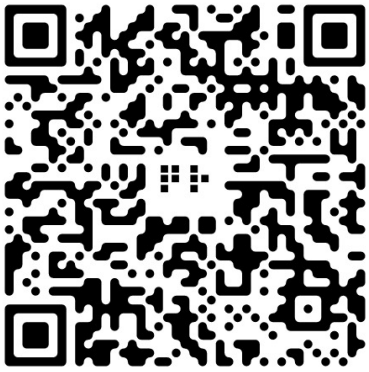

Développement d'un couple d'applications bureau et mobile

Génération et lecture de QR Codes pour l'institut Montéclair

David Dembele
Alassane Diop
Jules Leguy
Rahmatouwalet Mohamedoun

Décembre 2017

Table des matières

Chapitre 1

Introduction

1.1 Contexte

Dans le cadre du module Concrétisation Disciplinaire de notre première année de Master 1 mention Informatique à l'Université d'Angers, nous avons développé une application de bureau et poursuivi le développement d'une application mobile.

Ces applications répondent 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.

Ce projet a été encadré par Corentin Talarmain et Thomas Calatayud, deux étudiants en deuxième année de Master, dans le cadre de leur module Gestion de Projet.

L'objectif du projet est de fournir la possibilité aux transcripteurs 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 contenu dans des QR Codes. Nous avons donc développé une application de bureau permettant de générer ces QR Codes selon les contraintes décrites dans le cahier des charges. Nous avons également adapté l'application mobile existante à notre nouveau format de données.

1.2 Cahier des charges

Nous avons structuré notre projet pour répondre aux deux demandes principales qui nous ont été faites. La première demande était de pouvoir utiliser les QR Codes comme légendes de cartes de géographie. Cela implique de pouvoir gérer la lecture de QR Codes adjacents dans un ordre prédéfini, qu'importe l'ordre de lecture effectif (LIEN FAMILLES).

L'autre demande était de pouvoir utiliser des QR Codes pour lire des sons dans les pages d'un livre, 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 (LIEN ENSEMBLE).

De plus, 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 (LIEN GENERATION QR).

1. Travail Encadré de Recherche

Chapitre 2

Organisation

2.1 Répartition des tâches

2.2 Planning

Chapitre 3

Conception

3.1 Types de QRCodes

À partir des contraintes définies dans les cahier des charges (REF CAHIER CHARGES), il a fallu définir précisément quels types de QR Codes allaient être crées. 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 (LIEN TYPES QR) et à celle du Modèle (LIEN MODÈLE), afin d'assurer la cohérence de l'ensemble.

Notre choix s'est porté sur une structure XML [1]. Cette structure a montré ses faiblesses par la suite (voir REF CHOIX XML), mais avait déjà crée trop de dépendances pour être modifiée.

Afin de minimiser la quantité de données stockées dans le QR Code, seules les données indispensables à l'interprétation du QR Code par l'application mobile y sont stockées. Les données annexes nécessaires au chargement d'un QR Code par l'application de bureau sont stockées dans les métadonnées de l'image (REF CONTROLLEUR/METADONNÉES). On y trouve les données relatives au nom et aux couleurs du QR Code, ou encore au nom des fichiers contenus.

La structure fait apparaître clairement la dichotomie entre les données stockées dans le QR Code (noeud <donneesUtilisateur>) et les données stockées dans les metadonnées 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>).

1. Hypertext Markup Language

L'appartenance à une famille (REF TYPES QR) 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.

Un exemple complet de représentation XML d'un QR Code est visible dans la figure ci-dessous.

```
<qrcode>
  <donneesutilisateur type="atomique">
    <contenu>
      <texte>champ1</texte>
      <fichier url="URLFICHIER"></fichier>
      <texte>champ2</texte>
    </contenu>
    <famille nom="famille" ordre="4"></famille>
  </donneesutilisateur>
  <metadonnees>
    <fichiers>
      <fichier url="URLFICHIER" nom="fichier1"></fichier>
    </fichiers>
    <colorqrcode color="#085a0c"></colorqrcode>
    <textebraille texte="QR"></textebraille>
    <colorbraille color="#f11313"></colorbraille>
  </metadonnees>
</qrcode>
```

FIGURE 3.1 – Représentation d'un QR Code

3.3 Architecture de l'application de bureau

Chapitre 4

Réalisation

4.1 Application de bureau

4.1.1 Architecture

4.1.2 Modèle

4.1.3 Vue

4.1.4 Contrôleur

4.2 Application mobile

4.2.1 Application existante

4.2.2 Adaptation aux nouveaux QR Codes

4.2.3 Téléchargement Drive

4.3 Contraintes et choix

Au cours du développement, nous avons dû réaliser un certain nombre de choix techniques. Nous avions parfois plusieurs alternatives et nous avons dû faire des compromis entre complexité de mise en place et efficacité, tout en préservant au mieux la cohérence de l'ensemble.

