

BONNEAU Baptiste
VERDIER Nathan
1AG3

A close-up photograph of a pizza being cooked in a traditional wood-fired oven. The intense orange and yellow flames are visible at the top, casting a warm glow over the pizza. The pizza itself has a golden-brown crust and is topped with melted cheese and various toppings like pepperoni and onions. A metal peel is partially visible on the left, used for placing the pizza in the oven. The overall atmosphere is rustic and focused on the art of pizza making.

new Pizza()

SOMMAIRE

Informations

Idée principale	Page 3
Contexte	Page 4
Présentation & Fonctionnalités	Page 5

Personas & UserStories

1 ^{er} Persona	Page 6
2 ^{ème} Persona	Page 6
3 ^{ème} Persona	Page 7

Sketchs & StoryBoard

Parties communes	Page 8
Parties clients	Pages 9-10
Partie administrateur	Page 11
StoryBoard	Page 12

Annexes

Diagramme de cas	Page 13
Cas	Pages 14-17
Diagramme de classe	Page 18
Diagramme de séquence	Page 19
Diagramme de paquetage	Page 20
Architecture	Page 21-22

IDÉE PRINCIPALE

Permettre aux utilisateurs de commander une pizza via une application intuitive afin de ne pas perdre de temps contrairement à une commande téléphonique ou sur place.

Permettre au pizzaïolo de traiter les commandes plus facilement via une interface simple d'utilisation.

CONTEXTE

Suite à l'explosion du nombre de pizzerias en France, de nombreux propriétaires sont à la recherche d'applications permettant de traiter et de gérer leurs commandes. Pour parer au problème de notre pizzaïolo, nous avons décidé de développer une application de e-commerce.

Afin de faciliter l'achat de pizza via l'application, elle se devra d'être le plus simple possible d'utilisation et la plus intuitive possible.

new Pizza()

new Pizza() est une application regroupant une large panoplie de délicieuses pizzas, avec leur nom, leurs ingrédients et leur image ! Toutes les pizzas sont triées par catégorie afin de trouver directement celle que vous souhaitez.

Les paiements s'effectuent sur l'application via Carte bancaire et via Paypal.

En plus de pouvoir commander vos pizzas préférées, l'application est tout aussi utile pour le gérant de la pizzeria qui aura accès à panel afin de gérer toutes les commandes et d'en oublier aucune.

Parmi les fonctionnalités de l'application, on retrouve :

- ± Accès à un large choix de pizza triées par catégories
- ± Différents moyens de paiement

- ± Espace administrateur
- ± Liste des commandes
- ± Annoncer le début préparation d'une commande



PERSONAS & USERSTORIES



Anna BOUDOUL, 18 ans



Étudiante en IUT Informatique



Clermont-Ferrand, 63000

Objectifs :

- Gagner du temps sur la préparation de ses repas.
- Avoir suffisamment de temps pour manger
- Réussir ses études.

UserStory :

Anna est actuellement étudiante en informatique à Clermont-Ferrand et n'a pas toujours le temps de se faire à manger pendant ses temps de pause.

Depuis maintenant quelques semaines, son nombre d'heures de cours a augmenté et elle n'a désormais plus assez de temps pour rentrer chez elle et se faire à manger.

C'est à ce moment qu'elle a découvert new Pizza() pour résoudre son problème. En effet, new Pizza() est une application qui permet de commander une pizza à n'importe quel moment. L'application correspond à sa personnalité, elle est innovante et graphiquement agréable. Anna peut maintenant avoir le temps de manger et de suivre ses cours.

Personnalité :

C'est une étudiante brillante qui aime prendre le temps pour les petits détails de la vie, elle est énergique et motivée.



Luigi Peroni, 45 ans



Propriétaire d'une pizzeria



Clermont-Ferrand, 63000

Objectifs :

- Faire partager sa culture aux Clermontois.
- Organiser les commandes qu'il reçoit.
- Faciliter les démarches à faire pour commander.

UserStory :

Luigi Peroni est né en Italie, étant jeune, il a décidé de venir vivre en France. Luigi est très attaché à ses origines et à sa culture. Il a donc décidé d'ouvrir sa première pizzeria, grâce cette dernière il a voulu partager sa culture à travers ses créations. Étant donné que Luigi travaille seul dans sa pizzeria, il peine à trouver un moyen efficace de gérer ses commandes. Il a donc fait appel à nos services de développement et a découvert new Pizza(). L'application lui permet de prendre automatiquement les commandes et de les notifier sur une interface. Elle lui permet de traiter ses commandes de façon efficace. L'application convient parfaitement à Luigi puisqu'elle est simple d'utilisation et permet de faire une grande partie du travail à sa place.

Personnalité :

Luigi Peroni est un grand nostalgique de ses origines italiennes, il ne souhaite que des produits de qualité pour ses clients, il souhaite également que ses clients passent une agréable expérience au sein de sa pizzeria.

PERSONAS & USERSTORIES



Philippe Beckham, 30 ans



Coach sportif



Clermont-Ferrand, 63000

Objectifs :

- Gagner du temps sur la préparation de ses commandes.
- Arriver à l'heure pour les matchs.
- Réussir à faire cohabiter ses deux passions.

UserStory :

Philippe Beckham, coach sportif chez Opération minceur depuis plusieurs années, est un grand fan de sport et particulièrement de football. Philippe apprécie regarder les matchs de son équipe favorite avec ses amis en mangeant des pizzas.

Depuis peu, ses horaires ont fortement bougés et il arrive qu'il finisse très tard. Philippe ne voulant pas louper le début des matchs en attendant sa commande, il souhaite trouver un moyen efficace et rapide de commander une pizza. C'est alors qu'il a découvert new Pizza(), une application sur ordinateur qui lui permet de commander des pizzas en quelques clics. L'application est parfaite pour Philippe car il peut voir la liste des ingrédients des pizzas et faire attention à ce qu'il mange.

