Project Loyalty

L'objectif du présent rapport est de décrire l'architecture du projet scolaire Loyalty en se basant sur la documentation « arc42 ». Il est important de préciser que nos clients tout comme nous, prestataires, sommes encore étudiant à l'école ESGI et que ceci est un projet organisé dans le cadre de notre 4ème année.

1. Introduction and goals

1.1 Requirements Overview

Qu'est-ce que Loyalty ?

Le but de Loyalty est de remplacer le système de cartes de fidélité des entreprises en le dématérialisant. Les utilisateurs, qu'ils aient le statut d'entreprise ou bien de particulier peuvent trouver leur compte dans ce nouveau système. Alors que les particuliers n'auront plus à faire à de nombreuses et encombrantes cartes de fidélités, les entreprises pourront proposer leur propre parcours de fidélité (et non plus seulement le même cadeau à X tampons sur la carte), ainsi que leur propre cadeau. Loyalty se veut comporter une application mobile et un site web, tous deux accédant aux données via une API développée par nos soins.

Principales fonctionnalités

- Valider une visite pour un particulier
- Voir ses cadeaux pour un particulier
- Voir les entreprises scannées et les cadeaux possibles
- Remettre un cadeau pour une entreprise
- Accéder à l'historique des clients pour une entreprise

Cas d'utilisation

- **Entreprise** : elle pourrait s'inscrire, créer son propre parcours de fidélité correspondant à une catégorie de personnes puis y ajouter les cadeaux souhaites via le site web
- **Particulier** : il pourrait s'inscrire, et à chaque visite d'une entreprise scanner le QRCode proposé par celle-ci. Au nombre de visite souhaite, il pourrait récupérer le cadeau accessible pour son nombre de visites en présentant son propre QRCode à l'entreprise

1.2 Quality goals

Priorité	Qualité	Scénario
P1	Interopérabilité	Il doit être possible pour le site web et l'application mobile de communiquer de la manière la plus simple et fonctionnelle avec l'API. Les erreurs doivent être documentées et prévues par les appelants.

P2	Efficacité	Le temps de réponse des différents modules et en particulier de l'API doit être le plus rapide possible, le nombre d'utilisateurs pouvant s'incrémenter rapidement chaque route de l'API devra être le plus limité en temps de réponse.
P3	Ergonomie	L'application mobile ainsi que le site web devront être le plus accessible possible, chaque fonctionnalité devra être présentée à l'utilisateur et ne pas être cachée dans des sous-menus

2. Architecture constraints

2.1 Contraintes techniques

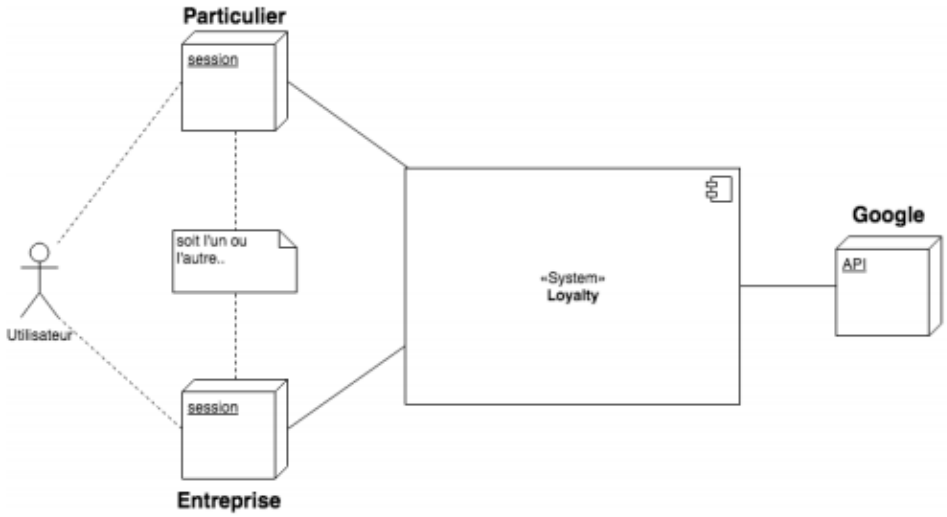
Numéro	Contrainte	Motivation
	Modules	
CT1	Application android	Il devait y avoir une application android accessible par les clients et particulier. Pour les langages, Kotlin et Java était possibles
CT2	Site web	Un site web devait être opérationnel pour que les entreprises puissent gérer leur compte, le langage n'était pas contraint
CT3	Module de base de données	Il fallait que les deux précédents modules puissent accéder à la base de données en passant par un troisième module indépendant; que ce soit une API hébergée sur un serveur ou bien directement firebase.
	Outils	
CT4	Intégration continue	Il était obligatoire dans le projet de mettre en place un système d'intégration continue (matérialisé par jenkins) pour les trois modules principaux du projet.
	Design	
CT6	Material design	Le site web et l'application android devait respecter les conventions material design définies par google
CT7	Responsivité	Le site web se devait d'être responsive, et donc visible et ergonomique sur tous les supports, ordinateur, tablette et smartphone

