Projet de Programmation

Alexandre Kervadec, Guillaume Verdugo, Jeremy Arrestier, Thibaut Fabre

4 février 2015

Table des matières

Chapitre 1

Présentation du projet

1.1 Résumé

Le projet qui nous est proposé, porte sur la relation entre prosodie et expression des sentiments. Notre but est de réaliser une interface graphique permettant de réaliser des tests prosodiques sur des cobayes. Ces sujets auront à faire un choix, d'une vidéo parmi plusieurs, à fusionner avec une bande son parmi une autre liste. Ce mixe de vidéo/son devra donner une réponse à une question du genre : "Réaliser une vidéo qui exprime l'excitation."

1.2 Approche logicielle du projet

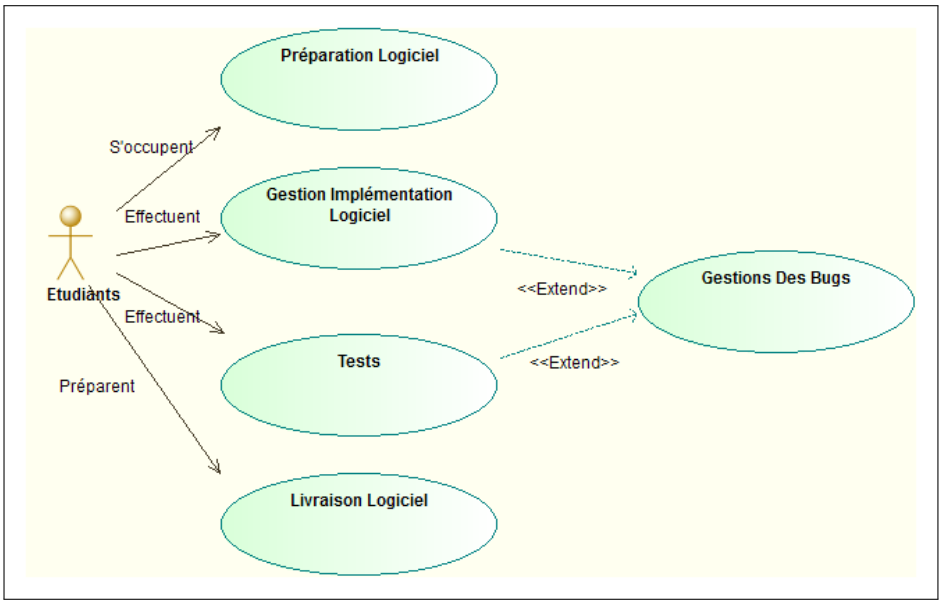


FIGURE 1.1 – Diagramme de cas d'utilisation - Approche logicielle

Ce diagramme de cas d'utilisation (FIGURE 1.1) présente l'approche génie logiciel du projet. Plus exactement les différentes étapes de développement de l'application.

1.3 Approche du projet avec les différents acteurs

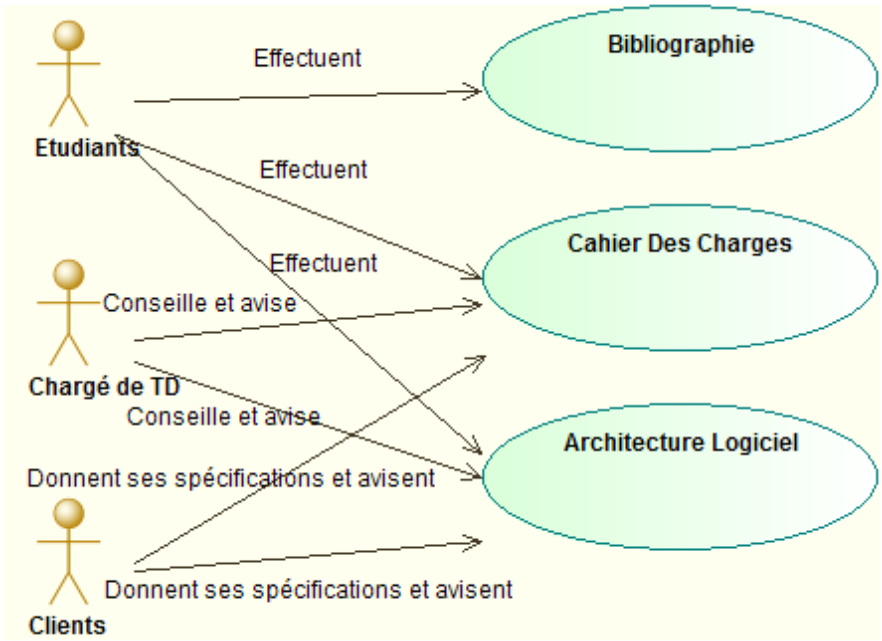


FIGURE 1.2 – Diagramme de cas d'utilisation - Approche Préparation projet

Ce digramme de cas d'utilisation (FIGURE 1.2) expose les intéractions entre les différents acteurs du projet.

1.4 Fonctionnement de l'application

La FIGURE 1.3 décrit le fonctionnement de l'application, *i.e.* les différents états dans lesquels il est possible de se trouver durant l'utilisation de l'application.

La FIGURE 1.4 est en rapport avec le diagramme précédent (FIGURE 1.3). Il décrit les relations entre les différents états de transition.

