Programmation Concurrente Répartie Réticulaire et Réactive

Romain Demangeon

Rapport du :

Micro-Projet Web

Réalisé par : ABBES Billel

Année Universitaire : 2019/ 2020

Contents

1. Contexte

Dans la majorité des gares TGV SNCF, l'équipe d'escale est composée des agents, des OIV, et des chefs d'escales. Chaque jour, plusieurs trains passent par ces gares, pour chaque train, il y'a un nombre de tache à réaliser, l'équipe de l'escale de la gare doit garantir la réalisation de ces taches Cette équipe est censé garantir la réalisation des tâches suivantes :

- La sécurité du train : un agent doit se présenter sur le quai durant le stationnement du train à la gare
- Le service voyageur : quand un train est composé de deux rame un agent supplémentaire doit se présenter sur le quai au niveau de la 2ème rame.
- La coupe et l'accroche : certains trains doivent être coupés ou accroché dans une gare
- PSH : L'assistance des personnes à mobilités réduite, cette assistance peut être une assistance simple, ou Rampe électrique -la personne est dans un fauteuil roulant personnel, et elle doit monter ou descendre du train sans besoin de descendre du fauteuil, ce qui nécessite des manipulations à faire au niveau du train pour pouvoir réaliser cette tâche-
- JVS : un groupe d'enfants qui doivent être accompagnés d'un ou plusieurs agents d'escales pour descendre ou monter du train. Cela nécessite un ou deux agents en fonction de nombre des enfants.
- L'embarquement des trains OUIGO : cela nécessite un chef d'escale et deux agents.

Certaines tâches ne peuvent pas être réalisé par n'importe quel agent, il faut que l'agent soit *éligible* de faire la tâche pour qu'il soit affecté à cette tâche.

Chaque début de journée, un **planning** doit être écrit pour distribuer les tâches aux agents. La personne qui fait ce planning doit avoir sous les yeux :

- *La liste des trains* qui passent par la gare ce jour-là, la **note de gare**.
- *La liste des PSH*, elle est en format Excel
- *La liste des JVS*, elle est également en format Excel
- *Les créneaux du travail des agents*.
- La connaissance des *éligibilités des agents* (cela n'est pas fourni).

Ensuite, en respectant la contrainte qu'un agent ne peut pas réaliser deux tâches simultanées, et que les tâches doivent être distribué le plus légalement possible, le planning se fait à la main et écrit sur la note de gare, la note de gare

est ensuite se prend en photocopie en plusieurs exemplaire, et chaque copie est donnée à un agent.

En cas de **perturbation**, un nouveau planning doit se créer, et se redonner aux agents.

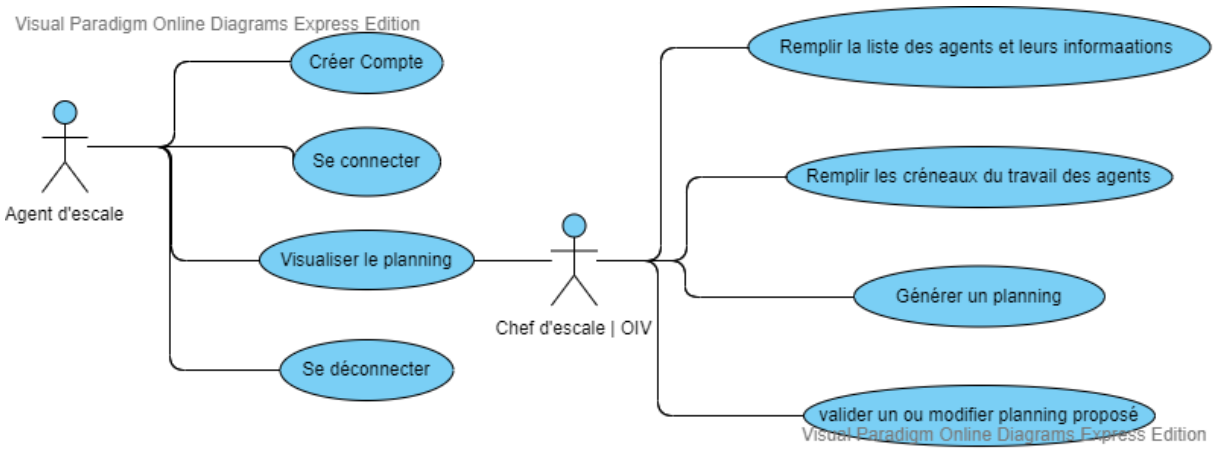
2. Objectif

Faire une application qui peut :

- Réunir dans une même base de données, les informations concernant les agents (notamment l'éligibilité) et les créneaux de travail de chaque agent.
- Récupérer la liste des trains qui passent et qui s'arrêtent par une gare et les informations concernant ces trains : numéro, nombre de rame, heure d'arrivé, heure de départ, si c'est un OUIGO, doit être coupé ou accroché.
- Lire un fichier Excel et en tirer la liste des PSH.
- Lire un fichier Excel et en tirer la liste des JVS.
- Proposer un planning.
- Partager ce planning à tous les agents en format numérique.

