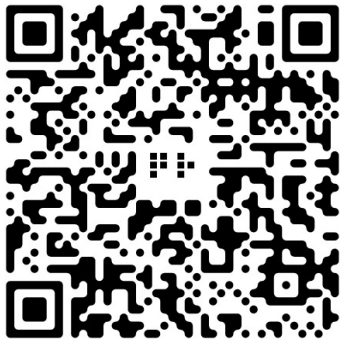
Master 2 Informatique
Management de projet
Février 2018



Développement d'un couple d'applications bureau et mobile

Génération et lecture de QR Codes pour l'institut Montéclair

Auteurs

Thomas Catalayud
Corentin Talarmain

Développeurs

David Dembele
Alassane Diop
Jules Leguy
Rahmatou Walet Mohamedoun

Encadrant

Vincent Barichard

université angers

Remerciements

Nous voulons tout d'abord remercier M. David Genest, M. Jean-Michel Richer, M. Laurent Garcia pour la mise en place du module Management de projet mais aussi et surtout M. Vincent Barichard pour son encadrement et son suivi de projet. Ensuite nous voulons remercier Mme. Rachel Coudray pour nous avoir enseigner les bases du management pour nous permettre d'appréhender le module le plus sereinement possible.

Nous souhaitons aussi particulièrement remercier l'insitut Montéclair pour nous avoir permis de travailler sur un sujet concret et intéressant, pour leur gentillesse et leur accueil. Nous pensons tout particulièrement à Mme. Cécile Barichard et Mme. Stéphanie Colin qui ont été attentives tout au long du projet à nos questions et interrogations et sont restées disponibles pour nous aider au mieux à exécuter ce projet.

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Contexte

Dans le cadre du cours de management de projet de notre deuxième année de Master 2 mention Informatique, option Analyse, Conception et Développement Informatique à l'Université d'Angers, nous avons été amené à diriger, en tant que chef de projet, quatre étudiants de Master 1 mention Informatique, sur un projet de développement d'un couple d'application de bureau et d'application mobile.

Ce couple d'applications répond à une demande de l'Institut Montéclair à Angers, dans le but d'apporter une dimension ludique à l'enseignement pour les mal-voyants et non-voyants inscrits à l'institut.

L'Institut Montéclair a été fondé en 1885. L'établissement, d'abord situé au sein de l'hôpital d'Angers, s'est installé dans le quartier du Lac de Maine à Angers depuis 1985. Depuis lors il est géré par la Mutualité Française Anjou-Mayenne. L'Institut est actuellement dirigé par Mme. Christelle MARÉCHAL.

L'institut met à disposition différents services pour tous âges tels que :

— Le Service d'Accompagnement Familial et d'Education Précoce (SAFEP) pour les enfants de 0 à 6 ans.
— Les Sections d'Enseignement et d'Education Spécialisés (SEES) pour les enfants de 6 à 20 ans.
— Service d'accompagnement à l'emploi (service interrégional d'appui en déficience visuelle (SIADV) pour les adultes.

FIGURE 1.1 – Logo Institut Montéclair

Ce projet a été développé par David DEMBELE, Jules LEGUY, Alassane DIOP et Rahmatou Walet MOHAMEDOUN quatre étudiants en première année de Master, dans le cadre de leur module Concrétisation Disciplinaire.

L'objectif du projet est de fournir la possibilité aux transcripteurs et enseignants de l'institut un moyen de générer des QR Codes contenant du texte et des sons et pouvant être interprétés par une application mobile. Une première version de l'application mobile avait déjà été développée par Corentin TALARMAIN lors de son TER [1] de fin de première année de Master. Celle-ci fournissait déjà la possibilité de lire avec une voix de synthèse du texte

1. Travail Encadré de Recherche

contenu dans des QR Codes.

1.2 Objectifs

Les transcripteurs souhaitaient néanmoins étendre les fonctionnalités de l'application mobile initiale, qui présentait quelques inconvénients limitant son utilisation. Ne présentant que la possibilité de lire des QR Codes, elle imposait le passage par des sites internet pour en générer. On ne pouvait ainsi pas contrôler la taille maximale des QR Codes générés, parmi lesquels certains étaient trop gros pour être détectables par l'application. En outre, elle ne fournissait pas la possibilité de lire des sons à partir de données contenues dans un QR Code.

Les transcripteurs de l'institut nous ont également fourni deux demandes précises autour desquelles nous avons structuré notre projet. La première demande était de pouvoir utiliser les QR Codes comme légendes de cartes de géographie pour des lycéens. Cela implique de pouvoir gérer la lecture de QR Codes adjacents dans un ordre prédéfini, peu importe l'ordre de lecture effectif.

L'autre demande était de pouvoir utiliser des QR Codes pour lire des sons dans les pages d'un livre pour enfants, sans avoir nécessairement accès à internet au moment de la lecture. Pour cela, un ou plusieurs QR Codes sur la page de couverture doivent permettre le téléchargement de tous les sons contenus dans le livre.

En outre, un espace central contenant du texte en braille devait pouvoir être inséré au centre des QR Codes, pour permettre aux non-voyants de les localiser, ces caractères étant imprimés avec une encre spéciale gonflant à température élevée.

Grâce à l'application de bureau, cette nouvelle génération de codes QR permet ainsi de faire évoluer le simple QR code généré au préalable sur un site internet vers un format plus complexe contenant plus d'informations structurée. Augmentant ainsi considérablement les possibilités d'utilisation de l'application mobile.

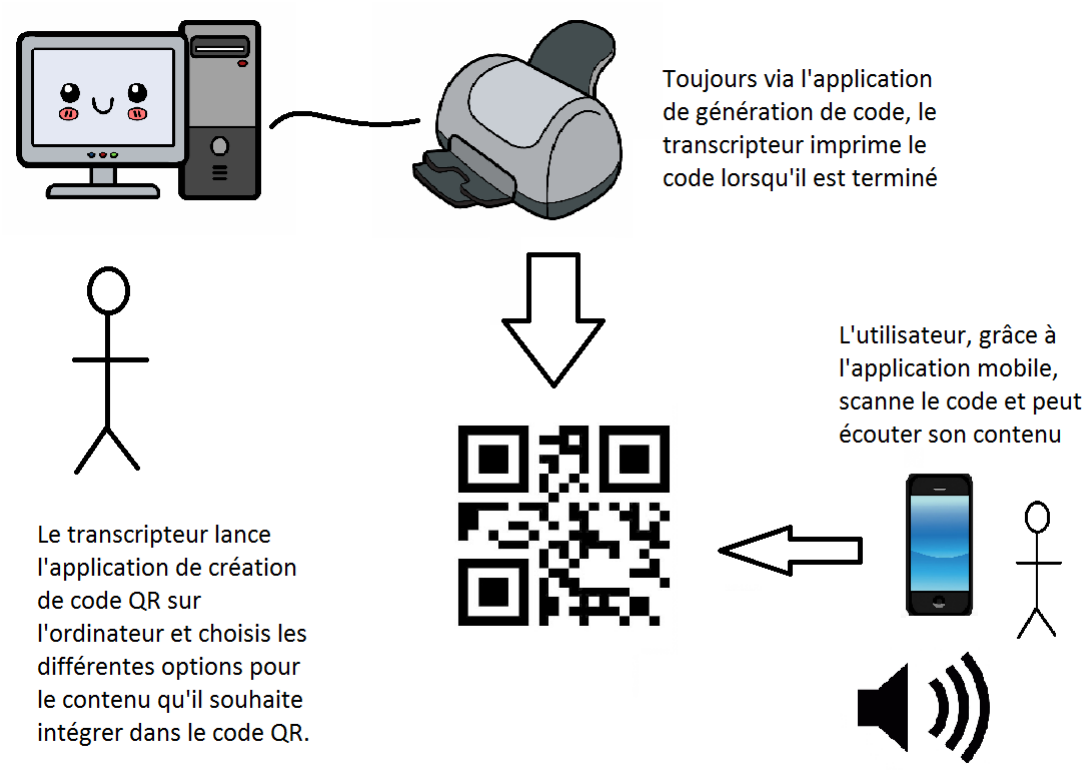


FIGURE 1.2 – Schéma d'utilisation

Chapitre 2

Management

2.1 Organisation

Dans un premier temps, nous avons dû rédiger des documents nécessaires à la bonne conduite d'un projet de développement. Ces documents sont trouvables en annexes de ce dossier (.1.2)

2.1.1 Rencontre avec les développeurs

Il a tout d'abord fallu nous présenter à notre équipe de développeurs et présenter le projet. Une réunion a donc été organisée avec eux, où nous avons expliqué le contexte du projet et l'intérêt de faire cette seconde application de bureau permettant de générer les QR codes rapidement et simplement pour les transcripteurs et enseignants de l'institut en parallèle à l'application mobile déjà existante.