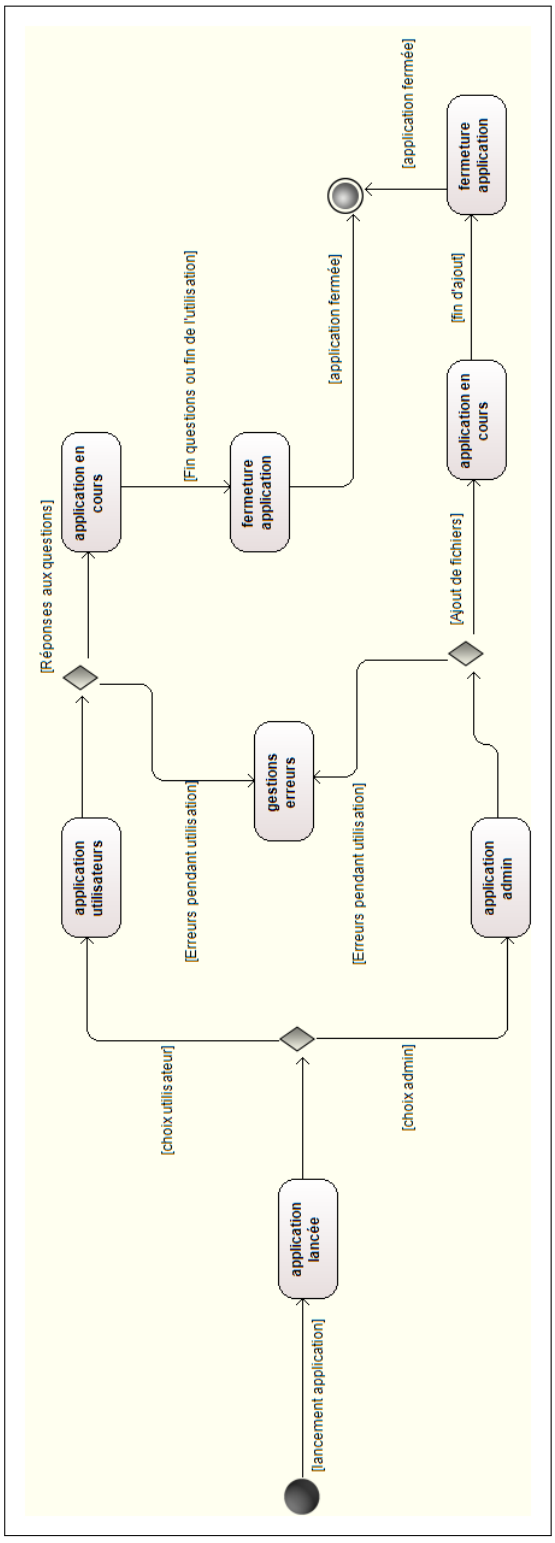


FIGURE 1.3 – Diagramme d'états - Fonctionnement de l'application

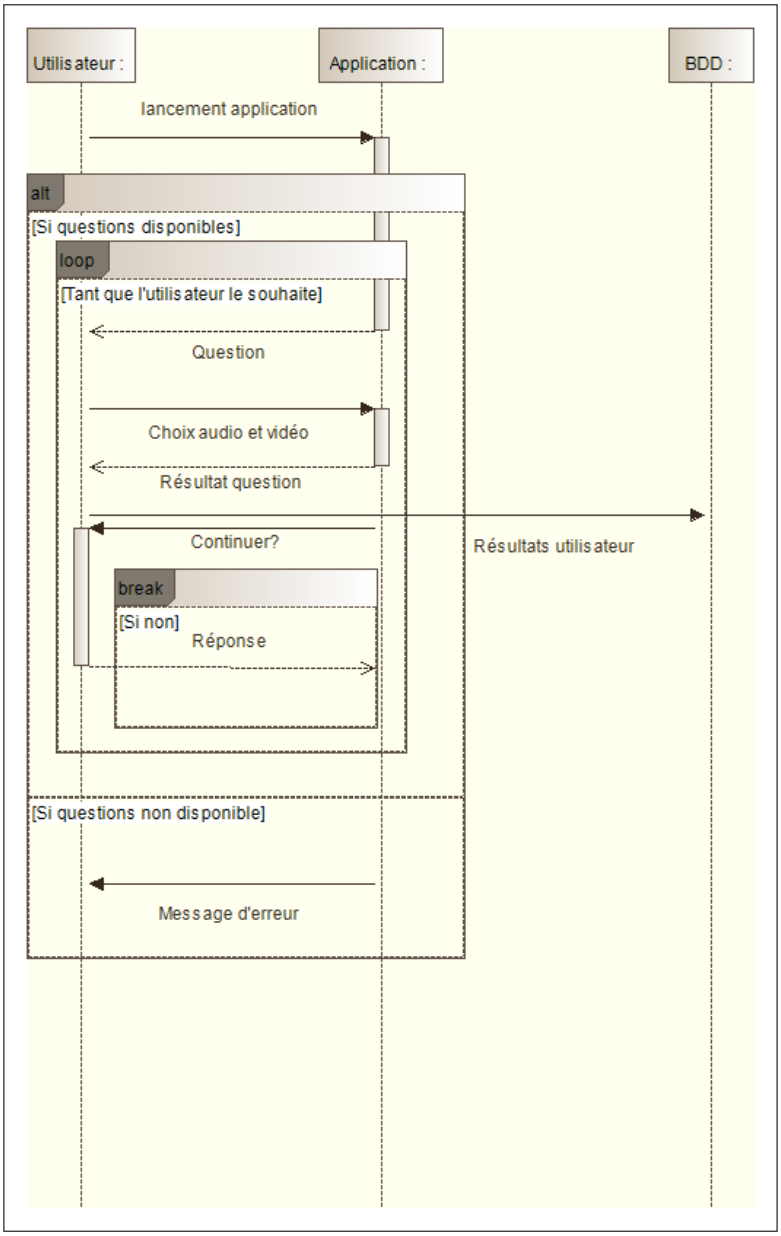


FIGURE 1.4 – Diagramme de séquences - Fonctionnement de l'application utilisateur mode Questions/Réponses

Chapitre 2

Analyse des besoins

2.1 Classement par priorité des besoins

Les besoins sont classés par priorité dans leur ordre d'apparition. De plus, chaque besoin se voit attribué un niveau de priorité comme suit :

- Priorité critique
- Priorité moyenne
- Priorité basse
- Facultatif

2.2 Besoins fonctionnels

2.2.1 Utiliser l'application sur les systèmes d'exploitation principaux et récents

Description

Etant donné que notre client sera amené à transporter l'application, il faut que celle-ci puisse fonctionner sur différents environnements à savoir *Microsoft Windows 7*, *Ubuntu 12.04*, *Debian 6.0*, *MAC OS X 10.9* et les versions plus récentes de ces systèmes d'exploitation. L'application est susceptible de fonctionner sur d'autres systèmes d'exploitation (par compatibilité de noyau) sans toutefois de garantie de ce fonctionnement.

