Université Nice Sophia Antipolis

Rendu no 2
Rapport Phonograph

I. Introduction

L'application est un site web de vente de **vinyles**. Ce type d'application nous a permis d'utiliser tous les éléments d'IHM demandés dans le sujet comme les images, les tableaux, les menus, la gestion de formulaires et d'utilisateurs, etc..

En effet, l'application permet d'afficher sur la page principale une liste de vinyles sous forme de grille, chaque vinyle aura sa page de présentation.

Nous avons fait chacun l'application de notre côté, on a donc :

- Une application Angular 2
- Une application Ionic2
- Une application Bootstrap
- Une application Pure.css

II. Tutoriel par technologies

I. Angular 2 (Melchor Jérémy)

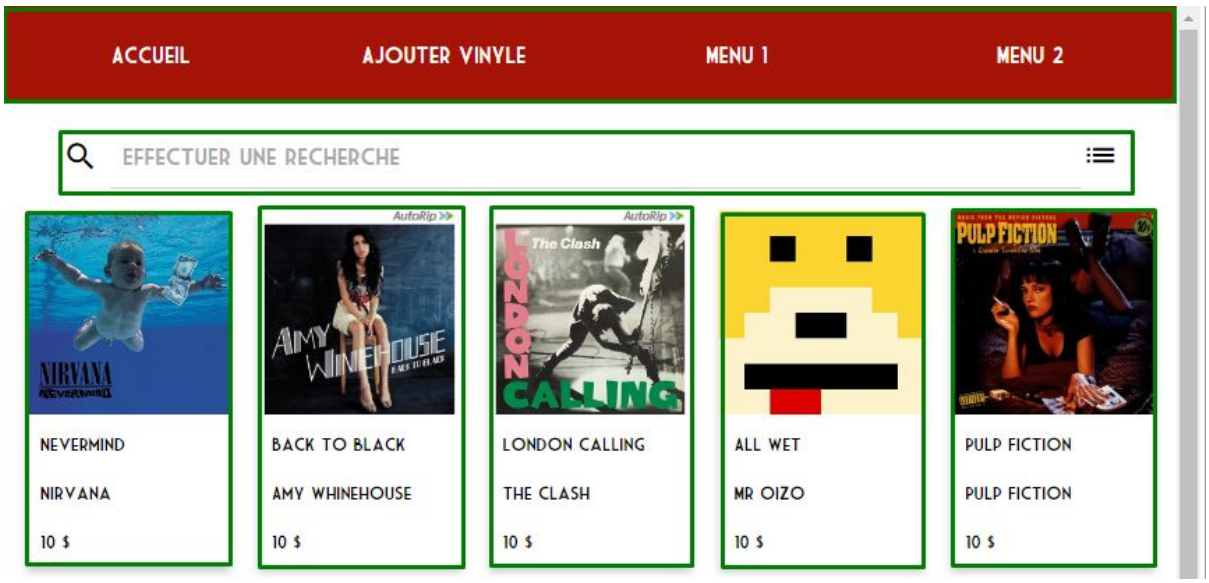
L'application web a été réalisée avec le Webpack Starter d'Angular 2, permettant de regrouper pleins de modules et fonctionnalités, telle que les modules d'angular 2, Typescript 2, TSLint etc… et permet d'installer n'importe quel module avec npm.

L'application a été développée en Typescript 2 (surcouche de Javascript ES6) avec le framework Angular 2 sur l'IDE WebStorm. Au niveau du CSS, je suis resté en CSS pure, celui-ci n'étant pas l'objet de mon exemple.

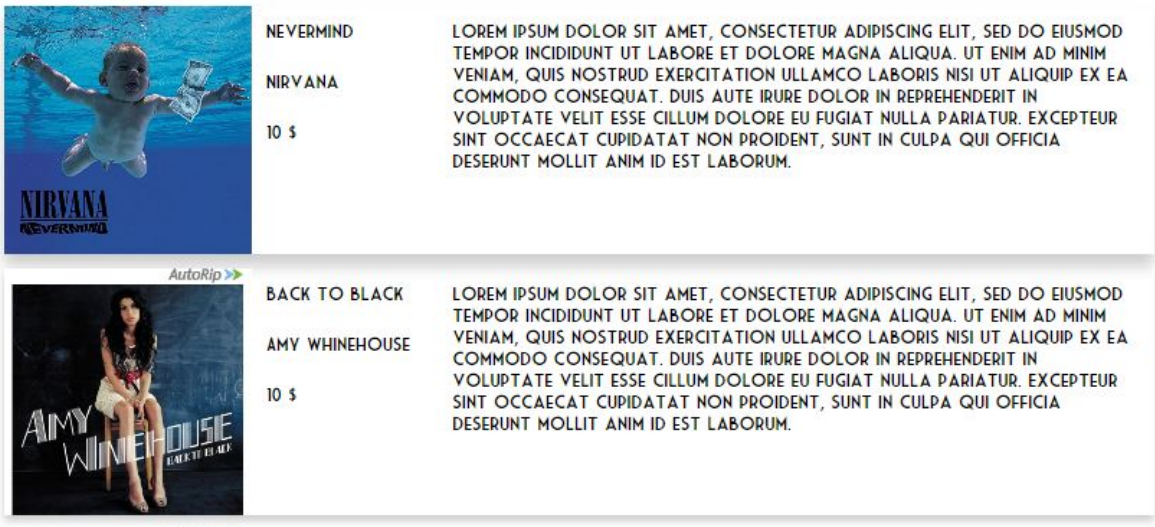
Le principe d'Angular 2 est de découper chaque partie de notre web application en composants. Comme montré sur la capture d'écran ci-dessous, chaque morceau encadré en vert est un composant. Nous avons donc 4 composants différents :

- Le composant de l'application elle même, qui contient tous les autres composants
- Le composant du menu de navigation, situé en haut du site

- Le composant de la barre de recherche et du changement d'affichage des vinyles
- Le composant vinyle, réutilisé plusieurs fois



Le point important de mon projet était la démonstration de la réutilisation des composants. Ici, le composant le plus réutilisé est le composant *Vinyle.* Au niveau du CSS, ce composant a été réutilisé pour être affiché sous forme de grille, ou alors sous forme de liste descriptive. C'est lorsque l'on passe en vue sous forme de liste qu'une description apparaît.



La plus grosse réutilisation du composant *Vinyle* est pour l'affichage détaillé de celui-ci : on peut cliquer sur un composant vinyle pour afficher son descriptif. Cette réutilisation est démontrée par la route : pour des routes différentes, on obtient le même composant avec des informations complètement différentes.

