Spécifications Recyc’Lyon

H4413

A. Concept et démarche

L'application Android

L'idée de départ du projet était de fournir à l'utilisateur une application qui faciliterait et motiverait le recyclage des bouteilles en verre à Lyon. Le verre est un des matériaux les plus recyclables que l'on rencontre quotidiennement (proche de 100%). Son recyclage est donc crucial à la démarche de développement durable d'une ville comme Lyon.

Nous avons donc cherché à mettre en place une politique qui permettra de récompenser l'utilisateur lorsqu'il dépose ses déchets en verre. L'idée première était de rémunérer directement l'utilisateur ou bien de fournir des avantages en nature (tickets de métro …). Cependant nous craignons que cela encourage la fraude et d'autre comportements malveillants. Nous avons donc décidé de se rapprocher de la démarche déjà en place et de permettre à l'utilisateur de choisir l'association à qui les gains seront reversés.

Une autre ambition de l'équipe était de permettre au trieur de connaître la position des bennes à verres autour de lui et de savoir si celles-ci sont pleines ou non.

L'application Web

L'objectif de notre démarche, concernant l'application web, était de rendre accessibles les données fournies par les capteurs des bennes et les informations recueillies au moment des collectes. Ces données concernent principalement le remplissage des silos et a pour vocation de déterminer des plannings de ramassage plus optimaux. Nous proposerons aussi des emplacements pour de nouveaux silos à verre qui semblent manquer.

B. Conception graphique

a. Charte graphique

Couleurs de l'application

Le choix de la couleur principale a été orientée par le thème de notre application : le recyclage. Nous nous sommes donc naturellement orienté vers le vert. De plus cette couleur est également associée à une idée d'espoir et de positif que nous voulions transmettre à l'utilisateur afin de travailler sur le sentiment de rendre service que l'application a pour but de procurer. Nous avons décliné le vert en un dégradé de 3 teintes présentes dans le nuancier google design.

Dégradé de 3 teintes vertes :

Couleur	Code Hexa	Transparence (%)	Contexte d'utilisation

Vert Clair	3BCC5B	50	Information
Vert	32C24D	100	Bandeau
Vert Foncé	217F33	100	Bouton

Material Design

Bouton :

Paramètre	Valeur
Corner Radius	10
Color	gE217F33
Text Color	FFFFFF
Drop Shadow	Oui

App Bar :

Paramètre	Valeur
Largeur	Parent-herited
Color	32C24D
Text Color	FFFFFF
Icon	Hamburger - Retour

b. Diagramme d'enchaînement des fenêtres

c. Mockup

a) Connexion à l’application

b) Inscription

c) Mot de passe oublié

d) Accueil

e) Menu hamburger

f) Faire un dépôt

g) Carte des bennes

h) Mon compte

i) Scanner un emballage

j) Planning de collecte

k) Paramètres

C. Spécifications fonctionnelles

a. Cas d’utilisation

b. Périmètre fonctionnel de l'application mobile

SPEC_FF_01 : Créer un compte utilisateur

L'utilisateur doit pouvoir créer un compte afin de s'identifier auprès du système. Lors de la création du compte, l'utilisateur définit ses informations personnelles :

- Identifiants : adresse mail et mot de passe
- Informations facultatives : Adresse, Âge, Sexe, Nom, Prénom

Il accepte aussi que ses données soient stockées sur des serveurs distants.

SPEC_FF_01 BIS : Accepter les conditions d'utilisation

L'utilisateur devra, après son inscription, lire et accepter les conditions d'utilisation du service (application). Ces conditions définissent notamment l'utilisation des données collectée conformément aux principes de protection des données personnelles définies par le CNIL (sécurisation, collecte minimale, objectifs …)

SPEC_FF_02 : Identification

L'utilisateur doit pouvoir, après avoir créé un compte utilisateur, se connecter avec ses identifiants(email + mot de passe) pour accéder à son espace personnel.

SPEC_FF_03 : Consulter ses informations personnelles

Une fois connecté, l'utilisateur peut consulter ses informations personnelles à partir de son espace personnel.

