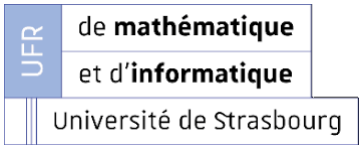
RENDU DE PROJET

« Recettes de cuisines »

BDD2

UFR Math-Info - Université de Strasbourg

3ième année de Licence Informatique



Date de rendu : 30 decembre 2021

Par

Matthieu FREITAG

https://github.com/Zapharaos/ProjetBDD2

Index

1 Introduction

On se propose de réaliser dans ce projet l'implémentation de la base de données d'un site pour gérer des recettes de cuisine dans le cadre de l'UE "Base de donnéee 2".

Pour simplifier cette étude, le sujet, disponible en annexe, nous guide de manière grossière et cite les informations que l'on souheterait sauvegarder. De ce fait, les interprétations seront multiples.

Nous sont également décrit, avec précision, les différentes requêtes, contraintes d'intégrité, procédures et fonctions à implémenter.

Ce présent rapport contient la premiere partie et la seconde partie. La premiere reste bien evidemment la même que lors du premier rendu, elle figure simplement dans ce rapport afin de tout rassembler dans un même rapport. La seconde, comme indiqué dans le sommaire, débute Page 23 et correspond à la réalisation des requêtes, contraintes d'intégrité, procédures et fonctions.

Nous avons décidé de garder la première partie de Matthieu FREITAG comme base pour cette seconde partie notamment car elle paraît plus complète.

Modèle entité-association

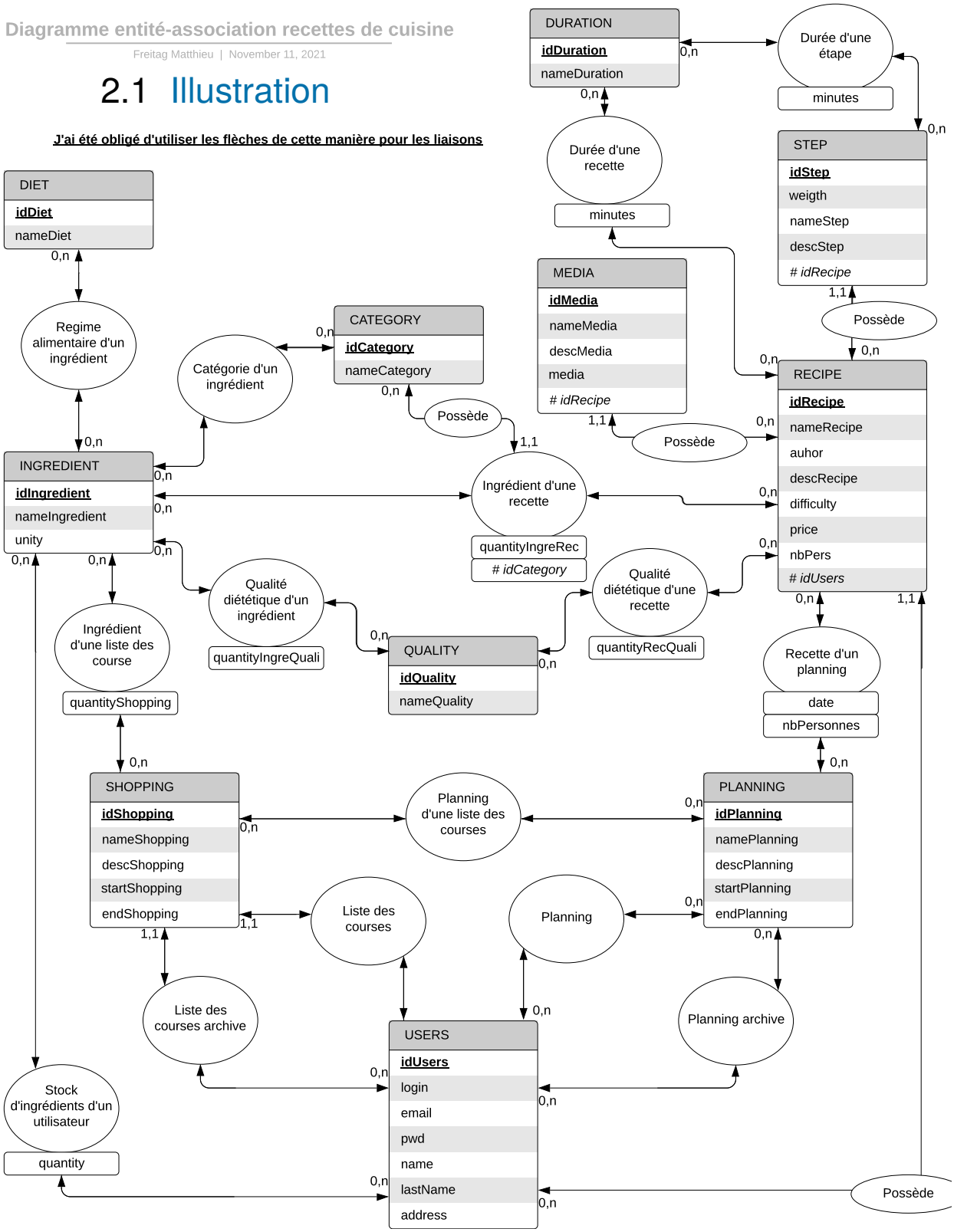
2

Diagramme entité-association recettes de cuisine

Freitag Matthieu | November 11, 2021

2.1 Illustration

J'ai été obligé d'utiliser les flèches de cette manière pour les liaisons



2.2 Implémentation et choix : tables principales

Utilisateur

La table "USERS" contient l'ensemble des informations sur les utilisateurs. Les champs "nom", "prénom" et "adresse" était uniquement demandé pour la deuxième partie, mais j'ai préféré les implémenter en amont.

- **idUsers** : CLE PRIMAIRE, elle permet d'identifier chaque utilisateur avec un numéro.
- **login** : pseudo, il permet d'identifier un utilisateur par un surnom.
- **email** : adresse e-mail, elle permet de joindre un utilisateur.
- **pwd** : mot de passe, il permet de prouver l'identité d'un utilisateur.
- **name** : prénom, il permet d'identifier un utilisateur.
- **lastName** : nom de famille, il permet d'identifier un utilisateur.
- **address** : lieu de résidence, il permet de localiser un utilisateur.

