



Projet de programmation

Skill Expert : Gestionnaire des compétences du personnel

## **Diagrammes UML**

23/05/2017

HENTATI Mahdi – TOUTAIN David – UZAN Lucas

L3 MIAGE – FA

# Table des matières

- Partie Interface : .....3**
  - Lancement de l’application : ..... 3
  - L’accueil de l’application : ..... 3
  - Les formulaires : ..... 3
  - La page des compétences : ..... 3
  - La page des employés : ..... 3
  - La page des missions : ..... 4
  - La page des formations : ..... 4
- Partie CSV: .....9**
- Partie Recommandation : .....13**
  - Présentation : ..... 13
  - Principe : ..... 13
  - Fonctionnement : ..... 13
  - Algorithme : ..... 14
  - Résultat : ..... 15

## Partie Interface :

### Lancement de l'application :

Au lancement de l'application, Program.java est exécuté : il charge les informations des fichiers csv, attribue les compétences aux employés pour les formations qui sont terminées puis affiche la fenêtre principale de notre programme avec la classe ProgramFrame.java. Ce dernier va ajouter à son interface le JPanel Header.java qui se décompose en deux parties : l'en tête de l'interface avec la navigation et le corps de l'interface. Le changement de pages s'effectue au travers de cette classe qui relie donc les actions utilisateur sur la navigation et l'affichage de la page souhaitée.

### L'accueil de l'application :

La page d'accueil de notre programme est chargée à partir de la classe Accueil.java. Elle se compose d'un tableau de missions actuellement en cours, de missions à venir dans le mois ainsi qu'un tableau composé d'alertes avec les missions où les employés ne sont pas affectés ainsi que celles qui sont en retard. Notons qu'un double clic sur une mission en retard permet de lui attribuer une date de fin réelle. Un diagramme camembert avec la répartition des missions par statut est également présent. Toutes ces informations permettent à l'utilisateur d'avoir un récapitulatif visuel de l'état des missions.

### Les formulaires :

La classe Formulaire.java est la classe parente de toutes les pages suivantes. Elle prépare le fond de formulaire, met en place les boutons d'actions « Nouveau », « Modifier », « Supprimer », « Enregistrer » et « Annuler » et propose deux ArrayList destinées à stocker les composants graphiques qui devront être alternativement désactivés et activés. De plus, chacune des fenêtres suivantes propose un système de filtre avec une barre de recherche et / ou un filtre par statut permettant à l'utilisateur d'arriver plus rapidement à l'objet recherché.

### La page des compétences :

Chargé à partir de la classe Compétences.java, la page offre à l'utilisateur les possibilités de gérer des compétences ainsi que les libellés de chacune. Il peut mettre deux libellés pour chaque compétence : un en français et un en anglais. Le changement du libellé affiché par défaut pour une compétence peut être rapidement mis en place. Le code d'une compétence doit impérativement respecter les normes Caractère.Chiffre. pour garder une homogénéité. Il ne doit également pas être déjà présent.

### La page des employés :

Chargé à partir de la classe Personnel.java, la page offre à l'utilisateur les possibilités de gérer des employés et les compétences que l'employé possède. Il peut mettre un nom, un prénom, une date d'entrée et de lui attribuer les compétences souhaitées. Le nom, le prénom et la date d'entrée doivent être remplis pour enregistrer.

