Projet DAR : Le Juste Prix

Vincent Ha Vinh & William Sergeant

Réalisation d'une application web
Développement d'Applications Réticulaires

1. Introduction

Ce projet s'est orienté vers un système de jeu afin de rendre le sujet et les contraintes plus ludiques.

1.1 Règles du jeu

Cette variante du jeu télévisé "Le Juste Prix" demande aux joueurs de répondre à une série de questions. L'utilisateur se voit donner un prix ainsi que deux images de produits récupérés sur le site Cdiscount.
Il lui est alors demandé de choisir le produit correspondant au prix affiché. Si la bonne réponse est donnée, alors le joueur gagne un point qui sera ajouté au score de la partie en cours ainsi que sur son score général affiché sur son profil.

2. Description technique

Cette partie effectue un tour d'horizon des technologies utilisées afin de réaliser la partie client et serveur de cette application. Les aspects de sécurité et de test feront l'objet d'une discussion dans des parties distinctes.
Nous utilisons un hébergement OVH car l'un des membres du groupe possédait déjà un compte chez eux.
Le schéma de l'application se trouve en annexe Figure 4

2.1 Client

Le client est une interface réalisée à l'aide des technologies `HTML`, `CSS`, `JavaScript`, `AJAX`. Il sert d'interface de jeu en récupérant les questions et en soumettant les réponses depuis et vers le serveur.

2.1.1 Apparence

Nous avons apporté un certain soin quand à l'apparence de celui-ci afin de fournir aux joueurs une expérience visuelle agréable et divertissante.
Pour obtenir un design d'une qualité suffisante nous utilisons des ressources fournies par le framework **Bootstrap**.
Le contenu des pages est créé à la fois

- *Statiquement* : les en-têtes, pieds de pages et tout le code commun aux pages
- *Dynamiquement* : le code HTML est généré depuis les fonctions `JavaScript` réalisant un appel à l'API du serveur

En outre, le site possède un design adaptatif et peut être consulté depuis n'importe quel terminal mobile.

2.1.2 Communication

En ce qui concerne la communication avec le serveur, les échanges se font sous forme de requêtes **XMLHttpRequest** (*XHR*) au travers de fonctions `JavaScript`. Les réponses du serveur sont reçues au format **JSON** et sont traitées par le client pour afficher le contenu adéquat.
Exemple :

- Le joueur répond à une question
- La client envoie la réponse en paramètre d'une **XHR**
- Le serveur traite la réponse
- Le serveur envoie une chaîne **JSON** contenant le statut, le score et les détails des deux produits de la question.

Lorsque le joueur répond à une question, son choix est envoyé comme paramètre d'une XHR, le serveur traite ce choix puis renvoie le résultat. Le client va alors demander une nouvelle question par XHR.

→ Requête XHR de récupération d'une question :

```
function questions_get() {
    var xhttp = new XMLHttpRequest();
    xhttp.responseType = 'json';

    /*sur reception de la reponse*/
    xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        var json =
         ↪  JSON.parse(JSON.stringify(this.response));
        if(json!=null){
            make_questions(json);
        }
        else{
            questions_get();
        }
    }};

    /*envoi asynchrone au servlet*/
    xhttp.open("GET", "server/questions" , true);
    xhttp.send();
}
```

L'utilisation du format **JSON** au lieu de **XML**, s'effectue dans un soucis d'économie des ressources réseau, la taille des messages à échanger étant significativement réduite.

2.2 Serveur

La partie Serveur de notre application s'occupe de répondre aux requêtes du client, de stocker et mettre à jour les données du jeu, et de communiquer avec l'API Products CDiscount (`https://dev.cdiscount.com/apiReference/v1`).

Nous avons choisi d'implémenter un serveur totalement indépendant du client. Cela permettra d'étendre notre application à un large public en proposant par exemple une application client Android et iOS dans le futur. Pour cela, notre serveur répond aux requêtes du client par des réponses au format JSON, facilement exploitables.

