OC Pizza

Système de gestion d'une pizzeria

Dossier de conception technique

Version 1.0

Auteur
Nathalie Ortonne
developpeur d'application

Specinov
Specinov.fr
08 rue de la réussite 78240 Chambourcy – 0119091988– <contact@specinov.fr>
S.A.R.L. au capital de 1 000,00 € enregistrée au RCS de Chambourcy – SIREN 654 987 321– Code APE : 6202A

TABLE DES MATIÈRES

1 - VERSIONS

Auteur	Date	Description	Version
Nathalie Ortonne	26/02/2021	Création du document	1.0

2 - INTRODUCTION

2.1 - Objet du document

Le présent document constitue le dossier de conception technique de l'application OC Pizza à l'attention des développeurs, à la maintenance applicative et à l'équipe technique.
Objectif du document…

Les éléments du présent dossier découlent :

- Du domaine fonctionnel
- Des spécifications techniques
- Du modèle physique de donnée

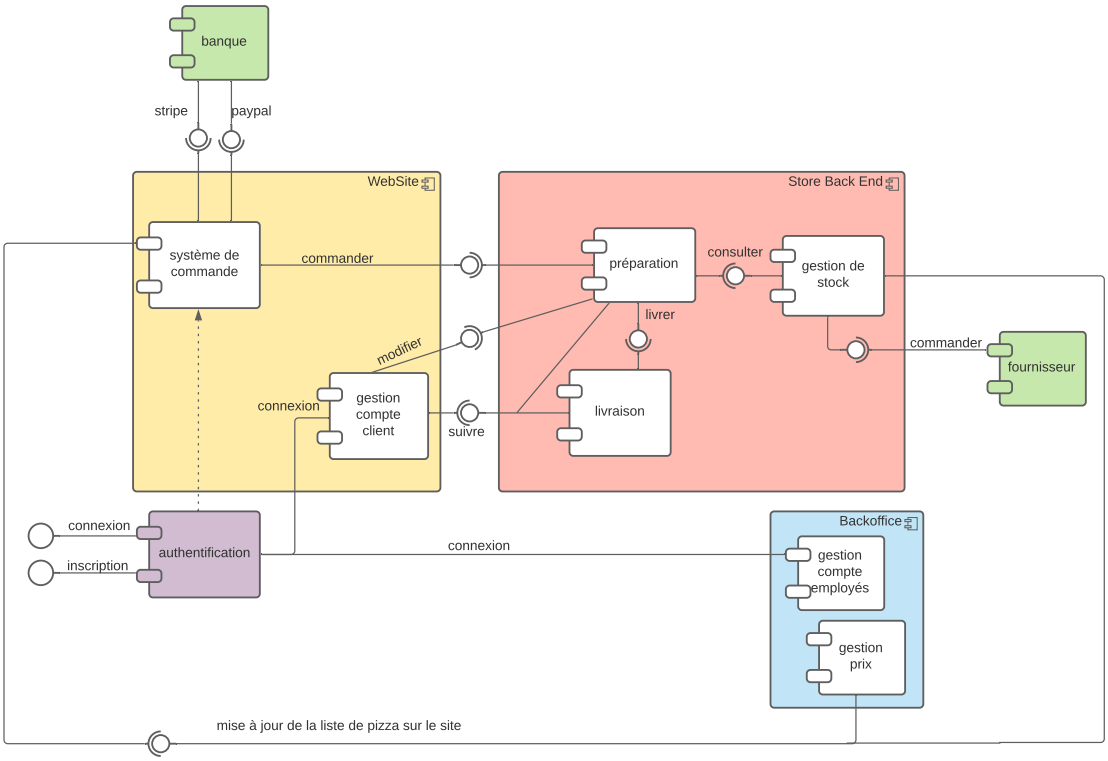
2.2 - Références

Pour de plus amples informations, se référer également aux éléments suivants :

1. Projet_08_Dossier_de_conception_fonctionnelle : Dossier de conception fonctionnelle de l'application
2. Projet_08_Dossier_d_exploitation : Dossier d'exploitation de l'application
3. Projet_08_PV_Livraison : Procès-Verbal de la livraison finale.

3 - Architecture Technique

Afin de répondre au mieux aux besoins du client, nous avons modélisé et identifié les éléments composant le système d'information à mettre en place et leurs interactions. Pour cela nous avons utilisé un diagramme de composant.



3.1 - Composants généraux

3.1.1 - Package Website

Il correspond au package gestion de commande en ligne et gère essentiellement la partie client.