Faisabilité

Cette condition sera remplie en utilisant une application web, supportée par tous les environnements possédant un navigateur web. Or cette qualité est remplie nativement par tous les systèmes d'exploitation cités précédemment.

Contingence

Le risque principal est que le client travaille sur une machine ne possédant pas de navigateur web. Pour pallier à cela, un navigateur web portable, comme par exemple *Mozilla Firefox Portable Edition*, sera installé sur le périphérique utilisé.

Test

Lancer l'application dans chacun des systèmes d'exploitation cités précédemment.

Priorité : *Critique*

2.2.2 Rendre l'application nomade

Description

Pour effectuer ses études, notre client ne souhaite pas transporter sa machine sur les lieux où elles se déroulent. Pour cela, il faut donc avoir une application transportable sur un périphérique externe de type clef USB.

Faisabilité

Pour remplir cette condition, tous les composants de l'application seront stockés sur une clef USB ou un disque dur externe afin que le client puisse l'utiliser sur n'importe quel ordinateur.

Contingence

Le risque serait que le formatage de la partition de la clef USB ne soit pas pris en compte par le système d'exploitation de la machine (par exemple le formatage *NTFS* de *Windows* n'est pas reconnu par le système *Mac OS*, le formatage *EXT4* du système *Ubuntu* n'est pas reconnu par les systèmes d'exploitation *Windows*). Pour éviter cela, la clef devra être formatée en *FAT32* qui est un formatage de partition reconnu par les systèmes d'exploitation cités dans la section 2.2.1.

Test

Faire fonctionner l'application à partir d'une clef USB (ou disque dur externe selon le support qui sera choisi).

Priorité : *Critique*

2.2.3 Ajouter du contenu multimédia et des questions

Description

Le client doit pouvoir enregistrer dans la base de données, de nouvelles vidéos et de nouveaux sons afin d'augmenter l'efficience de ses tests. De plus, pour améliorer ses recherches, notre client pourra ajouter des questions avec leurs correspondances audios et vidéos.

Faisabilité

Pour faciliter cette gestion, nous utiliserons une base de données. Les données seront plus facilement accessibles lors de l'utilisation de l'application.

Test

Ajouter du contenu multimédia et une question puis lister tout le contenu de la base de données pour voir si l'ajout à été pris en compte.

Priorité : ***Moyenne***

2.2.4 Exploiter différents formats de fichiers audio et vidéo

Description

Le contenu multimédia de notre client étant composé de différents formats vidéos et audios, l'application doit pouvoir assurer une lecture optimale.

Effectivement, c'est un obstacle que l'on rencontre dès que l'on commence à programmer dans le domaine de la vidéo et de l'audio (des références étudiant certaines de ces contraintes : [Gha99] et [HOD+13]). Le fond du problème est l'utilisation de Codecs (vidéo et audio) pour lire les différents types de fichiers, qui sont encodés selon différentes normes.

Les formats que l'on doit pouvoir supporter sont les suivants :

Formats audio	**Formats vidéo**
mpeg2	*mp4 (H.264)*
aac	*mov*
wav	

Faisabilité

Une solution que nous avons trouvé est le lecteur de médias *VLC Media Player* [Sol06].

Ce lecteur va nous permettre d'utiliser les formats vidéo et audio présentés précédemment. De plus, c'est un logiciel accessible (gratuit et sous licence open-source).

De surcroit, il existe une version portable de *VLC Media Player* permettant un transport optimal sur une clef USB ou un disque dur externe. Cette condition répond à notre besoin de transportabilité évoqué précédemment.

Contingence

Le risque d'utiliser ce genre de logiciel peut être l'impossibilité d'intégrer le lecteur dans une page web (solution choisie dans la section 2.2.1), si l'application doit s'ouvrir dans un navigateur. Si cette solution n'est pas possible, on pourra ouvrir un lecteur indépendamment de la page web (en *standalone*).

Test

Création d'un script qui ouvrira toutes les vidéos disponibles et vérifiera si une erreur est survenue.

Priorité : ***Moyenne***

2.2.5 Récupérer les questions et résultats des tests

Description

L'application a pour but de répondre à une question donnée en fusionnant une vidéo et un audio.
Ces combinaisons sont importantes pour notre client, nous devons donc les récupérer.

Faisabilité

La solution à ce besoin serait une base de données, stockant l'ensemble des questions, sons, vidéos et réponses/résultats. L'application devra aussi exporter les données sous forme d'un fichier *txt*, fichier qu'exploitera le client.

Test

Faire faire le test à un faux sujet, et vérifier les données exportées dans le fichier *txt*.

Priorité : ***Moyenne***

2.2.6 Rassembler des informations sur les sujets

Description

Afin de pouvoir exploiter les résultats des tests, le client veut récupérer des informations sur les sujets. Les informations que l'on doit récupérer sont les suivantes :

- nom
- prénom
- sexe
- date de naissance
- langue maternelle
- si la langue maternelle est différente de celle du test, nombre d'année d'études de cette langue

Faisabilité

Pour répondre à ce besoin, il faut implémenter un formulaire au lancement de l'application.