Les services fournis par le serveur sont mis à disposition par son API de servlets. Par volonté de documentation et de transparence vis-à-vis des développeurs côté client, les services proposés peuvent être testés sur une page simulant un client minimaliste, qui affiche les JSON retournés de façon structurée, à l'adresse: `http://bit.ly/DAR_API_overview`.

2.2.1 Architecture Serveur

Le serveur est composé de différents packages représentant les différente couches fonctionnelles de notre application :

- **servlets** :
  Couche "réseau" de l'application. Offre les services au client par le biais d'URLs et de paramètres. Fait le lien entre la requête du client et le bon service de la couche logique de l'application, en lui passant le JSON de retour dans lequel écrire. Cette couche permet d'abstraire la partie protocole http pour le reste de l'application.
- **controller** :
  Couche logique. Traite la demande reçue en manipulant les données du serveur, et écrit les résultats dans un JSON de retour fourni par la couche réseau.
- **model** :
  Couche modèle, contenant les POJOs (Plain Old Java Objects) représentant les données métier manipulées par le serveur. Ces classes Java ne contiennent que des attributs et les getters/setters allant avec.
- **database** :
  Couche possédant les DAOs (Data Access Objects) permettant à la couche *controller* de manipuler les données stockées sur la BDD avec les méthodes `CRUD`. Cette couche permet d'abstraire totalement le type de BDD utilisé. Il est par exemple possible écrire un nouveau package database (qui proposerait les mêmes noms de méthodes CRUD dans les DAOs) pour pouvoir utiliser une BDD relationnelle SQL avec JDBC, sans que cela ait un quelconque impact sur le reste du code.
- **tools** :
  Ce dernier package rassemble les fonctionnalités auxiliaires.
  Il contient entre autres, la classe CDiscountUtils qui permet d'abstraire la communication avec l'API Prorducts CDiscount. Ses méthodes prennent en arguments les paramètres à fournir, et renvoient le JSON résultat donné par CDiscount.

2.2.2 Sécurité Serveur

Plusieurs bonnes pratiques ont été respectées pour assurer la sécurité du serveur.

Tout d'abord, concernant la communication avec l'API Products CDiscount. Son utilisation nécessite une clé API accordée à un compte développeur CDiscount.Le code de notre application ne contient pas cette clé en dur dans ses fichiers source, pour éviter son vol sur le repo de versioning Github en accès publique.
A la place, nous stockons la clé en tant que première ligne d'un fichier "APIkey", situé dans le répertoire WEB-INF de notre application (non accessible depuis l'extérieur du site). Puis nous avons créé un ServletContextListener, nommé StartRoutine et situé dans le packages servlets, qui lors de l'initialisation du contexte du serveur, récupère le chemin du fichier APIkey depuis le contexte, pour pouvoir le lire et initialiser la clef API.

Ensuite, le serveur hash les mots de passe avec SHA256. Certes notre client s'occupait déjà de hasher le mot de passe avant son envoi au serveur, mais rappelons que nous avons développé le serveur en ayant comme objectif son interopérabilité. Donc pour que des clients ne hashant pas les mots de passe puissent utiliser le serveur avec le même niveau de sécurité que ceux hashant les mots de passe. Le hashage du mot de passe eest fait au niveau de la couche "réseau", c'est-à-dire dans les servlets. Ainsi le *controller* manipule et stocke sur la BDD les mots de passe déjà hashés.

Enfin, ça ne fait pas partie de la catégorie Sécurité à proprement parler, nous avons rendu le serveur robuste aux exceptions et erreurs Java avec notre système de gestion des erreurs :

- Les **erreurs métier** sont représentées par la classe CustomException. Elles se produisent lorsque le bon déroulement d'un service ne peut pas avoir lieu, à cause des inputs (client, CDiscount, BDD) vers le serveur, ou à cause d'un conflit au niveau des données métier.. Cela comprend :
  - les inputs invalides du client (mot de passe vide)
  - les inputs valides mais incorrects (conflits des données métier : par exemple, mot de passe et confirmation différents, inscription avec un username déjà

utilisé, envoi d'une réponse alors qu'aucune question n'est en cours dans la session).