3.1.1.1 - Composant système de commande

Ce composant permet la création d'une commande par un utilisateur. Selon le profil de la session utilisateurs il expose des options plus ou moins complètes, notamment la possibilité de passer une commande.

3.1.1.2 - Composant gestion compte client

Ce composant permet la modification des préférences du compte d'un utilisateur. C'est aussi grâce à se composant qu'un utilisateur peut modifier ou annuler sa commande, ainsi que suivre son évolution.

3.1.2 - Package Store Back End

Ce package correspond à interface pizzeria, et permet de gérer notamment le suivi d'une commande de la préparation à sa livraison

3.1.2.1 - Composant préparation

Ce composant permet l'affichage des commandes reçu par la pizzeria. Il contient les informations détaillées des produits commandé, notamment les fiches recettes nécessaires à la préparation des pizzas. Il interagit avec le composant gestion de stock.

3.1.2.2 - Composant gestion de stock

Ce composant permet de mettre à jour les mouvements de stock et de répondre aux requêtes d'information de stock, notamment pour l'affichage des produits, l'affichage des recettes et le calcul de disponibilité des produits.

3.1.2.3 - Composant livraison

Ce composant permet les traitements relatifs à la livraison.

3.1.3 - Package Back Office

Ce package correspond à la gestion administrative de la pizzeria.

3.1.3.1 - Composant gestion compte employés

Ce composant permet la création d'un compte employé et d'en définir sa fonction. Il permet aussi de consulter toutes les informations relatives à un employé.

