Groupe 10

20U2746 Sasha Junior Tsamo

20U2757 Chedjoun Kenguep Dave

20U2606 Nenzeko Tedjionang Dongmo Yannis-Arthur

20U2843 Kamda Malvina Eva

20U2933 Kom Tagne Alex Brondon

RAPPORT DU DEVOIR DE PROGRAMMATION ORIENTEE OBJET ET MODELISATION ORTIENTEE OBJET

PROJET D'ALIMENTATION

Système : l'application de nutrition sera en mesure de donner les données concernant une nourriture précise c'est-à- dire ses éléments constitutifs et leur niveau de teneur dans l'aliment ainsi permettre à l'usage de savoir avec précision quelle apport aura la nourriture dans son organisme. De plus, l'application sera en mesure de faire des propositions de repas et menus pour les utilisateurs.

I- ENVIRONNEMENT DE TRAVAIL

1- Installation d'Ubuntu
2- Environnement de modélisation (Android studio)
3- Environnement de développement (Java jdk)
4- Test avec « Hello World »

```java
public class HelloWorld{

    public static void main (string args[]){
        System.out.println("Hello World!");
    }
}
```

Output : Hello World !

II- ACTEURS ET CAS D'UTILISATIONS

1- Acteurs
Ce sont des éléments qui sont à l'extérieur du système et qui interagissent avec lui.

2- Cas d'utilisations par acteurs

Acteurs	Cas Utilisations
Utilisateur général	• Accéder à l'application
Consultant	• Accéder à l'application • Consulter la liste des repas • Consulter les ingrédients d'un repas • Sélectionner un type de repas • Rechercher un repas • Consulter les détails d'un repas

3- Description des cas d'utilisation en utilisant le formalisme textuelle
   - Cas d'utilisation : Rechercher un repas
   - Précondition : le repas doit se trouver dans la base de données
   - Post condition : le repas est retourné
   - Elément déclencheur : cliquez sur le bouton recherché
   - Scenario :

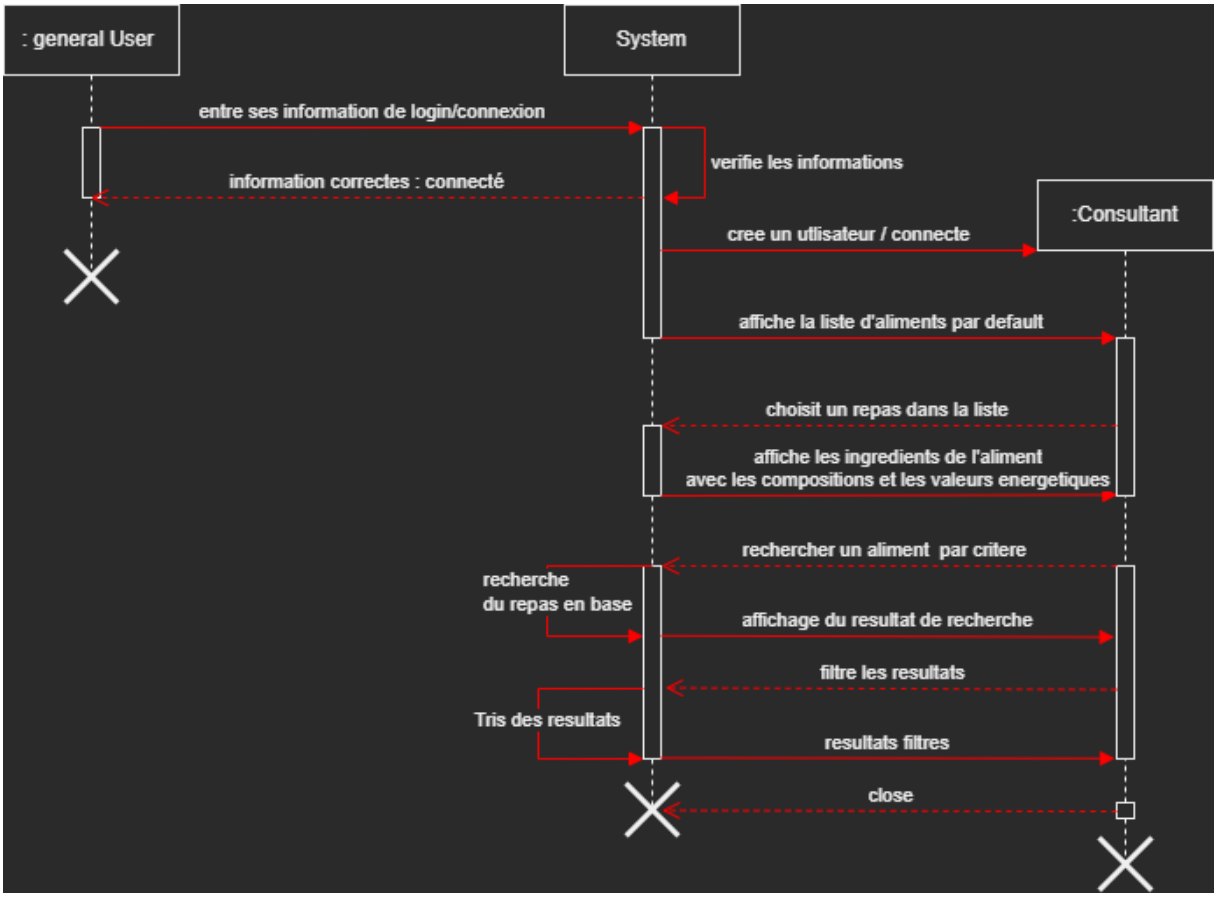
        - Le consultant clique sur la barre de recherche et entre le nom du repas
        - Le système vérifie si le repas est dans la base de données puis renvoie le résultat

