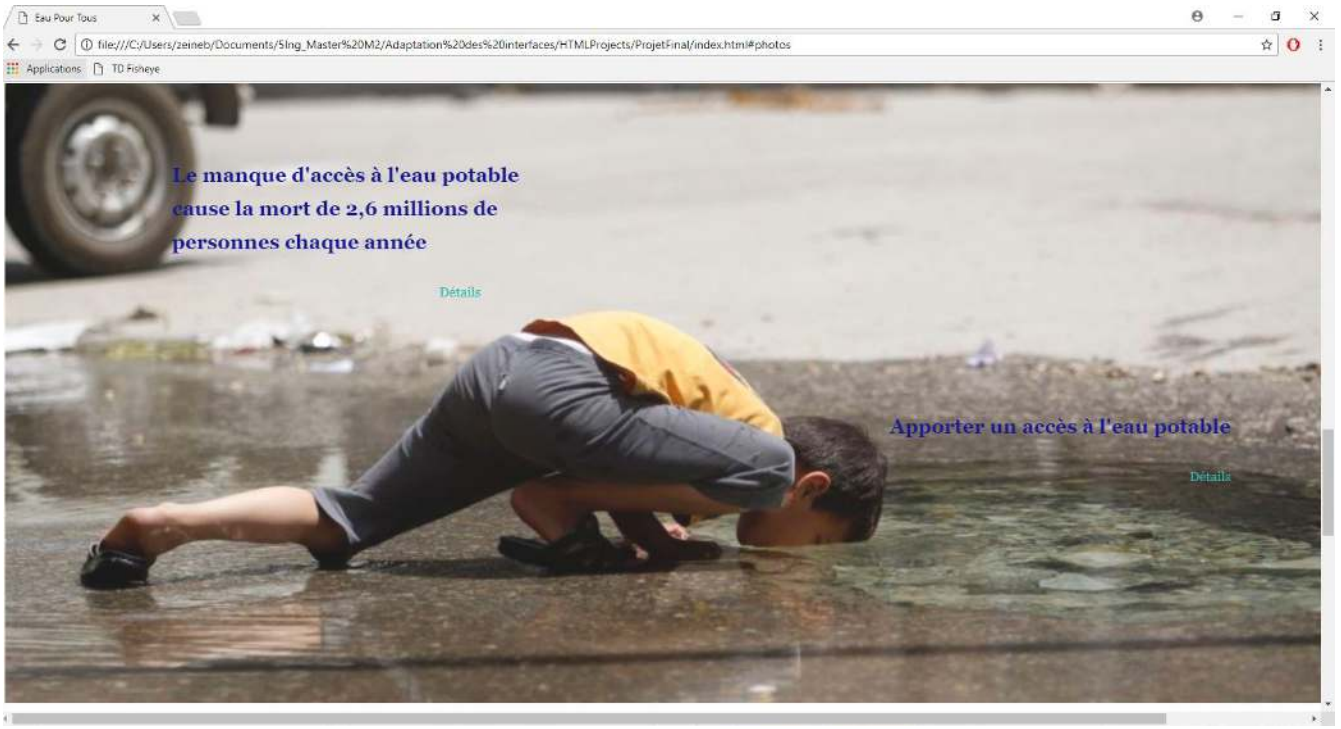


Figure 30: Aperçu du site obtenu

Difficultés rencontrées :

Dans cette partie, l'adaptation des écrans dont la résolution est inférieure ou égale à 1200 pixels a été difficile à gérer. En effet, des modifications au niveau de l'affichage du menu, des titres ainsi que des événements ont été produites et l'affichage est devenu illisible (le texte et les images sont écrasés au centre de l'écran). (Cf. figure6)