## La page des missions :

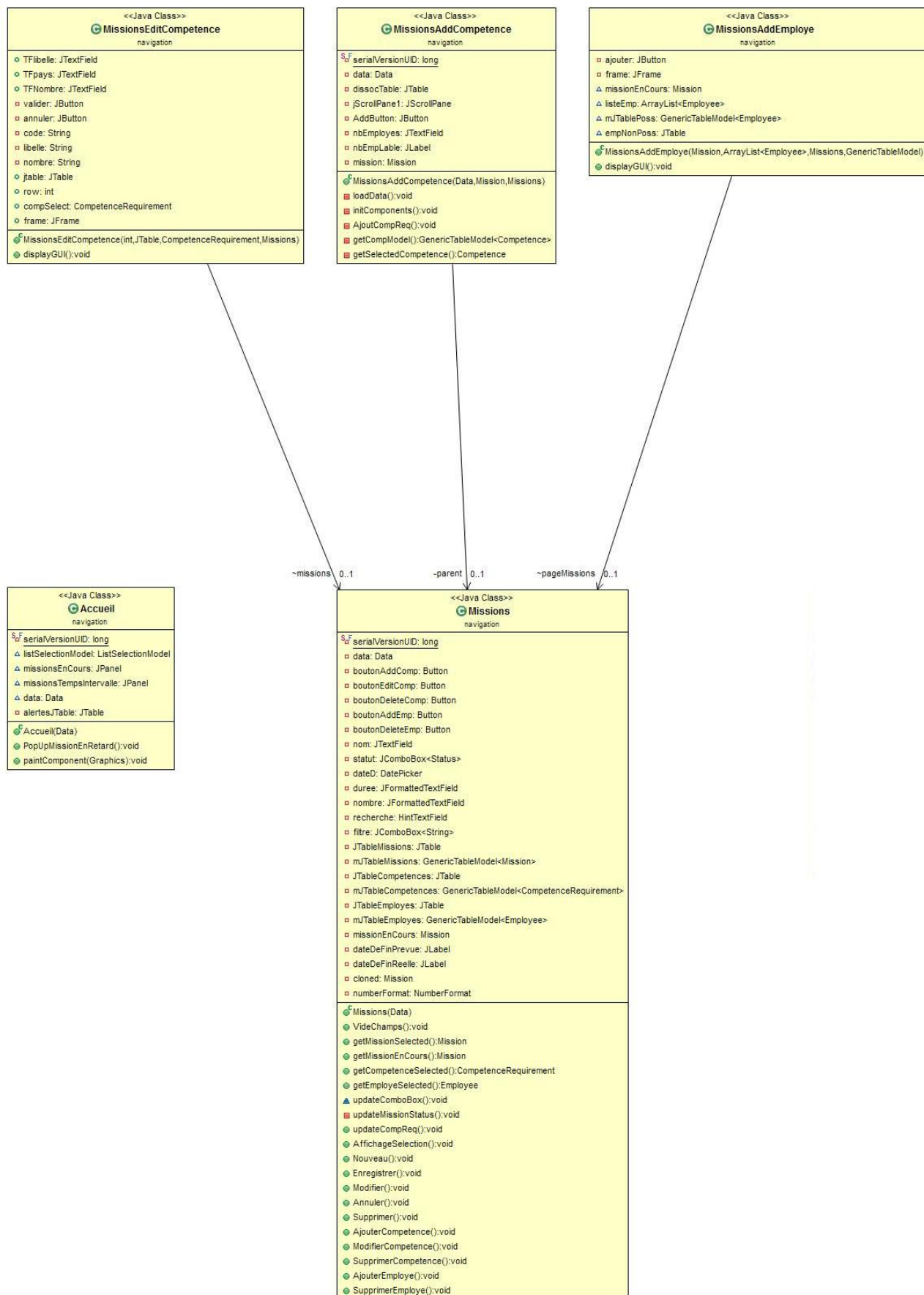
Chargé à partir de la classe Missions.java, la page offre à l'utilisateur les possibilités de gérer des missions, les compétences requises par la mission et les employés affectés. Il peut mettre un nom de mission, le nombre de personnes requises, la date de début de la mission, la durée en jours, les compétences requises et les employés affectés. Il peut également forcer la planification d'une mission en préparation qui n'a pas le nombre d'employés requis si cela est nécessaire.

