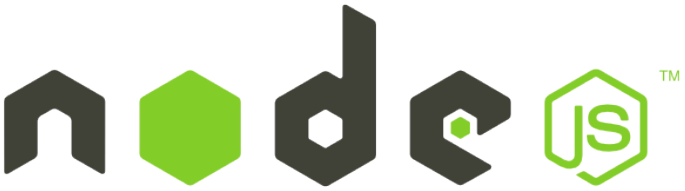

◆ ◆ ◆

Département informatique 2ème année
ENSEIRB-MATMECA

SGDB

Projet recettes de cuisine

Par : **Cyril Bos**
Clément Jacquet
Adrien Rodrigues

Client : A REMPLIR
Encadrant : A REMPLIR

Table des matières

1 Modélisation des données

1.1 Description du contexte

Spécifications client

L'objectif du projet est la création d'une base de données collaborative permettant la création, l'édition et la modification de recettes de cuisine par des internautes, dont voici un résumé des spécifications.

Une recette possède un nom, une date de création, pour combien de personnes est-elle destinée, un temps de préparation et de cuisson, ainsi que les étapes de préparation. Elles peuvent être assemblées en un menu et appartenir à une ou plusieurs catégories (ex : apéritif, entrée, plat, dessert, etc...). Comme les recettes sont collaboratives, n'importe qui peut modifier le détail de la préparation, mais il est nécessaire d'en garder un historique. À chaque modification, une date de début et de fin de la validité sont associées.

Un internaute, identifié par un pseudo, peut :

— donner une note unique mais modifiable à une recette ;

— laisser un commentaire ;

— composer un menu

Une recette utilise plusieurs ingrédients, dont les unités de quantité diffèrent (cuillère à café, à soupe, etc...). Chaque ingrédient possède plusieurs caractéristiques nutritionnelles avec une certaine valeur.

Première analyse conceptuelle

Ces spécifications nous ont conduit à imaginer les entités suivantes...

— `Recette`, comportant un nom, une date de création, un nombre de personnes, un temps de préparation et de cuisson, et le texte de préparation ;

— `Internaute`, comportant seulement un pseudo ;

— `Menu`, comportant seulement un nom ;

— `Commentaire`, comportant le texte du commentaire et la date d' ajout ;

— `Catégorie`, représentant une catégorie de recette, comportant seulement un nom ;

— `Ingrédient`, comportant uniquement un nom ;

— `Carac_Nutritionnelle`, comportant un nom et une valeur ;

— `Historique_Modif`, représentant une modification du texte de préparation d'une recette, comportant la date de la modification ainsi que le nouveau texte. L'historique de modification d'une recette s'obtient en récupérant toutes les modifications relatives à une recette (voir requêtes de consultation)

...ainsi que les associations basées sur les hypothèses expliquées ci-dessous.

— `composer`, qui associe à une recette un ingrédient, comprenant également en quelle quantité et l'unité de cette quantité. Une recette peut posséder plusieurs ingrédients, et un ingrédient peut composer plusieurs recettes ;
— `concerner`, qui associe à une recette un commentaire. Une recette peut avoir plusieurs commentaires, mais un commentaire ne concerne qu'une seule recette ;
— `appartenir_categorie`, qui associe une recette à une catégorie. Une recette peut appartenir à plusieurs catégories, et une catégorie comprend plusieurs recettes ;
— `appartenir_menu`, qui associe une recette à un menu. Une recette peut appartenir à plusieurs menus, et un menu peut contenir plusieurs recettes ;
— `archiver`, qui associe une recette et un texte issu de la modification d'une préparation. Une recette peut être modifié plusieurs fois, mais une modification n'est relative qu'à une recette ;
— `modifier`, qui associe un internaute à la modification du texte de préparation d'une recette. Un internaute peut modifier plusieurs textes de préparation, mais une modification n'est relative qu'à un seul utilisateur ;
— `noter`, qui associe un utilisateur à une recette qu'il a noté, comprenant à cet effet la valeur de cette note. Un internaute peut noter plusieurs recettes, et une recette peut être notée par plusieurs internautes ;
— `créer`, qui associe un internaute à un menu. Un internaute peut créer plusieurs menus, et un menu peut être créé et modifié par plusieurs internautes ;
— `ecrire`, qui associe un internaute à un commentaire. Un internaute peut écrire plusieurs commentaires, mais un commentaire n'est écrit que par un internaute ;
— `posseder_carac`, qui associe un ingrédient à une caractéristique nutritionnelle. Un ingrédient possède plusieurs caractéristiques naturelles, et une caractéristique naturelle peut être retrouvée dans plusieurs ingrédients

