



# Spécifications techniques

<b>Introduction</b>	<b>4</b>
Version du document	4
Objet du document	4
Contexte	4
Objectifs	4
Acteurs/Rôles et fonctionnalités attendues	6
<b>Specifications techniques</b>	<b>7</b>
Modélisation du domaine fonctionnel	7
N°1 Person	7
N°2 Role	8
N°3 Employed	8
N°4 Customer	9
N°5 Address	9
N°6 TypeAddress	10
N°7 Pizzeria	11
N°8 Supplier	11
N°9 Contact	12
N°10 UnitOfMeasure	12
N°11 Component	13
N°12 ComponentPrice	13
N°13 Pizza	14
N°14 PizzaLine	15
N°15 PizzaCard	15
N°16 Status	16
N°17 Order	17
N°18 OrderLine	17
N°19 Command	18
N°20 CommandLine	19
N°21 StockMovement	20
N°22 PaymentMethod	21
N°23 Payment	21
<b>Composants</b>	<b>24</b>
Composants système	24
Le Navigateur	24
Système d'exploitation	24
Serveur Web	25
Serveur d'application	25
Application django	25
Serveur de données	25
Composants externes	26
Google Maps API Web Services	26
Twilio API	27
Stripe API	27
<b>Diagramme de déploiement</b>	<b>28</b>

# Introduction

## Version du document

Date	Auteur	Description	Version
21/05/19	Sandrine	Initial document	V1

## Objet du document

Ce document permet de dégager les règles de gestion fonctionnelles du projet “système de gestion de pizzas” pour la société OC Pizza, il comprend :

- les différents acteurs interagissant avec le système proposé
- le descriptif des fonctionnalités
- les règles de gestion fonctionnelles
- le cycle de vie des commandes

## Contexte

La société OC Pizza est un jeune groupe de pizzerias en plein essor et spécialisé dans les pizzas livrées ou à emporter. Il compte déjà 5 points de vente et prévoit d'en ouvrir au moins 3 de plus d'ici la fin de l'année. Un des responsables du groupe a pris contact avec nous afin de mettre en place un système informatique, déployé dans toutes ses pizzerias. OC Pizza a déjà fait une petite prospection et les logiciels existants qu'il a pu trouver ne lui conviennent pas.

## Objectifs

L'application que nous proposons permet d'être plus efficace dans la gestion des commandes, de leur réception à leur livraison en passant par leur préparation. Notre outil permet :

- de suivre en temps réel les commandes passées et en préparation ;
- de suivre en temps réel le stock d'ingrédients restants pour permettre l'actualisation des cartes à pizzas pour n'afficher que celles qui sont encore réalisables ;

- de proposer un site Internet pour que les clients puissent :
  - passer leurs commandes, en plus de la prise de commande par téléphone ou sur place,
  - payer en ligne leur commande s'ils le souhaitent – sinon, ils paieront directement à la livraison
- modifier ou annuler leur commande tant que celle-ci n'a pas été préparée
- de proposer un aide mémoire aux pizzaiolos indiquant la recette de chaque pizza

## Acteurs/Rôles et fonctionnalités attendues

### Acteurs / Rôles

- Le/La responsable de toutes les pizzerias
- Le/La manager d'une pizzeria
- Le/La pizzaiolo
- Le/La Standardiste d'une pizzeria
- Le/La livreur
- Le client (ou client potentiel)

### Les fonctionnalités attendues côté client

- Passer une commande
- Créer un compte "client"
- S'authentifier
- Gérer son compte "client"
- Modifier le statut d'une commande

### Les fonctionnalités attendues côté personnel

- S'authentifier :
- Passer une commande
- Modifier le statut d'une commande

### Les fonctionnalités attendues côté administration



- S'authentifier
- Gérer son personnel

- Afficher les statistiques
- Gérer la carte à pizza
- .Gérer le stock

# Specifications techniques

## Modélisation du domaine fonctionnel

### N°1 Person

pizzapp_person	
	id : serial : <b>Non NULL</b>
	first_name : varchar (100) : <b>NULL</b>
	last_name : varchar (100) : <b>NULL</b>
	phone_number : varchar (50) : <b>NULL</b>
	email : varchar (50) : <b>NULL</b>
	password : varchar (100) : <b>NULL</b>

Tout acteur est représenté par une personne. Ce modèle est composé d'un champ

- id (clé primaire),
- nom, pouvant être null, limité à 100 caractères, non obligatoire,
- prénom, pouvant être null, limité à 100 caractères, non obligatoire,
- téléphone, pouvant être null, limité à 50 caractères, non obligatoire,
- email de connection, pouvant être null, limité à 50 caractères, unique et non obligatoire,
- password de connection, pouvant être null, limité à 100 caractères et non obligatoire.

Le nom, prénom et téléphone ne sont pas obligatoire, car un client qui commande depuis l'application, et qui prend toujours ses pizzas à emporter, peut ne pas remplir ces informations.

