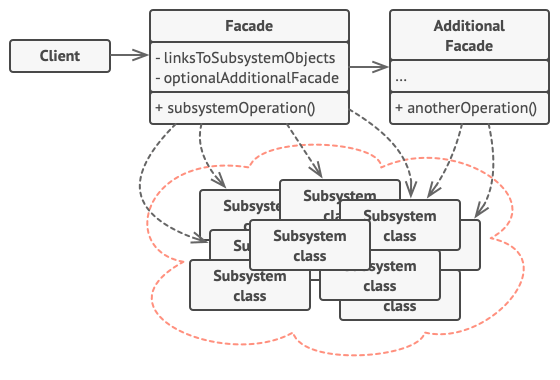
**Description de l’architecture :**

***Patrons de conceptions :***

**La Façade :**

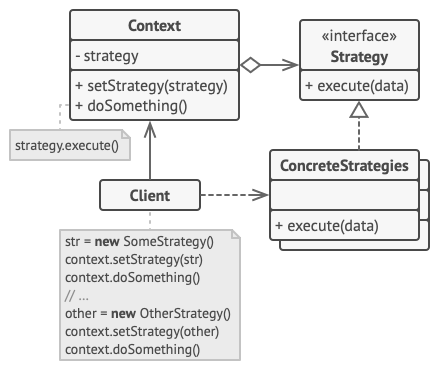
****

Nous avons utilisé le patron de conception nommé **Façade**. Le principe de ce patron de conception est d’avoir un “superviseur” qui connaît toutes les fonctionnalités des sous-systèmes contrairement aux autres sous-systèmes qui ne se connaissent pas entre eux. La façade permet de rediriger les requêtes d’un sous-système vers un autre, elle permet aussi de faire le lien entre la vue et le modèle.

Dans notre projet la classe Manager est celle qui contrôle notre application et la plupart des classes principales. Elle délègue la partie de gestion des utilisateurs au Manager Utilisateur et la partie de gestion des photos au Manager Photo.

Le Manager Utilisateur et le Manager Photo sont eux-mêmes des façades (Façade supplémentaire).

**La Stratégie :**



Nous avons utilisé le patron de conception nommé **Stratégie**. Le principe est de créer une interface qui est implémentée différemment par les stratégies concrètes. Le contexte possède une référence vers une stratégie concrète à travers l’interface. Chaque fois qu’il veut lancer un algorithme, le contexte appelle la méthode d’exécution de l’objet stratégie associé. Le contexte ne sait pas comment la stratégie ne fonctionne ni comment l’algorithme est lancé.

Dans notre projet, l’interface IPersistance sera implémenté par deux stratégies concrètes (JSON et …). Le contexte est notre Manager qui possède une référence vers une stratégie concrète à travers de l’interface IPersistance. Le Manager appelle une méthode

(à compléter !!)

**Découpage de notre architecture :**

**-PictYours**

Ce projet contient toute la partie vue de l’application que ce soit les différentes fenêtres, les user-controls ou encore les converters.

Ce projet dépend de la librairie “MaterialDesignTheme” pour tout ce qui est en rapport avec les vues et de la bibliothèque de classes “BiblioClasse” pour tout ce qui est en rapport avec le Modèle.

**-BiblioClasse**

Ce projet définit toutes les classes communes à notre application. Il n’a aucune référence vers d’autres projets car il n’a pas besoin de les connaître.

**-TestUnitaire**

Ce projet définit les tests unitaires sur certaines classes du modèle “BiblioClasse”, il possède donc une référence vers celui-ci. Les tests unitaires sont réalisés grâce à Xunit donc le projet possède aussi une référence vers cette librairie.

**-Test\_RechercheUtilisateur**

Ce projet est un test fonctionnel de la classe RechercheUtilisateur. Il permet de vérifier le bon fonctionnement des différentes recherches. Il possède donc une référence vers le modèle “BiblioClasse”.

**-Test\_ManagerPhoto**

Ce projet est un test fonctionnel de la classe ManagerPhoto. Il permet de vérifier le bon fonctionnement des différentes méthodes et d’obtenir les résultats attendus. Il possède donc une référence vers le modèle “BiblioClasse”.

**-PictYours.Persistance**

Ce projet définit les classes en rapport avec la persistance, comme le chargement et la sauvegarde de données. Ce projet dépend de la bibliothèque de classe “BiblioClasse”.