

**CONCEPTION D'UNE
APPLICATION DE RÉSERVATION
DE REPAS :
APPROCHE EN C ET JAVA**



ENCADRÉ PAR :
PROFESSEUR A. MAACH

Réalisé par :
BHAR Faiçal
IBRAHIMI Mohamed
JAAD Halima
KHMAICH Oualid

Sommaire

1. Introduction.....	(3)
- Contexte du projet	
- Objectifs	
2. Programmation en C.....	(4)
- Fondements de la Programmation	
- Gestion de la Mémoire	
- Logique Algorithmique	
3. Modélisation Conceptuelle.....	(9)
- Diagramme de Classes	
- Diagramme de Cas d'Utilisation	
- Diagramme d'Activité	
4. Déroulement de notre application C.....	(13)
5. Développement de l'Application en Java.....	(27)
- Architecture Logicielle	
- Implémentation des Fonctionnalités	
- Interfaces et Débogage	
6. Conclusion.....	(38)
- Récapitulation des Réalisations	
- Apprentissages et Perspectives Futures	

Introduction :

Dans le cadre du programme de notre école, l'École Mohammadia d'Ingénieurs (EMI), visant à promouvoir l'intégration des élèves et à les immerger dans des projets concrets pour appliquer leurs connaissances théoriques, notre groupe de quatre membres s'est engagé dans la conception et la réalisation d'un système de réservation de tickets en langage C, sous la tutelle éclairée du Professeur M. MAACH Adel. Ce projet, réalisé dans le cadre de notre formation, représente une opportunité unique pour mettre en pratique nos compétences et relever les défis techniques rencontrés lors du développement logiciel.

Le système que nous avons développé vise à répondre à un besoin crucial dans un contexte étudiantin : la réservation de repas dans un restaurant scolaire ou étudiant. Face aux problèmes récurrents tels que les files d'attente interminables, la gestion inadéquate des flux de clients et la complexité des processus de réservation, notre objectif était clair : créer une solution efficace et intuitive. En effet, notre application de réservation de tickets en C se propose d'optimiser le processus de réservation, de rendre l'accès aux repas plus fluide et de contribuer à une meilleure gestion des ressources disponibles.

À travers ce projet, nous nous sommes fixé pour mission de relever ces défis en offrant une solution informatique robuste et conviviale. En combinant nos connaissances en programmation, notre créativité et notre esprit d'équipe, nous aspirons à fournir une réponse concrète à ces problématiques quotidiennes rencontrées dans les restaurants scolaires et étudiants. En outre, ce projet représente une occasion précieuse d'acquérir une expérience pratique dans le développement de logiciels, de renforcer nos compétences en programmation et de nous familiariser avec les bonnes pratiques de conception logicielle.

Dans ce rapport, nous détaillerons le processus de conception et d'implémentation de notre système de réservation de tickets en C, en mettant en lumière les différentes étapes de développement, les choix architecturaux et les fonctionnalités clés. En outre, nous analyserons les défis rencontrés, les solutions apportées et les perspectives d'amélioration pour ce projet. Enfin, nous espérons que cette expérience nous aura permis de consolider nos connaissances et compétences, et de contribuer de manière significative à la résolution de problèmes réels dans un contexte académique et professionnel.

Description du système :

Le projet consiste en un système de réservation de repas en langage C, comprenant un gestionnaire de solde permettant aux étudiants de recharger leur compte, un gestionnaire de repas pour la gestion des plats disponibles et un système d'interaction entre les étudiants et le système pour la réservation de repas en fonction de leur solde et des repas disponibles.

Description de l'implementation en C :

Notre projet se base sur un code comportant plusieurs fonctions clés pour la gestion des repas et des étudiants.

1. `compterEtudiants`: Cette fonction ouvre un fichier contenant une liste d'étudiants, puis lit chaque ligne pour compter le nombre total d'étudiants présents dans le fichier.
2. `decrementerSolde`: Cette fonction prend en entrée le matricule d'un étudiant et la valeur à décrémenter de son solde. Elle ouvre le fichier contenant la liste des étudiants, recherche l'étudiant correspondant au matricule donné, puis décrémente son solde de la valeur spécifiée. Ensuite, elle met à jour les données dans le fichier.
3. `incrementerPetitDejeuner1`, `incrementerPetitDejeuner2`, `incrementerDejeuner1`, `incrementerDejeuner2`, `incrementerDiner1`, `incrementerDiner2`: Ces fonctions prennent en entrée le jour de la semaine et la quantité à incrémenter pour un plat spécifique (petit-déjeuner 1, petit-déjeuner 2, déjeuner 1, déjeuner 2, dîner 1, dîner 2). Elles ouvrent un fichier contenant les statistiques des repas pour chaque jour, recherchent le jour spécifié, puis incrémentent la quantité de repas correspondante. Enfin, elles mettent à jour les données dans le fichier.
4. `tickets_reservation`: Cette fonction enregistre une réservation de repas pour un étudiant. Elle prend en paramètres le matricule de l'étudiant, le jour de la réservation, ainsi que les quantités de chaque repas réservé (petit déjeuner 1, petit déjeuner 2, déjeuner 1, déjeuner 2, dîner 1 et dîner 2). Les informations sont enregistrées dans un fichier texte nommé "tickets.txt".
5. `check_reserve`: Cette fonction vérifie si une réservation existe déjà pour un étudiant donné à une date spécifique. Elle prend en paramètres le matricule de l'étudiant et le jour recherché. Elle lit le fichier "tickets.txt" pour rechercher une correspondance. Si une réservation est trouvée, la fonction retourne 1, sinon elle retourne 0.

6. **reserver**: Cette fonction permet à un étudiant de réserver des repas pour un jour donné. Elle demande à l'utilisateur de saisir le jour pour lequel il souhaite réserver les repas, puis affiche le menu de ce jour. Ensuite, elle demande à l'utilisateur de saisir les quantités de chaque plat qu'il souhaite réserver, vérifie si le solde de l'étudiant est suffisant pour effectuer la réservation, et enregistre la réservation si c'est le cas.
7. **afficherReservationsJour**: Cette fonction affiche les réservations pour un étudiant donné à une date spécifique. Elle prend en paramètres le matricule de l'étudiant et le jour recherché, puis lit le fichier "tickets.txt" pour afficher les réservations correspondantes.
8. **main**: La fonction principale du programme. Elle affiche un menu principal avec trois options : créer un compte, se connecter ou quitter. Selon le choix de l'utilisateur, elle appelle les fonctions correspondantes.
9. **creercompte**: Cette fonction permet à un nouvel utilisateur de créer un compte en saisissant son matricule, nom, prénom, génie, mot de passe et solde initial. Les informations sont enregistrées dans un fichier texte nommé "liste_etudiant.txt".
10. **log_in**: Cette fonction gère le processus de connexion des utilisateurs. Elle demande à l'utilisateur de saisir son matricule et son mot de passe, puis vérifie s'il s'agit d'un étudiant ou d'un gestionnaire en fonction de la longueur du matricule. Ensuite, elle appelle les menus correspondants.
11. **menu_SOLDEmanager**: Ce menu est destiné au gestionnaire. Il offre une option pour amplifier un solde.
12. **menu_etudiant**: Ce menu est destiné aux étudiants. Il offre des options pour réserver des repas, afficher les réservations et annuler une réservation.

13. **remplirMenu()** :

- Cette fonction permet à l'utilisateur de remplir le menu pour chaque jour de la semaine.
- Elle ouvre le fichier "menu.txt" en mode ajout ("a") pour vérifier si le jour existe déjà.
- Si le jour n'existe pas, elle demande à l'utilisateur de saisir les plats pour ce jour et les écrit dans le fichier.
- Si le jour existe déjà, elle informe l'utilisateur et ne fait rien.

14. **modifierMenu()** :

- Cette fonction permet à l'utilisateur de modifier le menu pour un jour spécifique.
- Elle ouvre le fichier "menu.txt" en mode lecture/écriture ("r+").
- Elle demande à l'utilisateur de saisir le jour à rechercher et les nouvelles informations pour ce jour.
- Elle écrit les modifications dans le fichier et tronque le fichier pour supprimer les anciennes données excédentaires.

15. **afficherStats()** :

- Cette fonction affiche les statistiques des réservations pour un jour spécifique.
- Elle ouvre le fichier "stats.txt" en mode lecture.
- Elle demande à l'utilisateur de saisir le jour à afficher.
- Elle recherche le jour dans le fichier et affiche les statistiques des réservations pour ce jour.
- Elle offre la possibilité de réinitialiser les compteurs pour ce jour.

16. **resetCompteurs()** :

- Cette fonction réinitialise les compteurs de réservation pour un jour spécifique.
- Elle ouvre le fichier "stats.txt" en mode lecture/écriture.
- Elle demande à l'utilisateur de saisir le jour à réinitialiser.
- Elle réinitialise les compteurs pour ce jour dans le fichier.

17. `Menu_Manager()` :

- Cette fonction gère le menu principal du programme.
- Elle affiche un menu avec différentes options comme remplir le menu, modifier le menu, afficher les statistiques, etc.
- Elle demande à l'utilisateur de choisir une option et appelle la fonction correspondante en fonction du choix.

18. `afficher_menu()` :

- Cette fonction affiche le menu pour un jour spécifique.
- Elle ouvre le fichier "menu.txt" en mode lecture.
- Elle demande à l'utilisateur de saisir le jour à afficher.
- Elle recherche le jour dans le fichier et affiche le menu pour ce jour.

19. `jutos()` :

- Cette fonction est la boucle principale du programme.
- Elle affiche le menu principal du programme en boucle.
- Elle demande à l'utilisateur de faire un choix et appelle la fonction correspondante en fonction du choix.

20. `suparo()` :

- Cette fonction permet d'ajouter de la solde à un étudiant.
- Elle ouvre le fichier "liste_etudiant.txt" en mode lecture/écriture.
- Elle demande à l'utilisateur de saisir le matricule de l'étudiant et la solde à ajouter.
- Elle recherche l'étudiant dans le fichier et met à jour sa solde.

21. `annulerReservation(int matricule)` :

- Cette fonction permet d'annuler une réservation pour un étudiant.

- Elle ouvre les fichiers "liste_etudiant.txt", "tickets.txt" et "stats.txt" pour récupérer les données.
- Elle demande à l'utilisateur de saisir le jour de la réservation à annuler.
- Elle recherche la réservation dans les fichiers et effectue les modifications nécessaires.
- Elle affiche un message indiquant si la réservation a été annulée avec succès ou non.

Description en UML :

Maintenant que nous avons décrit les différentes fonctions de notre programme de gestion de menus et de réservations, nous pouvons passer à une étape importante : la modélisation de notre système à l'aide de diagrammes UML. Les diagrammes UML nous permettront de visualiser la structure et le fonctionnement de notre système de manière plus claire et organisée. En utilisant des diagrammes de classes, de 'use case' et éventuellement d'activité, nous pourrons représenter les entités, les relations et les interactions entre les différents composants de notre système. Cette modélisation nous aidera à mieux comprendre l'architecture de notre programme et à planifier sa mise en œuvre de manière plus efficace. Passons maintenant à la création de nos diagrammes UML pour donner une vue d'ensemble plus détaillée de notre système.

1. Diagramme de classes :

Ce diagramme montre les différentes classes dans notre système (comme 'Jour', 'Quantité', 'Etudiant', 'Reservation', etc.) et leurs relations, telles que l'héritage, l'agrégation et l'association. Chaque classe est représentée par une boîte contenant son nom, ses attributs et ses méthodes, et les relations entre les classes sont illustrées par des lignes.

