CÉGEP DU VIEUX MONTRÉAL

Document de conception
Audience : application web pour gestion d'horaire

Réalisé par
Dany Viens

Présenté à
Jean-Christophe Demers

Dans le cadre du cours 420-C61-VM
« Projet Synthèse »

Le 4 mars 2021

Maquette

Patrons de conceptions

Stratégie (Strategy)

L'encapsulation est l'un des principes de programmation les plus fondamentaux. Cette philosophie d'isolement les décisions de conception les unes des autres permet d'obtenir un code qui est plus modulable et plus facile d'entretien. Pour résoudre ce problème d'optimisation d'horaire, nous avons choisi d'implémenter le patron de conception « strategy » dans la conception d'un l'algorithme génétique d'intelligence artificielle.

L'algorithme génétique consiste d'une collection de quatres classes de fonction : algorithme générique, reproduction, croisement et mutation. L'objectif serait de rendre, pour chacune des quatre fonctions, une série d'algorithme interchangeable pour résoudre le problème. Par exemple, il serait possible d'utiliser deux méthodes différentes pour la reproduction des chromosomes dans l'algorithme en choisisant une stratégie différente lors de la création de l'algorithme.

L'algorithme génétique aurait un lien vers la classe abstraite *SelectionStrategy* avec la fonction abstraite « selectionner() ». Des stratégies complètes pourraient être implémentée dans des classes dérivées définissant ce que pourrait accomplir la function « selectionner () ». Ces classes pourraient être *RouletteSelectionStrategy* ou bien encore *ValueProportionalSelectionStrategy*.

Les deux avantages principaux de ce patron de conception est qu'il sera possible de tester rapidement plusieurs options pour déterminer les stratégies optimales propre au problème. L'autre avantage est que le code qui sera produit pourra être réutiliser aisément pour résoudre d'autres type de problème.

Object d'accès aux données - DAO - *Data access object*

Il s'agit d'un patron de conception d'architecture logicielle. Il s'agit de regrouper toutes les fonctions donnant accès à la base de donnée dans une classe à part s'assurer de respecter le modèle d'encapsulation des données. Ainsi, il sera beaucoup plus aisé de maintenir un site web, une application où un logiciel si l'ont doit modifier le mode de stockage des données persistantes.

Les autres classes feront toujours références aux fonctions des classes DAO. Le contenus de ces fonctions pourrait changer sans affecter le fonctionnement du programme.

En ce qui concerne notre travail, nous avons choisi d'utiliser Node.JS et le *framework* Express pour programmer notre serveur back-end. Bien que nous utiliserons pas des classes dans cette structure, nous ferons référence à des objets et à des fichiers pour exporter les function en lien avec notre base de données.