  - les retours de l'API Products CDiscount (pas de produits retournés dans le JSON, indisponibilité de l'API)
  - les erreurs de la BDD MongoDB (impossibilité de mettre à jour le score)

- les **erreurs système**, c'est-à-dire toutes les erreurs au Runtime non prévues qui lèvent une Exception java.

Notre approche a été de laisser les CustomExceptions et Exceptions système remonter à travers notre application, jusqu'à la couche "réseau", où la servlet catches l'exception et produit un JSON indiquant l'échec du service et sa cause.

Les CustomExceptions ont des messages personnalisés, et "user friendly" (par exemple, la connexion avec un username invalide renvoie un JSON contenant "error": "username: l'utilisateur [test] n'existe pas.").

Les Exceptions système ont leur véritable message d'erreur dans le JSON de retour ("error": "exception: xxxxx"). Une précaution du monde réel serait de masquer les messages provenant des Exceptions système par un message d'erreur 500.

2.3 Base de données

Nous avons opté pour le **NoSQL** et utilisons **MongoDB** comme système de gestion de base de données. Celle-ci nous permet une gestion simple et intuitive de notre contenu aussi bien pour les utilisateurs que les données de jeu.
Ce choix est justifié par le fait que les données utilisées n'ont pour la plupart pas de liens relationnels forts entre elles. Les seuls liens relationnels sont à l'heure actuelle les listes des ids des amis d'un utilisateur, et les ids des produits composant une question. Comparativement au classique MySQL nous utilisons des *collections* et des *documents* en lieu et places des *tables* et *lignes* dans cette base de données non relationnelle.
Un autre avantage technique est de pouvoir gérer sa base de données avec des requêtes utilisant le format **JSON**.

Des indexs ont été créés sur les collections MongoDB :

- Collection **"Users"**: index unique sur le champ username. Cela évite d'avoir des comptes utilisateurs diffférents possédant le même username.
- Collection **"Products"** : index unique sur le champ pid. Le pid est un identifiant unique retourné par CDiscount pour chaque produit. Cela permet d'éviter de stocker 2 fois le même produit (avec les mêmes informations) dans notre BDD.
- Collection **"Questions"** : index unique sur le champ qid. Ce champ permet de relier la réponse d'un utilisateur à la question à la quelle il était en train de répondre, dont le qid est stocké dans sa session.
Ce champ qid est généré en incrémentant un compteur à chaque nouvelle question stockée. Pour que le serveur soit robuste aux pannes, ce compteur est initialisé au qid le plus haut dans la BDD à chaque redémarrage du serveur.

Schéma de la base de données

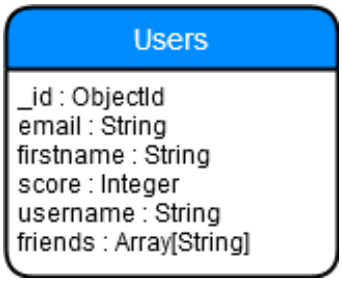

Figure 1. Collection Users

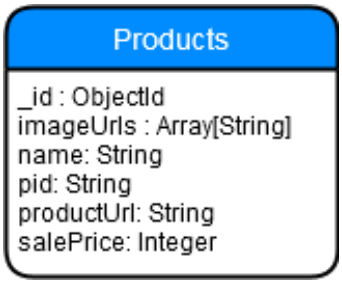

Figure 2. Collection Products

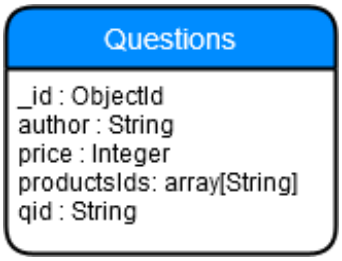

Figure 3. Collection Questions

3. Manuel d'utilisation

Pour accéder à l'application web du Juste Prix, se rendre à l'adresse `http://bit.ly/DAR_2018`

3.1 Interface

Lors de l'arrivée sur le site, l'utilisateur a accès à plusieurs ressources comme le classement des meilleurs joueurs ou l'accès à son espace personnel.
L'application est principalement composée des écrans d'accueil, de jeu, de profil et des utilisateurs.
Les actions sont réunies dans la barre de menu, accessible depuis toutes les vues de l'application.

