GRANGER

Alexis

Licence Professionnelle Projet Web

Rapport Pokedeck

**Sujet :**

Le but du projet est de créer un logiciel permettant à un utilisateur de gérer sa collection de cartes Pokemon. C’est à dire qu’il aura la possibilité de supprimer, ajouter ou éditer une carte de la collection qu’il aura créé.

**Réalisation:**

1. Etude réalisée :

Tout d’abord pour pouvoir travailler avec des Objets carte Pokémon, il a fallu identifier les différentes caractéristiques qu’elles comportent. J’ai retenu celles essentielles à l’identification de ces cartes selon leur type:

* Pokémon :

un nom, un type (ou élément), un nombre de HP, des attaques

* Dresseur (Trainer) :

un nom, un type (ou catégorie), un texte, une règle d’utilisation (trainer rules)

* Energie (Energy) :

un nom, un type (ou élément)

On remarque que 2 caractéristiques sont commune à chaque type de Cartes et donc seront hérités par une classe mère pour éviter les redondances de code.

Ainsi l’utilisateur lorsqu’il voudra modifier ou ajouter une de ses cartes, il aura affaire à 3 formulaires différents selon le type de sa carte. Et de même pour l’affichage de chaque carte qui sera différent pour chacune d’elle.

On comprend que ces cartes seront stockées dans une liste d’un objet qu’on appellera Pokedeck.

Pour réaliser toutes les actions dont à besoin l’utilisateur, j’estime qu’il faudra 3 interfaces :

une page de création ou d’import d’une collection de cartes, une liste de cartes et un formulaire d’édition ou de création.