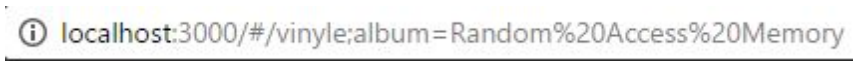
ACCUEIL AJOUTER VINYLE MENU 1 MENU 2

Album : All Wet

Artiste : Mr Oizo

Prix : 10 $

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



ACCUEIL AJOUTER VINYLE MENU 1 MENU 2

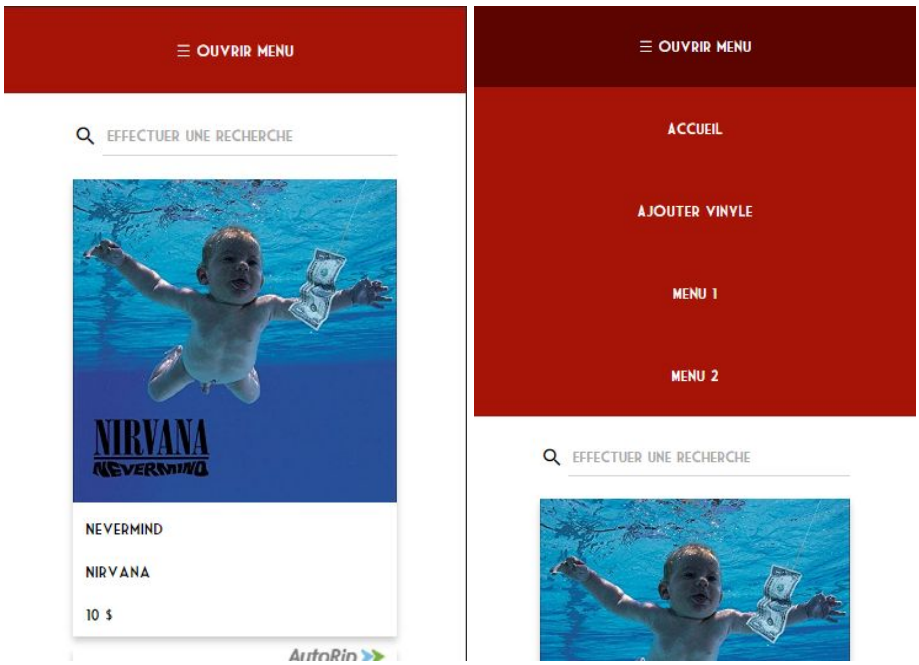
Album : Random Access Memory

Artiste : Daft Punk

Prix : 10 $

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Une adaptation mobile/tablette a été également faite :



En ce qui concerne le déploiement et l'exécution, on peut très bien faire tourner l'application en local. Pour la lancer, il suffit de faire un *npm start* dans le répertoire courant du projet et de se rendre sur http://localhost:3000/.

Au niveau des capacités d'adaptation, celle-ci sont testable avec l'outil développeur du navigateur qui permet de tester toute les tailles d'écrans possibles (mobiles/tablettes/desktop). Elles ne sont pas testables sur d'autres devices car il faudrait déployer l'application web.

II. PureCSS

Pour cette librairie CSS, j'ai voulu tester les différents éléments que propose PureCSS, à savoir le responsive sur une barre de menu, sur une grille d'éléments, et sur un tableau.

J'ai constitué un environnement simple, un fichier HTML par module testé, et une gestion de la navigation des pages en javascript.

J'ai configuré mon projet pour qu'il soit un projet PHP pour pouvoir le déployer sur un serveur hébergé par la plateforme Heroku.

Cette plateforme permet que lors d'un push du code sur le repository Git, Heroku le détecte, génère un build et déploie la nouvelle version.

Grâce à cela je peux tester directement sur mon smartphone, sans avoir à utiliser l'outil de développement des navigateurs sur l'ordinateur pour tester les différentes tailles.

L'IDE utilisé est PhpStorm de JetBrains.

Le menu:

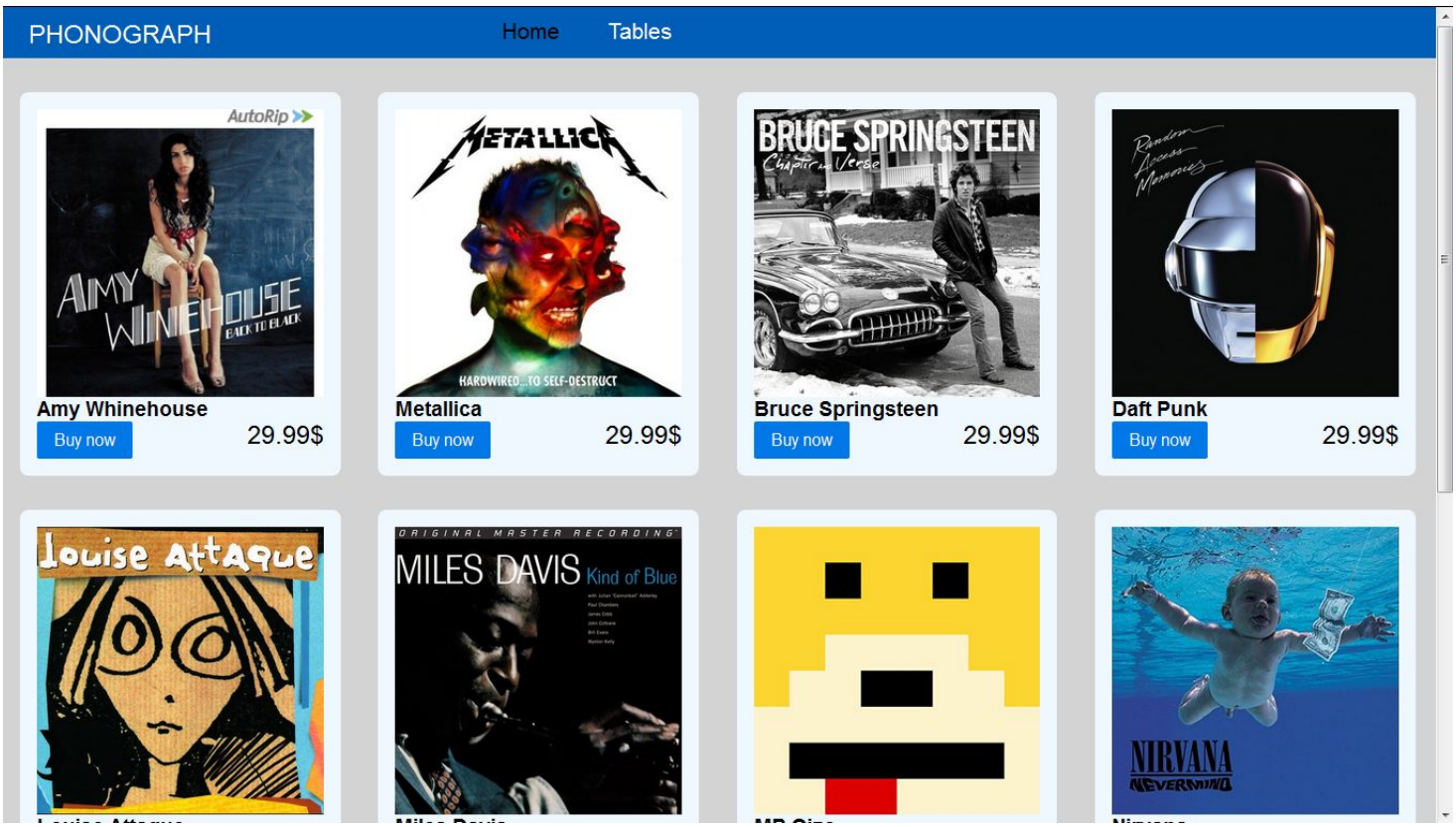
Le site est donc constitué d'un barre de menu simple, ici en version ordinateur:

Et en version mobile :

La librairie détecte la largeur d'écran et passé un certain seuil reforme la barre de menu à l'aide de fonctions javascript. Le menu devient alors déroulant:

Grille des éléments:

Conformément au cadre de notre exemple, j'ai utilisé le système de grille de PureCSS pour afficher une liste de vinyles disponibles, en affichant une photo de l'album, le nom de l'artiste, le prix et un bouton pour l'acheter.
Sur ordinateur:



Sur mobile:

On peut donc constater que le système s'adapte à la taille de l'écran lorsque qu'on est sur mobile. La même chose s'effectue pour les écran de très grande taille. Dans ce dernier cas, il faut simplement faire attention à ce que les images soient d'assez bonne qualité, celle ci étant aussi agrandie avec le reste.

Techniquement ce n'est pas très difficile d'appréhender le système, chaque classe possède un mot clé qui désigne comment va réagir la grille dans une situation (par exemple "md" pour toute taille d'écran moyenne, taille supérieure à 768px).

Ci dessous le tableau complet:

Key	CSS Media Query	Applies	Classname
None	*None*	*Always*	`.pure-u-*`
sm	`@media screen and (min-width: 35.5em)`	≥ 568px	`.pure-u-sm-*`
md	`@media screen and (min-width: 48em)`	≥ 768px	`.pure-u-md-*`
lg	`@media screen and (min-width: 64em)`	≥ 1024px	`.pure-u-lg-*`
xl	`@media screen and (min-width: 80em)`	≥ 1280px	`.pure-u-xl-*`

Il suffit donc de mettre une classe CSS pour chaque cas que l'on veut traiter. La dernière ligne du tableau montre que la librairie gère les écrans de très grande taille comme on a pu le voir, ce qui est plutôt un atout car certaines libraries CSS ne gèrent pas toujours ce type de taille.

Les tableaux:

PureCSS ne propose que des améliorations esthétiques sur les tables, ce qui fait qu'il n'y a aucun moyen fourni d'appliquer du responsive sur une table.

Sur ordinateur:

PHONOGRAPH Home Tables

#	Make	Model	Year	Year	Year	Year
1	Honda	Accord	2009	Honda	Honda	Honda
2	Toyota	Camry	2012	Honda	Honda	Honda
3	Hyundai	Elantra	2010	Honda	Honda	Honda
4	Ford	Focus	2008	Honda	Honda	Honda
5	Nissan	Sentra	2011	Honda	Honda	Honda
6	BMW	M3	2009	Honda	Honda	Honda
7	Honda	Civic	2010	Honda	Honda	Honda
8	Kia	Soul	2010	Honda	Honda	Honda
5	Nissan	Sentra	2011	Honda	Honda	Honda
6	BMW	M3	2009	Honda	Honda	Honda
5	Nissan	Sentra	2011	Honda	Honda	Honda
6	BMW	M3	2009	Honda	Honda	Honda
5	Nissan	Sentra	2011	Honda	Honda	Honda
6	BMW	M3	2009	Honda	Honda	Honda
5	Nissan	Sentra	2011	Honda	Honda	Honda

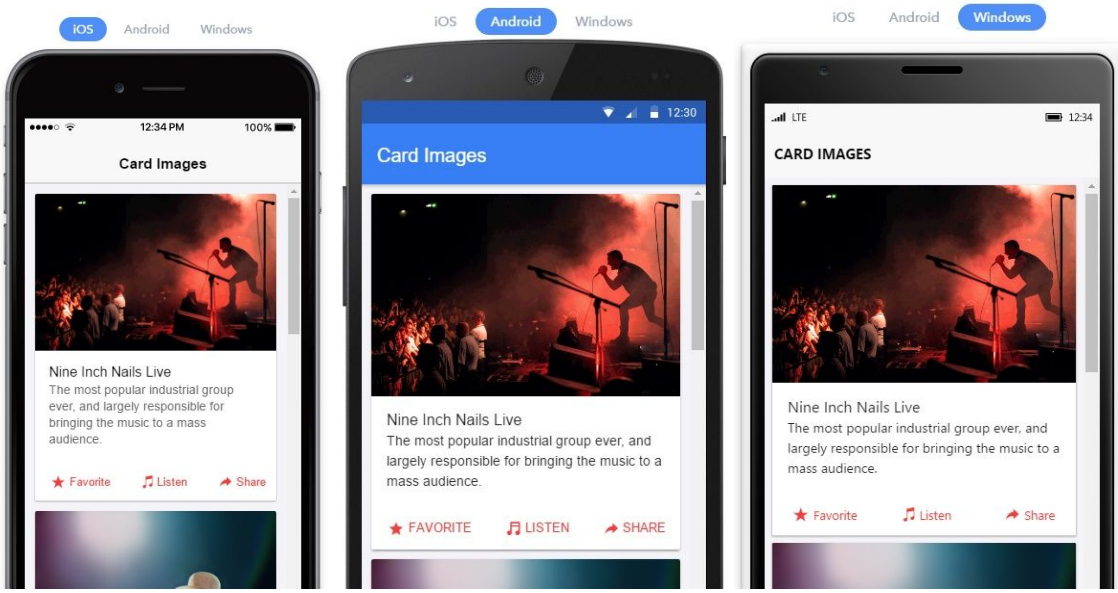
Sur mobile:

#	Make	Model	Year	Year
1	Honda	Accord	2009	Honda
2	Toyota	Camry	2012	Honda
3	Hyundai	Elantra	2010	Honda
4	Ford	Focus	2008	Honda
5	Nissan	Sentra	2011	Honda
6	BMW	M3	2009	Honda
7	Honda	Civic	2010	Honda
8	Kia	Soul	2010	Honda
5	Nissan	Sentra	2011	Honda
6	BMW	M3	2009	Honda
5	Nissan	Sentra	2011	Honda
6	BMW	M3	2009	Honda
5	Nissan	Sentra	2011	Honda
6	BMW	M3	2009	Honda