3.2 Fonctionnalités

3.2.1 Inscription / Connexion

L'utilisateur peut s'inscrire s'il ne l'est pas encore en cliquant sur **Inscription** dans le menu déroulant **Plus** et en renseignant les champs du formulaire. De même, il peut se connecter à l'aide de ses identifiants en sélectionnant l'action **Connexion** dans le même menu déroulant.

3.2.2 Déconnexion

L'utilisateur connecté peut mettre fin à sa session en cours en cliquant sur l'action **Déconnexion** dans le menu déroulant **Plus**

3.2.3 Espace personnel

En se rendant sur sa page personnelle (action **Profil**) le joueur peut visualiser les informations qu'il a renseignées, son score ainsi que ses quizzs. Malheureusement les fonctions de modification et les quizzs n'ont pas encore été implémentés.

3.2.4 Page utilisateur

De manière assez similaire, le joueur peut visiter la page des autres utilisateurs en ayant toutefois un accès réduit aux informations, le prénom, nom et adresse email n'apparaissant pas publiquement.

3.2.5 Recherche

En entrant des termes dans la barre de recherche située dans le même espace que le menu, l'utilisateur peut rechercher d'autres membres.

3.2.6 Amitiés

Lorsqu'un utilisateur se trouve sur la page personnelle d'un autre joueur, il pourra selon le contexte, soit ajouter celui-ci à sa liste d'amis, soit l'en retirer.

3.3 Liste d'amis

En cliquant sur l'action **Amis** dans le menu *Plus*, le joueur accède à sa liste d'amis.

3.3.1 Jeu

Le joueur peut initialiser une partie depuis la page principale ou depuis l'action **Solo** depuis la barre de menu.

3.4 Leaderboard

Le top 10 des meilleurs joueurs et leurs scores sont affichés publiquement. L'utilisateur y accède avec l'action **Leaderboards**.

3.5 Use Cases

- Prenons un cas d'utilisation intéressant : une personne n'étant jamais venue sur le site décide de jouer sa première partie :
  1. La personne se rend sur `http://bit.ly/DAR_2018`
  2. Elle clique sur l'action *Inscription*
  3. Elle rempli les champs du formulaire puis valide
  4. Si les champs sont corrects, étape 5, sinon étape 3
  5. Elle clique sur l'action *Connexion*
  6. Elle rempli les champs du formulaire puis valide
  7. Si les champs sont corrects, étape 8, sinon étape 6
  8. Elle clique sur l'action *Solo*
  9. Elle clique sur l'action *Jouer*
  10. Elle donne une réponse à la question posée (repétable)
  11. Elle s'arrête et ferme la page

  Lorsque la personne reviendra jouer, son score aura été enregistré et elle n'aura plus qu'à se connecter pour continuer de jouer et gagner des points.
- Un autre cas intéressant est celui du joueur décidant de trouver un autre utilisateur :
  1. L'utilisateur entre des termes dans la *barre de recherche*
  2. Des *résultats* lui son proposés
  3. Il clique sur un utilisateur et accède à sa *page*
  4. Il ajoute l'utilisateur à son *réseau*
  5. Il pourra y accéder facilement depuis sa liste d'amis et comparer leurs scores

4. Sécurité

Afin d'obtenir un produit plus sûr nous avons mis en place quelques bases de sécurité :

- Toutes les informations sensibles transitant sur le réseau sont hashées à l'aide du protocole **SHA256**. Cette méthode de hashage plus solide que le Digest Message 5 à l'avantage d'être à la fois **sécurisante** et **légère** (moins de temps de calcul que SHA512). Aussi, l'opération de décodage étant réalisée côté client, même le serveur ne sera jamais en connaissance des identifiants des joueurs.

