Projet 6 : OC Pizza

Auteur : *Pellegrini Emanuele*

PROJET 6: Solution technique d'un système de gestion de pizzeria

DESCRIPTION DU DOMAINE FONCTIONNEL

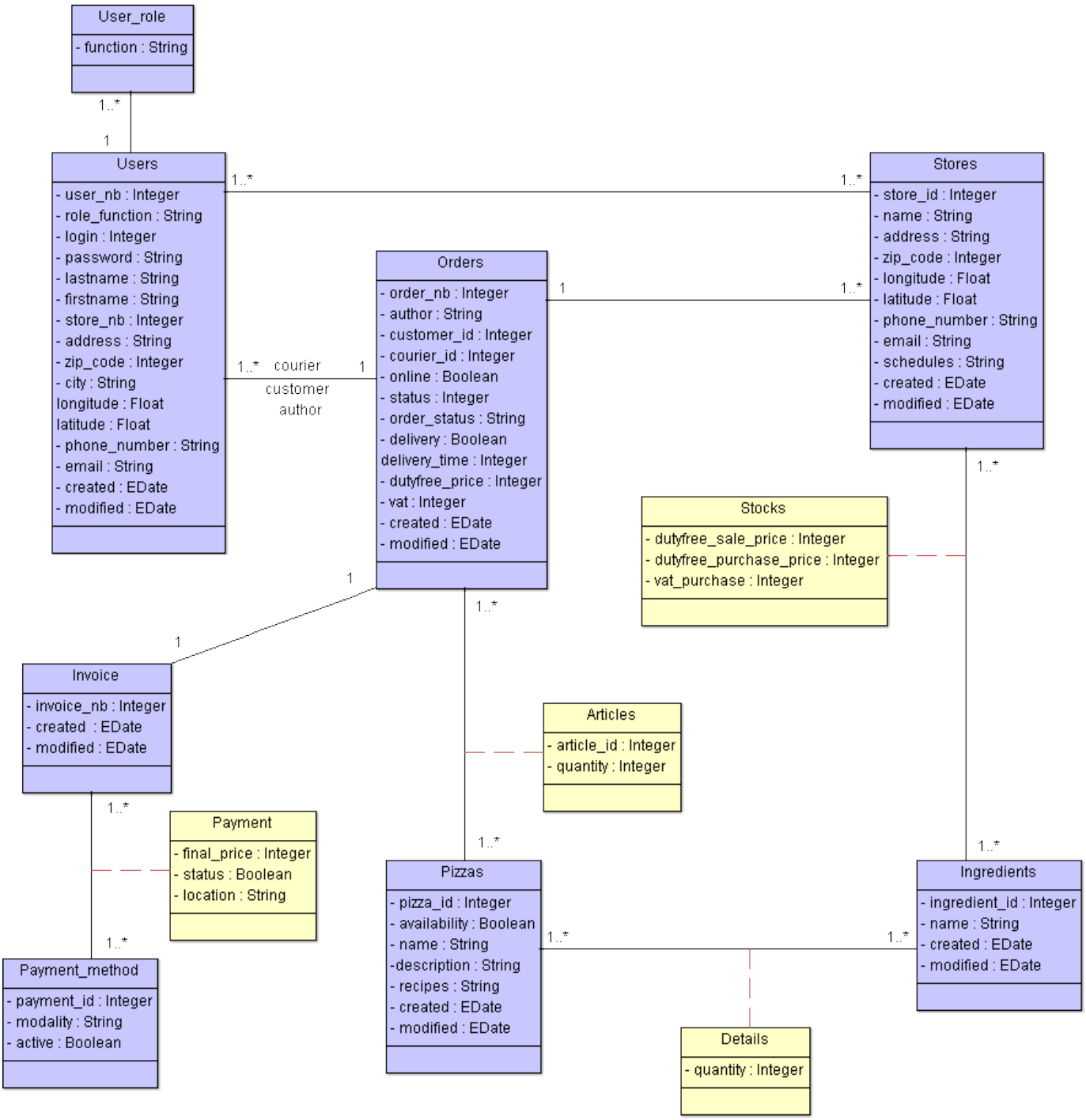
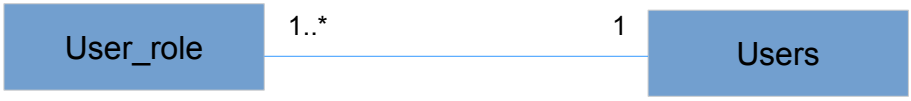


Diagramme de classes

RELATIONS ENTRE CLASSES

Explication du rôle et des liaisons entre les différentes entités du diagramme en dessus.
À noter que tous les noms utilisés dans les diagrammes sont écrits en anglais.

