Polytech Nantes
INFO 3 - Groupe 4A

Mini Projet Conception IHM - Web

Compte rendu

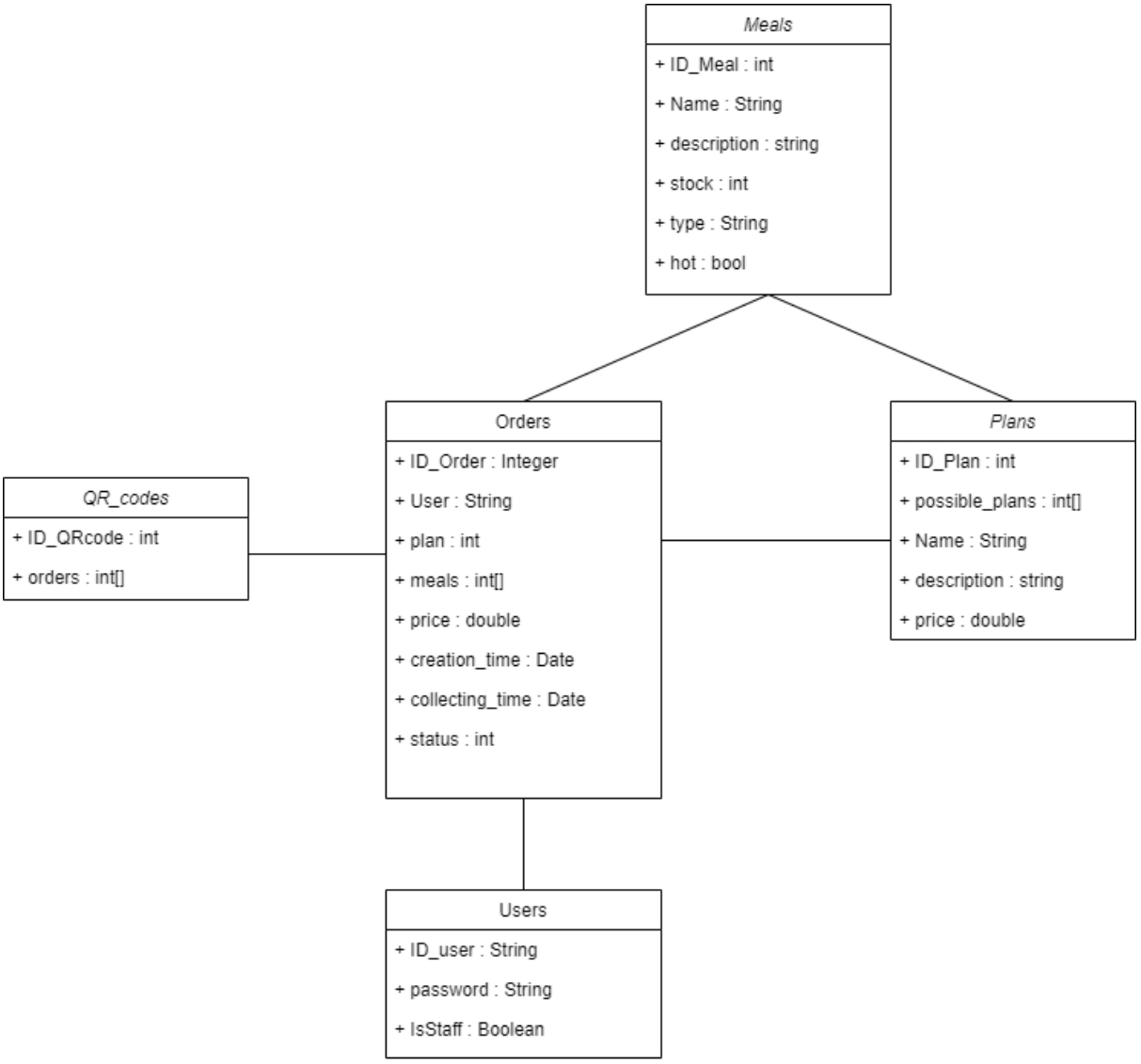
Sujet 2 : Commande de plats à emporter à la cafet' Polytech

Erwan Bouvron - Pierre Doumenc - Nawfal Benhadda

Tutrice: Marie-Pierre Nachouki

Sujet 2 : Commande de plats à emporter à la cafet' Polytech

1. Schéma base de données



Tout d'abord, les utilisateurs auront un identifiant (l'email universitaire), ainsi qu'un mot de passe pour se connecter à l'application. Il y a donc une table *Users*, qui stocke l'ensemble des utilisateurs de l'application. Ensuite, une commande est identifiée par un ID, elle est liée à l'utilisateur avec l'id (email) de celui-ci, elle comporte l'id de la formule, ainsi que l'id des plats choisis. La table *Orders* répertorie l'ensemble des commandes du système. La table *Plans* contient l'ensemble des formules proposé par la cafet, avec un identifiant, une description, un prix et une liste de plats associés à cette formule.
La table *Meals* contient l'ensemble des plats possibles, avec le stock disponible, une description, et un booléen qui permet de savoir si c'est un plat chaud ou froid.
Une table *QR_codes* va stocker l'ensemble des QR codes générées par les utilisateurs, avec une liste de commandes associées, car pour un QR codes, il peut y avoir plusieurs commandes associées.

2. Fonctionnalités principales

Nous avons déterminé 4 fonctionnalités principales indispensables au fonctionnement du système. Celles-ci sont : Création d'un compte, Connexion au compte, Formulaire de commande, Génération du QR code.