Test

Faire essayer l'application à une personne tierce, puis, vérifier que toutes les informations sur la personne ont bien été récupérées.

Priorité : *Faible*

2.3 Besoins non-fonctionnels

2.3.1 Permettre une maintenance du logiciel

Description

L'application que nous allons livrer ne sera pas une finalité mais seulement une étape dans un projet beaucoup plus vaste. Ainsi, d'autres personnes devront probablement modifier cette application afin de répondre à de nouveaux besoins. Ces nouveaux développeurs devront disposer de tous les éléments nécessaires pour comprendre notre travail.

Faisabilité

Il est possible de répondre à ce besoin en documentant notre code. Il existe par exemple en *Java*, la *Javadoc*, qui est générable avec des commentaires spéciaux. Elle est exportable en *PDF* et contient toutes les explications necessaires à la compréhension du code. Il existe aussi la *PHPDoc* qui est l'équivalent pour le *PHP*.

De plus, il faudra ajouter des commentaires quand une fonction sera trop complexe et que des indications intermédiaires seront nécessaires.

Test

Demander à un développeur externe au projet d'essayer de comprendre notre code.

Priorité : *moyenne*

2.3.2 Bénéficier d'une ergonomie efficiente

Description

L'application sera ergonomique pour permettre au sujet de se concentrer un maximum sur le test et pas sur le fonctionnement de ladite application. De plus, le sujet soumis au test sera possiblement novice en informatique, l'application devra ainsi être intuitive.

Faisabilité

Pour satisfaire ce besoin, nous devons mettre en place un graphisme épuré de tout accessoires détournant l'attention. De plus, nous utiliserons des couleurs de ton pastel pour éviter de fatiguer le regard de l'utilisateur.

Cette contrainte est satisfaisable en s'inspirant de nombreux designs qui ont déjà été développés et publiés sur le web. On s'appuiera notamment sur l'article [LBB14] qui étudie la *"collaboration entre Ergonomie, Design et Ingénierie"*.

Test

Se servir d'un novice en informatique pour tester l'application.

Priorité : ***basse***

2.4 Prototype d'application

En annexe, un ***prototype*** de l'application que nous allons développer est présenté. Le rendu final pourra différer de ce prototype.

Chapitre 3

Organisation

3.1 Gestion du temps

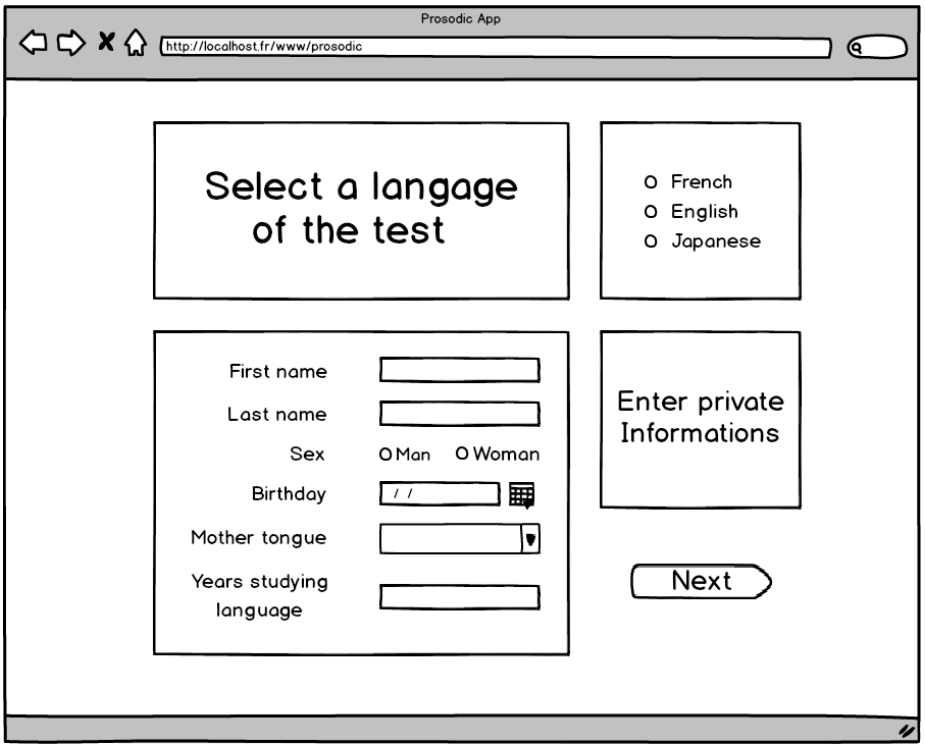
Nous avons établi un calendrier de projet qui présente le déroulement général du projet.