SPEC_FF_04 : Modifier ses informations personnelles

Lors de la consultation de ses informations, l'utilisateur doit pouvoir les modifier.

SPEC_FF_05 : Démarrer un dépôt (scan d'une benne)

Une fois connecté, l'utilisateur peut scanner un QR code sur une benne pour commencer un dépôt dans la benne scannée. Lors d'un dépôt l'utilisateur ne peut interagir avec le système tant qu'il n'a pas confirmé la fin du dépôt. Automatiquement, 10 minutes après le début d'un dépôt, il est clôturé.

SPEC_FF_06 : Confirmer la fin d'un dépôt

Après avoir démarré un dépôt, l'utilisateur peut uniquement confirmer la fin d'un dépôt. Cette confirmation lance automatiquement le calcul de sa contribution et lui permet d'accéder (via son historique) aux détails liés à sa contribution.

SPEC_FF_07 : Consulter son historique de contribution

Une fois connecté, l'utilisateur peut consulter son historique de contributions sous la forme d'une liste triée par date (de la plus récente à la plus ancienne). Il peut consulter les détails d'une contribution spécifique.

SPEC_FF_08 : Consulter les détails d'une contribution

Via son historique, l'utilisateur peut accéder aux détails de ses contributions et avoir accès aux différentes informations calculées lors de sa contribution (cf SPEC_FF_07)

SPEC_FF_9 : Consulter la carte

Une fois connecté, l'utilisateur peut consulter une carte de la ville de Lyon sur laquelle figure les emplacements ainsi que le remplissage binaire (remplie ou non) actuel des bennes à verre.

SPEC_FF_10 : Analyser un produit

Une fois connecté, l'utilisateur peut utiliser un scanner de code barre pour scanner son produit. Le système va afficher les règles de recyclage pour ce produit (poubelle jaune, verte, marron etc..)

SPEC_FF_11 : Consulter le planning de collecte

L'utilisateur doit pouvoir consulter les règles de collecte des encombrants, des bennes à verre, etc… Toutes les informations publiques liés à la collecte seront centralisées.

SPEC_FF_12 : Demander la collecte d'encombrants

L'utilisateur peut demander la collecte d'encombrant en suivant la procédure suivante : il demande la collecte d'encombrants, le système utilise sa géolocalisation afin de déterminer le responsable de sa zone et affiche à l'utilisateur le numéro de ce responsable.

SPEC_FF_13 : Modifier les paramètres

L'utilisateur doit pouvoir modifier les paramètres de l'application : par exemple les notifications, la confidentialité, ...

SPEC_FF_14 : Se déconnecter

L'utilisateur doit pouvoir se déconnecter.

SPEC_FF_15 : Récupérer mot de passe

En cas d'oubli de mot de passe, un utilisateur doit pouvoir récupérer saisir son mail afin de lancer la procédure de récupération de mot de passe. L'utilisateur devra alors suivre un lien qui lui sera envoyé par mail pour modifier son mot de passe.

SPEC_FF_16 : Supprimer ses données personnelles

L'utilisateur de l'application mobile doit pouvoir demander la suppression de son compte et de ses données, conformément à l'article 38 de la loi « informatique et libertés » du 6 janvier 1978 modifiée.

c. Périmètre fonctionnel de l'application web

SPEC_FF_17 : Connexion au portail

L'administrateur doit pouvoir se connecter au portail d'administration grâce à des identifiants et mot de passe spécifiques.

SPEC_FF_18 : Visualiser les demandes d'encombrant

L'administrateur doit pouvoir observer les demandes d'enlèvement d'encombrants sur une carte et dans une liste.

SPEC_FF_19 : Visualiser les données des poubelles

L'administrateur doit pouvoir visualiser sur une carte les données renvoyées par les poubelles.