Pour l'ajout d'un employé, un système de recommandation permet à l'utilisateur de savoir quel employé est le plus apte à exécuter la mission (plus d'informations à propos du fonctionnement des recommandations dans la partie concernée).

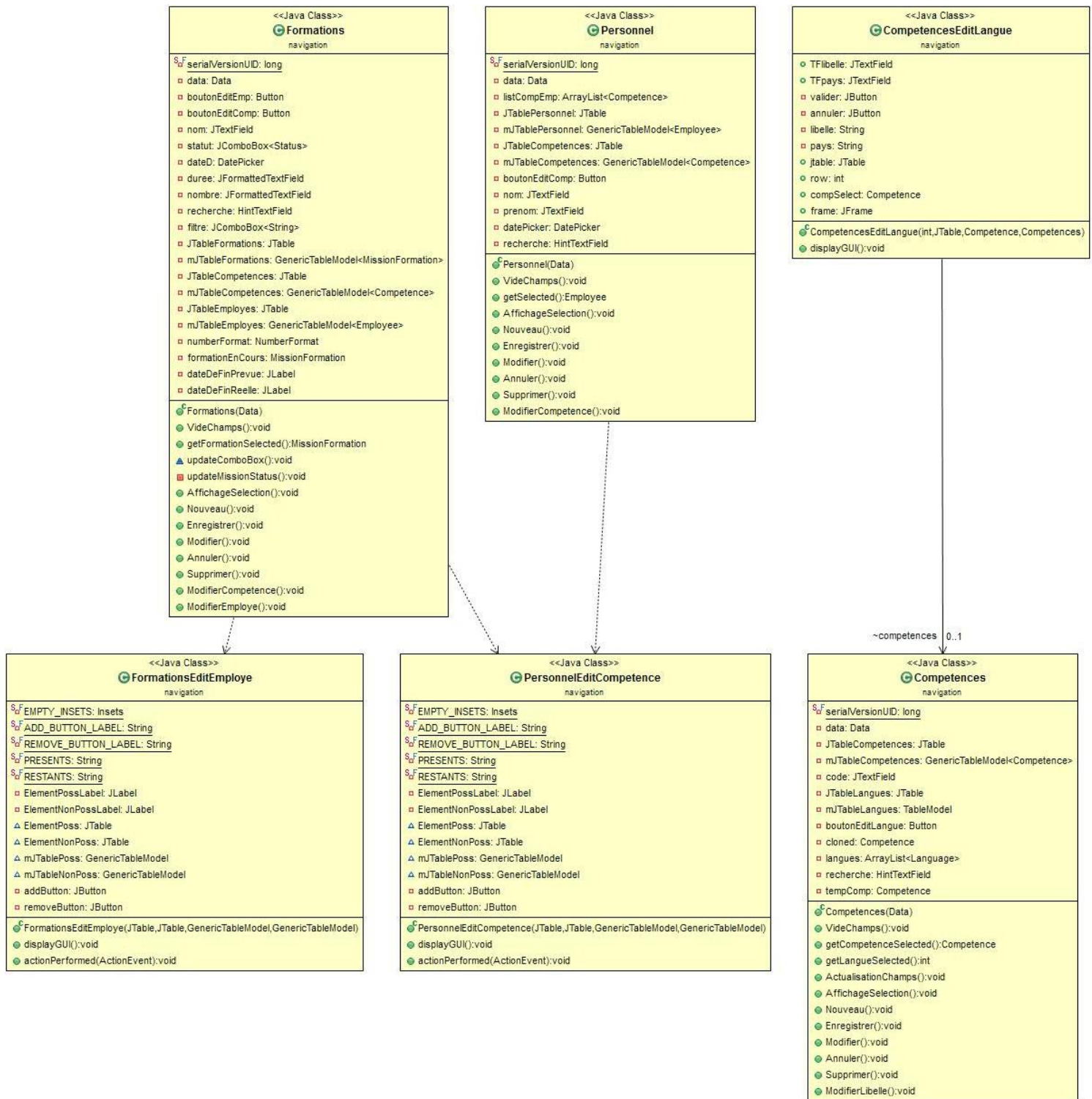
La date de fin prévue est calculée en interne en ajoutant la durée en jours à la date de début puis elle est affichée à l'écran. La date de fin réelle est initialisée à la date de fin prévue mais peut être destinée à être modifiée si la mission finie avant ou après ce qui était prévue. Elle est aussi affichée et permet d'avoir une trace du déroulement des missions.