1. Association User_*role* – *Users*



User_role représente la fonction d'un membre de l'équipe.

Users représente un utilisateur ou un potentiel client.

Un utilisateur peut avoir plusieurs fonctions dans l'application, pourtant une fonction ne peut correspondre que à un seul utilisateur.

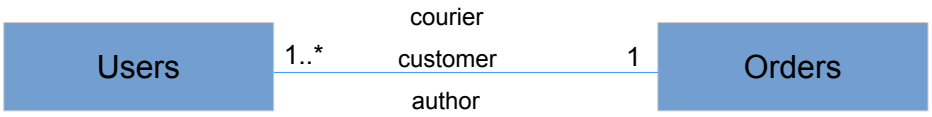
Users	
id [PK]	name
1	Louisa
2	Nicolas
3	Jordan

User_role
name [PK]
client
staff

Users		
id [PK]	name	role_function [FK]
1	Louisa	client
2	Nicolas	staff
3	Jordan	staff

Pour définir la fonction de chaque utilisateur, on se sert de l'attribut *role_function* utilisé en tant que clé étrangère dans la table *Users* et on spécifie la valeur de la clé primaire correspondante.

2. Association *Users – Orders*



Users représente la fonction de l'utilisateur (livreur, client ou auteur de la commande).

Orders représente la commande arrivée et à préparer.

Un utilisateur peut avoir plusieurs rôles par rapport à une commande (il peut être le client, le membre du staff qui prépare la commande ou le livreur), mais une commande ne peut que avoir un unique client, un seul auteur et un livreur spécifique.

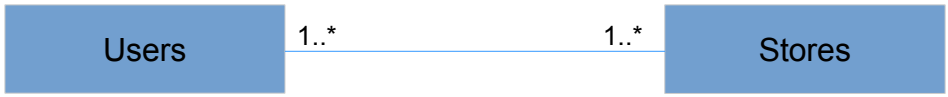
Users	
id [PK]	name
1	Louisa
2	Nicolas
3	Jordan

Orders
id [PK]
1

Orders			
id [PK]	author [FK]	customer[FK]	courier [FK]
1	2	1	3

Pour assigner une fonction à un utilisateur, on ajoute des clés étrangères dans la table *Orders* et on spécifie la valeur de la clé primaire correspondante.

3. Association *Users – Stores*



Users représente en ce cas un membre du staff du magasin en question.

Stores représente un point de vente du groupe OC Pizza.

Un employé peut travailler dans plusieurs points de vente du groupe et un point de vente peut avoir plusieurs membres du staff.

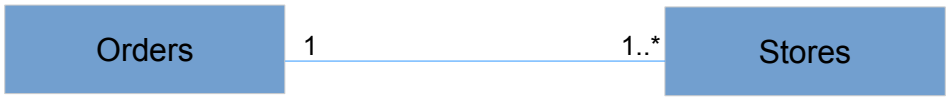
Users	
id [PK]	name
1	Louisa
2	Nicolas
3	Jordan

Stores	
id [PK]	name
1	OC Pizza Rivoli
2	OC Pizza Montmartre

Staff	
user_id [PFK]	store_id [PFK]
1	1
2	2
3	2
2	1

Pour définir les différentes relations, on ajoute une table *Staff* ayant deux clés primaires étrangères (PFK), c'est à dire des clés étrangères qui font référence aux clés primaires des tables *Users* et *Stores*.

4. Association *Orders – Stores*



Orders représente une commande arrivée et à préparer.

Stores représente un point de vente du groupe OC Pizza.

Une commande spécifique fait référence à un seul point de vente, alors qu'un point de vente peut avoir plusieurs commandes à traiter.