L'email et le password sont également non obligatoire, car un client qui se déplace à la pizzeria pour commander ses pizzas, peut ne pas remplir ces informations.

Le choix de laisser les identifiants de connection sur le modèle personne, est stratégique, car dans toutes les situations, la personne peut fournir son email afin de recevoir les publicités et promotions.

## N°2 Role

pizzapp_role	
	id : serial : Non NULL
	short_name : varchar (100) : Non NULL
	description : text : NULL

Chaque acteur de l'application sera identifié par un rôle. Ce modèle est composé d'un champ

- nom court qui est unique et non null limité à 100 caractères et obligatoire,
- description qui peut être nulle et sans limite de caractère.

En fonction du rôle connecté, l'application filtrera les fonctionnalités afin de ne rendre accessible que ce que l'acteur pourra effectuer comme action.




## N°3 Employed

pizzapp_employed	
	id : serial : Non NULL
	is_active : boolean : Non NULL
	registration_date : timestamp with time zone : Non NULL
	inactivity_date : timestamp with time zone : NULL
	comment : text : NULL
	person_id : integer * : Non NULL
	pizzeria_id : integer * : Non NULL

L'employé peut être un responsable administrateur, un standardiste, pizzaiolo, livreur.

Chaque employé est constitué d'un champ

- is\_active, boolean permettant de savoir si l'employé est actif dans la pizzeria, True par défaut
- date d'enregistrement, datetime avec timezone, défaut datetime de l'instance,
- date d'inactivation, datetime avec timezone non obligatoire,
- commentaire lié à l'employé, qui peut être null et sans limite de caractère.
- personne unique et obligatoire (instance du model Person),
- pizzeria dans laquelle il travail, obligatoire (instance du model Pizzeria).

pizzapp_employed_roles	
 id : serial : Non NULL	
 employed_id : integer * : Non NULL	
 role_id : integer * : Non NULL	

L'employé est également constitué d'une

- liste de rôles, ce qui permet en fonction de ses rôles d'accéder à certaines fonctionnalités.




## N°4 Customer

pizzapp_customer	
 id : serial : Non NULL	
registration_date : timestamp with time zone : Non NULL	
comment : text : NULL	
 person_id : integer * : Non NULL	

Le customer est le client, il peut commander ses pizzas par téléphone, sur place ou avec l'application. Il est composé d'une

- date d'enregistrement, au format datetime avec prise en charge de la timezone, défaut datetime de l'instance,
- commentaire qui peut être nul et sans limite de caractère.
- personne, qui est unique et obligatoire (instance du model Person).

## N°5 Address




pizzapp_address	
 id : serial : Non NULL	
title : varchar (100) : NULL	
address_one : varchar (400) : Non NULL	
address_two : varchar (400) : NULL	
zip_code : varchar (20) : Non NULL	
city : varchar (100) : Non NULL	
country : varchar (100) : NULL	
 location : geometry : NULL	
 google_place_id : text : NULL	
loc_address : text : NULL	



L'adresse est un modèle de données qui peut être géo-localisé par l'API Google. Une adresse peut être utilisée par plusieurs objets (customer, fournisseur, pizzeria). Elle est composée d'un champ

- titre, limité à 100 caractères et non obligatoire,
- première adresse, correspondant au numéro et nom de la rue, obligatoire et limitée à 400 caractères,
- deuxième adresse, si complément de rue, non obligatoire et limitée à 400 caractères,
- code postal, obligatoire et limitée à 20 caractères,
- ville, obligatoire et limitée à 100 caractères,
- pays, non obligatoire et limitée à 100 caractères,
- location, qui est un champ point contenant latitude et longitude, élément fourni par l'API Google, non obligatoire
- identifiant google place, text sans limite de caractère, élément unique fourni par l'API Google, non obligatoire,
- loc adresse qui correspond à l'adresse complète envoyé à l'API Google contenant numéro de rue, rue, code postal et ville, il n'est pas obligatoire et sans limite de caractère.

## N°6 TypeAddress

pizzapp_typeaddress	
 id : serial	Non NULL
type : varchar (100)	Non NULL
 address_id : integer	* : Non NULL
 customer_id : integer	* : Non NULL

Le type d'adresse est un modèle de données, qui permet d'enregistrer plusieurs adresse client avec un type différent. Ce modèle contient un champ

- type, permettant de donner un titre à l'adresse, limité à 100 caractères et obligatoire
- une adresse, obligatoire (instance du model Address),
- un client, obligatoire (instance du model Customer).


## N°7 Pizzeria

pizzapp_pizzeria	
	id : serial : Non NULL
	name : varchar (100) : Non NULL
	address_id : integer * : Non NULL

La pizzeria est modèle qui contient un champ

- nom, limité à 100 caractère et obligatoire,
- adresse, obligatoire (instance du model Address).