2. Diagramme de cas d'utilisation :

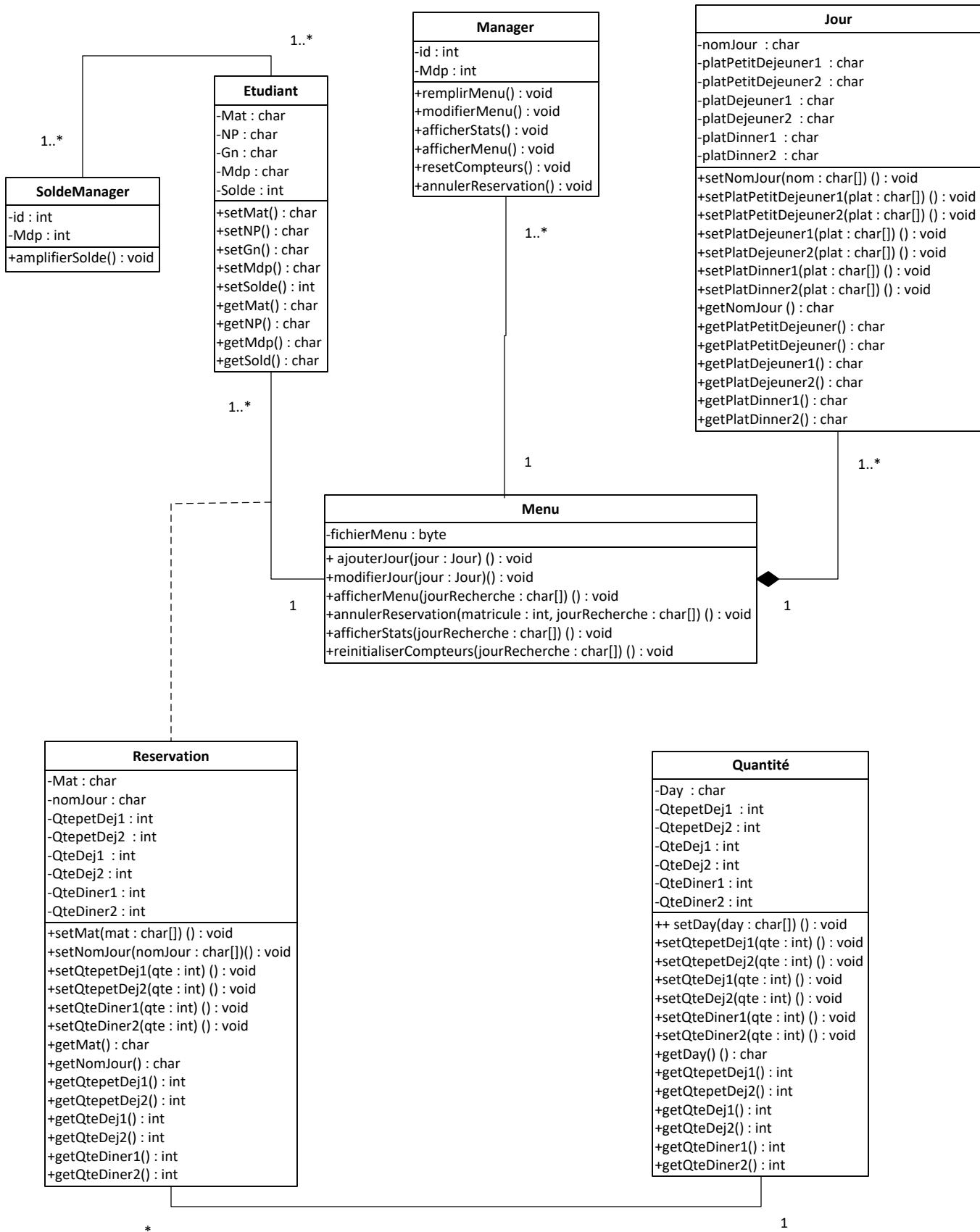
Ce diagramme représente les interactions entre les acteurs externes (comme les utilisateurs) et le système. Les cas d'utilisation sont des actions que les utilisateurs peuvent effectuer dans le système. Les acteurs sont représentés par des silhouettes d'humains ou de systèmes externes, et les cas d'utilisation sont des ovales. Les lignes reliant les acteurs aux cas d'utilisation représentent les interactions.

3. Diagramme d'activité :

Ce diagramme illustre le flux de contrôle à travers différentes activités dans le système. Chaque activité est représentée par une boîte rectangulaire, et les transitions entre les activités sont indiquées par des flèches. Ce diagramme est utile pour comprendre les étapes d'un processus ou d'une fonctionnalité dans le système, montrant comment les actions sont exécutées séquentiellement ou parallèlement.

Ci-joint, les diagrammes qui décrivent simplement le fonctionnement de notre programme:

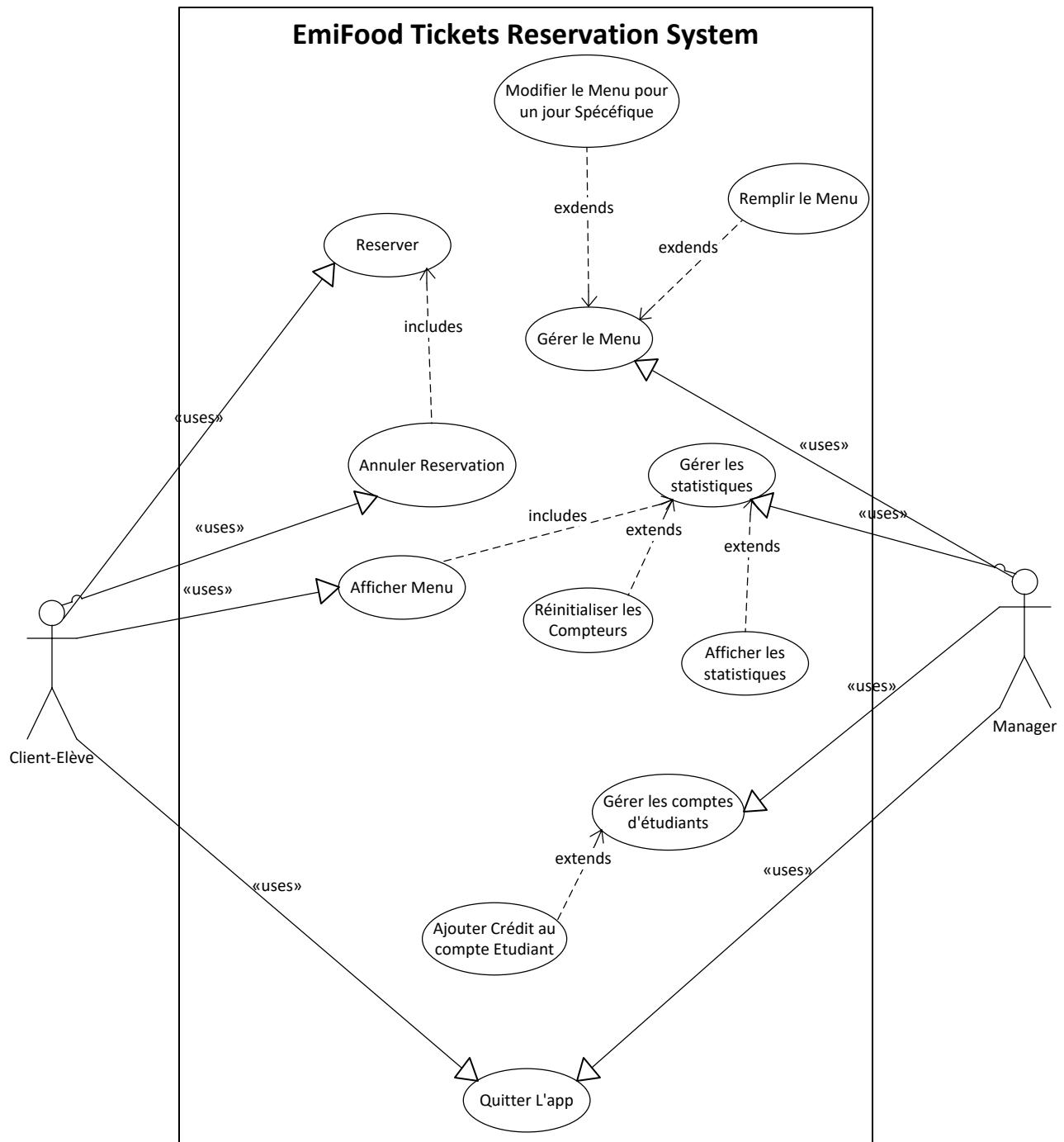
1. Diagramme de classes :



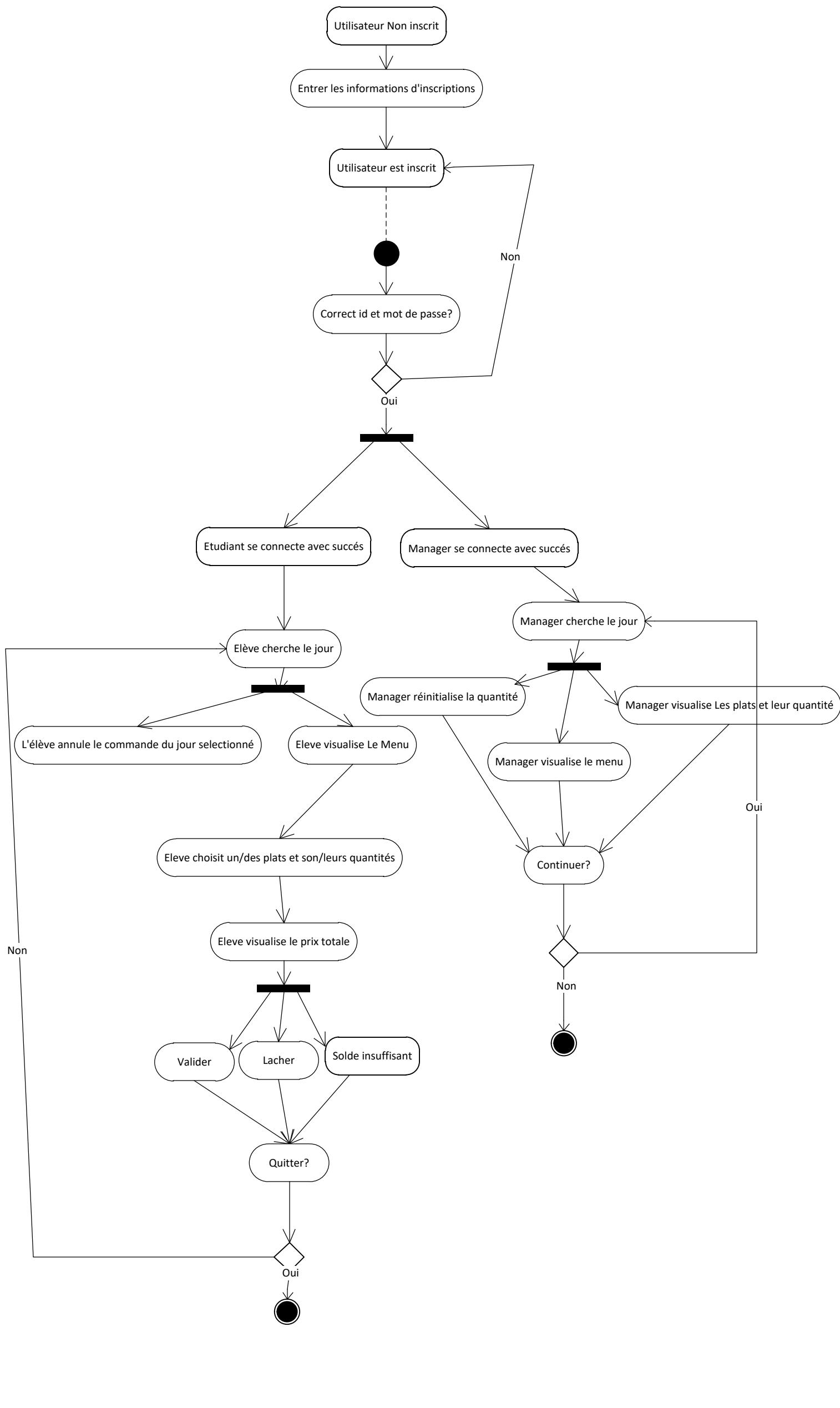
*

1

2. Diagramme de cas d'utilisation :



3. Diagramme d'activité :



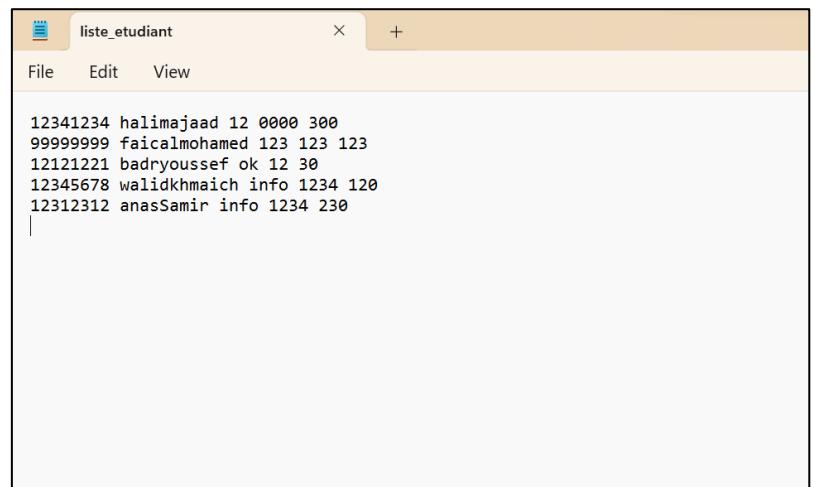
Déroulement de notre application

Bienvenue dans notre application de réservation de tickets scolaires !