Ingredient

La table "INGREDIENT" contient l'ensemble des ingredients que peut posséder une recette.

- **idIngredient** : CLE PRIMAIRE, elle permet d'identifier chaque ingredient avec un numéro unique.
- **nameIngredient** : nom, il permet d'identifier un ingrédient.
- **unity** : unité, elle permet d'attribuer une unité à ses quantités.

Dans le cas d'une unité, on indiquera le pluriel, par example : "oeuf(s)". Comme demandé dans le sujet, la quantité d'un ingrédient peut être indiquée soit en unité, soit en gramme. De ce fait, les recettes possédant des quantités comme "3 cuillères à soupe" sont biaisées.

Afin de palier à ce problème, on pourrait ajouter une table "EQUIVALENCE", reliée aux ingrédients nécessitant d'effectuer des conversions, et qui donnerait par exemple l'équivalent d'une cuillère à café de sucre en gramme.

Ainsi, par soucis de simplicité et afin de respecter la consigne, j'ai effectué manuellement la conversion de la quantité. Pour cela, je me suis servis d'un site (disponible dans bibliographie). De plus, j'admet une densité de 1 pour chaque liquides et poudres.

Recette

La table "RECIPE" contient l'ensemble des recettes.

— **idRecipe** : CLE PRIMAIRE, elle permet d'identifier chaque recette avec un numéro unique.

— **nameRecipe** : nom, il permet d'identifier une recette.

— **author** : auteur, il permet d'indiquer l'auteur.

— **descRecipe** : description, elle apporte des détails à une recette.

— **difficulty** : difficulté de la réalisation de la recette.

— **price** : coût de la réalisation de la recette

— **nbPers** : nombre de personnes, pour les quantités d'ingrédients.

— **idUsers** : CLE ETRANGERE, elle permet de relier une recette à un utilisateur.

Planning

La table "PLANNING" contient l'ensemble des plannings.

— **idPlanning** : CLE PRIMAIRE, elle permet d'identifier chaque planning avec un numéro unique.

— **namePlanning** : nom, il permet d'identifier un planning.

— **descPlanning** : description, elle apporte des détails à un planning.

— **startPlanning** : date du premier repas dans le planning

— **endPlanning** : date du dernier repas dans le planning

Il existe deux tables qui font le lien entre un planning et un utilisateur. Les deux sont des listes de plannings, mais une seule permet d'archiver. Pourquoi ne pas avoir simplement deux tables de plannings, une d'archive et une actuelle ?

- quand un planning doit être archivé, il suffit de supprimer la liaison dans la liste des plannings actuels et de l'insérer dans les archives, ce qui évite de devoir copier-coller des plannings à chaque fois.

De plus, avec cette implémentation, plusieurs utilisateurs peuvent se partager un planning.

Liste des courses

La table "SHOPPING" contient l'ensemble des listes de courses d'après un stock et un planning.

— **idShopping** : CLE PRIMAIRE, elle permet d'identifier chaque liste avec un numéro unique.

— **nameShopping** : nom, il permet d'identifier une liste des courses.

— **descShopping** : description, elle apporte des détails à une liste.

— **startShopping** : date de début d'une liste des courses

— **endShopping** : date de fin d'une liste des courses

Il existe deux tables qui font le lien entre une liste des courses et un utilisateur. Les deux sont des listes de listes des courses, mais une seule permet d'archiver. Pourquoi ne pas avoir simplement deux tables de listes des courses, une d'archive et une actuelle ?

- quand une liste des courses doit être archivée, il suffit de supprimer la liaison dans la liste des listes des courses actuelles et de l'insérer dans les archives, ce qui évite de devoir copier-coller des listes de courses à chaque fois.

Etant donné qu'une liste des courses ne peut être reliée qu'à un seul utilisateur, la table devrait contenir une clé étrangere vers un idUsers, mais pour les raisons évoquées juste au-dessus, je ne l'ai pas fait de cette manière.

De plus, il existe une table qui permet de relier un ou plusieurs plannings à une liste de courses. Ainsi, les dates d'une liste des courses correspondent à la date de début la plus petite et à la date de fin la plus grande parmis les plannings composant cette liste.

Regime alimentaire

La table "DIET" contient l'ensemble des régimes alimentaires que peut posséder un ingrédient.

— **idDiet** : CLE PRIMAIRE, elle permet d'identifier chaque régime avec un numéro unique.

— **nameDiet** : nom, il permet d'identifier un régime alimentaire.

La table "INGREDIENT" est reliée à celle ci. Pourquoi ne pas avoir ajouté directement ces champs dans la table "INGREDIENT" ?

- il sera plus simple d'ajouter des régimes par la suite.
- si par la suite on souhaite relier cette table à la table "RECIPE".

Je n'ai pas relié cette table à la table "RECIPE" car ce n'était pas explicitement demandé dans le sujet. En effet, on souhaite seulement savoir si elle est conforme à un régime et j'estime qu'une fonction sera suffisante.

Par défaut, j'estime qu'il existe cinq régimes alimentaires : le végétarisme, le pesco-végétarisme, le véganisme, le sans-gluten et le sans-lactose. Il en existe bien plus, mais par soucis de simplicité, je n'ai considéré que ces cinq là lors de l'insertion des données. De plus, pas tout sont présent dans les ingrédients que j'ai sélectionné.

Categorie

La table "CATEGORY" contient l'ensemble des catégories que peut posséder un ingrédient.

— **idCategory** : CLE PRIMAIRE, elle permet d'identifier chaque catégorie avec un numéro unique.

— **nameCategory** : nom, il permet d'identifier une catégorie.

Il existe une table qui fait le lien avec celle-ci et la table "INGREDIENT", mais aussi une clé étrangère dans la table "RECIPE". Pourquoi ne pas avoir ajouté un champ "category" dans ces tables ?

- il sera plus simple d'ajouter des catégories par la suite.
- pour vérifier l'unicité d'une catégorie.
- pour faire un lien entre la catégorie d'un ingrédient dans une recette donnée et tous les autres ingrédients possédant cette même catégorie.
- j'estime qu'un ingrédient peut posséder plusieurs catégories, comme le camembert : c'est à la fois un produit laitier, un fromage, un fromage pâte molle et un fromage croûte fleurie. Ainsi, en fonction d'une recette il pourrait avoir une utilisation/catégorie bien différente !