III. Ionic2 (El Koubi Ugo)

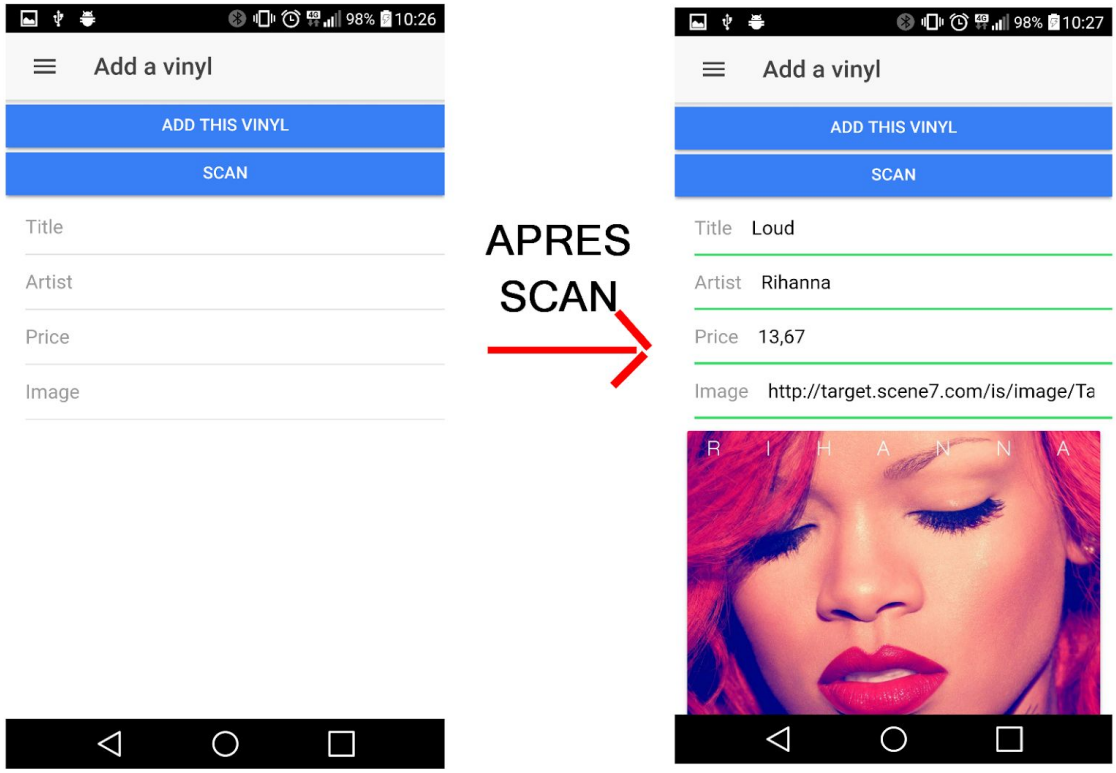
L'application a été réalisée avec le framework **Ionic2**. Ionic2 nécessite l'installation préalable de Cordova, permettant de build et de déployer l'application, mais aussi de Node.js pour une gestion simplifiée des packages.
A travers cet exemple du framework Ionic2, l'objectif était de montrer 2 types d'adaptations :

- Une adaptation de l'écran de type responsive. Dans notre cas, l'affichage grand écran sur une tablette doit montrer un tableau de vinyles, alors que l'affichage petit écran sur un smartphone doit afficher tableau à une seule colonne pour pouvoir scroller verticalement. De plus, la caractéristique principale de Ionic2 est de pouvoir proposer un **style d'affichage propre** à chaque OS (respectant la charte graphique de chaque), pour le même code source. Cependant, il faut avoir un ordinateur Mac pour pouvoir déployer sur iOS, et un téléphone sous Windows Phone pour cet OS. Je n'ai donc pas pu prendre de captures d'écran sur ces OS, mais voici un exemple des adaptations graphiques :



Comparaison de l'affichage sur les différents OS

- Une adaptation de l'utilisation pour comparer avec l'exemple Angular2, afin de voir le potentiel de ce framework et **l'utilisation de capteurs** propres aux devices portables (smartphone et tablette). Ici, j'ai exploité l'accès à l'appareil photo, afin de répondre à un cas d'utilisation. En effet, si je suis gérant du magasin de vinyles Phonograph, j'ai envie de vendre un nouveau vinyl sur le site. Or le fait remplir les informations à la main est un processus lent et ingrat, je préfère scanner le code barre du vinyle avec mon smartphone afin de récupérer les informations de l'album et la pochette automatiquement.



Il est possible de remplir manuellement les informations, mais un scan de l'album permet de récupérer toutes les informations nécessaires à l'ajout d'un vinyl.

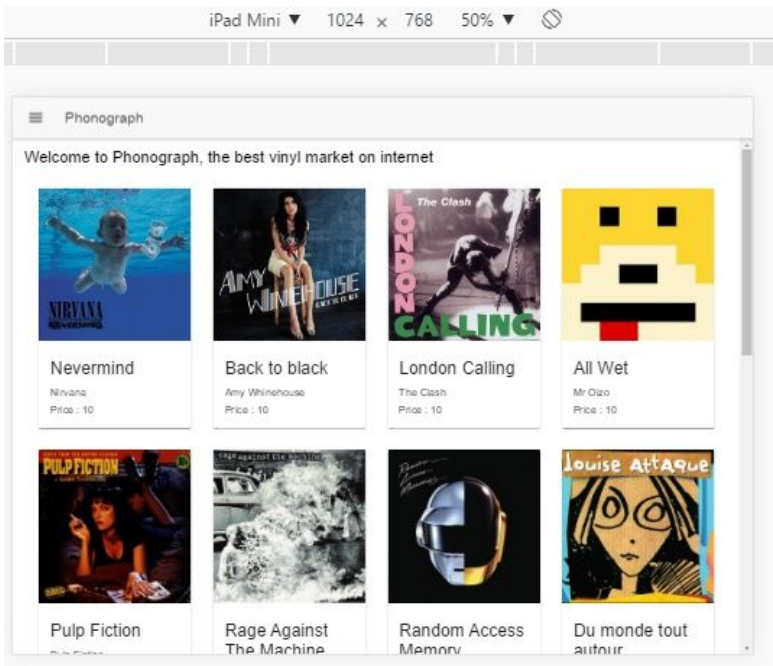
La fonctionnalité du scan a été implémentée grâce au plugin Cordova nommé **Barcode Scanner**. Il suffit d'installer ce plugin sur le projet grâce à la commande *ionic plugin add phonegap-plugin-barcodescanner*. Il permet de récupérer l'UPC (Universal Product Code) en scanner un code barre.
Maintenant que la lecture de code barre était fonctionnelle, il fallait trouver une API ou un web service qui renvoyait des informations sur un produit à partir de son UPC. Après plusieurs recherches, j'en ai conclu qu'il existe plusieurs solutions fiables et performantes mais elles sont pour la plupart payantes. J'ai fini par utiliser l'api UPCItemDB (http://www.upcitemdb.com/api/) qui permet une utilisation gratuite si le nombre de requêtes n'est pas supérieur à 100 par jour.
Le processus est simple, j'envoie une requête HTTP Get avec l'UPC en paramètre, et l'API me renvoie un JSON regroupant plusieurs informationss sur le produit.