Nous avons pu détailler et présenter les contraintes techniques et commencer à réfléchir aux différentes façon de les résoudre en imaginant les différentes technologies à utiliser. Il en est sorti trois principales technologies :

- C++/Qt
- Java/Swing
- HTML/PHP

Nous avons pu ensuite réfléchir à un premier plan d'action pour le début du développement de l'application, notamment concernant l'interface de l'application et les différentes fonctionnalités à implémenter.

Puis nous avons décidé d'utiliser l'outil Slack pour nous permettre de communiquer simplement entre nous et d'utiliser l'outil de gestion de version Github pour déposer et gérer le développement de l'application de bureau, sachant que Corentin Talarmain avait déjà utilisé cet outil pour le développement de la première version de l'application mobile.

Figure 2.1 – Github

2.1.2 Rencontre avec l'institut

Ce n'est qu'un peu plus tard que nous avons rencontré Cécile Barichard et Stéphanie Colin, respectivement transcriptrice et enseignante au sein de l'institut, en compagnie de Vincent Barichard lors d'une réunion ayant pour but de définir correctement leurs besoins au regard de ce projet.

C'est à la suite de cette réunion, Madame Barichard, nous a détaillé deux cas d'utilisation que l'application doit rendre possible :

- La possibilité de lire une légende de carte géographique. En effet, elle aimerait que les codes puissent contenir du texte structuré en plusieurs parties qui décriraient la légende d'une carte géographique.
- La possibilité de lire un livre pour enfant. Elle aimerait aussi qu'il soit possible de rattacher à un code le récit d'un livre pour enfant par synthèse vocale ou par message audio pré-enregistré. Il pourrait aussi être possible de rajouter en parallèle au texte braille du livre un code qui lancerait un son audio contenant un son "ambiance" en rapport avec le texte. Nous avions reçu comme exemple, un livre parlant d'un singe et l'enfant pourrait scanner un code qui lancerai le son d'un singe ou les bruits d'animaux dans la jungle.

Plusieurs questions techniques et problématiques ont été soulevées.

Il semble compliqué de stocker un son, musique, enregistrement vocal dans le code QR. Il a donc été proposé de stocker dans le code QR un lien internet téléchargeant le son en question et lançant le son sur l'application. Mais les jeunes n'ont pas toujours forcément accès à internet. Plusieurs solutions ont donc été proposées pour résoudre ce problème : - L'enseignant demande au jeune de télécharger en préalable le son grâce à un code puis lorsqu'il sera nécessaire d'utiliser ce son, il n'aura plus qu'à scanner un autre code qui lancera le son préalablement stocké sur le téléphone. Cette méthode a introduit l'idée d'une fonctionnalité permettant de générer un code "banque son" qui contient plusieurs liens pour tous les codes qui seront nécessaires pour le livre par exemple. L'application peut être livrée avec une banque son pré-enregistré défini au préalable avec l'institut.

Les sons devront donc être d'une manière ou d'une autre disponibles sur une plate-forme de stockage sur internet et téléchargés sur le téléphone grâce à un code. Il pourrait donc être possible pour l'application transcripteur, le générateur de code, de pouvoir récupérer le lien vers une plate-forme de stockage, exemple : Google drive de l'institut.

Nous avons rediscuter de la problématique concernant la limite de texte à inclure dans le code, plus le texte est long plus le code QR est complexe et difficile à scanner pour l'application mobile et donc difficile à être lu. Ce problème peut être contourné en implémentant le système de lien vers le contenu du son, ce qui montre l'importance et la priorité d'implémenter cette fonctionnalité.

Une autre problématique a été soulevée. Par exemple, pour l'application de la carte géographique et de sa légende, les légendes peuvent être trop longues et trop complexes pour stocker tout leur contenu dans un seul QR code même si nous pouvons générer un code contenant un texte structuré. Il a donc été proposé de stocker sur différents codes le contenu de la légende organisé en plusieurs parties. La première partie peut par exemple, concerner les zones agricoles, et la deuxième partie des flux de transports. Cependant, l'ordre peut être parfois important et le jeune utilisant l'application peut ne pas faire attention à l'ordre des codes et parfois inverser cet ordre en scannant les codes dans l'ordre inverse. Ainsi il lira les deux parties dans le désordre. Il a donc été proposé que nous puissions implémenter une fonction permettant de gérer entre deux codes différents un ordre de lecture, peu importe l'ordre du scan des codes.

Pour conclure, de cette réunion nous avons établi une liste de priorités concernant les fonctionnalités nécessaires à l'application. Donc dans la première phase de développement du logiciel, nous implémenterons ces fonctionnalités :

- Implémentation du son dans le code QR (via un lien internet, un chemin vers un fichier dans le téléphone, un code "banque son", etc.)
- Implémentation du livre pour enfant (lecture du texte par synthèse vocal ou par fichier audio, inclure un son "ambiance", etc.)
- Implémentation d'une légende pour une carte géographique (texte structuré dans le code QR, gestion d'une structure hiérarchique entre différents code QR, etc.)

D'autres fonctionnalités ont été proposées, moins importantes, mais intéressantes à développer :

- Pour la prévisualisation du code, il peut être intéressant que le transcripteur puisse entendre le contenu du code directement dans l'application transcripteur, le générateur de code QR avant de l'imprimer ou de l'utiliser.
- La transcriptrice utilise un logiciel de dessin pour créer les cartes et imprimer les codes donc il peut être important pour l'application, plutôt que d'imprimer directement le code de pouvoir copier le code QR pour le coller et l'utiliser sur le logiciel directement.

C'est donc à la suite de ses réunions que nous avons pu vraiment engager le projet ou nous avons après concertation avec Vincent Barichard, défini l'outil que nous allions utiliser pour le développement de l'application de bureau. C'est le framework Electron, un outil permettant la création d'application de bureau en utilisant les technologies du web (HTML, CSS, Javascript). Nous avons choisi cette solution car une application web permettait simplement de répondre aux différentes problématiques du projet et il nous paraissait indispensable d'avoir un client lourd pour des besoins d'utilisation et de déploiement de la solution.

Nous laissons à disposition des tableaux décrivant notre organisation pour cette mise en œuvre du projet en annexe de ce dossier (.2.2).

FIGURE 2.2 – Electron

2.2 Post-Mortem

Pour faire un bilan de notre expérience de management de cette équipe, on peut la séparer en deux points :

2.2.1 Générale

Le management dans sa définition générale a été assez laborieuse car elle nous a demandé des capacités de gestion d'équipe et de rédaction autour de projet qui nous manquaient au début. Cela n'a pas été facile de se mettre à la rédaction de tous les documents qui composent un projet (devis, cahier des charges,...) mais surtout ce n'était pas forcément évident de se placer en tant qu'autorité auprès d'étudiants qui sont nous 1 an auparavant. Cependant, cette expérience est intéressante pour qu'un de nous puisse se rendre compte que le management d'équipe est peut-être une option admettable dans la carrière.