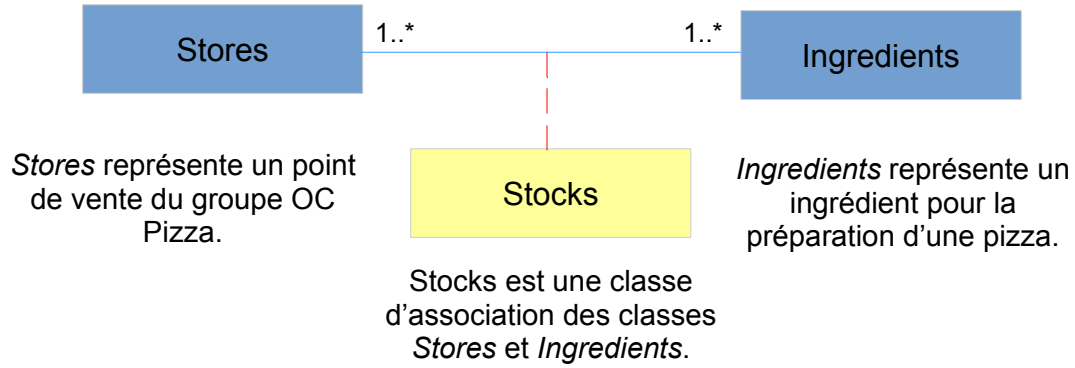
Orders
id [PK]
1
2

Stores	
id [PK]	name
1	OC Pizza Rivoli
2	OC Pizza Montmartre

Orders	
id [PK]	store_id [FK]
1	2
2	1

Pour établir à quel point de vente correspond une commande, on se sert de l'attribut *store_id* en tant que clé étrangère dans la table *Orders* et on spécifie la valeur de la clé primaire correspondante.

5. Association *Stores – Ingredients*



Un point de vente peut stocker plusieurs ingrédients (un stock); plusieurs ingrédients (un stock) peuvent être stockés dans plusieurs points de vente.

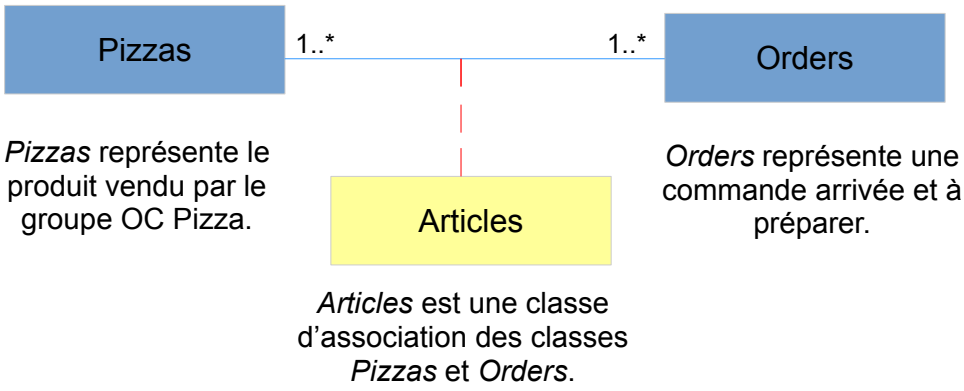
Stores	
id [PK]	name
1	OC Pizza Rivoli
2	OC Pizza Montmartre

Ingredients	
id [PK]	name
4	mozzarella
6	champignons

Stocks		
store_id [PFK]	ingredient_id [PFK]	quantity
1	4	80
2	6	89

Pour définir les différentes relations, on ajoute une table *Stocks* ayant deux clés primaires étrangères (PFK), c'est à dire des clés étrangères qui font référence aux clés primaires des tables *Stores* et *Ingredients*. Ainsi on peut regarder les informations relatives à un ingrédient dans un point de vente spécifique (exemple sa quantité).

6. Association *Pizzas – Orders*



Une commande est composée d'un ou plusieurs articles (pizzas). Une pizza peut concerner une seule ou plusieurs commandes.

