RAPPORT DE PROJET WEB

RU HUNGRY

Alex AUFAUVRE

-

IG3

2016-2017

Responsables de projet : Anne LAURENT, Arnaud CASTELLTORT

Table des matières

PRESENTATION DU PROJET

RU Hungry, qu'est-ce que c'est ?

RU Hungry (« Are you hungry ? ») est une application web destinée à afficher le menu du Restaurant Universitaire (RU) en ligne ainsi que les informations en relation.

A l'heure où les nouvelles technologies nous permettent d'accéder de plus en plus facilement à l'information, le Restaurant Universitaire Triolet à proximité de la Faculté de Sciences de Montpellier propose jusqu'ici ses menus du jour uniquement sous la forme de feuilles de papier A4 imprimées et accrochées sur un grand panneau à son entrée. Le problème est donc qu'il faut se rendre sur place physiquement pour les consulter, ce qui rend l'information difficilement accessible.

Grâce à RU Hungry, il sera possible de consulter à distance les menus du jour et de la semaine de tous les restaurants qui composent le RU. Les informations concernant chaque menu (apports nutritionnels) ainsi que les tarifs seront affichés. Un système d'avis sera également mis à disposition des utilisateurs afin que les employés du RU puissent avoir des retours concernant les produits qu'ils proposent.

Cette application permettra donc une meilleure communication de l'information aussi bien pour les utilisateurs que pour le personnel du RU.

Perspectives d'évolution

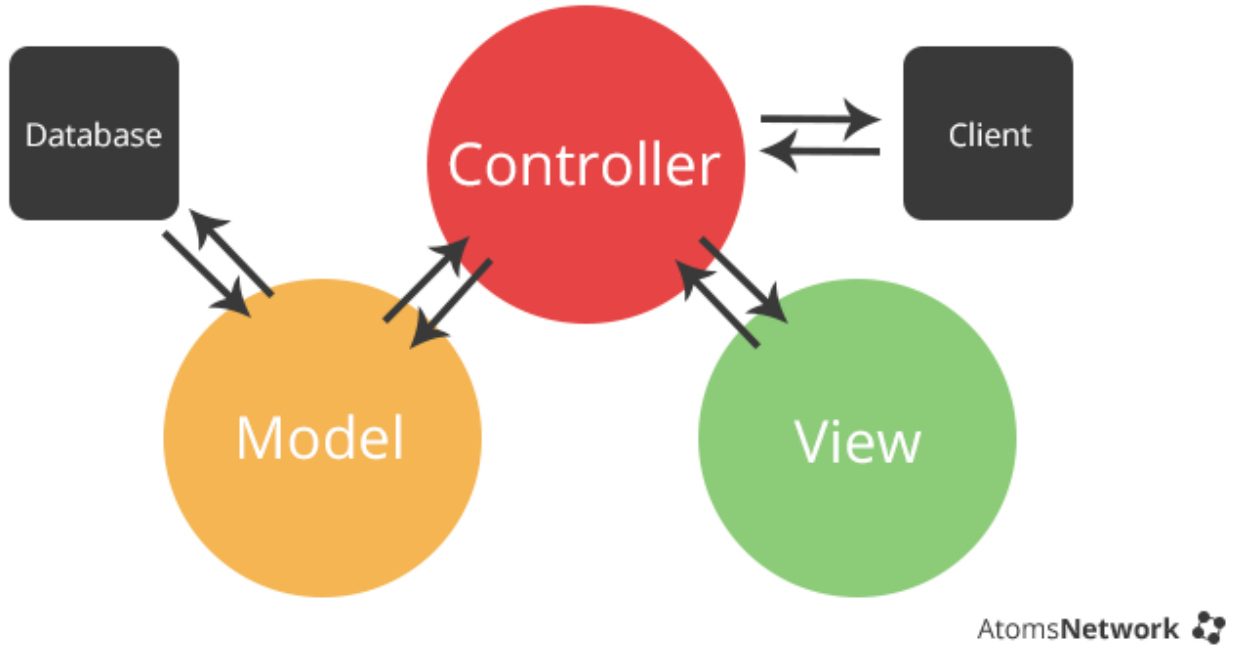
RU Hungry, initialement conçue pour répondre aux besoins du Restaurant Universitaire Triolet de Montpellier, pourra à terme cibler tous les restaurants universitaires de France.

