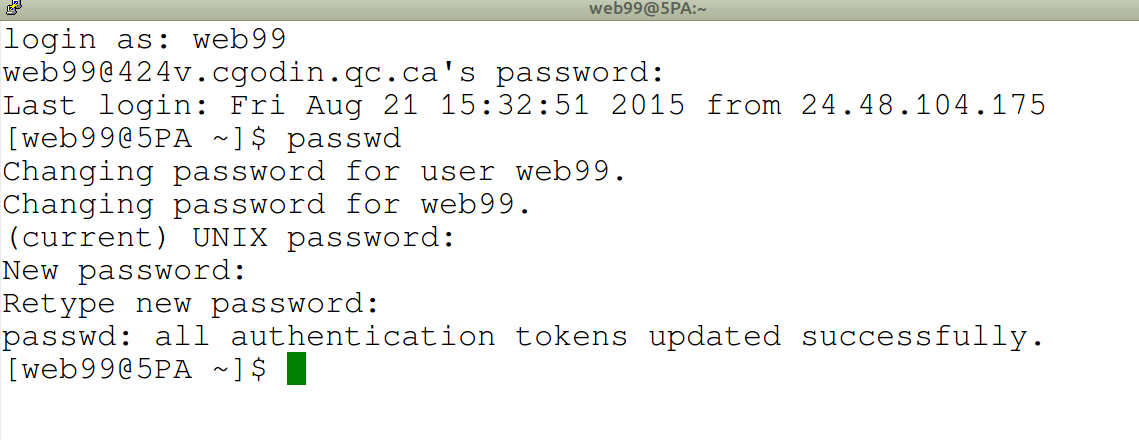
|  |  |
| --- | --- |
| attachment | **Programmation mobile**  **Automne 2016** |

(version 4 novembre 2016)

|  |
| --- |
| **Projet final (50% de la note finale)** |
| Partage de liste de lecture (PLDL) |

|  |
| --- |
| Étant donné la complexité du projet, il est possible que des erreurs ou des incohérences se glissent dans l’énoncé. Si c’est le cas, il est de votre responsabilité de le signaler à l’enseignant pour que les corrections se fassent le plus rapidement possible. Assurez-vous d’avoir toujours en main la dernière version de l’énoncé (voir la date dans l’en-tête du document). |

**Description**

Développer une application permettant le partage de listes de lecture. Cette application utilise les mêmes services qui ont été définis dans le petit projet #2. Les listes de lecture comprennent de la musique disponible sur ***Youtube***.

**Remises**

Ce projet est évalué en trois parties :

|  |  |  |
| --- | --- | --- |
| 23 novembre 2016 | Prototype | 15% de la note finale |
| 8 décembre 2016 | Présentation | 5% de la note finale |
| 8 décembre 2016 | Projet | 30% de la note finale |

**Attention**: Votre projet doit obligatoirement fonctionner avec les tables définies dans le fichier ***ProjetPLDL.sql*** qui est disponible sur LÉA. Votre application peut utiliser d’autres, mais doit fonctionner avec ces tables sans aucune autre manipulation. On ne corrigera pas un projet qui ne fonctionne pas avec ces tables. La note de zéro sera alors attribuée.

**Étapes**

1. Adapter les services concernant la sécurité et la création d'utilisateurs selon le modèle du petit projet #2;
2. Selon la liste des fonctionnalités requises, adapter et créer les autres services nécessaires à la réalisation du projet selon le modèle du petit projet #2;
3. Programmer les fonctionnalités sur Android Native;
4. Tester à partir d’un jeu d'essais.

**Critères**

1. **Le projet est réalisé par équipe de deux;**
2. Utiliser les modèles présentés lors des laboratoires et petits projets précédents;
3. Toutes les réponses du service sont en format JSON;

**Lors de la remise**

1. Le service est déployé sur le serveur 424T dans votre compte (web1 à web5);
2. L’application est déployée sur votre tablette;
3. Votre projet doit fonctionner avec la base de données qui est e sans la modifier;
4. Un jeu d'essais sera fourni. Votre application doit fonctionner avec le jeu d'essais est fourni;
5. Votre projet respecte les principes, contraintes et directives énoncés dans ce document.