Il était demandé d'avoir une date de début et de fin de validité pour une modification d'un texte de préparation. Pour éviter de gérer la cohérence des dates de début devant être antérieures aux dates de fin à l'aide d'une contrainte d'intégrité, et surtout pour éviter de vérifier qu'une date de fin de validité correspond à une date de début de validité d'une autre modification, nous avons préféré ne pas stocker les dates de fin. Il est néanmoins possible de retrouver cette information : il faut trier toutes les modifications par date de début ; et pour une entrée donnée, sa date de fin de validité est la date de début de l'entrée suivante. Si il n'y a pas d'entrée suivante, alors il s'agit du texte actuel et il n'y a donc pas encore de date de fin de validité. Cela oblige par contre de devoir stocker le texte de préparation actuel dans l'historique ; contrepartie que nous avons jugée moins gênante.

Il faudra par ailleurs s'assurer que la date de création d'un commentaire et d'une modification d'un texte de préparation soient postérieures à la date de création d'une recette à l'aide d'une contrainte d'intégrité.

1.2 Modèle Conceptuel des Données

A partir de ces différentes informations, on peut créer le modèle conceptuel des données suivant :

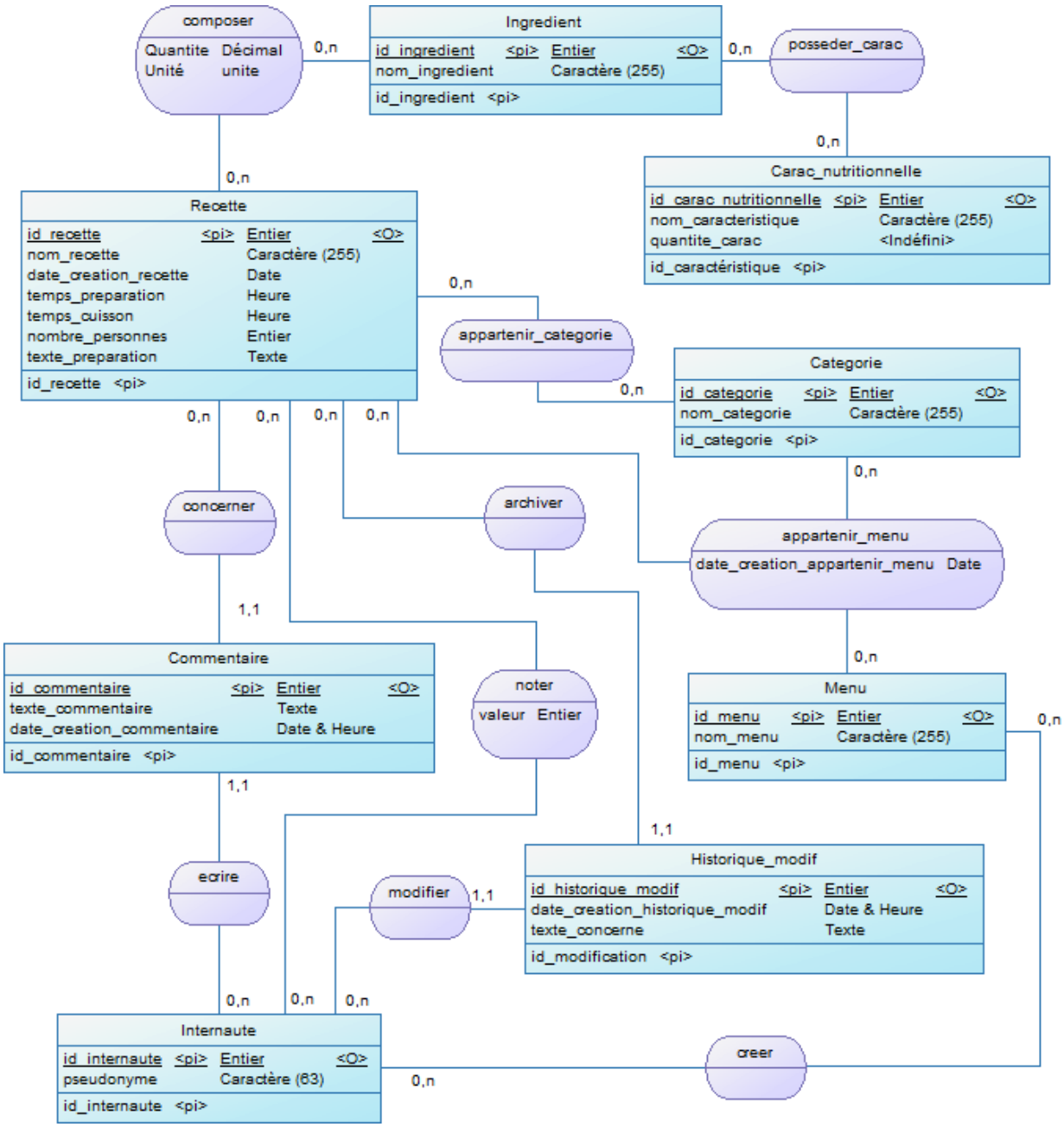

FIGURE 1 – Modèle conceptuel des données