Elle peut également s'adresser à tous restaurants ayant une carte qui change quotidiennement. Les utilisateurs pourront consulter les menus du jour ou du soir des restaurants à proximité sans avoir à se déplacer jusqu'à chaque établissement pour lire l'ardoise. Cela permettrait d'économiser du temps et de l'énergie (pour les travailleurs qui n'ont que peu de temps pour déjeuner par exemple), et de générer de la publicité pour ces restaurants par la même occasion (les clients peuvent être attirés par les plats proposés).

DOSSIER D'ARCHITECTURE TECHNIQUE

Architecture technique générale

Le type d'architecture utilisé pour la réalisation de ce projet est l'architecture Model-View-Controller (MVC).

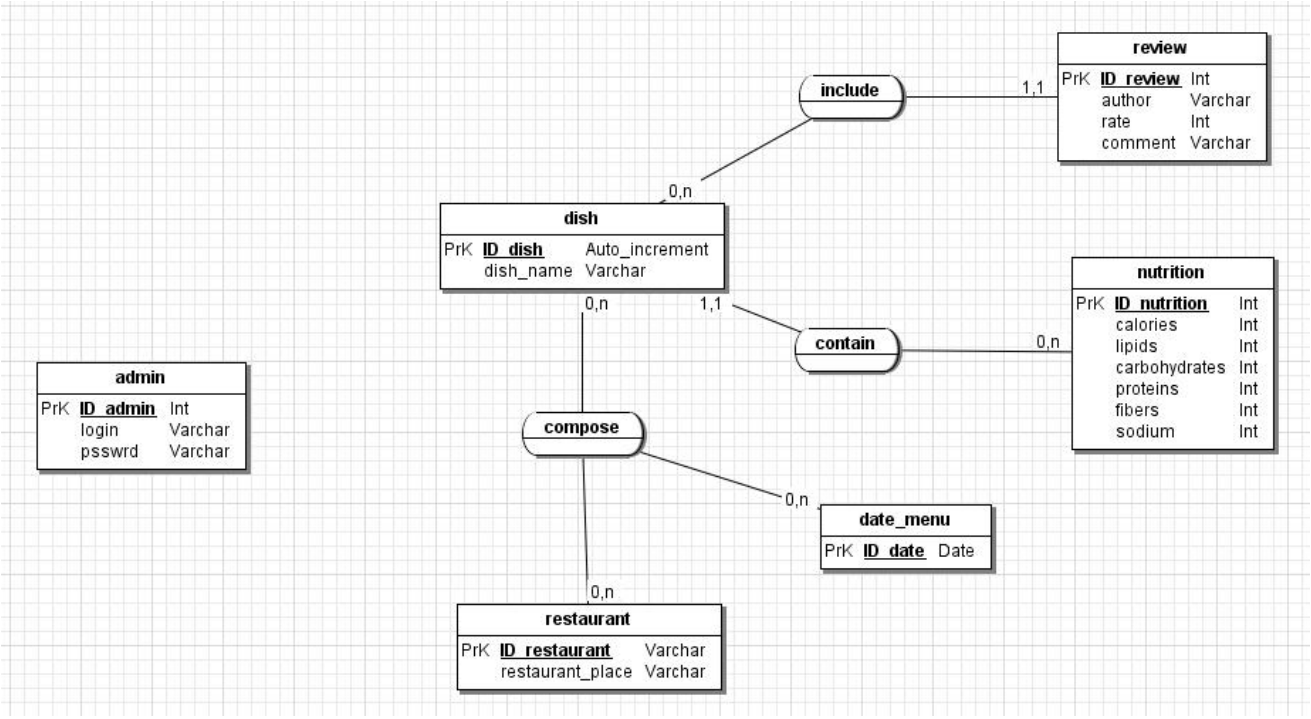


J'ai choisi ce modèle de conception (qui est très répandu dans les application web et les frameworks) pour avoir une organisation des fichiers claire. En effet, la partie Model regroupe la logique métier ainsi que l'accès aux données (c'est ici que s'effectue la communication avec la base de données), la partie View se charge des interactions avec l'utilisateur (présentation, saisie et validation des données), et la partie Controller fait le lien entre l'utilisateur et le reste de l'application en gérant à la fois les models et les views.