**Contraintes**

1. Votre projet doit absolument fonctionner avec les tables qui sont disponibles sur LÉA (fichier ***ProjetPLDL.sql***). En cas de non-respect de cette directive puisque le projet ne pourra pas être corrigé, la note de zéro sera attribuée pour l’ensemble du projet;
2. Afin d’ajouter une musique, votre projet doit absolument utiliser API de Youtube pour faire la recherche. Vous pouvez utiliser l’API de Youtube pour d’autres fonctionnalités de votre projet. Cependant, il faut respecter la liste des fonctionnalités. Il vous faudra créer un compte Google afin d’accéder aux fonctionnalités de développeur. En cas de non-respect de cette contraine, la note de zéro sera attribuée pour cette partie du projet;

(suite)

<https://developers.google.com/youtube/android/player/>

Exemple pour une recherche

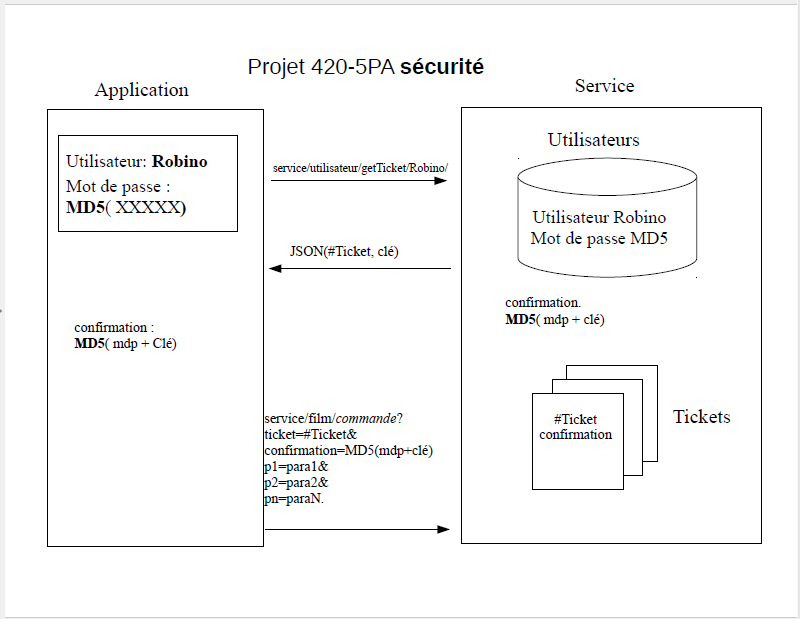
[https://www.googleapis.com/youtube/v3/search?part=snippet&key=**AIzaSyCyILNiZXjnqoN1eaCYW7NQvdj4\_qGaEjw**&q=**Bealtles**+**%22Hey%20Jude%22**](https://www.googleapis.com/youtube/v3/search?part=snippet&key=AIzaSyCyILNiZXjnqoN1eaCYW7NQvdj4_qGaEjw&q=Bealtles+%22Hey%20Jude%22)

1. La sécurité doit respecter les principes décrits dans cet énoncé (par exemple : utilisation d’un ticket pour exécuter une commande, un ticket n’est valide que pour l’exécuter d’une seule commande), En cas de non-respect, la note de zéro sera attribuée pour cette partie du projet;
2. Les services sont offerts à partir du modèle utilisé lors du petit projet 2 (EJB, JPA, modèle pour la sécurité et la création d’un utilisateur);
3. concernant la sécurité, le captcha peut être généré à partir de la classe fournie lors du petit projet 2 ou de l’API de Google mais en respectant les principes décrits dans l’énoncé.

**Sécurité**

Puisque dans le projet tous les services sont accessibles à partir du URL, il faut être en mesure de permettre l’exécution de certains services uniquement aux utilisateurs identifiés.