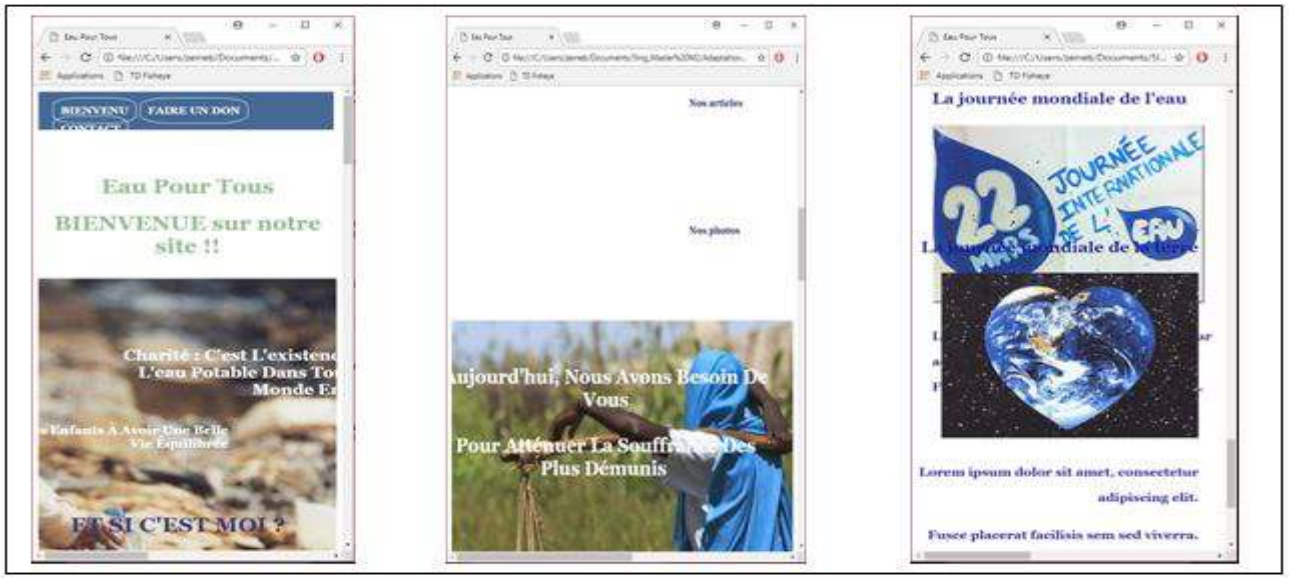


Figure 31: Aperçu du problème rencontré

Alors, pour faire l'adaptation à l'exécution de ses interfaces, on a eu recours au **Media Queries**. Il s'agit d'une fonctionnalité offerte par CSS afin de mettre en page le site web et appliquer les différentes règles de style CSS selon la taille, l'orientation ainsi que le ratio de dispositif.

==>Comment ajouter les Medias Queries au niveau du code ?

Dans notre cas, deux fichiers CSS ont été rajoutés au niveau du code HTML (Cf. figure 7). En effet, le premier fichier correspond aux écrans dont la largeur maximale est 440px. Le deuxième fichier est pour les écrans dont la largeur maximale est 1000px. Les tailles des écrans ont été précisées au niveau de l'attribut « media ».

```
<link href="./smallscreen.css" media="screen and (max-width: 1000px) " rel="stylesheet">
<link href="./smallscreen1.css" media="screen and (max-width: 440px) " rel="stylesheet">
```

Figure 32: Fichiers rajoutés au code HTML

Dans ces deux fichiers, les feuilles de style ont été modifiées afin de les adapter à un affichage en mode tablette et un affichage en mode Smartphone. Étant donné ce grand écart, il était très complexe d'assurer un design flexible.

Concernant l'affichage en mode tablette, le design était plus facile parce que les dimensions sont presque similaires. En effet, il s'agit des petites modifications au niveau des emplacements des titres, des boutons etc. Autrement dit, il n'y avait pas de modification des images ou de contenu, c'est juste la disposition des éléments qui a été reformée. (Cf. figure 8)