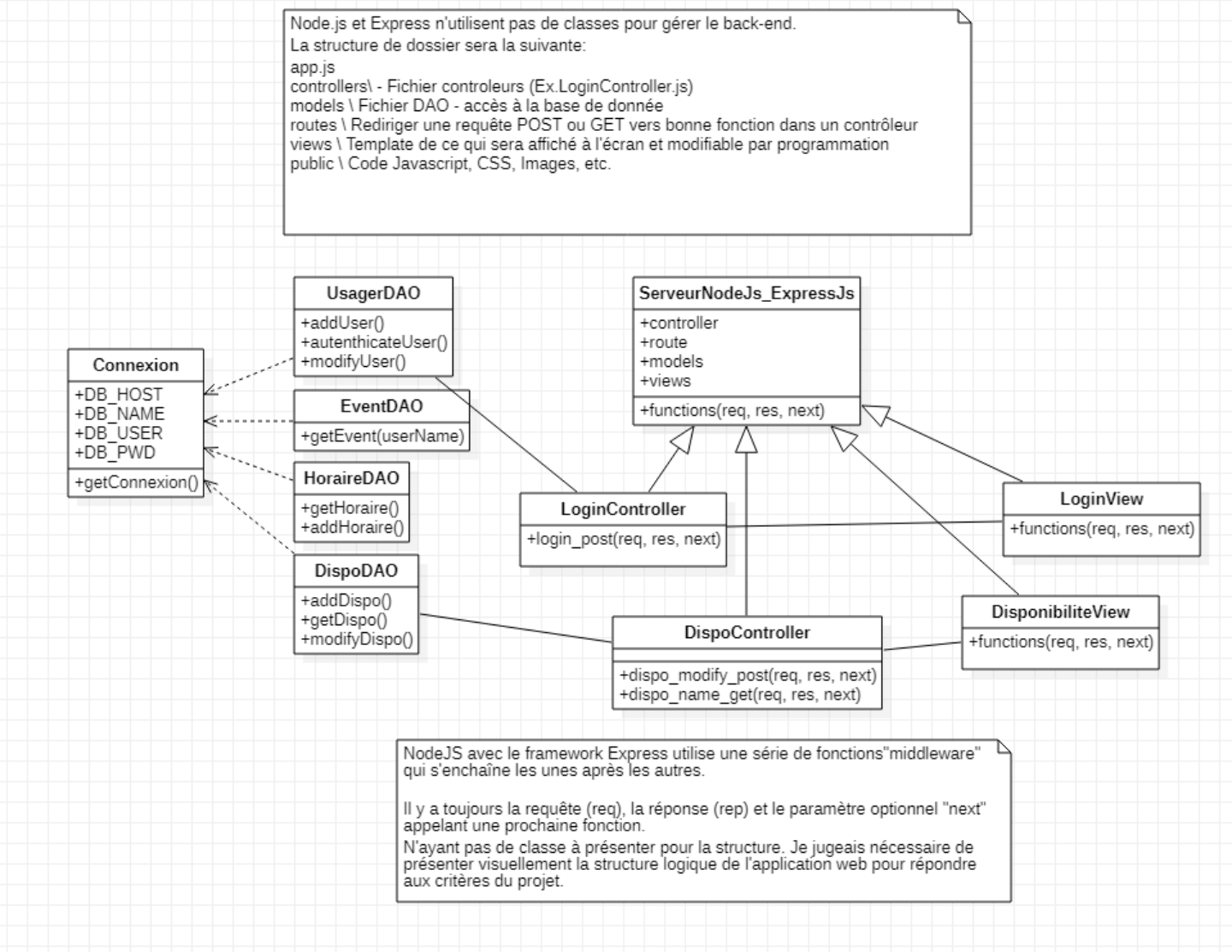
Par exemple, nous encapsulerons les méthodes visant à chercher les usagers dans un fichier User.js. Bien que ces méthodes seront utilisées dans d'autres fichiers, la logique interne de ces fonctions seront rassemblées au même endroit.

UML

L'ensemble des documents est disponible en annexe sous format .PDF et dans le format .MDJ originel. Star UML a été utilisé pour concevoir les diagrammes de cas d'usage, de classes et de séquences.

Diagramme de cas d'usage

Un diagramme différent est présenté pour chaque type de page disponible dans l'application web.