## N°8 Supplier

pizzapp_supplier	
	id : serial : Non NULL
	company_name : varchar (100) : Non NULL
	phone_number : varchar (50) : NULL
	email : varchar (50) : NULL
	is_active : boolean : Non NULL
	registration_date : timestamp with time zone : Non NULL
	inactivity_date : timestamp with time zone : NULL
	comment : text : NULL
	address_id : integer * : NULL

Le fournisseur est le modèle contenant un champ

- nom de société limité à 100 caractère et obligatoire,
- numéro de téléphone, limité à 50 caractère et non obligatoire,
- email, limité à 50 caractère et non obligatoire,
- is\_active, boolean permettant de savoir si le fournisseur est actif, True par défaut,
- date d'enregistrement, datetime avec timezone, défaut datetime de l'instance,
- date d'inactivité, datetime avec timezone, non obligatoire,
- commentaire, texte sans limite de caractère, non obligatoire,
- adresse, non obligatoire (instance du model Address).

## N°9 Contact

pizzapp_contact	
	id : serial : <b>Non NULL</b>
	is_active : boolean : <b>Non NULL</b>
	registration_date : timestamp with time zone : <b>Non NULL</b>
	inactivity_date : timestamp with time zone : <b>NULL</b>
	comment : text : <b>NULL</b>
	person_id : integer * : <b>Non NULL</b>
	supplier_id : integer * : <b>Non NULL</b>

Le contact est le modèle correspondant aux contacts de personnes travaillant pour un fournisseur. Il contient un champ

- actif, boolean à True par défaut, permettant de savoir si le contact est actif,
- date d'enregistrement, datetime avec timezone, défaut datetime de l'instance,
- date d'inactivité, datetime avec timezone, non obligatoire,
- commentaire, texte sans limite de caractère, non obligatoire,
- adresse, obligatoire.
- personne unique et obligatoire,
- fournisseur, obligatoire.

## N°10 UnitOfMeasure



pizzapp_unitofmeasure	
	id : serial : <b>Non NULL</b>
	short_name : varchar (20) : <b>Non NULL</b>
	description : varchar (100) : <b>NULL</b>

L'unité de mesure est attribué à chaque composant afin de connaître l'unité dans laquelle on le comptabilise. Ce modèle est composé d'un champ

- nom court qui est unique et non null limité à 20 caractères,
- description qui peut être nulle et limité à 100 caractères.

Cette unité de mesure servira pour tous les mouvements de stocks et commandes de composants.


## N°11 Component

pizzapp_component	
	id : serial : Non NULL
	name : varchar (100) : Non NULL
	stock : numeric (8,3) : Non NULL
	in_command : numeric (8,3) : Non NULL
	is_active : boolean : Non NULL
	registration_date : timestamp with time zone : Non NULL
	inactivity_date : timestamp with time zone : NULL
	comment : text : NULL
	unit_of_measure_id : integer * : Non NULL

Le composant, est le modèles de données des sous articles qui composent les pizzas mais également de tous les articles qui sont stockés dans les pizzérias en dehors des pizzas. Ce modèle contient un champ

- nom, limité à 100 caractères et obligatoire,
- quantité en stock, numérique de 8 chiffres dont 3 décimales, le stock est mis à jour à chaque mouvement de stock, numérique de 8 chiffres dont 3 décimales, obligatoire,
- quantité en commande, obligatoire, 0 par défaut,
- date d'enregistrement, datetime avec timezone, défaut datetime de l'instance,
- date d'inactivité, datetime avec timezone, non obligatoire,
- commentaire, texte sans limite de caractère, non obligatoire,
- unité de mesure, obligatoire.



## N°12 ComponentPrice

pizzapp_componentprice	
	id : serial : Non NULL
	cost_price_currency : varchar (3) : Non NULL
	cost_price : numeric (7,2) : Non NULL
	is_active : boolean : Non NULL
	registration_date : timestamp with time zone : Non NULL
	inactivity_date : timestamp with time zone : NULL
	comment : text : NULL
	component_id : integer * : Non NULL
	supplier_id : integer * : Non NULL

Le composant prix est un modèle permettant de stocker le prix d'un composant en fonction d'un fournisseur. Un composant peut avoir plusieurs prix différents chez plusieurs fournisseurs. Ce modèle contient un champ

- currency du prix d'achat, par défaut EUR,
- prix d'achat, numérique de 7 chiffres dont 3 décimales, obligatoire,
- is\_active, boolean permettant de savoir si le composant est actif, True par défaut,
- date d'enregistrement, datetime avec timezone, défaut datetime de l'instance,
- date d'inactivité, datetime avec timezone, non obligatoire,
- commentaire, texte sans limite de caractère, non obligatoire,
- composant, obligatoire,
- fournisseur, obligatoire.