Plongeons dans un univers où nous, les étudiants, pouvons gérer nos repas et nos initiatives en quelques clics. Avec une interface intuitive et conviviale, cette application simplifie notre quotidien scolaire.

On commence par présenter de manière détaillée le déroulement de chaque étape, ainsi que les changements qui surviennent après chaque clic.

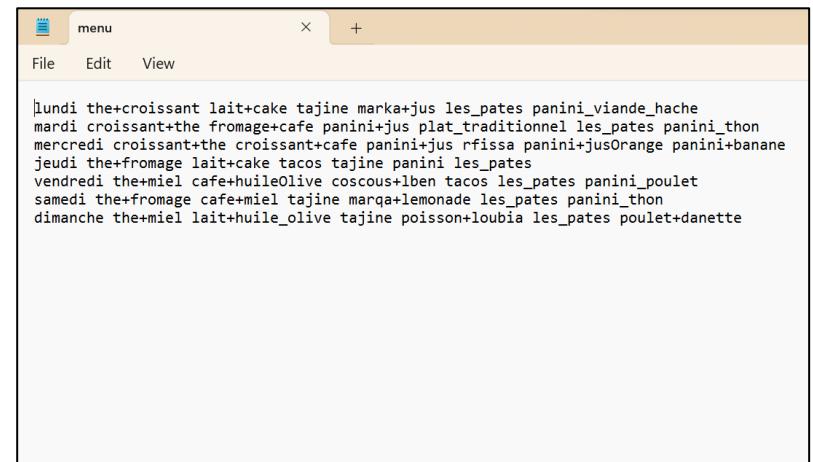
Tout d'abord, on initie notre application par certains nombreux élèves pour pouvoir remarquer les changements et bien comprendre le fonctionnement de notre application. (Capture n°1)



ID	Nom	1	2	3	4	5
12341234	halimajaad	12	0000	300		
99999999	faicalmohamed	123	123	123		
12121221	badryoussef	ok	12	30		
12345678	walidkhmaich	info	1234	120		
12312312	anasSamir	info	1234	230		

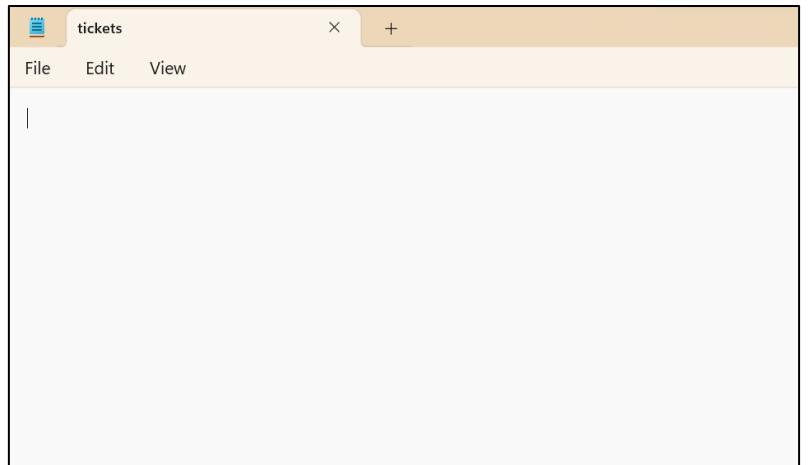
Capture n°1 : la liste initiative de quelques étudiants

De même pour le menu des plats suggérés pour chaque jour par le manager.



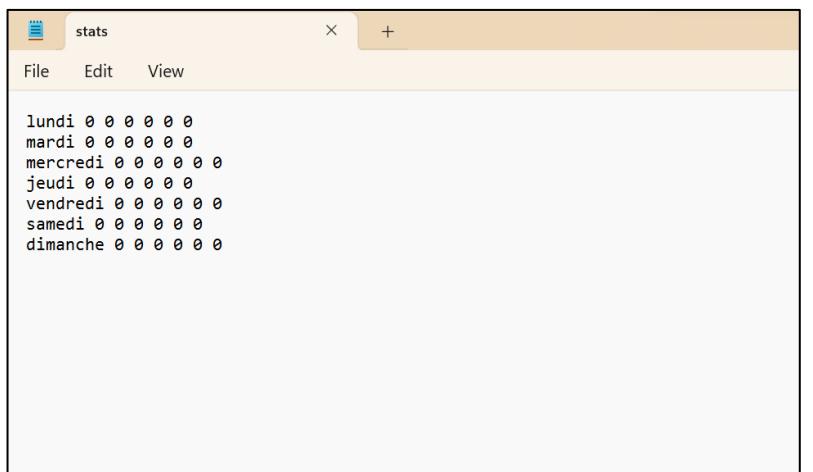
Jour	Plat Suggéré
Lundi	the+croissant lait+cake tajine marka+jus les_pates panini_viande_hache
Mardi	croissant+the fromage+cafe panini+jus plat_traditionnel les_pates panini_thon
Mercredi	croissant+the croissant+cafe panini+jus rfissa panini+jusOrange panini+banane
Jeudi	the+fromage lait+cake tacos tajine panini les_pates
Vendredi	the+miel cafe+huileOlive couscous+ben tacos les_pates panini_poulet
Samedi	the+fromage cafe+miel tajine marqa+lemonade les_pates panini_thon
Dimanche	the+miel lait+huile_olive tajine poisson+loubia les_pates poulet+danette

Capture n°2 : la liste initiative du menu des repas



Et ici on a une liste vide de tickets tant qu'on n'a pas encore initié aucune réservation .

Capture n°3 : la liste des tickets prises à t_0

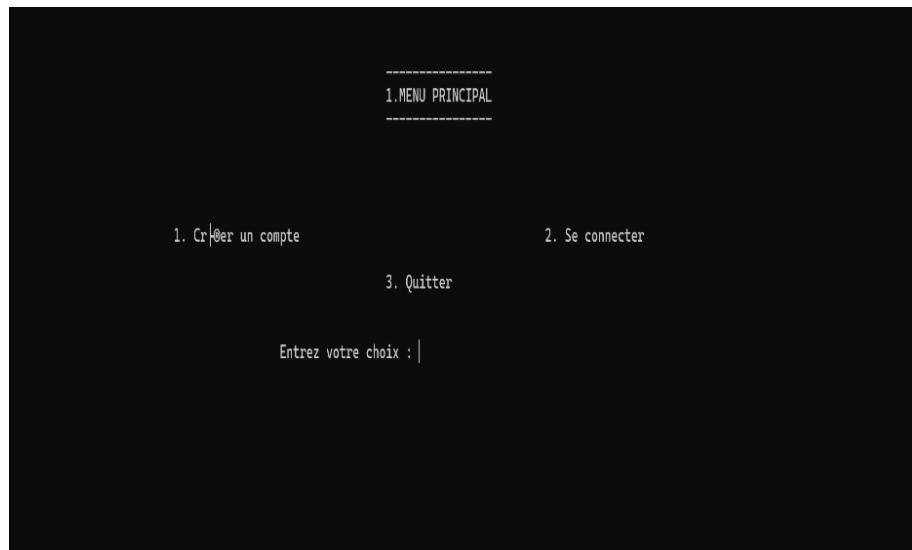


Et la même chose pour la liste des statistiques des repas qui va indiquer au manager combien de repas il doit préparer pour chaque jour et chaque plat .

Capture n°4 : la liste initiative de repas à préparer

Avant toute réservation, le système initial est rétabli à zéro, offrant ainsi une toile vierge à chaque étudiant. Cela signifie que chaque fonctionnalité, chaque option de menu, chaque possibilité d'initiative étudiante, ainsi que les statistiques des repas et les soldes, sont restaurés à leur état initial. Ainsi, chaque utilisateur peut commencer son parcours de réservation avec une expérience fraîche et sans aucune influence préalable, assurant une équité et une clarté totales dans le processus de réservation.

Ensuite nous verrons les changements qui vont survenir après chaque étape ou choix pour lequel on optera selon les cas :



Capture n°5 : le Menu principal de l'application



Capture n°6 : déroulement de la création d'un compte

liste_etudiant	
	x
File	Edit
12341234 halimajaad 12 0000 300	
99999999 faicalmohamed 123 123 123	
12121221 badryoussef ok 12 30	
12345678 walidkmaich info 1234 120	
12312312 anassamir info 1234 230	
12900921 ahmedyassine mecanique 1211 140	

Capture n°7 : création du nouveau compte pour Ahmed Yassine

```

1. Cr|@er un compte          2. Se connecter
3. Quitter

Entrez votre choix : 2
Entrez votre matricule: 12312312
Entrez votre mot de passe: 0000
mot de passe ou matricule errones !

-----
1.MENU PRINCIPAL
-----
```



```

1. Cr|@er un compte          2. Se connecter
3. Quitter

Entrez votre choix : 2
Entrez votre matricule: 12312312
Entrez votre mot de passe: 1234

-----
MENU ETUDIANT
-----
```



```

1. Reserver                 2. liste des reservations
3. Annuler la reservation
0. Quitter
```

Capture n°8 : déroulement de la connexion

```

-----  

MENU ETUDIANT  

-----
```



```

1. Reserver                 2. liste des reservations
3. Annuler la reservation
0. Quitter

Entrez votre choix : 1
Entrez le jour pour lequel vous souhaitez r|@server les plats : lundi

lundiLe menu de ce jour est pour :
Petit D|@jeuner :      the+croissant et    lait+cake
D|@jeuner :           tajine et     marka+jus
D|xner :              les_pates et     panini_viande_hache
Entrez la quantit|@ de petit d|@jeuner1 : 2
Entrez la quantit|@ de petit d|@jeuner2 : 0
Entrez la quantit|@ de petit d|@jeuner1 : 1
Entrez la quantit|@ de petit d|@jeuner2 : 3
Entrez la quantit|@ de d|xner1 : 2
Entrez la quantit|@ de d|xner2 : 1
merci

prix total = 18
Solde d|@cr|@ment|@.
Solde mis |@ jour avec succ|@s pour le matricule 12312312.
R|@servation confirm|@e avec succ|@s.
```



```

-----  

MENU ETUDIANT
-----
```

Capture n°9 : une réservation correctement faite

```

1. Reserver           2. liste des reservations
3. Annuler la reservation
0. Quitter
Entrez votre choix : 2
Entrez le jour pour afficher les r|@servations : mardi
Aucune r|@servation trouv|@e pour le matricule 12312312 et le jour mardi.

-----
MENU ETUDIANT
-----


1. Reserver           2. liste des reservations
3. Annuler la reservation
0. Quitter
Entrez votre choix : 2
Entrez le jour pour afficher les r|@servations : lundi
*****
*          TICKET R|@SERVATION      *
*****
Matricule: 12312312
Jour: lundi
-----
Petit D|@jeuner 1: 2
Petit D|@jeuner 2: 0
D|@jeuner 1:      1
D|@jeuner 2:      3
D|@kner 1:        2
D|@kner 2:        1
-----
Merci pour votre r|@servation!
*****

```

Capture n°10 : la liste des réservations

Et ainsi on voit le changement dans le fichier des étudiants où le solde devient 230-18=212dhs après la réservation :

```

liste_etudiant
File Edit View
X +
12341234 halimajaad 12 0000 300
99999999 faicalmohamed 123 123 123
12121221 badryoussef ok 12 30
12345678 walidkhmaich info 1234 120
12312312 anasSamir info 1234 212
2900921 ahmedyassine mecanique 1211 140

```

Capture n°11 : le changement après la réservation

Et ensuite le changement est apparent que ce soit dans le fichier des tickets pour pouvoir les voir et les montrer au responsable de restaurant pour jouir de son plat (Capture n°10), ou les statistiques que le manager voit pour savoir combien il doit préparer :

A screenshot of a terminal window titled "tickets". The window has a standard OS X-style title bar with a close button (X) and a plus sign (+). Below the title bar is a menu bar with "File", "Edit", and "View" options. The main area of the terminal contains the following text:

```
|12312312 lundi 2 0 1 3 2 1
```