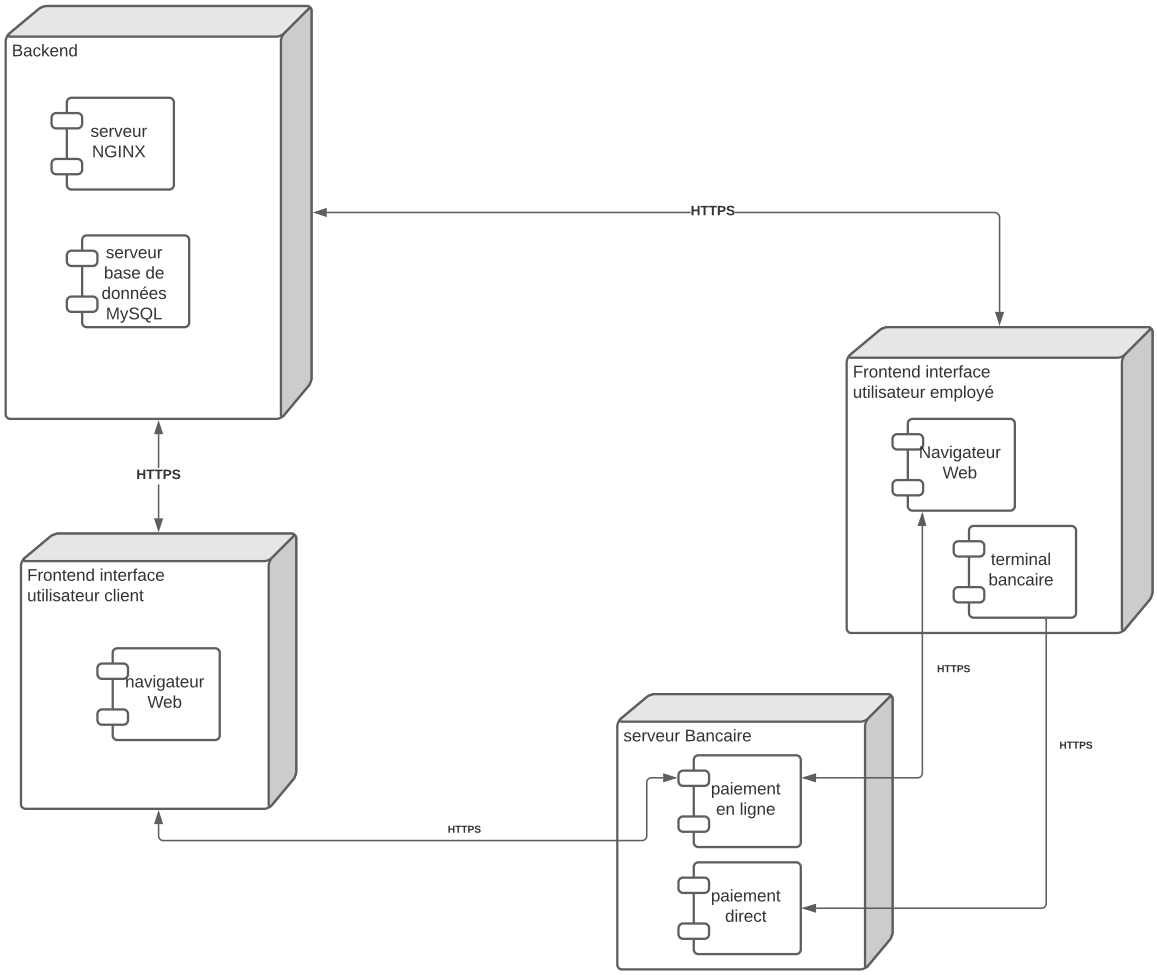
3.1.3.2 - Composant gestion de prix

Ce composant permet de gérer l'affichage de toutes les informations relatives à la pizzeria. Il permet ainsi de mettre à jour les produits disponibles dans une pizzeria sur l'interface client.

3.1.4 - Composant authentification

Ce composant permet l'authentification de l'utilisateur et la création d'un compte utilisateur. C'est ce composant qui permet l'accès aux autres composants.

4 - Architecture de Déploiement



Afin de déployer le système, nous sommes partis sur la base de 5 nœuds :

- Un serveur NGNIX, pour héberger le futur site web ;
- Un serveur MySQL pour la base de données ;
- Un serveur bancaire pour effectuer les paiements en ligne et en direct ;
- Une interface de navigation à destination du client de la pizzeria ;
- Une interface de navigation à destination du groupe OC Pizza.

L'ensemble des nœuds communiquerons entres eux via des liaisons HTTPS, exception faites pour la liaison entre le serveur NGINX et la base de données, qui sera de type TCP/ IP

4.1 - Serveur de Base de données

Pour répondre aux besoins du nouveau système informatique, la gestion des données devra se faire à travers un Système de Gestion de Base de Données (SGBD). Nous avons privilégié l'utilisation d'un serveur MySQL en raison du volume potentiel de celle-ci. En effet c'est une base de données open source, populaire et robuste, compatible avec le langage PHP et facilement.

4.1.1 - Modèle physique de données

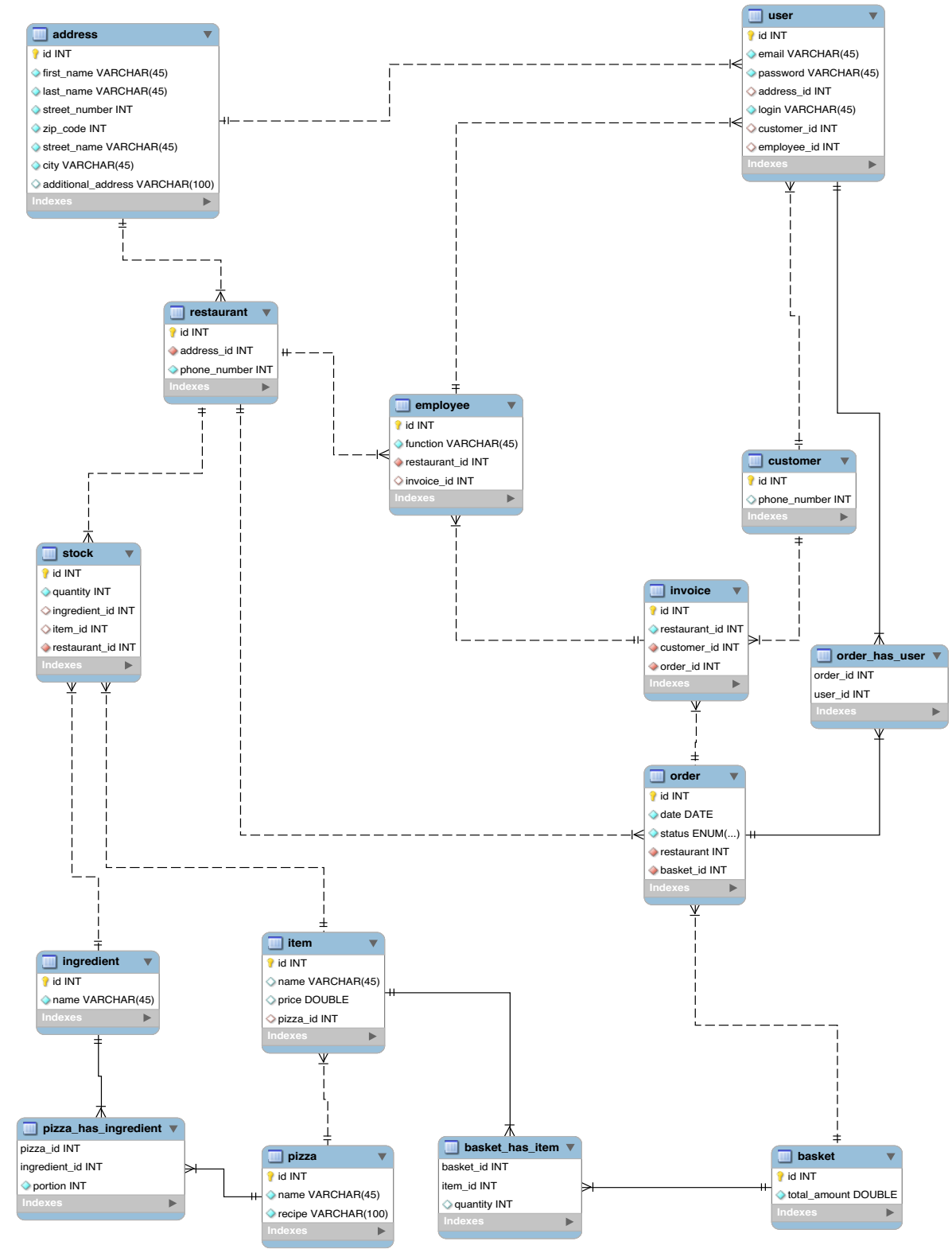
Le Modèle Physique de Données (MPD) sert à modéliser une base de données relationnelle.