2.2.2 Particulière à l'équipe QRLudo

Cependant, bien que l'expérience générale est plutôt mitigée, nous avons eu la chance de tomber sur des étudiants motivés qui ont été capable de donner de leur temps pour mettre en œuvre ce projet dans le temps imparti. Cela nous a permis d'avoir au final une bonne expérience du management et de permettre d'en avoir un aperçu correct. De plus, le projet était une continuité d'un des chefs de projet, il était plus facile d'avoir une accroche avec les développeurs.

2.2.3 Points positifs

— Une expérience globalement positive
— Mise en place d'un échange positif entre les managers et l'équipe
— Un projet avec un rendu fonctionnel
— Aperçu du travail de chef de projet
— Continuité d'un projet déjà connu
— Projet concret pour un client hors université

2.2.4 Points négatifs

— Un suivi peut-être trop faible de l'équipe
— Difficulté à avoir de l'autorité dans un cadre universitaires
— Peut être difficile à coupler avec les cours

2.2.5 Points d'amélioration

— Étant donnée la difficulté de coupler les cours avec ce projet, nous pensons qu'il pourrait être intéressant de voir si, plutôt que d'étaler le projet sur toute l'année, il n'est pas possible de regrouper au maximum le travail sur une période de temps plus courte mais plus intense. Cela permettrait de se concentrer plus sur le projet et de nous y impliquer encore plus, en plus de nous mettre un peu plus en condition réelle de projet.

Chapitre 3

Conception

3.1 Types de QR Codes

À partir des contraintes définies dans les cahier des charges (voir 1.2), il a fallu définir précisément quels types de QR Codes allaient être créés. Une première version de notre architecture collait aux besoins de l'Institut Montéclair qui nous ont été transmis. Elle comprenait des QR Codes pour les cartes géographiques, des QR Codes pour les pages d'un livre et des QR Codes pour les pages de couverture des livres. Mais cette architecture s'est avérée trop spécifique et empêchait les transcripteurs de l'institut de créer des QR Codes pour d'autres supports.

Nous avons donc créé une seconde version qui respectait les contraintes transmises par l'institut tout en étant la plus généraliste possible, afin d'être utilisable dans tous les contextes compatibles avec les contraintes transmises. Cette architecture est composée de QR Codes contenant un ensemble de textes et de fichiers et pouvant être organisés en familles afin d'être lus dans un ordre prédéfini (les QR Codes atomiques), et de QR Codes contenant un ensemble de liens vers des sons devant être téléchargés sans être joués (les QR Codes ensembles).

3.2 Représentation des données

Les QR Codes devant stocker plus d'informations que dans la version initiale de l'application mobile, il nous a fallu définir une structure de données pour les représenter. Cette représentation s'est construite parallèlement à l'élaboration des types de QR Codes (voir 3.1) et à celle du Modèle (voir 4.1.2), afin d'assurer la cohérence de l'ensemble.

Dans le but de minimiser la quantité d'informations contenue dans le QR Code, nous nous sommes appuyés sur un modèle séparant les données nécessaires à la lecture par l'application mobile et les données nécessaires seulement au chargement à l'identique du QR Code après son enregistrement par l'application de bureau.

Pour comprendre, prenons l'exemple d'une page d'un livre pour enfant décrivant des animaux. Afin de rendre cette page plus ludique pour les non-voyants, les transcripteurs de l'institut souhaiteraient coller sur cette page un QR Code pour chaque animal décrit. Ces QR Codes contiendraient chacun un petit texte explicatif et le cri de l'animal. Ils souhaiteraient qu'ils soient lus dans un ordre prédéfini, et qu'un texte en braille représentant les lettres *QR* soit présent au centre de chacun des QR Codes. De plus, le QR Code devrait être bleu et le texte central en braille rouge afin qu'il puisse être imprimé en relief.

Parmi ces données, certaines ne sont pas indispensables à l'application mobile pour interpréter le QR Code. Elle n'a par exemple pas besoin de connaître le texte en braille ou le nom du fichier contenant le cri de l'animal. Ces données seront en revanche nécessaires à l'application de bureau pour recharger les QR Codes à l'identique le jour où les transcripteurs voudront modifier la description de l'animal. Nous allons donc séparer ces données en deux parties : les données nécessaires à l'application mobile seront encodées dans le QR Code tandis que les données qui ne le sont pas seront insérées dans le fichier image, mais pas dans le QR Code en lui-même. Nous détaillons cette séparation dans le schéma suivant.

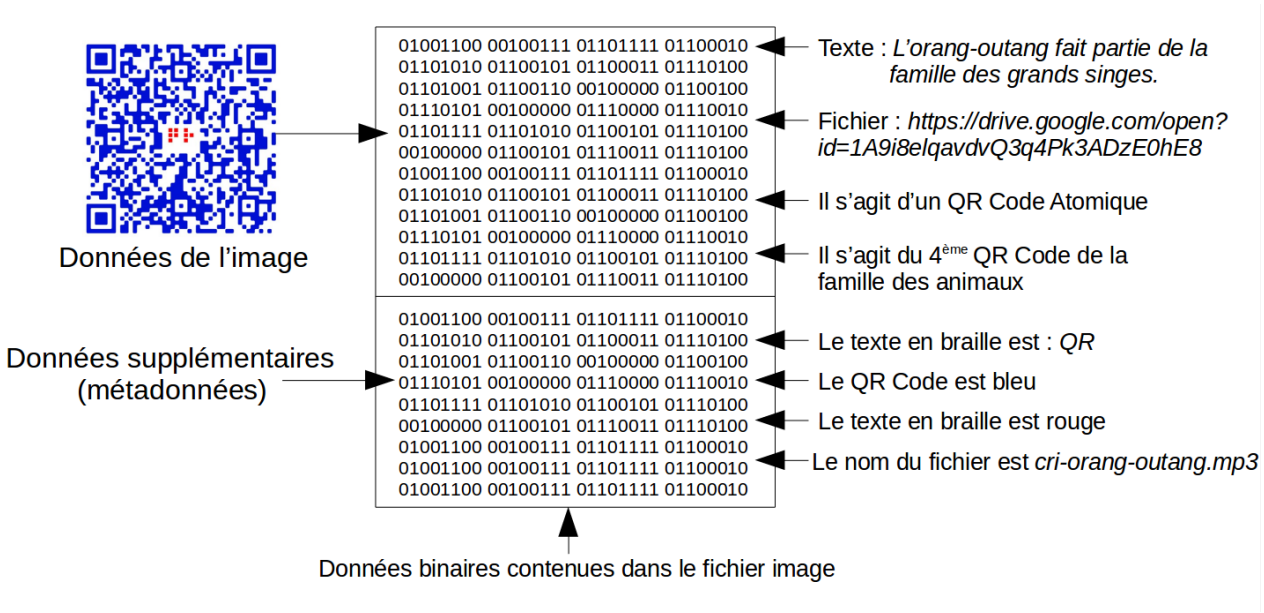


FIGURE 3.1 – Contenu d'une image d'un QR Code

Afin de stocker ces données de manière formelle, nous avons établi une structure de type XML [1] (voir 4.3.6). Elle fait apparaître clairement la dichotomie entre les données stockées dans le QR Code (noeud <donneesUtilisateur>) et les données stockées dans les métadonnées de l'image (noeud <metadonnees>).

Le type de QR Code (atomique ou ensemble) est indiqué par un attribut dans le noeud <donneesUtilisateur>. Ce noeud contient un noeud <contenu> contenant lui-même un ensemble de textes (<texte>) et de fichiers (<fichier>).

L'appartenance à une famille (voir 3.1) de QR Codes est indiquée par la présence d'un noeud <famille> ayant pour attributs le nom de la famille et la place du QR Code.

