# SEMAINE 42 : Précision du diagramme de composant

Lien vers la roadmap : https://github.com/AlexandreLon/AL5A-TeamA/wiki/ROADMAP

## Méthodes des interfaces du diagramme complet

### Interfaces entre clients et webservices

**DepartmentWS:**

```
CreateDepartment(string name, int type)
```

**MaintenanceWS:**

```
ScheduleMaintenance(Date date, int EquipmentId, int action); // le département est associé à l'équipement
VisualizeEquipments();
VisualizeEquipment(int EquipmentId);
```

**MishapWS**

```
CreateMishap(int departmentId, string description, int priority);
```

**TaskWS**

```
ConfirmEndOfTask(int taskId);
```

**ScheduleWS**

```
GetDepartmentPlanning(int departmentId);
GetGlobalPlanning();
```

### Interfaces au sein du système

**DepartmentManager**

```
AddDepartment(int type, string name);
```

**MaintenanceManager**

```
ScheduleMaintenance(Date date, Equipment equipment, int action);
GetMaintenance(int maintenanceId): Maintenance;
UpdateMaintenance(int MaintenanceId): Maintenance;
DeleteMaintenance(int MaintenanceId): void;
```

**DepartmentAssignater**

```
GetDepartmentToAssign(Equipment equipment): Department;
```

**MaintenanceScheduler**

```
CreateTimeSlot(Department department, Date date, int duration):  void
```

**EquipmentInformator**

```
GetEquipment(int equipment): Equipment;
GetEquipments(): Equipment[];
```

### MishapManager

```
CreateMishap(int departmentId, string description, int priority);
GetMishap(int MishapId): Mishap;
Update Mishap (int MishapId): Mishap;
Delete Mishap (int MishapId): void;
```

### MishapScheduler

```
CreateTimeSlot(Department department, Priority priority, int duration):  void
```

### MishapSubscriber

```
SubscribeToMishapCreation(string connection);
```

### TaskAction

```
endTask(int taskId) ;
```

## Méthodes des interfaces du diagramme MVP

### Interfaces entre clients et webservices

#### MaintenanceWS:

```
ScheduleMaintenance(int action); // the department is associated to the equipment
```

#### MishapWS

```
CreateMishap(string description);

GetMishap(int mishapId) ;
```

#### TaskWS

```
ConfirmEndOfTask();
```

#### ScheduleWS

```
getNextTask() ;
GetGlobalPlanning();
```

### Interfaces au sein du système

#### MaintenanceManager

```
ScheduleMaintenance(int action);
GetMaintenance(idMaintenance): Maintenance;
GetMaintenances() ;
```