## La page des formations :

Chargée à partir de la classe Formations.java, la page offre à l'utilisateur les possibilités de gérer des missions de formation, les employés destinés à être formés et la liste des compétences qui leurs seront affectés automatiquement à la fin de la formation. La formation a exactement les mêmes informations qu'une mission (puisque c'est un type de mission !). L'attribution automatique des compétences aux employés concernés lorsqu'une mission est terminée se fait au lancement de l'application.

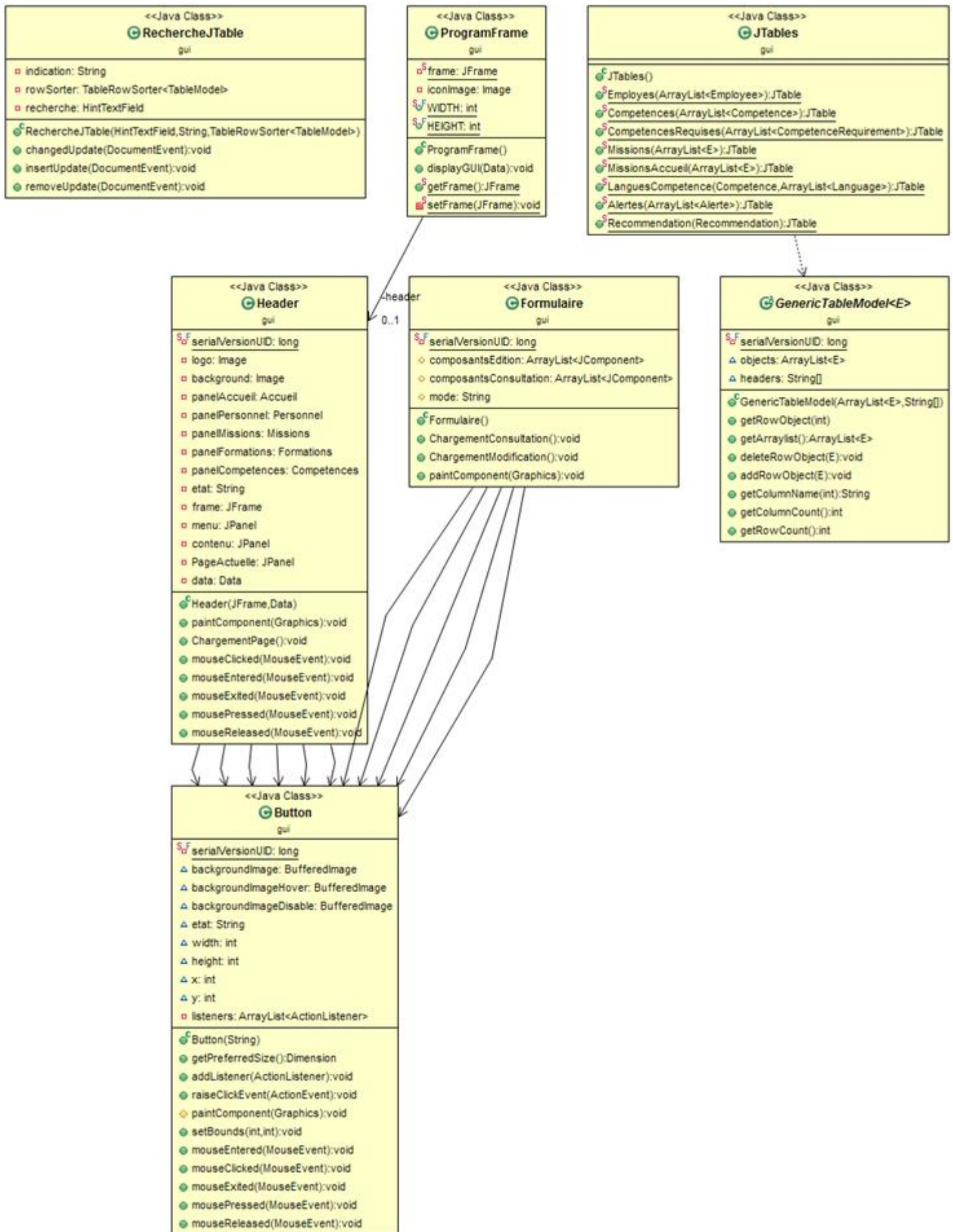


Package « Navigation » : Panneaux accueil et gestion des missions.



Package « Navigation » : Panneaux de gestion des compétences, du personnel, et des formations.





Package « GUI » : Formulaires et boutons de navigation



Package « Models » : Classes de données



## Partie CSV:

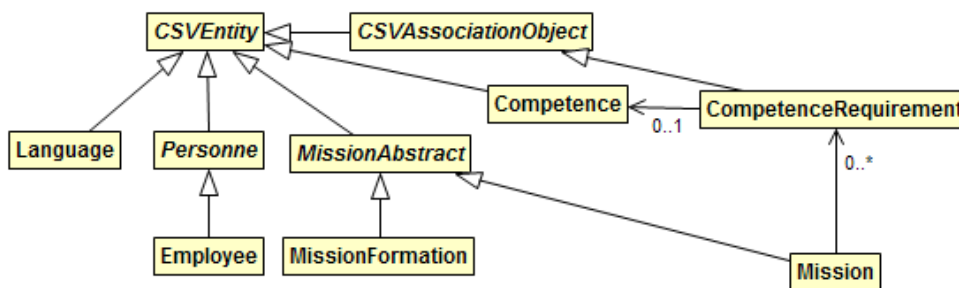
Ce système permet de gérer et synchroniser automatiquement :

- Les dépendances entre les objets de données chargés en mémoire (grâce aux méthodes de CSVEntity)
- leur représentation dans les fichiers CSV
- la mise en « cache » (objets stockés par référence)

CSVObjects fournit les méthodes de lecture / écriture des objets de données et gère le cache. Les informations nécessaires pour le bon fonctionnement du package sont contenues dans CSVConfig :

- CSVModel qui contient les relations entre les classes d'entité et les fichiers CSV correspondants
- Sérialiseurs et désérialiseurs des objets
- Gestionnaire de cache

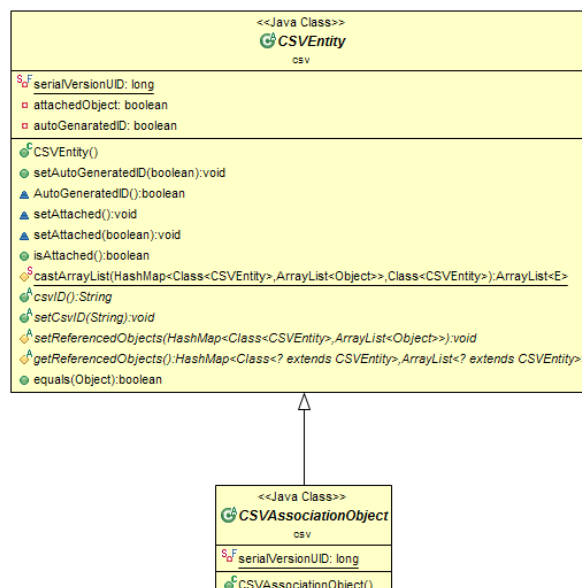
Cette configuration est encapsulée par l'objet Data qui est créé dans la classe Main et injecté par constructeur à toutes les classes qui dépendent des données persistantes.