1.3 Opérations prévues sur les données

Les opérations prévues sur les données sont :

— Les recettes peuvent être crées ou modifiées par n'importe quel internaute.

— Lors de la modification d'une recette, l'ancien texte de préparation est sauvegardé dans l'historique de modification. L'historique des modifications peut donc être obtenu en triant les différentes modifications qu'a reçues une recette dans l'ordre chronologique.

— La création de commentaires par les internautes, plusieurs commentaires peuvent être créés par le même internaute sur la même recette.

— Les internautes peuvent noter une recette et éventuellement modifier cette note ultérieurement.

— La composition (ingrédients, catégories) d'une recette peut être modifiée à tout moment par les internautes, de même pour les caractéristiques nutritionnelles des ingrédients.

— Les internautes peuvent créer pour cela des ingrédients, catégories et caractéristiques nutritionnelles mais pas les modifier ou les supprimer.

— Une recette peut être ajoutée et supprimée librement des différents menus crées par les utilisateurs. La seule contrainte est qu'une recette ajoutée dans un menu doit être ajoutée dans une des catégories dans laquelle elle apparaît.

2 Passage au Modèle Relationnel

2.1 Schéma Relationnel

Après avoir transformé le schéma conceptuel et en l'ayant optimisé, on obtient le schéma suivant :