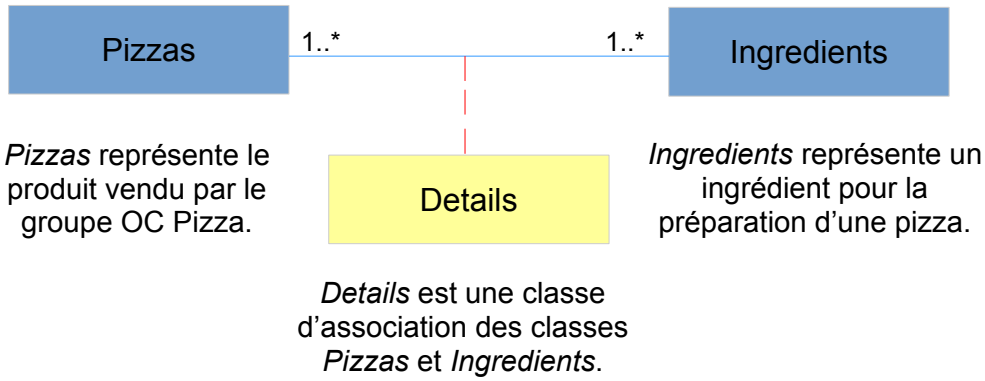
Pizzas	
pizza_id [PK]	name
1	Pizza Thon
2	Pizza Reine

Orders
id [PK]
1
2

Articles		
pizza_id [PFK]	order_id [PFK]	quantity
1	1	2
2	1	4

Pour définir les différentes relations, on ajoute une table *Articles* ayant deux clés primaires étrangères (PFK), c'est à dire des clés étrangères qui font référence aux clés primaires des tables *Pizzas* et *Orders*. Ainsi on peut récupérer les informations relatives à une commande.

7. Association *Pizzas – Ingredients*



Une pizza est composée de un ou plusieurs ingrédients, alors qu'un ou plusieurs ingrédients peuvent être utilisés pour préparer différentes pizzas.
En ce cas *Details* contient la quantité d'ingrédients d'une pizza.

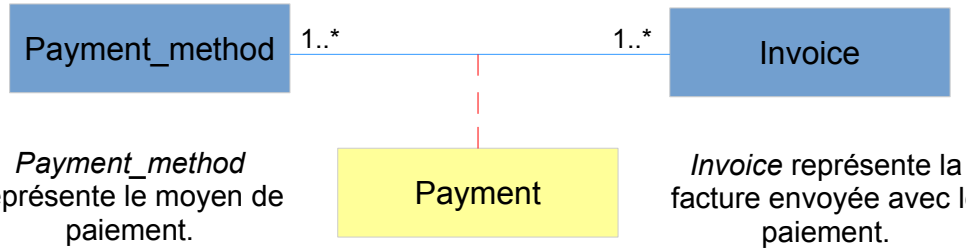
Pizzas	
pizza_id [PK]	name
1	Pizza Thon
2	Pizza Reine

Ingredients	
id [PK]	name
4	mozzarella
6	champignons

Details		
pizza_id [PFK]	ingredient_id [PFK]	quantity
1	4	2
2	6	1

Pour définir les différentes relations, on ajoute une table *Details* ayant deux clés primaires étrangères (PFK), c'est à dire des clés étrangères qui font référence aux clés primaires des tables *Pizzas* et *Ingredients*. Ainsi on peut récupérer les détails pour la composition d'une pizza.

8. Association *Payment_method – Invoice*



Payment_method représente le moyen de paiement.

Invoice représente la facture envoyée avec le paiement.

Payment est une classe d'association des classes *Payment_method* et *Invoice*.

Une méthode de paiement peut apparaître dans une ou plusieurs factures; une facture peut contenir un ou plusieurs méthodes de paiement.

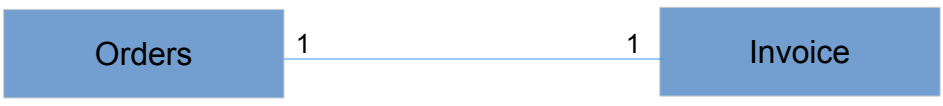
Invoice	
invoice_nb [PK]	order_id [FK]
1	1
2	2

Payment_method
id [PK]
1
2

Payment	
invoice_id [PFK]	payment_method_id [PFK]
1	1
2	1

Pour définir les différentes méthodes de paiement utilisées par les clients, on ajoute une table *Payment* ayant deux clés primaires étrangères (PFK), c'est à dire des clés étrangères qui font référence aux clés primaires des tables *Invoice* et *Paymeny_method*. Ainsi on peut récupérer les informations concernant le paiement d'une commande.