Qualité diététique

La table "QUALITY" contient l'ensemble des qualités diététique que peut posséder un ingrédient ou une recette.

— **idQuality** : CLE PRIMAIRE, elle permet d'identifier chaque qualité avec un numéro unique.

— **nameQuality** : nom, il permet d'identifier une qualité.

Il existe deux tables qui font respectivement le lien entre celle-ci et "INGREDIENT", respectivement "RECIPE". Pourquoi ne pas avoir ajouté directement ces champs dans les tables "INGREDIENT" et "RECIPE" ?

- il sera plus simple d'ajouter des qualités par la suite.
- pour vérifier l'unicité d'une qualité diététique.

Par défaut, j'estime qu'il existe quatre qualités diététique : les calories, les protides, les glucides et les lipides. Il en existe bien plus, mais par soucis de simplicité, je n'ai considéré que ces quatres là lors de l'insertion des données.

Illustration

La table "MEDIA" contient l'ensemble des illustrations que possède une recette.

— **idMedia** : CLE PRIMAIRE, elle permet d'identifier chaque illustration avec un numéro unique.

— **nameMedia** : nom, il permet d'identifier une illustration.

— **descMedia** : description, elle apporte des détails à une illustration.

— **media** : url, lien vers l'illustration

— **idRecipe** : CLE ETRANGERE, elle permet de relier un media à une recette.

Etape

La table "STEP" contient l'ensemble des étapes que possède une recette.

— **idStep** : CLE PRIMAIRE, elle permet d'identifier chaque étape avec un numéro unique

— **weigth** : poids, il permet d'ordonner les étapes d'une même recette.

— **nameStep** : nom, il permet d'identifier une étape.

— **descStep** : description, elle apporte des détails à une étape.

— **idRecipe** : CLE ETRANGERE, elle permet de relier une étape à une recette.

Durée

La table "DURATION" contient l'ensemble des types de durée que peut posséder une étape ou une recette.

— **idDuration** : CLE PRIMAIRE, elle permet d'identifier chaque type de durée avec un numéro unique

— **nameDuration** : nom, il permet d'identifier un type de durée.

Il existe deux tables qui font respectivement le lien entre celle-ci et "STEP", respectivement "RECIPE". Pourquoi ne pas avoir ajouté directementces champs dans les tables "STEP" et "RECIPE" ?

- il sera plus simple d'ajouter des types de durée par la suite.
- pour vérifier l'unicité d'un type de durée.

Par défaut, j'estime qu'il existe quatre types de durée : le temps de préparation, le temps de cuisson, le temps de repos et le temps total. De plus, je pense qu'avoir ces informations à disposition peut être utile autant pour les étapes que pour les recettes.

De cette manière, quand on ajoutera ou modifiera un type de durée pour une étape, on vérifiera que le type de durée correspondant dans la recette est supérieur ou égal, sinon on le modifiera de sorte à respecter cette consigne.

2.3 Implémentation et choix : tables de liaison

Ingrédients d'un utilisateur

Cette table permet d'attribuer une liste d'ingrédients disponibles (un stock d'ingrédients) à un utilisateur.

- **idUsers** : CLE ETRANGERE, lien vers un utilisateur.
- **idIngredient** : CLE ETRANGERE, lien vers un ingrédient.
- **qtyAvailable** : quantité en fonction de l'unité.

Un utilisateur peut posséder plusieurs ingrédients. De même, un ingrédient peut appartenir à plusieurs utilisateurs.

Planning d'un utilisateur

Cette table permet de relier un utilisateur à ses plannings.

- **idUsers** : CLE ETRANGERE, lien vers un utilisateur.
- **idPlanning** : CLE ETRANGERE, lien vers un planning.

Un utilisateur peut posséder plusieurs plannings. De même, un planning peut appartenir à plusieurs utilisateurs (une famille, un groupe d'amis en vacances, une colocation,etc...).

Planning d'un utilisateur (archive)

Cette table permet de relier un utilisateur à ses plannings archivés.

- **idUsers** : CLE ETRANGERE, lien vers un utilisateur.
- **idPlanning** : CLE ETRANGERE, lien vers un planning.

Un utilisateur peut posséder plusieurs plannings archivés. De même, un planning archivé peut appartenir à plusieurs utilisateurs (une famille, un groupe d'amis en vacances, une colocation,etc...)

Liste des courses d'un utilisateur

Cette table permet de relier un utilisateur à ses listes des courses.

— **idUsers** : CLE ETRANGERE, lien vers un utilisateur.

— **idShopping** : CLE ETRANGERE, lien vers une liste des courses.

Un utilisateur peut posséder plusieurs listes des courses. Tandis qu'une liste des courses ne peut appartenir qu'à un utilisateur.

Liste des courses d'un utilisateur (archive)

Cette table permet de relier un utilisateur à ses listes des courses archivées.

— **idUsers** : CLE ETRANGERE, lien vers un utilisateur.

— **idShopping** : CLE ETRANGERE, lien vers une liste des courses.

Un utilisateur peut posséder plusieurs listes des courses archivées. Tandis qu'une liste des courses archivée ne peut être reliée qu'à un utilisateur.

Plannings d'une liste des courses

Cette table permet d'attribuer des plannings à une liste des courses.

— **idShopping** : CLE ETRANGERE, lien vers une liste des courses.

— **idPlanning** : CLE ETRANGERE, lien vers un planning.

Une liste des courses peut posséder plusieurs plannings. De même, un planning peut appartenir à plusieurs liste des courses.

Ingrédients d'une liste des courses

Cette table permet d'attribuer des ingrédients à une liste des courses.

— **idShopping** : CLE ETRANGERE, lien vers une liste des courses.

— **idIngredient** : CLE ETRANGERE, lien vers un ingrédient.

— **qtyShopping** : quantité en fonction de l'unité.

Une liste des courses peut être composée de plusieurs ingrédients. De même, un ingrédient peut appartenir à plusieurs listes des courses.

Recettes d'un planning

Cette table permet d'attribuer des recettes à un planning.

— **idPlanning** : CLE ETRANGERE, lien vers un planning.

— **idRecipe** : CLE ETRANGERE, lien vers une recette.

— **dateMeal** : date à laquelle aura lieu le repas.

Un planning peut être composé de plusieurs recettes. De même, une recette peut appartenir à plusieurs plannings.