1. Structure du logiciel :

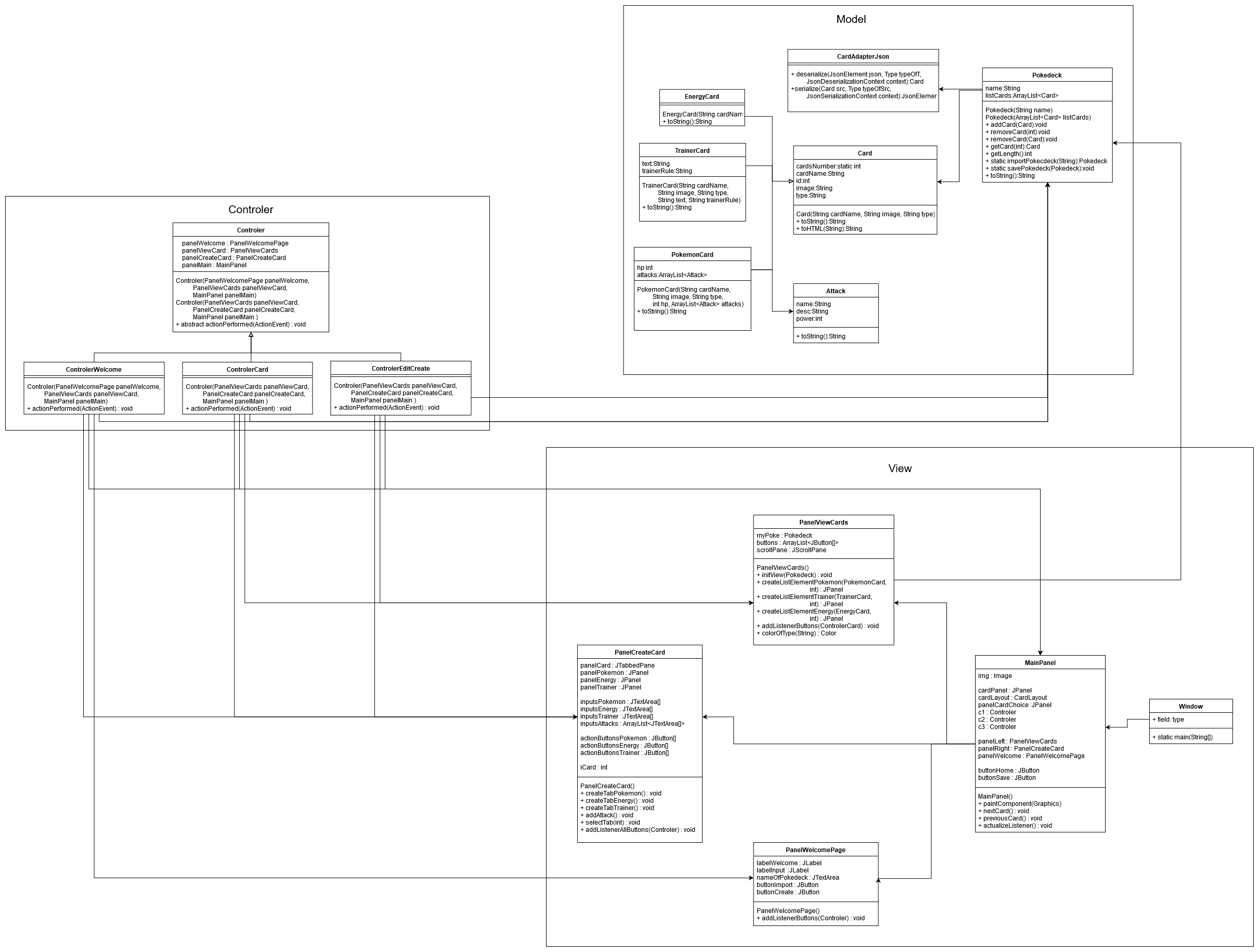


Figure 1 : Diagramme de classe complet

Pour rendre plus clair l’organisation des différentes classes j’opte pour le design pattern MVC (Modele Vue Contrôleur), qui va séparer en 3 catégories les fichiers de mon programme:

Vue(View) :

Les interfaces auxquelles sera confronté l’utilisateur. On retrouve :

* un écran d’accueil avec un input permettant de choisir un nom de collection à importer/créer (PanelWelcomePage)
* un écran présentant la liste de toutes les cartes enregistrées dans cette même collection (PanelViewCards)
* un écran présentant un onglet contenant un formulaire pour chaque type de carte (PanelCreateCard)