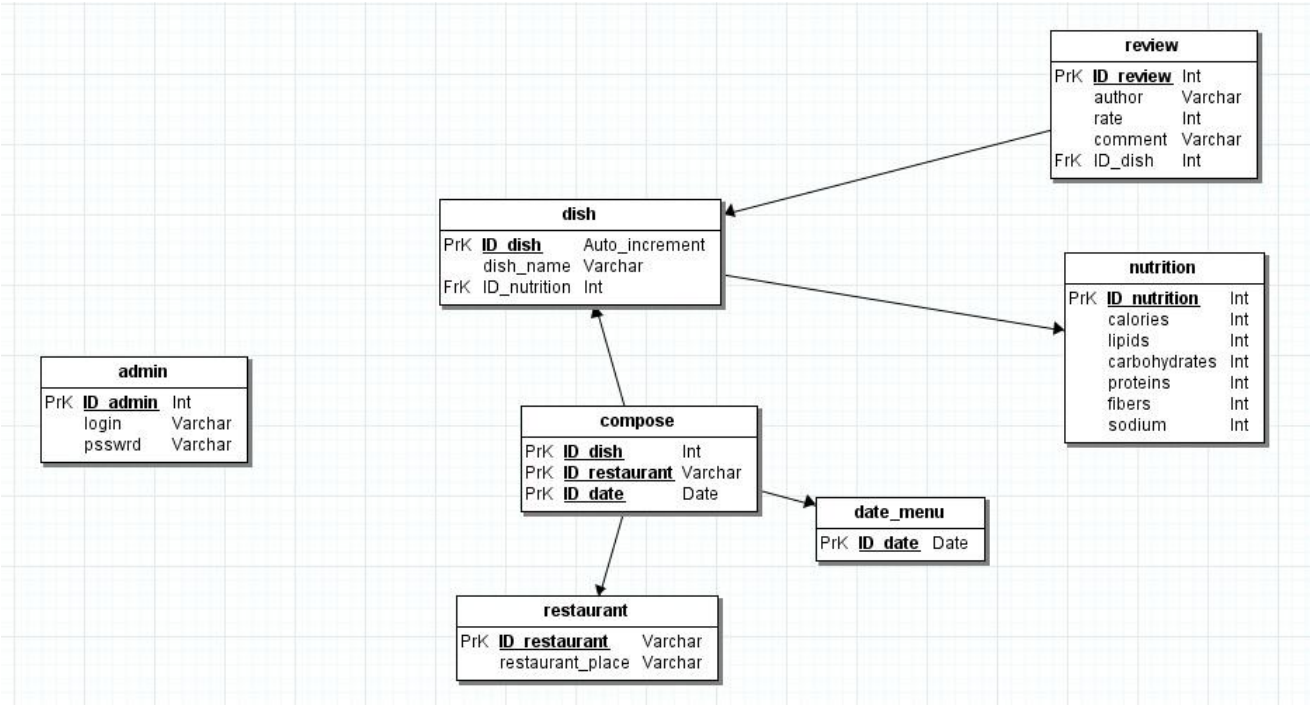
De ce fait, j'ai pu manipuler aisément mes fichiers sans avoir de confusion, et cela rend leur arborescence relativement facile à comprendre pour une personne qui aurait à travailler sur mon code.