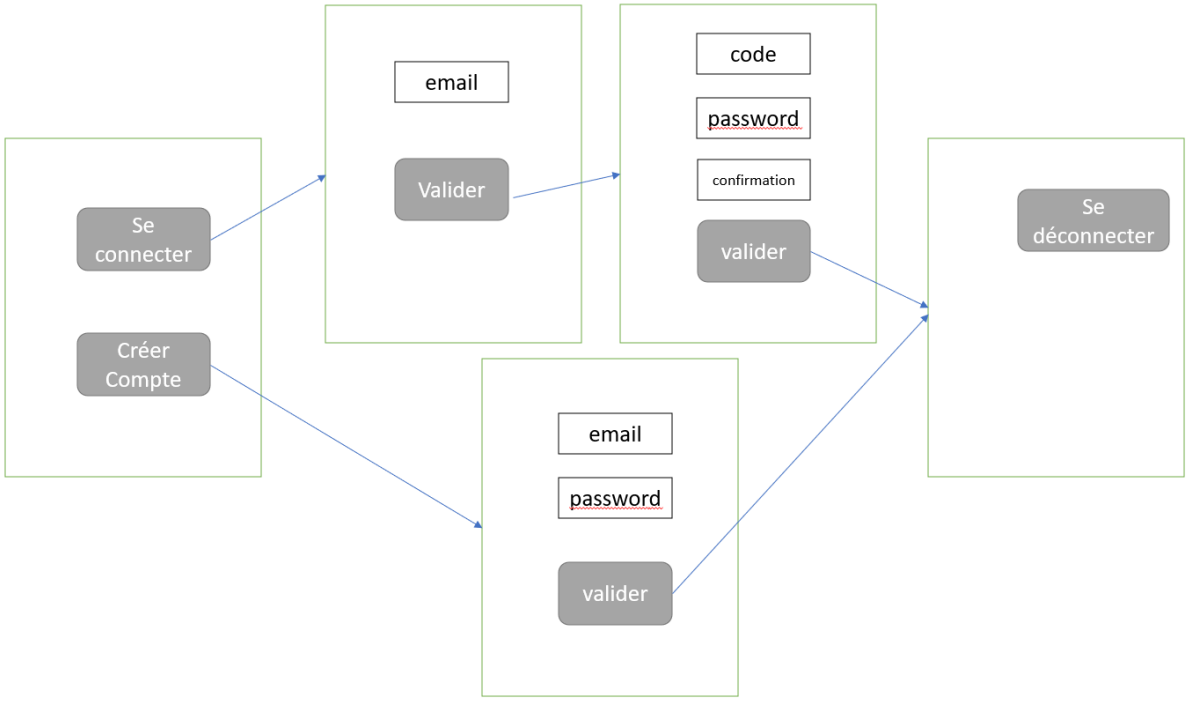
3. Architecture de l'application

a. Diagramme de cas d'utilisation :