Régimes alimentaires d'un ingrédient

Cette table permet d'attribuer des régimes alimentaires à un ingrédient.

— **idIngredient** : CLE ETRANGERE, lien vers un ingrédient.

— **idDiet** : CLE ETRANGERE, lien vers un régime alimentaire.

Un régime alimentaire peut contenir plusieurs ingrédients. De même, un ingrédient peut appartenir à plusieurs régimes alimentaires.

Catégories d'un ingrédient

Cette table permet d'attribuer des catégories à un ingrédient.

— **idIngredient** : CLE ETRANGERE, lien vers un ingrédient.

— **idCategory** : CLE ETRANGERE, lien vers une catégorie.

Une catégorie peut contenir plusieurs ingrédients. De même, un ingrédient peut appartenir à plusieurs catégories.

Qualités diététique d'un ingrédient

Cette table permet d'attribuer des qualités diététiques à un ingrédient.

— **idIngredient** : CLE ETRANGERE, lien vers un ingrédient.

— **idQuality** : CLE ETRANGERE, lien vers une qualité diététique.

— **qtyQuality** : quantité de la qualité diététique.

Un ingrédient possède plusieurs qualités diététiques. De même, une qualité diététique est reliée à plusieurs ingrédients.

Qualités diététique d'une recette

Cette table permet d'attribuer des qualités diététiques à une recette.

— **idRecipe** : CLE ETRANGERE, lien vers une recette.

— **idQuality** : CLE ETRANGERE, lien vers une qualité diététique.

— **qtyQuality** : quantité de la qualité diététique.

Une recette possède plusieurs qualités diététiques. De même, une qualité diététique est reliée à plusieurs recettes.

Ingrédients d'une recette

Cette table permet d'attribuer des ingrédients à une recette.

— **idRecipe** : CLE ETRANGERE, lien vers une recette.

— **idIngredient** : CLE ETRANGERE, lien vers un ingrédient.

— **quantity** : quantité en fonction de l'unité.

— **idCategory** : CLE ETRANGERE, lien vers une catégorie.

Une recette peut posséder plusieurs ingrédients. De même, un ingrédient peut appartenir à plusieurs recettes.

Durée d'une étape

Cette table permet d'attribuer des durées à une étape.

— **idStep** : CLE ETRANGERE, lien vers une étape.

— **idDuration** : CLE ETRANGERE, lien vers une durée.

— **duration** : durée en minutes.

Une étape peut avoir plusieurs types de durée. De même, un type de durée peut être relié à plusieurs étapes.

Durée d'une recette

Cette table permet d'attribuer des durées à une recette.

— **idRecipe** : CLE ETRANGERE, lien vers une recette.

— **idDuration** : CLE ETRANGERE, lien vers une durée.

— **duration** : durée en minutes.

Une recette peut avoir plusieurs types de durée. De même, un type de durée peut être relié à plusieurs recettes.

Contraintes d'intégrité 3

3.1 Contraintes statiques des tables principales

Utilisateur

- **idUsers** : CLE PRIMAIRE.
- **login** : unique, non vide, maximum 32 caractères.
- **email** : unique, non vide, maximum 128 caractères, de la forme "%@%.%".
- **pwd** : non vide, maximum 64 caractères (chiffré).
- **name** : maximum 32 caractères.
- **lastName** : maximum 32 caractères.
- **address** : maximum 128 caractères.

Ingredient

- **idIngredient** : CLE PRIMAIRE.
- **nameIngredient** : unique, non vide, maximum 64 caractères.
- **unity** : non vide, maximum 64 caractères.

Recette

- **idRecipe** : CLE PRIMAIRE.
- **nameRecipe** : non vide, maximum 128 caractères.
- **author** : non vide, maximum 64 caractères.
- **difficulty** : non vide, maximum 32 caractères, de la forme : 'Tres facile', 'Facile', 'Standard', 'Difficile'.
- **price** : non nul, un flottant entre 1 et 5.
- **nbPers** : non nul, strictement supérieur à 0.
- **idUsers** : CLE ETRANGERE.

Planning

— **idPlanning** : CLE PRIMAIRE.

— **namePlanning** : non vide, maximum 32 caractères.

— Pas de contraintes statiques sur les dates, elles seront insérées et vérifiées dynamiquement lors de l'ajout d'une recette au planning.

Liste des courses

— **idShopping** : CLE PRIMAIRE.

— **nameShopping** : non vide, maximum 32 caractères.

— Pas de contraintes statiques sur les dates, elles seront insérées et vérifiées dynamiquement lors de l'ajout d'un planning à la liste des courses.

Regime alimentaire

— **idDiet** : CLE PRIMAIRE.

— **nameDiet** : unique, non vide, maximum 32 caractères.

Categorie

— **idCategory** : CLE PRIMAIRE.

— **nameCategory** : unique, non vide, maximum 64 caractères.

Qualité diététique

— **idQuality** : CLE PRIMAIRE.

— **nameQuality** : unique, non vide, maximum 32 caractères.

Illustration

— **idMedia** : CLE PRIMAIRE.
— **nameMedia** : non vide, maximum 32 caractères.
— **media** : non vide, maximum 256 caractères.
— **idRecipe** : CLE ETRANGERE.

Etape

— **idStep** : CLE PRIMAIRE.
— **weigth** : non nul, strictement supérieur à 0.
— **nameStep** : maximum 32 caractères.
— **idRecipe** : CLE ETRANGERE.

Durée

— **idDuration** : CLE PRIMAIRE.
— **nameDuration** : unique, non vide, maximum 32 caractères.

3.2 Contraintes statiques des tables de liaisons

Ingrédients d'un utilisateur

— **idUsers, idIngredient** : CLE PRIMAIRE.
— **idUsers** : CLE ETRANGERE.
— **idIngredient** : CLE ETRANGERE.
— **qtyAvailable** : non nul, supérieur ou égal à 0.

Catégories d'un ingrédient

— **idIngredient, idCategory** : CLE PRIMAIRE.
— **idIngredient** : CLE ETRANGERE.
— **idCategory** : CLE ETRANGERE.

Ingrédients d'une recette

— **idRecipe, idIngredient** : CLE PRIMAIRE.
— **idRecipe** : CLE ETRANGERE.
— **idIngredient** : CLE ETRANGERE.
— **quantity** : non nul, supérieur ou égal à 0.
— **idCategory** : CLE ETRANGERE.