La représentation formelle de l'exemple du QR Code orang-outang est visible dans la figure ci-dessous.

```
<qrcode>
  <donneesutilisateur type="atomique">
    <contenu>
      <texte>L'organg-outang fait partie de la famille
             des grands singes.</texte>
      <fichier url="1A9i8elqavdvQ3q4Pk3ADzE0hE8"/>
    </contenu>
    <famille nom="animaux" ordre="4"/>
  </donneesutilisateur>
  <metadonnees>
    <fichiers>
      <fichier url="1A9i8elqavdvQ3q4Pk3ADzE0hE8"
               nom="cri-orang-outang.mp3"/>
    </fichiers>
    <colorqrcode color="#1606e7"/>
    <textebraille texte="QR"/>
    <colorbraille color="#ea0000"/>
  </metadonnees>
</qrcode>
```

FIGURE 3.2 – Représentation d'un QR Code

3.3 Stockage des données

L'impossibilité de stocker un fichier audio dans un QR Code nous a poussé à trouver une solution de stockage distant pour les fichiers volumineux. Les transcripteurs de l'application utilisant déjà Google Drive pour stocker des fichiers, cela nous a paru être une solution adaptée. Cette solution présente également l'avantage de ne pas nécessiter la mise en place et la location d'un serveur.

1. eXtensible Markup Language (https ://www.w3schools.com)

Chapitre 4

Application de bureau

4.1 Conception logicielle

4.1.1 Langages

Le choix des langages de programmation pour l'application de bureau a été discuté lors de la première séance. Nous avons évoqué le couple Java/Swing qui avait l'avantage d'être le plus simple à programmer et qui aurait été en cohérence avec l'application Android, mais qui nécessitait un environnement Java sur les machines des utilisateurs. Nous avons également évoqué le couple C++/Qt qui était le plus portable mais potentiellement trop lourd pour l'utilisation qu'on en aurait.

Nous avons finalement opté pour le framework [1] Electron entre les deux premières séances. Ce framework permet de développer des applications de bureau avec des langages web (Javascript/HTML/CSS), et d'utiliser les modules Node.js. Ce choix a été motivé par l'intérêt qu'avait le Javascript d'être facilement utilisable pour accéder à Google Drive (voir 3.3).

4.1.2 Architecture

L'application est structurée selon un modèle MVC (Modèle Vue Contrôleur). Les données sont représentées par les classes du modèle, sont traitées par les classes du contrôleur et affichées par la vue. Nous utilisons en outre le patron de conception [2] Façade, qui nous permet de limiter la dépendance de la vue au contrôleur à une seule classe. Nous schématisons ces interactions dans la figure suivante.

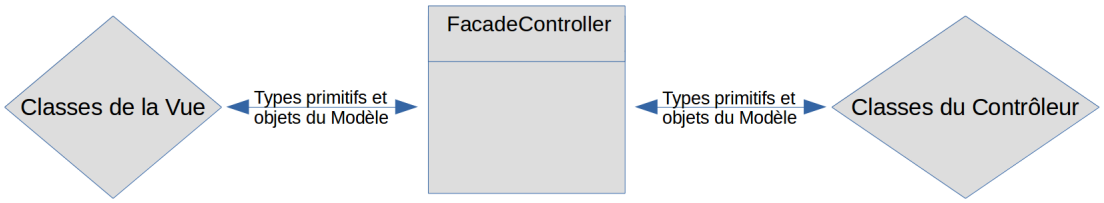


FIGURE 4.1 – Interactions entre la vue et le contrôleur à travers la classe FacadeController

Le modèle permet de représenter les QR Codes des différents types (voir 3.1). Il est composé d'une superclasse abstraite *QRCode*, étendue par deux classes *QRCodeAtomique* et *QRCodeEnsemble* représentant les spécificités des deux types de QR Codes gérés. Le modèle n'effectue pas de traitement des données mais fournit seulement des méthodes permettant d'y accéder et de les modifier, en faisant une abstraction de la façon dont elles sont représentées. Le diagramme UML des classes du modèle est visible ci-dessous.

1. Un framework désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (Wikipédia).
2. Un patron de conception (souvent appelé design pattern) est un arrangement caractéristique de modules, reconnu comme bonne pratique en réponse à un problème de conception d'un logiciel. Il décrit une solution standard, utilisable dans la conception de différents logiciels. (Wikipédia)

FIGURE 4.2 – Diagramme UML des classes du modèle

Le contrôleur se charge de tous les traitements permettant de générer des images de QR Codes (voir 4.3.3), ou d'instancier des objets du modèle à partir d'une image déjà générée (voir 4.3.5). L'architecture des classes du contrôleur est visible ci-dessous.

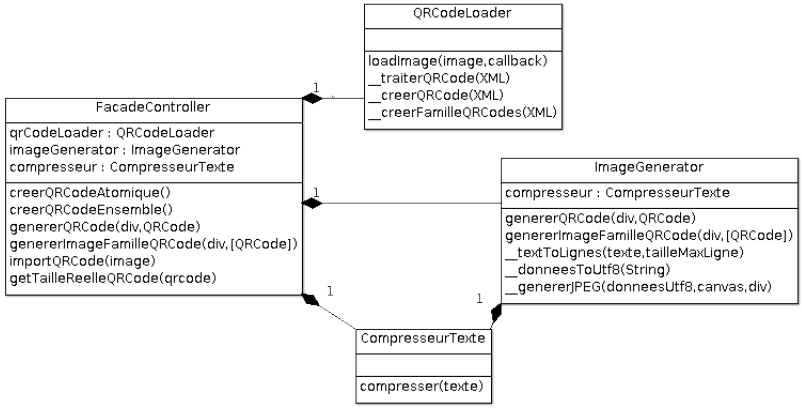


FIGURE 4.3 – Diagramme UML des classes du contrôleur

La vue est responsable des interactions avec l'utilisateur (voir 4.3.1). Elle offre une interface permettant de charger, de créer, d'afficher et de gérer des QR Codes.

4.2 Fonctionnalités

4.2.1 Création

Pour répondre au besoin principale de l'institut, il était tout d'abord fondamental de donner la possibilité de créer différents types de structures répondant aux cas d'utilisations

4.2.1.1 Unique

Le QRCode dit "Unique" est la structure basique de création dans l'application bureau. Ce QRCode va être indépendant, ne va être lié à aucun autre QRCode et pourra donc convenir à des utilisations simple des QRCodes comme par exemple un jeu de cartes.

4.2.1.2 Famille

La structure nommée "QRCode Famille" correspond à un ensemble de QRCodes. La notion de "famille" est lié au fait que les QRCodes qui feront partie de cette famille seront liés entre eux de telle manière que quelque soit l'ordre dans laquel ils sont détectés, ils seront toujours lus dans le même ordre. Cette structure pourra être utile sur des cas d'utilisations telles que des légendes de cartes géographique ou des livres.

NB : Une famille peut contenir autant de QRCodes souhaité, qui pourront être réorganisés selon le souhait de l'utilisateur. De plus, il n'est pas obligatoire de lire tous les QRCodes d'une famille pour pouvoir en écouter le contenu.

4.2.1.3 Ensemble

La dernière structure est sans doute la plus abstraite, la structure "Ensemble". Bien qu'ayant en apparence le même usage que la structure "Famille", l'ensemble de QRCodes à une toute autre utilité. Lorsqu'une "Famille" de QRCodes possèdent des QRCodes qui seront liés entre eux, l'"Ensemble" de QRCodes possèdent des QRCodes qui ne sont pas liés entre eux.

4.2.2 Modification

L'application de bureau concernant principalement des travaux de création de QRCodes, il était important de fournir des outils permettant de modifier les travaux effectués.

4.2.2.1 Unique

Après avoir créé le QRCode "Unique", il est possible de le modifier de façon à convenir à l'utilisation souhaitée. Pour ce faire, de nombreuses fonctionnalités sont fournies.

Ajout d'un texte
La première fonctionnalitée et la plus importante est l'ajout d'un texte dans le QRCode. Ce texte va faire partie d'un champ dans le code, et quand le QRCode sera détecté, sera lu de manière individuelle par l'application.

Ajout d'un son
La deuxième fonctionnalitée est l'ajout d'un son dans le QRCode. Ce son doit être présent en tant que fichier ".mp3" sur le drive de l'institut. Il pourra alors être ajouté dans le QRCode. Le son sera lu de manière individuelle par l'application.

