# Énonciation du problème et des exigences

L’application web que nous avons décidé d’implémenter est un magasin en-ligne de bière. L’application doit permettre de naviguer parmi des bières et des fournisseurs. Une personne qui veut utiliser l’application pour passer une commande devra avoir un compte. Une personne connectée à son compte pourra se créer un panier de commande et se faire livrer (théoriquement) sa commande. Une personne pourra rechercher des fournisseurs ou des bières, en plus de filtrer ses recherches pour obtenir des résultats plus précis selon ses préférences. Les utilisateurs vont aussi pouvoir laisser des commentaires sur les bières.

Le contexte de notre application est décrit par la liste suivante

* Une bière doit posséder une fiche descriptive détaillée afin de satisfaire une clientèle plus informée. Une bière peut fournir les informations : nom, pourcentage d’alcool, volume, provenance, odeur, goût, fournisseur, disponibilité et type.
  + Une annonce devra fournir obligatoirement le nom, le pourcentage d’alcool, la disponibilité, le fournisseur, un prix et le volume. Les autres informations sont optionnelles.
* Une bière peut avoir un rabais. Un rabais peut être un montant fixe, un pourcentage ou un montant calculé à partir des points de fidélité du client.
* Un fournisseur doit posséder un nom, une adresse, un numéro de téléphone, toutes les bières qu’il offre ainsi que tous les moyens de communication possibles. Un fournisseur peut offrir plusieurs types de bière sur l’application
* Un client possède un compte, qui contient son nom, son adresse à domicile, son adresse électronique, son âge, méthode de paiement, adresse de facturation et son historique d’achat
* Un client peut avoir une carte fidélité ou non. La carte permet d’accumuler des points qui permet d’obtenir des rabais sur des commandes
* Les clients vont pouvoir se construire un panier avec des bières qu’ils ont sélectionnées. Le client peut choisir la quantité de bières qu’il veut ajouter à son panier. Le panier présente aussi les sous totaux pour chacune des bières

# Spécifications du système et des responsabilités des trois niveaux

Les fonctionnalités de notre application seront les suivantes :

* Création d’un compte utilisateur
* Connexion à son compte
* Modifier les informations de son compte
* Un client peut consulter son historique d’achat
* Rechercher des bières
* Rechercher des fournisseurs
* Filtrer le résultat de recherche
* Afficher la page d’une bière
* Afficher la page d’un fournisseur
* Choisir la quantité d’une bière à commander
* Ajouter une bière dans son panier de commande
* Retirer un élément de son panier
* Modifier la quantité de son panier
* Passer la commande de son panier

**Gestion des données**

La couche de données est implémentée en utilisant mySQL. Cette couche permet de stocker toutes les données de l’application : les données sur les bières, sur les fournisseurs, les paniers, les comptes des clients, etc.

Principalement, la base de données ne répond qu’à des requêtes de la couche de logique. Par exemple, si la couche de logique demande à obtenir l’information sur une bière précise, la base de données va seulement faire un SELECT pour obtenir la bière correspondante et la retourner à la couche de logique. Ceci consistera au rôle principal de la base de données, soit de faire des SELECT correspondant à la demande de la couche de logique pour fournir de l’information.

Un autre rôle important sera le stockage du panier d’achat d’un utilisateur. De base, chaque client va posséder un panier. La base de données va modifier le panier suite à des requêtes de la couche logique en ajoutant des entrées d’une relation à la relation Panier à l’aide d’INSERT.

**Logique de l’application**

La couche logique de notre application est un API REST et est développé en utilisant Flask en Python. Cette couche reçoit les requêtes émises à partir des actions de l’utilisateur dans la couche interface. La logique de l’application ne fait que d’intercepter la requête, aller chercher l’information demandée dans la couche gestion des données, convertie les données dans le format requis et envoie l’information transformé vers la couche utilisateur pour être affichée à l’utilisateur.

Cette couche va recevoir des requêtes de l’utilisateur sous la forme d’appels http. Un GET sur une ressource va permettre d’obtenir l’information. Par exemple, un utilisateur clique sur une bière précise pour obtenir l’information sur cette bière. Le clique sur la bière déclenche un appel GET sur la ressource et la couche logique intercepte cet appel. La couche détermine l’ID de la bière à aller chercher dans la couche de donnée, fait la requête à la BD, reçoit l’information nécessaire et envoie l’information vers la couche interface.

La couche logique va permettre aussi de gérer toutes les requêtes de type POST/PUT, ce qui va permettre de créer ou de modifier une ressource. Par exemple, lorsqu’un utilisateur va vouloir ajouter un objet dans son panier, la couche utilisateur va envoyer un requête http que la couche logique va intercepter. Cette dernière va alors exécuter la fonction nécessairement pour ajouter l’objet au panier et va donc modifier le panier dans la base de données en ajoutant une entrée dans le panier.

**Interface de l’utilisateur**

La couche interface est la couche avec laquelle interagit l’utilisateur. Son rôle est d’afficher les données à l’utilisateur de l’application et de faire le lien entre les demandes de l’utilisateur et la couche inférieure, soit la couche de logique. La couche interface de notre application a été développé en utilisant Vue.js et le gestionnaire de paquet NPM.

La couche de notre interface se divise en plusieurs « components » et « views ». Par exemple, nous avons une « view » qui permet d’afficher toutes les bières correspondantes à la recherche de l’utilisateur et chacune des bières affichées est un « component ». L’interface permet de capter les actions de l’utilisateur et de le rediriger vers les bonnes « views ».

Pour communiquer avec la couche logique, on utilise des requêtes AJAX. Les réponses reçues sont dans le format JSON.

# Modélisation des données et de la BD

**Modèle entité-relation**

**A close up of a map

Description automatically generated**

**Modèle relationnel**

Customers (

id :  integer,

name : varchar(100),

phone : varchar(20),

email: varchar(100),

username : varchar(40),

login\_password : varchar(40),

address\_line\_1 : vachar(100),

address\_line\_2 : varchar(100),

city : varchar(20),

country : varchar(20)

)

* Cette relation contient les informations sur les utilisateurs de l’application

Orders (

order\_id : integer,

customer\_id : integer,

order\_date : Date,

status : vachar(10),

total\_price : Decimal,

comment : vachar(100)

)

* order\_id est la clé primaire de Orders
* customer\_id est une clé étrangère en référence à Customers

OrderItems (

item\_id : integer,

order\_id : integer,

beer\_id : integer,

quantity : integer

)

* item\_id est la clé primaire de OrderItems
* order\_id est une clé étrangère en référence à Panier
* beer\_id est une clé étrangère en référence à Bière

Beers (

beer\_id : integer,

brand\_id : integer,

beer\_name : varchar(20),

abv : Decimal,

ibu : integer,

volume : integer,

style\_id : integer,

type\_id : integer,

beer\_price : Decimal,

disponibility : integer,

description : varchar(250)

)

* beer\_id est la clé primaire
* brand\_id est une clé étrangère en référence à Brands
* style\_id est une clé étrangère en référence à Styles
* type\_id est une clé étrangère en référence à Types

Brands (

brand\_id : integer,

brand\_name : varchar(40),

brand\_phone: varchar(20),

brand\_address: varchar(100),

brand\_city: varchar(40),

brand\_country: varchar(40)

)

* brand\_id est la clé primaire

Styles (

style\_id : integer,

style\_name: varchar(20)

)

* style\_id est la clé primaire

Types (

type\_id : integer,

type\_name : vachar(20)

)

* type\_id est la clé primaire