Durée d'une étape

— **idStep, idDuration** : CLE PRIMAIRE.
— **idStep** : CLE ETRANGERE.
— **idDuration** : CLE ETRANGERE.
— **durStep** : non nul, supérieur ou égal à 0.

Durée d'une recette

— **idRecipe, idDuration** : CLE PRIMAIRE.
— **idRecipe** : CLE ETRANGERE.
— **idDuration** : CLE ETRANGERE.
— **durRecipe** : non nul, supérieur ou égal à 0.

On modifiera dynamiquement un type de durée d'une recette si la somme de ses étapes est supérieur.

Qualités diététiques d'un ingrédient

— **idIngredient, idQuality** : CLE PRIMAIRE.
— **idIngredient** : CLE ETRANGERE.
— **idQuality** : CLE ETRANGERE.
— **qtyQuality** : non nul, supérieur ou égal à 0.

Qualités diététiques d'une recette

— **idRecipe, idQuality** : CLE PRIMAIRE.
— **idRecipe** : CLE ETRANGERE.
— **idQuality** : CLE ETRANGERE.
— **qtyQuality** : non nul, supérieur ou égal à 0.

Recettes d'un planning

— **idPlanning, idRecipe** : CLE PRIMAIRE.
— **idPlanning** : CLE ETRANGERE.
— **idRecipe** : CLE ETRANGERE.
— **dateMeal** : non nul, supérieur à la date du jour.
— **nbPersMeal** : non nul, strictement supérieur à 0.

Plannings d'un utilisateur

— **idUsers, idPlanning** : CLE PRIMAIRE.
— **idUsers** : CLE ETRANGERE.
— **idPlanning** : CLE ETRANGERE.

Plannings archivés d'un utilisateur

— **idUsers, idPlanning** : CLE PRIMAIRE.
— **idUsers** : CLE ETRANGERE.
— **idPlanning** : CLE ETRANGERE.

Ingrédients d'une liste des courses

— **idShopping, idIngredient** : CLE PRIMAIRE.
— **idShopping** : CLE ETRANGERE.
— **idIngredient** : CLE ETRANGERE.
— **qtyShopping** : non nul, supérieur ou égal à 0.

Plannings d'une liste des courses

— **idShopping, idPlanning** : CLE PRIMAIRE.
— **idShopping** : CLE ETRANGERE.
— **idPlanning** : CLE ETRANGERE.

Listes des courses d'un utilisateur

— **idUsers, idShopping** : CLE PRIMAIRE.
— **idUsers** : CLE ETRANGERE.
— **idShopping** : CLE ETRANGERE.

Listes des courses archivées d'un utilisateur

— **idUsers, idShopping** : CLE PRIMAIRE.
— **idUsers** : CLE ETRANGERE.
— **idShopping** : CLE ETRANGERE.

Régimes alimentaires d'un ingrédient

— **idIngredient, idDiet** : CLE PRIMAIRE.
— **idIngredient** : CLE ETRANGERE.
— **idDiet** : CLE ETRANGERE.

3.3 Contraintes dynamiques

Je ne mentionne pas les contraintes dynamiques précisée dans la deuxième partie du sujet.

Utilisateur

— **name** : première lettre en majuscule.
— **lastName** : nom de famille en majuscule.

Etape

— **nameStep** : si le champ n'est pas renseigné, on initialise le nom à 'Etape POIDS', POIDS étant la valeur du champ weigth dans la table.

Recettes d'un planning

— **dateMeal** : la date du repas de peut pas etre inférieure à la date d'ajout (j'ajoute une marge de cinq minutes).

Ainsi, à chaque opération sur une recette d'un planning, on modifie la date du planning correspondant et des listes de courses contenant ce planning (uniquement si la modification est nécessaire).

Planning

— **startPlanning** : date de repas, la plus petite.

— **endPlanning** : date de repas, la plus grande.

Ces dates dépendent de chaque repas inséré dans le planning. Sachant qu'un repas est inséré uniquement si sa date est supérieure à la date à laquelle il est ajouté, je ne pense pas avoir besoin d'effectuer des verifications ici.

Liste des courses

— **startShopping** : date de planning, la plus petite.

— **endShopping** : date de planning, la plus grande.

Ces dates dépendent de chaque planning inséré dans la liste des courses. Ainsi, pour des raisons évidentes, je ne pense pas non plus avoir besoin d'effectuer des verifications. De plus, un utilisateur pourrait avoir besoin de générer une liste de courses après le début ou la fin d'un planning.