Description des données

MCD



MLD



Description des tables

admin : Contient tous les identifiants et mots de passe des administrateurs. Les administrateurs sont des employés du Restaurant Universitaire. Eux seuls ont accès à l'édition des menus, c'est-à-dire à l'ajout ou la suppression de menus.

dish : Contient la liste de tous les plats que le Restaurant Universitaire peut proposer.

review : Contient tous les avis laissés par les utilisateurs à propos d'un plat.

nutrition : Contient tous les apports nutritionnels d'un plat.

restaurant : Contient la liste de tous les restaurants dont est composé le Restaurant Universitaire.

date_menu : Contient toutes les dates pour lesquelles le Restaurant Universitaire a servi des repas.

compose : Résulte de la ternaire entre dish, restaurant, date_menu. Contient l'ensemble des menus. Chaque ligne renseigne donc un plat qui compose le menu d'un restaurant à une date. Le menu est obtenu en effectuant la sélection d'une ou plusieurs lignes de cette table.

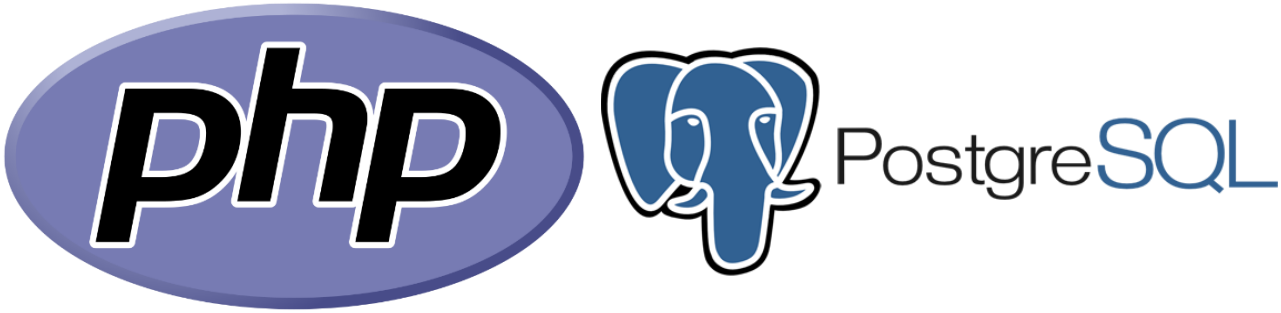
Remarque concernant la table date_menu :

L'existence d'une entité date dans un MCD est souvent contestée et j'en suis conscient.

J'ai beaucoup réfléchi à ce propos, en essayant de la supprimer et d'insérer l'attribut « ID_date Date » comme attribut dans la relation « compose ». Or, lors du passage en MLD, la table « compose » ne contient plus que les clés primaires « ID_dish » et « ID_restaurant », ce qui rend empêche un restaurant de servir un même plat à plusieurs dates différentes.

Il fallait donc une troisième clé primaire pour la date, c'est pour cela que j'ai fait le choix de garder cette entité « date_menu » qui contient les dates d'ouverture du RU et qui facilite l'organisation de la base de données.

Technologies utilisées



Les technologies que j'ai utilisées dans ce projet sont PHP pour le langage (donc HTML et CSS) et PostgreSQL pour la base de données.

Etant novice en matière de développement web, je n'avais jamais vraiment utilisé de langage orienté web. Le PHP (allié au HTML et au CSS) est un langage très utilisé avec une documentation très riche, de nombreux tutoriels et communauté très importante. Sa syntaxe se rapproche du C et il permet également de réaliser toutes les fonctionnalités dont j'ai besoin pour mon application. C'est pour cela que j'ai choisi ce langage pour ce projet.

Concernant la base de données, j'ai utilisé au départ MySQL pour travailler en local sur WampServer (c'est la base de données fournie par défaut). Mais me sentant plus à l'aise sur PostgreSQL, j'ai décidé de l'utiliser lorsque j'ai déployé mon application (explications dans la sous-partie suivante). J'ai utilisé pgAdmin III pour y travailler dessus.

De plus, PostgreSQL est un système open-source puissant, robuste, offrant de nombreuses possibilités et gérant mieux les grandes bases de données. La base de données de RU Hungry est censée s'étoffer un peu chaque jour, et encore davantage si on considère les perspectives d'évolutions de l'application de la partie précédente.