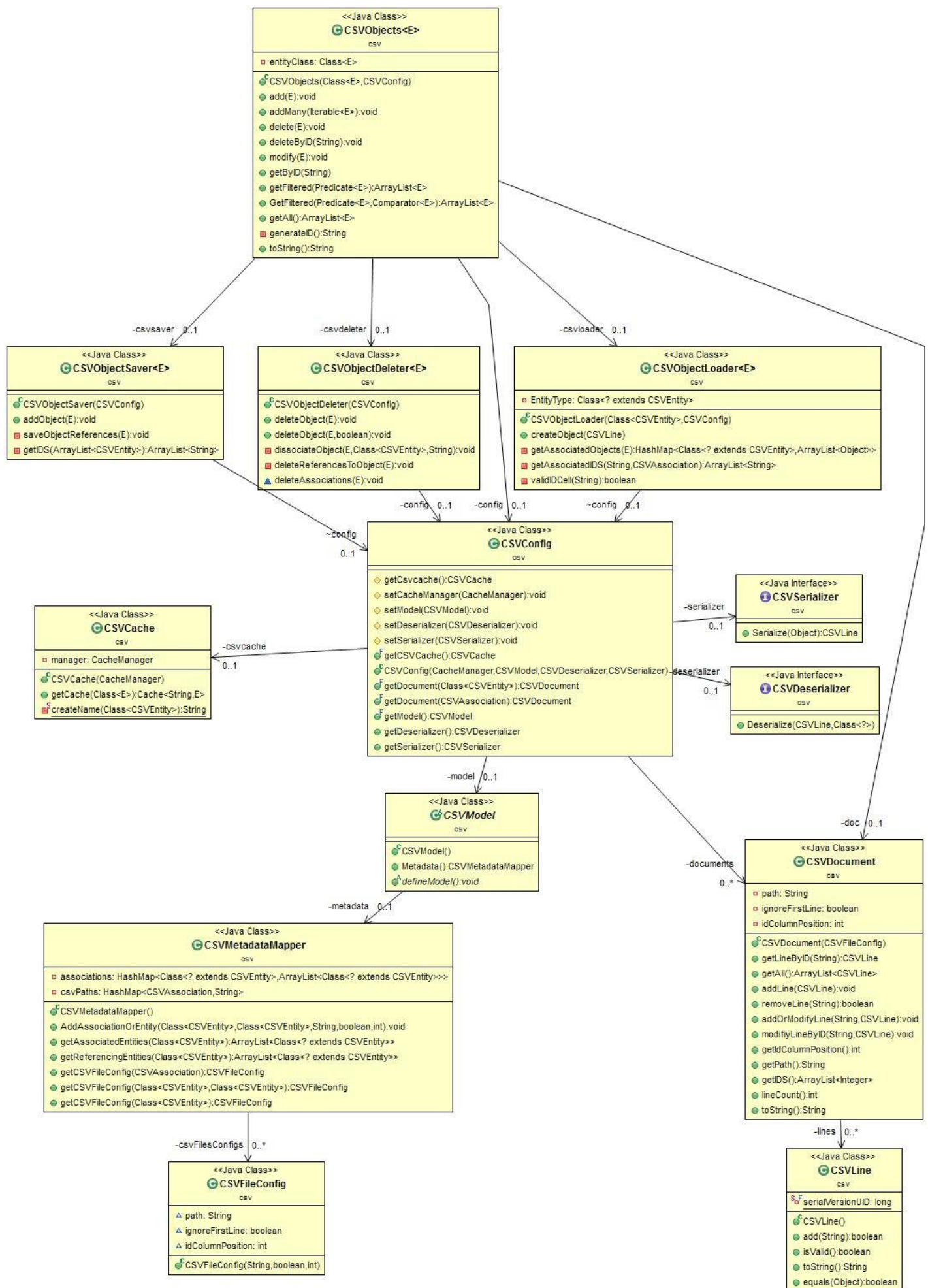


### Persistence des données

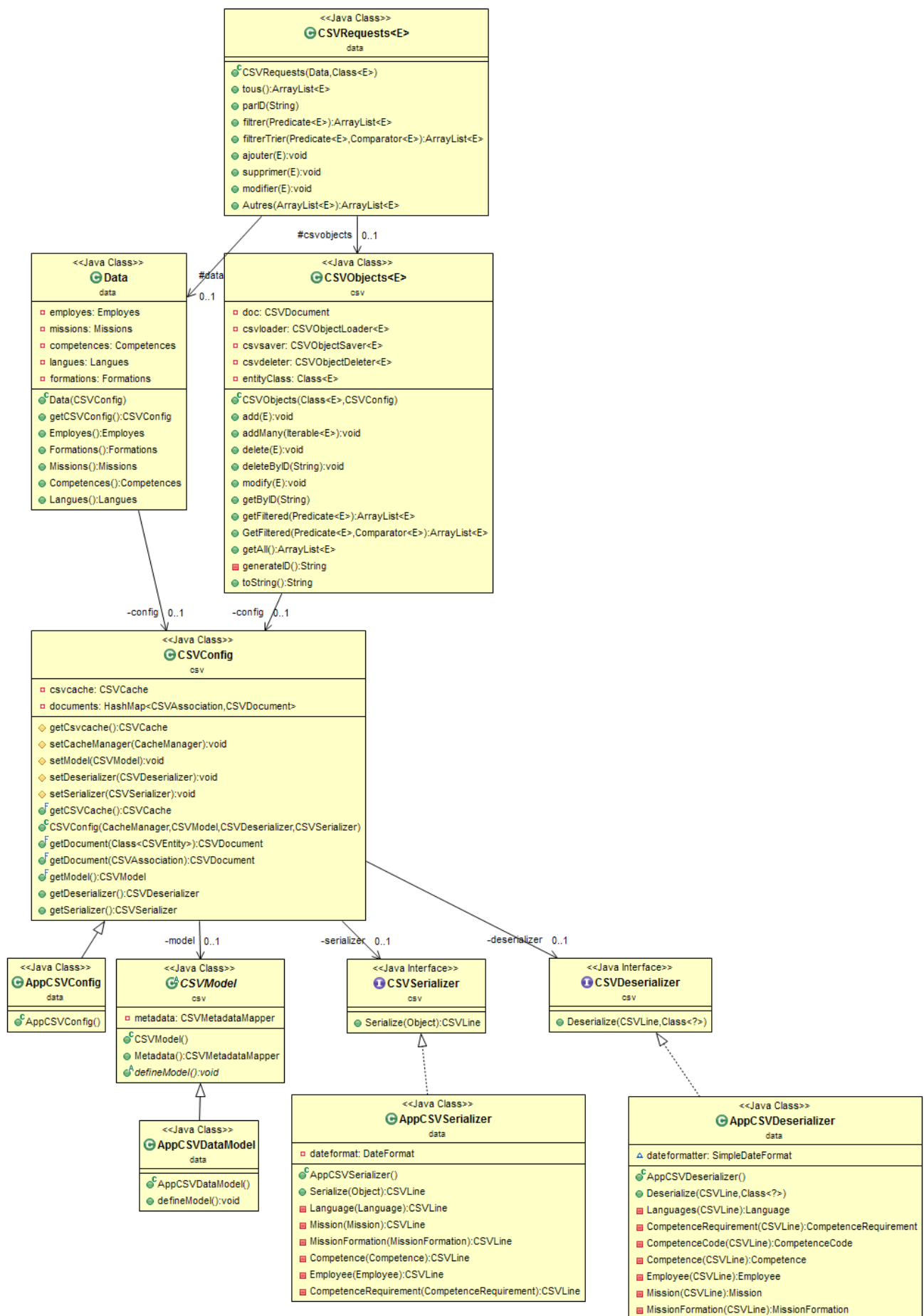
**CSVEntity** : Classe parente de toutes les classes d'entités du programme.