Création Compte

Nom: Création compte
Acteur(s): Utilisateur, BDD
Description: L'utilisateur crée un compte, et la base de données sauvegarde les informations

ID	Action Utilisateur	Retour système
Scénario Nominal		
1	L'utilisateur clique sur le bouton "créer un compte"	Affichage du formulaire de création de compte, champs email et mot de passe
2	L'utilisateur rentre son email et son mot de passe	/
3	L'utilisateur appuie sur le bouton "valider	Affichage du menu principal (page avec l'historique des commandes)
Scénarios alternatifs		
3.a	L'email entré n'est pas au bon format	Affichage du formulaire de création de compte, avec l'erreur en rouge
3.b	L'email est déjà associé à un compte	Affichage du formulaire de création de compte, avec l'erreur en rouge

Tests d'acceptations:

1. L'utilisateur crée un compte avec une bonne adresse email (scénario nominal), est-ce que le compte a bien été créé ?
2. L'utilisateur crée un compte avec une fausse adresse email, par exemple fausse.universitaire@gmail.com (scénario alternatif 3.a), est-ce qu'une erreur est bien affichée ?
3. L'utilisateur a déjà un compte, et recrée un compte avec la même adresse email (scénario alternatif 3.b), est-ce qu'une erreur est bien affichée ?

Connexion

Nom: Connexion
Acteur(s): Utilisateur, BDD
Description: L'utilisateur se connecte à son compte

ID	Action Utilisateur	Retour système
1	Scénario Nominal	
2	L'utilisateur ouvre l'application	Affichage du formulaire de connexion avec les champs email et mot de passe
3	L'utilisateur rentre ses identifiants et clique sur "se connecter"	Affichage de la page "Mes commandes"
	Scénario alternatif	
3.a	Un des champs entré est invalide (email ou mot de passe)	Identifiants inconnus, affichage de la page du formulaire de connexion

Tests d'acceptations:

1. L'utilisateur ouvre l'application, se connecte (scénario nominal), a-t-il accès à ces commandes ?
2. L'utilisateur rentre un mauvais mot de passe (scénario alternatif 3.a), est-ce qu'une erreur est bien affichée ?
3. L'utilisateur rentre une mauvaise adresse email (scénario alternatif 3.a), est-ce qu'une erreur est bien affichée ?

Etablir une commande

Nom: Etablir une commande
Acteur(s): Utilisateur, Staff Cafet
Description: L'utilisateur génère son QR code et récupère sa/ses commande(s)

ID	Action Utilisateur	Retour système
Scénario Nominal		
1	L'utilisateur se connecte (voir cas ***Connexion***)	Affichage du menu principal
2	L'utilisateur clique sur "Commander"	Affichage de la page "sélection des formules"
3	L'utilisateur choisit une formule et clique sur "Sélectionner"	Affichage de la page "Sélection des plats" liée à la formule choisi
4	L'utilisateur coche les plats souhaités	Affiche la pastille de sélection sur les plats en question
5	L'utilisateur clique sur "ajouter au panier"	Affichage d'un message de confirmation de l'ajout de la formule au panier et demande à l'utilisateur s'il veut continuer sur achats
6	L'utilisateur ne veut pas continuer ses achats et refuse la demande	Affichage du panier
7	L'utilisateur clique sur valider pour confirmer la commande	Affichage du menu principal, l'utilisateur voit sa commande avec le statut "en cours"
Scénarios alternatifs		
5.a	Aucun plat n'a été sélectionné dans au moins une catégorie de plat	Affichage du message "Veuillez sélectionner un plat"
6.a	L'utilisateur veut continuer ses achats et valide la demande	Retour à l'étape 2

<u>Tests d'acceptations:</u>

1. L'utilisateur ouvre l'application, se connecte, et crée une commande avec une formule, des plats, ajoute au panier, valide (scénario nominal), l'utilisateur peut-il voir sa commande ?
2. L'utilisateur choisit une formule, et valide (scénario alternatif 5.a), est-ce qu'une erreur demandant de sélectionner un plat est affichée ?
3. L'utilisateur crée une formule, puis à l'étape 6 continue ses achats (scénario alternatif 6.a), l'utilisateur peut-il ajouter d'autres formules à son panier ?

Récupérer une commande

Nom: Récupérer une commande
Acteur(s): Utilisateur, Staff Cafet
Description: L'utilisateur génère son QR code et récupère sa/ses commande(s)

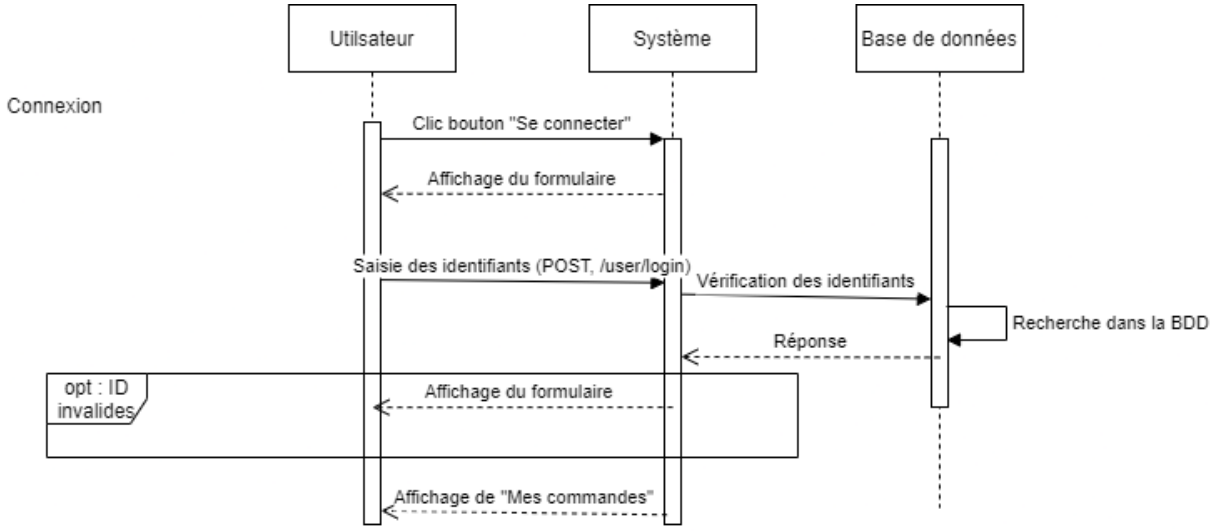
ID	Action Utilisateur	Retour système
Scénario Nominal		
1	L'utilisateur ouvre l'application et se connecte (voir cas ***Connexion***)	Affichage du menu principal
2	L'utilisateur clique sur "Générer QR Code"	Génération du QR Code et affichage de celui-ci
3	L'utilisateur montre le QR Code à un staff de la cafet, qui le scanne	Retour au menu principal, le statut est "payé" pour la/les commande(s) récupérée(s)
Scénarios alternatifs		
2.a	Il n'y a aucune commande avec le statut "Prêt"	Retour sur le menu principal
2.b	L'utilisateur clique sur "détail" pour vérifier les informations de la commande	Affichage du détail de la commande, avec un bouton pour revenir au menu principal

