C# - Diagramme de classe::DDC

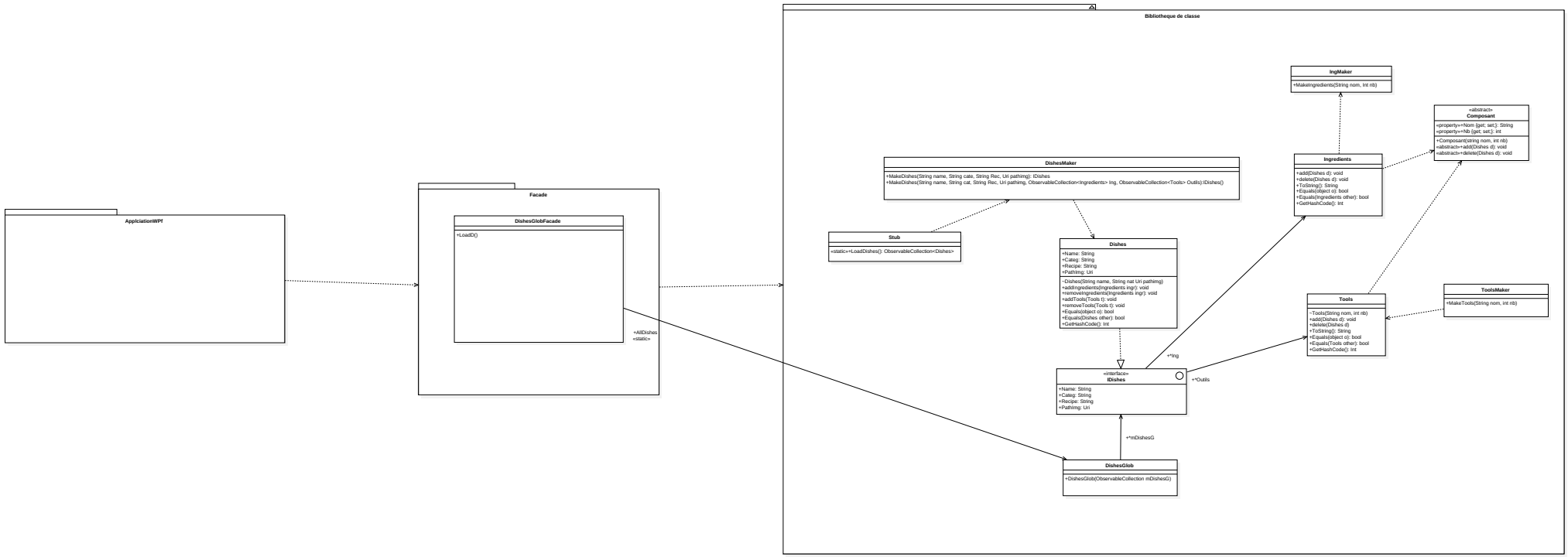
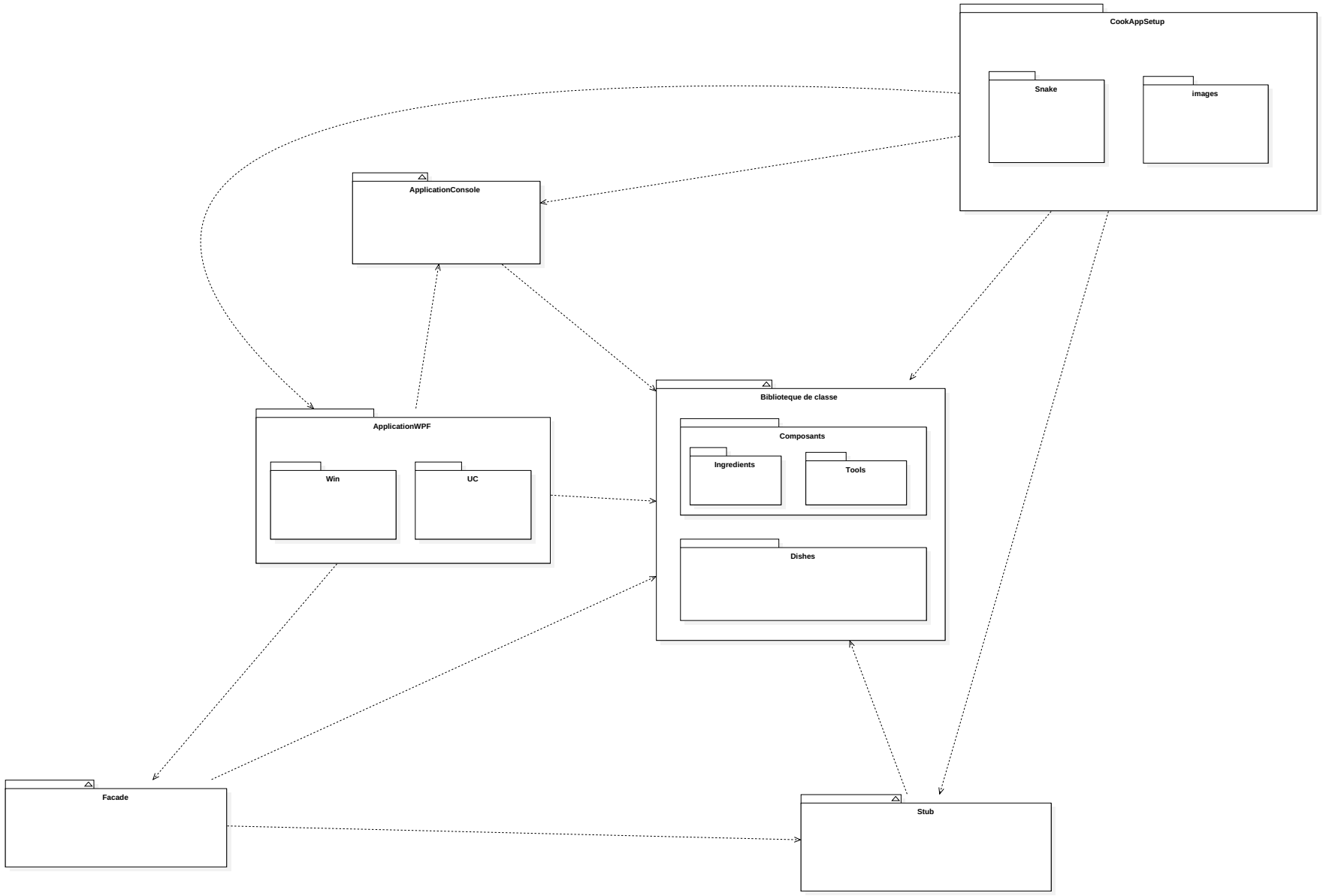