**CSVAssociationObject** : Classe parente de toutes les classes d'associations ayant au moins une propriété.

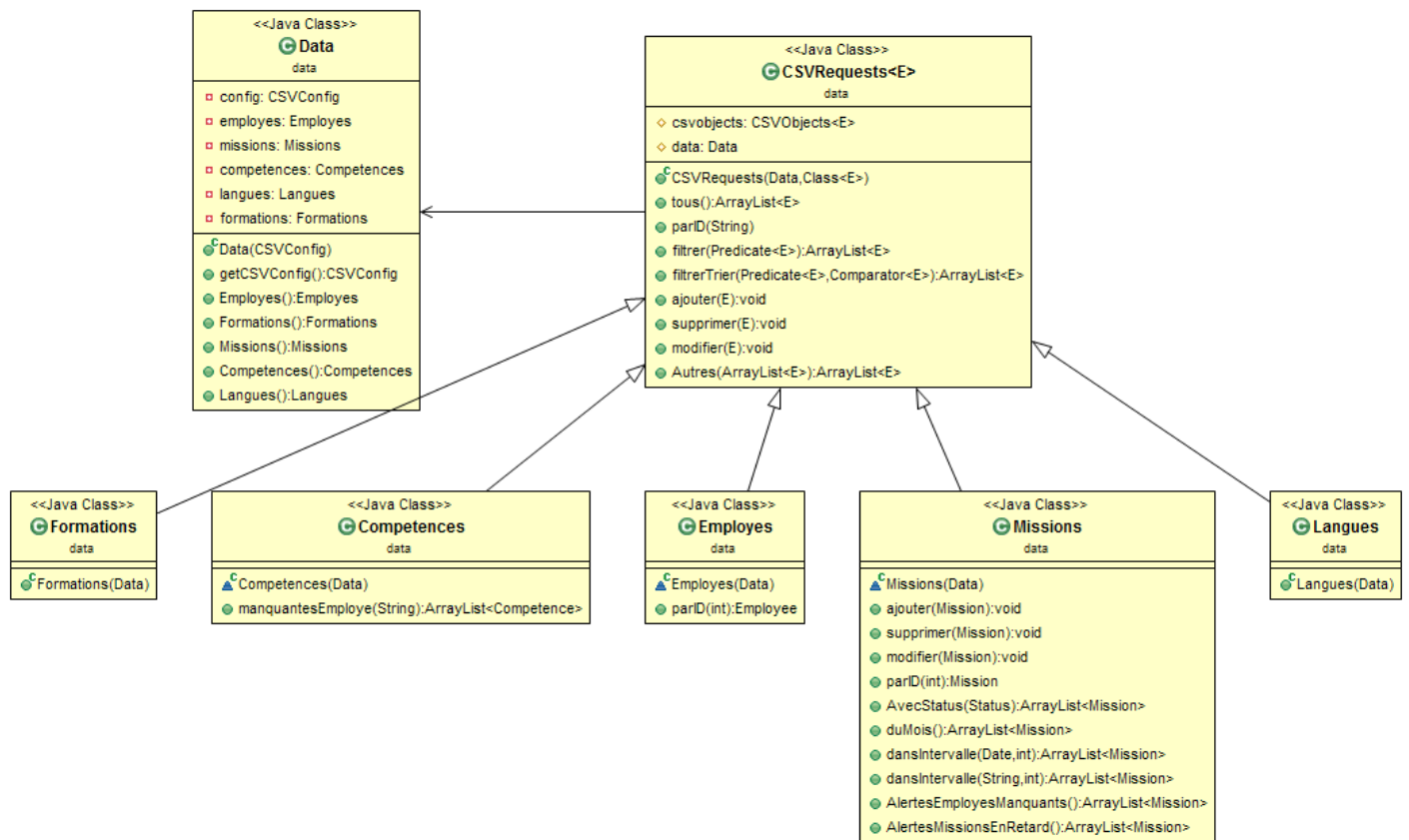




Package « CSV »



Package « Data » : Configuration du modèle et de la sérialisation des données de l'application



Package « Data » : Interface d'accès aux données

## Partie Recommandation :

### Présentation :

Le principe des recommandations est simple, il s'agit lors de l'affectation des employés aux missions de dire quels sont les personnes qui sont le plus aptes à travailler sur cette mission

### Principe :

L'algorithme qui nous sert à déterminer qui sont ces employés se base sur les compétences qui sont nécessaires et les employés qui les ont. Il est aussi calculé un degré de vérification pour le non gaspillage des ressources.

### Fonctionnement :

On regarde combien de compétences ont les employés au total, on vérifie ensuite quels sont leurs compétences puis on les compare aux compétences nécessaires sur la mission et nous prenons le rapport entre les compétences totales de l'employés, les compétences nécessaires qu'a l'employé et une fois convertie en pourcentages est transmis à l'interface pour donner un pourcentage de correspondance entre l'employé et les missions.





## Algorithme :

### Code de l'algorithme

#### ▼ VARIABLES

- nombre\_de\_compétences\_employé EST\_DU\_TYPE NOMBRE
- nombre\_de\_compétences\_employé\_nécessaires EST\_DU\_TYPE NOMBRE
- compétences\_nécessaires\_mission EST\_DU\_TYPE LISTE
- cpt EST\_DU\_TYPE NOMBRE
- compétences\_employé EST\_DU\_TYPE LISTE
- liste\_employés EST\_DU\_TYPE LISTE
- cpt2 EST\_DU\_TYPE NOMBRE
- cpt3 EST\_DU\_TYPE NOMBRE
- liste\_recomm\_employés EST\_DU\_TYPE LISTE

#### ▼ DEBUT\_ALGORITHME

- LIRE nombre\_de\_compétences\_employé
- cpt2 PREND\_LA\_VALEUR 0
- cpt PREND\_LA\_VALEUR 0
- cpt3 PREND\_LA\_VALEUR 0

#### ▼ TANT\_QUE (liste\_employés[cpt] à des lignes) FAIRE

- DEBUT\_TANT\_QUE

#### ▼ TANT\_QUE (compétences\_nécessaires\_mission[cpt2] à des lignes) FAIRE

- DEBUT\_TANT\_QUE

#### ▼ TANT\_QUE (compétences\_employé[cpt3] à des lignes) FAIRE

- DEBUT\_TANT\_QUE

#### ▼ SI (compétences\_employé[cpt3] = compétences\_nécessaires\_mission[cpt2]) ALORS

- DEBUT\_SI

- nombre\_de\_compétences\_employé\_nécessaires PREND\_LA\_VALEUR nombre\_de\_compétences\_employé\_nécessaires ++

- FIN\_SI

- cpt3 PREND\_LA\_VALEUR cpt3 ++

- FIN\_TANT\_QUE

- cpt2 PREND\_LA\_VALEUR cpt2 ++

- FIN\_TANT\_QUE

- liste\_recomm\_employés[cpt] PREND\_LA\_VALEUR (nombre\_de\_compétences\_employé\_nécessaires/nombre\_de\_compétences\_employé)\*100

- cpt PREND\_LA\_VALEUR cpt ++

- FIN\_TANT\_QUE

- cpt PREND\_LA\_VALEUR 0

#### ▼ TANT\_QUE (liste\_recomm\_employés[] à des lignes) FAIRE

- DEBUT\_TANT\_QUE


- AFFICHER liste\_recomm\_employés[cpt]