Suppression d'un champ
La troisième fonctionnalitée permet de supprimer un champ précédemment créé, que ce soit un texte ou un son. Pour se faire, il suffit de cliquer sur la croix présente dans le champ correspondant.

Ajout d'un texte en braille
Il est possible d'ajouter un texte en braille qui sera alors situé au milieu du QRCode. Cette fonctionnalité est nécessaire pour permettre de donner un repère en braille pour situer le QRCode. Pour se faire, il suffit de cocher la case "Texte en braille" puis d'indiquer le texte souhaité dans le champ qui est apparu.

Changement de la couleur
Enfin, il est possible de changer la couleur du QRCode et / ou du texte en braille. Pour se faire, il suffit de cliquer sur le bouton avec un rectangle de couleur à l'intérieur (noir par défaut) puis de sélectionner la couleur souhaitée. Il est possible de changer la couleur du texte en braille et du QRCode indépendamment l'un de l'autre.

4.2.2.2 Famille

Après avoir créé une famille de QRCode, il est possible de la modifier pour la faire correspondre aux besoins de l'utilisateur.

Nom de la famille
Tout d'abord, il est possible de modifier le nom de la famille. Celui-ci servira à identifier la famille, que ce soit pour savoir quels QRCode sont à regrouper ensemble que pour que l'utilisateur puisse retrouver cette famille.

Ordre des QRCodes
L'ordre des QRCodes dans la famille est important puisqu'il permet de savoir l'ordre dans lequel ils doivent être lus. De ce fait, une liste est mise à disposition dans laquelle tous les QRCodes apparaissent.

Ajout d'un QRCode
Pour ajouter un QRCode à la famille, il suffit de cliquer sur la croix présente en bas la liste et d'indiquer le nom du nouveau QRCode. Cela permet de mettre autant de QRCodes que souhaité dans la famille.
NB : La gestion individuelle des QRCodes est similaire à la gestion d'un QRCode "Unique".

4.2.3 Consultation

4.2.3.1 Prévisualisation

Dans l'application Bureau, il est possible à tout moment dans la création d'un QRCode d'en voir l'apparence. Cela permet notamment de vérifier les concordances des couleurs, l'affichage du texte en braille correct ou une densité du code correcte.

Famille
Il est possible aussi de prévisualiser l'apparence de l'image qui permettra d'enregistrer la famille sur l'ordinateur. L'exportation d'une famille étant un peu particulière, sa prévisualisation permet d'avoir une emprise sur son exportation. Pour plus d'explications sur l'exportation d'une famille(voir 4.2.3.4).

4.2.3.2 Lecture

Individuelle
Lorsqu'un QRCode est en cours de modification, il est possible de lire individuellement chaque champ du QRCode. Cela permet d'être sûr que le rendu du QRCode en terme d'écoute correspond aux attentes.

Générale
Toujours lors de la modification d'un QRCode, il est possible d'écouter la totalité des champs présents dans le code pour être sûr que l'ordre et le rendu correspond aux attentes.

4.2.3.3 Exportation

Unique
Il est possible de sauvegarder un QRCode créé en l'exportant sur l'ordinateur. Le QRCode va alors prendre la forme d'une image au format JPEG qui va contenir toutes les informations du QRCode. Cette image va avoir deux fonctions :
— Pouvoir utiliser le QRCode en tant qu'image dans un document pour l'imprimer et s'en servir
— Pouvoir utiliser le fichier pour charger à nouveau le QRCode dans l'application

Famille
Il est aussi possible de sauvegarder une famille de QRCode en exportant aussi une image, mais celle-ci ne sera pas un QRCode. En effet, à l'exportation d'une famille de QRCode, une image est créée, indiquant les informations de la famille (le nom et le nombre de QRCodes) et qui, au chargement, importe la famille dans l'application pour la modifier à nouveau.

4.2.3.4 Charger

Il est possible dans l'application bureau de charger les QRCodes précédemment exportés. Cette fonctionnalité est primordiale puisqu'elle permet de reprendre des travaux effectués. Il est possible de charger :

— Un QRCode Unique
— Une famille de QRCode

4.3 Implémentation

4.3.1 Interface graphique

L'interface graphique offre la possibilité à l'utilisateur d'interagir avec l'application.

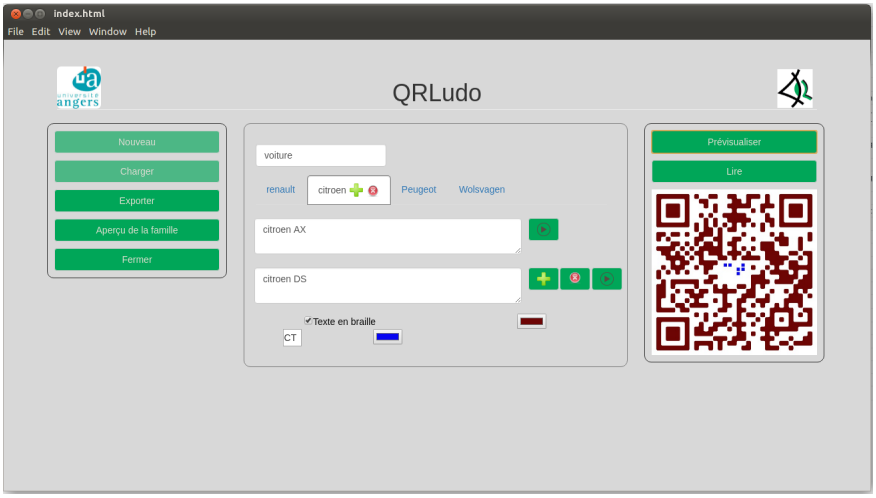


FIGURE 4.4 – Interface de création d'une famille de qrcodes

Elle est constituée de trois blocs partant de la gauche vers la droite :

1. Bloc 1 : partant de haut en bas, il regroupe les boutons permettant les opérations suivantes :
   — la création d'un nouveau projet de QR Code unique ou d'une famille de QR Codes.
   — l'importation d'un projet de QR Code unique ou d'une famille de QR Codes.
   — l'exportation d'un projet de QR Code unique ou d'une famille de QR Codes.
   — l'aperçu de l'image d'une famille de QR Codes.
   Les boutons Sauvegarder et Fermer permettent de sauvegarder le projet et de fermer l'aperçu.

   — la fermeture d'un projet.