## N°13 Pizza





pizzapp_pizza	
	id : serial : Non NULL
	name : varchar (100) : Non NULL
	price_currency : varchar (3) : Non NULL
	price : numeric (7,2) : Non NULL
	is_active : boolean : Non NULL
	registration_date : timestamp with time zone : Non NULL
	inactivity_date : timestamp with time zone : NULL
	comment : text : NULL

Le modèle pizza représente l'objet pizza qui sera sur la carte à pizza, il contient un champ

- nom, limité à 100 caractère, unique et obligatoire,
- currency du prix de vente, par défaut EUR, obligatoire
- prix de vente, numérique de 7 chiffres dont 3 décimales, obligatoire,
- is\_active, boolean permettant de savoir si le fournisseur est actif, True par défaut,
- date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- date d'inactivité, datetime avec timezone, non obligatoire,
- commentaire, texte sans limite de caractère, non obligatoire.






## N°14 PizzaLine

pizzapp_pizzaline	
	id : serial : Non NULL
	quantity : numeric (8,3) : Non NULL
	is_active : boolean : Non NULL
	registration_date : timestamp with time zone : Non NULL
	inactivity_date : timestamp with time zone : NULL
	comment : text : NULL
	component_id : integer * : Non NULL
	pizza_id : integer * : Non NULL
	unit_of_measure_id : integer * : Non NULL

Le modèle ligne pizza permet de composer les lignes des composant de la pizza avec les quantité ce qui permet les mouvement de stocks et la mise à jour en temps réel des composants et donc de la mise à jour de la carte à pizza. Ce modèle contient un champ




- quantité du composant, numérique de 8 chiffres dont 3 décimales, obligatoire,
- is\_active, boolean permettant de savoir si la ligne de pizza est active, True par défaut,
- date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- date d'inactivité, datetime avec timezone, non obligatoire,
- commentaire, texte sans limite de caractère, non obligatoire,
- composant, obligatoire,
- pizza, obligatoire,
- unité de mesure du composant, obligatoire.

## N°15 PizzaCard

pizzapp_pizzacard	
	id : serial : Non NULL
	name : varchar (100) : Non NULL
	is_active : boolean : Non NULL
	registration_date : timestamp with time zone : Non NULL
	inactivity_date : timestamp with time zone : NULL
	comment : text : NULL
	employed_id : integer * : Non NULL
	pizzeria_id : integer * : Non NULL

Le modèle de carte à pizza, permet de composer une ou plusieurs cartes à pizzas pour une pizzeria. Une validation métier limitera le nombre de cartes à pizzas active à 1, mais ceci laissera la possibilité aux pizzérias de garder en mémoire des cartes à pizzas non actives. Ce modèle contient un champ

- nom, limité à 100 caractère, unique et obligatoire,
- currency du prix de vente, par défaut EUR, obligatoire
- prix de vente, numérique de 7 chiffres dont 3 décimales, obligatoire,
- is\_active, boolean permettant de savoir si le fournisseur est actif, True par défaut,
- date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- date d'inactivité, datetime avec timezone, non obligatoire,
- commentaire, texte sans limite de caractère, non obligatoire,
- employé, responsable de la création de la carte, obligatoire,
- pizzeria, obligatoire,

pizzapp_pizzacard_pizzas	
	id : serial : Non NULL
	pizzacard_id : integer * : Non NULL
	pizza_id : integer * : Non NULL

- une liste de pizzas.






## N°16 Status

pizzapp_status	
	id : serial : Non NULL
	short_name : varchar (20) : Non NULL
	description : varchar (100) : NULL

Le statut constitue une table, car des énumérations limites l'ajouts de nouvelles instances. Ce modèle contient un champ

- nom court qui est unique et non null limité à 100 caractères et obligatoire,
- description qui peut être nulle et sans limite de caractère.

## N°17 Order




pizzapp_order	
	id : serial : Non NULL
	delay : timestamp with time zone : Non NULL
	is_active : boolean : Non NULL
	registration_date : timestamp with time zone : Non NULL
	inactivity_date : timestamp with time zone : NULL
	comment : text : NULL
	customer_id : integer * : Non NULL
	employed_id : integer * : NULL
	pizzeria_id : integer * : Non NULL
	status_id : integer * : Non NULL

Le modèle commande représente les commandes des pizzas pour les clients. Le status de la commande sera mis à jour tout au long du processus de la commande, permettant ainsi de suivre en temps réel l'état des commandes. Ce modèle contient un champ

- délai de livraison, datetime avec timezone, obligatoire,
- is\_active, boolean permettant de savoir si la commande est active, True par défaut,
- date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- date d'inactivité, datetime avec timezone, non obligatoire,
- commentaire, texte sans limite de caractère, non obligatoire,
- client, responsable de la création de la commande, obligatoire,
- employé, responsable de la création de la commande, non obligatoire,
- pizzeria, obligatoire,
- status de la commande, obligatoire.