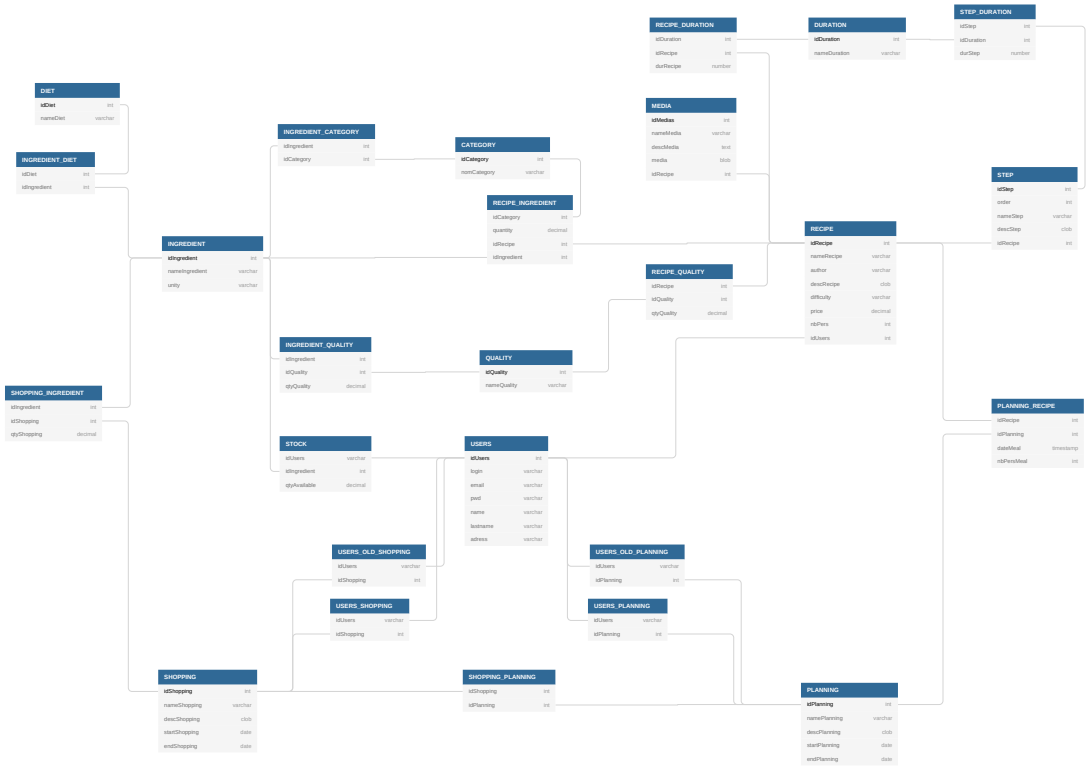
Modèle logique relationnel

4

4.1 Illustration

Je m'excuse concernant la taille de l'image. J'ai dû écarter les tables pour plus de lisibilité, mais il faut zoomer sur le modèle pour y voir quelque chose...

Si vous le souhaitez, l'image est également disponible au format PNG dans le répertoire "img".



4.2 Implémentation

Je ne précise pas les types et liaisons des clés car je l'ai déjà fait auparavant et j'estime que les noms sont assez explicite. De plus, le schéma est disponible.

Tables principales :

- USERS (**idUsers**, login, email, pwd, name, lastName, adress)
- INGREDIENT (**idIngredient**, nameIngredient, unity)
- RECIPE (**idRecipe**, nameR, auth, descR, diff, price, nbPers, #idUsers)
- PLANNING (**idPlanning**, nameP, descP, startP, endP)
- SHOPPING (**idShopping**, nameS, descS, startS, endS)
- DIET (**idDiet**, nameDiet)
- CATEGORY (**idCategory**, nameCategory)
- QUALITY (**idQuality**, nameQuality)
- MEDIA (**idMedia**, nameMedia, descMedia, media, #idRecipe)
- STEP (**idStep**, weight, nameStep, descStep, #idRecipe)
- DURATION (**idDuration**, nameDuration)

Tables de liaisons :

- STOCK (**#idUsers**, **#idIngredient**, qtyAvailable)
- USERS_PLANNING (**#idUsers**, **#idPlanning**)
- USERS_OLD_PLANNING (**#idUsers**, **#idPlanning**)
- USERS_SHOPPING (**#idUsers**, **#idShopping**)
- USERS_OLD_SHOPPING (**#idUsers**, **#idShopping**)
- SHOPPING_PLANNING (**#idShopping**, **#idPlanning**)
- SHOPPING_INGREDIENT (**#idShopping**, **#idIngredient**, qtyShop)
- PLANNING_RECIPE (**#idPlanning**, **#idRecipe**, dateMeal)
- INGREDIENT_DIET (**#idIngredient**, **#idDiet**)
- INGREDIENT_CATEGORY (**#idIngredient**, **#idCategory**)
- INGREDIENT_QUALITY (**#idIngredient**, **#idQuality**, qtyQuality)
- RECIPE_QUALITY (**#idRecipe**, **#idQuality**, qtyQuality)
- RECIPE_INGREDIENT (**#idRecipe**, **#idIngre**, quantity, #idCategory)
- RECIPE_DURATION (**#idRecipe**, **#idDuration**, duration)
- STEP_DURATION (**#idStep**, **#idDuration**, duration)

Requêtes

5

Les recettes qui ont moins de 200 calories par personne, dont tous les ingrédients sont sans gluten et qui apparaissent sur le planning d'un utilisateur

Chaque recette est réliée à la table QUALITY via RECIPE_QUALITY. Il suffit donc de joindre les tables et de sélectionner les recettes qui possèdent un nombre de calories par personne inférieur à 200.

De même, la table PLANNING est reliée à la table RECIPE via PLANNING_RECIPE. Il suffit donc de vérifier si la recette existe dans cette table. Il serait facile de vérifier l'existence pour un utilisateur en particulier.

Je n'ai pas réussi à vérifier que tous les ingrédients sont sans gluten. L'explication se site plus bas, au niveau de la quatrième recette, qui demandait plus ou moins la même chose.

La recette la plus souvent présente dans des plannings d'utilisateurs

On a la table PLANNING_RECIPE qui contient l'ensemble des recettes reliées à un planning. On va donc compter le nombre d'occurrence de chaque recette et trier les résultats dans l'ordre décroissant. On sélectionnera la première.

Le problème avec cette requête est qu'on sélectionne uniquement le premier dans la liste (conformément à l'énoncé). Il pourrait être plus judicieux de sélectionner toutes les recettes ayant le même nombre d'occurrence.

Pour chaque ingrédient, nombre de recette et nombre de catégorie dans lesquelles il apparaît

Chaque ingrédient est relié à une recette via la table RECIPE_INGREDIENT. Il suffit donc de compter le nombre d'occurrence de chaque ingrédient dans la table RECIPE_INGREDIENT (sachant qu'il ne peut pas y avoir de doublon). On procède de la même manière pour les catégories puisque la table INGREDIENT est reliée à la table CATEGORY via INGREDIENT_CATEGORY. Dans le cas où un ingrédient ne serait pas présent, on utilisera COALESCE.

Les utilisateurs qui n'ont ajouté à la base de données que des recettes végétariennes

Comme indiqué un peu plus haut, je n'ai pas réussi à faire ce type de requête. Pourtant ça n'est pas faute d'avoir essayé, j'ai passé plusieurs demi-journées à tenter de régler ce problème, j'ai cherché sur les forums et demandé à de nombreux camarades mais je n'ai pas réussi à trouver de solutions.

Je vous ai également contacté le 07/12 à ce propos, mais je n'ai pas réussi à me servir du mot-clé "ALL" et je n'ai pas réussi à faire fonctionner la double négation. Je pense avoir compris ce que vous vouliez dire, mais je n'ai pas réussi à le mettre en pratique, j'imagine que j'aurais du combiner les deux.

J'ai laissé des esquisses de requêtes en commentaire dans le fichier "requetes.sql" pour que vous puissiez tout de même voir.

Avec mon implémentation, il faudrait écrire une requête qui sélectionne les auteurs qui n'ont ajouté que des recettes qui resepectent la condition suivante : chaque ingrédient d'une recette doit respecter le régime végétarien.