Personnalité :

Philippe est un passionné de sport, au point qu'il en a dédié sa vie en étant coach sportif. Il apprécie aussi passer du temps avec ses amis autour d'un match de football.

SKETCHS

The sketch shows a login screen with a placeholder image of a pizza in the top-left corner. The title "NOM PIZZERIA" is at the top. Below it is a "CONNEXION" section containing two input fields: "Identifiant" and "Mot de passe", followed by a "SE CONNECTER" button. At the bottom, there is a link "Pas encore de compte ? S'inscrire". Three yellow arrows point from the text descriptions on the right to specific elements: one arrow points to the "Mot de passe" field, another to the "S'inscrire" link, and a third to the "Pas encore de compte ?" link.

Page de connexion :

La page de connexion permet de se connecter à l'application via un identifiant et un mot de passe

Redirige vers la page d'inscription si l'utilisateur n'a pas de compte

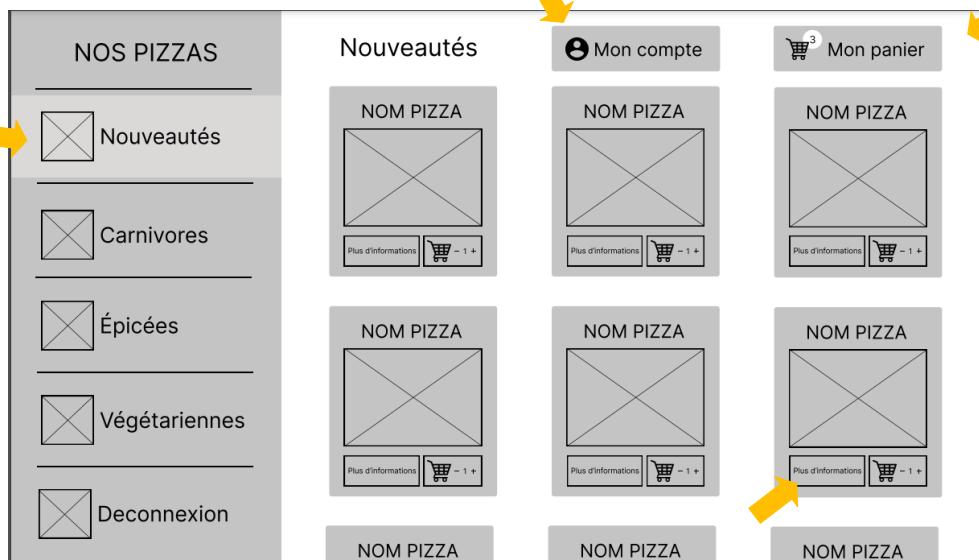
The sketch shows a sign-up screen with a placeholder image of a pizza in the top-left corner. The title "NOM PIZZERIA" is at the top. Below it is an "INSCRIPTION" section containing three input fields: "Identifiant", "E-mail", and "Mot de passe", followed by a "S'INSCRIRE" button. At the bottom, there is a link "Déjà inscrit ? Se connecter". Three yellow arrows point from the text descriptions on the right to specific elements: one arrow points to the "E-mail" field, another to the "S'INSCRIRE" button, and a third to the "Déjà inscrit ?" link.

Page d'inscription :

La page d'inscription permet de s'inscrire en renseignant un identifiant, un e-mail et un mot de passe.

Redirige vers la page de connexion si l'utilisateur a déjà un compte

SKETCHS



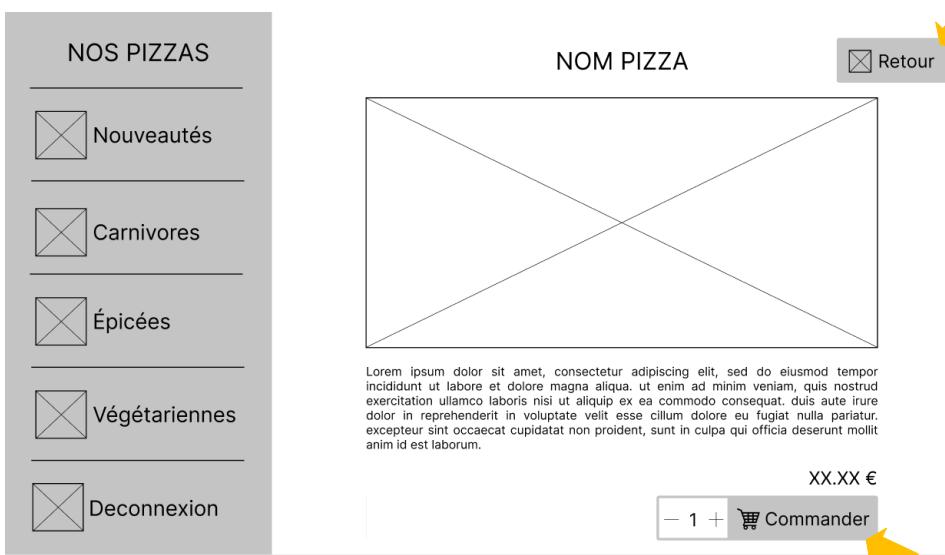
Page des catalogues :

Pouvoir de changer de catégorie via le menu à gauche

Permet d'obtenir des informations supplémentaires sur une pizza.

Permet d'ajouter des pizzas à son panier avec la quantité souhaitée.

Permet d'avoir accès à ses informations personnelles via le bouton « Mon compte » et à son panier via « Mon panier ».



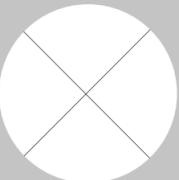
Page d'informations d'une pizza :

Permet d'obtenir des informations supplémentaires sur une pizza.

Permet d'ajouter une pizza avec la quantité souhaitée à son panier.