Nous listons ci-dessous les principaux choix effectués ainsi que les raisons qui nous ont poussé à les faire.

4.3.1 Langages

Développement avec le framework Electron

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 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.

Les responsables du projet ont finalement opté pour le framework Electron entre les deux premières séances. Ce framework permet de développer des applications de bureau avec des langages web (Javascript/HTML/CSS). Ce choix a été motivé par l'intérêt qu'avait le Javascript d'être facilement utilisable pour accéder à Google Drive (REF STOCKAGE DRIVE).

Utilisation d'une structure XML et compression

Nous avons choisi au début du projet d'utiliser une structure XML pour représenter les données stockées dans le QR Code (REF REPR. DONNEES). 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 (REF COMPRESSION). Cette solution offre de très bon résultats (toutes les chaînes XML de plus de deux caractères sont compressées en un caractère utf-8 codé sur deux octets). Toutefois, elle ne compresse que la représentation des données et pas les données en elles-mêmes.

4.3.2 Technologies et formats

Stockage des fichiers distants dans Google Drive

Le stockage des fichiers sur un serveur distant était indispensable car on ne peut pas stocker des données binaires représentant un fichier mp3 dans un QR Code de taille raisonnable. Les transcripteurs de l'institut l'utilisant déjà, le drive de Google a été la solution la plus simple. Cette solution présente également l'avantage de ne pas nécessiter la mise en place et la location d'un serveur.

Connexion à Google Drive dans l'application Android

L'utilisation de Google Drive a pour principal inconvénient de forcer l'utilisation des API [1] de Google. En effet, même pour un fichier disponible publiquement, on ne peut pas le télécharger directement sans passer par un navigateur ou par une des API Google. La commande wget sur l'URL du fichier renvoie la page de téléchargement HTML du fichier et pas le fichier lui-même. Nous aurions peut-être pu récupérer le fichier en analysant la page HTML reçue, mais cela aurait rendu l'application dépendante à la syntaxe de cette page pouvant être modifiée par Google à tout moment.

Pour ces raisons, nous avons été contraints d'implémenter l'API Google Drive REST dans l'application Android, et donc d'imposer une authentification pénible aux utilisateurs.

Format de sauvegarde des images (JPG)

Afin de pouvoir insérer des informations dans les métadonnées des images générées par l'application de bureau (REF STRUC DONNEES), nous avons recherché les librairies Javascript permettant de créer une image à partir d'un canvas HTML5 et d'insérer les métadonnées pendant le processus de génération de l'image. Nous n'avons trouvé que des librairies permettant d'insérer des métadonnées de type EXIF [2], qui sont présentes dans les fichiers JPEG mais pas PNG. Il s'agit de la raison pour laquelle nous avons utilisé ce format.

4.3.3 Choix d'implémentation

Gestion des droits et bouton de connexion dans l'application Android

Le bouton de connexion au drive de Google (REF CONNEXION DRIVE ANDROID) est né de la volonté de rendre l'application Android utilisable pour lire les QR Codes ne nécessitant pas de connexion au drive sans avoir à se connecter. Cela respecte la philosophie des dernières versions d'Android de ne demander les permissions que lorsqu'elles sont nécessaires.

1. Application Programming Interface : Ensemble ce classes, de méthodes et de fonctions. Ici pour accéder aux données stockées sur Google Drive.
2. Exchangeable Image File Format

Les permissions indispensables au fonctionnement de l'application (caméra, vibreur) sont demandées au démarrage de l'application, tandis que la connexion au drive se fait de manière volontaire.

La présence du bouton sur l'écran de détection provient de la complexité de passer des objets complexes entre deux activités sur Android. Il était initialement prévu de placer ce bouton dans l'écran d'options mais cela nous aurait demandé trop de temps à l'approche du délai de fin de projet. Pour laisser tout de même la possibilité de se déconnecter du drive quand la connexion est effectuée et que le bouton disparaît dans un souci d'ergonomie, un bouton de déconnexion est présent dans l'activité d'options.

Chapitre 5

Conclusion