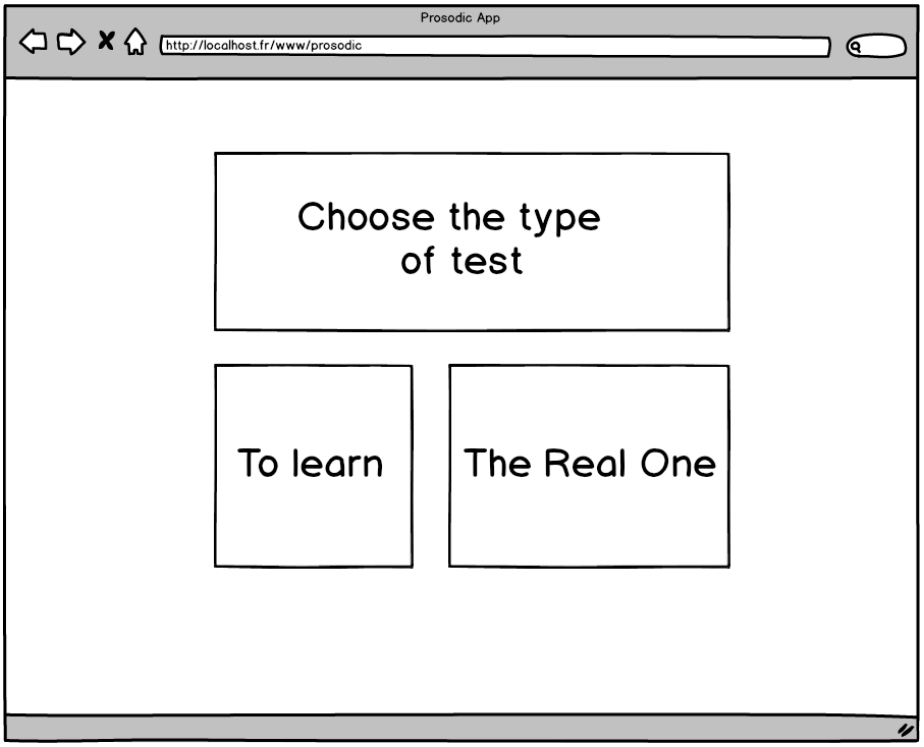
FIGURE 3.1 – Gantt - Calendrier du projet