**Le modèle**



**Description**

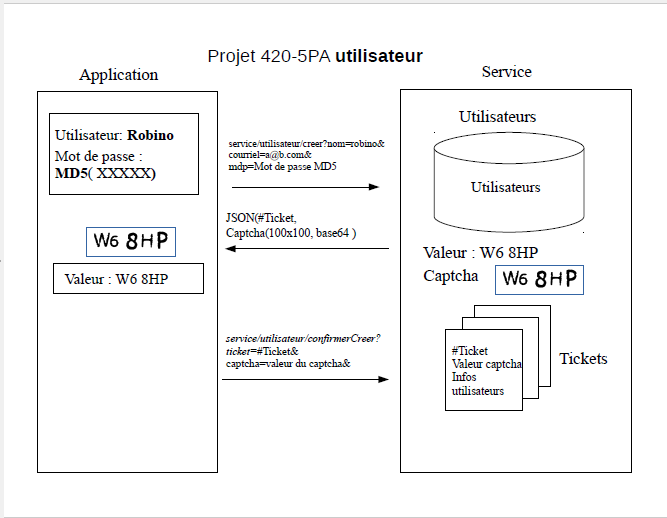
Chaque fois qu'une application veut accéder un service réservé aux utilisateurs identifiés, elle doit le faire la manière suivante :

1. L'**application** demande un ticket afin de pouvoir exécuter une commande (GetTicket/*utilisateur);*
2. Le **service** vérifie que l'utilisateur est dans la base de données. En cas d'échec un message d'erreur est retourné;
3. Le **service** génère une clé. Une séquence aléatoire de caractère (par exemple : Patate)
4. Le **service** produit une **chaîne de confirmation** en format MD5 à partir de la clé et du mot de passe de l'utilisateur en format MD5 qui est disponible dans la base de données;
5. Le **service** produit un ticket (#Ticket, chaîne de confirmation). Ce ticket est stocké dans une structure de données;
6. Le **service** retourne un flux JSON comprenant le #ticket et la clé (exemple Patate);
7. L'**application** produit une **chaîne de** **confirmation** MD5 à partir de la clé reçue dans le flux JSON et du mot de passe de l'utilisateur en format MD5 qui est disponible dans une variable locale;
8. L'**application** demande au service l'exécution d’une commande (Commande/#Ticket/**Chaîne de confirmation**/p1/p2…/pn;
9. Le **service** reçoit la demande. Il compare la chaîne de confirmation reçue avec la chaîne de confirmation qu’il a conservée dans la structure Tickets.
10. Le **service** retire le ticket de la structure et la commande est exécutée uniquement si les deux chaînes de confirmation sont identiques;
11. Le **service** retour un message approprié en flux JSON selon le résultat de la requête.

**Créer un compte**

Cette partie explique comment créer un compte.

**Le modèle**



1. L'**application** indique son intention de créer un compte (créer/*nom/courriel/mot de passe);*
2. Le **service** vérifie que l'utilisateur n'est pas présent dans la base de données. En cas de présence un message d'erreur est retourné;
3. Le **service** génère un code de sécurité. Une séquence aléatoire de caractères (par exemple : valeur W6 8HP)
4. Le **service** produit un captcha en format base64.
5. Le **service** insère un ticket dans la structure de données (#Ticket, informations utilisateurs et valeur du captcha);
6. Le **service** retourne un flux JSON comprenant le #ticket et le captcha en format base64 (c’est une image qui représente la valeur);
7. L'**application** : l'utilisateur voit le captcha et tape sa valeur. C’est le code de sécurité;
8. L'**application** confirme au service d'ajouter l'utilisateur en se référant à un ticket (confirmer/#Ticket/valeur du captcha);
9. Le **service** reçoit la demande. Il compare la valeur de captcha reçue avec la valeur du captcha conservée dans la structure Tickets.
10. Le **service** retire le ticket de la structure et l'utilisateur est ajouté uniquement si les deux valeurs de captcha sont identiques.
11. Le **service** retour un message approprié en flux JSON.

**Liste des fonctionnalités**

Votre application offre les fonctionnalités suivantes :

**Publique**

1. Créer un utilisateur;
2. Consulter une liste de lecture publique;
3. Consulter une musique publique;
4. Écouter une musique publique.