SPEC_FF_20 : Visualiser les règles de gestions calculées par le serveur

L'administrateur doit pouvoir avoir accès aux règles suggérées par l'analyse des données des poubelles (ramassage, …).

d. Périmètre fonctionnel back office

SPEC_FB_01 : Calcul d'une contribution

Lors de la confirmation de la fin d'un dépôt, le système calcul la contribution apportée avec les informations suivantes :

- Durée du dépôt
- Nombre de bouteilles déposées et l'équivalent monétaire du don
- Identifiant de la benne à verre utilisée
- Association à laquelle les dons sont reversés

SPEC_FB_02 : Suggestion de passage

Le système va calculer des suggestions de passage ou non par poubelle en fonction des données des capteurs.

D. Spécifications techniques

a. Choix technologiques

Pour la partie application mobile, nous avons choisi d'utiliser l'**API 21 d'Android** (version 5.0 minimum), afin d'obtenir un compromis intéressant entre fonctionnalités intéressante et conserver un nombre d'utilisateurs potentiels important. La partie mobile ne sera pas destinée à iOS, uniquement Android.

L'application web qui permettra au Grand Lyon de gérer et d'administrer les capteurs utilisera le framework **Bootstrap**. Tous les traitements seront effectués sur le serveur web, qui utilisera les framework **Nodejs** (javascript) et **Express** et stockera les données dans un système de base de données **MongoDb**. L'application mobile et l'application web, ainsi que les capteurs, communiquerons avec le serveur par des requêtes HTTP.

Le schéma suivant récapitule le paragraphe précédent.

b. Connexion au serveur

1. Lorsqu'un utilisateur veut se connecter à l'application, il envoie ses credentials (mail + mdp).
2. Le serveur chiffre le mot de passe et vérifie la validité.
3. Si les identifiants correspondent, il accepte la connection et instancie une session dont l'id est renvoyé dans un cookie. L'idSession contenu dans ce cookie permet d'authentifier une requête.
4. Seules les requêtes authentifiées sont acceptés sinon le serveur renvoie un statut 401.

Certaines routes ne requièrent pas d'authentification (notamment pour la gestion de mot de passe où un token unique est généré et transmis par email assurant la sécurité de la démarche).

Le principe de fonctionnement est le même entre le serveur et le site web accessible aux administrateurs.

Les utilisateurs de l'application et les utilisateurs du portail sont strictement distincts.

c. End Points

BASE_URL = "https://pld-smart.azurewebsites.net/api"

i. Users

Créer un nouvel utilisateur		
POST	/users	Code de sortie : 200, 401, 500

json :
{
"nom": string,
"motDePasse": string,
"mail": string,
"adresse": string,
"dateNaissance": string,
"sexe": int,
"idAssoc": id,
"isAdmin": bool
}
		json :
		{
		"idUtilisateur": id
		}

Choisir une association		
PUT	/users/{idUtilisateur}	Code de sortie : 200, 401, 500

json :
{
"idAssoc": id
}
		json :
		{
		"nom": string,
		"motDePasse": string,
		"mail": string,
		"adresse": string,
		"dateNaissance": string,
		"sexe": int,
		"idAssoc": id,
		"isAdmin": bool
		}

Modifier un utilisateur		
PUT	/users/{idUtilisateur}	Code de sortie : 200, 401, 500

json :
{
"nom": string,
"mail": string,
"adresse": string,
"dateNaissance": string,
"sexe": int,
"isAdmin": bool
}
if request body contains "idAssoc",
call the "Choisir une association"
		json :
		{
		"nom": string,
		"motDePasse": string,
		"mail": string,
		"adresse": string,
		"dateNaissance": string,
		"sexe": int,
		"idAssoc": id,
		"isAdmin": bool
		}

Supprimer un compte utilisateur		
DELETE	/users/{idUtilisateur}	Code de sortie : 200, 401, 500

json :
{
"idUtilisateur": int,
}		

Récupérer les informations d'un utilisateur par son id		
GET	/users/{idUtilisateur}	Code de sortie : 200, 401, 500

		json :
		{
		"idUtilisateur": id,
		"nom": string,
		"mail": string,
		"adresse": string,
		"dateNaissance": string,
		"sexe": int,
		"idAssoc": id,
		"isAdmin": bool
		}

Récupérer les informations d'un utilisateur par son id		
POST	/users/forgotPassword	Code de sortie : 200, 401, 500

json:
{
"mail": string
}
		html: html send in the mail
		&&
		email send.