Tests d'acceptations:

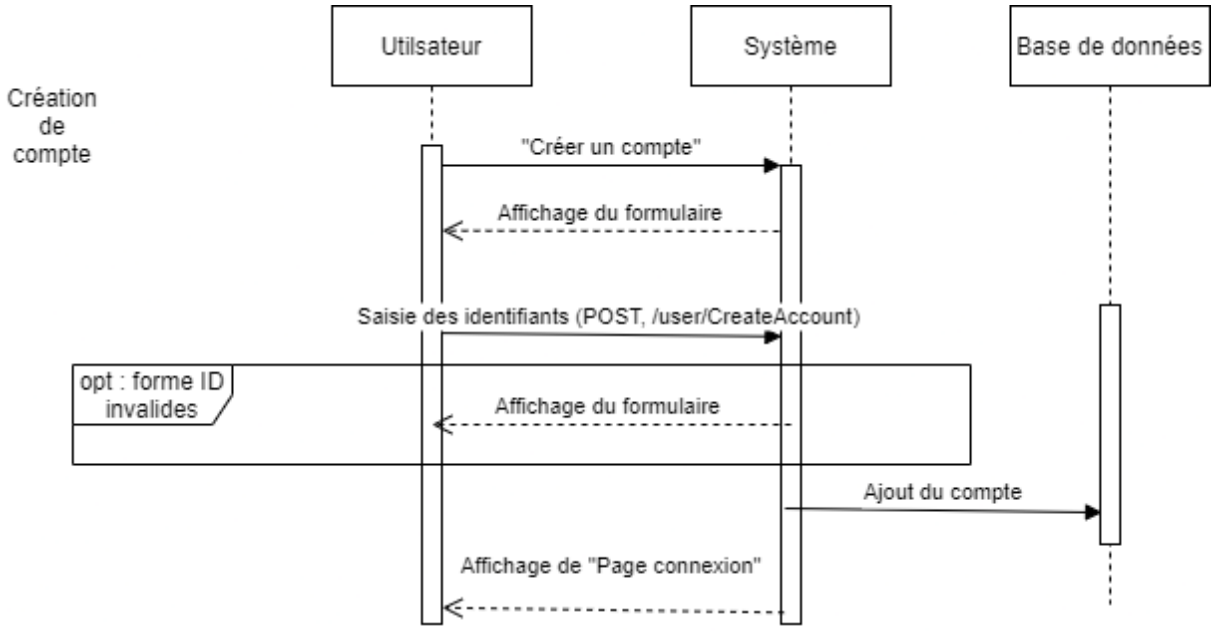
1. L'utilisateur ouvre l'application, se connecte, et génère un QR (scénario nominal), a-t-on un QR code affiché à l'écran ?
2. L'utilisateur n'a aucune commande de prête (scénario alternatif 2.a), est-ce qu'on est redirigée sur le menu principal de l'application ?
3. L'utilisateur clique sur "détail" (scénario alternatif 2.b), a-t-on les détails de la commande affichés ?