Capture n°12 : la liste des tickets

A screenshot of a terminal window titled "stats". The window has a standard OS X-style title bar with a close button (X) and a plus sign (+). Below the title bar is a menu bar with "File", "Edit", and "View" options. The main area of the terminal contains the following text:

```
|lundi 2 0 1 3 2 1
|mardi 0 0 0 0 0 0
|mercredi 0 0 0 0 0 0
|jeudi 0 0 0 0 0 0
|vendredi 0 0 0 0 0 0
|samedi 0 0 0 0 0 0
|dimanche 0 0 0 0 0 0
```

Capture n°13 : la liste des statistiques des repas

Et pour le cas contraire, si l'étudiant n'a pas le solde suffisant pour effectuer son achat etachever sa réservation :

```

1. Reserver           2. liste des reservations
3. Annuler la reservation
0. Quitter

Entrez votre choix : 1
Entrez le jour pour lequel vous souhaitez r|@server les plats : mardi

lundi

mardiLe menu de ce jour est pour :
Petit D|@jeuner :      croissant+the    et      fromage+cafe
D|@jeuner :            panini+jus     et      plat_traditionnel
D|xner :                les_pates      et      panini_thon
Entrez la quantit|@ de petit d|@jeuner1 : 100
Entrez la quantit|@ de petit d|@jeuner2 : 100
Entrez la quantit|@ de petit d|@jeuner1 : 100
Entrez la quantit|@ de petit d|@jeuner2 : 0
Entrez la quantit|@ de d|xner1 : 0
Entrez la quantit|@ de d|xner2 : 0
merci

prix total = 600
Votre solde n'est pas suffisant pour cette op|@ration. Vous devez minimaliser le prix de 388 Dh
Souhaitez-vous continuer la r|@servation ? 0 si non, autre si oui
1
Vous |-tes en train de rechoisir les quantit|@s des plats...
Entrez le jour pour lequel vous souhaitez r|@server les plats : mardi

lundi

mardiLe menu de ce jour est pour :
Petit D|@jeuner :      croissant+the    et      fromage+cafe
D|@jeuner :            panini+jus     et      plat_traditionnel
D|xner :                les_pates      et      panini_thon
Entrez la quantit|@ de petit d|@jeuner1 : 200
Entrez la quantit|@ de petit d|@jeuner2 : 100
Entrez la quantit|@ de petit d|@jeuner1 : 0
Entrez la quantit|@ de petit d|@jeuner2 : 0
Entrez la quantit|@ de d|xner1 : 00
Entrez la quantit|@ de d|xner2 : 0
merci

prix total = 600
Votre solde n'est pas suffisant pour cette op|@ration. Vous devez minimaliser le prix de 388 Dh
Souhaitez-vous continuer la r|@servation ? 0 si non, autre si oui
0

Process returned 0 (0x0)   execution time : 311.259 s
Press any key to continue.

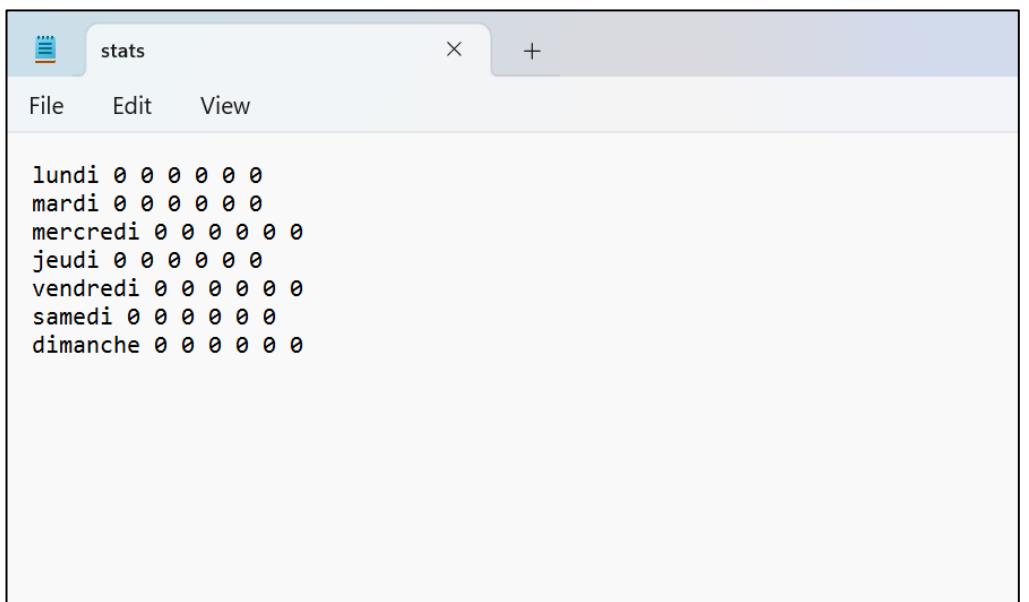
```

Capture n°14 : une réservation inachevée

Et pour traiter le cas où un étudiant voudrait annuler sa réservation , notre application recherche la réservation correspondante, annule la réservation si elle est trouvée en ajustant le solde de l'étudiant et en mettant à jour les statistiques des repas. En cas d'échec, elle affiche un message indiquant qu'aucune réservation n'a été trouvée pour l'étudiant et le jour spécifiés :

```
-----  
MENU ETUDIANT  
-----  
  
1. Reserver          2. liste des reservations  
3. Annuler la reservation  
0. Quitter  
  
Entrez votre choix : 3  
Entrez le jour de la r@ervation h@ annuler : jeudi  
Aucune r@ervation trouv@e pour cet h@tudiant et ce jour.  
  
-----  
MENU ETUDIANT  
-----  
  
1. Reserver          2. liste des reservations  
3. Annuler la reservation  
0. Quitter  
  
Entrez votre choix : 3  
Entrez le jour de la r@ervation h@ annuler : lundi  
R@ervation annul@e avec succ@s.  
  
-----  
MENU ETUDIANT  
-----
```

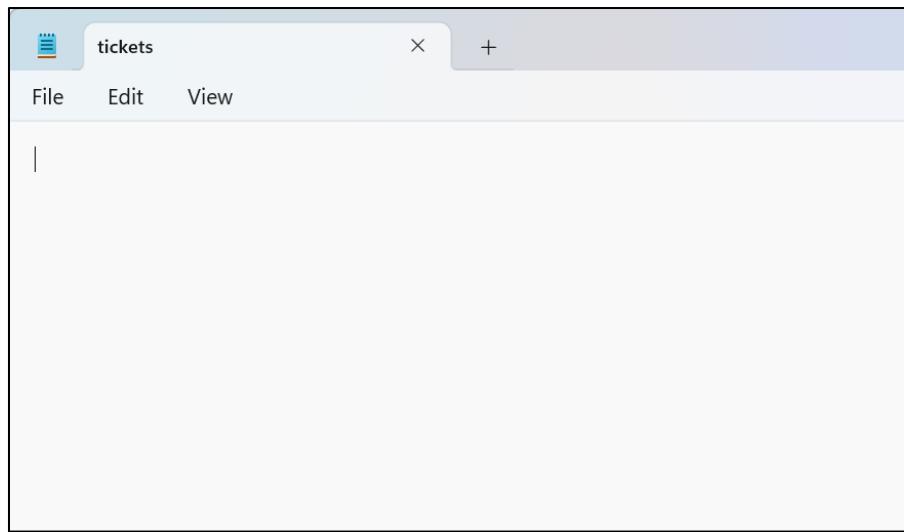
Capture n°15 : Annuler une réservation



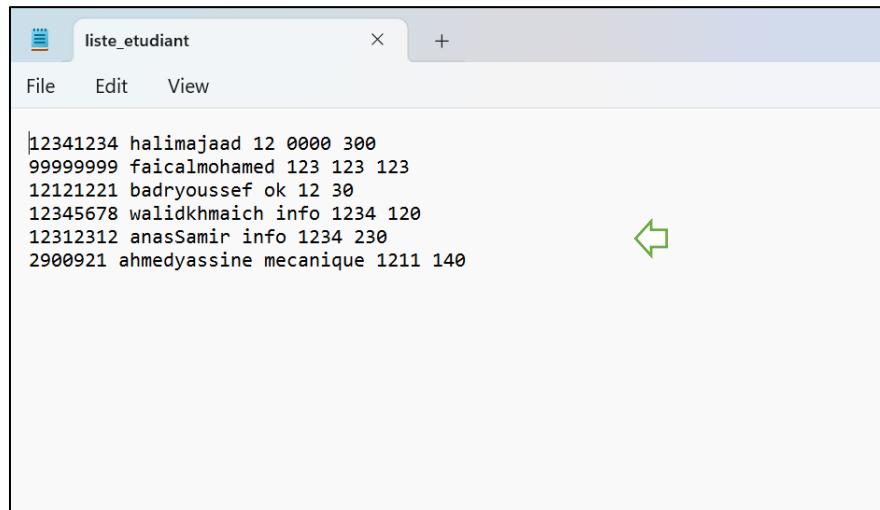
The screenshot shows a terminal window with the title bar 'stats'. The menu bar includes 'File', 'Edit', and 'View'. The main content area displays a grid of numbers representing weekly statistics:

	lundi	mardi	mercredi	jeudi	vendredi	samedi	dimanche
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0

Capture n°16 : le changement des statistiques après l'annulation



Capture n°17 : le changement des tickets après l'annulation

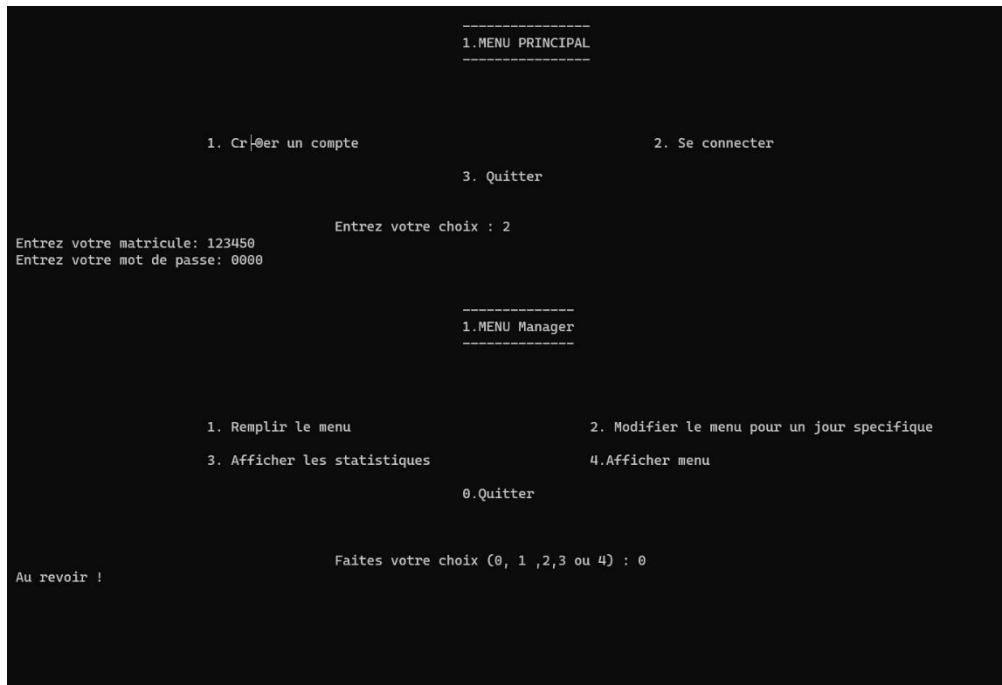


Capture n°18 : le solde reprend sa valeur avant la réservation

```
int matriculeManager = 123450;
int mdpManager = 0000;

int matrSoldemanager=112200;
int mdpsoldemanager=0000;
```

Capture n°19 : les mots de passe des managers



Capture n°20 : la Connexion en tant que manager



Capture n°21 : la Modification du menu des plats par le manager

```
-----  
1. MENU Manager  
-----  
  
1. Remplir le menu          2. Modifier le menu pour un jour spécifique  
3. Afficher les statistiques 4. Afficher menu  
0. Quitter  
  
Faites votre choix (0, 1 ,2,3 ou 4) : 4  
Entrez le nom du jour dont vous voulez afficher le menu : lundi  
Jour non trouv|0 dans le menu.  
  
-----  
1. MENU Manager  
-----  
  
1. Remplir le menu          2. Modifier le menu pour un jour spécifique  
3. Afficher les statistiques 4. Afficher menu  
0. Quitter  
  
Faites votre choix (0, 1 ,2,3 ou 4) : 4  
Entrez le nom du jour dont vous voulez afficher le menu : lundi  
Menu pour le jour lundi :  
Petit Djeuner 1: a+a  
Petit Djeuner 2: a+b  
Djeuner 1: a+c  
Djeuner 2: a+d  
Diner 1: a+e  
Diner 2: a+f
```