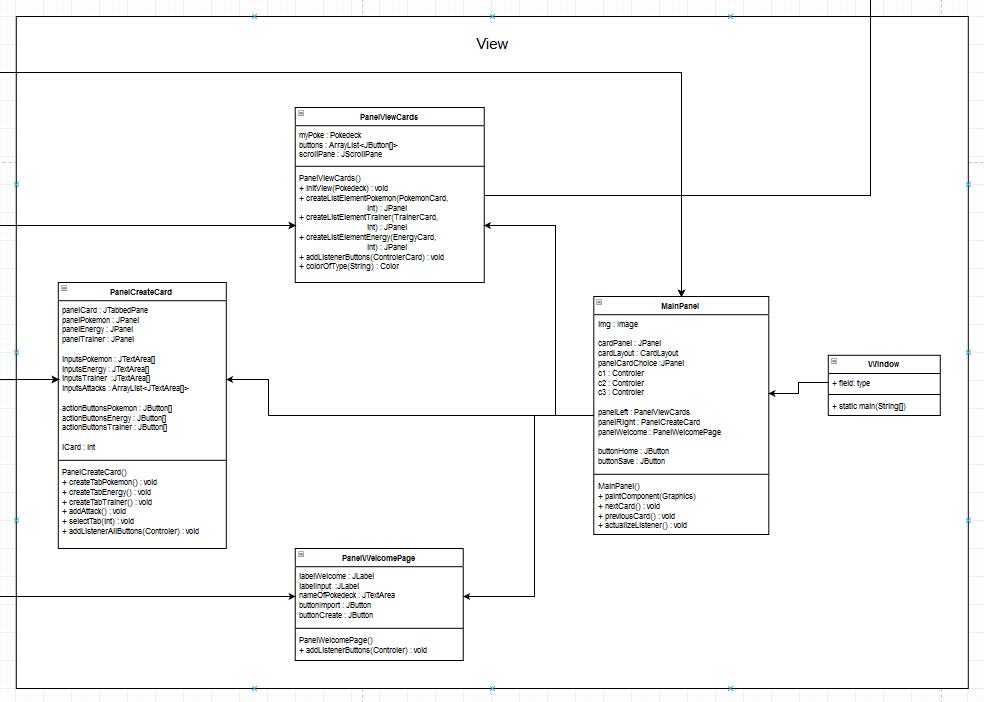


Figure 2 : zoom sur les Vues

Modèle(Model) :

Les structures de données qui seront manipulés :

* la classe mère Card et ses fils PokemonCard, TrainerCard, EnergyCard
* la class Attack qui décrit une attaque d’une carte Pokemon
* la classe Pokedeck permettant de gérer la collection dans sa globalité

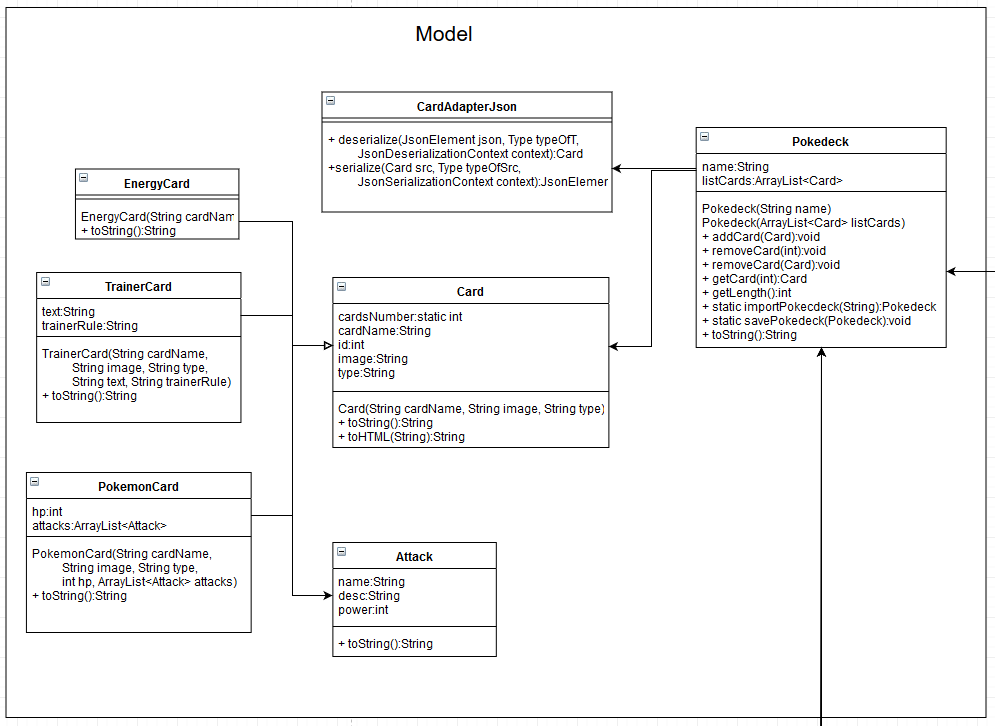


Figure 3 : zoom sur les modèles

Contrôleur (Controler) :

Les classes gérant les actions de l’utilisateur :

* La classe mère Controler et ses fils
* ControlerWelcome gère les actions liées à la page d’accueil et aux redirections faites dans le logiciel
* ControlerCard gère les actions liées à la liste de cartes comme supprimer ou alors éditer qui remplira les champs du formulaire dans PanelCreateCard
* ControlerEditCreate gère les actions liées à la création et édition de carte à partir des formulaires. L’édition n’est vraiment réalisée qu’à partir des formulaires