3. Diagrammes de séquence UML

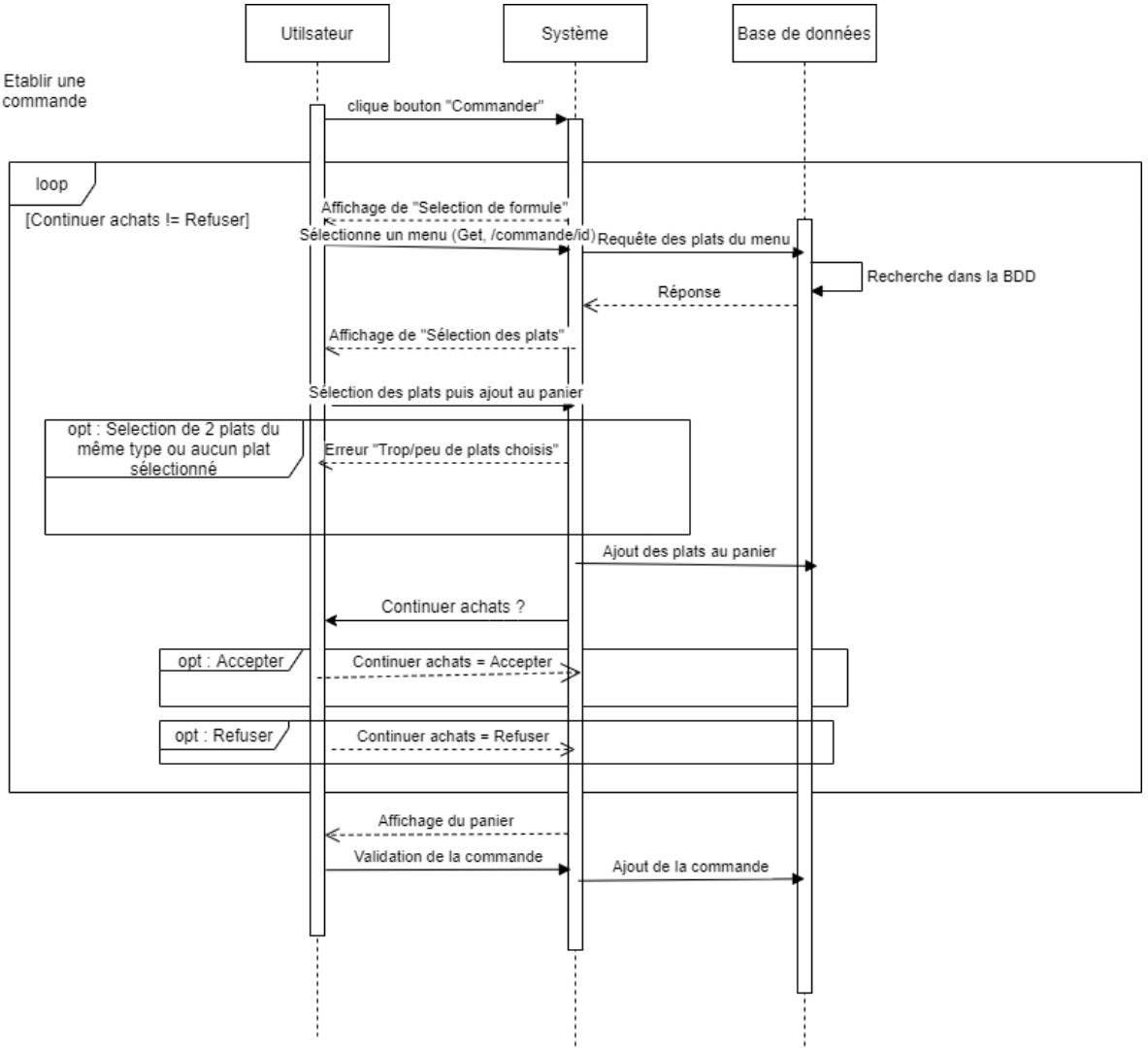
Connexion



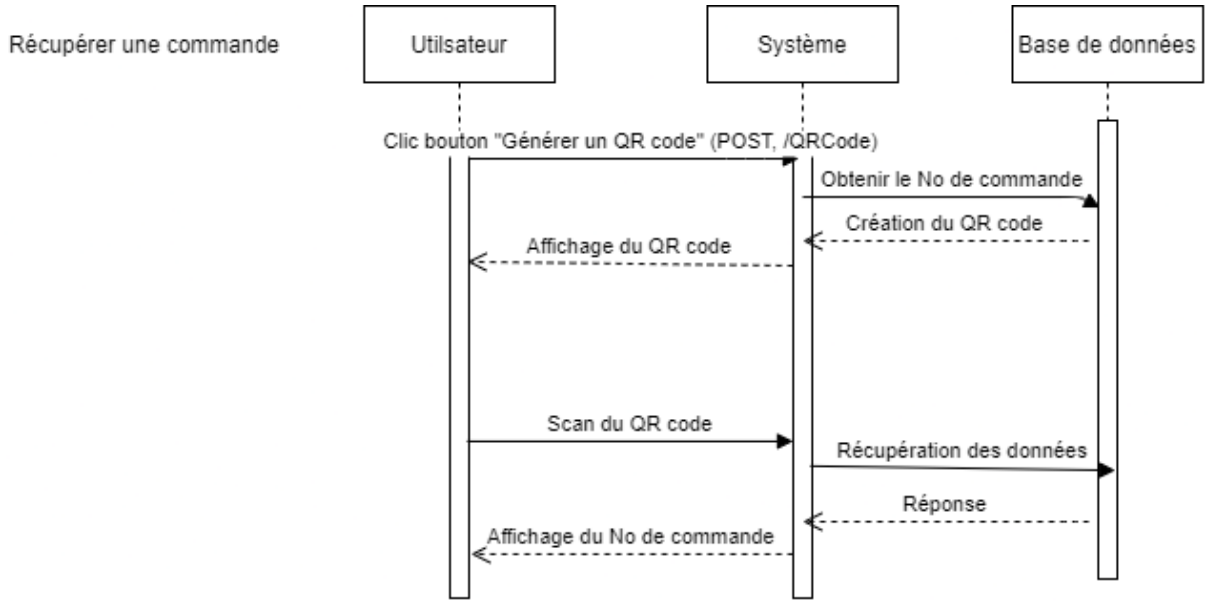
Création de compte



Commander



Récupérer une commande



4. Définition des différentes routes

Pour l'application nous avons besoin de 5 routes, qui seront détaillées dans la partie 4 avec l'api RESTful.
Pour l'authentification, la route /user sera créé, afin de créer un compte et se connecter. Cette route va aussi servir à récupérer les informations de l'utilisateur avec un GET (les commandes par exemple).
Pour la création et la consultation des commandes, une route /order est utile, avec des requêtes POST et GET.

Une route /QRCode doit être mise en place afin de générer un qrcode en lien avec une ou plusieurs commande
Enfin deux routes similaires, /plan et /meal, qui vont permettre au staff d'ajouter, modifier, ou supprimer des formules/plats, et aux utilisateurs d'accéder aux informations de ceux-ci.

5. Détail API RESTful

I. Route *user*

La route sur le serveur local est : *http://localhost:3000/api/user*

Type requête	URL	Paramètres	Requête SQL	Réponse en cas de succès	Réponse en cas d'échec	Utilité
POST	/api/user/login	Objet JSON: - email - password	`select * from Users where ID_User=email and password=password ;`	Code 200, { "success": true }	Code 1XX, { "success": false, "error": error } 110 : Mauvais email 120 : Mauvais mot de passe 130 : Autre	Permettre à l'utilisateur de se connecter à son compte
POST	/api/user/createAccount	Objet JSON: - email - password	`insert into Users values(email, password);`	Code 200, { "success": true }	Code 1XX, { "success": false, "error": error } 110 : Forme email incorrecte 120 : Forme mdp incorrecte 130 : Autre	Permettre à l'utilisateur de créer son compte
GET	/api/user/:email	email de l'utilisateur	`SELECT plans, meals, price, creation_time, collecting_time, status FROM Plans, Users WHERE Plans.user=Users.ID_user`	Code 200, { "success": true, commandes: [...] } commandes correspond à un tableau de toutes les commandes de l'utilisateur (statuts, détails des plats, heures, ...)	Code 1XX, { "success": false, "error": error } 110 : Utilisateur non trouvé 120 : utilisateur non connecté 130 : Autre	Permettre à l'utilisateur d'avoir toutes ses commandes