2.2 Contraintes organisationnelles

Numéro	Contrainte	Motivation
OC1	Equipe	William MORGADO, Michael PEAN, Jean POUGETOUX
OC2	Planning	Le planning s'étendait du 12 février 2018 au 15 juillet 2018, mais une livraison du produit devait s'effectuer chaque mois sous forme d'itération, avec une partie du produit fini que nous avons dû définir à l'avance
OC3	Mode de travail	Nous devions pratiquer une démarche itérative et incrémentale, c'est-à-dire que le projet devait avancer sur tous les modules de manières équivalentes pour chaque rendu.
OC4	Versioning	Nous avons dû utiliser Git comme gestionnaire de version pour nous partager le code sur un repository public, avec historique des commits

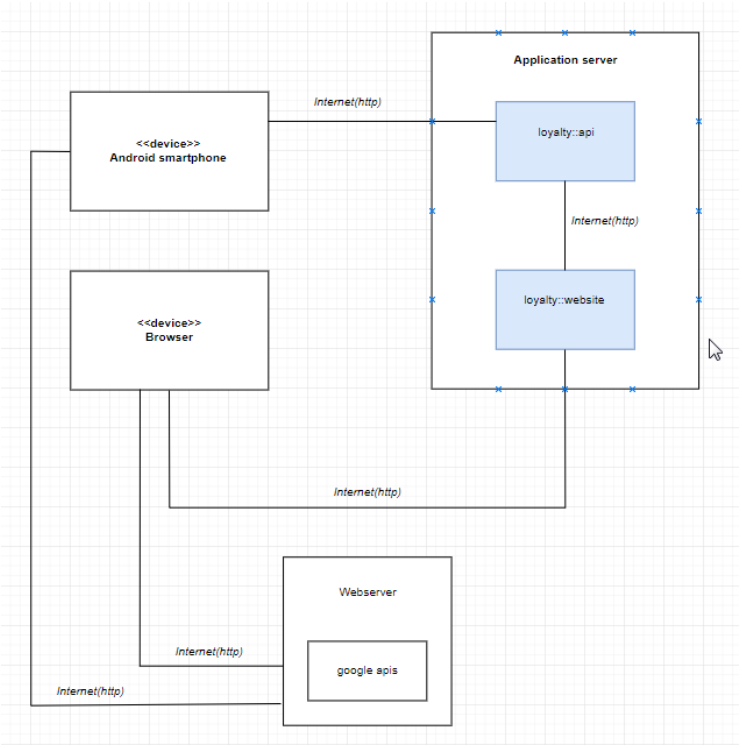
3. Context and scope

3.1 Contexte business



Plusieurs API google ont été utilisées pour gérer la localisation des particuliers et entreprises. En effet, à l'inscription les deux partis doivent pouvoir saisir une adresse réelle et reconnue par google, nous avons donc utilisé Google Maps Javascript API pour la zone de saisie automatique du site web, Google Places Javascript API pour récupérer l'objet généré par cette zone et Places SDK for Android en ce qui concerne la zone de saisie de l'application android.

3.2 Contexte technique



4. Solution strategy

Numéro	Qualité	Scénario	Approche
I1	Interopérabilité	Impossible de faire des requêtes post en ajax vers le server	Activation du CORS sur le server et dans le code des appels pour autoriser ces requêtes du site web et de l'application android
I2	Interopérabilité	Avoir des réponses récupérables par les différents modules	Utilisation du json comme réponse de l'API plutôt que d'autres protocoles, dans le but de faciliter sa récupération quelque soit le langage qui l'appelle
EF1	Efficacité	Eviter les lenteurs lors d'appels aux routes de l'API	Multiplication du nombre de routes pour éviter les sous-appels et requêtes sequelize complètes pour éviter plusieurs appels à la bdd
ER1	Ergonomie	Accessibilité aux fonctionnalités par l'utilisateur	Utilisation du framework materialize pour rendre l'interface web plus nette et compatible avec le material design
ER2	Ergonomie	Garder une bonne interactivité sur le site web	Utilisation du jquery pour les appels à l'api sur le site web pour pouvoir afficher les résultats en temps réel et éviter le rechargement de la page

M1	Maintenabilité	Structure du code	Utilisation du MVC pour l'application android et le site web, dans le but de maintenir une bonne lisibilité et donc une meilleure maintenabilité du code
M2	Maintenabilité	Structure du code	Utilisation d'express generator pour générer une structure nodejs claire au début de développement de l'API et site web, nette et facilement maintenable

5. Runtime view

Diagramme présentant l'action du scan d'un QRCode sur l'application android :