9. Association *Orders – Invoice*



Orders représente une commande arrivée et à préparer.

Invoice représente la facture envoyée avec le paiement.

Une commande ne fait référence qu'à une facture. Une facture ne correspond qu'à une commande spécifique.

Orders
id [PK]
1
2

Invoice
invoice_nb [PK]
1
2

Invoice	
invoice_nb [PK]	order_id [FK]
1	1
2	2

Pour établir la relation entre une commande et une facture, on ajoute une clé étrangère à la table Invoice, en lui donnant la valeur de la clé primaire de la table *Orders.*

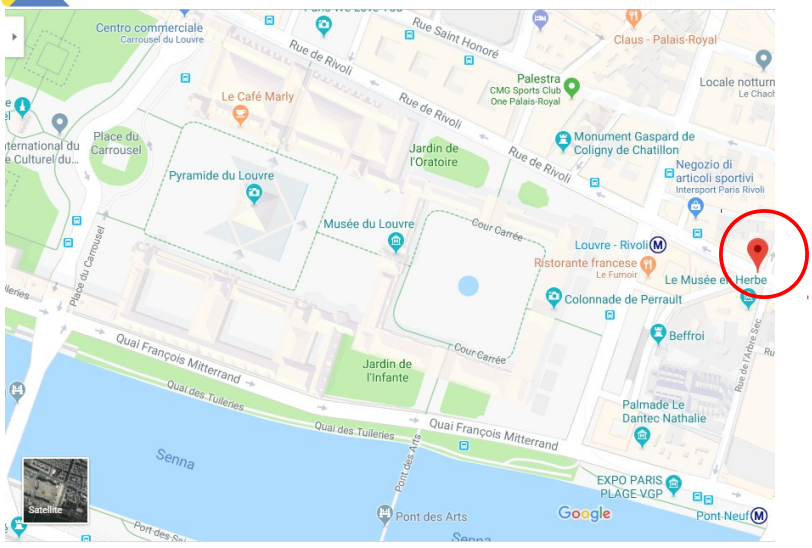
COMPOSANTS EXTERNES

Les composants externes identifiés pour le fonctionnement de l'application sont:

- l'API de géolocalisation Google Maps, indispensable afin de localiser les différents points de vente OC Pizza et les mettre en contact avec les clients, les fournisseurs et les livreurs, mais également pour permettre la liaison entre client et livreur au moment de la commande avec le parcours le plus rapide;

- l'API pour la gestion des paiements en ligne, pour permettre aux clients un paiement sécurisé et rapide.

API Google Maps



Interface de géolocalisation
Oc Pizza Rivoli par un client.

La classe *Stores* et la classe *Users* utiliseront cette API en spécifiant leurs informations de localisation (très utile la fonctionnalité Google de suggestion automatique de l'adresse, qui permet de ne pas avoir à écrire les phrases en entier, envisagée en phase d'analyse fonctionnelle à page 10 du pdf concernant le projet 4). En réponse, l'API renvoie leurs coordonnées en complétant les champs *latitude* et *longitude* dans les tables ayant ces attributs.

En sachant que l'API Google Maps peut être soit gratuite, soit partiellement ou totalement payante selon le nombre de requêtes de géolocalisation reçues et après avoir analysé le nombre de potentiels clients du groupe OC Pizza, on a opté ici pour une solution totalement payante pour satisfaire les besoin de tous les utilisateurs.