## N°18 OrderLine

pizzapp_orderline	
	id : serial : Non NULL
	quantity : integer : Non NULL
	is_active : boolean : Non NULL
	registration_date : timestamp with time zone : Non NULL
	inactivity_date : timestamp with time zone : NULL
	comment : text : NULL
	order_id : integer * : Non NULL
	pizza_id : integer * : Non NULL

Les lignes de commandes permettent de gérer les pizzas commandées par les clients. Ce modèle contient un champ

- quantité de pizzas, integer, obligatoire,
- is\_active, boolean permettant de savoir si le fournisseur est actif, True par défaut,
- date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- date d'inactivité, datetime avec timezone, non obligatoire,
- commentaire, texte sans limite de caractère, non obligatoire,
- commande, obligatoire,
- pizza, obligatoire.

## N°19 Command





pizzapp_command	
	id : serial : Non NULL
	delay : timestamp with time zone : Non NULL
	is_active : boolean : Non NULL
	registration_date : timestamp with time zone : Non NULL
	inactivity_date : timestamp with time zone : NULL
	comment : text : NULL
	employed_id : integer * : Non NULL
	supplier_id : integer * : Non NULL

La commande fournisseur, permet de gérer l'approvisionnement des composants. Ce modèle contient le champ

- délai de livraison, datetime avec timezone, obligatoire,

- is\_active, boolean permettant de savoir si la commande est active, True par défaut,
- date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- date d'inactivité, datetime avec timezone, non obligatoire,
- commentaire, texte sans limite de caractère, non obligatoire,
- client, responsable de la création de la commande, obligatoire,
- employé, responsable de la création de la commande fournisseur, obligatoire,
- fournisseur, obligatoire.






## N°20 CommandLine

pizzapp_commandline	
	id : serial : <b>Non NULL</b>
	quantity : numeric (8,3) : <b>NULL</b>
	delay : timestamp with time zone : <b>Non NULL</b>
	is_active : boolean : <b>Non NULL</b>
	registration_date : timestamp with time zone : <b>Non NULL</b>
	inactivity_date : timestamp with time zone : <b>NULL</b>
	comment : text : <b>NULL</b>
	command_id : integer * : <b>Non NULL</b>
	component_id : integer * : <b>Non NULL</b>
	unit_of_measure_id : integer * : <b>Non NULL</b>

La ligne de commande fournisseur, est le détails des lignes de commandes de composant, ce modèle sera utilisé directement pour les mouvements de stocks, il contient un champ

- quantité du composant, numérique de 8 chiffres dont 3 décimales, obligatoire,
- délai de livraison, datetime avec timezone, obligatoire,
- is\_active, boolean permettant de savoir si la ligne de commande est active, True par défaut,
- date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- date d'inactivité, datetime avec timezone, non obligatoire,
- commentaire, texte sans limite de caractère, non obligatoire,
- commande, obligatoire,
- composant, obligatoire,
- unité de mesure, obligatoire.

## N°21 StockMovement

pizzapp_stockmovement	
	id : serial : Non NULL
	quantity : numeric (8,3) : Non NULL
	stock_before : numeric (8,3) : Non NULL
	stock_after : numeric (8,3) : Non NULL
	is_active : boolean : Non NULL
	registration_date : timestamp with time zone : Non NULL
	inactivity_date : timestamp with time zone : NULL
	comment : text : NULL
	command_line_id : integer * : NULL
	component_id : integer * : Non NULL
	order_line_id : integer * : NULL
	unit_of_measure_id : integer * : Non NULL

Le modèle mouvement de stock permet à chaque changement de statut des commandes (passage en “en attente de livraison”), de décrémenter les composants qui auront théoriquement servi à la fabrication de la commande. Le réajustement pourra se faire par une modification manuelle du stock. Ce modèle contient le champ

- quantité du composant, numérique de 8 chiffres dont 3 décimales, obligatoire,
- stock précédent du composant, numérique de 8 chiffres dont 3 décimales, défaut à zéro,
- stock après mouvement du composant, numérique de 8 chiffres dont 3 décimales, défaut à zéro,
- is\_active, boolean permettant de savoir si le mouvement de stock est actif, True par défaut,
- date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- date d'inactivité, datetime avec timezone, non obligatoire,
- commentaire, texte sans limite de caractère, non obligatoire,
- ligne de commande fournisseur, non obligatoire,
- composant, obligatoire,
- ligne de commande client, non obligatoire,
- unité de mesure, obligatoire.




## N°22 PaymentMethod

pizzapp_paymentmethod	
	id : serial : Non NULL
	short_name : varchar (20) : Non NULL
	description : varchar (100) : NULL

Le modèle méthode de paiement existe afin de ne pas mettre d'énumération et se trouver limitée dans la création des instance. Ce modèle représente les différent moyens de paiements autorisés par les pizzérias. Ce modèle contient un champ

- nom court qui est unique et non null limité à 100 caractères et obligatoire,
- description qui peut être nulle et limité à 100 caractères.