- Cas d'utilisation : consulter la liste des repas
- Précondition : se connecter
- Post condition : les liste est visible
- Elément déclencheur : cliquer sur login
- Scenario :
        - Le consultant entre son login et le mot de passe
        - Le système effectue une vérification et affiche la liste

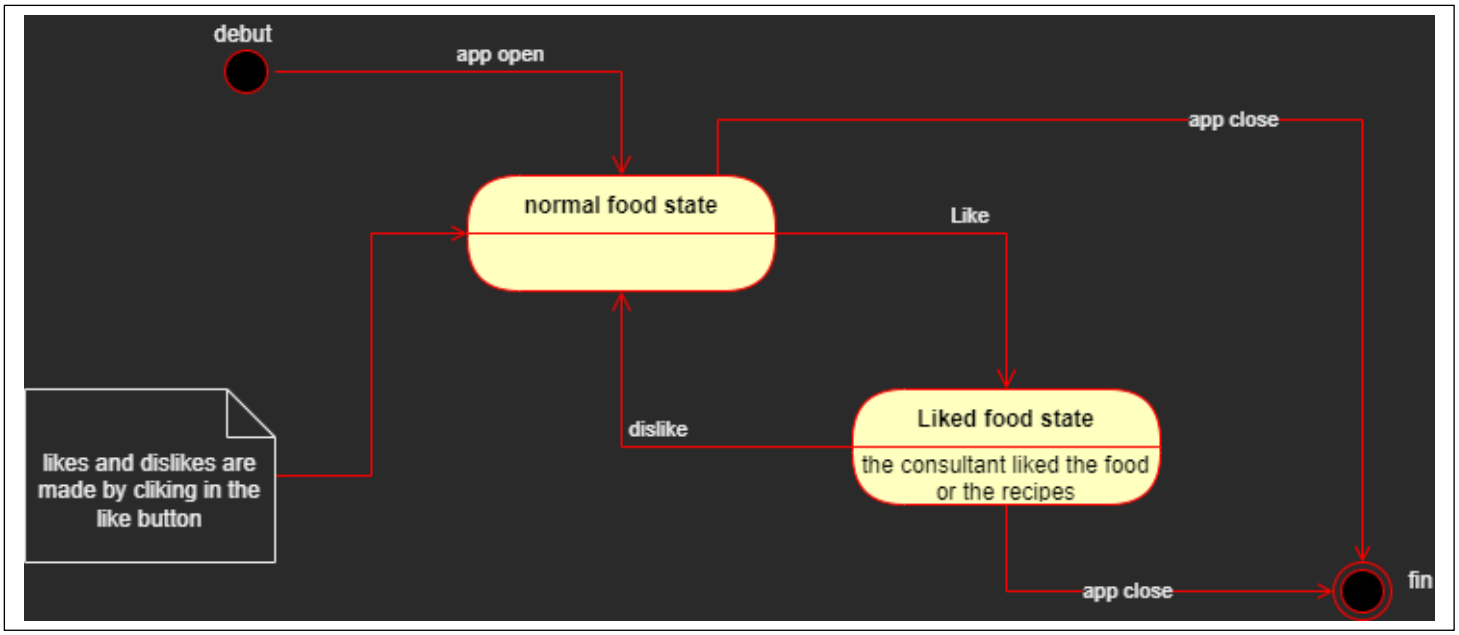
- Cas d'utilisation : consulter les ingrédients d'un repas
- Précondition : l'ingrédient fait partie du repas
- Post condition : les ingrédients sont visibles
- Elément déclencheur : cliquer sur un repas
- Scenario :
        - Le consultant clique sur le repas
        - Le système affiche le repas et les ingrédients