- Nous appliquons le principe ”do not trust the user” et vérifions ses saisies lorsqu'elles pourraient porter atteinte à l'intégrité de l'application (**injections SQL** ou **failles XSS**).
- Plus simplement, des contrôles de session / privilèges sont réalisés sur des pages spécifiques au contenu à l'accès restreint.

5. Tests

Afin de vérifier que l'application avait été implémentée avec le comportement attendu, nous avons procédé à une série de tests validant les différentes fonctionnalités du cahier des charges.

5.1 Jeux de tests

5.1.1 Base

- **Inscription**
  Tests avec aucune / mauvaises / bonnes valeurs
  Couverture : 100 %
- **Connexion**
  Tests avec aucune / mauvaises / bonnes valeurs
  Couverture : 100 %
- **Déconnexion**
  Test de fonctionnalité
  Couverture : 100 %
- **Amitiés**
  Test de fonctionnalité
  Couverture : 100 %
- **Recherche**
  Test de fonctionnalité
  Couverture : 100 %

5.1.2 Sécurité

- **Restriction**
  Tests d'accès aux pages nécessitant une session active/privilèges
  Couverture : 100 %
- **Injections SQL**
  Tests de passage de requêtes dans les formulaires Couverture : 100 %
- **Faille XSS**
  Tests de passage de scripts dans les formulaires Couverture : 100 %

5.1.3 Jeu

- **Récupération**
  Tests d'accès aux questions avec une bonne/mauvaise réponse du serveur
  Couverture : $\sim$50 %
- **Réponse**
  Tests de réponses correctes/partiels/mauvais formatage
  Couverture : $\sim$50 %

5.2 Couverture

Nous avons testé 100% des fonctionnalités identifiées dans nos use-cases exceptés celles de la partie jeu. Soit une couverture de 90% des cas d'utilisations.

6. Points forts / Points faibles

Dans cette section nous aborderons les fonctionnalités et technologies clés de cette application mais aussi quelles en sont les limitations et les axes d'améliorations envisageable. Nous pouvons dégager quelques points forts :

- Technologies adaptées
- Ajout simple de fonctionnalités
- Application divertissante
- Gameplay intuitif
- Business plan développé (chapitre suivant)
- Design

Toutefois nous aurions aimé pousser le projet plus loin en intégrant des fonctionnalités enrichissantes telles que :

- Plus de personnalisation
- Plus d'interactions entre les joueurs
- Un système de messagerie / chat global
- Des catégories thématiques de quizzs

7. Business Plan

Nous avons souhaité savoir si notre application serait commercialisable. A cette fin nous avons réalisé une étude de marché afin de savoir si des bénéfices pourraient être dégagés et quels seraient les contraintes juridiques ou administratives à prendre en compte. Afin de maximiser les parts de marché, une version mobile du jeu devra être développée et commercialisée sur les plateformes types Apple Store et GooglePlay.

7.1 Business Model

Nous souhaiterions conserver l'usage de l'application gratuit mais tout en proposant des options payantes aux joueurs.
S'impose alors le modèle Freemium : utilisation gratuite mais avec un système de micro-paiements permettant au joueur d'acheter de la monnaie virtuelle du jeu donnant accès à certaines fonctionnalités. Il peut donc obtenir plus facilement des bonus, des personnalisations graphiques et du contenu supplémentaire.
Quelques exemple d'options (non implémentées) qui pourraient être vendues :

- Accès à des ressources de personnalisation du profil
- Jokers utilisables durant les quizzs
- Multiplicateur de score

7.2 Stratégie et Marketing

Le contenu du jeu étant à base de devinettes sur le prix d'un produit, nous ciblons les personnes possédant un pouvoir d'achat ainsi qu'une habitude de la vente e-commerce : nous identifions ainsi la tranche des 18-35 ans comme clients les plus judicieux.