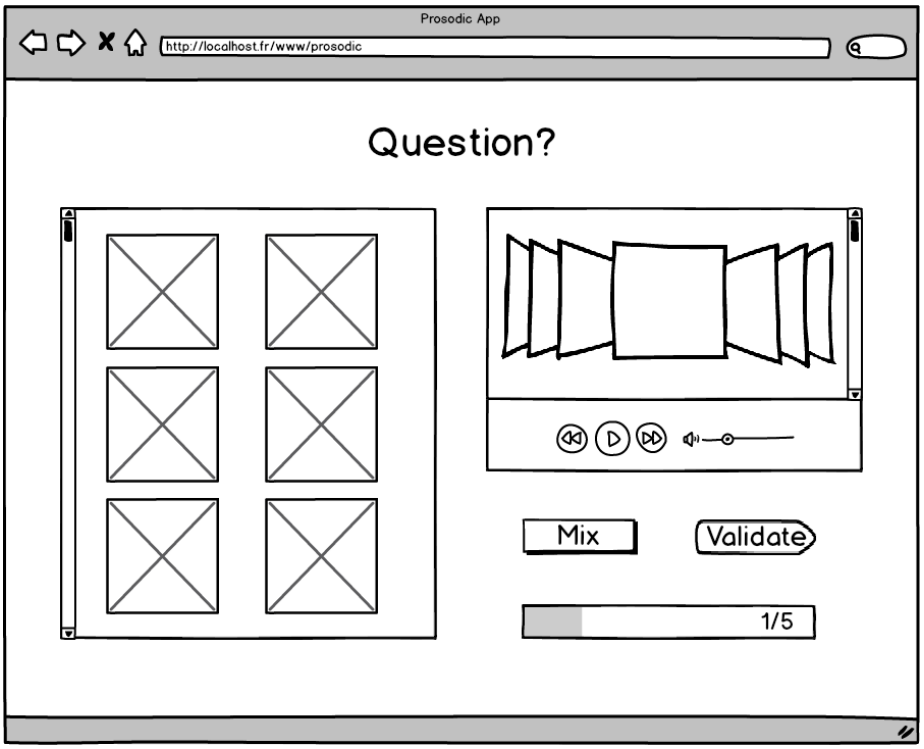
Chapitre 4

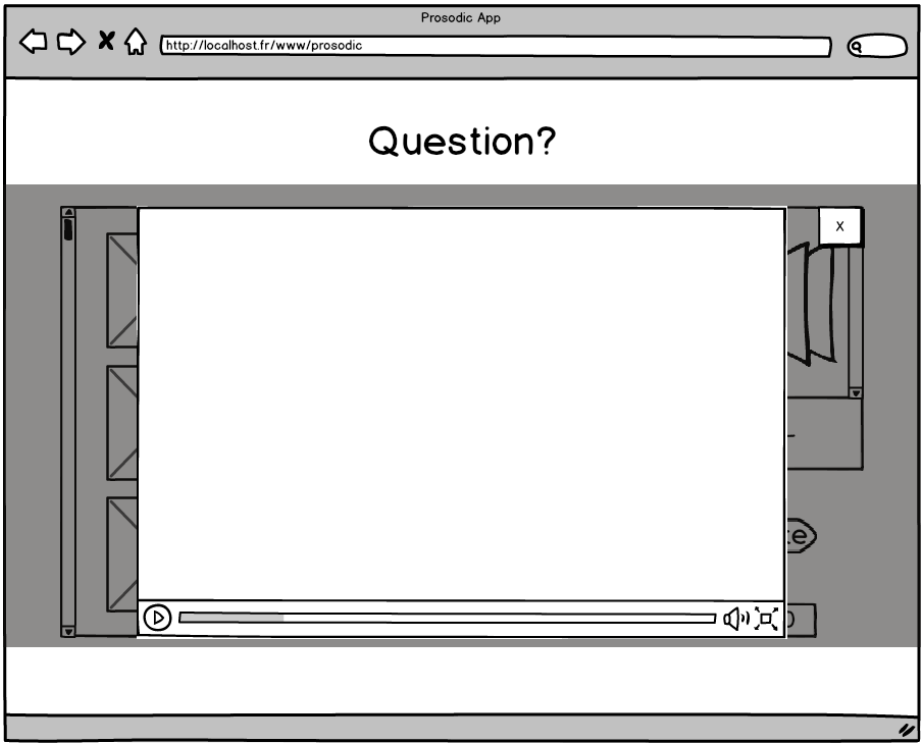
Annexe

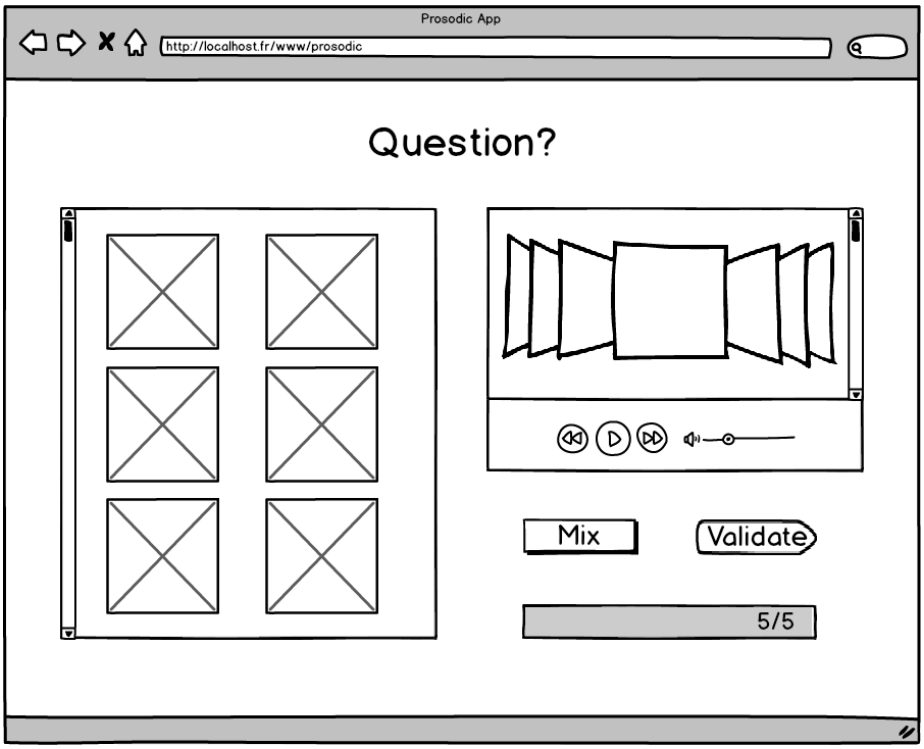
4.1 Prototype