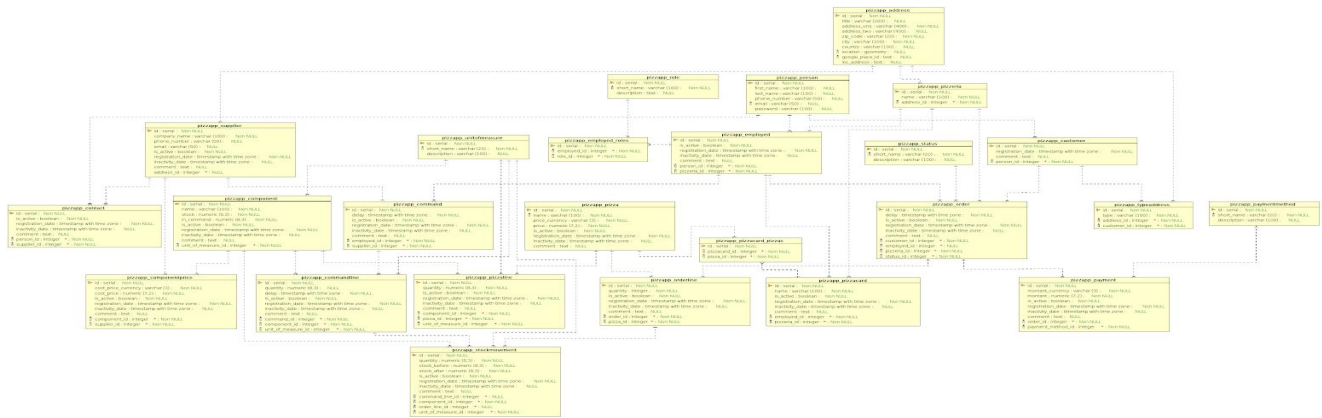
## N°23 Payment

pizzapp_payment	
	id : serial : Non NULL
	montant_currency : varchar (3) : Non NULL
	montant : numeric (7,2) : Non NULL
	is_active : boolean : Non NULL
	registration_date : timestamp with time zone : Non NULL
	inactivity_date : timestamp with time zone : NULL
	comment : text : NULL
	order_id : integer * : Non NULL
	payment_method_id : integer * : Non NULL

Le modèle de données payment correspond à chaque moyen de paiement utilisé afin de régler une commande client. Ce modèle contient un champ

- currency du montant réglé, par défaut EUR, obligatoire
- montant réglé, numérique de 7 chiffres dont 3 décimales, obligatoire,
- is\_active, boolean permettant de savoir si le paiement est actif, True par défaut,
- date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- date d'inactivité, datetime avec timezone, non obligatoire,
- commentaire, texte sans limite de caractère, non obligatoire,
- commande client, obligatoire,
- méthode de paiement, obligatoire.

## Synthèse du modèle physique de données reprenant les spécifications fonctionnelles de l'application :



une version fichier image png est jointe aux livrable à des fins de lisibilités.



# Composants

## Composants système

### Le Navigateur

Nous avons fait le choix de développer une application Web pour une question financière. L'application est donc accessible avec une connexion internet, et depuis un navigateur, sur tout type de support (PC, tablette, smartphone) dans la mesure où le support prend en charge l'installation de navigateur récent.

- **L'interface utilisateur** - ce qui inclut la barre d'adresse, les boutons avant et arrière, le menu de marque-page, etc. En fait, chacune des parties affichées par le navigateur excepté la fenêtre principale dans laquelle vous voyez la page demandée ;
- **Le moteur du navigateur** - contrôle les actions entre l'interface et le moteur de rendu ;
- **Le moteur de rendu** - responsable de l'affichage du contenu demandé. Par exemple, si le contenu demandé est au format HTML, il est chargé d'analyser le code HTML et CSS et d'afficher le contenu analysé à l'écran ;
- **Le réseau** - utilisé pour les appels réseau, comme les requêtes HTTP. Il possède une interface indépendante de la plateforme et en dessous des implémentations pour chaque plateforme ;
- **L'interface utilisateur** - utilisée pour dessiner des widgets de base comme des listes déroulantes et des fenêtres. Le navigateur expose une interface générique qui n'est pas spécifique à la plateforme. En dessous, il utilise l'interface utilisateur du système d'exploitation ;
- **L'interpréteur JavaScript** - utilisé pour analyser et exécuter le code JavaScript ;
- **Le stockage de données** - il s'agit d'une couche de persistance. Le navigateur doit enregistrer toutes sortes de données sur le disque dur, par exemple, des cookies. La nouvelle spécification HTML (HTML5) définit le terme « base de données Web », qui est un système complet (bien que léger) de base de données dans le navigateur.