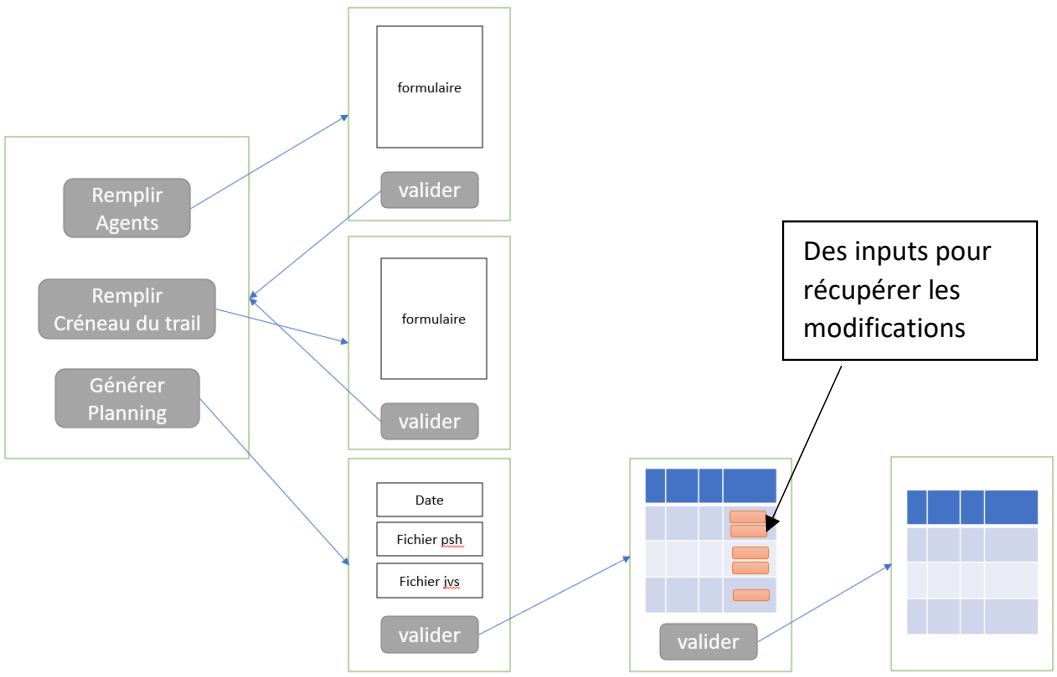


b. Architecture des boutons et pages :

i. Pour les agents



ii. Pour les responsables



c. Scénario

i. Scénario de création de compte :

Le système cherche dans la base de données s'il y a un agent qui possède cet adresse mail et qu'il n'y a pas un compte crée avec cet email, si oui, le système dirige l'agent à un deuxième formulaire ou il doit renseigner le code que l'application vient d'envoyer à son adresse mail, et le mot de passe et sa confirmation. Le mot de passe est vérifié au niveau du client, cependant le code est ensuite vérifié au niveau du serveur. Si le code donné par l'agent est identique avec celui qui a été envoyé le compte sera créer avec succès.

ii. Scénario de génération et validation de planning

En cliquant sur le bouton « générer planning » l'utilisateur va avoir un formulaire ou il doit renseigner la date, et deux inputs de type fichiers, l'un correspond aux PSH et le deuxième correspond au JVS, ces deux champs acceptent une valeur NULL (dans le cas où il n'y a pas de prestations PSH, JVS ou les deux). Le programme ensuite récupère la liste des trains qui passent à cette date de l'api SNCF, et les créneaux de travail des agents qui travaille au jour donnée, puis il calcule le planning et le montre a l'utilisateur, l'utilisateur vérifie le planning si c'est bon, si oui il le valide, sinon il modifie les noms des agents des prestations où il veut changer et il le valide. Après avoir validé le planning, les agents connectés visualise le planning sur leur page d'accueil.

d. L'api

- Nom : L'api de la SNCF.
- L'url :
- Mon Token :
- La requête :

Url Root

```
https://a8007b94-0bed-4237-8d32-f1c6da0566ff:@api.sncf.com/v1/
```

Token ***lien***

Path

```
stop_areas/"+ID_Gare+"/vehicle_journeys?
```

Station ***Gare en question*** ***End Point : Train en circulation***

Paramètres

```
since="+date+"T000000&until="+date+"T235959
```

Pour avoir que les trains d'une date donnée, on filtre avec les paramètres '*since* ' et '*until*' en passant la date en format 'YYYYMMJJ' + l'heure en format 'HHMMSS'.

Voici le code java utilisé pour se connecter à l'api :

```
String URL = "https://a8007b94-0bed-4237-8d32-f1c6da0566ff:@api.sncf.com/v1/coverage/sncf/"
        + "stop_areas/"+outils.VariableGlobal.ID_Gare+"/vehicle_journeys?"
        + "since="+date+"T000000&until="+date+"T235959";
try {
    FileWriter myWriter = new FileWriter(new File(VariableGlobal.PATHTrains).getAbsoluteFile());

    URL oracle = new URL(URL); // URL to Parse
    URLConnection yc = oracle.openConnection();
    yc.setRequestProperty("Authorization", "a8007b94-0bed-4237-8d32-f1c6da0566ff" );
```

e. L'architecture des données :

i. Base de données : 3 tables

1. Agent : Rempli par les responsables

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
☐	1	id	varchar(32)	utf8_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
☐	2	nom	varchar(32)	utf8_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
☐	3	prenom	varchar(32)	utf8_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
☐	4	post	enum('chef_escale', 'oiv', 'agent')	utf8_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
☐	5	email	varchar(32)	utf8_general_ci		Non	*Aucun(e)*	email@sncf.fr		Modifier Supprimer Plus
☐	6	elig_jvs	tinyint(1)			Non	*Aucun(e)*			Modifier Supprimer Plus
☐	7	elig_psh	tinyint(1)			Non	*Aucun(e)*			Modifier Supprimer Plus
☐	8	elig_psh_rampe	tinyint(1)			Non	*Aucun(e)*			Modifier Supprimer Plus
☐	9	elig_coupe	tinyint(1)			Non	*Aucun(e)*			Modifier Supprimer Plus