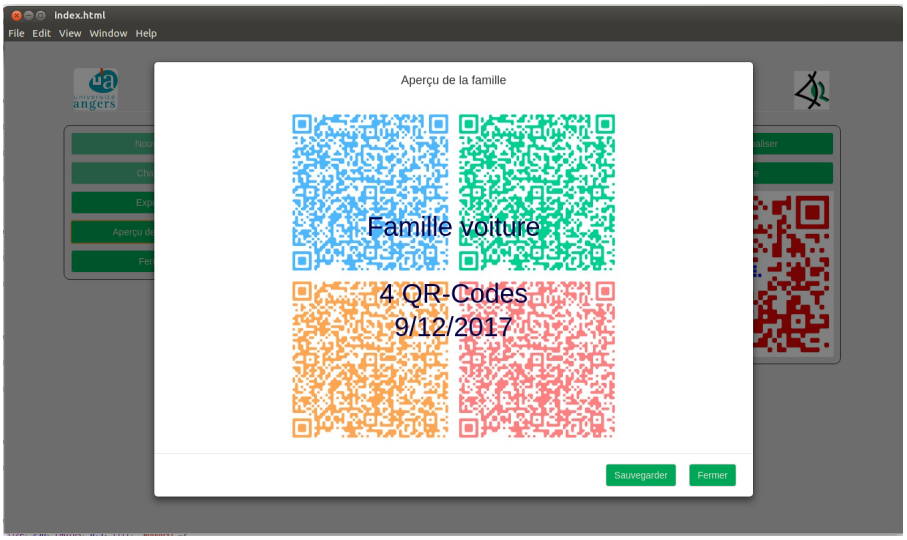


FIGURE 4.5 – Aperçu d'une image famille de QR Codes

2. Bloc 2 : il contient un formulaire qui regroupe des champs (musique ou texte). À coté de chaque champ se trouvent des boutons pour créer, supprimer et lire le contenu du champ par synthèse vocale.
   À la fin du formulaire, de la gauche vers la droite, se trouvent :
   — une case à cocher pour afficher ou masquer les options du texte en braille à savoir le texte et la couleur du texte via une palette de couleurs. Notons que le texte en braille ne peut pas dépasser deux caractères, afin de ne pas empêcher la lecture du QR Code.
   — une palette de couleurs pour la couleur du QR Code.
   Dans le cas d'un projet de famille de QR Codes, on note l'apparition :
   — d'une zone de texte pour le nom de la famille, en haut du formulaire.
   — d'une liste d'onglets, chacun faisant référence à un formulaire. À coté de chaque onglet, figurent deux boutons pour ajouter un onglet ou supprimer l'onglet actif ; chaque onglet représente un QR Code.
3. Bloc 3 : partant du haut vers le bas, il contient :
   — un bouton pour prévisualiser un QR Code à partir des informations du formulaire (celui de l'onglet actif dans le cas d'un projet de famille de QR Codes).
   — un bouton lire pour lire par synthèse vocale tous les champs du formulaire (le formulaire de l'onglet actif dans le cas d'une famille de QR codes).

Un manuel plus exhaustif destiné aux utilisateurs de l'application est disponible en annexe de ce dossier (.3).

4.3.2 Prévisualisation

La prévisualisation permet de générer le QR Code avec les informations du formulaire de l'interface.
Nous faisons alors appel au contrôleur pour générer le QR Code (voir 4.3.3). L'image générée sera celle qui sera enregistrée lors de la sauvegarde d'un projet de QR Code. Cette sauvegarde se fait de manière interactive.
Dans le cas d'un projet de famille de QR Code, l'interaction concerne la sauvegarde de l'image de la famille ; la sauvegarde des images des QR Codes de la famille se fait de manière non interactive (automatiquement).

4.3.3 Génération de QR Codes

Notre application utilise le script jquery-qrcode [3] pour générer les images des QR Codes. Il a l'avantage d'être très souple d'utilisation et de pouvoir générer des QR Codes aux formats assez complexes. Il permet notamment d'insérer une image ou un champ texte au QR Code, ainsi que de définir les couleurs du texte central et du QR Code. Nous avons tiré profit de ces caractéristiques pour laisser la possibilité aux transcripteurs de l'institut d'insérer un texte en braille central, d'une couleur pouvant être imprimée en relief (voir 1.2). Un QR Code de type atomique (voir 3.1) possédant des caractères centraux en braille est visible dans la figure 4.6.

3. https ://larsjung.de/jquery-qrcode/

La génération des images est effectuée par la classe du Contrôleur *ImageGenerator* (voir 4.1.2). En plus des QR Codes simples, elle permet de générer des images de sauvegarde des familles de QR Codes (voir 3.1 et 4.3.5). Ces images contiennent le contenu des QR Codes de la famille dans les métadonnées, et des informations sur la famille imprimées sur l'image. Un exemple d'une image famille générée par l'application est visible dans la figure 4.7

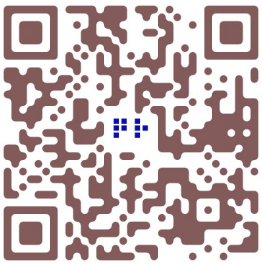


FIGURE 4.6 – QR Code simple

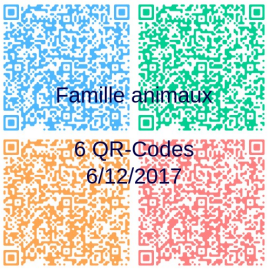


FIGURE 4.7 – Image de sauvegarde famille

4.3.4 Gestion des métadonnées

Dans l'objectif de minimiser la quantité de données stockées dans le QR Code, toutes les données qui ne sont pas nécessaires à l'application mobile mais qui ont leur intérêt dans le chargement de QR Codes déjà enregistrés par l'application de bureau sont stockées dans les métadonnées de l'image du QR Code (REF Représentation des données). De plus, les images de type sauvegarde de famille (REF chargement QRCode) contiennent l'intégralité de la représentation XML des QR Codes la formant dans les métadonnées.

Pour insérer des métadonnées dans les images générées, nous utilisons le module Node.js piexifjs dans le processus suivant. Le QR Code est d'abord généré dans un canvas HTML 5, à partir duquel on va générer une image JPEG contenant dans les métadonnées le noeud racine de notre structure XML. Elles sont stockées dans le noeud XMLPacket des métadonnées EXIF. Nous avons choisi ce noeud pour éviter la perte d'information des caractères spécifiques au français, car il peut contenir des données binaires et pas seulement des caractères ASCII comme la plupart des noeuds EXIF. Les métadonnées EXIF n'existent pas dans les fichiers PNG, et nous n'avons pas trouvé de bibliothèque permettant d'insérer un autre type de métadonnées, c'est la raison pour laquelle nous générons des images en JPG.

4.3.5 Chargement de QR Codes

Dans le but de créer une application utilisable de façon concrète et régulière, nous avons élaboré des mécanismes permettant de modifier des QR Codes déjà enregistrés. Pour les QR Codes n'étant pas liés à d'autres (n'appartenant pas à une famille), il suffit d'enregistrer l'image représentant le QR Code en insérant dans les métadonnées le nœud racine de notre structure XML (voir 3.2). L'utilisation du nœud XML <metadonnees> est par ailleurs trompeuse car l'intégralité de la structure XML est stockée dans les métadonnées de l'image. Nous avons choisi de procéder ainsi car nous n'avons pas trouvé de librairie Javascript permettant de scanner facilement un QR Code à partir d'un fichier image, et la quantité de données insérées dans les métadonnées de l'image est négligeable par rapport à la taille des données représentant l'image.

Nous avons élaboré un mécanisme plus complexe en ce qui concerne l'enregistrement de QR Codes appartenant à une même famille. En effet, nous souhaitions toujours pouvoir enregistrer ces QR Codes séparément afin qu'ils puissent être imprimés, mais nous voulions empêcher la modification de l'un des QR Codes appartenant à une famille sans mettre à jour tous les autres. Pour cela, nous avons élaboré une façon d'enregistrer tous les QR Codes d'une famille dans un fichier unique, en plus des images représentant chacun des QR Codes. Nous avons choisi d'enregistrer ce fichier comme une image (voir 4.3.3) contenant la représentation XML de tous les QR Codes de la famille dans ses métadonnées, et affichant certaines informations concernant la famille sous forme de texte imprimé sur l'image. Les informations imprimées sont le nom de la famille, le nombre de QR Codes contenus et la date de création. Cette image possède en outre un fond contenant quatre QR Codes colorés afin de notifier son importance et d'éviter qu'elle soit supprimée par mégarde. Pour s'assurer que cette image soit utilisée, nous avons empêché le chargement de QR

Codes seuls appartenant à une famille.

Le chargement d'un QR Code ou d'une famille de QR Codes s'effectue dans la classe *QRCodeLoader* du Contrôleur. Elle instancie des sous-classes de *QRCode* à partir de la racine XML des QR Codes lus dans les métadonnées de l'image, et les renvoie sous forme d'objet unique pour les QR Codes seuls ou sous forme de tableau pour les QR Codes appartenant à une famille.

4.3.6 Compression

Nous avons choisi au début du projet d'utiliser une structure XML pour représenter les données stockées dans le QR Code (voir 3.2). Ce choix était motivé principalement par la facilité de gestion du XML en Javascript.