Capture n°22 : le menu après modification

```

1. Reserver                               2. liste des reservations
                                         3. Annuler la reservation
                                         6. Quitter

Entrez votre choix : 1
Entrez le jour pour lequel vous souhaitez réserver les plats : lundi

lundiLe menu de ce jour est pour :
Petit Djeuner :      a+a      et      a+b
Djeuner :            a+c      et      a+d
Diner :              a+e      et      a+f
Entrez la quantité à de petit djeuner1 : 1
Entrez la quantité à de petit djeuner2 : 2
Entrez la quantité à de petit djeuner3 : 3
Entrez la quantité à de petit djeuner4 : 4
Entrez la quantité à de dinner1 : 1
Entrez la quantité à de dinner2 : 1
merci

prix total = 24
Solde d'écritement 0.
Solde mis à jour avec succès pour la matricule 12312312.
Réervation confirmée avec succès.

MENU ETUDIANT

```

```

1. Reserver                               2. liste des reservations
                                         3. Annuler la reservation
                                         6. Quitter

Entrez votre choix : 1
Entrez le jour pour lequel vous souhaitez réserver les plats : vendredi

lundi
mardi
mercredi
jeudi

vendrediLe menu de ce jour est pour :
Petit Djeuner :      the+miel      et      cafe+huileOlive
Djeuner :            couscous+liban   et      tacos
Diner :              les_pates      et      pandini_poulet
Entrez la quantité à de petit djeuner1 : 2
Entrez la quantité à de petit djeuner2 : 2
Entrez la quantité à de petit djeuner3 : 2
Entrez la quantité à de petit djeuner4 : 1
Entrez la quantité à de dinner1 : 2
Entrez la quantité à de dinner2 : 3
merci

prix total = 24
Solde d'écritement 0.
Solde mis à jour avec succès pour la matricule 12312312.
Réervation confirmée avec succès.

```

Capture n°20 : deux réservations en lundi et vendredi

Le manager du restaurant commence par l'exécution de la fonction modifierMenu, qui lui permet de mettre à jour le menu pour un jour spécifique. Il saisit le nom du jour à rechercher et entre ensuite les nouvelles informations pour les différents plats du petit déjeuner, du déjeuner et du dîner. Une fois les modifications apportées, le menu est mis à jour avec succès et un message de confirmation est affiché. Si le jour spécifié n'est pas trouvé dans le menu, un message d'erreur approprié est affiché. Ensuite, le manager peut afficher les statistiques des réservations pour un jour spécifique en exécutant la fonction afficherStats. Après avoir saisi le jour à afficher, les statistiques des réservations pour ce jour sont présentées, y compris le nombre de plats à préparer pour chaque repas. Le manager a également la possibilité de réinitialiser les compteurs pour ce jour, ce qui met à zéro le nombre de plats à préparer quand

il a fini de préparer un certain nombre déjà demandé de plats. Si le jour spécifié n'est pas trouvé dans les statistiques, un message d'erreur approprié est affiché.

```

Faites votre choix (0, 1 ,2,3 ou 4) : 3
Entrer le jour a afficher : lundi      Statistiques des reservations pour le jour lundi :
                                         petit Dejeuner 1 :    1 plats a preparer
                                         petit Dejeuner 2 :    2 plats a preparer
                                         Dejeuner 1 :        3 plats a preparer
                                         Dejeuner 2 :        4 plats a preparer
                                         diner 1 :          1 plats a preparer
                                         diner 2 :          1 plats a preparer

Choisissez une option :
1. Reinitialiser les compteurs pour ce jour
2. Ne rien faire
1
Entrer le jour a reinitialiser : lundi
Compteurs r@initialis@os avec succ@is pour le jour lundi.

-----1.MENU Manager-----
```

```

1. Remplir le menu           2. Modifier le menu pour un jour specifique
3. Afficher les statistiques 4. Afficher menu
0.Quitar
```

```

Faites votre choix (0, 1 ,2,3 ou 4) : 3
Entrer le jour a afficher : lundi      Statistiques des reservations pour le jour lundi :
                                         petit Dejeuner 1 :    0 plats a preparer
                                         petit Dejeuner 2 :    0 plats a preparer
                                         Dejeuner 1 :          0 plats a preparer
                                         Dejeuner 2 :          0 plats a preparer
                                         diner 1 :            0 plats a preparer
                                         diner 2 :            0 plats a preparer

Choisissez une option :
1. Reinitialiser les compteurs pour ce jour
2. Ne rien faire
2

-----1.MENU Manager-----
```

Capture n°24 : les plats à préparer et réinitialisation du compteur

Ici, on saisit le matricule de l'étudiant et la solde à ajouter. Ensuite, elle recherche l'étudiant correspondant dans le fichier, met à jour sa solde et enregistre les modifications dans le fichier. En cas de succès, elle affiche un message d'erreur

```

-----1.MENU PRINCIPAL-----
```

```

1. Creer un compte           2. Se connecter
3. Quitter
```

```

Entrez votre choix : 2
Entrez votre matricule: 112288
Entrez votre mot de passe: 0000
```

```

-----menu_SOLDE_manager-----
```

```

1. Amplifier un solde         0. Quitter
```

```

Entrez votre choix : 1
Saisir le matricule de l'étudiant : 123456788
Saisir la solde a ajouter : 9
|étudiant non trouv@ dans la liste.
```

Capture n°25 : la Connexion en tant que manager du solde

```
-----  
menu_SOLDE_manager  
-----  
  
1. Amplifier un solde          0. Quitter  
  
Entrez votre choix : 1  
Saisir le matricule de l'etudiant : 12312312  
Saisir la solde a ajouter : 300  
Solde mis à jour avec succès pour le matricule 12312312.  
  
-----  
menu_SOLDE_manager  
-----  
  
1. Amplifier un solde          0. Quitter  
  
Entrez votre choix : 0  
Sortie du menu.
```

Capture n°26 : Amplification du solde de l'étudiant avec 300dhs

Application en Java

On a développé une application en Java visant à simplifier le processus de réservation des repas au restaurant de l'EMI (École Mohammadia d'Ingénieurs). Cette application se décline en trois volets principaux : client, chef et manager, chacun avec ses fonctionnalités distinctes et complémentaires.

Dès le départ, on a mis l'accent sur l'accessibilité pour les étudiants en proposant un processus d'inscription convivial. Au lieu de se perdre dans des procédures complexes, les étudiants peuvent simplement s'enregistrer en utilisant leur numéro de téléphone et un mot de passe, offrant ainsi une expérience utilisateur fluide dès le début.

Une fois connectés, les étudiants peuvent explorer les options de repas disponibles pour la journée. Grâce à une interface conviviale, ils peuvent parcourir les plats du jour, régulièrement mis à jour par le chef du restaurant. Cette fonctionnalité garantit une expérience culinaire variée et dynamique, tout en permettant au chef de partager les dernières créations et spécialités.

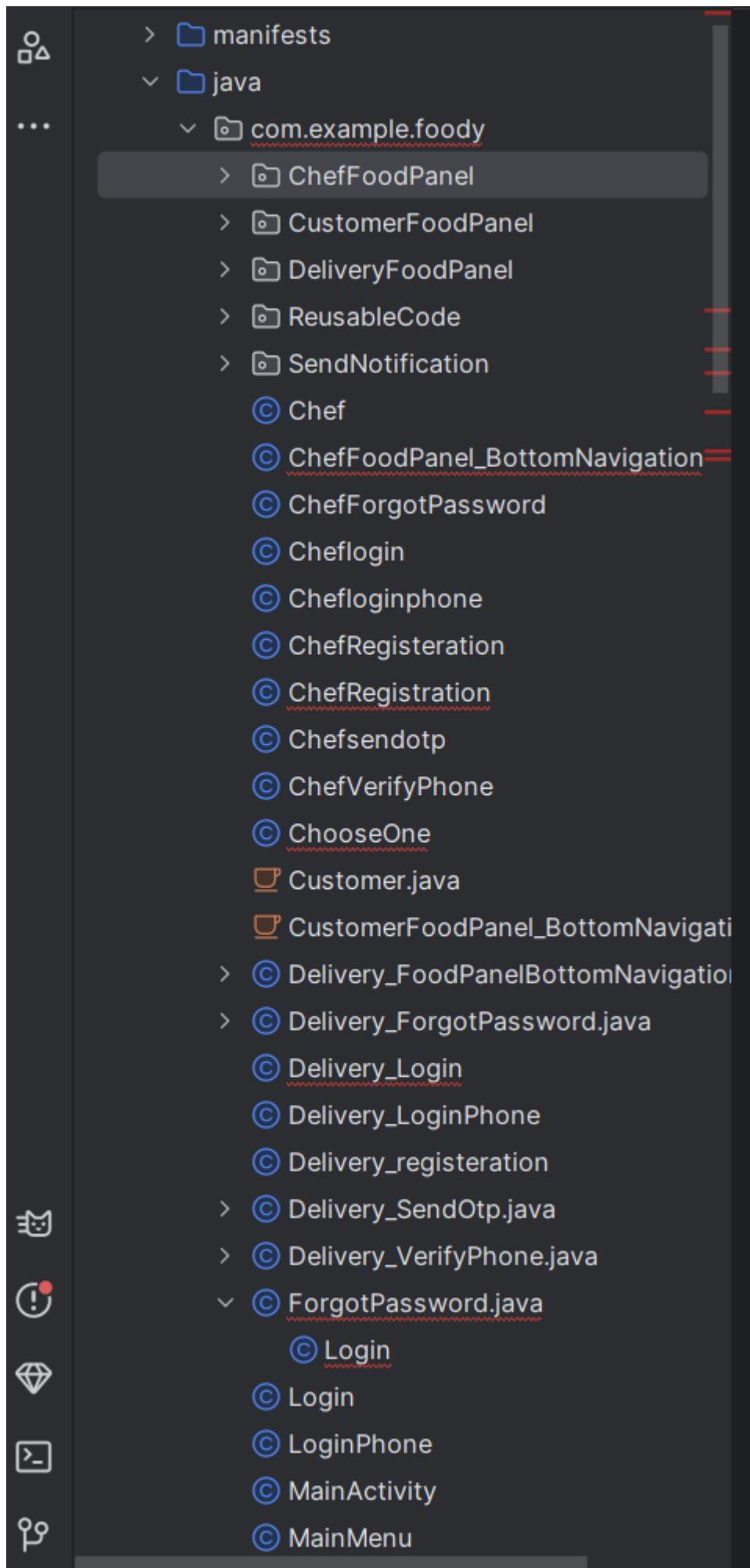
Quant au volet "chef", il joue un rôle essentiel dans la gestion des menus et des disponibilités. À travers une interface dédiée, le chef peut mettre à jour le menu en temps réel, en ajoutant de nouveaux plats, en ajustant les portions disponibles et en tenant compte des préférences des étudiants. Cette collaboration étroite entre le chef et les clients garantit une offre culinaire toujours à jour et répondant aux besoins de la communauté étudiante.

Parallèlement, le volet "manager" assure la supervision et la coordination de l'ensemble du processus. En surveillant les commandes, en résolvant les éventuels problèmes et en garantissant un service de qualité, le manager contribue à maintenir une expérience utilisateur optimale.

En résumé, on a conçu une application qui établit un lien direct entre le restaurant et les étudiants, offrant ainsi une plateforme flexible et interactive. Cette application ne se contente pas de simplifier le processus de réservation, mais elle crée également une expérience culinaire immersive et enrichissante pour la communauté étudiante de l'EMI.

Pour la réalisation de cette application, on a opté pour Figma pour la conception du logo et des détails graphiques, tandis que Firebase a été choisi comme base de données pour assurer la gestion efficace des informations relatives aux utilisateurs, aux menus et aux commandes.