Récupérer les informations d'un utilisateur par son id		
GET	/users/forgotPassword/{idUser}/{idToken}	Code de sortie : 200, 401, 500
		html: forgotten password html page

Récupérer les informations d'un utilisateur par son id		
POST	/users/forgotPassword/{idUser}/{idToken}	Code de sortie : 200, 401, 500

json:
{
"motDePasse": string
}
		redirection: /connexion.

Récupérer tous les utilisateurs		
GET	/users/	Code de sortie : 200, 401, 500

		json : {
		[
		{
		"idUtilisateur": id,
		"nom": string,
		"mail": string,
		"adresse": string,
		"dateNaissance": string,
		"sexe": int,
		"idAssoc": id,
		"isAdmin": bool
		},
		…]}

ii. Poubelles

Récupérer la liste des coordonnées et le statut des poubelles		
GET	/poubelles/infos	Code de sortie : 200, 401, 500

		json :
		{
		[
		{
		“latitude”: double,
		“longitude”: double,
		“remplissage” : int
		},
		...]
		}

Récupérer l’adresse d’une poubelle		
GET	/poubelles/adresse/{idPoubelle}	Code de sortie : 200, 401, 500

		json :
		{
		“_id”: id,
		“id_grandlyon”: String,
		“adresse”: String,
		“code_insee”: Number,
		“code_postal”: Number,
		“commune”: String,
		“gestionnaire”: String,
		“observation”: String,
		“numero_voie”: String,
		“voie”: String,
		“lattitude”: Number,
		“longitude”: Number,
		“remplissage”: Number
		}

Récupérer l'adresse d'une poubelle		
GET	/poubelles/	Code de sortie : 200, 401, 500

		json : {
		[{
		"_id": id,
		"id_grandlyon": String,
		"adresse": String,
		"code_insee": Number,
		"code_postal": Number,
		"commune": String,
		"gestionnaire": String,
		"observation": String,
		"numero_voie": String,
		"voie": String,
		"lattitude": Number,
		"longitude": Number,
		"remplissage": Number
		}, …]
		}

iii. Depots

Récupérer l'historique		
GET	/depot/historique/{idUtilisateur}	Code de sortie : 200, 401, 500

		json :
		{
		[{
		"id": id,
		"date": String,
		"montant": int
		"association": String
		}, ...]
		}

Récupérer tous les dépots		
GET	/depot	Code de sortie : 200, 401, 500

		json :
		{
		[{
		"id": int,
		"date": String,
		"montant": int
		"idAssociation": id,
		"idUser": id,
		"idCapteur": id
		}, ...]
		}

iv. Depots en cours

Indiquer qu'un dépôt dans un trou de benne va commencer		
POST	/depotsEnCours/demarrerScan/	Code de sortie : 200, 401, 500

json :
{
"idUtilisateur": id,
"idCapteur": id
}
		json :
		{
		"idDepot": int
		}

Indiquer qu'un dépôt dans un trou de benne est terminé		
POST	/depotsEnCours/terminerScan/{idCapteur}	Code de sortie : 200, 401, 500

		json :
		{
		"date": date,
		"montant": double,
		"idAssoc": id
		}

Incrémente un dépot en cours		
PUT	/depotsEnCours/ajoutDechet/{idCapteur}	Code de sortie : 200, 401, 500

(requête envoyé par le capteur)		json :
		{
		"_id": id,
		"date": Date,
		"montant": Number,
		"idUtilisateur": id,
		"idAssoc": id,
		"idCapteur": id
		}

v. Associations

Récupérer les informations d'une association		
GET	/associations/{idAssoc}	Code de sortie : 200, 401, 500

		json :
		{
		"idAssoc": id,
		"nom": string,
		"rib": int,
		"description": string,
		"logoUrl": string
		}

Récupérer toutes les associations		
GET	/associations	Code de sortie : 200, 401, 500

		json :
		{
		[{
		"idAssoc": int,
		"nom": string,
		"description": string,
		"rib": int,
		"logoUrl": string
		}, …]
		}