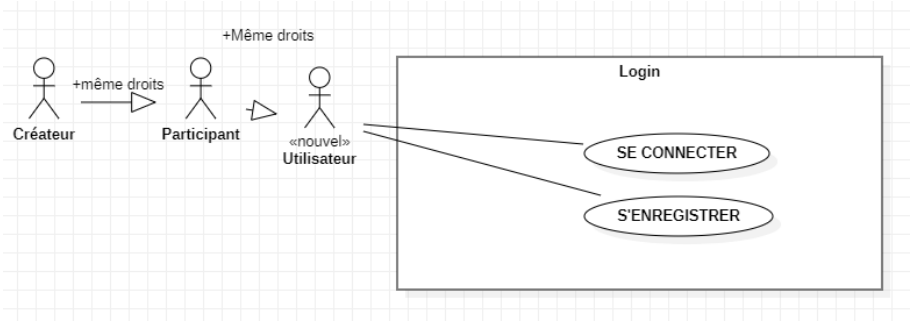


Figure VII : Diagramme des cas d'usage - login

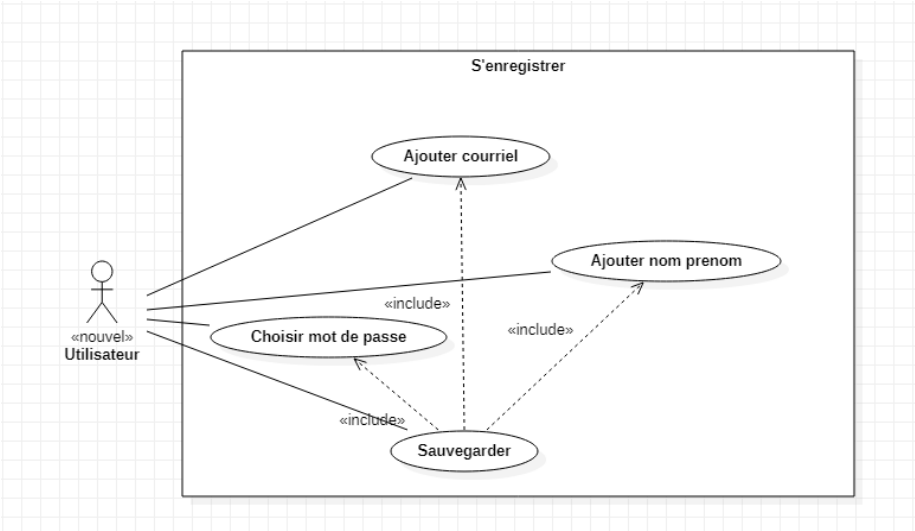