Déploiement

Heroku est un service de cloud computing PaaS (Platform-as-a-Service) qui propose une plateforme de développement d'application web.

J'ai donc pu héberger gratuitement mon application web grâce à ce service à cette adresse :

https://woaruhungry.herokuapp.com

Les interfaces proposées pour gérer ses applications sont ergonomiques et efficaces. J'ai lié mon compte GitHub à Heroku, et plus particulièrement mon dépôt projetWeb (https://github.com/alexaufauvre/projetWeb) à l'app woaruhungry. Ainsi, je peux travailler en local sur WampServer pour effectuer mes essais, puis lorsque j'ai une nouvelle version de mon application, je peux envoyer les modifications directement sur GitHub en lignes de commandes et grâce à Heroku mon site est automatiquement mis à jour ce qui est très pratique.

Heroku m'a également fourni une base de données PostgreSQL sur laquelle j'ai pu travailler aisément.

POST MORTEM

Objectifs

- **Objectifs fixés**

Les objectifs fixés initialement dans la réalisation de ce projet étaient de livrer une application web opérationnelle permettant :

- De pouvoir consulter les menus du jour de tous les restaurants qui composent le RU. Ces informations seraient la majeure partie de la page d'accueil car c'est la fonctionnalité principale que l'on veut proposer aux utilisateurs.
- De pouvoir consulter les menus de la semaine selon le jour ou le restaurant.
- De renseigner les informations concernant chaque menu (apports nutritionnels) ainsi que les tarifs. Ces indications seraient contenues dans la page d'informations du menu.
- De gérer un système d'avis d'utilisateurs afin que les employés du RU puissent avoir des retours concernant les produits qu'ils proposent. Ces avis seraient disponibles dans la page d'informations du menu.
- D'avoir un système de connexion administrateur afin d'ajouter, de supprimer ou de modifier des menus.
- D'aller chercher le plus d'informations possible en base de données pour un affichage dynamique. Utiliser une base de données en lecture et en écriture.
- De pouvoir évoluer et s'adapter aux demandes au fil du temps.

Autres objectifs :

- Etablir une base de données efficace, optimisée et explicite.
- Ecrire un code clair, commenté et pouvant être réutilisé par autrui.
- Avoir des URI propres.
- Avoir une application agréable, ergonomique, responsive, pro et user-friendly.
- Faire une bonne utilisation des technologies, acquérir de nouvelles connaissances et compétences.
- Bien gérer le temps et le planning.

- **Objectifs atteints**

Concernant les fonctionnalités de l'application, la fonctionnalité principale d'affichage des menus du jour pour tous les restaurants fonctionne. RU Hungry est automatiquement mis à jour tous les jours, et les données de la BD sont correctement affichées (lecture de la base de données).

De plus on peut ajouter ou supprimer des plats ou menus pour n'importe quelle date ou n'importe quel restaurant.

Du côté de la base de données, son utilisation est aisée et les requêtes SQL sont claires et explicites.

Le code est organisé et commenté. J'ai essayé d'utiliser le plus possible l'anglais pour tout ce qui n'est pas du FrontEnd (l'application est destinée à un établissement français) car c'est la langue de base pour toute personne qui travaille dans l'informatique.

L'application peut être utilisée sur ordinateur, smartphones ou tablettes.

J'ai acquis des connaissances en architecture et conception web, des compétences en PHP/HTML/CSS.

- **Problèmes rencontrés**

Le principal problème que j'ai rencontré dans ce projet a été le manque de temps et de connaissances. Cela m'a amené à livrer une version beaucoup plus simplifiée que mon idée de départ.

En effet, avant de commencer ce projet, je n'avais aucune compétence en web et deux semaines pour réaliser une application fonctionnelle. J'avais seulement quelques notions vues en projet Piscine et dans les cours de M.Castelltort.