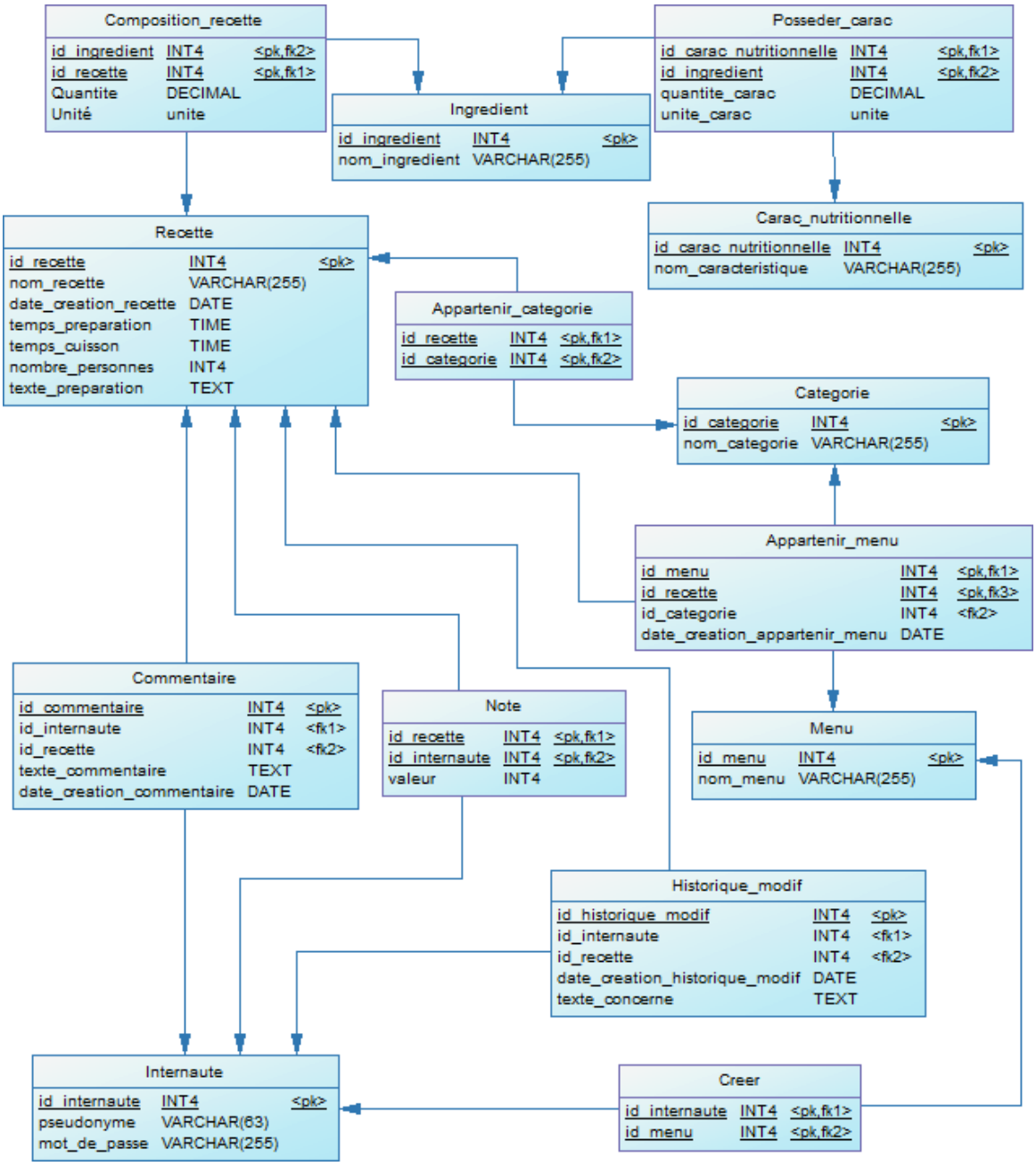

FIGURE 2 – Modèle relationnel des données

Les changements effectués sont les suivant :

— La table internaute se voit augmentée d'un champ `mot_de_passe` afin de permettre à ceux-ci de se créer un compte et de s'identifier avant de pouvoir modifier les éléments de la base de données.

— Le champ `quantite_carac` de la table `Carac_nutritionnelle` a été déplacé dans la table `posseder_carac`. Cela permet de limiter le nombre d'entrées dans la table `Carac_nutritionnelle` qui passe alors d'une taille en $\Theta(nb\_caracs \times nb\_recettes)$ à une taille $\Theta(nb\_caracs)$ dans le pire des cas sans changer celle de la table `posseder_carac`. De plus cela facilite l'écriture des requêtes de statistiques utilisant cette table.

— Le champ `quantite_carac` se voit attaché à un champ `unite_carac` qui permet de différencier les unités utilisées (kcal, g, mg ...).

— Bien que la table `appartenir_menu` possède des dépendances avec les tables `menu`, `categorie` et `recette`, sa clé primaire n'est que le couple de clés étrangères `(#id_menu, #id_recette)`. En effet la clé étrangère `#id_categorie` n'est là que pour s'assurer qu'une recette apparaissant dans un menu, n'y apparaisse qu'avec une catégorie existante dans la base. Ne pas l'avoir en clé primaire empêche une recette d'apparaître plusieurs fois dans le même menu avec des catégories différentes.