**Privée (Utilisateur)**

1. Modifier ses paramètres utilisateurs (mot de passe, alias, avatar, etc.)
2. Valider un utilisateur;
3. Créer une liste de lecture qui lui appartient ;
4. Consulter une liste de lecture qui lui appartient;
5. Modifier une liste de lecture qui lui appartient;
6. Activer/désactiver une liste de lecture qui lui appartient;
7. Rendre publique/privée une liste de lecture qui lui appartient;
8. Créer une musique qui lui appartient;
9. Consulter une musique qui lui appartient; //????
10. Écouter une musique qui lui appartient; //?????
11. Modifier une musique qui lui appartient ;
12. Activer/désactiver une musique qui lui appartient ;
13. Rendre publique/privée une musique qui lui appartient;
14. Ajouter une musique qui lui appartient/ publique à une liste de lecture qui lui appartient;
15. Retirer une musique qui lui appartient/ publique à une liste de lecture qui lui appartient;
16. Transférer une musique qui lui appartient/publique d'une liste de lecture qui lui appartient/publique vers une autre liste de lecture qui lui appartient ;
17. Copier une musique qui lui appartient/publique d'une liste de lecture qui lui appartient/publique vers une autre liste de lecture qui lui appartient ;
18. Copier une liste de lecture qui lui appartient/publique vers une nouvelle liste de lecture qui lui appartient ;

A1 getActionToken

A2 login

A3 logof

A4 getMyPlaylists

A5 getPublicPlaylistList

A6 getMySongs

A7 getPublicSongsLis

A8 getAvatarList

A8 getAvatarList

AXX retirer differents objects

**Prototype (15 points de la note finale, remise 23 novembre 2016)**

Ce que vous devez présenter:

Pour chaque fonctionnalité (représente *15/nb fonctionnalités* points)

1. **10%** - créer une page HTML utilisant jQuery afin de tester le service. Cette étape est impérative. **ATTENTION** : Si la page n’existe pas, on ne peut pas tester le service et la note zéro est attribuée pour la fonctionnalité;
2. **90%** - **programmer** les services nécessaires pour accomplir la fonctionnalité.

Note 1: Une description du contenu de la page HTML et la liste des tests pour chaque fonctionnalité seront fournies ultérieurement.

Note 2 : Les tables (et les contenus) disponibles sur LÉA seront utilisées pour la correction. Si ça ne fonctionne pas avec ces tables, la note zéro est attribuée.

**Présentation (5 points de la note finale, remise 8 décembre 2016)**

Remise **: 8 décembre 2016** (avant le cours)

Quoi : Un document PowerPoint et une présentation.

* La présentation est individuelle;
* Chaque étudiant produit son propre PowerPoint;
* Le PowerPoint est remis sur le lecteur R avant le cours du 10 décembre;

Ce document contient les éléments suivants :

1. Une description de votre contribution dans le projet;
2. Vos défis;
3. Les problèmes que vous avez rencontrés;
4. Les solutions aux problèmes;
5. Une conclusion.

Critères de correction

|  |  |
| --- | --- |
| Présentation et structure du document | |
|  | Environ 10 diapos (jusqu’à -100% maximum) |
|  | Maîtrise du sujet, clarté des explications (**20%**) |
|  | Structure (**20%**) |
|  | Pertinence des exemples (**60%**) |
|  | Qualité générale du document (jusqu’à -100% maximum) |
|  | Fautes de Français (jusqu’à -10% maximum) |
|  | Durée 10 minutes (10%) |

**Projet (30 points de la note finale, remise 8 décembre 2016)**

8 décembre 2016 avant le cours (à partir du début du cours, c’est un retard. La pénalité est de 10% par jour de retard)

1. Sur le serveur 424T vos services sont déployés;
2. Sur votre tablette votre application est installée;

Durant la semaine 16 (le moment est à déterminer)

1. Votre base de données est initialisée avec les tables disponibles sur LÉA;
2. La correction de votre projet se fait durant la semaine 16. À partir d’un jeu d’essais, votre équipe démontre au professeur le bon fonctionnement de votre projet. Ce jeu d’essais sera bientôt disponible.