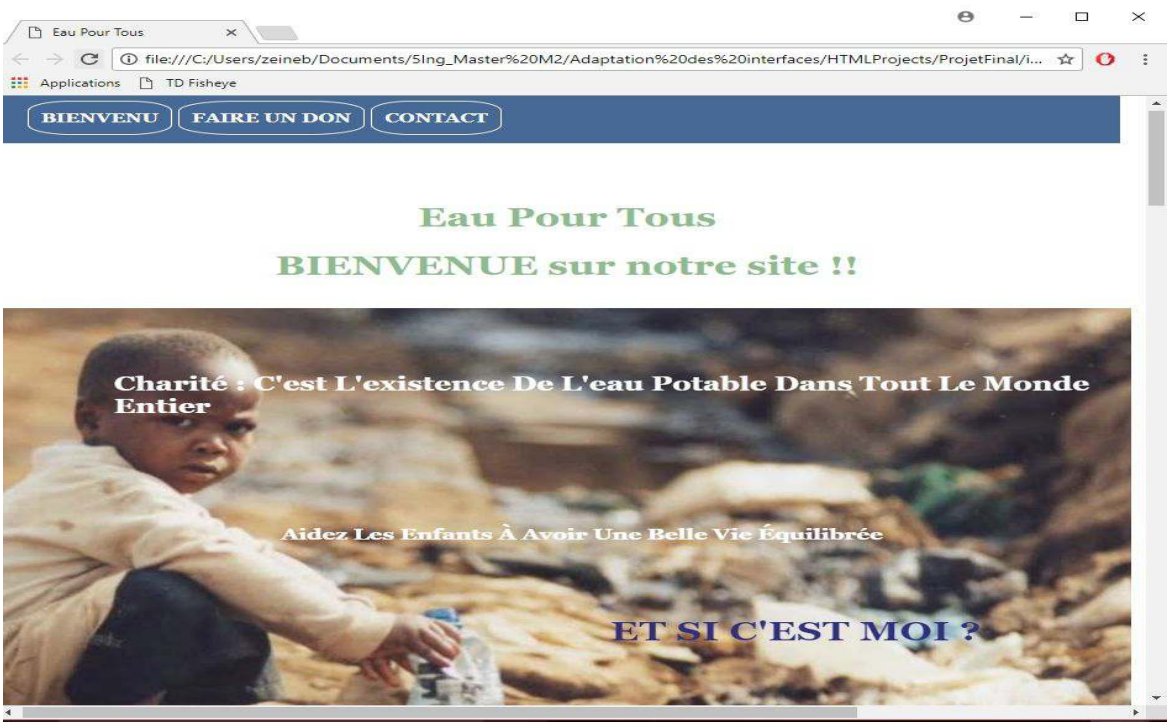


Figure 33: Affichage en mode tablette

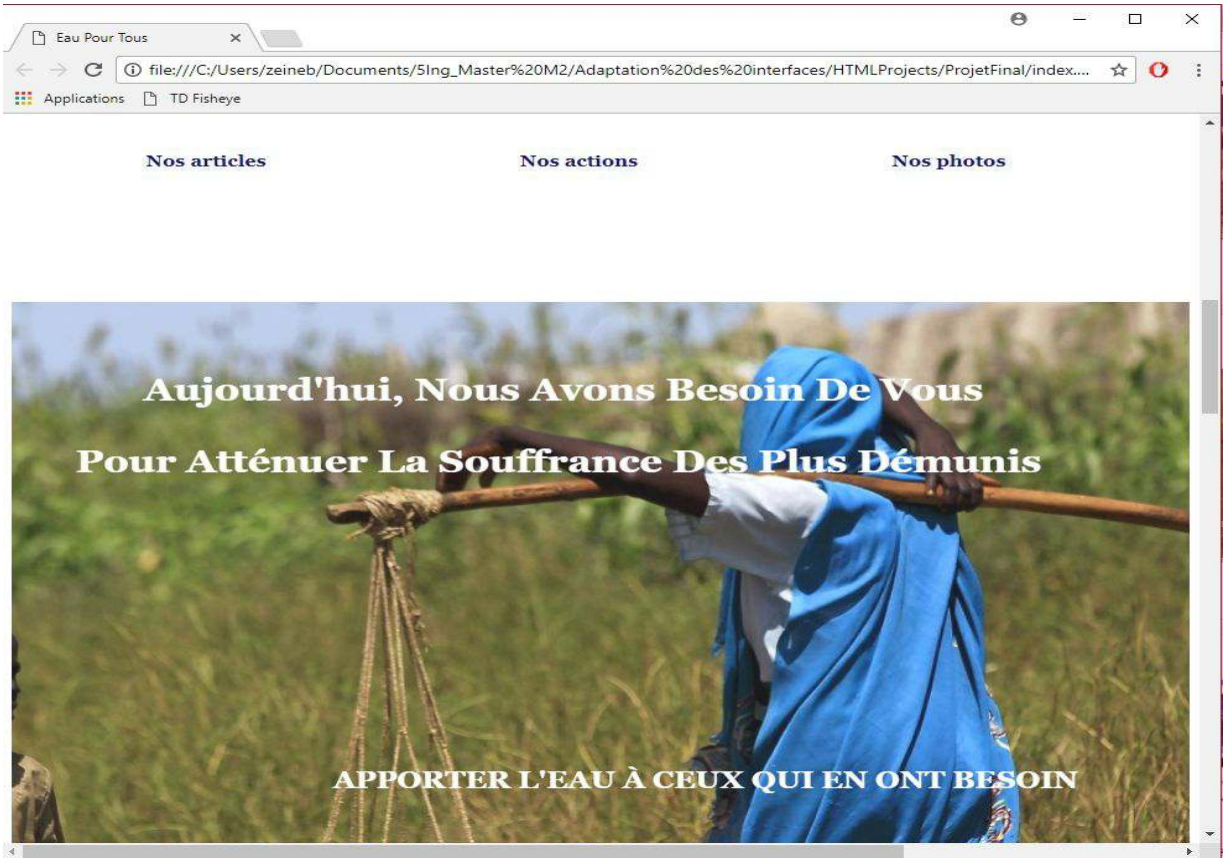


Figure 34: Affichage en mode tablette

Figure 35: Affichage en mode tablette

Par contre, pour l'affichage en mode Smartphone, il y avait plusieurs modifications pour mieux s'adapter à l'écran. Il fallait changer toutes les images en petites dimensions, la disposition des éléments, changement de la police et parfois élimination de certains éléments.

Dans notre cas, les éléments importants ont été gardés pour que le contexte du site web ne soit pas modifié. (Cf. figure 8, 9 et 10).