L'outil de développement adopté est **WebStorm** de Jetbrains, car d'une part cet IDE est récent et donc supporte les nouveaux langages ES6 et Typescript utilisés par Ionic2, et d'autre part il permet de lancer les commandes nécessaires aux tests et au déploiement directement depuis l'éditeur, ce qui fournit un **gain de temps** notable.

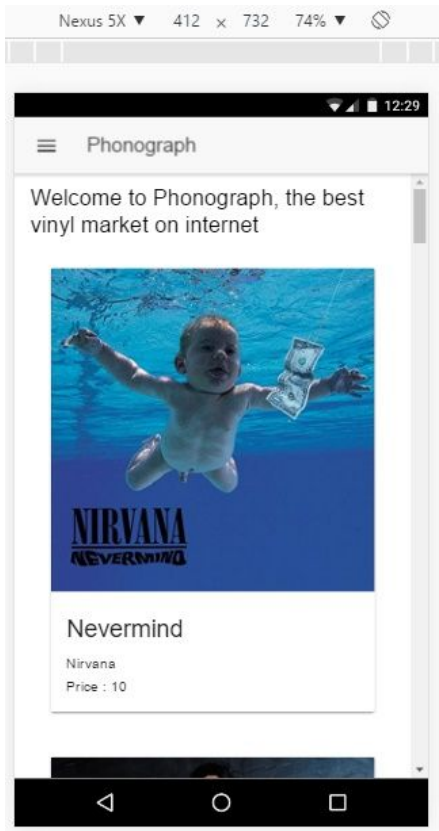
Au cours du développement, j'ai testé l'application de 2 manières différentes :

- Grâce à la commande *ionic serve* qui permet de lancer l'application localement, et donc de la visualiser dans un navigateur web à l'adresse localhost avec un port défini. Dans le cadre de cet application, j'ai utilisé Google Chrome et son outil de

développement afin de voir facilement le rendu sur différentes résolutions (smartphones et tablettes) et ainsi vérifier que l'adaptation est responsive pour ces 2 cas d'utilisation.



Affichage sur Ipad Mini (Device de type tablette)



Affichage sur Nexus 5X, device de type smartphone

- 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 (comme l'appareil photo). 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.

IV. Bootstrap (Sherif Meimari)

Comment avez vous réalisé l'exemple?

Brièvement, j'ai commencé mon projet par télécharger les fichiers Css et les fichiers Javascript déjà offerts par Bootstrap y inclus les classes et les components que j'aurais besoin pour coder ma page Web.
Le test était fait tout simplement sur mon navigateur utilisant l'émulateur et c'était le cas pour les écrans de différentes tailles commençant par le mobile vu que c'est un framework "mobile first" (mobile, tablette, Desktop, et plus large).

<u>Une page d'accueil:</u>

Pour l'adaptation et la responsivité, Bootstrap utilise le système grid habituel avec des points de ruptures déjà implémentés dans les fichiers.

J'ai commencé donc par la page d'accueil avec des éléments de base:

1. **Le menu (barre de navigation)**

Une barre de navigation qui s'adapte à tout écran:

Mobile:

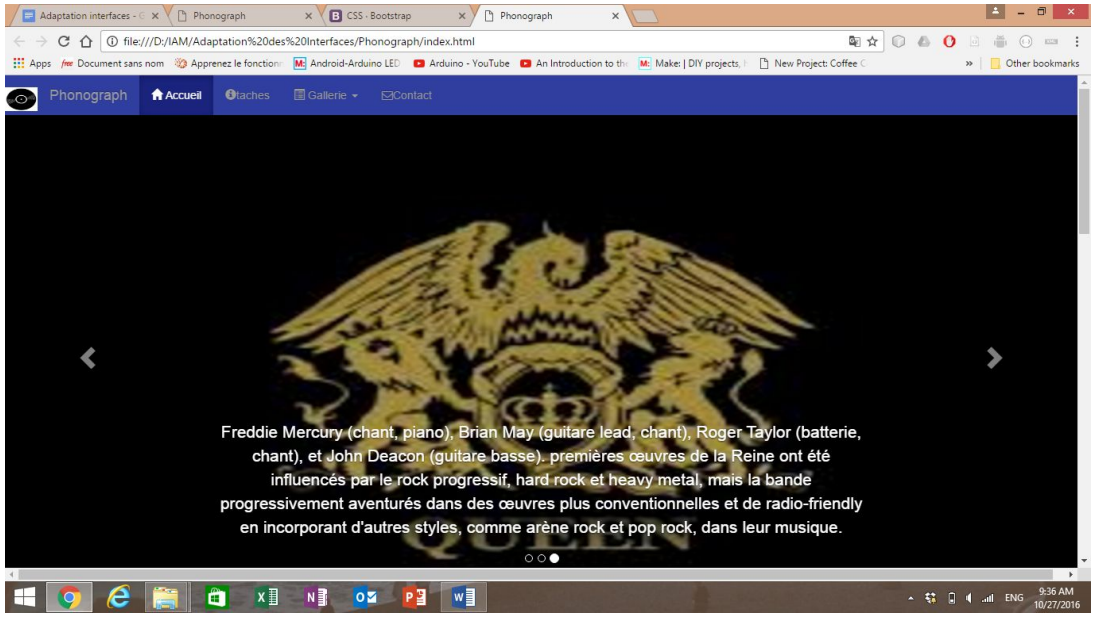
Ordinateur:

On voit bien l'adaptation du menu sur les différents écrans.
En fait, pour qu'elle s'adapte, j'ai simplement utilisé la classe navbar et ses sous-classes (ex: navbar-collapse) faisant intervenir le Javascript pour l'adaptation (apparition et disparition des éléments du menu par un bouton) sans coder une seule ligne de javascript, grâce au support de JQuery pour les Javascript components de Bootstrap. Donc tout ce que j'avais à faire c'est d'utiliser les classes du framework en précisant la taille d'écran (ex: col-xs-6 col-sm-4 col-md-3 col-lg-3) et en l'enrichissant avec les glyphicons, Icon Fonts (Font Awesome) et Social buttons.