7.2.1 Revenus de l'application

Le Juste Prix sera intégralement jouable gratuitement et des options payantes non obligatoires seront disponibles aux joueurs afin de débloquer du contenu et personnaliser leurs expériences de jeu.

Ces options payantes reposent sur le principe d'un porte-monnaie virtuel que l'on peut alimenter avec des payements allant de 1 à 20 euros. Ce porte-monnaie sera implémenté d'une manière à ce que le joueur n'est pas l'impression de dépenser de l'argent réel (par exemple avec l'implémentation d'une devise artificielle, type pièces d'or, joyaux..) et obtienne un fort sentiment de richesse en jeu (ratio très élevé de conversion euro $\rightarrow$ devise). Les options que l'on pourra acheter seront dans les prix du marché des jeux mobiles, de l'ordre de 5-10 centimes par option.

Le porte-monnaie virtuel pourra également être alimenté en jouant simplement sans dépenser d'argent, ce qui permettra au joueur ne désirant pas investir de profiter des mêmes avantages que les joueurs payants, même si la récolte de cette monnaie sera plus lente et demandera plus d'investissement.

D'autre part, il est envisageable de mettre en place un partenariat avec CDiscount, en créant des liens entre les produits servant de contenu du jeu et ceux étant en vente sur leur site. La rémunération pourra se faire de manière forfaitaire (au nombre de redirection) ou au pourcentage des achats effectués sur CDiscount en provenance de notre application.

7.3 Juridique

Afin d'obtenir la propriété intellectuelle de l'application nous souhaiterions déposer un copyright au nom d'un société qui serait créée dans ce but.
En ce sens nous devrions alors changer le nom de l'application, afin de ne pas avoir à payer la licence du célèbre et non moins fameux jeu télévisé *Le Juste Prix* ©
La forme juridique la mieux adaptée serait la SAS : "Société Anonyme Simplifiée", c'est une structure très souple à la fois gage de sérieux et évolutive pour accueillir notamment de nouveaux investisseurs.
Avantages :

- Il n'y a pas de capital social minimum
- Les associés sont libres de faire des apports en numéraire, en nature ou en industrie sous certaines conditions
- Les offres de titres financiers s'adressant à des investisseurs qualifiés ou des sociétés de gestion
- La responsabilité des associés est limitée à leurs apports
- Les associés déterminent librement dans les statuts et les règles d'organisation de la société
- Tant que le total du bilan n'est pas supérieur à 1 million d'euros, la désignation d'un commissaire aux comptes n'est pas obligatoire

(Source : www.apce.com)

De plus, un pacte d'actionnaire régissant les règles et relations entre les différents associés protégerait la société d'une mésentente entre ses actionnaires. De même la rédaction des statuts prévoira des clauses pour protéger les membres fondateurs et leur assurer une plus grande assurance dans le contrôle de la société, notamment si celle-ci a du succès et que d'autres investisseurs rentre dans son capitale.

7.4 Fiscalité

Sur le plan fiscal nous pourrions obtenir le statut Jeune Entreprise Universitaire (JEU). Il faut pour cela que :

- La société soit détenue au moins à 50% par des personnes physiques et au moins à 10% par des étudiants
- L'activité principale de ce genre de société doit être la valorisation de travaux de recherche auxquels les dirigeants ont participé, par travaux de recherche sont sous-entendus : " les travaux de création mis en œuvre en vue d'accroître la somme des connaissances, ainsi que l'utilisation de ces connaissances pour de nouvelles applications "

Ce statut nous permet de nombreux avantages fiscaux:

- Exonération totale de l'impôt sur les sociétés pendant le premier exercice ou la première période d'imposition bénéficiaire, puis 50% pour la deuxième.
- Exonération de la contribution économique territoriale (CET) et de la taxe foncière pendant 7 ans au maximum sous délibération des collectivités territoriales
- Exonération des cotisations sociales patronales des salariés chercheurs, techniciens, gestionnaires de projet, juriste et mandataires sociaux (président et dirigeants de SAS) pendant 8 ans au maximum.