II. Route *order*

La route sur le serveur local est : *http://localhost:3000/api/order*

Type requête	URL	Paramètres	Requête SQL	Réponse en cas de succès	Réponse en cas d'échec	Utilité
POST	/api/order	Objet JSON: -ID_order -user -plan -meals -price -creation_time -collecting_time -status	`INSERT INTO Orders VALUES (ID_order, user, plan, meals, price, creation_time, collecting_time, status)`	Code 200, { "success": true, idOrder: …. } idOrder est l'id de la commande créée	Code 1XX, { "success": false, "error": error } 110 : plat indisponible 120 : erreur avec la formule 130 : Autre	Permettre à l'utilisateur de passer une commande
GET	/api/order/:id	ID de la commande	`SELECT * FROM Orders WHERE ID_order=ID`	Code 200, { "success": true, … } Toutes les infos sur la commande (statuts, détails des plats, heures, …)	Code 1XX, { "success": false, "error": error } 110: commande non trouvée 130 : Autre	Permettre à l'utilisateur d'avoir toutes les infos sur une commande

III. Route *QRCode*

La route sur le serveur local est : *http://localhost:3000/api/QRCode*

Type requête	URL	Paramètres	Requête SQL	Réponse en cas de succès	Réponse en cas d'échec	Utilité
POST	/api/qrcode	Objet JSON: - ID du QR code - Tableau des id de commande	`INSERT INTO QR_codes VALUES(ID_QRcode, ID_plan1, ID_plan2...)`	Code 200, { "success": true, ID_qrcode: ... }	Code 1XX, { "success": false, "error": error } 120 : Forme ID invalide 130 : Erreur rencontrée	Permettre de générer son QRCode pour retirer une ou plusieurs commande
GET	/api/qrcode/:id	ID du QRCode	`SELECT ID_QRcode FROM QR_codes WHERE ID_QRcode=ID`	Code 200, { "success": true, id_plans: ... } id_plans correspond aux identifiants des commandes correspondant au qr code	Code 1XX, { "success": false, "error": error } 120 : QR code non trouvé 130 : Autre	Permettre au staff de la cafet d'avoir les informations liées au QRCode

IV. Route *plan*

La route sur le serveur local est : *http://localhost:3000/api/plan*

Type requête	URL	Paramètres	Requête SQL	Réponse en cas de succès	Réponse en cas d'échec	Utilité
POST	/api/plan	Objet JSON: -ID_plan -meal -name -description -price	`INSERT INTO Plans VALUES (ID_plan, meal, name, description, price)`	Code 200, { "success": true }	Code 1XX, { "success": false, "error": error } 120 : Forme champ 1 invalide 130 : Forme champ 2 invalide … 170 : Autre	Permettre au staff d'ajouter une formule
DELETE	/api/plan/:id	ID de la formule	`DELETE FROM Plans WHERE ID_plan=ID`	Code 200, { "success": true }	Code 1XX, { "success": false, "error": error } 120 : Formule non trouvée 130 : Autre	Permettre au staff de supprimer une formule
PUT	/api/plan/:id	Objet formule mis à jour	`UPDATE Plans SET name=nouveau_nom, description=nouvelle_desc... WHERE ID_plan=ID`	Code 200, { "success": true }	Code 1XX, { "success": false, "error": error } 120 : Formule non trouvée 130 : Forme nom invalide 140 : Forme description invalide 150 : Autre	Permettre au staff d'éditer une formule
GET	/api/plan/:id	ID de la formule	`SELECT * FROM Plans WHERE ID_plan=ID`	Code 200, { "success": true } + toutes les infos de la formule	Code 1XX, { "success": false, "error": error } 120 : Formule non trouvée 130 : Autre	Permettre à l'utilisateur d'avoir le détail de la formule

V. Route *meal*

La route sur le serveur local est : *http://localhost:3000/api/meal*

Type requête	URL	Paramètres	Requête SQL	Réponse en cas de succès	Réponse en cas d'échec	Utilité
POST	/api/meal	Objet JSON: -ID_meal -name -description -stock -type -hot (true ou false)	`INSERT INTO Meals VALUES(ID_meal, name, description,stock,type,hot)`	Code 200, { "success": true }	Code 1XX, { "success": false, "error": error } 120 : Forme champ 1 invalide 130 : Forme champ 2 invalide … 170 : Autre	Permettre au staff d'ajouter un plat
DELETE	/api/meal/:id	ID du plat	`DELETE FROM Meals WHERE ID_meal=ID`	Code 200, { "success": true }	Code 1XX, { "success": false, "error": error } 120 : Plat non trouvé 130 : Autre	Permettre au staff de supprimer une formule
PUT	/api/meal/:id	Objet plat, et objet plat mis à jour	`UPDATE Meals SET Nom=nouveau_nom , description=nouvelle_desc... WHERE ID_meal=ID`	Code 200, { "success": true }	Code 1XX, { "success": false, "error": error } 120 : Plat non trouvé 130 : Forme nom invalide 140 : Forme description invalide 150 : Autre	Permettre au staff d'éditer un plat
GET	/api/meal/:id	ID du plat	`SELECT * FROM Meals WHERE ID_meal=ID`	Code 200, { "success": true } + toutes les infos du plat	Code 1XX, { "success": false, "error": error } 120 : Plat non trouvé 130 : Autre	Permettre à l'utilisateur d'avoir les détails d'un plat