Diagramme de paquetage::DDP



Description de l'architecture

Composants:

Nous avons d'abord choisi de créer une classe abstraite "Composant", de laquelle dérivent les classes "Ingrédients" et "Tools", pour réunir des propriétés comme nom et nb ainsi que des méthodes comme add et delete. Ce qui nous permet également de mettre en internal les constructeurs et ainsi donc empêcher l'instanciation en dehors du modèle.

Ingrédients/Tools :

La classe Ingredients/tools hérite donc de la classe abstraite « Composants » afin de gérer correctement les instanciations. Elle a pour propriétés de base un nom et un nombre.

Dishes :

La classe Dishes contient une observable collection d'ingrédients et une observable collection d'ustensiles. Nous avons choisi de ne pas faire de dictionnaires pour faciliter l'itération mais aussi car la collection observable est très pratique pour le binding. De plus, un plat contient 4 propriétés. Cette classe implémente par ailleurs une interface « IDishes » permettant là aussi de mettre les constructeurs en internal et donc de protéger les données par l'encapsulation.

DishesGlob :

Pour réunir tous nos plats nous avons créé une classe DishesGlob qui contient une observable collection de Dishes.

On a 2 constructeurs : Un de base, par défault. Puis un second qui implémente IEnumerable pour itérer la collection (Linq).

PersistDishes :

Classe permettant de load et save les données. Contenant 2 méthodes : Sérialize and Deserialize, provenant d'une classe abstraite « Persistance » dont hérite PersistDishes. En faisant cela, je falicite l'accès à mon mode de séralization. Si un jour je dois changer la façon de sauvegarder/charger mes données, alors je n'aurais qu'à changer mes méthodes.

Persistance :

Classe abstraite dont hérite PersistDishes. Contient donc le corps des deux méthodes ainsi qu'une troisième permettant de set le répertoire courant.

Stubb :

Classe qui instancie une observable collection de plats et charge toutes les données prévues à l'intérieur.

DishesGlobFacade :

Cette classe me permet de faire en sorte que ma vue n'ai pas à gérer des instances de mon modèle. Elle contient une méthode « LoadD » qui charge les données à partir du stub s'il n'y a pas de fichiers XML présent dans %appdata%.

DishesMaker :

Cette classe me permet de contrôler mes données et d'encapsuler correctement les classes principales. Elle contient 2 méthodes correspondant à 2 types d'instanciation de plats.

IngredientsMaker :

Comme vu juste au dessus, facilite le contrôle des données. Elle contient une méthode qui instancie un ingrédient.

IngredientsMaker :

Comme les précédentes, cette classe contient une méthode qui instancie un outil.