Figure VIII : Diagramme des cas d'usage : enregistrement

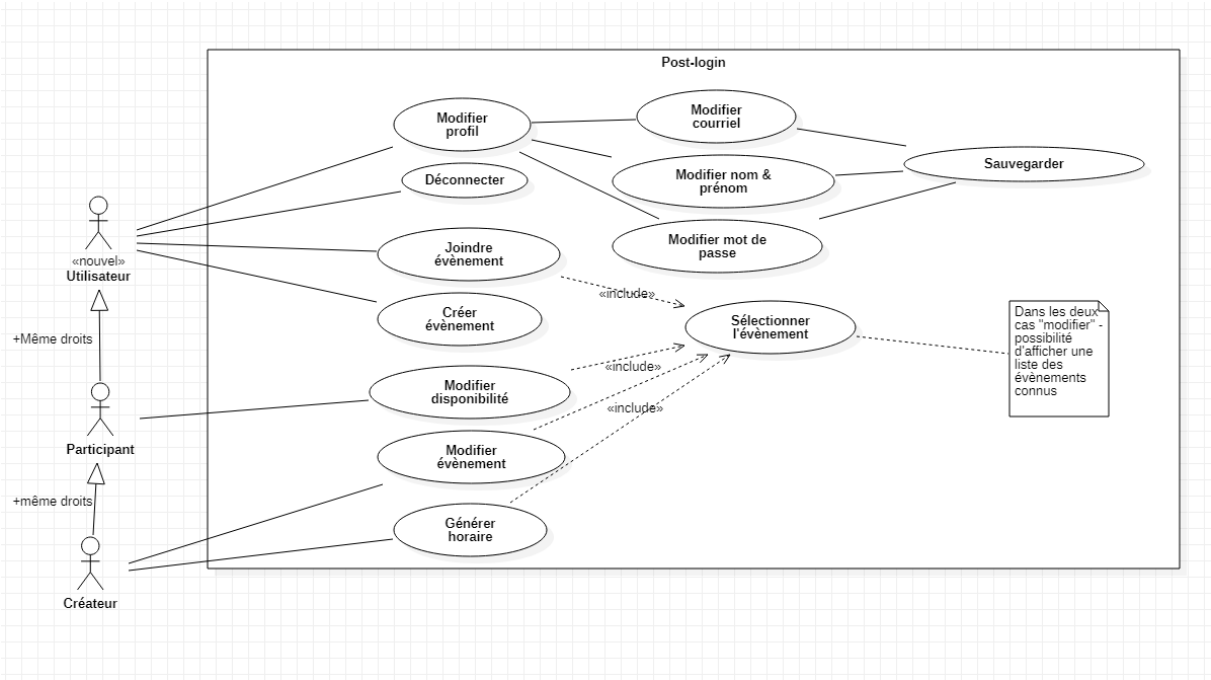


Figure IX : Diagramme des cas d'usage : écran principal