Permet de revenir sur la page des catalogues avec le bouton « Retour » ou via le menu.

SKETCHS

MON COMPTE	Mes informations
	NOM XXXXXXXXX PRENOM XXXXXXXXX E-MAIL: XXXXXXXXX@XXXXXXX.XXX TÉLÉPHONE XXXXXXXXXX ADRESSE XXXXXXXXXXXXXXXXXXXXXXXX VILLE XXXXXXXXXXXXXXXXXX CODE POSTAL XXXXX
Pseudo	
<input type="button" value="Retour"/>	
<input type="button" value="Deconnexion"/>	<input type="button" value="Enregistrer les modifications"/>

Page mon compte :

Permet de se déconnecter de l'application

Pouvoir revenir sur la page des catalogues via le bouton « Retour »

Permet de modifier ses informations en cliquant sur chacune d'entre elles.
Permet aussi modifier son image de profil en cliquant dessus.
Permet d'enregistrer les modifications effectuées en cliquant sur « Enregistrer les modifications ».

MON PANIER	NOM PIZZA  Plus d'informations  - 1 +	NOM PIZZA  Plus d'informations  - 1 +	PAIEMENT
Pseudo	NOM PIZZA  Plus d'informations  - 1 +	NOM PIZZA  Plus d'informations  - 1 +	PayPal <input type="button" value="Carte bancaire"/>
<input type="button" value="Retour"/>			PRIX TOTAL XX,XX €
<input type="button" value="Deconnexion"/>	NOM PIZZA	NOM PIZZA	<input type="button" value="Payer"/>

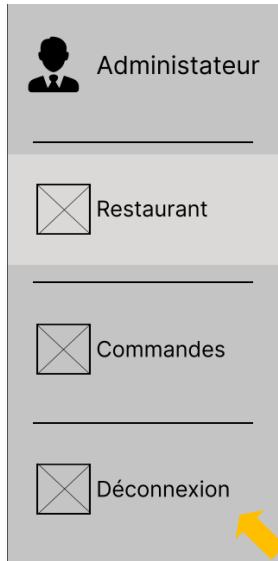
Page mon panier :

Permet de se déconnecter de l'application

Pouvoir revenir sur la page des catalogues via le bouton « Retour »

Permet de modifier la quantité d'une pizza de son panier.
Permet d'obtenir des informations supplémentaires sur une pizza avec « Plus d'informations»
Permet de choisir le moyen de paiement souhaité et de payer.

SKETCHS



Informations de l'entreprise: XXXXXXXXXXXX

NOM DU GERANT
XXXXXXXXXX

PRENOM DU GÉRANT
XXXXXXXXXX

E-MAIL
XXXXXXXXXX@XXXXXXX.XXX

TÉLÉPHONE
XXXXXXXXXX

ADRESSE
XXXXXXXXXXXXXXXXXXXXXX

VILLE
XXXXXXXXXXXXXXXXXX

CODE POSTAL
XXXXXX

Enregistrer les modifications

Permet de modifier les informations du chef en cliquant sur chacune d'entre elles.

Permet d'enregistrer les modifications effectuées en cliquant sur « Enregistrer les modifications ».

Permet de voir les commandes en cours et en attente via le menu à gauche.

Page restaurant :

Page réservée à l'administrateur du restaurant.

Permet de se déconnecter avec le bouton « Déconnexion »

The page title is "COMMANDES". It displays three order cards, each with a circular icon containing a large 'X'. The first card shows the status "En cours..." and has a "FINIR" button. The second card has a "COMMENCER" button. The third card also has a "COMMENCER" button.

Page commandes:

Page réservée à l'administrateur du restaurant.

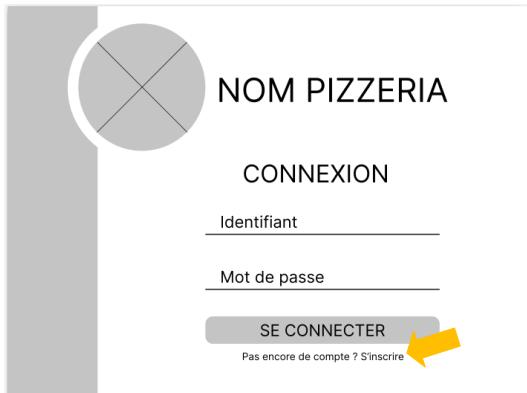
Permet de se déconnecter avec le bouton « Déconnexion »

Permet de voir les commandes en cours et celles en attentes.

Permet d'annoncer le début d'une commande via « Commencer » et d'annoncer la fin de la commande avec « Finir ».

STORYBOARD

Dans ce storyboard, nous allons retracer la création du compte d'un nouveau client et sa première commande



NOM PIZZERIA

CONNEXION

Identifiant _____

Mot de passe _____

SE CONNECTER

Pas encore de compte ? S'inscrire

Lors du lancement de l'application, l'utilisateur tombe sur la page de connexion mais n'a pas encore de compte, il doit cliquer sur « S'inscrire » pour créer son compte



NOM PIZZERIA

INSCRIPTION

Identifiant _____

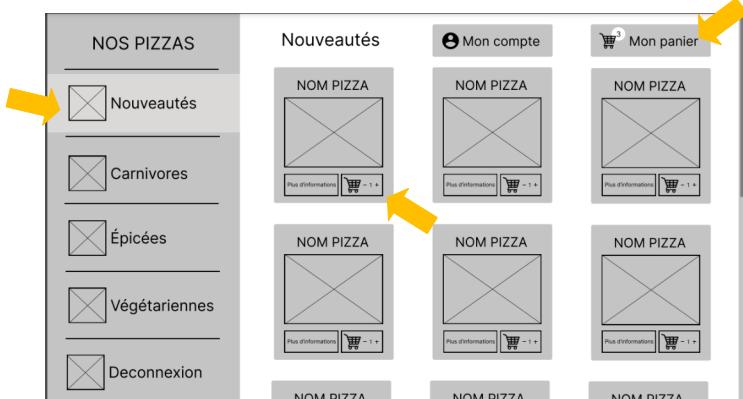
E-mail _____

Mot de passe _____

S'INSCRIRE

Déjà inscrit ? Se connecter

Arrivé sur la page d'inscription, notre utilisateur doit remplir les différents champs pour s'inscrire. Une fois ces champs remplis, il doit cliquer sur « S'inscrire » pour valider la création de son compte