L'id est le matricule de l'agent

2. Compte : Rempli lors de la création de compte par l'agent

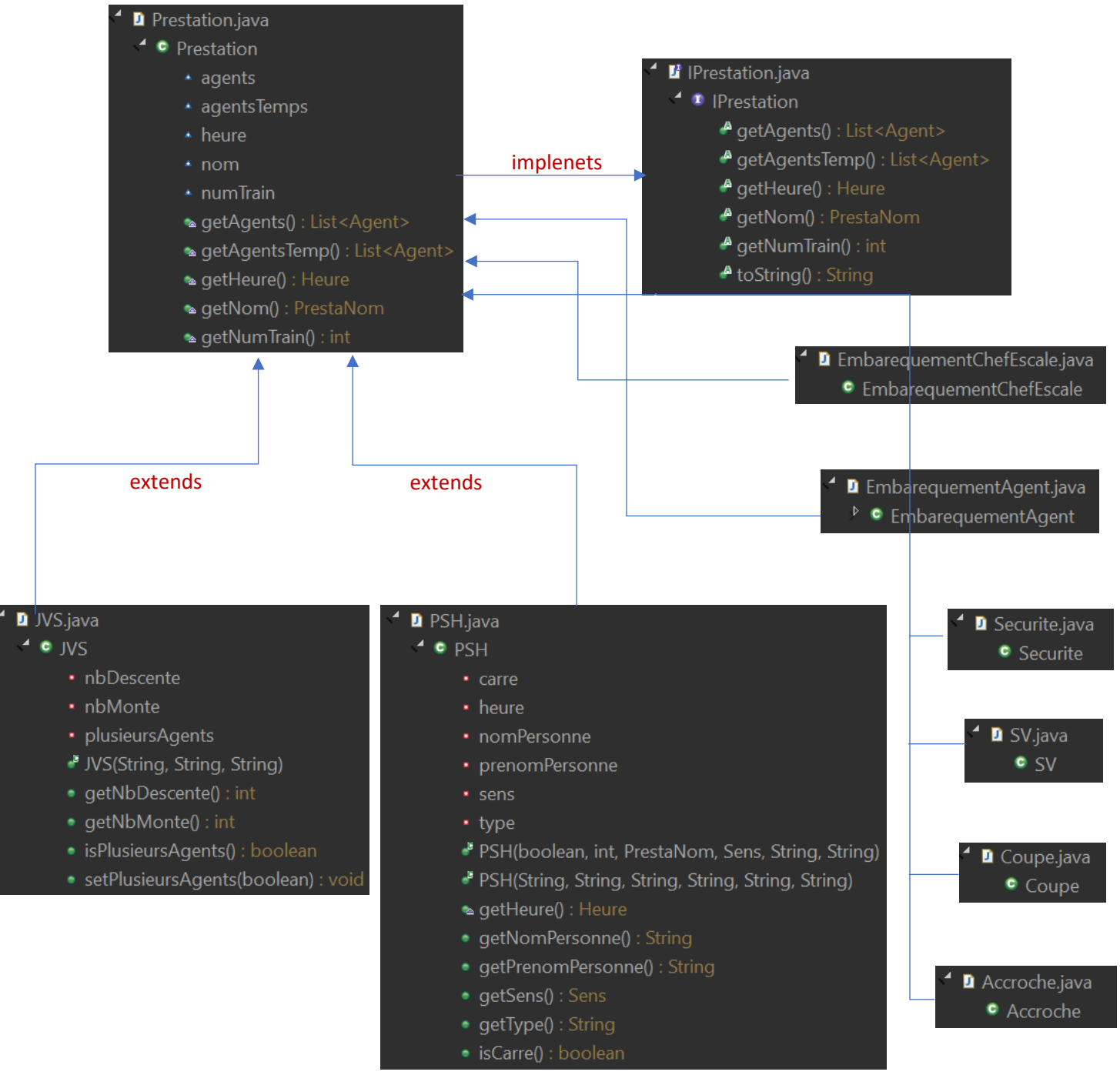
	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
☐	1	email	varchar(32)	utf8_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
☐	2	pass	varchar(32)	utf8_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus

3. *Planning : « créneau du travail des agents »*

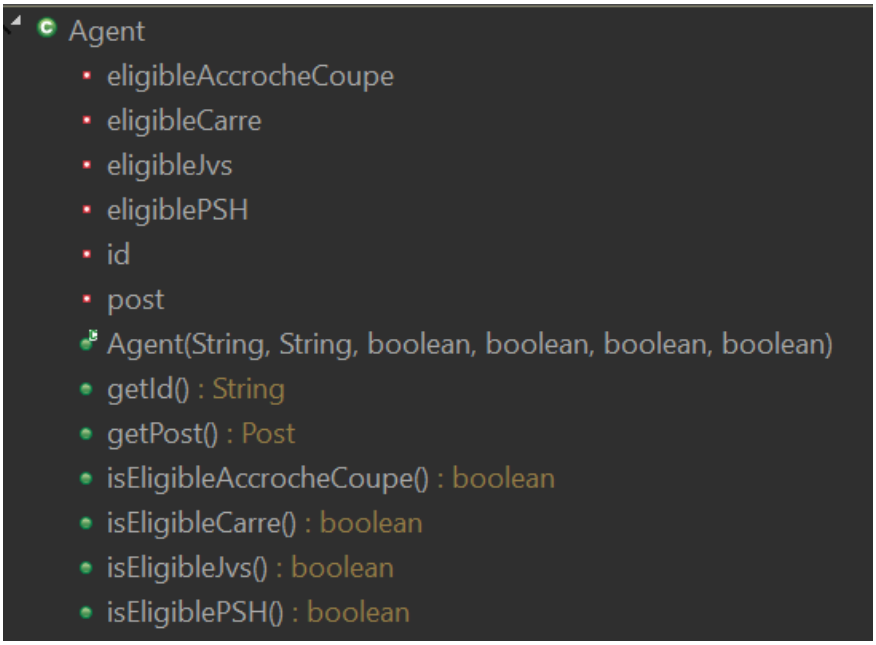
#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
1	**date**	varchar(32)	utf8_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
2	**id**	varchar(32)	utf8_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
3	**debut**	varchar(5)	utf8_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
4	**fin**	varchar(5)	utf8_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
5	**pause**	tinyint(1)			Non	*Aucun(e)*			Modifier Supprimer Plus
6	**debut_pause**	varchar(5)	utf8_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus

i. Les Objet Java :