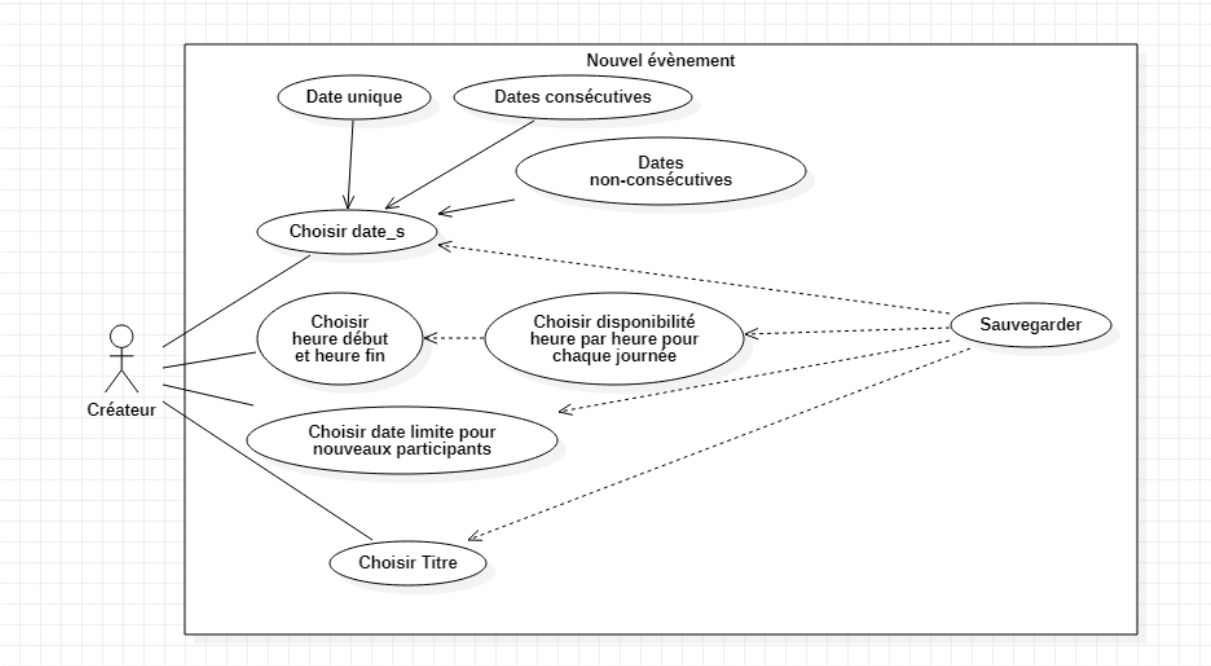


Figure X : Diagramme des cas d'usage : ajout d'évènement

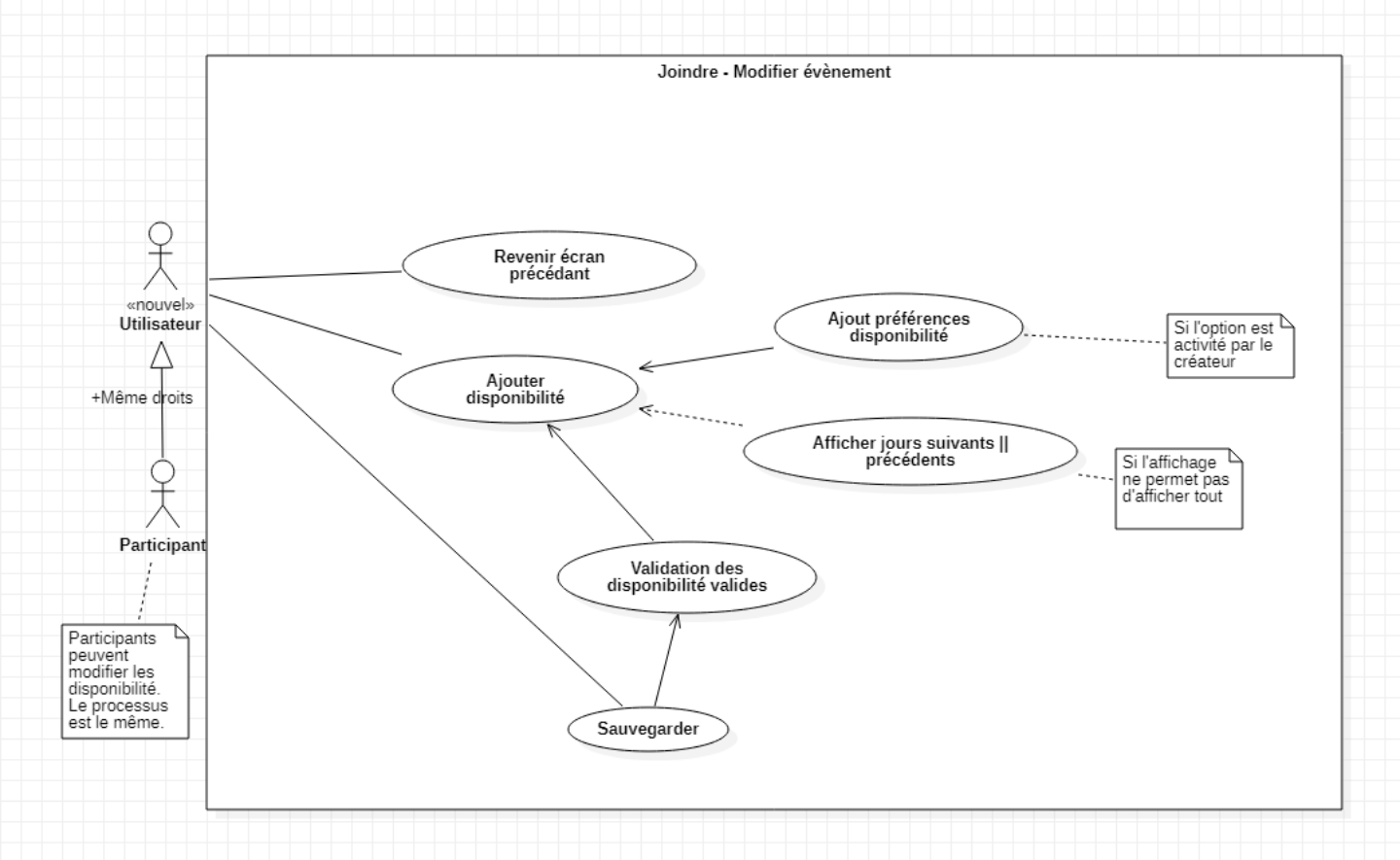