NOS PIZZAS

- Nouveautés
- Carnivores
- Épicées
- Végétariennes
- Deconnexion

Nouveautés

Mon compte

Mon panier

NOM PIZZA

NOM PIZZA

NOM PIZZA

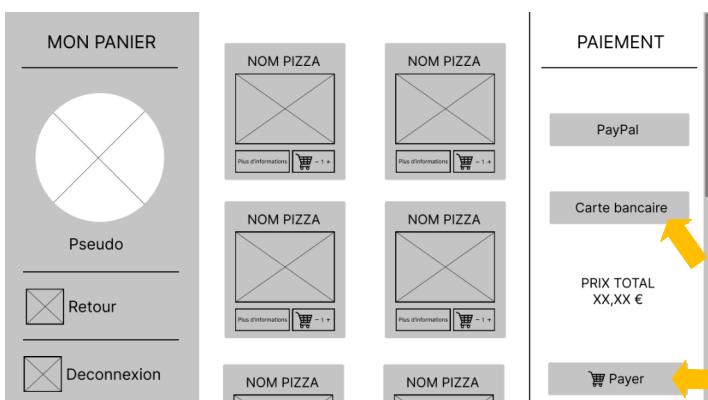
NOM PIZZA

NOM PIZZA

NOM PIZZA

Une fois connecté, l'utilisateur arrive sur le catalogue. Il peut naviguer entre les différentes catégories à l'aide du menu à gauche. Une fois la pizza souhaitée repérée, il doit saisir la quantité voulu en cliquant sur le + ou sur le -. Une fois la quantité choisi, il doit valider sa commande en cliquant sur l'icone de panier.

Une fois sa commande choisie, l'utilisateur clique sur le bouton « Mon panier » pour finaliser sa commande.



MON PANIER

Pseudo

Retour

Deconnexion

NOM PIZZA

NOM PIZZA

NOM PIZZA

NOM PIZZA

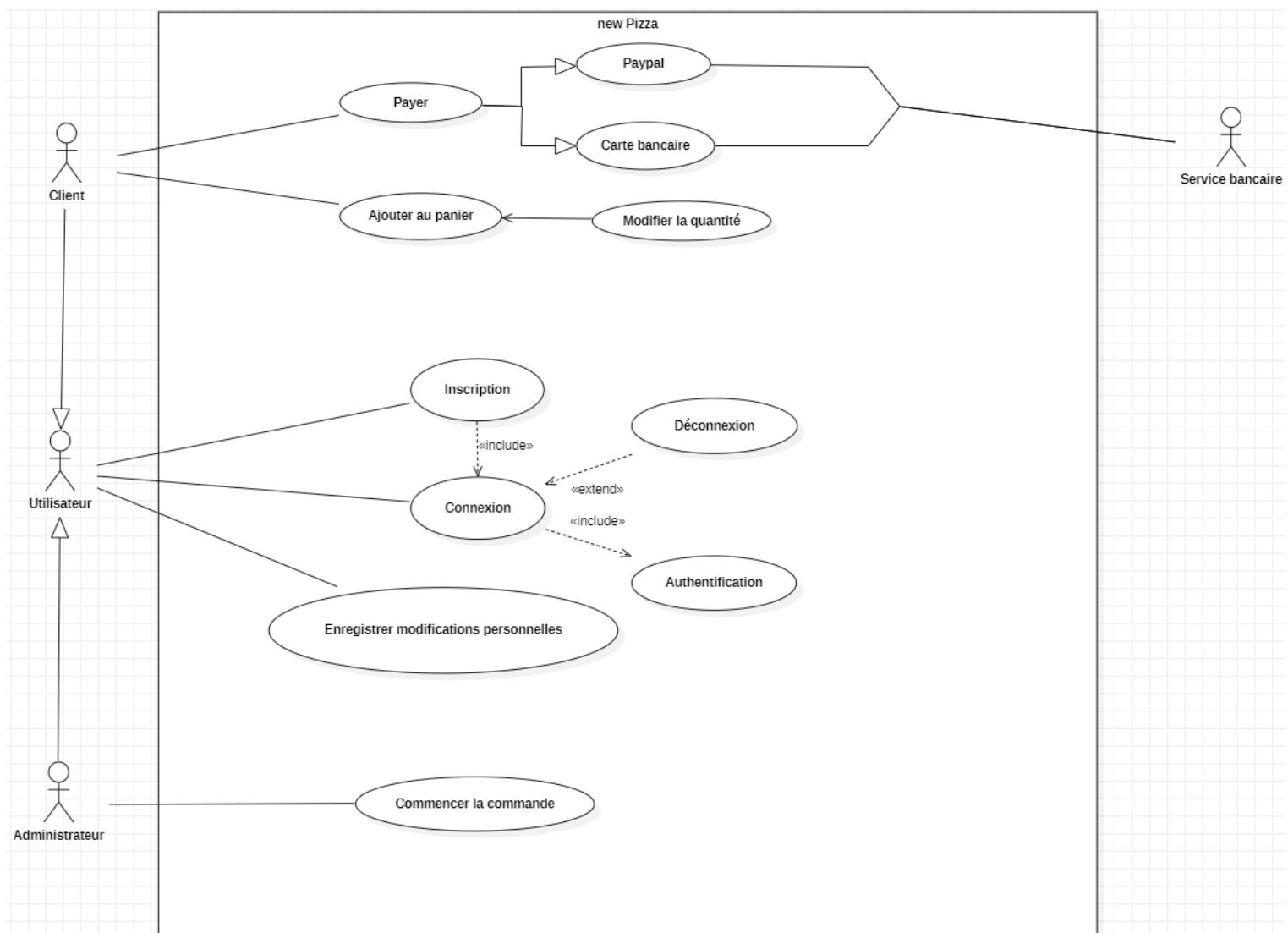
NOM PIZZA

NOM PIZZA

Arrivé sur son panier, l'utilisateur peut payer en choisissant son moyen de paiement « Paypal » ou « Carte bancaire ». Ensuite, il lui suffit de cliquer sur « Payer » pour être rediriger vers le service souhaité pour payer.