Ajouter une nouvelle association		
POST	/associations	Code de sortie : 200, 401, 500

json :
{
"nom": string,
"description": string,
"rib": int,
"logoUrl": string
}
		json :
		{
		"id": id,
		"rib": int,
		"nom": string,
		"description": string,
		"logoUrl": string
		
		}

vi. Capteurs

Recupère tous les capteurs		
GET	/capteurs	Code de sortie : 200, 401, 500

		json :
		{
		[{
		"_id": id,
		"tokenCapteur": int,
		"idPoubelle": id,
		}...]
		}

Créer un capteur		
POST	/capteurs	Code de sortie : 200, 401, 500

json :
{
"tokenCapteur": int,
"idPoubelle": id,
}
		json :
		{
		"_id": id,
		"tokenCapteur": int,
		"idPoubelle": id,
		}

vii. Suggestion Poubelles

Connexion au service		
GET	/suggestionPoubelles/locations	Code de sortie : 200, 401, 500

		json :
		{
		[{
		“latitude”: double,
		“longitude”: double
		} …]
		}

viii. Auth

Connexion au service (for Android app)		
POST	/login	Code de sortie : 200, 401, 500

json :
{
“motDePasse”: string,
“mail”: string
}
		json :
		{
		“idUtilisateur” : int
		}

Inscription au service (for Android app)		
POST	/signup	Code de sortie : 200, 401, 500

json :
{
“motDePasse”: string,
“mail”: string,
“adresse”: string,
“nom”: string,
“dateDeNaissance”: string,
“sexe”: string,
“idAssoc”: string
}
		json :
		{
		“idUtilisateur” : int
		}

Déconnexion		
GET	/logout	Code de sortie : 200, 401, 500
		redirect: /connexion

Connexion au portail (for Website)		
POST	/connexion	Code de sortie : 200, 401, 500

json :
{
“motDePasse”: string,
“mail”: string
}
		redirect: /Dashboard

ix. Admin

Récupère la liste des relèves d'une benne		
GET	/admin/benneDetails/{idPoubelle}	Code de sortie : 200, 401, 500

		json :
		{
		[{
		"_id": id,
		"date": date,
		"tauxRemplissage": double,
		"idPoubelle": id,
		}...]
		}

Récupère la liste des relèves à une date		
GET	/admin/benneDetails/{date}	Code de sortie : 200, 401, 500

		json :
		{
		[{
		"_id": id,
		"date": date,
		"tauxRemplissage": double,
		"idPoubelle": id,
		"latitude": double,
		"longitude": double
		}...]
		}

Récupère toutes les relèves		
GET	/admin/releves	Code de sortie : 200, 401, 500

		json :
		{
		[{
		"_id": id,
		"date": date,
		"tauxRemplissage": double,
		"idPoubelle": id
		}...]
		}

Créer une relève		
POST	/admin/releves	Code de sortie : 200, 401, 500

json:
{
"date": date,
"tauxRemplissage": double,
"idPoubelle": id
}
		json :
		{
		"_id": id,
		"date": date,
		"tauxRemplissage": double,
		"idPoubelle": id
		}

x. Utils

Récupère les infos d'un emballage à partir d'un code barre		
GET	/utils/codeBarre/{codeBarre}	Code de sortie : 200, 401, 500

		json :
		{
		"product": {
		"_id": id,
		"image": image,
		"nom": string,
		"emballage": tag
		}
		}

xi. Website

BASE_URL = “https://pld-smart.azurewebsites.net/”

Accéder à l’accueil		
GET	/	Code de sortie : 200, 401, 500
		`html: index.html`

Accéder aux Dashboard remplissage		
GET	/dahsboard	Code de sortie : 200, 401, 500
		`html: dash-remplissage.html`

Accéder aux Dashboard collecte		
GET	/dahsboard-collecte	Code de sortie : 200, 401, 500
		`html: dash-collecte.html`

Accéder aux Dashboard analyse		
GET	/dahsboard-analyse	Code de sortie : 200, 401, 500
		`html: dash-analyse.html`

d. Entité association de la BD