API Système Bancaire

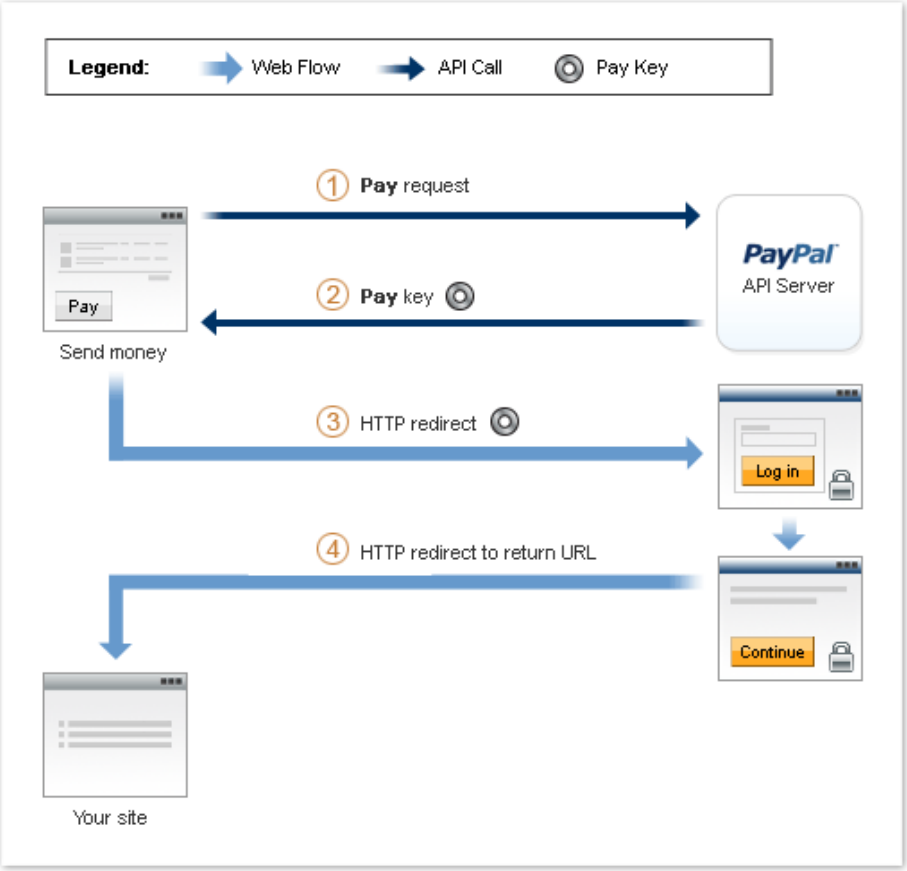


Schéma de fonctionnement d'une API bancaire (ici PayPal).

[*source*: https://www.supinfo.com/articles/single/5529-api-presentation-exemple-une-utilisation-php]

La classe *Users* et la classe *Payment* utiliseront cette API. Les clients seront redirigés sur une page externe et invités à composer leur identifiant et mot de passe pour effectuer le login, ensuite à spécifier les informations relatives au paiement d'une commande et au final à confirmer et finaliser la commande via la méthode de sécurisation choisie par la banque.
L'API alors enverra au client une confirmation de paiement et les redirigera à nouveau à la page précédente.
Dans cet exemple est traitée l'interaction avec l'API PayPal mais il faudra prévoir aussi l'intégration avec d'autres APIs (notamment Visa et MasterCard). Le fonctionnement toutefois sera très similaire.