- Cas d'utilisation : sélectionner un type de repas
- Précondition : la catégorie de repas existe dans la base de données
- Post condition :la catégorie est visible
- Elément déclencheur : clique sur le menu déroulant de la page d'accueil
- Scenario :
        - Le consultant clique sur le menu déroulant puis choisit une catégorie
        - Le système vérifie si la catégorie appartient à la base de données puis renvoie tous les repas de cette catégorie

4- Diagramme de séquence



5- Diagramme d'Etat machine

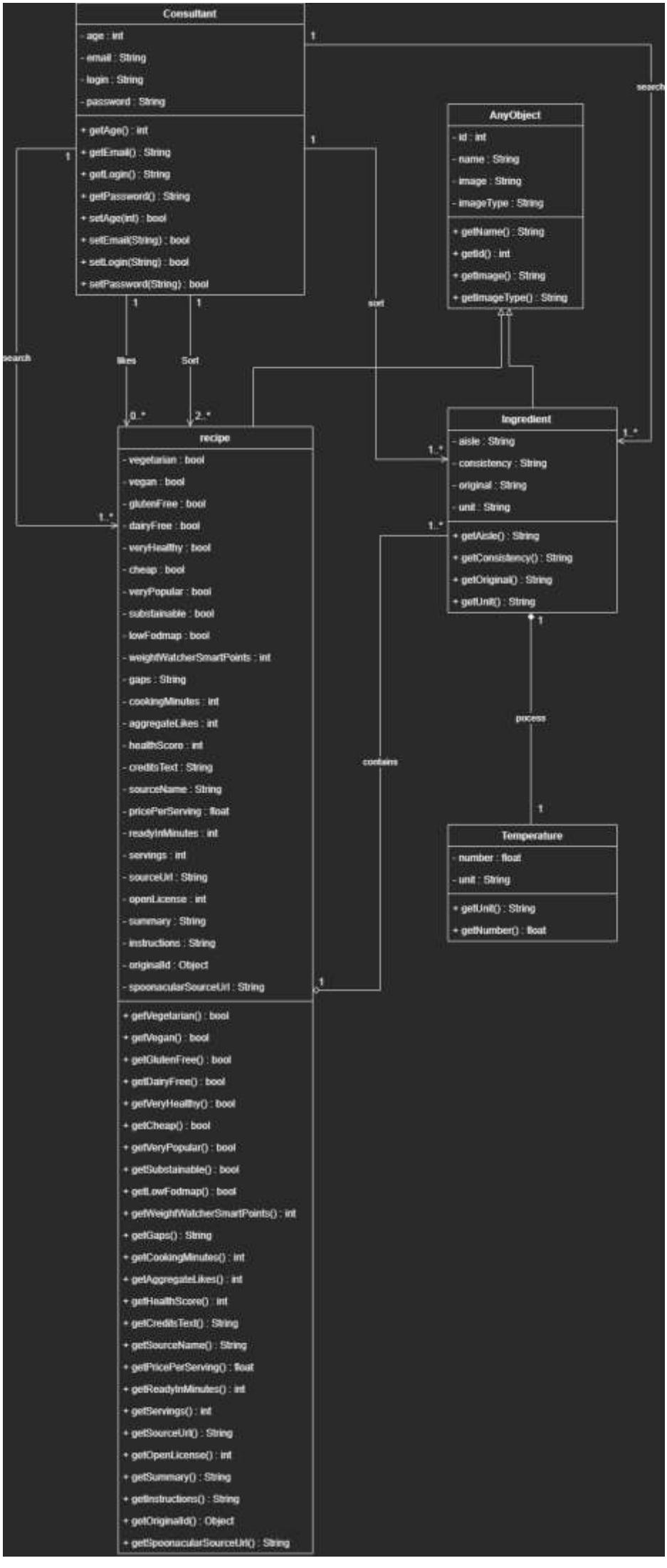


III- DIAGRAMMES DE CLASSES ET OBJETS

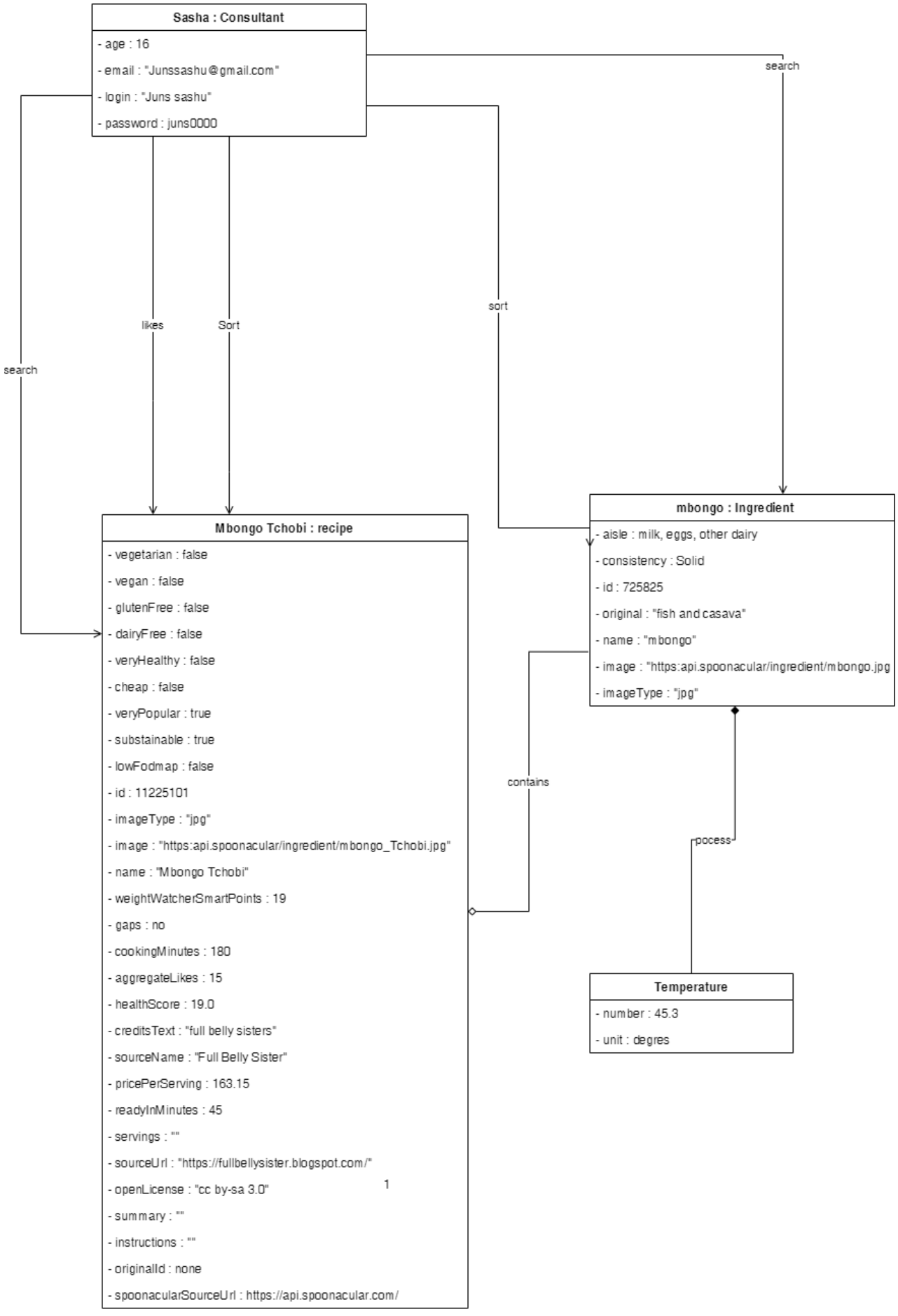
1- Tableau de classes

CLASSES	ATTRIBUTS	OPERATIONS
Consultant	- password : String - login : String - email : String - age : int	+ setPassword(String) : bool + setLogin(String) : bool + setEmail(String) : bool + setAge(int) : bool + getPassword() : String + getEmail() : String + getLogin() : String + getAge() : int
Temperature	- unit : String - number : float	+ getNumber() : float + getUnit() : String
recipe	- vegetarian : bool - vegan : bool - glutenFree : bool - dairyFree : bool - veryHealthy : bool - cheap : bool - veryPopular : bool - substainable : bool - lowFodmap : bool - weightWatcherSmartPoints : int - gaps : String - cookingMinutes : int - aggregateLikes : int - healthScore : int - creditsText : String - sourceName : String - pricePerServing : float - readyInMinutes : int - servings : int - openLicense : int - summary : String - instructions : String - originalId : Object - spoonacularSourceUrl : String	+ getVegetarian() : bool + getVegan() : bool + getGlutenFree() : bool + getDairyFree() : bool + getVeryHealthy() : bool + getCheap() : bool + getVeryPopular() : bool + getSubstainable() : bool + getLowFodmap() : bool + getWeightWatcherSmartPoints() : int + getGaps() : String + getCookingMinutes() : int + getAggregateLikes() : int + getHealthScore() : int + getCreditsText() : String + getSourceName() : String + getPricePerServing() : float + getReadyInMinutes() : int+ getServings() : int + getSourceUrl() : String + getOpenLicense() : int + getSummary() : String + getInstructions() : String + getOriginalId() : Object + getSpoonacularSourceUrl() : String
Ingredient	- unit : String - original : String - consistency : String - aisle : String	+ getUnit() : String + getOriginal() : String + getConsistency() : String + getAisle() : String
AnyObject	- imageType : String - image : String - name : String - id : int	+ getImageType() : String + getImage() : String + getId() : int + getName() : String