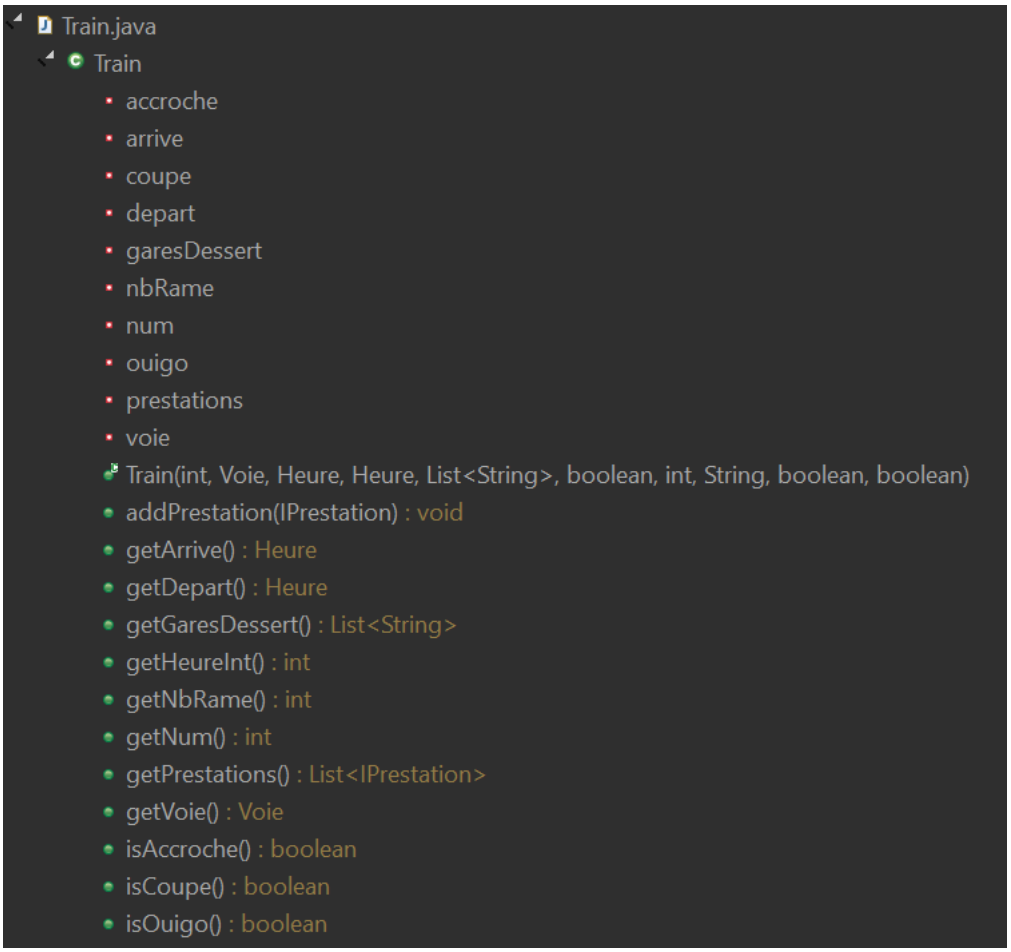
1. *Prestation : « tache »*



2. *Agent*



3. *Train*



ii. Les fichiers Excel :

1. PSH :

un exemple de fichier des prestations d'assistance à la gare de Massy-TGV pour la date de 05/01/2019

Identifiant	Gare	Sense	Prov/Dest	Date	Heure	Train N°	Code Mission	N° Voit	Place	Garantie	Service		Nom	Prénom	BS	Détail BS	Type ass.
21595415	Massy TGV	AC	Rennes	05/01/19	11:22	5332		018	064	Oui	Acces Plus	Mme	**********	**********	PG	AP	AS
21596877	Massy TGV	A	Rennes	05/01/19	11:22	5332		015	037	Oui	Acces Plus	Mme	**********	**********	LR	AP	AS
21593193	Massy TGV	D	Nîmes	05/01/19	11:25	5330		003	035	Oui	Acces Plus	Mme	**********	**********	LR	CA	BF
21595838	Massy TGV	A	Nantes	05/01/19	11:40	7628		11	103	Oui	Acces Plus	Mme	**********	**********	FM	PL	RE
21595415	Massy TGV	DC	Lille Flandres	05/01/19	11:55	5265		005	023	Oui	Acces Plus	Mme	**********	**********	PG	AP	AS
21592268	Massy TGV	A	Lille Europe	05/01/19	13:34	5226		005	046-045	Oui	Acces Plus	Mme	**********	**********	LR	CA	AS
21596870	Massy TGV	A	Lille Europe	05/01/19	13:34	5226		005	046-045	Oui	Acces Plus	Mr	**********	**********	LR	CA	AS
21592400	Massy TGV	A	Aix-en-Provence TGV	05/01/19	15:34	5372		001	015-016	Oui	Acces Plus	Mlle	**********	**********	FM	PL	RE
21596215	Massy TGV	D	Angers Saint-Laud	05/01/19	16:08	5228		003	015	Oui	Acces Plus	Mme	**********	**********	LR	SE	BF
21595289	Massy TGV	A	Le Mans	05/01/19	16:22	5480		002	051-052	Oui	Acces Plus	Mr	**********	**********	LR	SE	BF
21596835	Massy TGV	A	Le Mans	05/01/19	16:22	5460		015	017	Oui	Acces Plus	Mr	**********	**********	FM	PL	AS
21591743	Massy TGV	A	Poitiers	05/01/19	17:52	5284		016	048	Oui	Acces Plus	Mme	**********	**********	LR	SE	AS
21598783	Massy TGV	A	Moûtiers - Salins - Brides-les-Bains	05/01/19	19:22	5393		017	083-084	Oui	Acces Plus	Mme	**********	**********	LR	AP	AS
21597475	Massy TGV	A	Bordeaux Saint-Jean	05/01/19	19:52	7660		1	103	Oui	Acces Plus	Mr	**********	**********	FM	PL	RE
21597769	Massy TGV	A	Montpellier Saint-Roch	05/01/19	20:34	5380		011	037-031	Oui	Acces Plus	Mlle	**********	**********	FE	TR	RE