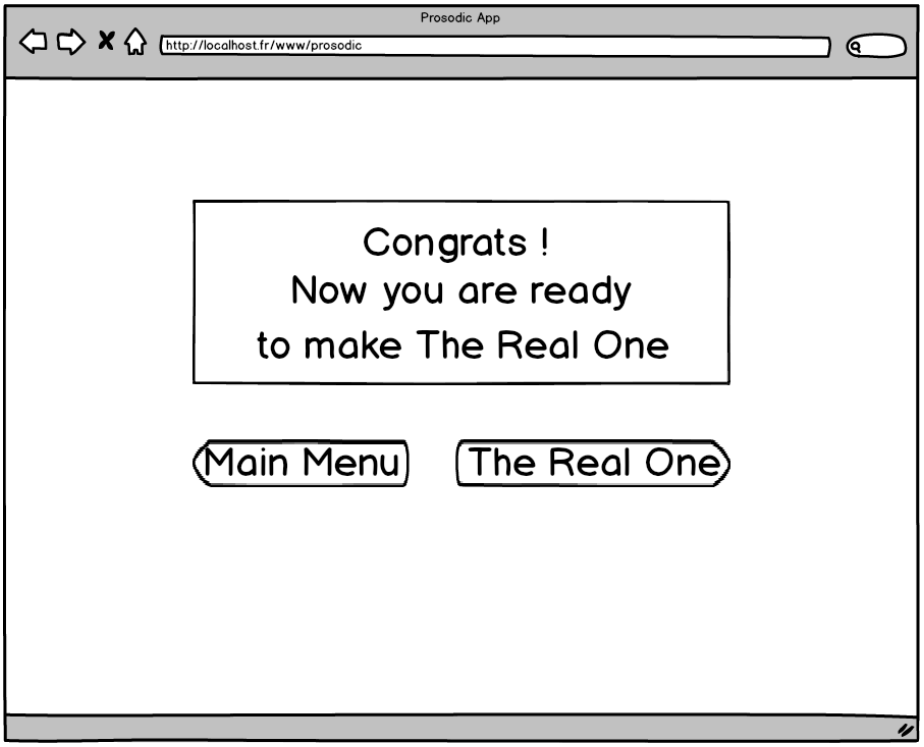


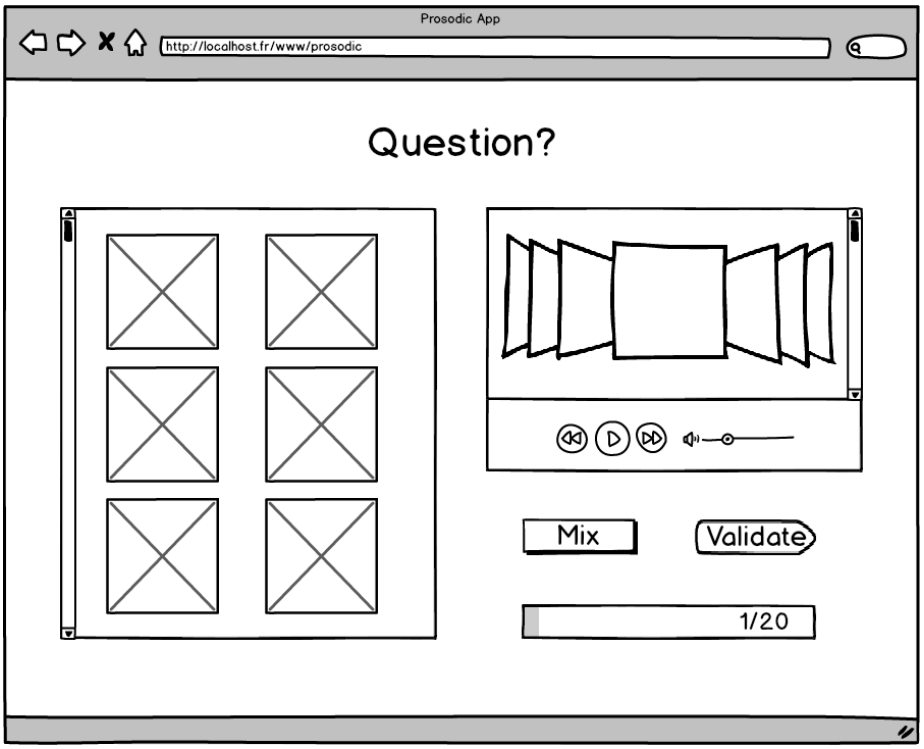

Prosodic App
http://localhost.fr/www/prosodic
Choose the type of test
To learn
The Real One