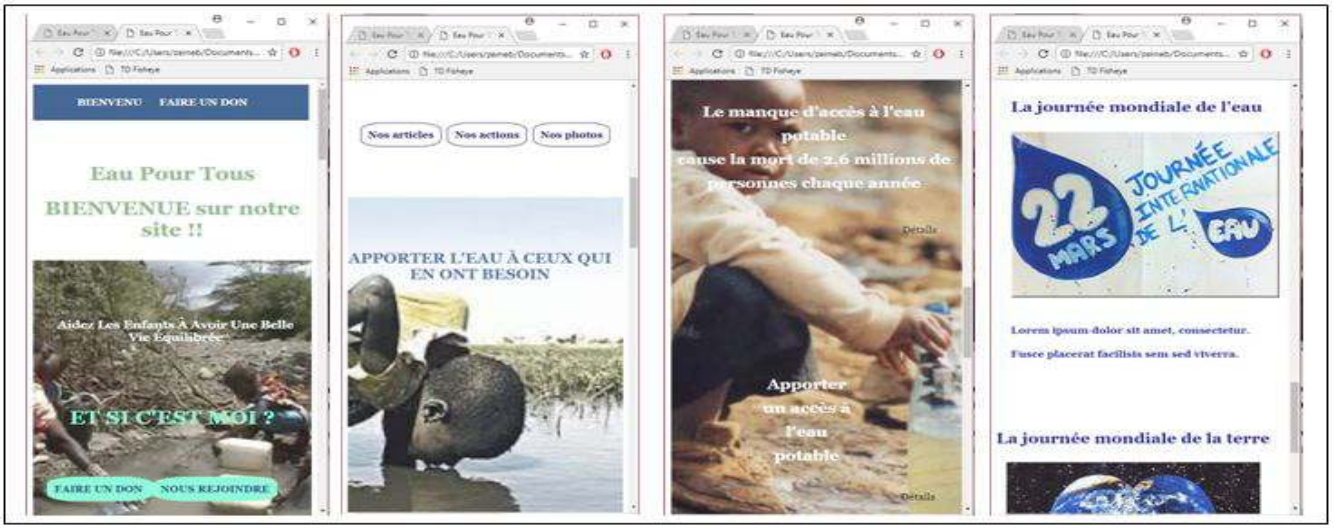


Figure 36: Affichage en mode smartphone

Sur la figure ci-dessus, toutes les images ont été changées (implémentation des images de petites

dimensions). Aussi, le bouton « Contact », étant au niveau du menu, a été éliminé. De plus, certains textes ont été reformés et disposés sur les images. Seulement les parties principales comme la consultation des événements, l'accès aux articles ainsi que faire un don (partie la plus importante dans notre site) ont été gardées.