#### MishapManager

```
CreateMishap(int action);
GetMishap(int MishapId): Mishap;
```

#### TaskAction

```
endTask() ; // dequeue on taskList
```

#### TaskCreator

```
CreateTask(Task task) ; // push in the list
```

**ScheduleVisualizer**

```
GetNextTask() ;

GetGlobalPlanning();
```

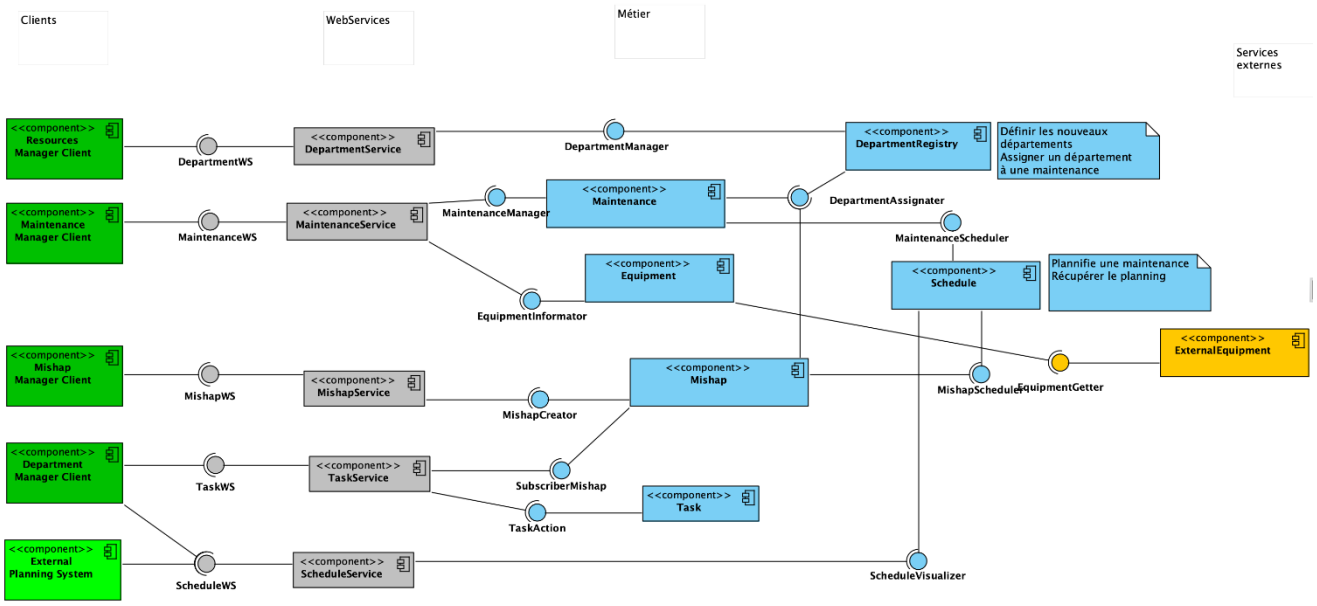
## Description des composants

- **Maintenance** : Gère le cycle de vie des maintenances, communique avec le service de Department pour savoir quel département doit gérer quelles maintenances, et envoie ses informations au service Schedule pour qu'une date soit assignée à la maintenance
- **Equipment** : S'occupe de demander au service externe un ou plusieurs équipements, pour permettre aux utilisateurs de les visualiser.
- **Mishap** : Fonctionne comme le composant Maintenance, mais permet aussi aux services externes de s'abonner pour être tenu au courant d'un incident, et gère aussi la priorité des incidents entre eux.
- **DepartmentRegistry** : Ce composant permet de définir de nouveaux départements, et sait quel département assigner à une maintenance ou un incident.
- **Task** : Ce composant permet de mettre fin aux maintenances et aux incidents. Nous ne sommes pas encore sûrs de garder ce composant car il semble ne pas avoir un réel intérêt (Les composants Maintenance et Mishap pourraient eux même valider ou invalider les tâches).
- **Schedule** : Gère la logique de planification des tâches, comme par exemple décaler des maintenances pour donner la priorité à un incident. Permet aussi aux utilisateurs de visualiser le planning.

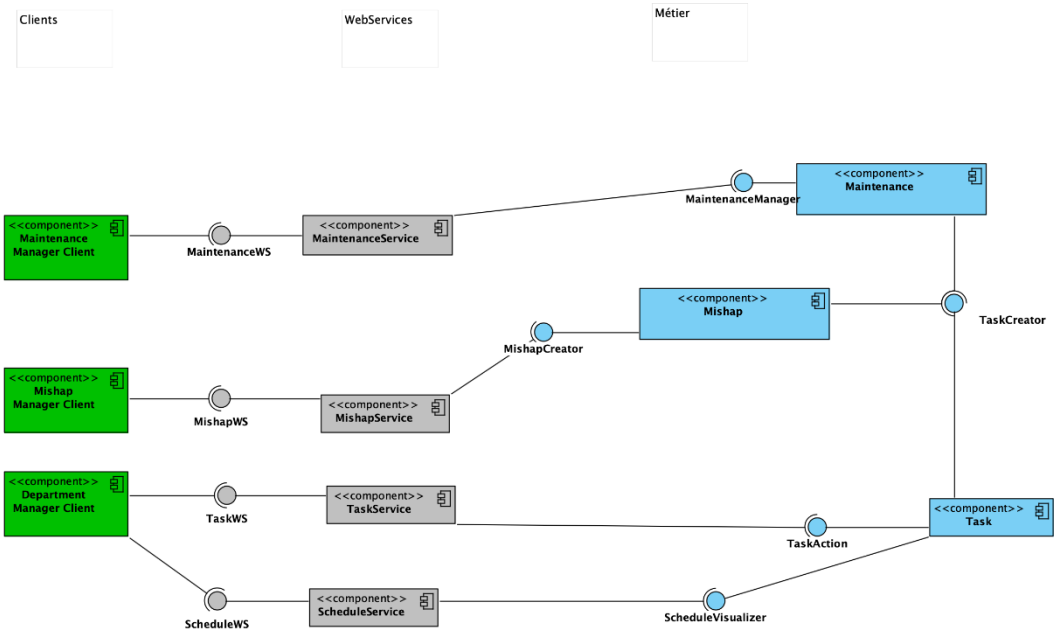
Choix des technos pour chaque composant / couches

# SEMAINE 41 : Diagrammes de composants

## Diagramme complet



## Diagramme MVP




# SEMAINE 40 : Scope du projet

## Personas

Voici les quatre personas qui font partie du système :

- Resource manager : Renaud (gère les ressources de maintenance du réseau ferroviaire)
- Maintenance manager : Manu (gère les maintenances prévues et assigne les départements aux maintenances)
- Mishap manager : Mira (gère les maintenances non prévues dues aux incidents et assigne un département aux interventions)
- Department manager : Derek (gère son département de maintenance)

Ces persona sont également les quatre utilisateurs que nous avons identifiés pour l'instant.