J'ai d'abord perdu du temps à comprendre comment aborder le projet, comment utiliser les technologies, dans quel ordre faire les choses, et comment. De par ce manque d'expérience, je n'ai pas avancé très vite et efficacement tout le long du projet par rapport au temps que j'y passais. Pendant un peu plus d'une semaine, je passais mon temps à chercher les informations dans des documentations ou des cours en ligne (OpenClassRooms, w3schools, etc.) et à faire des tests de PHP/HTML/CSS.

Je n'ai pas rencontré de problèmes techniques concernant les logiciels ou autres. Néanmoins j'ai souvent eu des erreurs avec mon code qui m'ont coûté encore plus de temps.

Tous ces facteurs font qu'au bout des deux semaines, je suis seulement capable de livrer une application qui se limite à afficher, ajouter et supprimer des menus en liaison avec la base de données. Les autres fonctionnalités manquent (Menu de la semaine, Authentification admin, Informations sur le menu), le CSS est pauvre et je n'ai pas eu le temps de faire fonctionner la réécriture de mes URI. De plus, dans les formulaires d'édition de menu, les propositions de plats ne sont pas liées à la base de données.

Je n'ai également pas réussi à lier WampServer et PostgreSQL en local pour avoir la même base de données tout le temps.

Enfin, j'ai rencontré des problèmes concernant la conception de ma base de données. J'ai dû recommencer de nombreuses fois mon MCD pour obtenir quelque chose de viable (le problème de la date n'y était pas pour rien).

Plan personnel

Ce que je referais autrement

Tout d'abord, je commencerais à apprendre à bien utiliser toutes ces technologies (surtout l'architecture MVC et le PHP/HTML/CSS) beaucoup plus tôt, bien avant le lancement du projet. M'autoformer en intensif au début du projet n'était pas une bonne idée si je voulais pouvoir livrer toutes les fonctionnalités et atteindre tous les objectifs.

Je passerais également moins de temps à effectuer des tests de code pour voir ce que ça fait et comment ça rend. J'ai également fait l'erreur de croire qu'utiliser un framework tel que Bootstrap ou Materialize vers le début du projet m'aiderait à mieux imaginer, clarifier, organiser mon contenu et à aller plus rapidement par la suite. Je me concentrerais plus sur les fonctionnalités pour y donner la forme en dernier lieu.

Concernant le travail en local et le déploiement, l'idéal serait de travailler dans les deux cas avec la même base de données. Utiliser MySQL avec WampServer et PostgreSQL sur Heroku n'était pas la chose la plus pratique.

Je reverrais aussi mes objectifs ou mon projet à la baisse. Etant donné mon niveau, j'ai voulu trop faire. Je pensais que les fonctionnalités allaient être nombreuses mais pas très compliquées mais j'avais tort, je n'ai pas bien su évaluer le temps que ça prendrait.

Expérience acquise

Je me suis évidemment confronté à ces problématiques et technologies lors du projet Piscine du premier semestre d'IG3, mais je devais aussi me servir des autres technologies que nous utilisions à ce moment-là (JavaScript entre autres). N'ayant aucune méthode ni notion sur le sujet, je n'avais pas pu me former efficacement sur une technologie en particulier, on m'avait surtout expliqué et j'avais donc un peu touché à tout.

Ce manque de maîtrise et de la bonne méthodologie dû à mon manque d'expérience dans ce domaine m'amène à livrer une application web simple pour laquelle il y aurait encore beaucoup d'améliorations à apporter. Je suis déçu de l'aspect « en travaux » celle-ci, mais d'un autre côté je suis satisfait des fonctionnalités que j'ai eu le temps de réaliser, et de tout ce que j'ai pu apprendre.

Ce projet m'a permis d'appréhender le développement web, domaine auquel j'étais totalement étranger. C'est le premier projet de ce type, seul et en peu de temps, que j'ai eu à entreprendre et ce fut une bonne expérience. Même si j'y ai passé tout mon temps depuis deux semaines, je reste encore motivé à apprendre ces technologies pendant mon temps libre afin de pouvoir être plus à l'aise avec celles-ci et de poursuivre et terminer la réalisation de RU Hungry ultérieurement.