2. Carousel

Mobile:

Ordinateur:

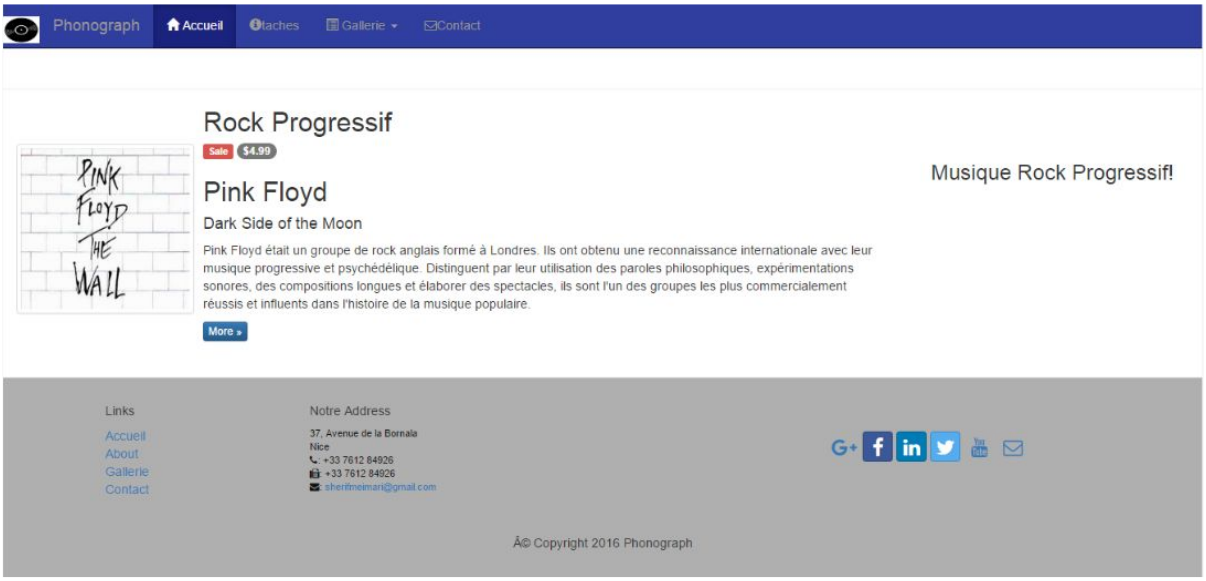


On voit bien une adaptation de la carousel sur les différents écrans pour l'image et le background. Une telle adaptation est automatiquement faite en changeant la taille de l'écran vu que la hauteur (height) de la carousel ne s'adaptait pas le mieux pour les écrans larges au début. Pour cela, je devais intervenir avec un fichier mystyles.css ajoutant des media queries. D'une autre part, la carousel de Bootstrap avait des limites d'adaptation à l'écran: on voit bien la disparition du paragraphe dans les écrans petits (mobile). J'étais obligé à le faire disparaitre pour cette taille d'écran et laisser juste le titre afin de s'adapter utilisant la classe ("hidden-xs") ou bien j'aurais du changé le design pour tous les écrans.

3. Media (image avec titre et légende)

Au début, j'avais beaucoup de soucis pour adapter la légende avec les tout petits écrans vu qu'elle se trouvait à la droite de l'image (media-object media-left media-middle):

Ordinateur:



Mobile (Avant):

Mobile (Après):

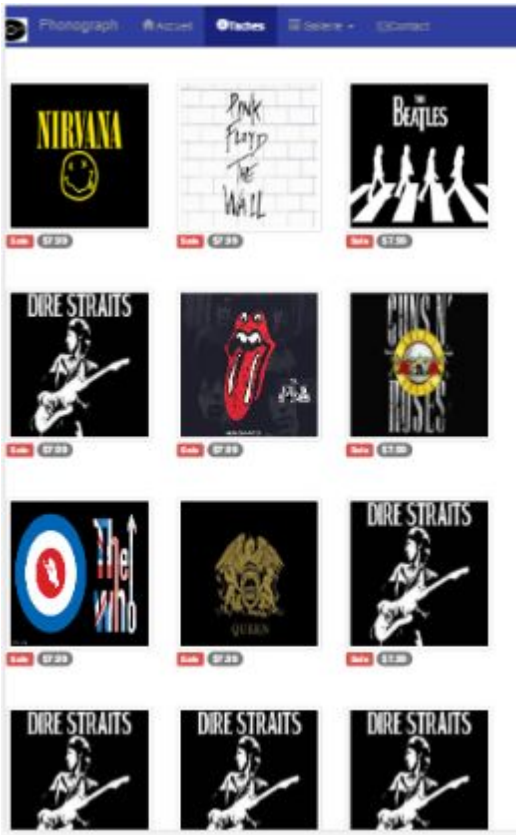
Trouvant une solution de ma part, j'ai fait disparaître la légende pour les petits écrans et j'ai ajouté le code pour ces derniers, mais avec la position de la légende au-dessus de l'image. Dans cet exemple, on dirait que j'avais moi-même du problème pour m'adapter aux components de Bootstrap.

<u>Une page de tâches à comparer avec PureCss:</u>

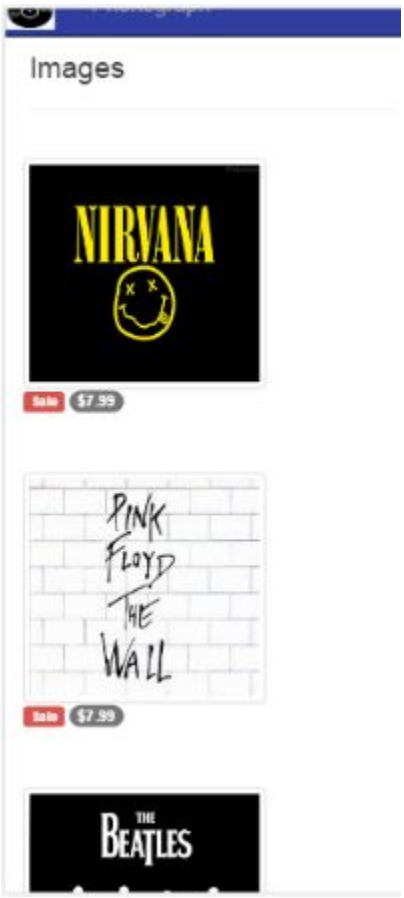
Dans cette page, j'ai pris charge de construire deux tâches, autres que le menu de la page d'accueil, pour comparer leur réalisation et adaptation avec mon collègue (Matthieu)travaillant avec PureCss.