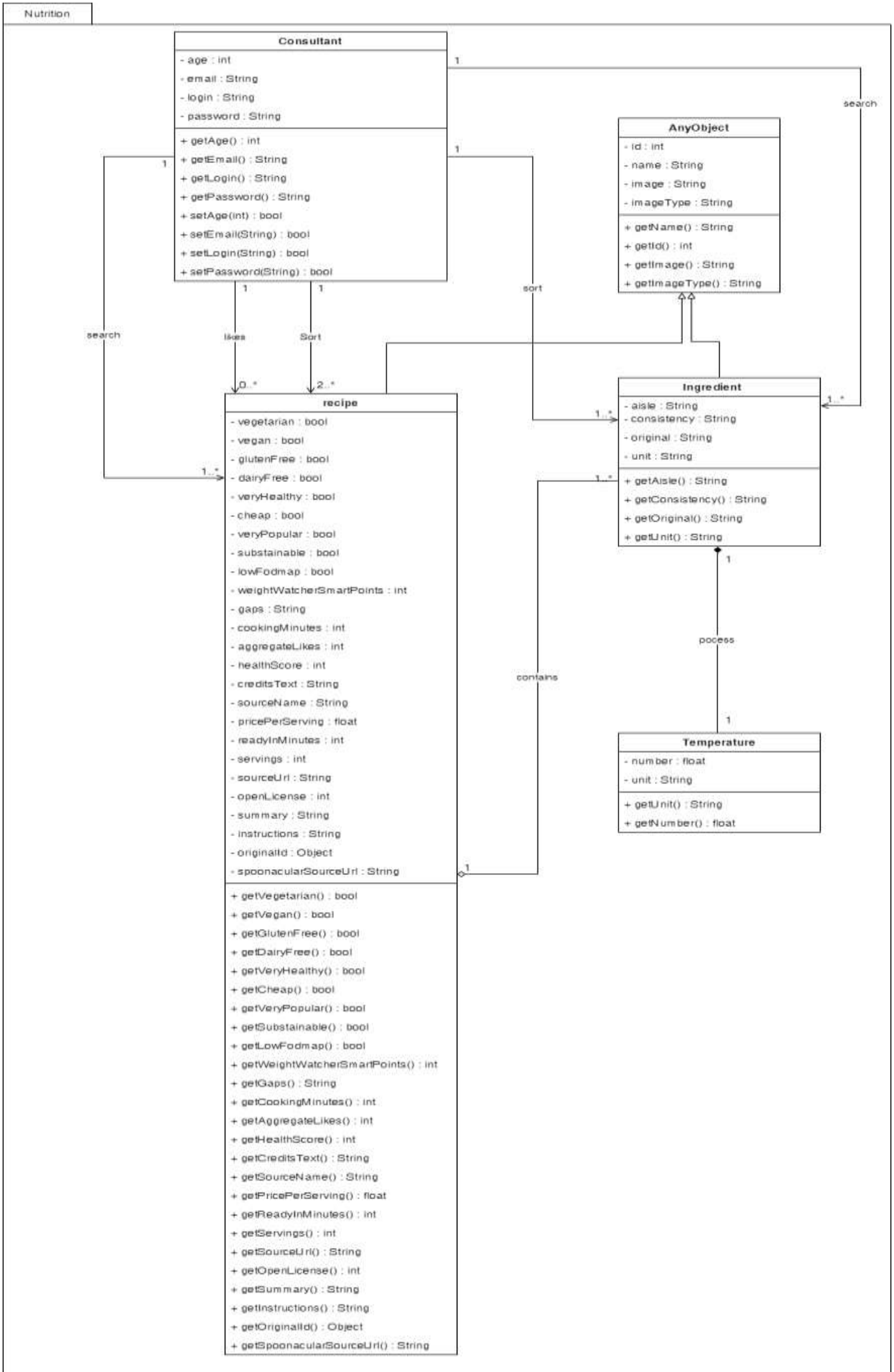
2- Diagramme de classe


Consultant
- age : int
- email : String
- login : String
- password : String
+ getAge() : int
+ getEmail() : String
+ getLogin() : String
+ getPassword() : String
+ setAge(int) : bool
+ setEmail(String) : bool
+ setLogin(String) : bool
+ setPassword(String) : bool
AnyObject
- id : int
- name : String
- image : String
- imageType : String
+ getName() : String
+ getId() : int
+ getImage() : String
+ getImageType() : String
recipe
- vegetarian : bool
- vegan : bool
- glutenFree : bool
- dairyFree : bool
- veryHealthy : bool
- cheap : bool
- veryPopular : bool
- substainable : bool
- lowFodmap : bool
- weightWatcherSmartPoints : int
- gaps : String
- cookingMinutes : int
- aggregateLikes : int
- healthScore : int
- creditsText : String
- sourceName : String
- pricePerServing : float
- readyInMinutes : int
- servings : int
- sourceUrl : String
- openLicense : int
- summary : String
- instructions : String
- originalId : Object
- spoonacularSourceUrl : String
+ getVegetarian() : bool
+ getVegan() : bool
+ getGlutenFree() : bool
+ getDairyFree() : bool
+ getVeryHealthy() : bool
+ getCheap() : bool
+ getVeryPopular() : bool
+ getSubstainable() : bool
+ getLowFodmap() : bool
+ getWeightWatcherSmartPoints() : int
+ getGaps() : String
+ getCookingMinutes() : int
+ getAggregateLikes() : int
+ getHealthScore() : int
+ getCreditsText() : String
+ getSourceName() : String
+ getPricePerServing() : float
+ getReadyInMinutes() : int
+ getServings() : int
+ getSourceUrl() : String
+ getOpenLicense() : int