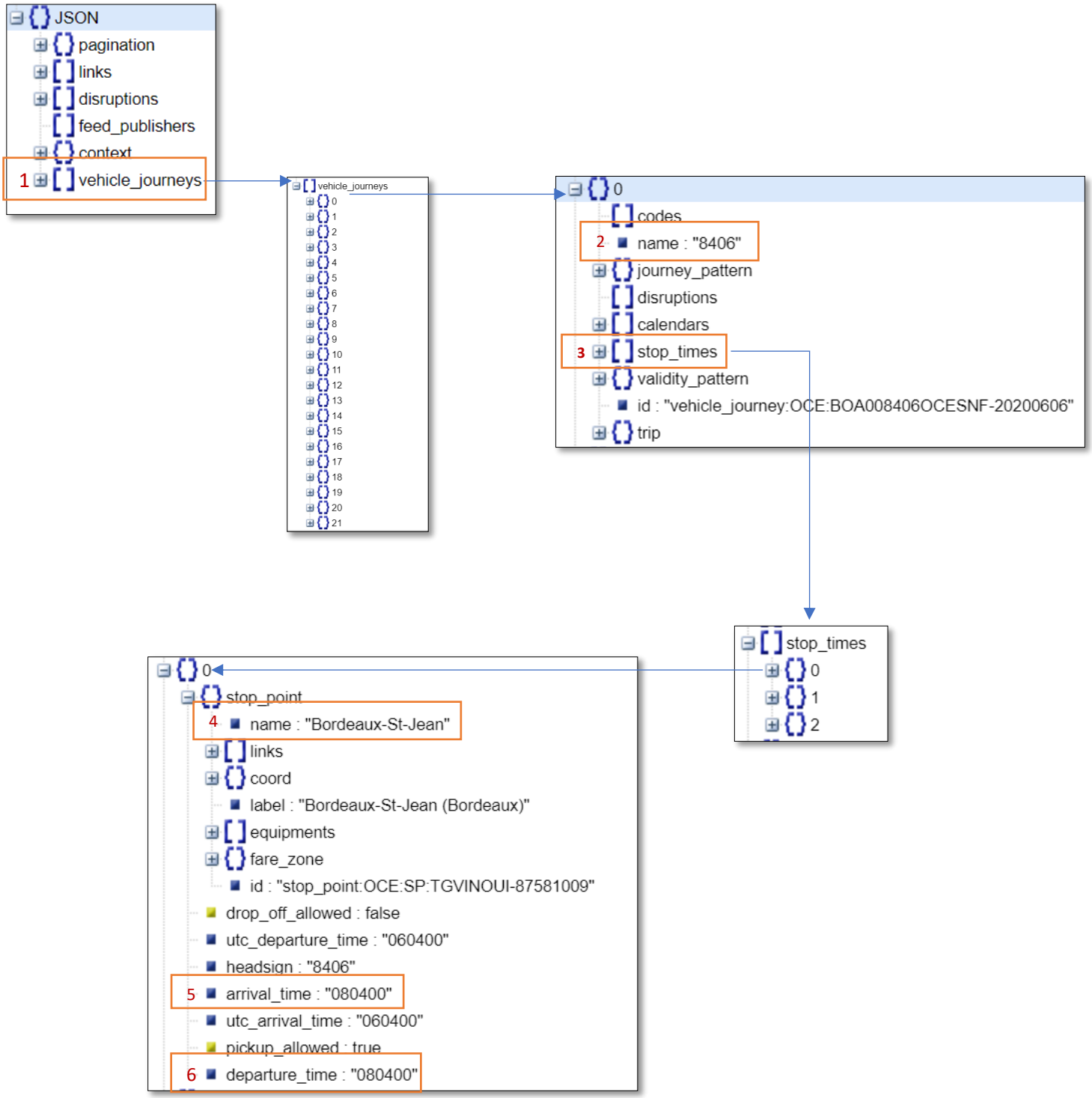
Les champs qui nous intéressent sont : « sens, Heure, Train N°, Nom et Prénom, Type ass »

2. JVS

Identifiant	Gare	Prove	Dest	Date	Heure	Train N°	Nombre montés	Nombre de descentes	N° Voit	Commentaire	Garantie	Service
21595415	Massy TGV	Lilles	Rennes	22/01/2019	11:22	5332	0	0	18		Oui	Junior &Cie
21596877	Massy TGV	Lilles	Rennes	22/01/2019	12:45	5332	4	10	18		Oui	Junior &Cie
21593193	Massy TGV	Paris mont	Nimes	22/01/2019	13:30	5330	3	4	18		Oui	Junior &Cie
21595838	Massy TGV	Lilles	Nantes	22/01/2019	14:43	7628	5	3	18		Oui	Junior &Cie
21595415	Massy TGV	Rennes	Lille Flandres	22/01/2019	15:00	5265	6	0	18		Oui	Junior &Cie
21592268	Massy TGV	Nantes	Lille Europe	22/01/2019	16:25	5226	2	9	18		Oui	Junior &Cie
21596870	Massy TGV	Nantes	Lille Europe	22/01/2019	17:45	5226	0	10	18		Oui	Junior &Cie

Les champs qui nous intéressent sont : « Heure, Train N°, Nombre de montés et nombre de descentes »

iii. Les fichiers JSON :



1-La liste des trains qui passent par la gare donnée, à la date donnée.

2- Le numéro du train.

3-La liste des gares desservies par le train.

4- le nom de la gare.

5-le temps d'arrivée à la gare.