- cpt PREND\_LA\_VALEUR cpt ++

- FIN\_TANT\_QUE

- FIN\_ALGORITHME

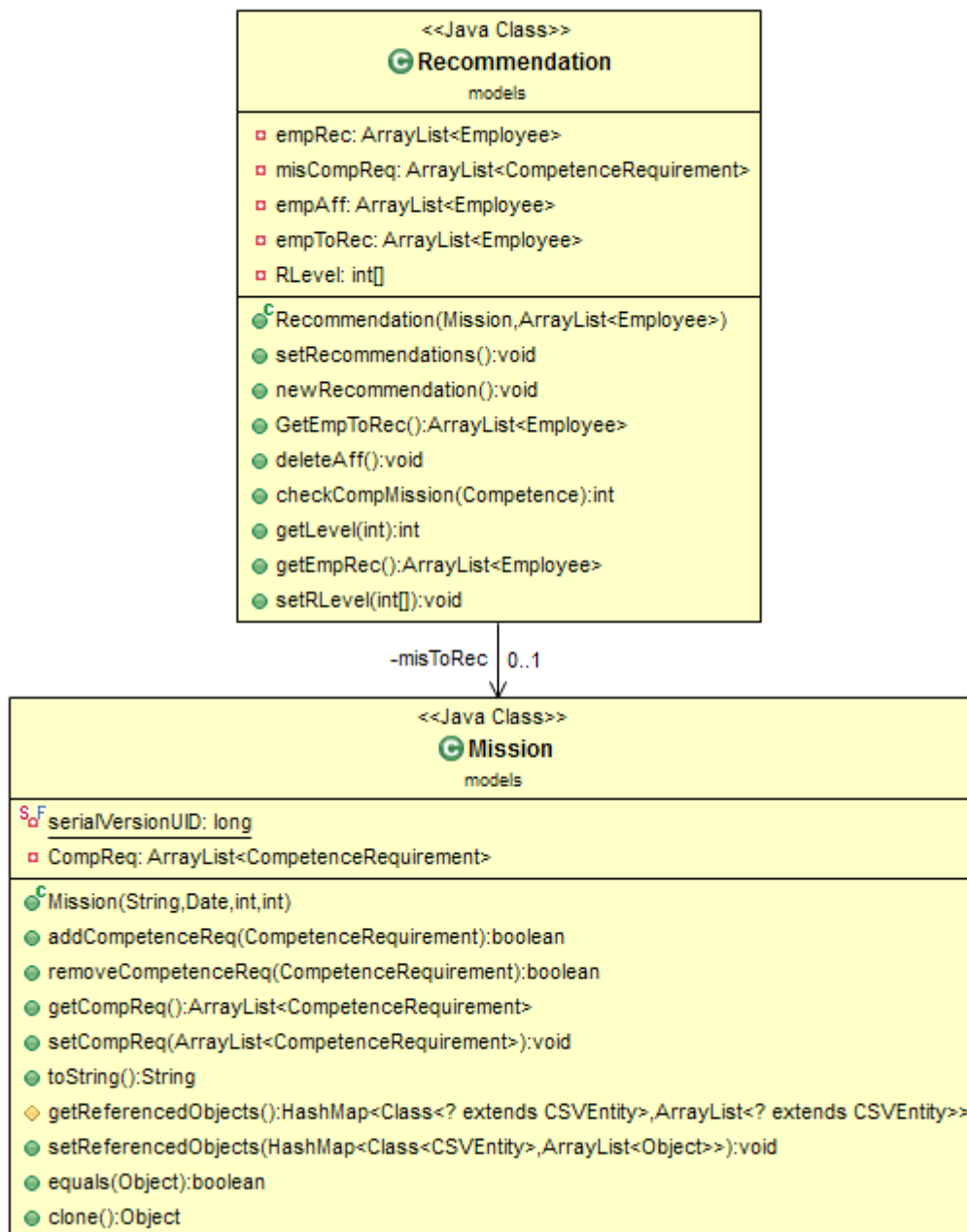
Résultat :

 Affectation d'employés

— ☐ X

Nom	Prénom	Recommandation ▼	
Sanburn	Lamar	50 %	▲
Syres	x555555x	50 %	
Pommier	Wilbur	50 %	
Melen	Augustus	33 %	
Martie	Ryan	33 %	
Cummings	Hosea	33 %	
Schachte	Stuart	25 %	≡
Riesenberger	Chung	25 %	
Gerrish	Luigi	20 %	
Tebeest	Norbert	20 %	
Elsbree	Jamey	20 %	
Viehweg	Ty	20 %	
Ed	Hassan	17 %	
Objio	Kelvin	17 %	
Novinger	August	14 %	
Roosa	Carmelo	14 %	
Quaife	Myron	14 %	
Sanner	Ahmed	14 %	
Campolo	Edmund	0 %	
Muckler	Cedrick	0 %	
Defrancisco	Greg	0 %	
Arnaiz	Lindsey	0 %	
Shumer	Lyman	0 %	
Abrahamson	Santo	0 %	
Besmer	Bernardo	0 %	▼

Ajouter



Package « Models » : Système de recommandation des employés pour une mission