Voici les dossiers qu'on aura besoin pour bâtrir notre application:



et voici finalement l'interface activity_chef_login:

En effet , vu qu'il ya énormément de classes, on va se focalisé que sur les plus importantes:

Voici la classe Chef qui correspond au chefs du restaurant de l'EMI:

```
1 package com.example.foody;
2
3 public class Chef {
4
5     2 usages
6     private String Area,City, ConfirmPassword,EmailID,Fname,House,Lname,Mobile,Password,Postcode, State,Suburban;
7
8     public Chef(String Area, String city, String confirmPassword, String emailID, String fname, String house, String lname, String m
9         this.Area = Area;
10        City = city;
11        ConfirmPassword = confirmPassword;
12        EmailID = emailID;
13        Fname = fname;
14        House = house;
15        Lname = lname;
16        Mobile = mobile;
17        Password = password;
18        Postcode = postcode;
19        State = state;
20        Suburban = suburban;
21    }
22
23    public Chef() {
24    }
25
26    no usages
27    public String getArea() { return Area; }
28
29    public String getCity() { return City; }
30
31    no usages
32    public String getConfirmPassword() { return ConfirmPassword; }
33
34    no usages
35    public String getEmailID() { return EmailID; }
36
37
38
39
40
```

et voici finalement l'interface activity_chef_login:

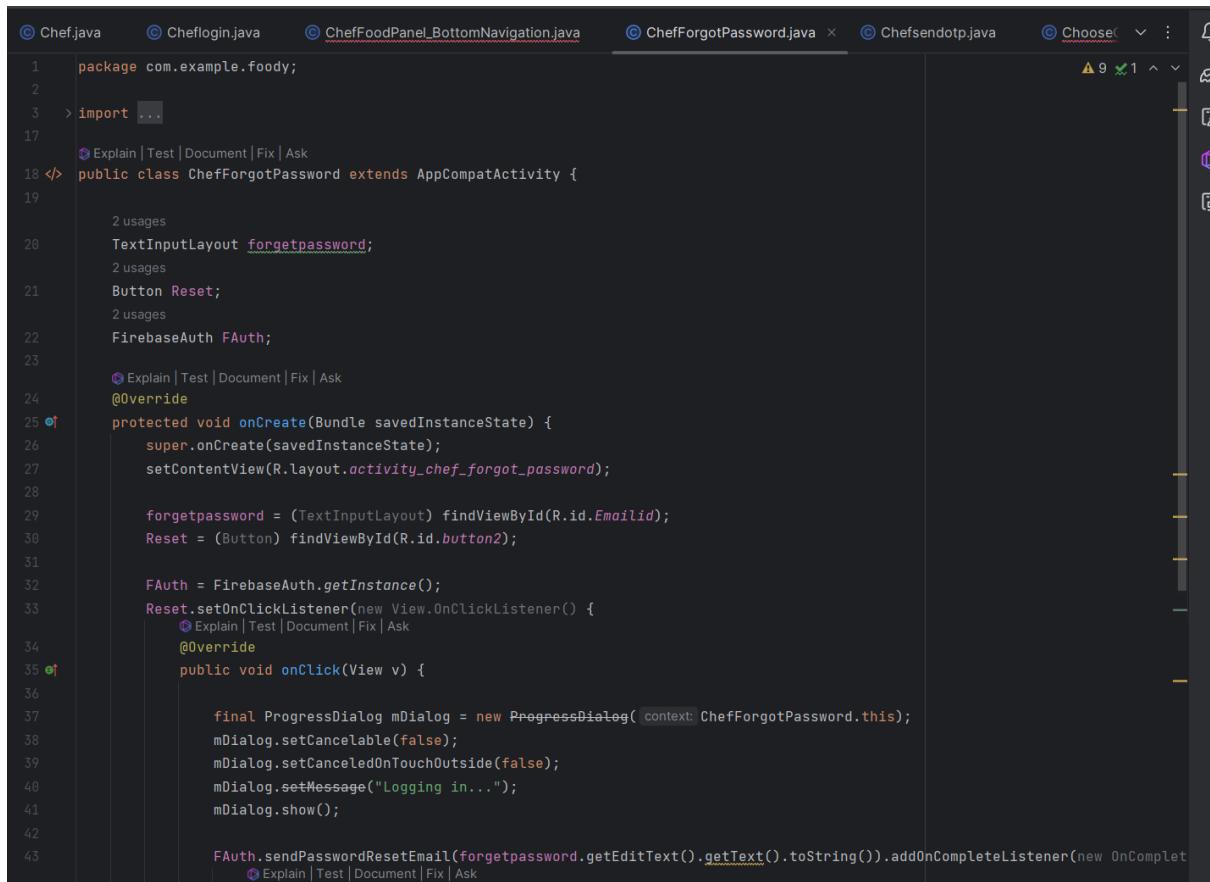
```
28 >     public String getCity() { return City; }
29
30     no usages
31 >     public String getConfirmPassword() { return ConfirmPassword; }
32
33     no usages
34 >     public String getEmailID() { return EmailID; }
35
36     2 usages
37 >     public String getName() { return Fname; }
38
39     no usages
40 >     public String getHouse() { return House; }
41
42     2 usages
43 >     public String getLname() { return Lname; }
44
45     public String getMobile() { return Mobile; }
46
47     public String getPassword() { return Password; }
48
49     no usages
50 >     public String getPostcode() { return Postcode; }
51
52     public String getState() { return State; }
53
54     4 usages
55 >     public String getSuburban() { return Suburban; }
56
57 }
```

et pour le login du chef , on écrit:

```
1 package com.example.foody;
2
3 > import ...
4
5 </> public class Cheflogin extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_cheflogin2);
11    }
12 }
```

et pour retrouver son mot de passe perdu, on a établi:

et voici finalement l'interface activity_chef_login:

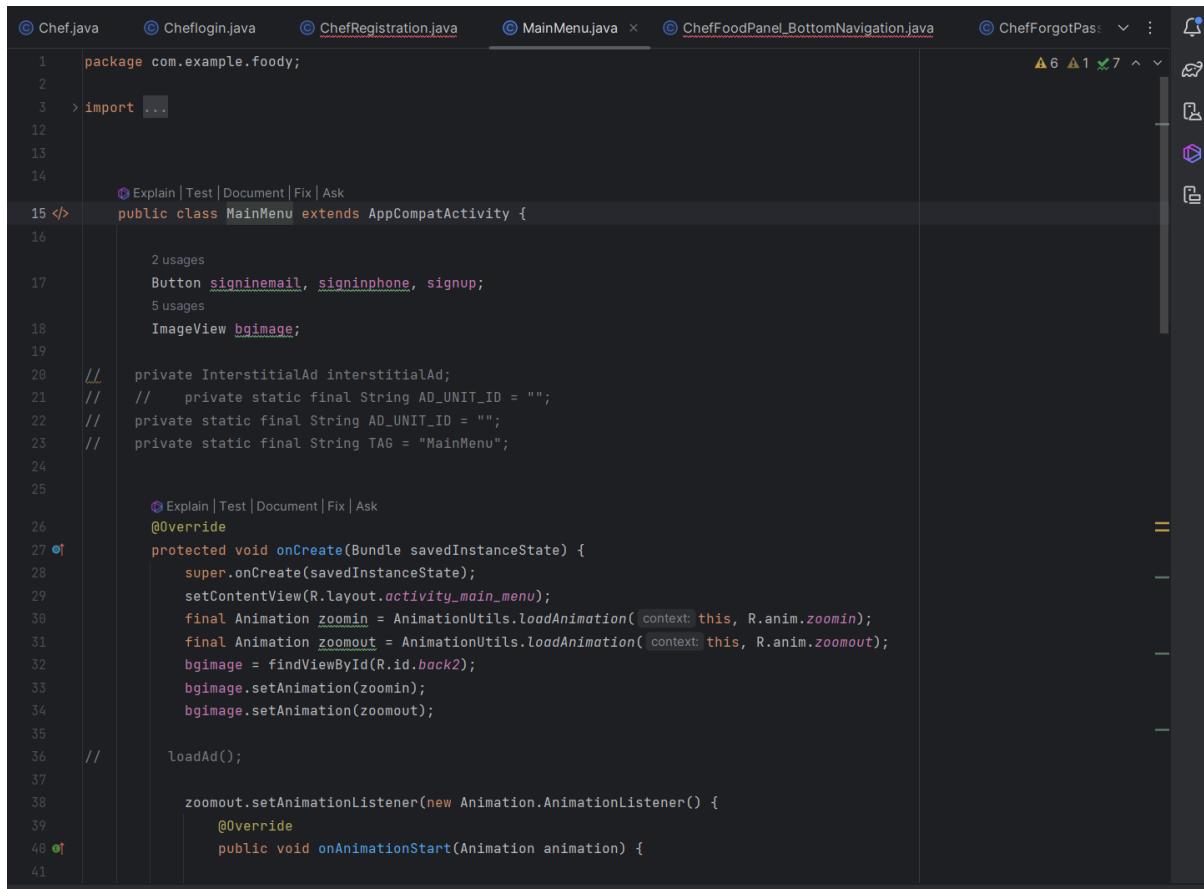


The screenshot shows the Android Studio code editor with the file `ChefForgotPassword.java` open. The code is for a `Forgot Password` screen. It imports `com.example.foody`, `... ,`, and `AppCompatActivity`. The class extends `AppCompatActivity` and overrides `onCreate`. In `onCreate`, it sets the content view to `R.layout.activity_chef_forgot_password`, finds views for `forgetpassword` and `Reset`, and initializes `FAuth`. It then sets an `onClick` listener for the `Reset` button. The `onClick` method creates a `ProgressDialog`, sets its message to "Logging in...", and shows it. Finally, it calls `FAuth.sendPasswordResetEmail` with the email entered in `forgetpassword`. The code uses `EditText` and `String` types.

```
1 package com.example.foody;
2
3 > import ...;
4
5 <> public class ChefForgotPassword extends AppCompatActivity {
6
7     2 usages
8     TextInputLayout forgetpassword;
9     2 usages
10    Button Reset;
11    2 usages
12    FirebaseAuth FAuth;
13
14    <> Explain | Test | Document | Fix | Ask
15    @Override
16    protected void onCreate(Bundle savedInstanceState) {
17        super.onCreate(savedInstanceState);
18        setContentView(R.layout.activity_chef_forgot_password);
19
20        forgetpassword = (TextInputLayout) findViewById(R.id.Emailid);
21        Reset = (Button) findViewById(R.id.button2);
22
23        FAuth = FirebaseAuth.getInstance();
24        Reset.setOnClickListener(new View.OnClickListener() {
25            <> Explain | Test | Document | Fix | Ask
26            @Override
27            public void onClick(View v) {
28
29                final ProgressDialog mDialog = new ProgressDialog(context: ChefForgotPassword.this);
30                mDialog.setCancelable(false);
31                mDialog.setCanceledOnTouchOutside(false);
32                mDialog.setMessage("Logging in...");
33                mDialog.show();
34
35                FAuth.sendPasswordResetEmail(forgetpassword.getText().toString()).addOnCompleteListener(new OnComplete
36                <> Explain | Test | Document | Fix | Ask
37            }
38        });
39    }
40
41    43
```

La classe mainmenu est la suivante:

et voici finalement l'interface activity_chef_login:



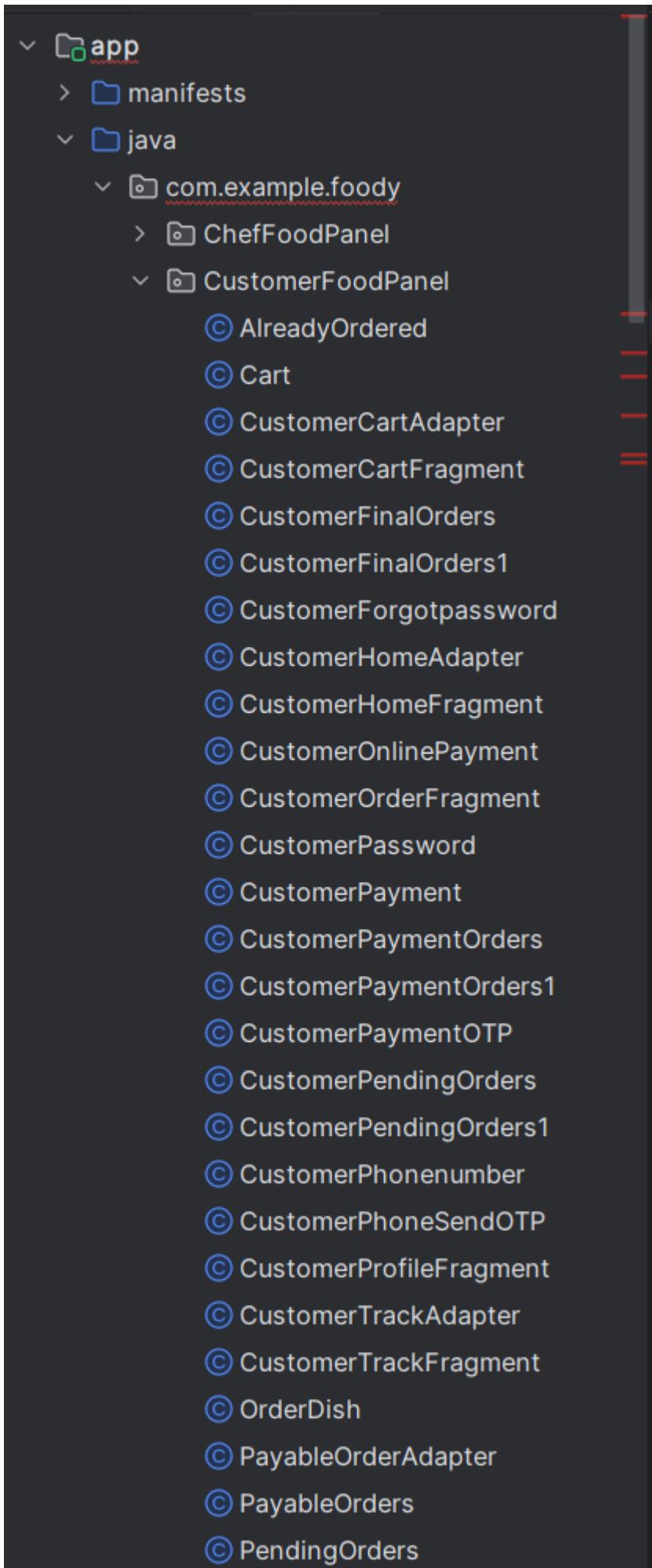
The screenshot shows the Android Studio code editor with the file `MainMenu.java` open. The code is as follows:

```
1 package com.example.foody;
2
3 > import ...
4
5
6     2 usages
7     Button signinemail, signinphone, signup;
8     5 usages
9     ImageView bgimage;
10
11 // private InterstitialAd interstitialAd;
12 // //    private static final String AD_UNIT_ID = "";
13 // private static final String AD_UNIT_ID = "";
14 // private static final String TAG = "MainMenu";
15
16     2 usages
17     Button signinemail, signinphone, signup;
18     5 usages
19     ImageView bgimage;
20
21 // private InterstitialAd interstitialAd;
22 // //    private static final String AD_UNIT_ID = "";
23 // private static final String AD_UNIT_ID = "";
24 // private static final String TAG = "MainMenu";
25
26     2 usages
27     Button signinemail, signinphone, signup;
28     5 usages
29     ImageView bgimage;
30
31     2 usages
32     Button signinemail, signinphone, signup;
33     5 usages
34     ImageView bgimage;
35
36     2 usages
37     Button signinemail, signinphone, signup;
38     5 usages
39     ImageView bgimage;
40
41     2 usages
42     Button signinemail, signinphone, signup;
43     5 usages
44     ImageView bgimage;
```

The code is heavily annotated with comments and usage counts. The interface includes buttons for signing in via email or phone and a sign-up button, along with an ImageView for a background image. It also includes logic for loading an interstitial ad.

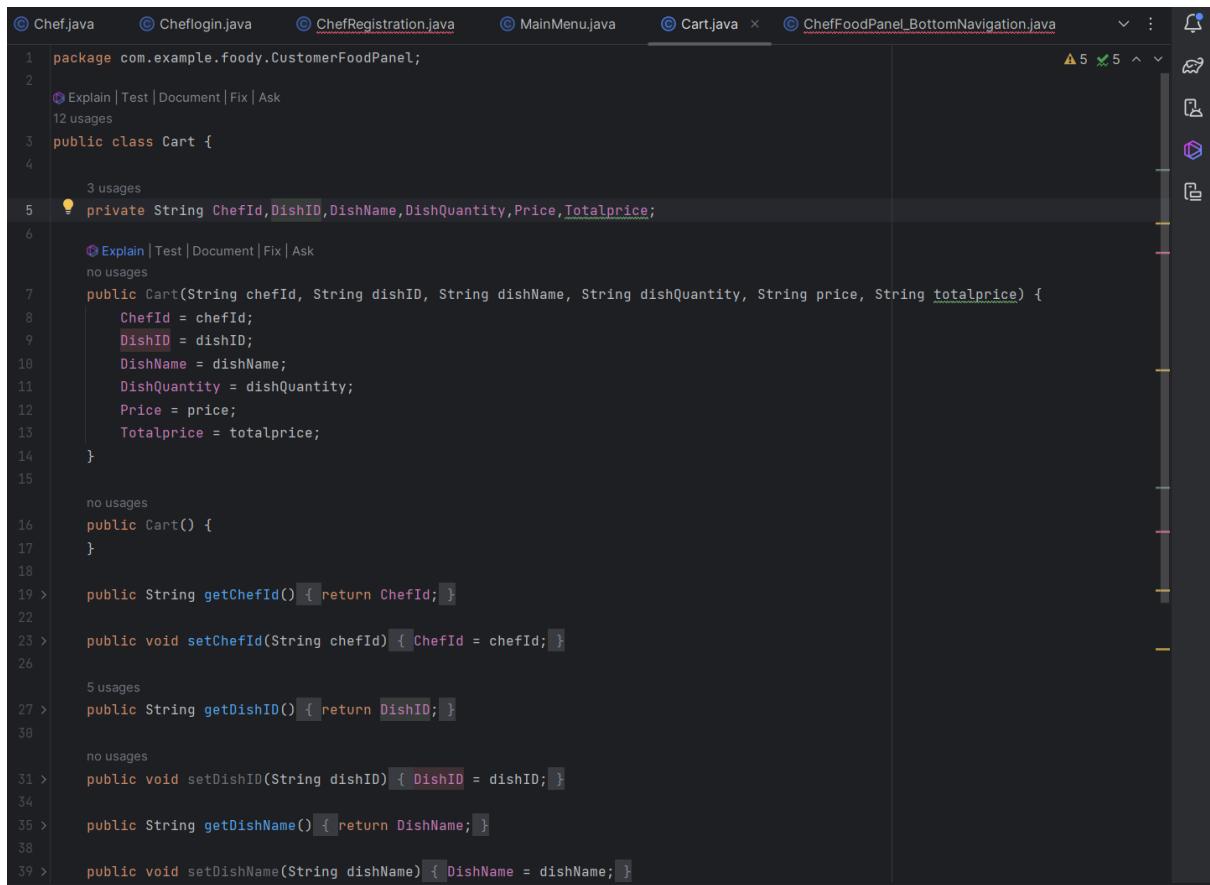
Voici les autres classes du Customer(les élèves et les enseignants):

et voici finalement l'interface activity_chef_login:



et voici finalement l'interface activity_chef_login:

Prenons la classe Cart ou on trouve:



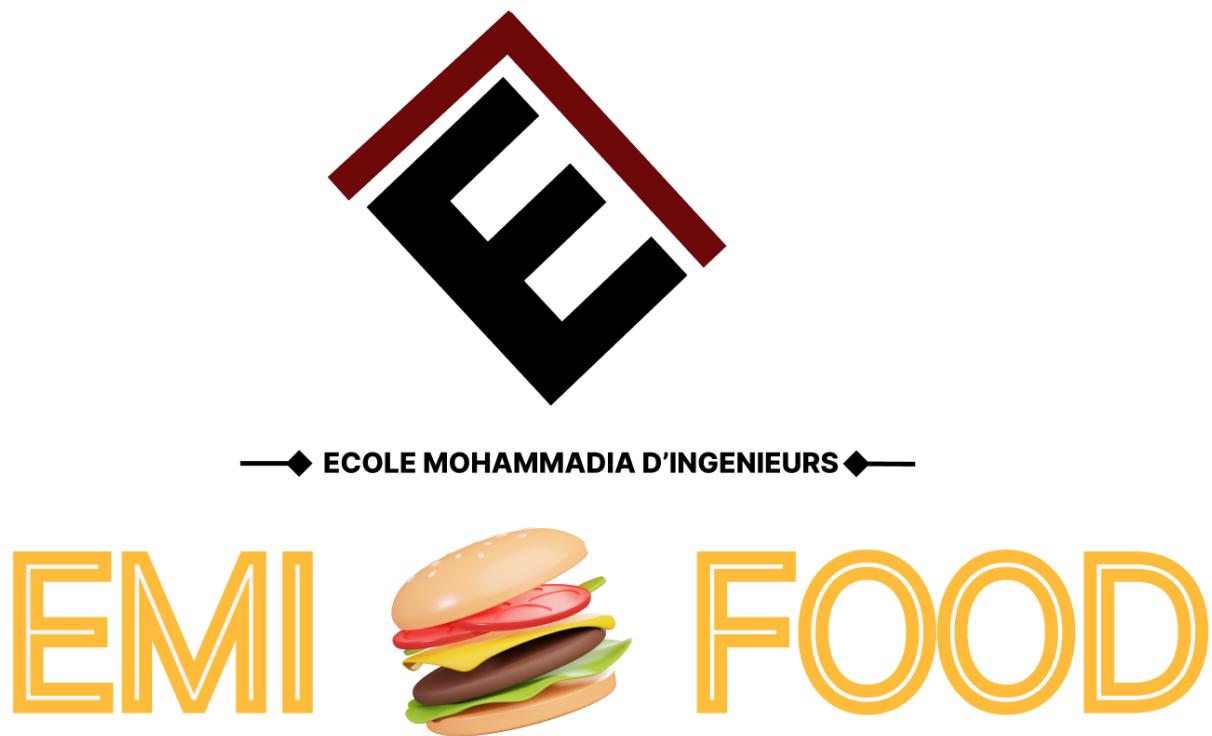
The screenshot shows the Android Studio code editor with the file `Cart.java` open. The code defines a class `Cart` with private fields `ChefId`, `DishID`, `DishName`, `DishQuantity`, `Price`, and `Totalprice`. It includes a constructor, a default constructor, and methods for getting and setting each field. The code is annotated with Javadoc-style comments and usage counts.

```
1 package com.example.foody.CustomerFoodPanel;
2
3 // Explain | Test | Document | Fix | Ask
4 // 12 usages
5 public class Cart {
6
7     // 3 usages
8     private String ChefId,DishID,DishName,DishQuantity,Price,Totalprice;
9
10    // Explain | Test | Document | Fix | Ask
11    // no usages
12    public Cart(String chefId, String dishID, String dishName, String dishQuantity, String price, String totalprice) {
13        ChefId = chefId;
14        DishID = dishID;
15        DishName = dishName;
16        DishQuantity = dishQuantity;
17        Price = price;
18        Totalprice = totalprice;
19    }
20
21    // no usages
22    public Cart() {
23    }
24
25    // 5 usages
26    public String getChefId() { return ChefId; }
27
28    public void setChefId(String chefId) { ChefId = chefId; }
29
30    // 5 usages
31    public String getDishID() { return DishID; }
32
33    public void setDishID(String dishID) { DishID = dishID; }
34
35    public String getDishName() { return DishName; }
36
37    public void setDishName(String dishName) { DishName = dishName; }
```