La commande est envoyé à l'administrateur qui se chargera de la préparer.

DIAGRAMME DES CAS



CAS

Cas “Connexion”:

Nom: Connexion.

Objectif: Accéder à l'application via un utilisateur .

Acteurs principaux: Membre.

Conditions initiales: Avoir un réseau internet.

Scénario d'utilisation:

- L'utilisateur doit entrer son compte et son mot de passe.
- Vérifie s' il s'agit du compte administrateur.
- Vérifie l'existence du compte client.
- Valider la connexion par le serveur.

Condition de fin:

- Affichage de l'accueil de l'utilisateur ou de l'administrateur.

Scénario alternatif:

- Refus du serveur.
- Redemande l'identifiant et le mot de passe.

Cas “Inscription”:

Nom: Inscription.

Objectif: Permettre à l'utilisateur de créer un compte.

Acteurs principaux: Membre.

Conditions initiales: Avoir un réseau internet et cliquer sur « S'inscrire ».

Scénario d'utilisation:

- L'utilisateur doit entrer son nouvel identifiant et deux fois son mot de passe.
- Vérifie l'existence du compte client.
- Valider la création par le serveur.

Condition de fin:

- Retour sur la page connexion et l'utilisateur peut se connecter.

Scénario alternatif:

- Refus du serveur.
- Redemande l'identifiant.
- Redemande le mot de passe.

CAS

Cas “Déconnexion”:

Nom: Déconnexion.

Objectif: Se déconnecter de l’application.

Acteurs principaux: Membre.

Conditions initiales: Avoir une connexion internet et être connecté à l’application.

Condition de fin:

- Appuyer sur le bouton « Déconnexion ».

Cas “Enregistrer informations personnelles”:

Nom: Enregistrer informations personnelles.

Objectif: Ajouter et modifier ses informations personnelles.

Acteurs principaux: Membre

Conditions initiales: Avoir une connexion internet, être connecté à son compte et être sur la page « Mon compte ».

Scénario d’utilisation:

- L’utilisateur peut ajouter, modifier ou supprimer diverses informations personnelles.
- Vérification de la présence du @ pour l’adresse mail.
- Vérification que le code postal contient 5 chiffres et qu’il existe.
- L’utilisateur valide en cliquant sur le bouton « Enregistrer »

Condition de fin:

- Sauvegarde des informations.

Scénario alternatif:

- Affiche un message d’erreur et ne sauvegarde pas.

CAS

Cas “Modifier ingrédient d'une pizza”:

Nom: Modifier les ingrédients d'une pizza.

Objectif: Choisir une pizza et ajouter des ingrédients.

Acteurs principaux: Client.

Conditions initiales: Avoir une connexion internet, être connecté à son compte client.

Scénario d'utilisation:

- Choisir une des pizzas disponibles sur la page principale.
- Cliquer sur plus d'informations.
- Personnaliser la pizza.

Condition de fin:

- Les conditions seront finies une fois le cas “Ajouter un certain nombre de pizza au panier” sera rempli.

Cas “Ajoutez un certain nombre de pizza au panier”:

Nom: Ajoutez un certain nombre de pizzas au panier.

Objectif: Ajouter X fois une pizza au panier.

Acteurs principaux: Client.

Conditions initiales: Avoir une connexion internet, être connecté à son compte client.

Scénario d'utilisation:

- Choisir une des pizzas disponible sur la page principale.
- Vous pouvez faire, le cas “Modifier les ingrédients d'une pizza” puis sélectionner le nombre de fois que vous souhaitez votre pizza et cliquer sur commander.
- Ou bien, sur la page principale, sélectionner le nombre de fois que vous souhaitez votre pizza et cliquer sur le caddie.

Condition de fin:

- Affichage de la page d'accueil utilisateur.

CAS

Cas “Gestion du paiement”:

Nom: Gestion du paiement.

Objectif: Client paye la pizzeria pour la réalisation de sa commande.

Acteurs principaux: Client.

Conditions initiales: Avoir une connexion internet, être connecté à son compte client et avoir des pizzas dans son panier.

Scénario d'utilisation:

- Depuis la page principale cliquer sur « Mon panier ».
- Une fois dessus vous pouvez vérifier que toutes les pizzas souhaitées sont bien présentes.
- Une fois cela fait vous avez à droite le prix total de la commande, il vous suffit de sélectionner votre mode de paiement (Paypal ou carte bancaire).

Condition de fin:

- Cliquer sur le bouton paiement qui vous redirigera sur un site qui s'occupera de la transaction.

Cas “Accepter commande”:

Nom: Accepter commande.

Objectif: Valider la commande reçue.

Acteurs principaux: Administrateur.

Conditions initiales: Avoir une connexion internet, être connecté au compte administrateur.

Scénario d'utilisation:

- La pizzeria peut voir toutes les commandes en attente.

Condition de fin:

- Appuie sur le bouton valider: annonce le début de la préparation.

Cas “Supprimer commande”:

Nom: Supprimer commande

Objectif: Supprimer une commande validée.