Le XML a toutefois posé des problèmes au niveau de la concision des QR Codes générés. Nous avons tenté de remplacer la structure XML par une structure JSON plus concise lors de la première semaine de décembre, mais nous avions déjà trop de dépendances au XML. Nous avons alors tenté d'utiliser des librairies convertissant le XML déjà généré en JSON avant l'insertion des données dans le QR Code, mais nous n'en avons trouvé aucune suffisamment fiable (les attributs des noeuds XML étaient très mal gérés, et les noeuds de même nom étaient fusionnés sans respecter leur ordre).

Nous nous sommes donc orientés vers une solution de compression pour compenser l'intérêt qu'avait le JSON sur le XML. La solution que nous avons adoptée consiste à remplacer toutes les chaînes de caractères connues constituant le XML par un caractère UTF-8[4] prédéfini. Notre choix s'est porté sur les caractères de l'alphabet cyrillique car la probabilité qu'ils soient utilisés par les utilisateurs de l'application est très faible, et ils ne sont codés que sur deux octets (les caractères de l'UTF-8 sont représentés sur un nombre d'octets variant de un à quatre). Nous avons donc fait l'inventaire de toutes les chaînes de caractères de longueur supérieure à deux apparaissant dans notre représentation XML, et nous avons attribué à chacune un caractère de l'alphabet cyrillique. La classe *CompresseurTexte* du Contrôleur se charge de remplacer par le caractère correspondant toutes les chaînes de caractères appartenant à la représentation d'un QR Code avant qu'il ne soit généré.

Nous avons rendu cette solution la plus évolutive possible, en ajoutant deux caractères prédéfinis au début de la chaîne compressée pour notifier qu'elle a été compressée, et donc laisser la possibilité à l'application mobile avec un test simple d'interpréter les QR Codes n'ayant pas subi la compression. Le premier caractère inséré est le premier caractère de l'alphabet cyrillique, et le second est le chiffre 1, indiquant qu'il s'agit de la première version de cette méthode de compression. D'autres versions de la compression pourront ainsi être proposées dans le futur si la structure XML de représentation des données est modifiée.

Le principal inconvénient de cette solution est qu'elle ne compresse que la représentation des données et pas les données elles-mêmes. Un QR Code contenant beaucoup de texte sera toujours volumineux. Elle est toutefois très efficace pour les QR Codes contenant peu de données comme en témoigne l'exemple ci-dessous.

```
<donneesutilisateur xmlns="http://www.w3.org/1999/xhtml" type="atomique">
  <contenu>
    <texte>Le lion appartient a la famille des felides</texte>
  </contenu>
  <famille nom="animaux" ordre="3"></famille>
</donneesutilisateur>
```

FIGURE 4.8 – Données stockées dans un QR Code simple sans compression (214 octets)

```
È1ËLe lion appartient à la famille des félidésÙanimauxH3E
```

FIGURE 4.9 – Données stockées dans un QR Code simple avec compression (63 octets, soit un gain de 70%)

4. UTF-8 (abréviation de l'anglais Universal Character Set Transformation Format - 8 bits) est un codage de caractères informatiques conçu pour coder l'ensemble des caractères du « répertoire universel de caractères codés » (Wikipédia)

4.3.7 Lecture des fichiers du drive

Pour accéder aux fichiers du drive, nous utilisons l'API [5] Google Drive REST.
Le paramétrage [6] de cette API nécessite un compte Google et les modules googleapis et google-auth-library de Node.js. À l'issue du paramétrage, un dossier caché nommé .credentials est créé ; ce dossier contient un fichier contenant toutes les informations privées permettant à l'application de se connecter au drive. Ces informations sont chargées à chaque requête de connexion.

4.3.8 Limite de la taille des QR Codes

La taille du QR Code peut influer sur sa lecture. En effet si la taille du QR Code est trop grande, l'application mobile ne pourra pas le détecter. Pour pallier à ce problème, nous avons mis en place une restriction sur la taille. Pour ce faire, à chaque fois qu'il y a une requête pour une prévisualisation, on vérifie si la taille du QR Code après compression (voir 4.3.6) ne dépasse pas un seuil que nous avons établi.

5. Application Programming Interface : est ensemble normalisé de classes, de méthodes ou de fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. (Wikipédia)
6. https ://developers.google.com/drive/v3/web/quickstart/nodejs

Chapitre 5

Application mobile

5.1 Application existante

Plusieurs fonctionnalités étaient déjà implémentées dans l'application mobile initiale (voir 1.1). Elle était notamment déjà capable de lire le contenu d'un QR Code via la synthèse vocale de l'API Google text-to-speech et d'interpréter des données structurées en JSON[1]. En outre, elle offrait déjà la possibilité de lire à la suite plusieurs QR Codes détectés simultanément.

Notre intervention sur l'application a donc consisté à l'adapter aux nouveaux besoins. Nous avons ainsi modifié certaines parties du code de l'application, notamment ce qui concerne l'interprétation de la structure de données qui est maintenant représentée en XML (voir 3.2 et 4.3.6).

La lecture d'un QR Code se fait donc désormais de la manière suivante. Pour les QR Codes n'appartenant pas à une famille, seul le premier champ est lu (avec la synthèse vocale s'il s'agit d'un texte, ou joué s'il s'agit d'un fichier audio). Il faut effectuer un balayage de l'écran vers la gauche ou la droite pour lire les champs adjacents.

Pour les QR Codes lus simultanément et appartenant à une famille, ils sont lus dans l'ordre spécifié par la famille, tout en permettant toujours le balayage de l'écran pour passer d'un champ à un autre, ou d'un QR Code de la famille à un autre.

5.2 Téléchargement de fichiers

Le téléchargement des fichiers audio s'effectue en utilisant la classe *DownloadManager* d'Android, qui permet de récupérer un fichier distant à partir de son URL[2]. Elle requiert cependant des permissions pour pouvoir stocker les données téléchargées dans l'espace mémoire du téléphone. Seul l'identifiant Google Drive du fichier est stocké dans le QR Code, l'application doit donc recréer l'URL de téléchargement à partir de cet identifiant. Une fois le fichier téléchargé, il est joué avec la classe *MediaPlayer* d'Android.

Une version antérieure de l'application utilisait l'API Google Drive REST pour télécharger les fichiers. Cette version avait pour inconvénient majeur qu'elle forçait la connexion à un compte Google pour télécharger les fichiers, ce qui était peu commode pour les utilisateurs, mais surtout qui posait de graves problèmes de sécurité pour le compte utilisé. En effet, tous les utilisateurs de l'application obtenaient un accès aux identifiants du compte Google utilisé.

5.3 Adaptation aux nouveaux QR Codes

Afin de pouvoir interpréter correctement les QR Codes scannés, nous avons créé un modèle en Java pour l'application Android. Ces classes sont calquées sur les classes du modèle de l'application de bureau (voir 4.1.2). Les objets du modèle sont ainsi instanciés à partir des informations lues dans les QR Codes. Ces objets sont ensuite traités pour obtenir le comportement voulu selon chaque type de QR Code scanné.

1. JavaScript Object Notation (https ://www.json.org)
2. Uniform Resource Locator (Wikipédia)

Chapitre 6

Conclusion

Tout au long du développement, nous nous sommes efforcés de produire des applications répondant concrètement aux demandes de l'institut Montéclair, tout en gardant à l'esprit que les besoins de l'institut pourraient évoluer. Par conséquent, nos applications ne se veulent pas être des versions définitives. Nous avons tenté de les rendre ouvertes à l'extension pour des versions futures avec peut-être d'autres types de QR Codes, d'autres types de contenus ou d'autres méthodes d'optimisation de la quantité de données contenues dans les QR Codes.
Cette volonté s'est traduite par la réalisation d'une architecture des classes souple et non spécifique aux besoins qui nous ont été transmis. Nous avons en outre fait en sorte que la compression des données, qui est la partie la plus susceptible d'être modifiée, soit modulable tout en restant facilement compatible avec les QR Codes que nous générons actuellement.