Il permet d'avoir une représentation graphique de la structure d'une base de données et de mieux comprendre les relations entre les différentes tables.

A partir du diagramme de classes présenté dans le dossier de conception fonctionnelle, nous avons élaboré le MPD.

Les Tables qui composent ce diagramme dispose d'un attribut « id » qui permet d'identifier chaque ligne de la table de manière unique. Il correspond à la clé primaire.

Cette clé primaire sera utilisée pour définir les clés étrangères et ainsi permettre de définir une relation entre deux tables, et d'assurer la cohérence des données.



Specinov
Specinov.fr

08 rue de la réussite 78240 Chambourcy – 0119091988– <contact@specinov.fr>
S.A.R.L. au capital de 1 000,00 € enregistrée au RCS de Chambourcy – SIREN 654 987 321– Code APE : 6202A

4.1.1.1 - La Tables Order

Elle est le point central de l'application. Pour être validée elle doit être associé à un restaurant, un client, et doit être composé d'un panier pour pouvoir éditer une facture.

4.1.1.2 - La Table Address

Que ce soit le client où le restaurant, ils doivent impérativement posséder une adresse pour être valide.

4.1.1.3 - La Table Restaurant

Elle correspond à l'entité où sera préparée la commande.

4.1.1.4 - La Table Ingredient

Cette table correspond à l'entité qui permet la réalisation de pizza.

4.1.1.5 - La Table Pizza

Elle est composée d'un ou plusieurs ingrédients.

4.1.1.6 - La Table Item

Elle regroupe l'ensemble des produits vendu par le restaurant. Que ce soit une pizza, mais aussi une boisson ou un dessert etc…

4.1.1.7 - La Table Basket

Cette table correspond au panier composé par un client. Il contient tous les articles que le client souhaite commander.

4.1.1.8 - La Table User

C'est l'utilisateur du système, qu'il soit client ou employé. L'utilisateur peut être rattaché à une adresse (personnel pour un client, du restaurant pour un employé), et à une commande.

4.1.1.9 - La Table Customer

C'est un utilisateur client de la pizzeria. Il peut s'authentifier ou créer un compte, composer un panier pour passer une commande, mais aussi modifier ou annuler celle-ci. Il a aussi la possibilité de consulter le statut de sa commande. Ce qui correspond aux composants système de commande et gestion compte client.

4.1.1.10 - La Table Employee

C'est un utilisateur employé de la pizzeria. En fonction de son statut, il a accès aux différentes fonctionnalités du site. Il peut ainsi préparer une commande, la livrer, avoir accès à la gestion administrative. Il a ainsi en fonction de son statut accès aux package Store Back End ou Back Office.

4.1.1.11 - La Table Invoice

Cette table correspond à la facture éditer par le restaurant une fois la commande validée par la pizzeria.

4.1.1.12 - Les Tables de Liaisons

Une table n'est pas à géométrie variable. Ainsi pour les relation dites « many to many » on doit utiliser une table de liaison.