Pour chaque utilisateur, son login, son nom, son prénom, son adresse, son nombre de recette créé, son nombre d'ingrédients enregistrés, le nombre de recette qu'il a prévu de réaliser (la recette est dans son planning à une date postérieure à la date d'aujourd'hui)

Le login, le nom, le prénom et l'adresse sont simples à sélectionner et se trouvent directement dans la table USERS. Pour le nombre de recette crée, chaque recette possède une clé étrangère vers la table USERS, il suffit donc de compter le nombre de fois où un utilisateur apparaît dans la table RECIPE. On a également la table STOCK qui relie USERS et INGREDIENT, il suffit encore une fois de compter le nombre d'occurrence de l'utilisateur dans cette table. De la même manière, un utilisateur est relié à un planning via deux tables : les plannings actifs et les archives. On comptera simplement le nombre d'occurrence d'un utilisateur dans la table des plannings actifs (USERS_PLANNING).

Fonctions et procédures 6

Définir une fonction qui change le nom d'un ingrédient par un autre dans les étapes de réalisation d'une recette

On récupère en argument l'ID d'une recette et deux ID d'ingrédients. On va d'abord récupérer les noms des ingrédients depuis les ID dans la table INGREDIENT. A chaque occurrence dans la table STEP, on remplacera (grace a REPLACE) le nom du premier ingréidnet par le nom du second ingrédient.

Définir une fonction qui retourne la liste des ingrédients où la quantité d'ingrédient a été adaptée pour un nombre de personnes différent du nombre de personnes de la recette d'origine

On prend en argument l'ID d'une recette et un nombre de personnes. On retournera un string qui contient la liste des ingrédients avec les nouvelles quantités si le nouveau nombre de personnes est différent du nombre de base.

Pour cela, on utilisera un curseur avec tous les ingrédients et leur quantité pour la recette donnée en argument. Pour chaque ingrédient, on l'ajoutera dans le string et on recalculera sa quantité par rapport au nouveau nombre de personnes.

Définir une fonction qui retourne un booléen si la recette ne contient que des ingrédients valides pour un certain régime (« végétarien », « sans-gluten », . . .)

On prend en paramètre l'ID d'une recette et le nom d'un régime alimentaire. On va compter le nombre d'ingrédient de la recette qui respecte le régime alimentaire et le nombre d'ingrédient total de la recette. Si les deux sont égaux, alors la recette est valide pour un certain régime.

Pour compter le nombre d'ingrédients d'une recette c'est assez simple, on compte le nombre d'ingrédients qui sont relié à l'ID de la recette donné en paramètre dans la table RECIPE_INGREDIENT. Pour compter le nombre d'ingrédients de cette même recette qui respecte un régime, il faut relier les tables RECIPE_INGREDIENT, INGREDIENT, INGREDIENT_DIET et DIET en recherchant uniquement l'ID de la recette et le nom du régime.

Définir une fonction qui génère une liste d'ingrédients à acheter. La liste d'ingrédient sera générée pour un utilisateur, à partir d'un planning de recette à réaliser, une liste d'ingrédients disponibles et la date estimée des achats

Chaque utilisateur possède une unique liste d'ingrédient disponible, un unique stock, il ne sera logiquement pas renseigné dans les paramètres de cette fonction. Pour la date des listes de courses, je prévoyais de conserver les mêmes dates que le planning correspondant, mais je change ici pour respecter la consigne.

On récupère en paramètre l'ID de l'utilisateur qui veut créer une liste de courses, l'ID du planning qui contient l'ensemble des recettes prévues et la date de fin. Il faut donc récupérer l'ensemble des ingrédients des recettes prévues avec leur quantité et comparer par rapport au stock déjà disponible avant de l'insérer dans une nouvelle liste de courses.

Pour cela, on utilisera un curseur qui contient chaque ingrédient et sa quantité prévue de chaque recette du planning. On créera ensuite une liste de courses (SHOPPING), on la reliera au planning (SHOPPING_PLANNING) et à l'utilisateur (USERS_PLANNING). Si le planning n'est pas déjà relié à l'utilisateur, on le reliera également (USERS_PLANNING).

On entre ensuite dans la boucle du curseur : on regarde si l'ingrédient est disponible dans le stock, si oui on modifie la quantité à acheter avant de l'insérer dans la liste des courses (si la quantité n'est pas nulle après modification).

Il se peut qu'un ingrédient soit prévu dans plusieurs recettes et ce cas n'est pas géré car la quantité en stock n'est pas actualisé. J'ai décidé de ne pas complexifié le code par soucis de simplicité, mais ce problème n'est pas difficile à gérer.

Définir une procédure qui crée une copie de recette où certains ingrédients ont été remplacés par d'autres équivalents et où le nombre de personnes peut-être différent de celui de la recette originale

Cette procédure prend un argument l'identifiant d'une recette, un nombre de personne (potentiellement différent du nombre de personne de la recette d'origine), l'identifiant d'un ingrédient à remplacer et l'identifiant de l'ingrédient qui le remplacera.

J'ai décidé de ne fournir qu'un seul ingrédient en argument par soucis de simplicité, mais je sais qu'il aurait également été possible de fournir une liste d'ingrédients à la place, via un string par exemple (comme dans la fonction edit_recipe() à la ligne 35 du fichier "functions.sql".

Encore une fois par soucis de simplicité, j'ai décidé de directement renseigner l'ingrédient remplaçant en paramètre, mais il aurait également été possible de sélectionner l'ensemble des ingrédients qui appartienne à la catégorie de l'ingrédient (définie dans RECIPE_INGREDIENT par la colonne "idCategory"). Mais du point de vue d'un utilisateur, ça n'est pas vraiment réaliste

selon moi et on supposera donc que l'ingrédient remplaçant est renseigné manuellement.

Il faudrait également vérifier que l'ingrédient remplaçant n'est pas déjà présent dans les ingrédients de la recette.

On se servira de cinq curseurs dans cette procédure : les ingrédients, les médias, les étapes, les durées de la recette et des étapes. Ce sont tous des éléments propres à chaque recette.

On commence donc par copier la recette et récupérer l'identifiant de cette nouvelle recette. Ensuite, on copie les ingrédients, les médias, les étapes, les durées (recette et étapes), les valeurs nutritionnelles et on remplace l'ingrédient.