Acteurs principaux: Administrateur.

Conditions initiales: Avoir une connexion internet, être connecté au compte administrateur et avoir une commande validée.

Scénario d'utilisation:

- La pizzeria peut voir toutes les commandes en attente et celles qui ont été validées.

Condition de fin:

- Appuie sur le bouton finir la commande.

DIAGRAMME DE CLASSE

Vous pourrez trouver le diagramme de classe dans le « DiagrammeDeClasse.svg » car l'image est trop grande pour être mise sur un document pdf et serait donc illisible..

Dans ce diagramme, nous utiliserons la notations
-/Ppté pour désigner une propriété Ppté avec un
public get et un private set. Par exemple : -/Ppté =
{ +get; -set}

Le but de notre application est de donner la possibilité à n'importe quel client de commander une ou plusieurs pizzas et en parallèle à un administrateur de gérer les commandes que reçoit sa pizzeria, la classe principale est donc Utilisateur : elle est abstraite et est composée d'un nom, d'un prénom, d'une adresse email, d'un numéro de téléphone, d'une adresse, d'un mot de passe, d'un nom de ville et d'un code postal.

Les classes Clients et Administrateur héritent de la classe abstraite Utilisateur.

La classe Client est composée d'un pseudo et d'une photo et pour la classe Administrateur nous avons en plus un nom de pizzeria.

L'héritage permet de factoriser les lignes de code de ces classes comme par exemple les méthodes pour modifier le mot de passe d'un utilisateur (ModifierMdp()), enregistrer les modifications sur les informations personnelles d'un utilisateur (enregistrerModif() return un Utilisateurs) et les Equals et GetHashCode.

La classe Commande a une liste de pizza, un Client et un Statut qui est une classe de type énumération et qui contient 3 statuts différents (Commencer, en cours et fini).

Elle permet donc d'ajouter la commande d'un client à la liste total des commandes d'un administrateur, elle contient une fonction permettant de changer ce statut.

Par conséquent, la classe Client possède des méthodes permettant d'ajouter une pizza à sa commande (ajouterPizzaCommande(p:Pizza)), une qui permet d'en enlever (supprimerPizzaCommande(p:Pizza)) et une qui permet d'envoyer cette liste (son panier) à la pizzeria (envoyerListeCommande(admin: Administrateur)).

L'administrateur quant à lui possède seulement deux méthodes, une qui fait appel à la méthode présente dans la classe Commande permettant de changer le statut d'une commande (ChangerStatusCommande(C1:Commande)) et une autre qui permet de supprimer une commande (SuppCommande(C1:Commande)).

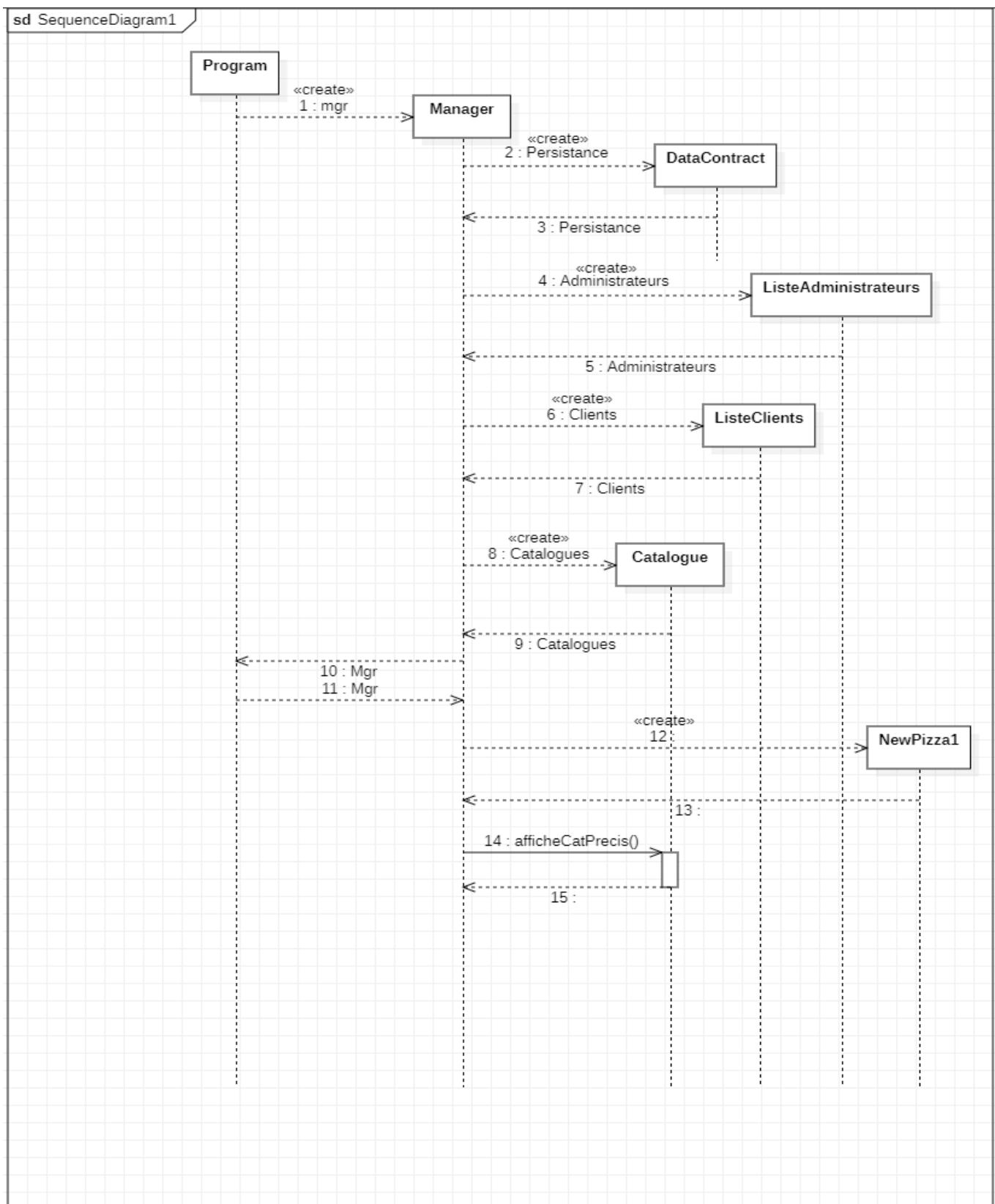
En parallèle nous avons une classe Pizza qui avec 5 attributs dont un nom, une quantité, une description, une image et un prix, elle contient une méthode modifQte(qte:int) qui permet de modifier la quantité de pizza souhaitée et de retourner cette quantité.