## Scénario complet

Nous avons défini une liste d'actions que les persona pourront faire pour interagir avec le système :

- Renaud définit un certain nombre de services de maintenance dans le système. Exemple : Maintenance des rails, maintenance des locomotives, etc.
- Manu visualise l'état du matériel (dernière révision, km parcourus etc.), suite à quoi il planifie des interventions de maintenance auxquelles il associe un service spécifique.
- Le planning peut être récupéré à tout moment par un département, afin qu'il puisse faire la tâche de maintenance assignée. Par exemple, le département qui gère les départs des trains pourra au préalable vérifier que l'équipements qu'il veut utiliser n'est pas en maintenance.
- Derek peut à tout moment consulter le planning de son département, puis, au moment venu, si la tâche a été complétée, il pourra confirmer que le tâche est terminée.
- Mira peut signaler un incident dans lequel elle précise le service assigné (dans notre cas celui de Derek) et la priorité de l'incident.

- Le service de Derek est notifié de l'incident et peut agir en conséquence, il signalera que l'incident est réparé quand le service aura terminé.
- Les départements abonnés au système seront notifiés de l'incident.

## Scénario couvert par le MVP

- Manu peut prévoir une maintenance
- Mira peut entrer un incident
- Derek peut voir la liste des incidents et la liste des maintenances
- Derek peut confirmer la bonne résolution d'un incident

## Diagramme de cas d'utilisation complet

Voici le diagramme de cas d'utilisation (fig. 1) couvrant le scope complet du projet. Ce diagramme ne représente que le scope tel qu'il est vu au cours de la première semaine.