2.2 Contraintes d'intégrité et dépendances fonctionnelles

Les contraintes d'intégrité basiques telles que vérifier qu'une date est inférieure à la date présente ou qu'une note a bien une valeur entre un et trois sont directement inclues dans la définition des tables.
Pour les contraintes plus compliquées, impliquant plusieurs tables, des triggers (déclencheurs) sont utilisés. On a ainsi les triggers suivant :

— Lors de l'ajout d'un commentaire, on s'assure que la date de ce dernier est postérieure à celle de la recette concernée à celle des autres commentaires déjà présents dans la base et concernant la même recette.

— Lors de l'ajout d'une modification du texte d'une recette, on vérifie que celle-ci est postérieure à la création de la recette ainsi qu'aux autres modifications afin de s'assurer que l'historique des modifications reste cohérent.

— Lors de l'ajout d'une recette à un menu, on vérifie que la catégorie avec laquelle la recette est rajouté dans le menu est bien une des catégories de la recette.

Pour s'assurer des dépendances fonctionnelles, une contrainte d'intégrité est associée à chaque clé étrangère dans une table. Des contraintes d'intégrités sont également créées dans toutes les tables sur leur jeu de clé primaire pour que celui-ci soit unique. Enfin, des index sont créés pour l'ensemble des clés qu'elles soient étrangères ou primaires.

3 Implémentation

3.1 Choix des technologies

Nous avons choisi d'utiliser le SGBD `PosgreSQL` pour la réalisation du projet pour plusieurs raisons :

- Il est libre et *opensource* ce qui nous a permis de facilement l'utiliser sur des machines en local, contrairement à `Oracle` qui est propriétaire et payant.
- Il est plus complet que `MySQL`, particulièrement au niveau de la gestion des contraintes d'intégrité. Il permet par exemple d'ajouter des contraintes sur les tables lors de la création de la base de donnée, ce que ne permet pas `MySQL`. En effet, la documentation de MySQL spécifie : *"The CHECK clause is parsed but ignored by all storage engines"* (`http://dev.mysql.com/doc/refman/5.7/en/create-table.html`) ;
- Il est très connu et utilisé (quatrième SGBD le plus populaire) ce qui permet de facilement trouver la documentation ainsi que le support pour apprendre à correctement l'utiliser et résoudre rapidement les éventuels problèmes rencontrés

Nous avons choisi, pour réaliser l'interface d'utiliser la plateforme `Node.js`. Le choix de cette technologie est porté par les raisons suivantes :

- Il nous semblait logique de réaliser un site web avec un cahier des charges visant à créer un outil de gestion de recettes communautaire.
- `Node.js` est très populaire malgré que ce soit un logiciel récent (2009) et possède donc un très grand nombre de modules crées par la communauté qui sont facilement utilisables et permettent d'améliorer la vitesse de conception et les fonctionnalités d'un site web.
- Il est libre et open source, offrant souvent un plus grand panel de technologies compatibles
- Il permet d'exploiter des fonctions asynchrones, c'est à dire des fonctions qui lorsqu'elles sont exécutées ne bloquent pas le fil d'exécution général. C'est ce fonctionnement notamment qui permet à `Node.js` de répondre à de nombreuses requêtes sans avoir à exécuter plusieurs instances de ce dernier pour répondre aux requêtes comme fait Apache pour PHP par exemple en créant un thread par requête. De plus cela permet de facilement paralléliser plusieurs requêtes lisant la base de données pour charger une page. On peut prendre par exemple l'affichage de la page d'une recette qui nécessite cinq requêtes indépendantes toutes réalisées en parallèle afin d'accélérer le temps de rendu de la page internet.
- Enfin nous avons choisi d'utiliser `Node.js` car nous avions tous déjà programmé des sites web en PHP et cela nous a permis de découvrir d'autres moyens de développer un site web.

Le framework principal `Node.js` utilisé pour la réalisation de l'interface est `Express`. Il offre une surcouche au module *http* de `Node.js` qui donne accès à un système de routage amélioré et un système de plugins (*middleware*) qui vont traiter à la chaine les requêtes et réponses *http*. Il nous permet notamment de facilement respecter une architecture MVC. En effet celui-ci permet de créer des routeurs permettant d'associer facilement une adresse URL vers le contrôleur correspondant (par exemple l'URL /recette/ :id : donne vers la page affichant la recette ayant pour id :id : tandis que recette/add_comment/ :id : est la page ajoutant un commentaire).