+ getSummary() : String
+ getInstructions() : String
+ getOriginalId() : Object
+ getSpoonacularSourceUrl() : String
Ingredient
- aisle : String
- consistency : String
- original : String
- unit : String
+ getAisle() : String
+ getConsistency() : String
+ getOriginal() : String
+ getUnit() : String
Temperature
- number : float
- unit : String
+ getUnit() : String
+ getNumber() : float
search
likes
Sort
sort
search
contains
process
1
0..*
2..*
1..*

3- Diagramme d'objet



4- Diagramme de paquets



IV- PSEUDO CODE

```
Définition-structure Consultant{
        password : String;
        login : String;
        email : String;
        age : int;
}
Fontion getAge(): entire{
    Var structure Consultant ^this;
     Retourne this->age;
}

Function getLogin():chaine{
     Var structure Consultant ^this;
      Retourne this->login;
}

Function getEmail():chaine{
      Var structure Consultant ^this;
      Retourne this->email;
}

Function getPassword(){
     Var structure Consultant ^this;
      Retourne this->password;
}
```

V- CONCEPTION DE L'INTERFACE UTILISATEUR

1- Interface utilisateur principle

2- Autres interfaces

FOOD SPACE

Connexion

connect your account to access and have a good meal



Login _____

password _____

Mot de passe oublier ? recupperer

Se connecter →






VI- DEVELOPPEMENT ET INTERFACE UTILISATEUR

Voir l'application Food Space : Food-App-project\application\App

VII- EXERCICES 7 ET 8

1- Exercice 7

Chemin d'acces : Food-App-project\Exercice7.java

2- Exercice 8

Chemin d'acces : Food-App-project\Exercice8.java