Adaptation des interfaces à l'environnement
Groupe 3, Rendu 1

I. Membres du groupe

- Théo Donzelle, theo.donzelle@gmail.com, **Angular2**
- Arnaud Garnier, arnaud.garnier@etu.unice.fr, **Pure CSS**
- Chloé Guglielmi, guglielmi.chloe@etu.unice.fr, **Ionic**
- Lucas Sauvage, lucas.sauvage@etu.unice.fr, **Bootstrap**

II. Description de l'application

Sujet de l'application : Notre application sera une plateforme de partage de recettes de cuisines.

Fonctionnalités proposées :

L'application sera composée de plusieurs pages accessibles via un menu. Ces différentes pages peuvent être découpées en différentes fonctionnalités :

- Afficher une recette :
  - Titre
  - Temps de préparation
  - Temps de cuisson
  - Nombre de parts
  - Liste des ingrédients et quantité (visualisé dans un tableau)
  - Niveau de difficulté de la recette
  - Ustensiles particuliers
  - Vidéo explicative (optionnelle)
  - Note des utilisateurs sur 5
  - Photo du résultat
  - Prix par part
  - Texte explicatif
- Entrer une recette (formulaire)
- Visualiser les recettes les plus populaires (page d'accueil)
- Recherche une recette

Certaines de ces pages intégreront des fonctionnalités supplémentaires. La page permettant d'afficher une recette offrira la possibilité à l'utilisateur de noter et commenter la recette. De plus, nous essayerons d'intégrer des emplacements pour des publicités sur la plupart des pages.

III. Liens entre l'application et l'adaptation

Pour notre application, nous aurons différentes adaptations. Toutes ces adaptations ne seront pas forcément implémentées pour toutes les technologies. En effet, certaines adaptations n'auront aucun intérêt ou ne pourront pas être implémentés suivant la technologie traitée.

Nos adaptations seront :

- Adaptation à la plateforme (responsive)
- Réutilisabilité du code
- Utilisation des fonctionnalités du dispositif (capteurs, caméra…)

Concernant les technologies, il existe actuellement 3 manières différentes de gérer l'adaptation des IHM : le développement responsive web design, le développement cross-platform et le développement Web Components.

1. Responsive web design

Le développement responsive web design comprend une multitude de technologies dont les plus connues sont : Bootstrap, Foundation, Pure CSS et Web Kit Framework. La technologie RWD permet aux sites web de s'adapter à tout type d'appareil de manière transparente pour l'utilisateur. Nous allons, pour notre part, utiliser Bootstrap et Pure CSS.

Les avantages de la technologie RDW sont :

- Coûts et délais inférieurs aux solutions « ad-hoc »
- Maintenance facilitée
- Mise à jour transparente
- Déploiement multi-plateformes
- Envisageable en reconception
- Sortent en premier dans les résultats Google

Ces inconvénients sont :

- Fortes connaissances techniques
- Nécessite des veilles technologiques constantes
- Importance des tests (« device labs », simulateurs)
- Difficile de contourner les limites ergonomiques et de performances des navigateurs web

Plus spécifiquement, les avantages et inconvénients de Pure CSS sont :

Avantages :

- Simplicité d'intégration à une page web (en ajoutant une feuille de style), et par conséquent un gain de temps assez important
- Utilisation de grilles de layout pour organiser les éléments
- Facilité d'utilisation de ces grilles

Inconvénients :

- Peu de support en ligne, dû à une communauté restreinte
- Facilité d'utilisation impliquant moins d'options disponibles
- Peu de templates disponibles en ligne

Pour Bootstrap les avantages et inconvénients sont :

Avantages :

- Gain de temps
- Facile pour faire du responsive design
- Utilisation d'une grille pour mettre rapidement en forme

Inconvénients :

- Tous les sites en Bootstrap finissent par se ressembler
- Volumineux
- Peut entrer en conflit avec la configuration existante

Dans le cadre de notre application, en utilisant une technologie RWD nous allons pouvoir facilement adapter notre site de recettes de cuisine aux différents supports qui existent. Ainsi, il sera possible d'avoir accès facilement aux différentes fonctionnalités que propose notre application, et ce, indépendamment du support sur lequel on consulte le site.

Si l'utilisateur est en déplacement, qu'il désire consulter notre application sur son téléphone, il sera plus simple pour lui de ne pas avoir à scroller sur la page à la recherche d'informations. La disposition des éléments devra être différent, afin par exemple de faciliter la visualisation des éléments d'une recette. Les besoins utilisateurs varient en fonction de l'environnement dans lequel il se trouve : si l'utilisateur est au supermarché par exemple, et qu'il recherche les ingrédients dont il a besoin pour composer sa recette, il n'aura pas forcément besoin d'avoir l'accès immédiat à certaines fonctionnalités (comme par exemple les instructions de la recette).