1. Une grille d'images

Ordinateur:


Phonograph
Accueil
Tshirts
Galerie
Contact
NIRVANA
PINK FLOYD THE WALL
THE BEATLES
DIRE STRAITS
GUNS N' ROSES
The Who
QUEEN
DIRE STRAITS
DIRE STRAITS
DIRE STRAITS
DIRE STRAITS

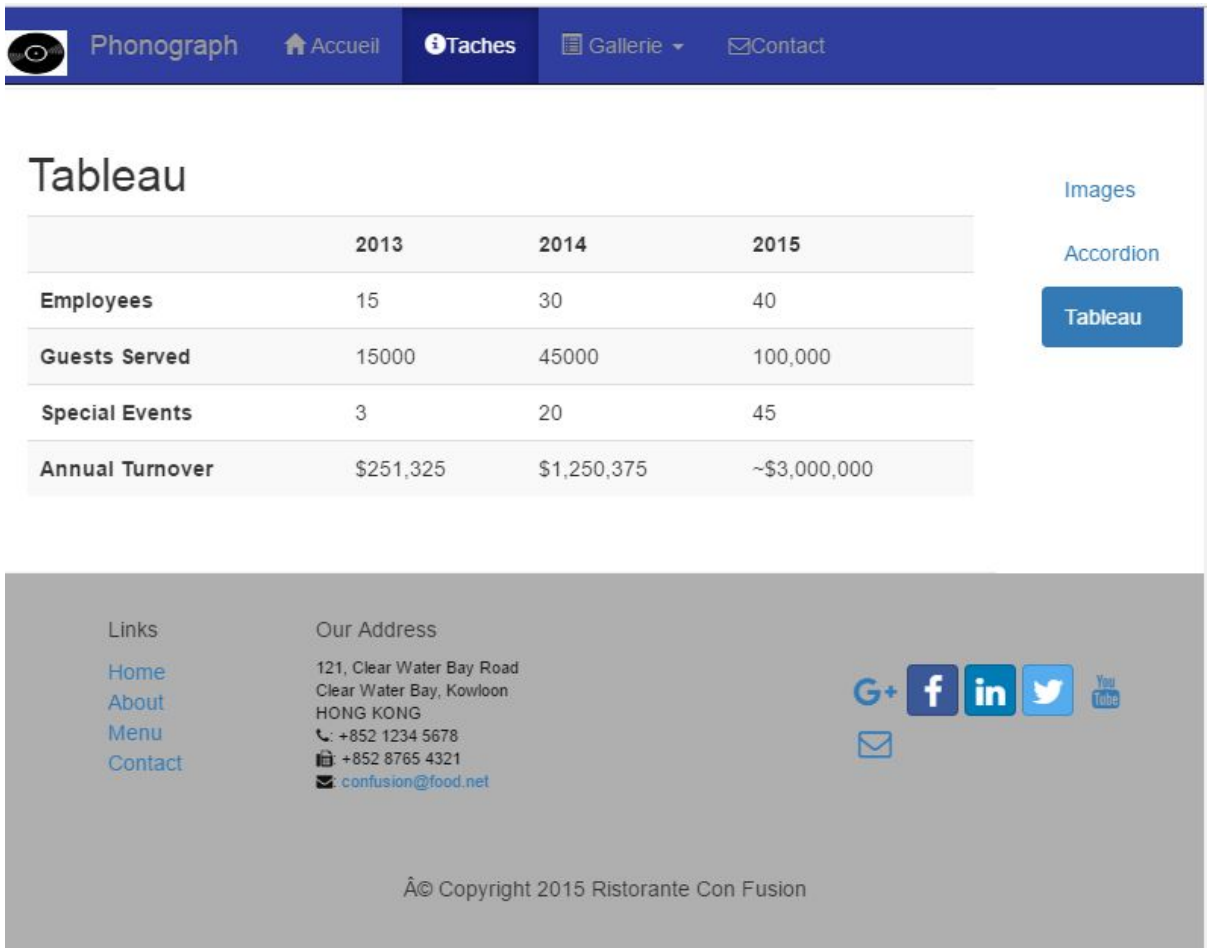
Mobile:


Images
NIRVANA
Sale
$7.99
PINK FLOYD THE WALL
Sale
$7.99
THE BEATLES

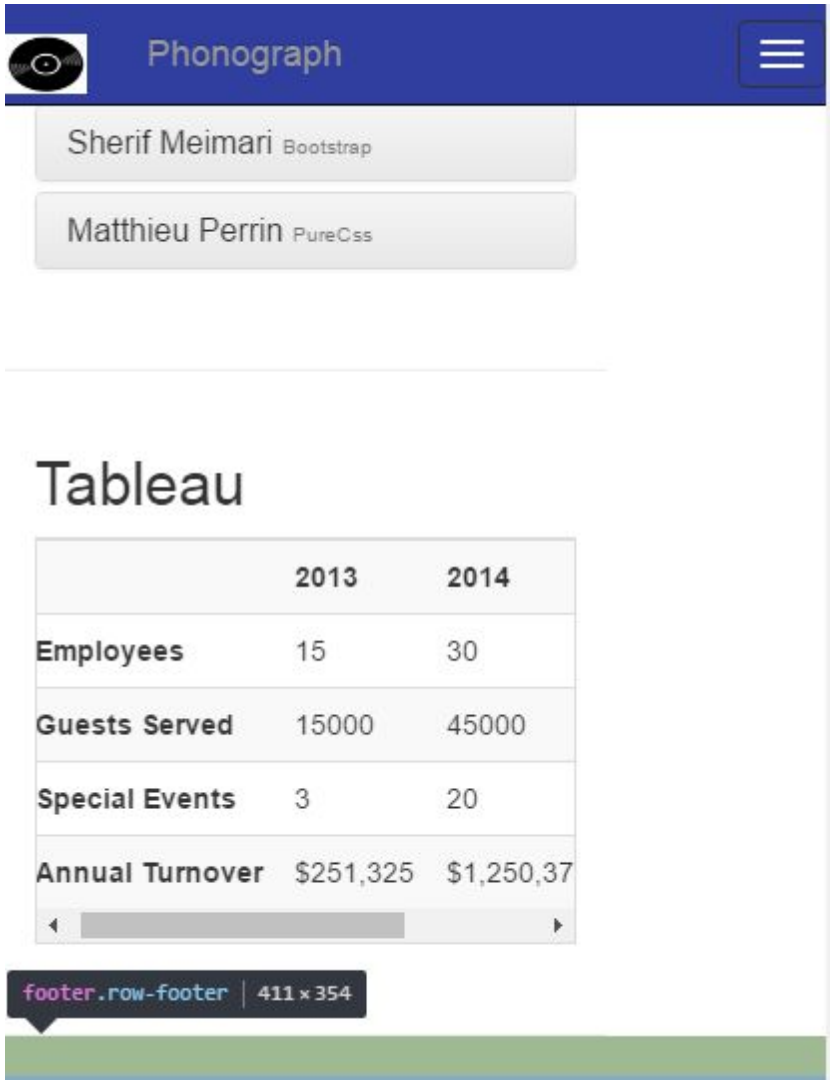
Comme on le voit, les images s'adaptent sur tous les écrans (img-responsive) mais il faut faire attention à la qualité de celles ci en élargissant l'écran.