(Source : www.apce.com)

7.5 Environnement Macroéconomique

7.5.1 Analyse PESTEL

Les différents facteurs macroéconomiques influant sur notre activité sont majoritairement positifs. En effet, de nombreuses initiatives visent à favoriser l'entreprenariat dans le numérique car c'est un secteur clé de la croissance économique. De plus, le marché du mobile, sur lequel nous nous situerons avec un portage de l'application, est en plein essor avec la démocratisation des smartphones dans le monde.
Les facteurs politiques :

- Politiques de soutien à la création d'entreprise particulièrement dans les nouvelles technologies
- Politiques de soutien à l'innovation étudiante
- Assouplissements fiscaux pour ce type d'entreprises
- Simplification de la fiscalité
- Encadrement apporté aux entrepreneurs par diverses institutions notamment la CCI

Les facteurs économiques :

- Dynamisme de l'économie numérique
- De plus en plus de circuits de financement type crowdfunding
- Le jeu vidéo ne subit pas la crise économique

Les facteurs socioculturels :

- Boom du nombre d'utilisateurs de smartphones et d'applications web dans le monde
- Le smartphones multiplie les occasions de jouer à des jeux vidéo
- Démocratisation du marché du jeu vidéo pour des âges de plus en plus larges et notamment les femmes
- Avènement du modèle freemium plébiscité par les joueurs
- Reconnaissance internationale des éditeurs de jeux français

Les facteurs technologiques :

- Possibilité d'utiliser les capacités tactile des smartphones dans le gameplay des jeux
- Démocratisation des smartphones low-cost

Les facteurs environnementaux :

- Pas de distribution physique du jeu, tout est numérique.

Les facteurs législatifs :

- Tolérance vis-vis des start-up employant majoritairement des stagiaires

Toutefois, plusieurs inconvénients ne sont pas à écarter:

- Système juridique complexe, que ce soit au niveau fiscal, au niveau du code du travail
- Difficulté d'employer quelqu'un et d'offrir toutes les garanties demandées par le code du travail lorsque l'activité débute et est soumises à de nombreux risques
- Protection de la propriété intellectuelle dans le secteur du numérique difficile
- Variété des plateformes mobiles
- Très forte concurrence
- Beaucoup de plagiat

7.5.2 Analyse SWOT

Forces :

- Nous sommes localisés à Paris, lieu numéro 1 de la création numérique en France
- Bonne connaissance du marché cible
- Grande autonomie et puissance technique de par notre formation

Faiblesses :

- Manque d'expérience des associés dans l'entrepreneuriat
- Dépendance vis-à-vis de financements externes

Opportunités :

- Boom du jeu mobile & en ligne
- Faible coût de mise sur le marché

Menaces :

- De nombreux plagiats possibles
- Très forte concurrence

7.6 Finances

Nous avons simulé les besoins financiers nécessaires pour démarrer l'activité en considérant que nous n'obtenions aucune aide extérieure et avons calculé des résultats prévisionnels sur la première année fiscale.

7.6.1 Financement initial

Notre équipe possède l'ensemble des compétences technique pour lancer cette activité. Nous avons donc décidé de centrer au maximum notre budget pour toutes les dépenses inhérentes à la création de l'entreprise et les frais de distribution. Avec un coup de 200€ pour l'obtention de la licence développeur iOS, une location de serveur robuste pour 70€/mois et une accès premium à l'API CDiscount (sans limite d'appels à l'API), nous envisageons un apport personnel de 1100€ de la part de chacun des membres de l'entreprise soit 2200€.