### Système d'exploitation

Nous avons choisi d'héberger l'application sur une Plateform As A Service (type Heroku).

Le système d'exploitation installé sera **LINUX** qui est open-source.

# Serveur Web

**NGINX** est un serveur web léger et performant. Il est particulièrement performant pour servir des fichiers statiques et pour analyser des URL. Pour cette raison, il est couramment employé en tant que reverse-proxy. Le backend sera un serveur Daphné configuré pour gérer les requêtes Http et les websockets.

Les deux raisons principales qui peuvent amener à utiliser un reverse-proxy sont l'amélioration :

- de la sécurité
- des performances.

Nginx transmet au serveur d'application les requêtes Http et WebSocket.

## Serveur d'application

**DAPHNE** est un des serveur d'application (python) utilisé pour servir les requêtes websockets à l'application, il peut également lui servir les requêtes Http, gérant ainsi les demandes synchrones et asynchrones.

## Application django

**DJANGO** est un framework open-source python, permettant de créer rapidement des applications web. Il permet de générer des templates Html, composé de CSS de Javascript, voir de JQuery, qui seront interprétés par le navigateur.

## Serveur de données

**POSTGRESQL** est un SGBDR (système de gestion de base de données relationnelles). L'application django communique avec le serveur de données en protocole TCP/IP et transmet les ajouts, modifications et suppressions des données stockées.

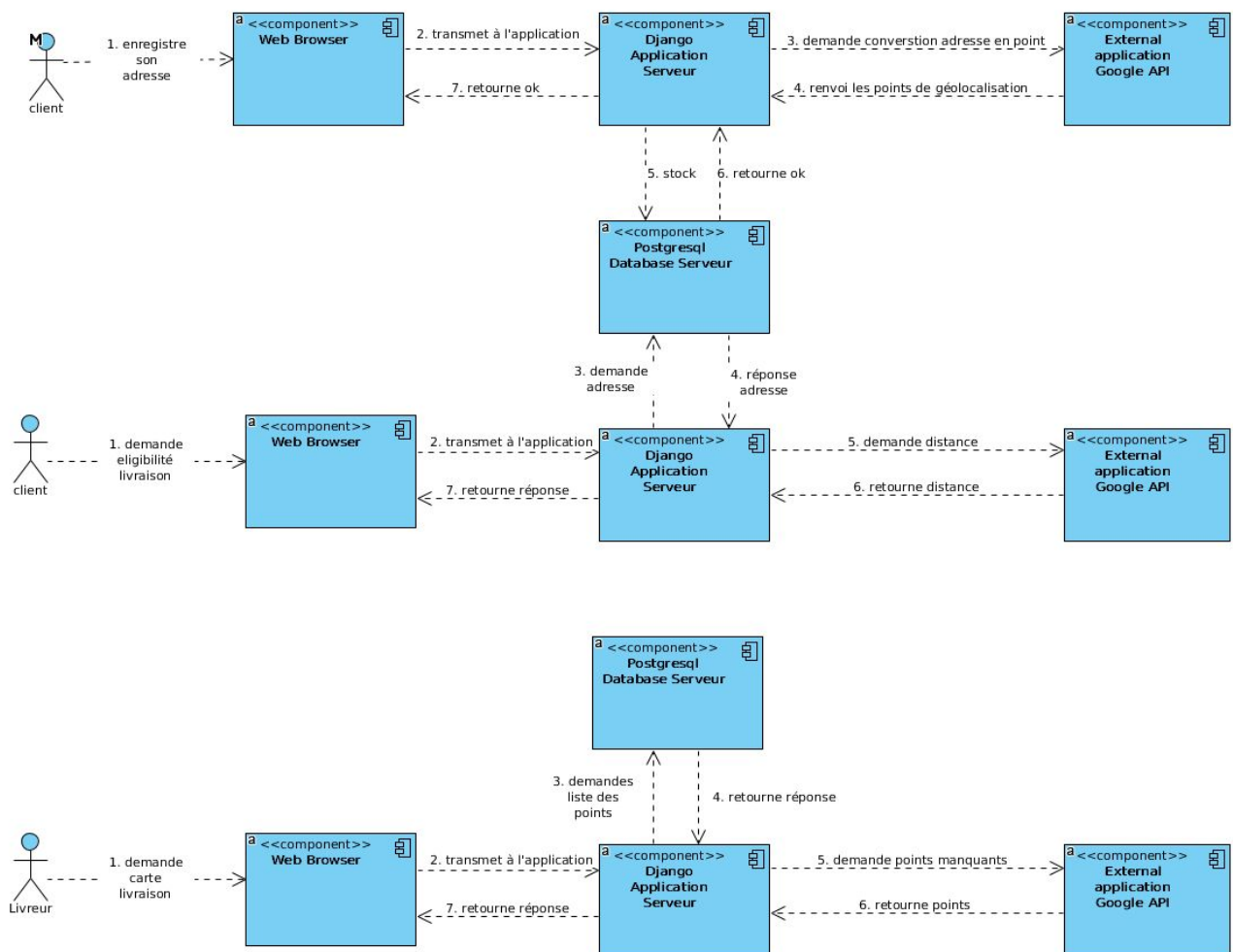
# Composants externes

## Google Maps API Web Services

L'API Google Map permettra de convertir une adresse postale en point géolocalisable, ce qui permettra à l'application de placer sur une carte les points de livraisons pour les livreurs des pizzerias.