En Décembre, lorsque le cours dans lequel Jules, Rahmatou Walet, David et Alassane s'est terminé, nous avions un couple qui nécessitait encore quelques modifications des deux côtés pour être livrables et fonctionnels. Nous avons pu compter sur Jules du côté de l'application mobile et Alassane du côté de l'application bureau pour avoir finalisé le tout pour le rendre fonctionnel et livrable.

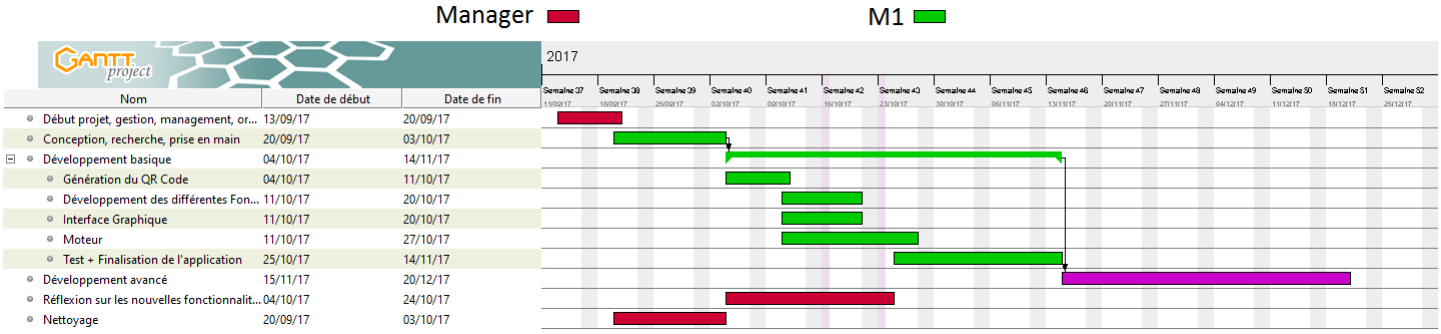
Un nouveau projet peut débuter à partir de là, comme nous l'avons fait à partir de l'application mobile initiale. En plus de l'implémentation d'éventuelles nouvelles demandes de l'institut Montéclair, un travail complexe mais intéressant pourra être effectué pour compresser le contenu des QR Codes de manière plus efficace, en compressant le texte contenu dans les QR Codes à partir d'un dictionnaire fixe externe. Une solution de transformation du texte en son directement sur l'application de bureau pourrait également permettre de créer des QR Codes contenant des textes de grande taille. Un travail supplémentaire sur l'interface graphique de l'application de bureau peut être également intéressant, notamment au niveau de l'ergonomie.

Annexes

.1 Encadrement

.1.1 Diagramme de Gantt

Nous avons établis un premier découpage simple et approximatif des tâches.



.1.2 Devis

Ce devis n'est en aucun cas contractuel, et ne reflète en aucun cas le réel coût du projet. Il n'est ici que pour nous permettre d'apprendre et découvrir la rédaction d'un tel document.

UFR Sciences

Devis No. 1

2 Boulevard de Lavoisier
49045 ANGERS cedex 01
France

Date d'émission du devis	05/10/2017
Emis par	UFR Sciences
Contact client	BARICHARD Cécile
Date de début de la prestation	27/09/2017
Durée estimée prestation	

Destinataire :
Institut Montéclair
62 Rue de la Picotière
49000 Angers
France

Description	Quantités	Unités	Prix unitaires HT	% TVA	TVA	TOTAL TTC
Développeur	4	h.	300,00 €/jours	20,00 %	1920,00 €	11 520,00 €
Jours	8	jours				0
Chef de projet	2	h.	500,00 €/jours	20,00%	1600,00 €	9600,00 €
Déplacements à l'institut	10	km	0,16 €	20,00 %	0,32 €	1,92 €

Total HT	17601,60 €
TVA	3520,32 €
Total TTC	**21121,92 €**

Signature du client (précédée de la mention « Bon pour accord »)

UFR Sciences Angers
2 Boulevard Lavoisier
49045 ANGERS cedex 01
France

Contact
Corentin Talarmain & Thomas Calatayud
Téléphone : 02 41 73 53 53
Mail : corentin.talarmain@etud.univ-angers,fr

.2 Organisation

.2.1 Répartition des tâches

	David Dembele	**Alassane Diop**	**Jules Leguy**	**Rahmatou Walet Mohamedoun**
1	Conception, uml, diagramme de classe, diagramme de séquence, cas d'utilisation.	Conception, uml, diagramme de classe, diagramme de séquence, cas d'utilisation.	Conception, uml, diagramme de classe, diagramme de séquence, cas d'utilisation.	Conception, uml, diagramme de classe, diagramme de séquence, cas d'utilisation.
2	Prise en main de l'application android.	Mise en place HTML,CSS(bootstrap) pour l'interface graphique.	Conception Modèle.	Mise en place HTML, CSS(bootstrap) pour l'interface graphique.
3	Classe de gestion des Qrcodes.	API google: -google_drive -text_to_speech	Implémentation Modèle Application Bureau.	Développement de l'interface graphique, pour correspondre aux attentes graphiques.
4	Fonction de parsage des Qrcodes.	Prévisualisation QR Code.	Implémentation de QR Codes.	Travail sur une version alternative de l'interface graphique.
5		Générer QR Code à partir du formulaire.	Génération de QR Codes avec texte en braille central.	
6		Charger QR Code.	Insertion et lecture des métadonnées des images.	
7		Exporter famille de QR Code.	Création d'objets du modèle à partir d'une image.	
8		Écoute individuelle d'un champ.	Compression données QR Code/Décompression.	
9		Champ en braille et palette couleur.	Gestion des droits Android.	
10			Téléchargement fichier Android.	

.2.2 Calendrier

<table>
<tr><th rowspan="2"></th><th colspan="3">Application de bureau</th><th rowspan="2">Application mobile</th></tr>
<tr><th>Modèle</th><th>Vue</th><th>Contrôleur</th></tr>
<tr><td>Semaine 2
04/10</td><td>Conception Modèle.</td><td>Mise en place HTML, CSS(bootstrap) pour l’interface graphique.</td><td></td><td></td></tr>
<tr><td>Semaine 3
11/10</td><td rowspan="2">Implémentation Modèle</td><td rowspan="2">Développement de l’interface.</td><td></td><td></td></tr>
<tr><td>Semaine 4
18/10</td><td></td><td></td></tr>
<tr><td>Semaine 5
25/10</td><td></td><td>Utilisation de text_to_speech pour la lecture des données.
Développement de l’interface.</td><td>Génération QR Codes avec image centrale.</td><td rowspan="3">Prise en main Application Android.</td></tr>
<tr><td>Semaine 6
01/11</td><td></td><td>Prévisualisation QR code.
Développement de l’interface.</td><td rowspan="2">Écriture de métadonnées dans l’image générée.</td></tr>
<tr><td>Semaine 7
8/11</td><td></td><td>Chargement QR code.
Développement de l’interface.</td><td rowspan="3">Création d’une classe de gestion des QR codes.</td></tr>
<tr><td>Semaine 8
15/11</td><td></td><td colspan="2" rowspan="2">Création Champ en braille et palette couleur.
Développement de l’interface.
Création d’un objet QR Code à partir d’une image enregistrée.
Compression texte avec algo LZ.</td></tr>
<tr><td>Semaine 9
22/11</td><td></td><td rowspan="2">Création fonction de parsage des données QR code.</td></tr>
<tr><td>Semaine 10
29/11</td><td>Texte central en braille</td><td colspan="2">Génération image de famille.
Compression texte avec solution ad-hoc
Exportation image de famille de QR codes.</td></tr>
<tr><td>Semaine 11
06/12</td><td></td><td colspan="2">Release démonstration.</td><td>Téléchargement fichier Android.
Droits Android.
Compression/Décompression données.
-Modification de la fonction swipe.</td></tr>
</table>

.3 Manuel utilisateur

En complément du rapport, nous proposons un manuel destiné aux utilisateurs de l’application de bureau. Ce manuel décrit les différentes interfaces et boutons permettant d’utiliser l’application.