Enfin, pour le site côté client il a été choisi d'utiliser le framework `bootstrap`. Celui-ci permet de facilement présenter une page internet grâce à son système de découpage en colonnes et grâce aux nombreux composants visuels qu'il intègre (barre de navigation, tableau, formulaire ...). On utilise également la bibliothèque JavaScript `jQuery` qui facilite l'écriture des interactions sur la page web coté client notamment en permettant de sélectionner des éléments de la page avec des sélecteurs CSS. Il est également pratique pour réaliser des requêtes AJAX et ainsi pouvoir réaliser des actions dynamiques sur une page sans avoir à la recharger entièrement (comme par exemple pour se connecter ou pour poster un commentaire et le voir directement).

3.2 Installation et utilisation de ces technologies

Cette partie présente comment installer et utiliser `Postgres` et `Node.js`, notamment créer la base et lancer l'application. Les instructions suivantes sont données pour une distribution Linux basée sur Debian, avec comme interpréteur shell `bash`.

3.2.1 Installation de `postgreSQL` et création de la base

À l'aide du gestionnaire de paquet (deb, aptitude), installer les paquets postgresql et postgresql-client (pour utiliser psql, un terminal interactif semblable à SQL*Plus). Ensuite, en étant loggé sous le compte root, passez sous le compte utilisateur postgres :

```
su - postgres
```

Vous pouvez maintenant accéder à la base de données "postgres" créée automatiquement (elle contient notamment les informations de toutes les autres bases, rôles, ...) :

```
psql
```

D'abord, créer une base de données du nom de "cuisine", puis exécuter le script create.sql pour créer toutes les tables, index, contraintes, ... :

```
CREATE DATABASE cuisine;
\c cuisine
```

```
\i sql/create.sql
\i sql/triggers.sql
```

Créons maintenant l'utilisateur pour l'interface web avec comme nom et mot de passe "reader", et ajoutez les privilèges à cet utilisateur sur cette base de données :

```
CREATE USER reader WITH PASSWORD 'reader';
GRANT ALL PRIVILEGES TO reader ON ALL TABLES IN SCHEMA
   'public';
GRANT ALL PRIVILEGES TO reader ON ALL SEQUENCES IN SCHEMA
   'public';
```

Attention, il se peut que le mode de connexion soit réglé sur l'authentification avec les utilisateurs Linux. Pour que l'interface fonctionne, il se peut qu'il faille modifier le fichier `/etc/postgresql/9.5/main/pg_hba.conf`. Rajoutez cette ligne à la fin du fichier (attention, on vous conseille de garder une copie originale de ce fichier en cas de problème).

```
host all all 0.0.0.0/0 md5
```

Le script testdata.sql exécuté de la même manière permet de peupler la base avec des données suffisantes pour vérifier le bon fonctionnement des requêtes. Pour détruire la base de données, il faut effectuer la commande `DROP DATABASE cuisine;` depuis la base postgres. Attention, si vous voulez le faire depuis l'utilisateur reader, il faut d'abord modifier le possesseur de la base :

```
ALTER DATABASE cuisine OWNER TO reader;
```

3.2.2 Installation de `Node.js`

À l'aide du gestionnaire de paquet ou bien en téléchargeant les dernières versions ici, installer Node.js et npm. Ensuite, naviguez dans le dossier nodejs/ du projet. Ce dossier contient le fichier package.json contenant les dépendances de l'application. Ainsi :

```
npm install #installe les dépendances présentes dans package.json
npm start #lance l'application suivant le paramètre "start" défini dans
   package.json
```

L'application peut aussi être lancée par la commande node bin/www.

3.3 Structure de la base

Le script create.sql permet de créer la base. Il commence par un drop de tous les index, types et tables si ils existent, pour permettre de modifier la structure de ces éléments plus facilement en les recréant.