DIAGRAMME DE DÉPLOIEMENT

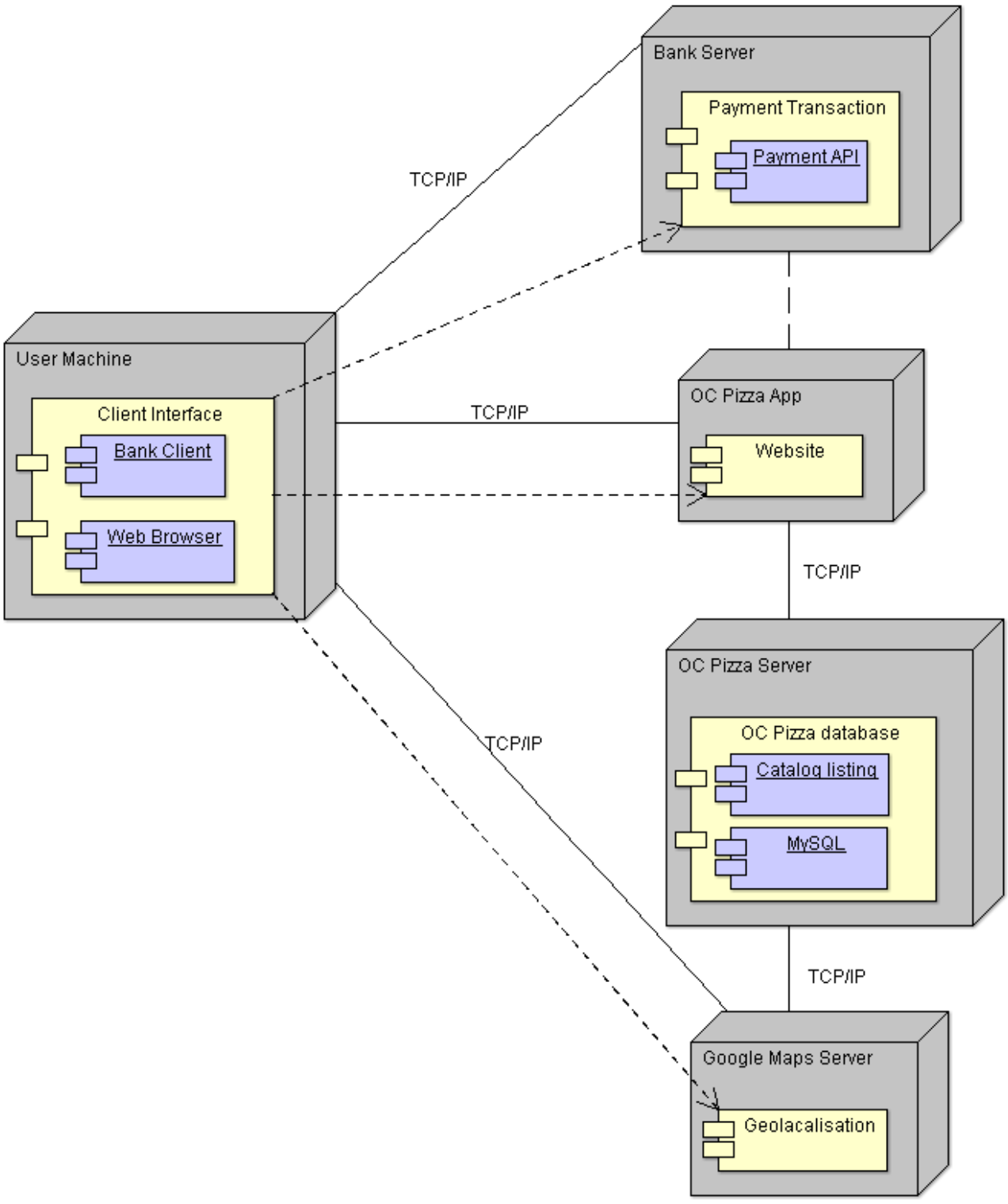


Diagramme de déploiement du système

Comme on peut le voir dans le diagramme, il y a cinq nœuds représentant les éléments hardware ou software faisant partie du système.

- *User Machine* indique l'ordinateur avec lequel l'utilisateur se connecte. *Client Interface* représente l'interface utilisateur, à travers laquelle il peut accéder au site web d'OC Pizza (via le navigateur web *Web Browser*) et payer en ligne (via le client de la banque *Bank Client).*

- *Bank Server* indique le serveur qui gère le requêtes pour les payements en ligne, effectués grâce aux APIs des banques (*Payment API*).

Le transfert des données entre les deux nœuds se fait grâce au protocole TCP/IP. La dépendance entre les composants *Client Interface* et *Payment Transaction* est marquée par la flèche en pointillés.

- OC Pizza App indique l'ensemble de pages web qui composent le site web Oc Pizza, comme décrit par le composant *Website*.

Le transfert de données entre ce nœud et le nœud *User Machine* se fait grâce au protocole TCP/IP. La dépendance entre les composants *Client Interface* et *Website* est marquée par la flèche en pointillés.

- *OC Pizza Server* indique le serveur chargé de gérer les requêtes des pages du site web. Il repère les informations grâce à la base de données dédiée O*c Pizza database* et les affiche à l'utilisateur sous forme de catalogue (*Catalog listing*). Les interactions avec la base de données sont assurées par le SGBDR MySQL.

L'échange des données entre les nœuds *OC Pizza App* et *OC Pizza Server* se fait toujours avec TCP/IP.

- Google Maps Server indique le serveur Google chargé de gérer les requêtes de géolocalisation.

Le transfert des données entre ce nœud et OC Pizza Server est réalisé encore à l'aide du protocole TCP/IP.
Google Maps Server est relié via TCP/IP aussi au nœud *User Machine* et il y a une relation de dépendance entre les composants *Client Interface* et *Geolocalisation*.