Pendant le développement de l'exemple, on a pu constater plusieurs avantages et inconvénients avec l'utilisation de Pure CSS :

Les avantages :

* Simple à mettre en place.
* Léger et modulable.
*Layouts facile à utiliser
* Système de grille facilitant l'adaptation des éléments (comme Bootstrap).
* Sa simplicité diminue le temps de chargement de l'application.
* Avoir une page claire, facile à utiliser par les utilisateurs cibles.

Les Inconvénients :

* Peu minimaliste : on peut notamment citer le faite qu'il est insuffisant pour adapter une interface à toutes les différentes supports puisque le Media Queries définit les valeurs minimales et maximales à respecter pour le support.

==>Donc nécessite l'utilisation du JavaScript (comme Bootstrap).

*Adaptation des images choisies : modifier les images pour les petits écrans.

Conclusion :

Au cours de travail, une même application était développée avec des "Framerwork" distincts. En effet, chaque membre du groupe avait des objectifs particuliers à atteindre ainsi que des contraintes différentes. Ces trois différentes expériences, nous ont permis d'appréhender des nouvelles technologies. Pour cela, on s'est mis à place de l'utilisateur afin de répondre à ses besoins et lui adaptation qui répond à ses besoins.

Les 2 solutions : Bootstrap et Pure CSS restent très intéressantes, et proposent des fonctions au final très similaires, ils utilisent un système pour faire disposer ses éléments sans que CSS n'utilise pas de JavaScript ou d'autres technologies.

Conclusion générale

Ces trois expériences nous ont permis de découvrir et d'apprendre à maîtriser de nouvelles technologies. Mais aussi de réfléchir aux différents types d'interactions qu'il existe pour ensuite les mettre en œuvre. Cette mise en œuvre nous a également poussés à nous mettre à la place des l'utilisateurs pour faire en sorte que notre adaptation réponde au mieux à leur attente.

Avec ces frameworks nous avons développé la même application, avec des buts différents : réaliser du responsive web design, du cross-platform.

Ils permettent de développer rapidement une application (web ou smartphone), dont les interfaces sont toutes basées sur le principe de grilles. Ce principe est particulièrement utile pour avoir des interfaces "propres", utilisant le responsive design. Dans l'ensemble, ces frameworks nous ont offert les fonctionnalités annoncées, avec certaines limites à prendre en compte.

Ionic 2 est un framework pratique pour le cross-platform et les web components, mais il souffre encore d'une personnalisation irrégulière et de performances faible. Il propose néanmoins de bons composants générique qui ressemble aux composants natifs dans les cas simple.

Le principal atout de Bootstrap est qu'il permet de développer des sites et des applications web pour toute cible disposant d'un navigateur. Le produit développé est donc compatible avec des smartphones, des tablettes et ordinateurs.

Ionic 2 ne cible que les smartphones et c'est là l'avantage d'utiliser Bootstrap. En revanche, Bootstrap permet de créer des applications web qui ne peuvent être accessibles que par l'intermédiaire d'un navigateur et donc qui nécessitent d'avoir Internet. De plus, les produits développés avec Bootstrap sont faits en langage WEB et doivent être hébergés sur un serveur.

Un autre avantage concerne la facilité de développement, de test et de déploiement. Ionic 2 produit des applications mobiles qui doivent être compilées et être déployées sur des dispositifs, ce qui prend du temps. Ce n'est pas le cas de Bootstrap puisqu'il s'intègre directement dans une page HTML qui n'a pas besoin d'être compilée.

En revanche, Bootstrap ne permet pas d'accéder aux composants de l'appareil sur lequel il est

utilisé. Il est donc impossible d'accéder aux données de l'accéléromètre, du GPS ou à l'appareil photo du dispositif sur lequel il est utilisé.