2. Tableau

Ordinateur:



Mobile:



Bootstrap offre des classes pour les tableaux pour être bien responsives “table-responsive” et on le voit bien pour les petits écrans avec l’apparition d’un Scrollbar pour s’adapter. En fait, les classes sont très bien construites pour la responsivité mais, il n’y a pas beaucoup de styles à utiliser et pas beaucoup d’options. De même, la classe “table” ajoutée au tag <table> précise que les éléments du tableau sont tabulaires.

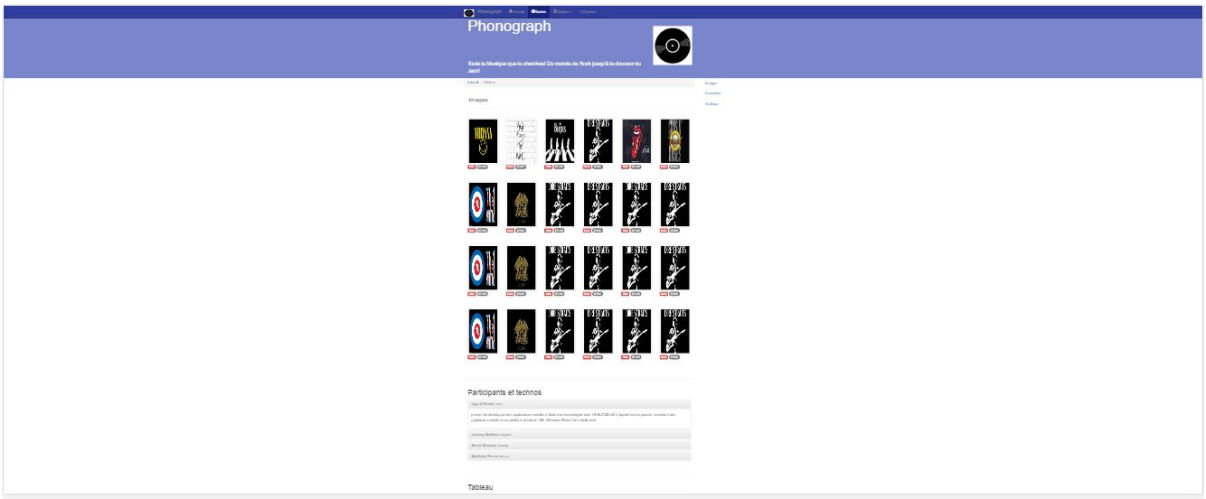
3. Autres

A la fin, je me suis intéressé à utiliser les outils de javascript prédéfinis par Bootstrap supportant JQuery comme les modals, ScrollSpy et Affix plugins offrant une bonne navigation à l’utilisateur, où l’usage du Javascript améliore beaucoup l’animation sur la page.

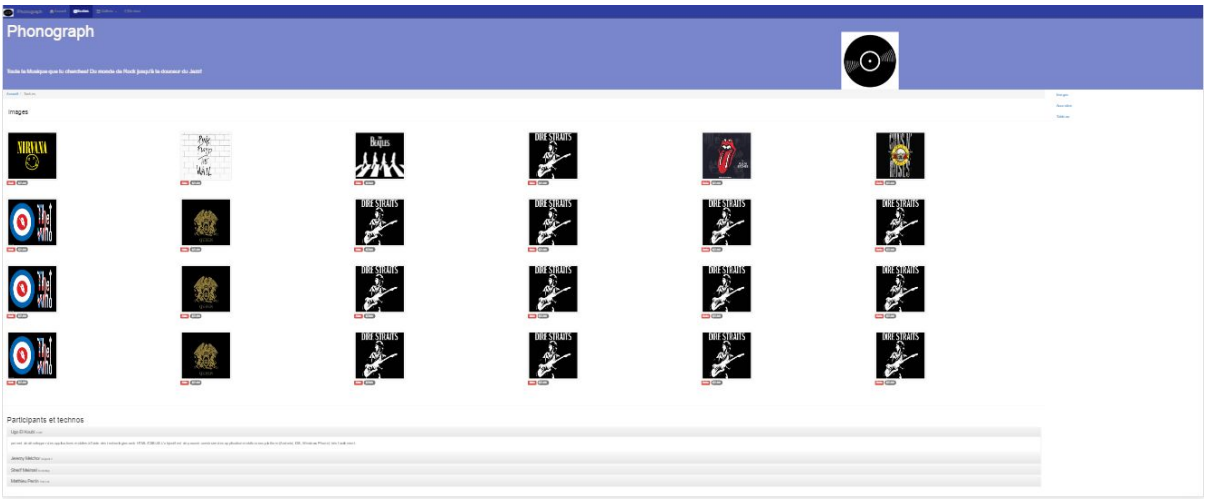
4. Larges écrans (> 1200px)

En fait, j'ai testé l'adaptation pour les plus larges écrans et afin d'élargir la page, la propriété flexible "fluid" des classes "container-fluid" et "row-fluid" étaient d'une grande importance sauf que la taille d'écriture n'est pas tout à fait réglable (toujours à tester).

Container:



Container-fluid:



V. Conclusion

Comparer les frameworks basé sur des web composants tels qu'ionic 2 et Angular 2 avec des frameworks de mise en page n'aurait aucun intérêt, leur utilisation étant complètement différente : l'un est pour ajouter de la fonctionnalité, l'autre est pour styliser le site.

PureCSS et Bootstrap:

Ces deux librairies ont une utilisation différente. Bootstrap est bien plus complet que PureCSS, cependant cette dernière n'a pas pour but de proposer un set complet d'outils, mais plutôt un package de base, léger et facile à surcharger.

Ionic2 et Angular:

Ionic2 ressemble fortement à Angular 2 : ce sont tous les deux des frameworks basé sur les web components. Ionic2 utilise également le framework Angular 2 mais il a aussi la possibilité d'accéder à tous les capteurs du téléphone ainsi que de rendre chaque élément d'affichage compatible mobile grâce à une redéfinition des éléments html.
Ce plus d'Ionic2 permet des interactions totalement différente pour la même action : par exemple, sur Angular 2, je peux ajouter un album à la liste des albums avec un formulaire en rentrant toute les informations. Alors qu'avec Ionic, on peut accéder à l'appareil photo du smartphone, afin de prendre en photo le code-barre et que le vinyle s'ajoute automatiquement à la base de donnée.
Cette surcouche de fonctionnalité pousse Ionic devant Angular 2 en ce qui concerne de l'adaptation pour mobiles et tablettes, surcouche non forcément utile sur desktop.