7 Contraintes d'intégrité seconde partie

Pas plus de 20 ingrédients par recettes

On compte le nombre d'ingrédient dans une même recette (via idRecipe dans RECIPE_INGREDIENT). Si c'est supérieur à 20, on raise une erreur.

La liste des ingrédients à acheter ne peut pas être générée plus d'un mois à l'avance

On regarde si la date de fin d'une liste des courses est supérieure à la date actuelle + 1 mois, si oui on raise une erreur.

Dans ce cas précis, on prendra six mois au lieu d'un pour simplifier le test de la contrainte et les insert de base.

La durée d'une recette est égale au moins au minimum de la durée de ses étapes

D'abord il faut récupérer l'ID de la recette qui correspond à la durée de l'étape qu'on vient d'insérer ou de mettre à jour. Ensuite on compte le nombre d'étapes de la recette et on calcule la somme de la durée des étapes. On recupère ensuite le temps total de la recette et on compare les deux. Si le temps total est inférieur, on le met à jour.

Techniquement, il faudrait aussi ajouter un trigger pour sur RECIPE_DURATION, mais le système serait le même et c'est pour ça qu'il n'est pas réalisé ici.

Le nombre de calorie d'une recette est similaire à celui de la somme des calories de ses ingrédients (+/- 20%)

Techniquement, il faudrait également placer des triggers à l'ajout d'un ingrédient (RECIPE_INGREDIENT), à la modification des qualités de la recette (RECIPE_QUALITY) et à la modification des qualités d'un ingrédient (INGREDIENT_QUALITY). Ainsi, par soucis de simplicité, j'estime que le trigger que j'ai réalisé est suffisant et la logique reste plus ou moins la même pour les autres. De toute manière, on partira du principe qu'un utilisateur ne peut pas renseigner les qualités globales d'une recette et quelles sont simplement recalculées à l'ajout d'un ingrédient, mais aussi que le nombre de calorie d'un ingredient est stable et ne devrait pas changer (pour 100 grammes).

D'abord il faut récupérer le total des calories de la recette depuis RECIPE_QUALITY et le comparer à la somme des calories de chaque ingrédient de la recette depuis INGREDIENT_QUALITY (les calories de chaque ingrédient sont multipliées par sa quantité de l'ingrédient). On comparera ensuite les valeurs et on modifiera en fonction (si nécessaire).

Les plannings de recettes et la liste des courses sont archivés lorsqu'ils sont supprimés ou une fois les dates associées dépassées

Avant de supprimer un planning de USERS_PLANNING (respectivement une liste de courses de USERS_SHOPPING), on l'ajoutera dans USERS_OLD_PLANNING (respectivement USERS_OLD_SHOPPING).

Idéalement, j'aurais souhaité utilisé des schedules pour vérifier de manière quotidienne les dates des plannings et des listes des courses mais je n'ai pas les droits donc je vais simplement vérifier avant chaque mise à jour sur PLANNING et SHOPPING que la date de fin n'est pas inférieure à la date du jour et si oui, alors on supprime le planning ou la liste des courses de USERS_PLANNING ou USERS_SHOPPING.

Techniquement, il faudrait également vérifier avant des insertions sur RECIPE_PLANNING, SHOPPING_INGREDIENT et SHOPPING_PLANNING mais je ne vais pas le faire par soucis de simplicité et car j'estime que les triggers que j'ai realisé sont suffisant. De plus, la logique reste la même.

Index

8

Dans cette base de données, on utilisera deux index. Un pour accélerer le processus de connexion et l'autre pour accélerer la recherche des recettes qui pourra être utilisé dans différents cas. C'est index sont principalement utiles dans le cas général avec une utilisation classique.

Sinon, je n'ai pas placé d'index propre au code réalisé dans ce projet car je n'ai pas eu l'impression que ça serait vraiment utile. En effet, j'estime que les requêtes sont appliquées à de nombreuses tables différentes et créer un index à chaque table n'apporterait rien car il ne serait utilisé qu'une seule fois. En revanche, il y a certaines tables qui sont très souvent utilisées, notamment les tables de correspondance, mais comme elles possèdent très peu de colonnes, je n'ai pas non plus juger nécessaire d'y appliquer des index.

Conclusion

9

Afin d'apporter plus de contexte, je pense que relier ce type de projet à l'UE "Programmation Web" serait une bonne idée. En effet, le sujet étant assez libre, les interprétations sont nombreuses et certaines zones restent floues.

J'imagine qu'implémenter un site en complement nous permettant d'insérer/-modifier/supprimer des données nous apporterait une vision différente du projet. Ainsi, nous aurions deux projets à relier, ce qui ôterait ces zones floues et permettrait de laisser libre cours à notre imagination.

De plus on pourrait imaginer un planning hebdomadaires de suggestion, en fonction des tendances, d'un régime, etc...

Dans l'objectif d'une amélioration future, il faudrait commencer par apporter les modifications que vous, en tant que correcteur, jugerez nécessaire. Par la suite, il faudrait finir d'implémenter chaque élément du projet, comme ceux que j'ai décidé de ne pas développer complètement par soucis de simplicité.

Bibliographie

[1] laNutrition. *Qualité diététique du miel (par exemple)*. Date non mentionnée. URL : https://sante.journaldesfemmes.fr/calories/miel/aliment-31008.

[2] Le journal des femmes santé. *Qualité diététique du miel (par exemple)*. Date non mentionnée. URL : https://www.lanutrition.fr/miel.

[3] Non mentionné. *Recette : Béchamel rapide et facile*. Date non mentionnée. URL : https://www.marmiton.org/recettes/recette_bechamel-rapide-et-facile_14764.aspx.

[4] Non mentionné. *Recette : Pain d'épices*. Date non mentionnée. URL : https://www.marmiton.org/recettes/recette_pain-d-epices_11087.aspx.

[5] Océanne Alisson Przymirski. *Comment peser ses ingrédients sans balance ?* 17/04/2020. URL : https://cuisine.journaldesfemmes.fr/astuces-termes-et-tournemains/1195515-comment-peser-ses-ingredients-sans-balance/.

[6] romy_15714486. *Recette : Camembert rôti au miel*. Date non mentionnée. URL : https://www.marmiton.org/recettes/recette_camembert-roti-au-miel_45038.aspx.