Cependant, on peut entrevoir certaines difficultés. L'une d'elles sera l'affichage de publicités : en effet, il faudra garder en tête que les publicités doivent rester bien visibles à l'écran, peu importe le type d'appareil utilisé. Un second problème est celui du débit : en effet, l'application sera développée sur ordinateur et avec une connexion stable. Les connexions des appareils mobiles ne seront pas forcément aussi bonnes, et il faudra donc faire attention à certains éléments (comme par exemple la qualité des images et des vidéos que nous allons ajouter). On aura donc des médias de faible qualité, même lors de la visualisation sur ordinateur.

2. Cross-platform

Le développement cross-platform comprend également une multitude de technologies dont les plus connues sont : Ionic, PhoneGap / Cordova, Xamarin, Native Script. Le développement coss-platform a pour objectif de permettre un développement d'applications mobiles pour plusieurs plateformes en réutilisant au maximum le code. Voici les avantages et inconvénients de cette famille:

Avantages :
- Diminution du temps de développement (et donc du coût)
- Viser un marché large
- Réutilisation de code
- Un seul langage pour les différentes plateformes

Inconvénients :
- Evolution fréquentes des systèmes ciblent
- Pas toujours accès à toutes les fonctionnalités du mobile (ce n'est pas le cas pour nous)
- Pas de gestion fine des performances

Nous allons, pour notre part, utiliser le Framework Ionic, qui permet d'accéder aux fonctionnalités et aux UI du natif. Cette technologie nous permet d'accéder aux fonctionnalités de l'appareil. Nous avons donc décidé d'en profiter pour permettre aux utilisateurs plus de fonctionnalités une meilleure ergonomie.

Nous pensons utiliser le capteur de position pour permettre à un utilisateur d'être localisé lorsqu'il poste une recette. Le fait de voir qu'ils sont de la même ville peut rapprocher les utilisateurs.

On peut aussi imaginer que les ingrédients que l'on peut trouver d'une ville à une autre peuvent différer. Ainsi, si un utilisateur poste une recette à base de fleurs de courge, un utilisateur du nord pourra voir que cette recette a été postée à Nice, et en déduire qu'ils n'ont pas les mêmes ingrédients à Lille.

Nous avons aussi pensé à l'utilisation de la caméra du Smartphone pour partager des photos et des vidéos. Au moment de poser une recette, on peut proposer d'ajouter une photo ou une vidéo depuis l'album ou depuis la caméra. Il serait aussi intéressant de pouvoir commenter une recette en postant une photo du résultat de la recette.

En plus de ces fonctionnalités, d'autres capteurs permettent à l'application de s'adapter à l'environnement. Par exemple, le capteur de luminosité permet de modifier la luminosité de l'écran pour qu'il reste lisible et l'accéléromètre permet de savoir si l'utilisateur a tourné son écran et de tourner l'application en conséquence. L'accès à ces capteurs est automatique sur le téléphone (indépendant de l'application), mais c'est à nous de permettre leur utilisation sur l'application (en proposant des couleurs qui restent visibles à tout moment, des vues pour lorsque l'écran est tourné…).

Enfin, expliqué dans la partie RWD, la disposition des éléments devra être différente selon l'environnement dans lequel se trouve l'utilisateur.

3. Web-components

Le développement Web Components comprend lui aussi une diversité de technologies dont les plus connues sont : React / ReactNative, Ionic 2, Polymer, Angular 2. Le développement Web Components permet de combiner plusieurs éléments pour créer des widgets réutilisables avec un niveau de richesse et d'interactivité sans commune mesure. Nous allons, pour notre part, utiliser Angular 2, en voici les différents avantages et inconvénients.

Ces avantages sont les suivants :

- Il offre une meilleure réutilisabilité, les différents composants créés peuvent être réutilisés dans différents contexte
- Il permet une meilleure maintenabilité, le code est plus structuré, mieux organisé
- Les pages sont moins lourdes au chargement, en effet les composants sont chargés au besoin
- De plus, la communauté est de plus en plus importante

Ces inconvénients sont les suivants :

- C'est une technologie encore assez récente et on ne sait pas encore vraiment comment elle va évoluer
- Pour la même raison, Angular 2 manque encore de stabilité, il est fréquemment mis à jour et peut donc poser des problèmes de compatibilité entre les différentes versions

En utilisant Angular 2, nous allons démontrer la réutilisabilité de notre code en fonction du contexte (différents supports, différents environnements d'utilisations, etc.).

Nous allons aussi mettre en valeur les évolutions possibles de notre application avec cette technologie. En effet le fait que le code soit très modulaire et bien structuré peut permettre d'ajouter facilement de nouvelles fonctionnalités à notre application. On pourrait imaginer vouloir rajouter, par exemple, la possibilité de poster des commentaires aux recettes afin de poser des questions au créateur de la recette. D'autres évolutions peuvent être envisageables, nous verrons donc comment l'utilisation de cette technologie peut les rendre plus simple à intégrer.