

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

1	Introduction	2
1.1	Contexte	2
1.2	Objectifs	2
2	Modélisation	3
2.1	Types de QR Codes	3
2.2	Représentation des données	3
3	Application de bureau	5
3.1	Conception	5
3.1.1	Langages	5
3.1.2	Architecture	5
3.2	Implémentation	5
3.2.1	Interface graphique	5
3.2.2	Prévisualisation	5
3.2.3	Génération de QR Codes	5
3.2.4	Gestion des métadonnées	6
3.2.5	Chargement de QR Codes	6
3.2.6	Compression	6
3.2.7	Lecture des fichiers du drive	6
3.2.8	Limite de la taille des QR Codes	6
4	Application mobile	7
4.1	Application existante	7
4.2	Connexion à Google Drive	7
4.3	Adaptation aux nouveaux QR Codes	7
5	Conclusion	8
5.1	Réponse aux demandes initiales	8
5.2	Évolutions possibles	8
	Appendices	8
.1	Organisation	10
.1.1	Répartition des tâches	10
.1.2	Calendrier	10
.2	Manuel Utilisateur	10

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 transpositeurs 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 TER¹ 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.

1.2 Objectifs

Les transpositeurs 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 transpositeurs 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 (LIEN FAMILLES).

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 (LIEN ENSEMBLE).

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 (LIEN GENERATION QR), ces caractères étant imprimés avec une encre spéciale gonflant à température élevée.

1. Travail Encadré de Recherche

Chapitre 2

Modélisation

2.1 Types de QR Codes

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 créés. Une première version de notre architecture collait aux besoins de l’Institut Montclair 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).

2.2 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 XML¹. Cette structure a montré ses faiblesses par la suite (voir REF CHOIX XML), mais avait déjà créé 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 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>`).

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 2.1 – Représentation d'un QR Code

Chapitre 3

Application de bureau

3.1 Conception

3.1.1 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.

Les responsables du projet ont finalement opté pour le framework Electron entre les deux premières séances. Ce framework¹ 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 (REF STOCKAGE DRIVE).

3.1.2 Architecture

3.2 Implémentation

3.2.1 Interface graphique

3.2.2 Prévisualisation

3.2.3 Génération de QR Codes

Notre application utilise le script jquery-qrcode² 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 transcrip-teurs de l'institut d'insérer un texte en braille central, d'une couleur pouvant être imprimée en relief (LIEN INTRO/OBJECTIFS). Un QR Code de type atomique (REF TYPES QR) possédant des caractères centraux en braille est visible ci-dessous.

La génération des images est effectuée par la classe du Contrôleur *ImageGenerator* (LIEN ARCHITECTURE). En plus des QR Codes simples, elle permet de générer des images de sauvegarde des familles de QR Codes (LIEN TYPES QR et LIEN CHARGEMENT QR). 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 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. <https://larsjung.de/jquery-qrcode/>

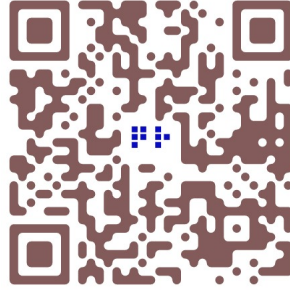


FIGURE 3.1 – QR Code simple

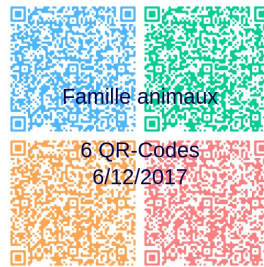


FIGURE 3.2 – Image de sauvegarde famille

3.2.4 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 Android 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). Pour cela, nous utilisons le module Node.js piexifjs³.

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 les informations annexes. Elles sont stockées dans le noeud XMLPacket des métadonnées EXIF⁴. 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.

3.2.5 Chargement de QR Codes

3.2.6 Compression

3.2.7 Lecture des fichiers du drive

3.2.8 Limite de la taille des QR Codes

3. <https://github.com/hMatoba/piexifjs>

4. Exchangeable Image File Format

Chapitre 4

Application mobile

4.1 Application existante

4.2 Connexion à Google Drive

4.3 Adaptation aux nouveaux QR Codes

Chapitre 5

Conclusion

5.1 Réponse aux demandes initiales

5.2 Évolutions possibles

Annexes

- .1 Organisation
 - .1.1 Répartition des tâches
 - .1.2 Calendrier
- .2 Manuel Utilisateur