6-le temps de départ de la gare.

iv. Le dossier src :

b. bdd : ce package contient des class qui communique avec la base de données sql, et qui effectuent des requêtes pour obtenir des données ou en insérer.
c. Outils : contient des classes qui permettent de convertir des données :
   i. De Java object à JSON object
   ii. De JSON à Java Object
   iii. De Excel à Java Object,

   une classe qui permet de lire un fichier JSON en line à partir d'un URL, une classe qui envoie des mails et une classe qui s'appelle VariableGlabale ou je mets toutes les configurations de l'application notamment les paths des fichiers où je stock des données, ou les path des dossiers ou je sauvegarde les fichier uploaded…etc

d. Planificateur : La partie qui s'occupe de calculer les plannings.
e. Services : le package qui fait le lien entre les servlets et les autres packages, il contient également des fonctions dont a besoin les servlets.
f. servlets : il contient les servlets.

- web Contenant
  a. css : il contient les feuilles de style.
  b. L'interface client : les fichiers jsp et les script js.
- Les librairies ajoutées :
  a. *commons-fileupload* : pour upload des fichiers dans un formulaire
  b. *javax.mail* : pour envoyer des mails
  c. *com.googlecode.json-simple* : pour manipuler des fichier JSON
  d. *org.apache.poi* et *org.apache.poi* : pour lire des fichier Excel
  e. *mysql-connector-java* : pour connecter la base de données.

- L’algorithme de génération de planning :

Trains : list des trains

Creneaux_agents :map <agent , creneau>

JVSs : list des jvs

PSHs : list des psh

Agents_file : file initialisée par les agents qui travaille pendant le jour donné

Pour tous les trains :

Ajouter les taches suivantes : (sécurité, service voyageur, coupe, accrcohe, embarequement ouigo) en fonction des boolean (is coupe, is accroche, isOuigo et nombre de rame)

Pour tous les psh

Pour tour les trains

Si le numéro de train de la prestation = numéro de train

Et si l’heure de la prestation égale à l’heure du train

Alors ajouter cette prestation psh à ce train.

Pour tous les jvs

Pour tour les trains

Si le numéro de train de la prestation = numéro de train

Et si l’heure de la prestation égale à l’heure du train

Alors ajouter cette prestation jvs à ce train.

Pour tous les trains :

Pour toutes les taches (prestations) de ce train :

Ajouter à la liste ‘agents temp’ de la prestations tous les agents qui travaille pendant le temps cette prestation et qui sont éligibles de la réaliser.

Pour tous les trains

Pour toutes les taches (prestations) de ce train :

Tirer de la liste ‘agents temp’ le premier agent dans la file.

affecter cette prestation à cet agent.

Réinsérer l’agent dans la queue de la file

- ## Des extensions :

En exploitant l'architecture de l'application et les données nous pouvons rajouter facilement des fonctionnalités qui seront utiles :

- Utiliser un Webhook pour se mettre à l'écoute de l'API quand il y'a des perturbations, si c'est le cas on doit examiner le planning en cours s'il est toujours réalisable en fonction des nouvelles données, ou régénérer un nouveau planning, dans ce cas on tient à notifier le responsable pour valider le nouveau planning et après la validation il faudra notifier tous les agents du nouveau planning.
- Les agents peuvent consulter leur planning de travaille sur l'applications.
- Récupérer les informations liées aux prestations d'assistances et rajouter sur le planning d'une sorte que quand l'agent clique sur la prestation aura toutes les informations à propos la personne prise en charge : numéro de voiture, place, destination, provenance…etc.
- Récupérer les informations liées aux trains, notamment les gares qui desservit ce train et l'heure d'arrivée et départ à chaque gare.

  Pour ce faire on peut suivre les étapes suivantes :

  - TrainV2 qui hérite de Train avec un nouveau attribue (List<String> gares)
  - Au niveau du fichier JSON récupérer par l'API :

    Avant : tester si (station == notre gare) alors récupérer l'heure de départ et l'heure d'arrivée

    Après : tester si (station == gare en question) alors récupérer l'heure de départ et l'heure d'arrivée

    Sinon ajouter la station à la liste des gares

    Manipuler l'interface client pour afficher la liste des gares.

- Une Simulation :

Tout d'abord, pour tester si on obtient les bonnes informations des trains de l'api on peut les comparer avec celles données par le site internet « gare et connexion » en indiquant le nom de la gare en question,

Voici le lien : https://www.garesetconnexions.sncf/fr/gare/

Dans la base de données il y'a rempli une simulation d'une équipe. Et leurs créneaux de travail pour les dates suivantes de 11-06-2020

Il y' a également deux fichiers Excel correspondent aux prestations PSH et JVS pour ces deux dates.