m

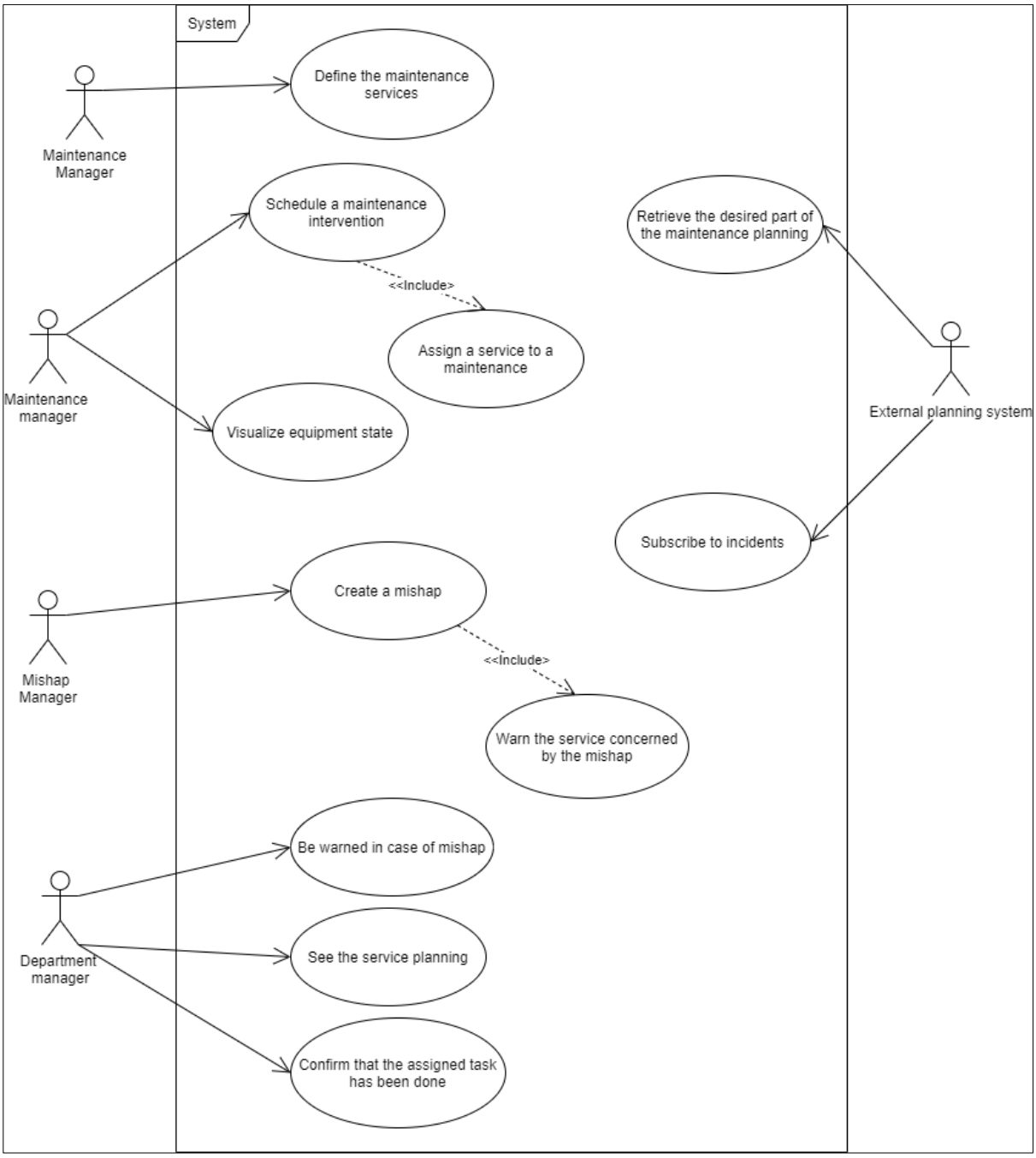



*Figure 1 : Diagramme de cas d'utilisation complet*

## Diagramme de cas d'utilisation du MVP

Le diagramme en fig. 2 représente le diagramme de cas d'utilisations couvrant le scope du MVP tel qu'il a été défini lors de la première semaine.

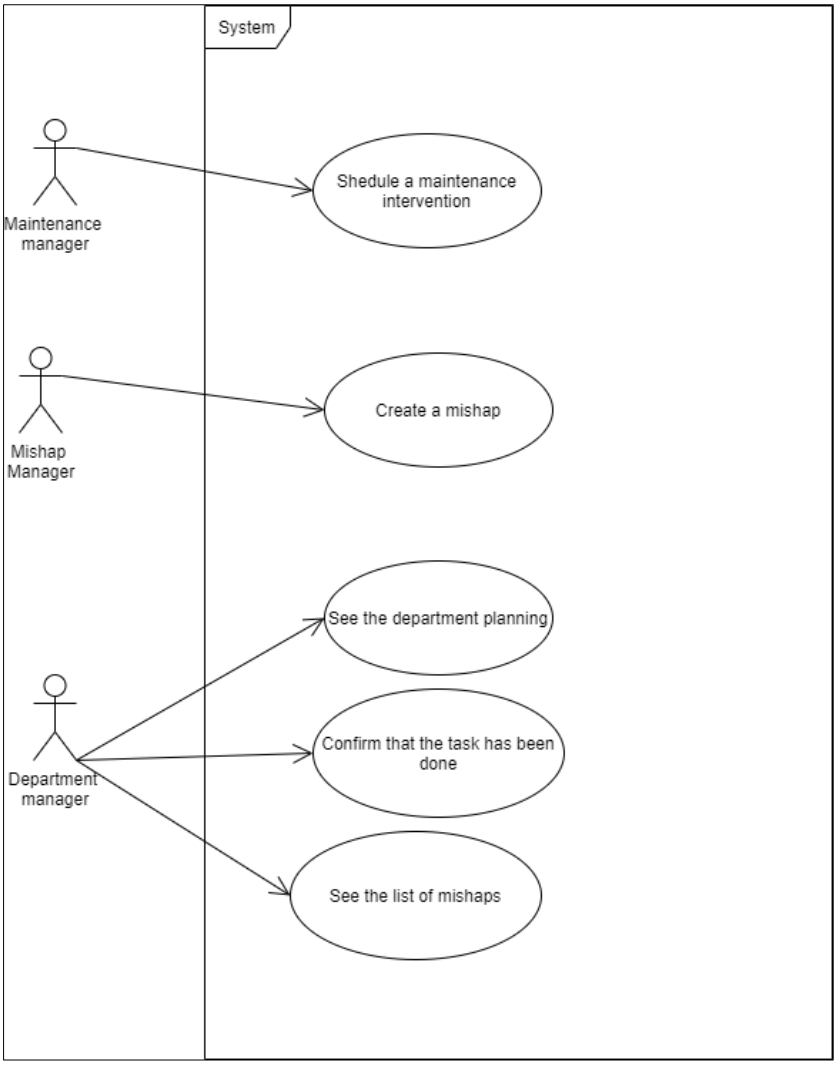



*Figure 2: Diagramme de cas d'utilisation du MVP*