7.6.2 Résultats prévisionnels

Afin de faire ces prévisions financières nous avons considéré que seulement 5% des joueurs du Juste Prix sont des joueurs payant, avec un panier moyen de 1,20 € par mois *(Source : www.braze.com)*. Ces micro-paiements sont commissionnés à hauteur de 10% par la plateforme, puis soumis à 20% de TVA. Nous avons aussi émis l'hypothèse que la communauté des joueurs atteindrait 110000 personnes.
Nous calculons ainsi les bénéfices :
$((((110000 * 0.05) * 1.20) * 12) * 0.90) * 0.80) = 57024 - 2200 = 54824$€

8. Conclusion

En cette fin de projet, nous avons mis à profit nos connaissances en développement d'applications au travers des langages `Web`, de `Java` et des technologies telles que les `Servlets` et les requêtes `XHR`.
De plus nous avons approfondi notre maîtrise de l'analyse de marché, de gestion de coûts et d'expérience utilisateur.
Si nous regrettons cependant l'absence de certaines fonctionnalités qui nous paraissent interéssantes, nous demeurons satisfaits du travail réalisé.

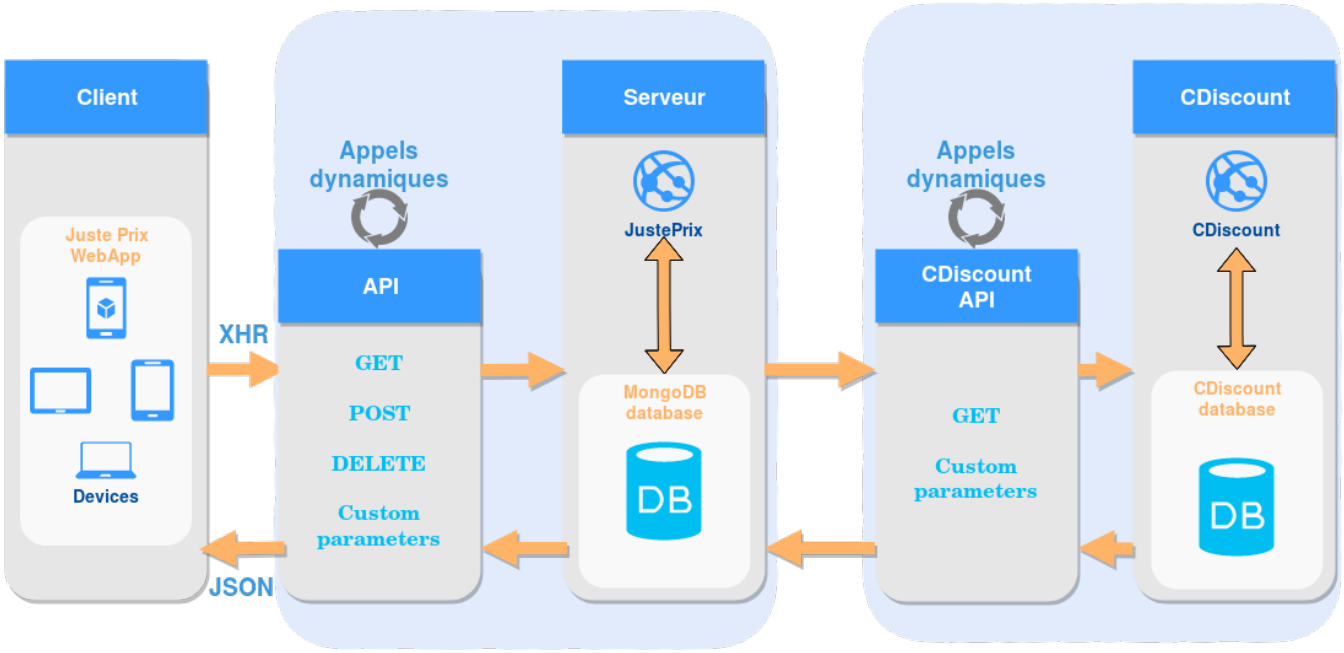

Figure 4. Schéma application Juste Prix