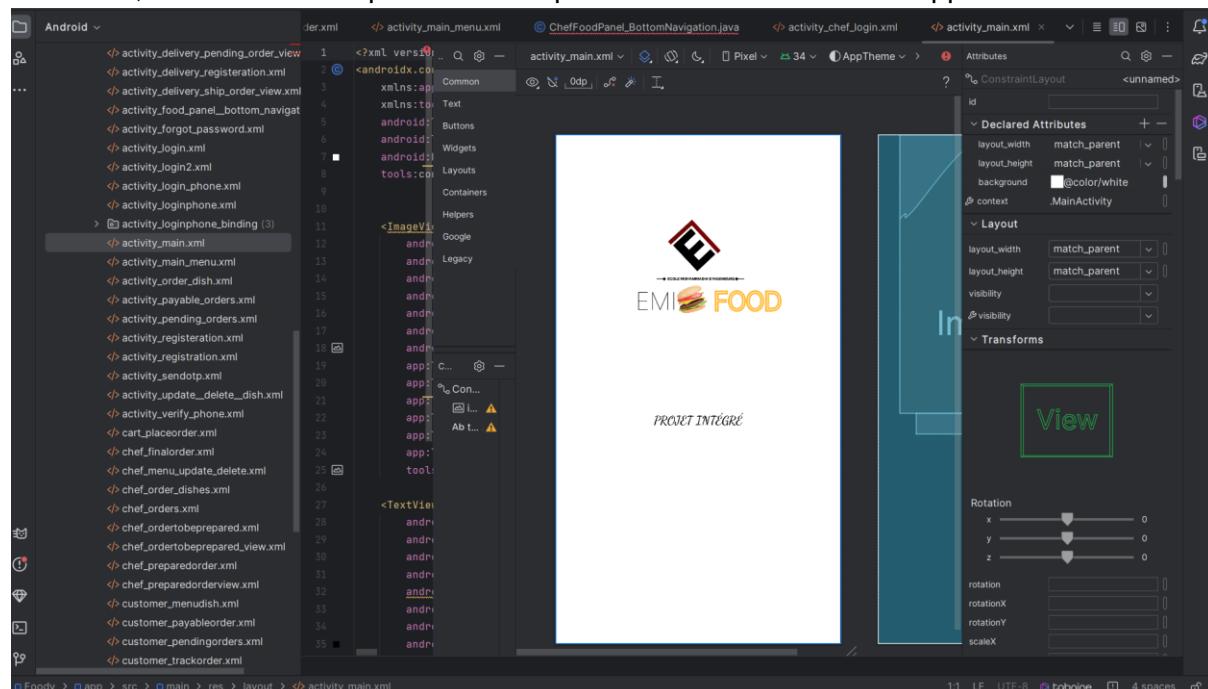
Et pour l'interface graphique, on a utilisé Android studio et bien évidemment Figma pour créer un logo à notre app:

Voici le logo:

et voici finalement l'interface activity_chef_login:

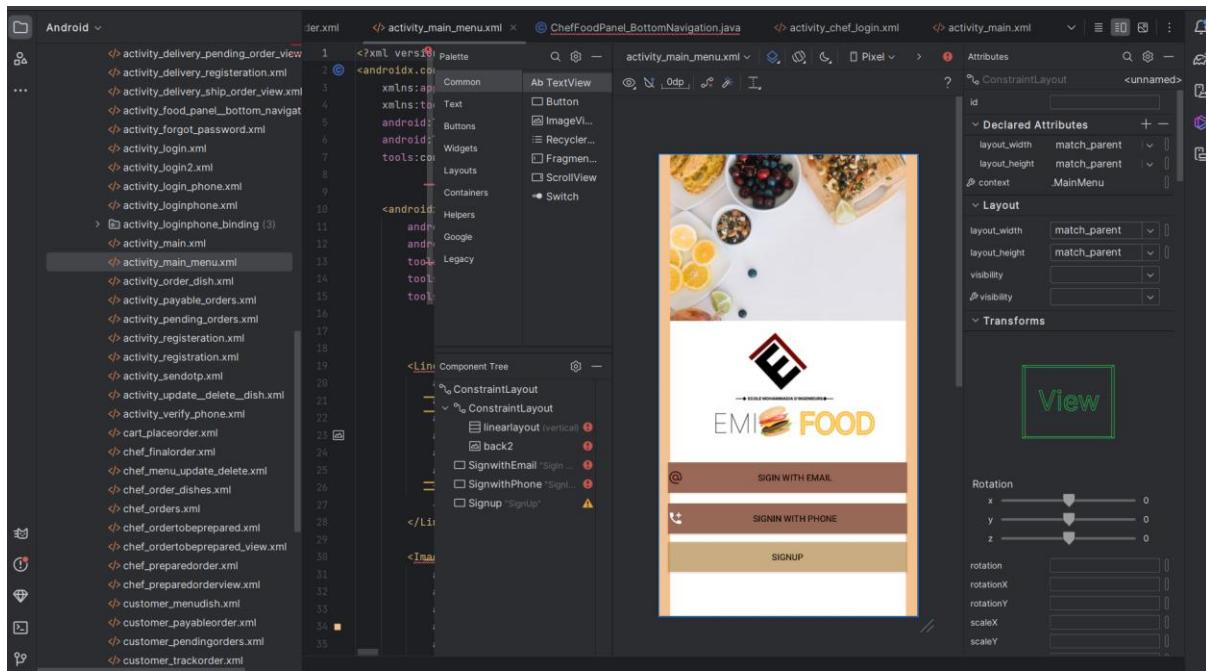


De ce fait, on a commencé par réaliser la première interface de notre application:

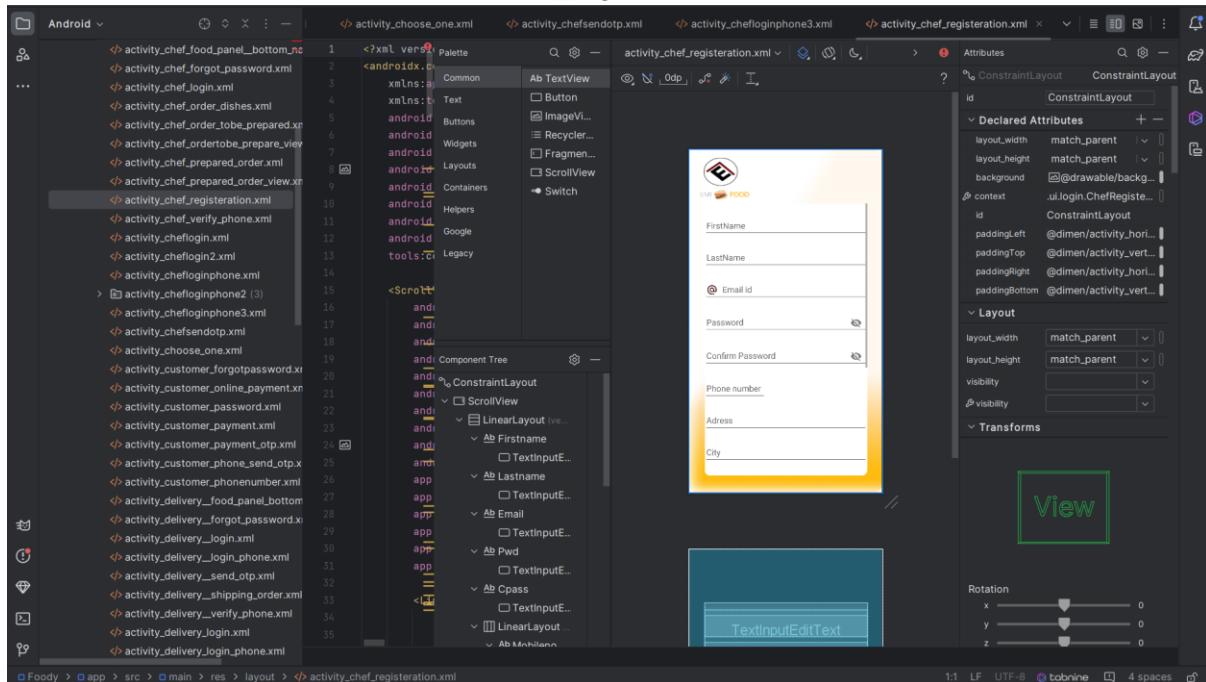


Puis, l'interface de l'activité main_menu:

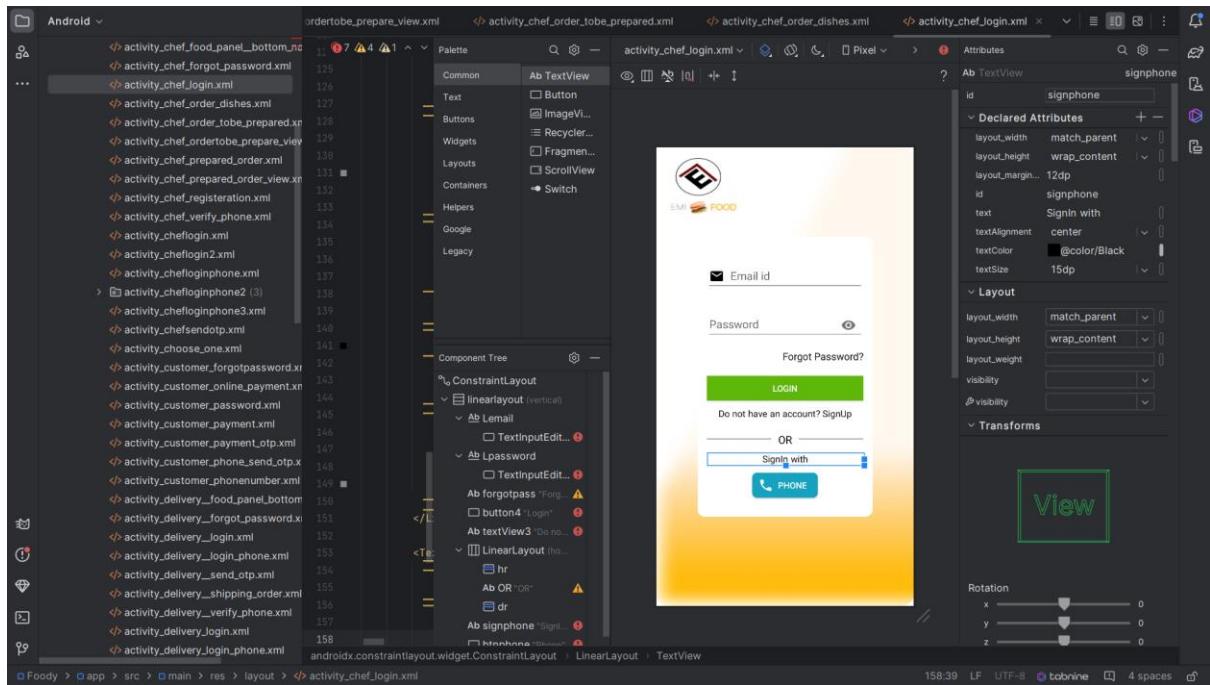
et voici finalement l'interface activity_chef_login:



Sans pour autant oublié l'interface de chef_registration qui est comme suit:



et voici finalement l'interface activity_chef_login:



Conclusion

Au terme de ce projet multidisciplinaire, nous avons parcouru un parcours riche en expériences, explorant les différents aspects du développement logiciel, de la modélisation conceptuelle à l'implémentation pratique. Notre voyage a débuté par la modélisation des systèmes à l'aide de diagrammes UML, une étape cruciale qui nous a permis de visualiser et de conceptualiser la structure et le comportement de notre application.

Grâce aux diagrammes de classes, nous avons pu identifier les entités clés de notre système et les relations entre elles, jetant ainsi les fondations de notre architecture logicielle. Les diagrammes de cas d'utilisation nous ont aidés à comprendre les besoins fonctionnels de nos utilisateurs et à définir les fonctionnalités principales de notre application. Quant aux diagrammes d'activité, ils nous ont permis de modéliser le flux de contrôle à travers les différentes étapes des processus métier de notre système.

Fort de cette modélisation initiale, nous avons ensuite plongé dans le monde de la programmation, en commençant par le langage C. Cette immersion nous a offert une perspective approfondie sur les concepts fondamentaux de la programmation, de la gestion de la mémoire à la logique algorithmique. Bien que notre application finale ait été développée en Java, les connaissances acquises en C ont été précieuses dans notre compréhension des principes de base de la programmation et ont renforcé notre capacité à résoudre des problèmes de manière efficace et élégante.

En parallèle, nous avons exploré le développement d'interfaces utilisateur avec Figma, un outil puissant qui nous a permis de donner vie à nos conceptions graphiques. De la création de maquettes initiales à la conception de détails graphiques, Figma nous a offert une plateforme flexible et collaborative pour concrétiser nos idées et les partager avec notre équipe.

Enfin, nous avons mis en pratique nos compétences acquises en UML, en C et en conception d'interface utilisateur pour développer une application complète en Java. Notre application de réservation de repas pour le restaurant de l'EMI illustre parfaitement l'intégration harmonieuse de ces différents domaines d'expertise. Grâce à une modélisation UML solide, une logique de programmation robuste et une conception d'interface utilisateur intuitive, nous avons créé une solution innovante répondant aux besoins de notre clientèle.

En conclusion, ce projet nous a offert une expérience enrichissante du développement logiciel, de la conception à l'implémentation. En combinant la modélisation conceptuelle, la programmation et la conception d'interface utilisateur, nous avons pu relever avec succès les

défis de ce projet et créer une application fonctionnelle et conviviale. Cette expérience nous a non seulement permis d'acquérir de nouvelles compétences et connaissances, mais elle nous a également préparés à relever les défis futurs du monde de l'informatique avec confiance et détermination.