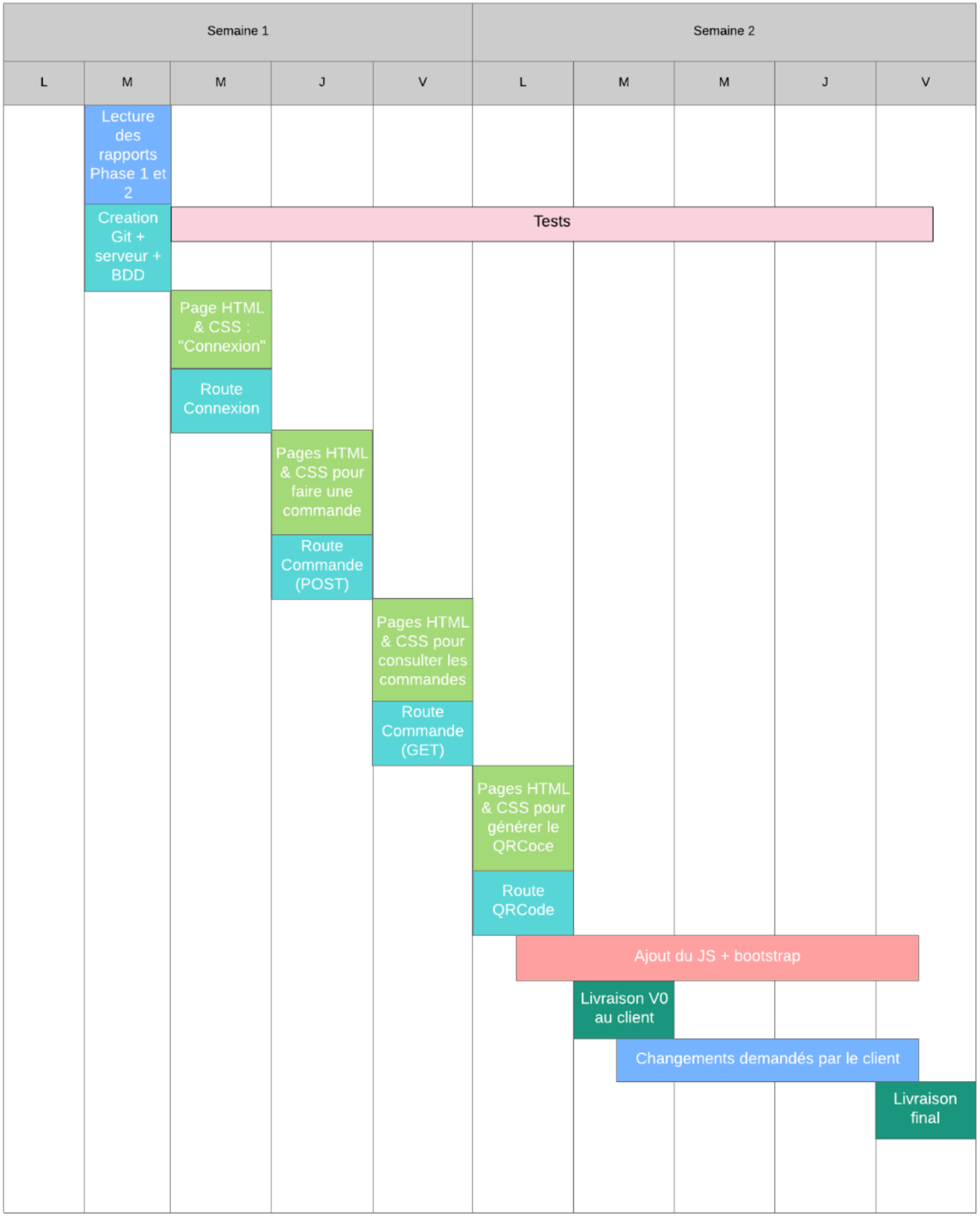
6. Planification du développement

Nous avons identifié les points principaux de l'application, c'est-à-dire les fonctionnalités primordiales à développer pour avoir un prototype du système de commande de plats à emporter :

- création de la base de données et de l'ensemble des tables
- connexion à un compte
    - créer la page html / css
    - créer la route /user sur l'api, avec vérification des informations dans la base de données
- création d'une commande
    - créer les pages html / css du formulaire avec choix de la formule, plats, et confirmation
    - créer la route /order, avec insertion de la commande dans la base de données
- affichage des commandes
    - créer les pages html / css pour afficher la ou les commandes de l'utilisateur
    - créer la page html / css pour afficher les détails d'une commandes
    - ajouter à la route /user les méthodes nécessaires pour récupérer les commandes de l'utilisateur
- génération de QRCode
    - créer la route /qrcode qui permet de générer un qrcode, et de modifier la base de données
    - implémenter la conversion de l'identifiant de qrcode en image qrcode

Les autres fonctionnalités telles que la création de comptes, ou l'ajout / modification des formules/plats sont secondaires, et les développeurs pourront travailler et tester l'application avec des tables de la base de données rempli à la main.

Une répartition du travail sous la forme de Gantt est la plus adaptée.
Le premier jour consiste à prendre connaissance du sujet, créer le dépôt GitLab, les différents serveurs, bases de données, etc.
Ensuite une phase d'implémentation et de créations des pages html / css, et de l'api RESTful.
Le mardi de la deuxième semaine, il faudrait montrer une version v0 au client, prendre son avis. Ensuite faire les modifications demandées, ainsi qu'améliorer les pages html avec du JS et du bootstrap, jusqu'au rendu final du vendredi 4 juin.


Semaine 1
Semaine 2
L
M
M
J
V
L
M
M
J
V
Lecture des rapports Phase 1 et 2
Creation Git + serveur + BDD
Tests
Page HTML & CSS : "Connexion"
Route Connexion
Pages HTML & CSS pour faire une commande
Route Commande (POST)
Pages HTML & CSS pour consulter les commandes
Route Commande (GET)
Pages HTML & CSS pour générer le QRCoce
Route QRCode
Ajout du JS + bootstrap
Livraison V0 au client
Changements demandés par le client
Livraison final

7. Retour d’expérience

Au cours de l’élaboration de la phase 2 du projet, nous avons appris à détailler les concepts superficiels présentés dans la phase 1. Réfléchir aux contraintes et les adapter aux besoins du client ont été des étapes primordiales pour concevoir nos diagrammes, schémas et routes.

Cette étape du projet fut l’occasion de faire l’intermédiaire entre le client et les développeurs. Il était important de concevoir l’architecture de la base de données et du système de manière à ce que celle-ci soit cohérente et logique au développement mais aussi fonctionnelle pour l’utilisateur.

Les principales difficultés résidaient dans la conception du schéma de la base de données, car ce dernier est crucial pour toute la suite du projet. Il définit la structure du système et précise la manière dont le système communique avec la base de données. Une fois ce dernier élaboré, la suite en découlait sans problèmes particuliers.