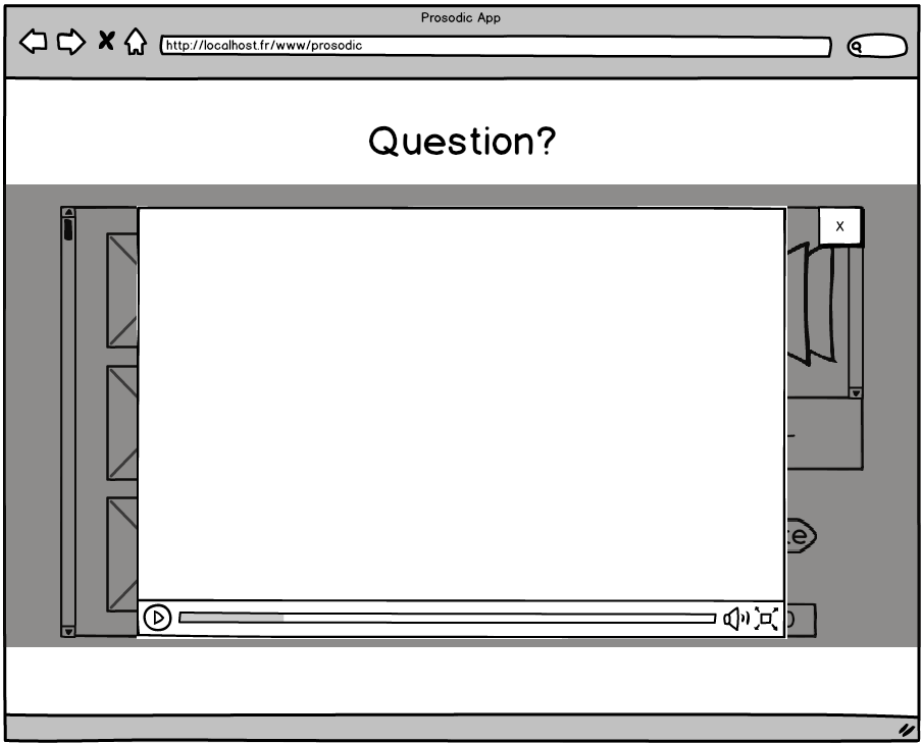

Prosodic App
http://localhost.fr/www/prosodic
Question?
Mix
Validate
1/5

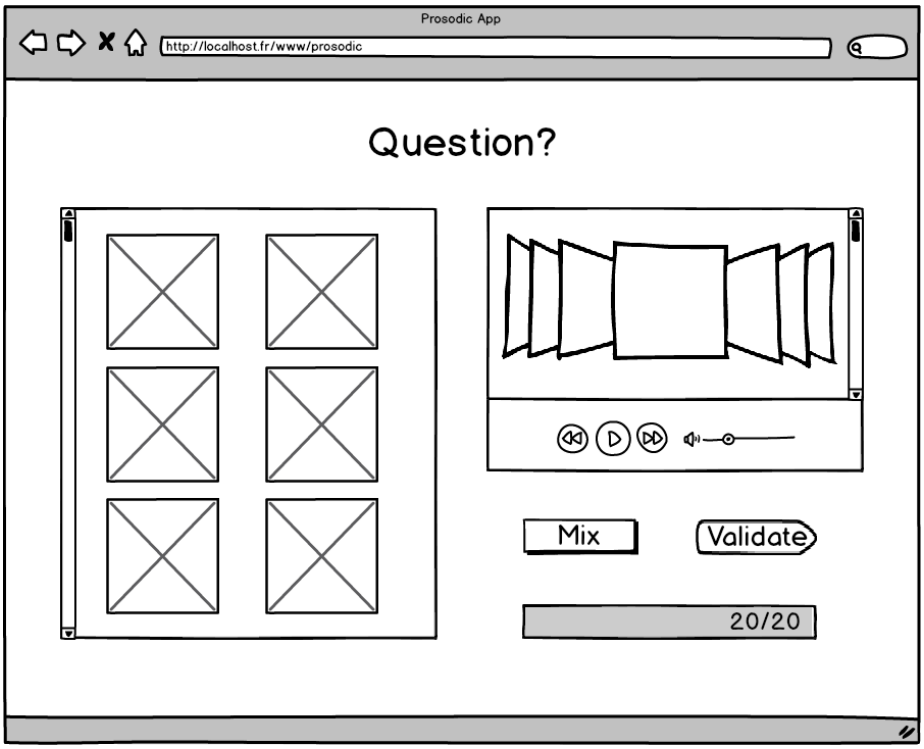

Prosodic App
http://localhost.fr/www/prosodic
Question?
X

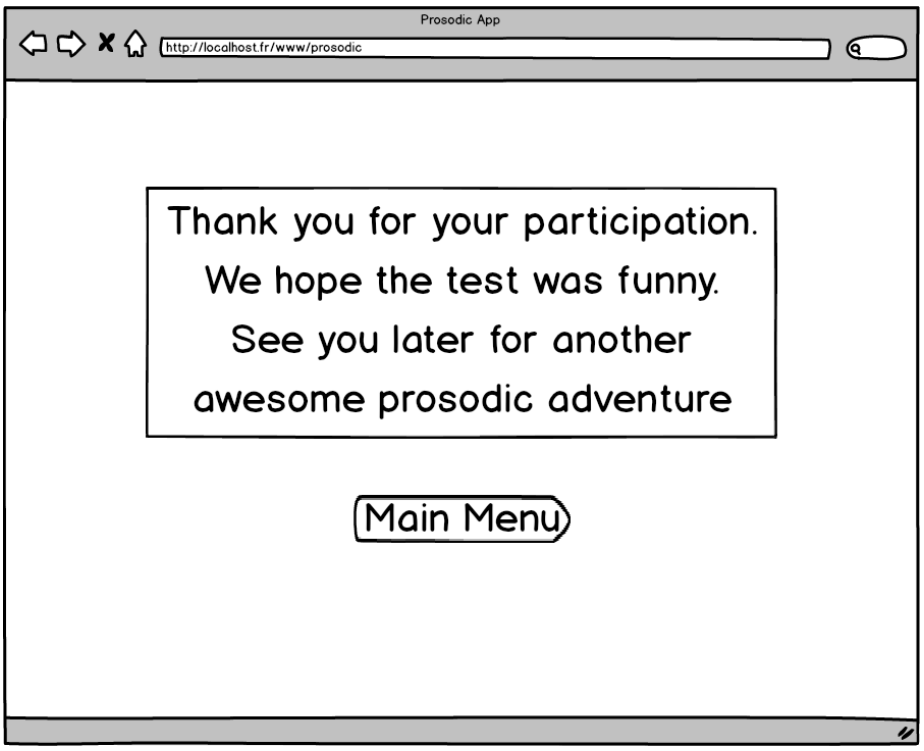

Prosodic App
http://localhost.fr/www/prosodic
Question?
Mix
Validate
5/5


Prosodic App
http://localhost.fr/www/prosodic
Congrats !
Now you are ready
to make The Real One
Main Menu
The Real One


Prosodic App
http://localhost.fr/www/prosodic
Question?
Mix
Validate
1/20


Prosodic App
http://localhost.fr/www/prosodic
Question?
X


Prosodic App
http://localhost.fr/www/prosodic
Question?
Mix
Validate
20/20


Prosodic App
http://localhost.fr/www/prosodic
Thank you for your participation.
We hope the test was funny.
See you later for another
awesome prosodic adventure
Main Menu

Bibliographie

[Ace98] A. Acero. Source-Filter Models for Time-Scale Pitch-Scale Modification of Speech. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*, Seattle, WA (USA), May 1998.

[Aub02a] Véronique Aubergé. A gestalt morphology of prosody directed by functions : the example of a step by step model developed at icp. In *Speech Prosody 2002, International Conference*, 2002.

[Aub02b] Véronique Aubergé. Prosodie et émotion. *Actes des deuxiemes assises nationales du GdR I*, 3 :263–273, 2002.

[Bac99] Jo-Anne Bachorowski. Vocal expression and perception of emotion. *Current directions in psychological science*, 8(2) :53–57, 1999.

[Bes95] Jonas Beskow. Rule-based visual speech synthesis. In *EUROSPEECH*, 1995.

[EF03] Paul Ekman and Wallace V Friesen. *Unmasking the face : A guide to recognizing emotions from facial clues*. Ishk, 2003.

[FGS+14] Dominique Fourer, Marine Guerry, Takaaki Shochi, Jean-Luc Rouas, Jean-Julien Aucouturier, and Albert Rilliard. Analyse prosodique des affects sociaux dans l'interaction face à face en japonais. In *XXXèmes Journées d'études sur la parole*, Le Mans, France, June 2014.

[Gha99] Mohammed Ghanbari. Video coding : an introduction to standard codecs. Institution of Electrical Engineers, 1999.

[GO] Ricardo Gutierrez-Osuna. Prosodic modification of speech. Introduction to Speech Processing.

[Haa94] Jesper Haagen. Transformation and decomposition of the speech signal for coding. *IEEE signal processing letters*, 1(9) :136, 1994.

[HOD+13] Yun He, Jörn Ostermann, Marek Domanski, Oscar C Au, and Nam Ling. Introduction to the issue on video coding : Hevc and beyond. *Selected Topics in Signal Processing, IEEE Journal of*, 7(6) :931–933, 2013.

[LBB14] Chloé Lenté, Soizick Berthelot, and Stéphanie Buisine. Scénariser l'usage pour améliorer la collaboration entre ergonomie, design et ingénierie. 2014.

[Mar05] Philippe Martin. Winpitch ltl, un logiciel multimédia d'enseignement de la prosodie. *Alsic [en ligne]*, 8(2), 2005.

[Mat92] David Matsumoto. American-japanese cultural differences in the recognition of universal facial expressions. *Journal of cross-cultural psychology*, 23(1) :72–84, 1992.

[RBDP+06] Jean-Luc Rouas, Melissa Barkat-Defradas, François Pellegrino, Rym Hamdi, et al. Identification automatique des parlers arabes par la prosodie. *Journées d'Etude sur la Parole*, 2006.

[Sol06] VideoLan Streaming Solutions. Vlc media player, 2006.

[VMT92] Hélène Valbret, Eric Moulines, and Jean-Pierre Tubach. Voice transformation using psola technique. *Speech Communication*, 11(2) :175–187, 1992.