De plus cette classe reçoit une liste d'ingrédients de la classe de type énumération Ingrédient.

Toutes ces pizzas sont rangées dans différentes listes de la classe Catalogue, elle en contient 4 comme catalogueVegetarien, cataloguePizzeta, catalogueCarnivore, catalogueEpicee.

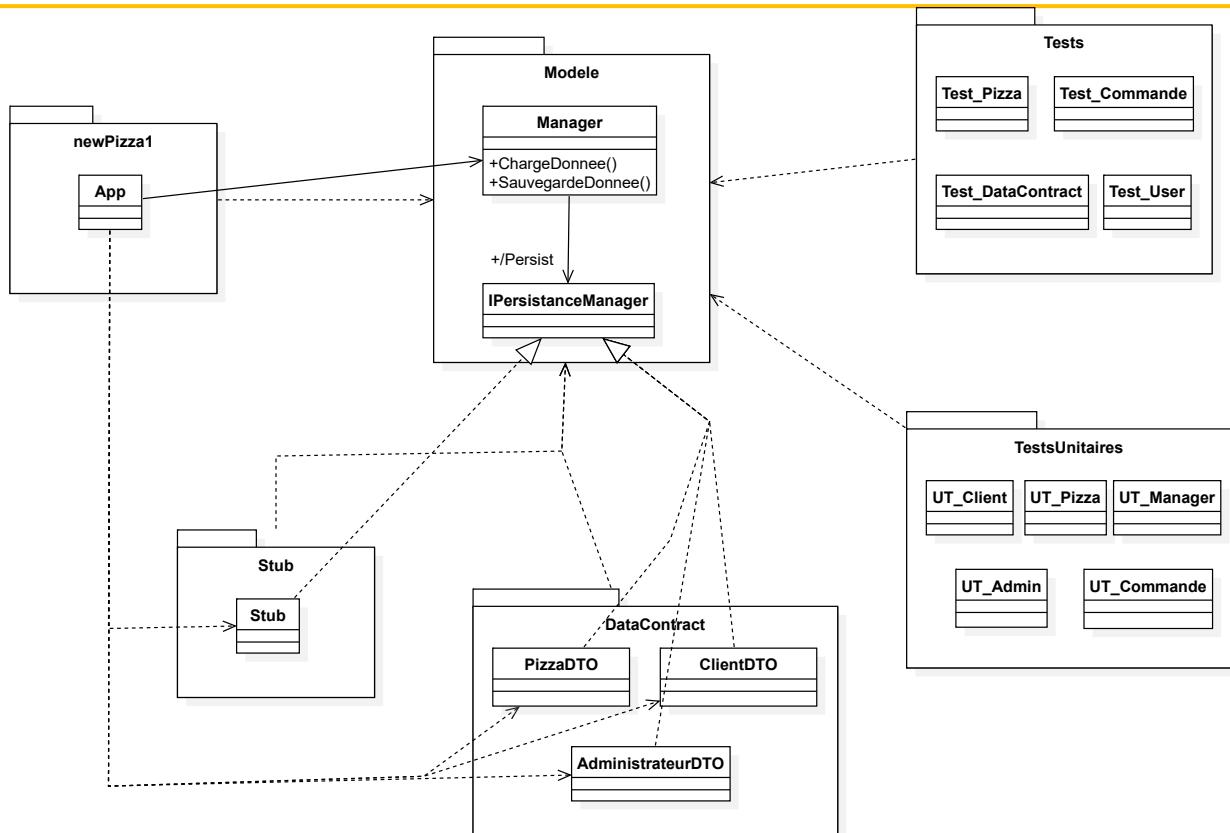
Pour finir le manager possède une liste de Clients, une liste d'administrateur . Il possède aussi une variable catalogue.

DIAGRAMME DE SÉQUENCE



Dans un premier temps le programme va instancier un Manager qui se nommera Mgr, ce dernier instancie un Data contract, une ListeAdministreurs nommée Administrateurs, une ListeClients nommée Clients et un Catalogue nommée Catalogues. Ensuite, l'adresse du mgr ira dans le program, puis le manager instanciera la page NewPizza1 mais également affichera tout les catalogues disponible dans l'app.

DIAGRAMME DE PAQUETAGE



Le package **newPizza1** est l'ensemble de toutes nos vues et de l'application **App** qui gère l'ensemble des vues. Le package **newPizza1** possède une dépendance avec **Modele**, cette dépendance fait le lien entre le code XAML et les classes en C# pour permettre la gestion des actions de l'utilisateur (navigation, clics, remplissage, etc...) et la liaison de données (Data Binding).