Nous utiliserons également cette api afin de calculer la distance par route entre les clients et les pizzerias afin de valider l'acceptabilité des livraisons.

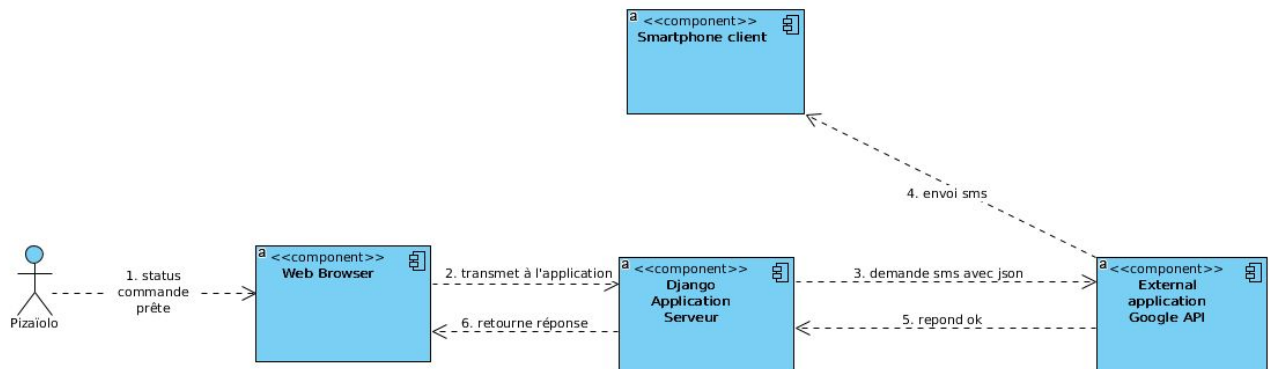
Nous utiliserons la bibliothèque python Google map service afin de nous faciliter la mise en place des outils google (outils conseillé par l'API).





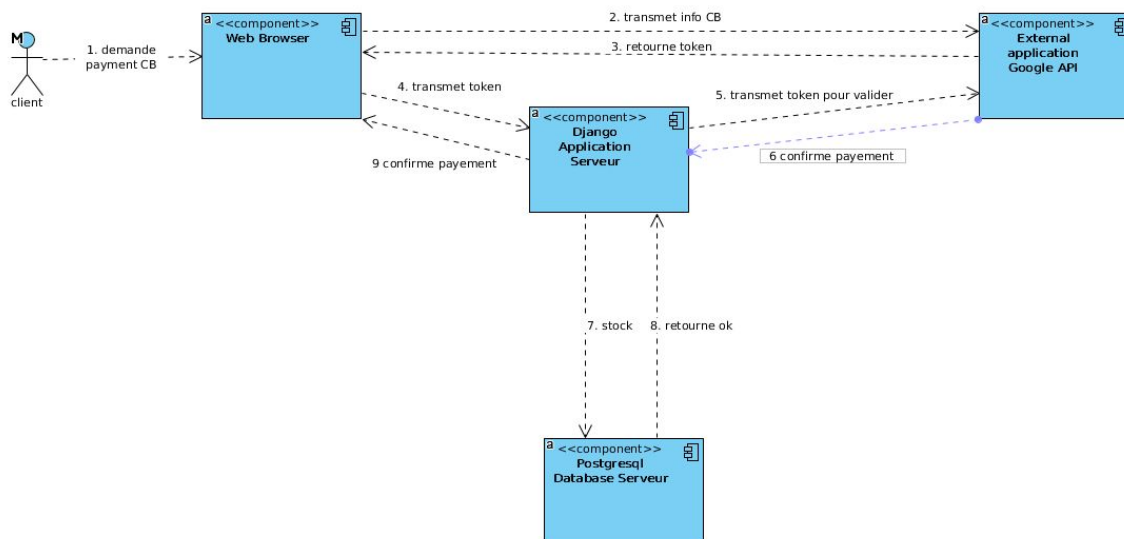
# Twilio API

L'API Twilio, permettra de paramétrer et d'envoyer des sms aux clients pour les informés lorsque leurs commandes seront prêtes.



# Stripe API

L'API Stripe sera utilisée afin de mettre en place un système de règlement des commandes par cartes bancaires.



# Diagramme de déploiement