Cette phase du mini-projet nous a aussi permis d’apprendre à travailler en groupe autour d’un même projet, ainsi que de construire un rapport qui résume notre travail afin de le transmettre à un groupe suivant. De plus, répartir les tâches au sein du groupe fut primordial pour être efficace dans cette phase. Cette gestion commune des tâches nous sera sûrement précieuse pour la phase 3 de conception.

8. Annexes

Modèle de domaine

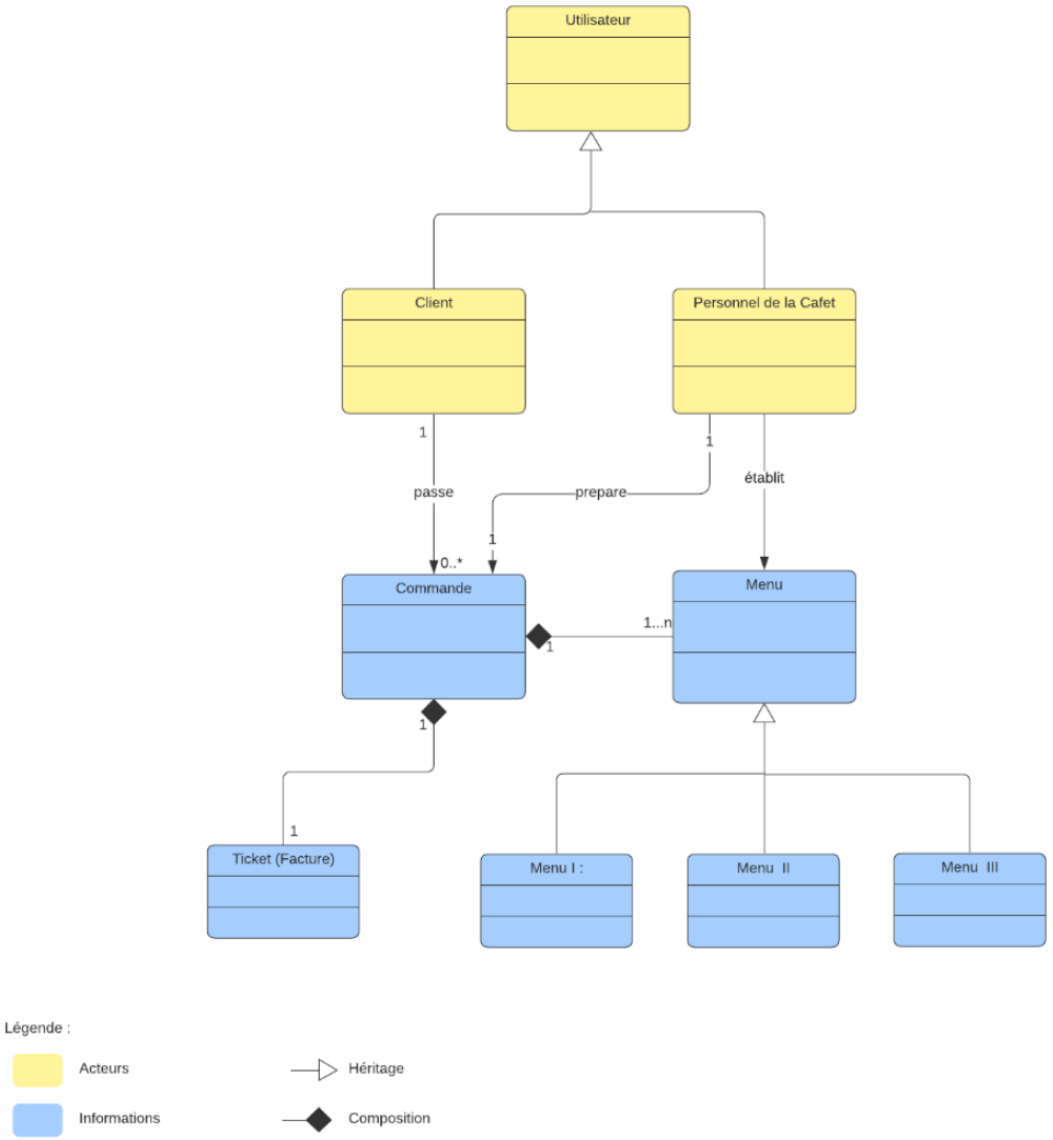
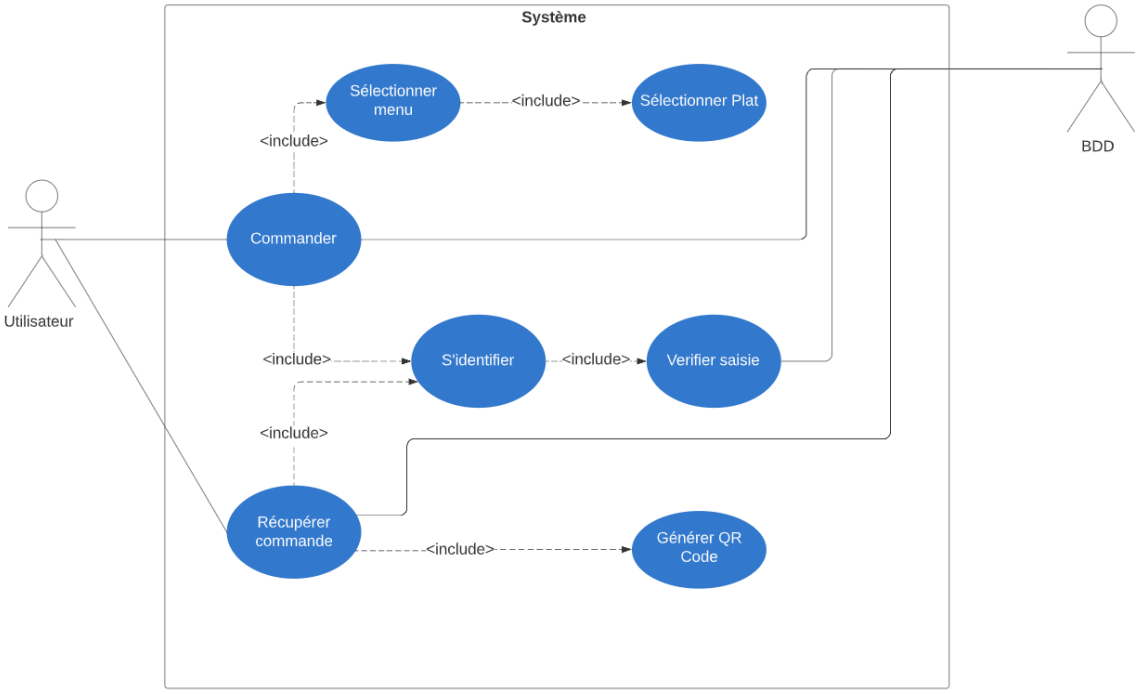
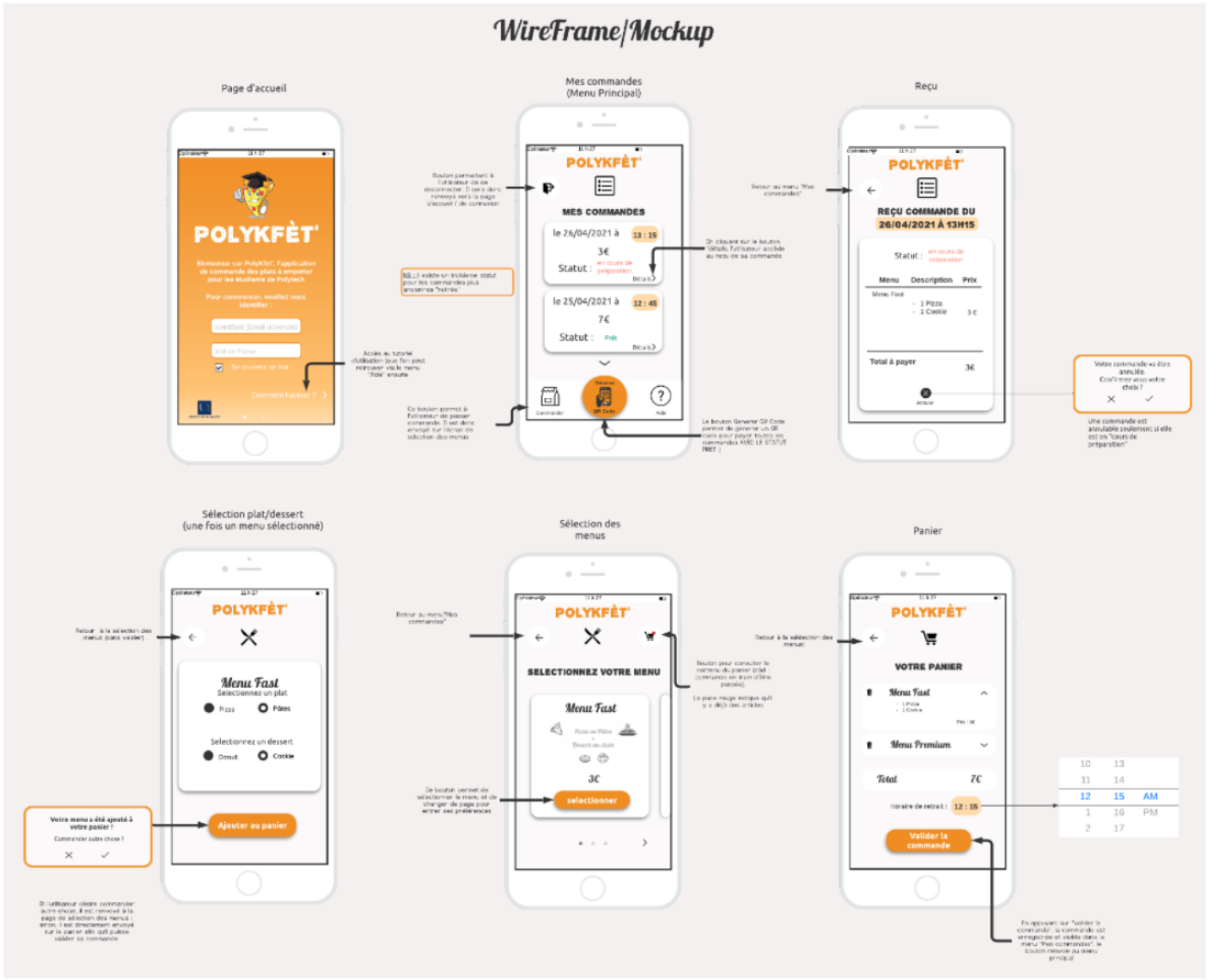