Pour avoir du contenu sur l'application, nous passons par le **Manager** qui est une classe qui va instancier des **ReadOnlyCollection** de clients, un d'administrateur et une autre sera pour la liste de catalogue de pizza qui est une encapsulation d'une liste de clients(clients), d'administrateur(administrateur) et une liste d'utilisateur(utilisateurs) permettant de limiter les actions sur cette liste. Le manager possède une méthode **ChargeDonnées**, comme son nom l'indique elle va charger les données de l'application sur nos vues par le biais du **DataBinding**. Les données après sont disponibles sur les vues grâce à **App** qui instancie le manager. Les données seront sauvegardées à la fermeture de l'application via la méthode **SauvegardeDonnées()**. Il reste plus qu'à faire le lien entre chaque fenêtre qui a besoin des données et du manager, pour cela on passe par **App** qui instancie déjà un manager et on fait pointer chaque vue grâce à son code behind sur le manager de **App** grâce à une propriété calculée.

Ensuite il y a deux autres dépendances, une de **Test** à **Modele** et une autre de **TestUnitaires** à **Modele**. Ces deux packages permettent de tester toutes les fonctions présentes dans **Modele** comme les affichages, les méthodes, les constructeurs, les propriétés... de chaque classe.

App possède deux dépendances qui elles dépendent de **Modele**, une vers le **Stub** qui va permettre de stocker en dur les données de l'appli pour faire des tests sur la persistance. **Stub** va implémenter l'interface **IPersistenceManager** de **Modele** qui possède deux méthodes **SauvegardeDonnées** et **ChargeDonnées** pour permettre à l'application de garder les données modifiées après avoir quitté l'application. La deuxième dépendance est avec le package **DataContract** qui lui au contraire de **Stub** va enregistrer les données dans un fichier et charger l'application depuis ce fichier.

ARCHITECTURE

Notre application possède plusieurs packages pour permettre de différencier chaque partie de l'application et faciliter la navigation et la programmation du projet.

Pour commencer on a le package newPizza1, il rassemble toutes les vues de l'application et App qui gère la vue de démarrage et permet de faire des modifications sur toutes les vues. newPizza1 dépendent de Modele car lors du databinding les vues ont besoin d'être remplies par du contenu qui est géré par les classes du package Modele et plus précisément le Manager. L'appli et le modèle sont séparés pour permettre de créer une nouvelle appli avec le même Modèle.

Modele est le package où vont se situer les classes, les interfaces, les collections... Modele est constitué de la classe abstrait Utilisateur qui est définie par un nom, prénom, email, téléphone, adresse, ville, codePostal. Les classe Administrateur et Client vont tout d'eux hériter de la classe Utilisateur, ce qui leur permet d'avoir également un nom, prénom, email, téléphone, adresse, ville, codePostal. En ce qui concerne le Client il aura aussi également un pseudo et une photo lors de sa création mais également cette classe contient une listPizzaClient qui correspondra au panier de chaque client alors qu'un manager aura quant à lui un nomDePizzeria sera précisé lors de sa création et auras une listeCommande qui est une classe recevant un client et une liste de pizza en paramètre, cette dernière a également un statuts de type enum que l'administrateur pourras changer lors de l'avancer de la commande.

Nous avons aussi en parallèle une classe enum composer d'ingrédient qui servira à la classe Pizza via une liste. La classe pizza est quant à elle construite avec un nom, une description, image, une liste d'ingrédient et un prix, elle aura également besoin par la suite d'une quantité ce qui permettra aux clients de précisé le nombre de fois qu'ils souhaitent avoir cette pizza.

Par la suite le manager pourra facilement constituer 4 listes via catalogue destiné à contenir des pizzas.

De plus, le manager gère l'identification des personnes car un utilisateur possède aussi un identifiant (Pseudo, nomPizzeria ou Email) et un mot de passe permettant de se connecter soit aux catalogues des pizza pour un client ou à la page admin pour un administrateur. La plupart des listes sont en ObservableCollection permettant le INotifyPropertyChanged sur ces dernières et donc d'avoir un meilleur rendu sur nos vues concernant car permettent la mise à jour de nos listes en cas de changement des valeurs comme pour la listPizzaClient, la listeCommande mais également les 4 Catalogues présent dans la classe Catalogues.

ARCHITECTURE

Le package Modele possède une classe Manager qui possède une liste de d'Administrateur et de Client qui sont encapsulés ReadOnlyCollection pour limiter les actions sur ces listes mais aussi 4 Catalogue présent dans une variable de type Catalogue. Manager possède une propriété Persistance de type IPersistanceManager. Le Manager possède un UtilisateurActuel, une PizzaActuelle et un CommandeActuelle, permettant de gérer l'élément de la collection sélectionnée. Le Manager permet de remplir les vues avec du contenu grâce à ChargeDonnees.

Notre persistance, implémente l'interface IPersistanceManager se situant dans le package Modele pour permettre au Manager d'être instancié avec un IPersistanceManager en paramètre, c'est-à- dire, une classe qui possède une méthode Charge et une méthode Sauvegarde. Pour le moment l'application a une persistance xml grâce àDataContract. Ce dernier dépend de Modele car les classes implémentent IPersistanceManager, elles doivent connaître les collections à charger et sauvegarder. Les données sont chargées et sauvegardées grâce aux méthodes Charge et Sauvegarde dans le Manager qui va appeler Persistance.ChargeDonnées et Persistance.SauvegardeDonnées Ces deux méthodes utiliserons les liste Clients, Administrateurs et la variable C1 pour charger et sauvegarder les données. Le Stub est une classe qui simule une persistance, elle stocke des données et à travers du code simule un effet de persistance. Il implémente aussi l'interface IPersistanceManager et dépend du package Modele pour avoir accès aux classes et à l'interface.

Test et TestUnitaires sont séparés de l'appli et du modele lors d'un rendu éventuel à un client, ces deux parties de son pas nécessaire, elles ne feront que surcharger le projet. Comme leur nom l'identique, elles font des tests sur les classes de modele donc on besoin d'une dépendance à modele. Test permet de tester globalement les classes du projet alors que TestUnitaire testent précisément une propriété, une méthode d'une collection.