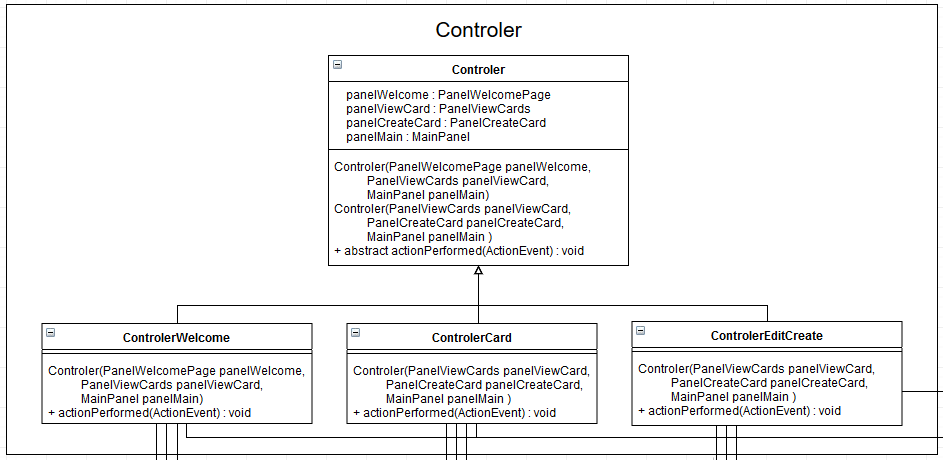


Figure 4 : zoom sur les contrôleurs

1. Ce qui a été fait :

Pour l’instant les fonctionnalités principales du projet sont toutes réalisées. L’utilisateur peut ajouter, modifier, créer des cartes et créer ou importer une de ses anciennes collections.

La sauvegarde des collections se réalisent par sérialisation dans un fichier json, pour cela j’utilise la librairie “gson”.

Des problèmes ont été rencontrés surtout lors de changements dans la liste de cartes. En effet cela nécessite un rafraichissement des JPanels via “revalidate()” et “repaint()” après une nouvelle initialisation de la liste avec le PokeDeck modifié.

De plus maîtriser le layout “GridBagLayout” a demandé un peu de temps pour réaliser un affichage propre.

1. Spécificités pour modification :

Le logiciel a sa limite de flexibilité entre le modèle et la vue. En effet si l’on veut rajouter des champs descriptif pour une carte, il faudra alors modifier également les vues pour les afficher. De même dans un contrôleur pour qu’ils soient pris en compte dans le formulaire de création.

Les champs du formulaires sont identifiés dans le contrôleur par leur index dans le tableau de JTextArea correspondant au type de carte.

Pour ajouter, créer et retirer une carte j’enlève le premier component de mon panel de liste de cartes pour ensuite l’initialisé à nouveau avec la fonction “initView(Pokedeck myPoke)”. En paramètre je lui donne le pokedeck que j’aurais modifié dans mon contrôleur.

La classe MainPanel englobe tout dans un cardLayout. La première card est la page d’accueil, la deuxième card est un JPanel composé de la liste à gauche et du formulaire à droite.

1. En amélioration :

Outre le fait de compléter les caractéristiques des cartes, on peut envisager d’autres améliorations :

* un changement de couleur selon l'élément d’une carte afin que l’on comprenne d’un coup d’oeil cette caractéristique. (fonction avec un switch créée mais les codes RVB restent à être renseignés)
* un écran qui permet de visualiser l’image de la carte, on peut imaginer un champs image de type String qui contient le chemin d’accès à l’image. Puis on a simplement à l’afficher dans une popup. (champs existant mais pas utilisé)
* un outil de recherche et de filtre de carte dans sa collection. Un petit input qui cherche par nom ou autres caractéristiques ses cartes.
* une option de trie
* un aperçu des collections déjà créer dans une comboBox par exemple. Actuellement il faut regarder si un fichier json du même nom existe