Les tables correspondant à des entités (recette, menu, internaute, ...) ont un champ de clé primaire respectant comme convention de nommage `ID_NOM_TABLE`. Ces champs ont pour type `SERIAL`, qui représente un entier dont la valeur s'incrémente lors d'une insertion (semblable à la propriété `AUTO_INCREMENT` disponibles dans d'autres SGBD). Ils ont comme propriété `NOT NULL` pour garantir la cohérence des données : on ne pourra pas modifier la valeur à `NULL` après insertion d'un enregistrement. Une contrainte de clé primaire nommée `PK_NOM_TABLE` est systématiquement crée sur ce champ.

Les tables correspondant à des associations entre plusieurs entités (appartenir_menu par exemple) ont comme clé primaire un tuple d'identifiants d'autres tables. Ainsi le type utilisé pour ces champs est `INT4` (entier non signé codé sur 4 octets). Une contrainte de clé primaire nommée `PK_NOM_TABLE` est systématiquement crée sur ce tuple.

Les tables qui possèdent une ou plusieurs clés étrangères ont une contrainte portant sur le champ en question, avec les propriétés `on delete restrict` et `on update restrict`. Ainsi, un enregistrement d'une table référencé dans une autre table ne peut être supprimé que si toutes les références à cet enregistrement sont supprimées au préalable. Par exemple, une recette ne peut-être supprimée que si tous les enregistrements dans la table appartenir_menu sont supprimées. Ces contraintes permettent de garantir la cohérence des données enregistrées.

Les dates sont de type `timestamp with time zone`, permettant d'obtenir l'heure suivant différents fuseaux horaires plus facilement qu'à partir d'un timestamp classique.

Nous utilisons aussi des types personnalisés qui s'ajoutent aux types SQL standards (INT, VARCHAR, ...) :

- `"unite"` qui est une énumération des différents unités possibles d'un ingrédient (g, L, cuillère à café, ...)
- `"unite_nutrition"` qui est une énumération des unités utilisées par les caractéristiques nutritionnelles (kcal pour l'énergie par exemple)

3.4 Requêtes implémentées

Toutes les requêtes demandées ont été implémentées (version dite avancée dans le sujet). Elles ne sont pas toutes utilisées dans l'interface. Cette dernière utilise d'ailleurs souvent des requêtes préparées, contenue dans les différentes fonctions des différents modèles dans le dossier nodejs/app/models/. Néanmoins le dossier sql présente des exemples sous forme non préparée :

- queries.sql contient des requêtes de consultation
- update.sql contient des requêtes d'ajout modification et de suppression
- stats_queries.sql contient les requêtes de statistiques demandées

Les contraintes fonctionnelles sont implémentées à l'aide de triggers contenus dans triggers.sql, qui appellent des procédures stockées (dû au fonctionnement des triggers en

Postgres).

4 Utilisation de l'interface

L'application décrite ici et préchargé avec les données de tests fournis est disponible en ligne sur le site http://app.crowdpick.xyz:3000/recetator3000.
Elle respecte le modèle de conception Modèle-Vue-Contrôleur à l'aide du framework Express.js.

Une barre de navigation facilite l'accès aux différentes catégories du site. La page /404 donne l'arborescence du site en plus.
Il est possible de créer un compte et de se connecter depuis le bouton situé en haut à gauche. Cette étape est nécessaire pour modifier tout élément de la base de donnée (ajouter un commentaire, noter une recette, éditer une recette).

L'application présente un menu principal avec un carrousel de 3 recettes au hasard parmi celles de la base de données. En cliquant sur une des images de ce carrousel, on accède à la page descriptive de cette recette contenant :

- les informations directement relatives à la recette (nom, texte de préparation, moyenne des notes ...)
- une liste des ingrédients, dont le nom est un lien renvoyant vers la page descriptive d'un ingrédient
- les commentaires, avec la possibilité d'en ajouter un
- un lien vers l'historique des modifications de la recette

La page de recherche d'une recette permet de voir toutes les recettes si l'on effectue une recherche sans aucun paramètre.

La page descriptive d'un ingrédient donne ses caractéristiques nutritionnelles pour 100g, son score au classement avec un lien vers ce dernier (qui correspond à la requête de statistiques la plus complexe).

La page descriptive d'une catégorie donne les recettes appartenant à cette catégorie.

La page de l'historique d'une recette donne les anciennes versions du texte de préparation de celle-ci.

Elle a été testée majoritairement sous Google Chrome et Firefox.