Diagramme des cas d'utilisation



WireFrame


WireFrame/Mockup
Page d'accueil
POLYKFÈT'
Mes commandes
(Menu Principal)
POLYKFÈT'
MES COMMANDES
le 26/04/2021 à 13 : 15
3€
Statut :
le 25/04/2021 à 12 : 45
7€
Statut : Prêt
Reçu
POLYKFÈT'
REÇU COMMANDE DU
26/04/2021 À 13H15
Statut :
Menu
Description
Prix
Total à payer
Votre commande va être annulée. Confirmez vous votre choix ?
Une commande est annulable seulement si elle est en "cours de préparation"
Sélection plat/dessert
(une fois un menu sélectionné)
POLYKFÈT'
Menu Fast
Sélectionnez un plat
Pizza
Pâtes
Sélectionnez un dessert
Donut
Cookie
Ajouter au panier
Votre menu a été ajouté à votre panier !
Commander autre chose ?
Sélection des
menus
POLYKFÈT'
SELECTIONNEZ VOTRE MENU
Menu Fast
3€
selectionner
Panier
POLYKFÈT'
VOTRE PANIER
Menu Fast
Menu Premium
Total
7€
Horaire de retrait : 12 : 15
Valider la commande
10 13
11 14
12 15 AM
1 16 PM
2 17