- Stock : Il permet de définir la quantité d'ingrédients et/ou d'article conservée dans un restaurant pour ses ventes.
- Pizza_has_Ingredient : il permet de définir la quantité d'ingrédient, en portion, que contient une pizza.
- Basket_has_Item : il définit la quantité d'article qui compose un panier.
- Order_has_User : Il définit le nombre d'utilisateur qui traite la commande. Le client qui passe la commande et les employés qui s'en occupe.

5 - ARCHITECTURE LOGICIELLE

5.1 - Principes généraux

Les sources et versions du projet sont gérées par Git et les dépendances parsymfonyDependencyInjection.

Le composant SymfonyDependencyInjection fournit une méthode standard pour instancier des objets et gérer la gestion des dépendances.

5.1.1 - Les couches

L'architecture du projet sera basée sur le modèle n-tiers avec 3 couches distinctes :

- **La couche de données** : elle correspond à la base de données ou au système de stockage. Dans notre cas nous utiliserons MySql
- **La couche d'application** : il s'agit du middleware de l'application. Elle sera implémentée dans le langage PHP. Cette couche est le "moteur" du logiciel et elle relie la couche de présentation à la couche de données.
- **La couche de présentation** : l'interface de votre application. Elle sera développée en HTML / CSS. Cette couche affiche les données qu'elle récupère.

5.2 - Les choix techniques

5.2.1 - Back End

5.2.1.1 - Symfony

Notre choix pour Symfony2 s'est fait grâce à plusieurs facteurs :

- Avoir un framework qui gère la base de données à partir de sa propre ORM (Ici, Doctrine).
- Symfonyutilise une architecture MVC. Cette architecture basée sur la forte cohésion et le faible couplage nous permet de manier les changements plus facilement.
- Il est facilement testable grâce à PHPUnit, nous pouvons faire une application de qualité, totalement automatisée. Les tests d'intégrations et unitaires étant un bien non négligeable sur la robustesse du produit, nous mettons tous nos efforts pour ne pas laisser de côté la partie qualité du projet.

5.2.2 - Front End

5.2.2.1 - JQuerry

JQuery est une bibliothèque JavaScript rapide, petite, légère et riche en fonctionnalités.

Il nous permet de rendre le site plus attractif en le rendant dynamique via les animations mais aussi grâce à l'envoie et la récupération de données. On évite de recharger les pages dans certains cas grâces aux appels Ajax.

5.2.2.2 - AngularJs

AngularJS est un framework JavaScript MVC côté client pour développer une application Web dynamic.

Il permet d'utiliser HTML comme langage de modèle et d'étendre la syntaxe de HTML pour exprimer les composants de l'application de manière claire et succincte.

5.2.3 - Hébergement

5.2.3.1 - GitHub

Nous utilisons les bienfaits de git et du service d'hébergement GitHub pour notre projet. Nous fonctionnons par issues et pull requests. Les pull requests forcent les développeurs à voir le travail de toute l'équipe. Les collaborateurs sont libres de donner leurs critiques et commentaires. S'il n'y a aucune amélioration à apporter, nous procédons au merge. Si le code n'est pas suffisamment robuste, clair ou que nous détectons du code mort, le développeur se doit de modifier son travail pour que sa pull request soit acceptée. Les issues peuvent être créées par n'importe quel développeur et être assignées à n'importe quel autre.

Plusieurs labels sont présents pour faciliter la gestion de ces issues :

• Tâche du lab EIP: On répertorie les tickets liés à une interaction directe avec les serveurs, mais aussi pour les mises à jour du site.

• Bugs: la partie résolution et amélioration des incidents rencontrés est disponible dans ce label. Le ticket est fermé seulement si une solution est rencontrée pour la correction du bug.

• Feature: Ici seront présentes les différentes branches nécessaires à la réalisation du projet, ainsi que le système de pull-request.

• Questions: Les différentes questions que possèdent les membres du groupe en lien avec le projet, ou avec le Lab EIP lors des suivis.

• Mises à jour: Les différentes parties implémentées ne sont pas toujours pleinement fonctionnelle ou alors certaines parties peuvent être plus générique. C'est donc ici que se retrouveront toutes les mises à jour du projet.

• Design: L'ergonomie et l'interface de notre site en matière de design sera directement géré via ce label.

• Tests: Si des bundles ou des librairies doivent être testées pour une éventuelle implémentation dans le projet, ils seront répertoriés ici.

6 - GLOSSAIRE

ORM	**Object-RelationalMapping** C'est l'interface qui permet de faire le lien ou "mapping" entre nos objets et les éléments de la base de données.