Figure XI : Diagramme des cas d'usage : joindre évènement

Diagramme de classes

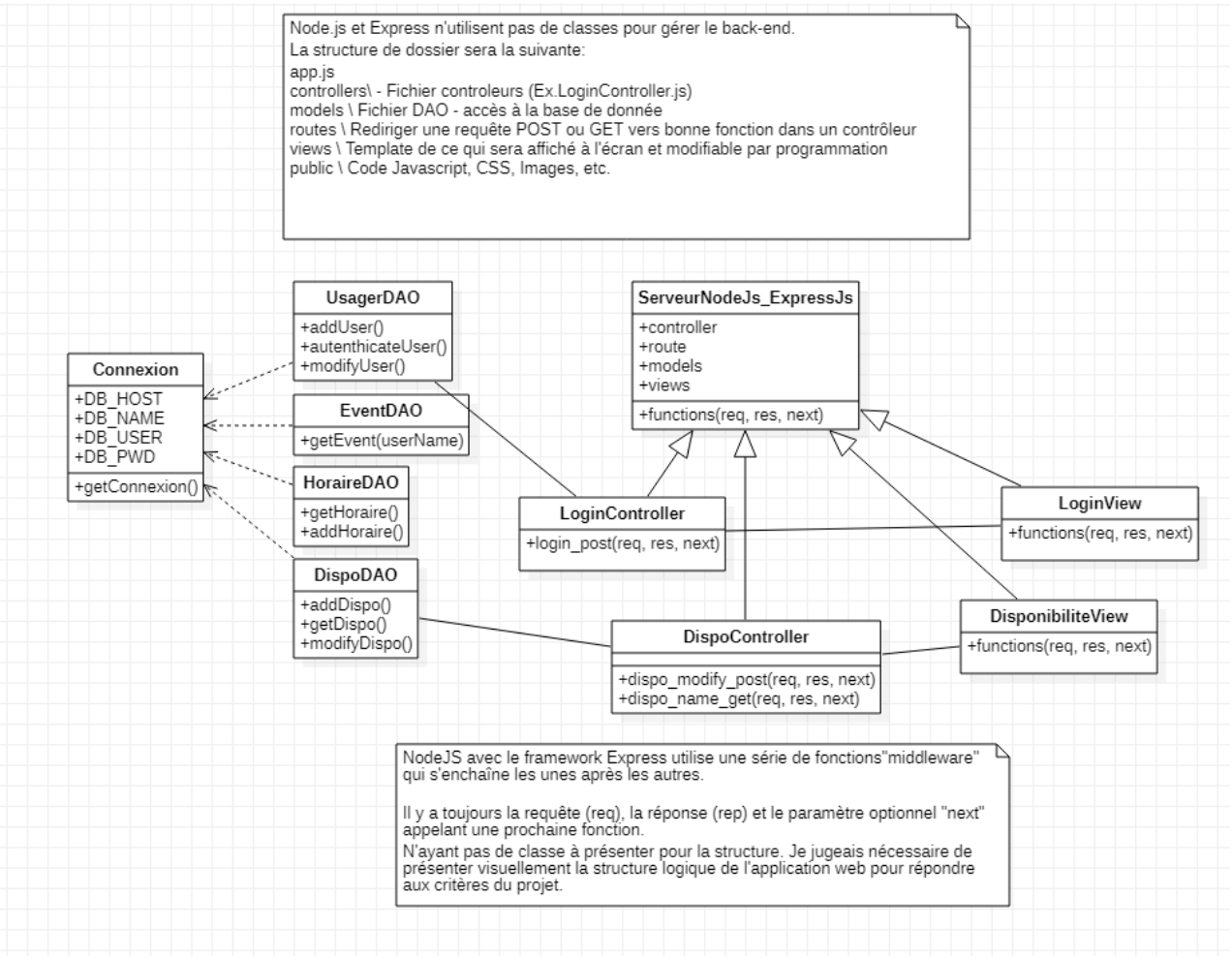


Figure XII : Diagramme de classe : webapp

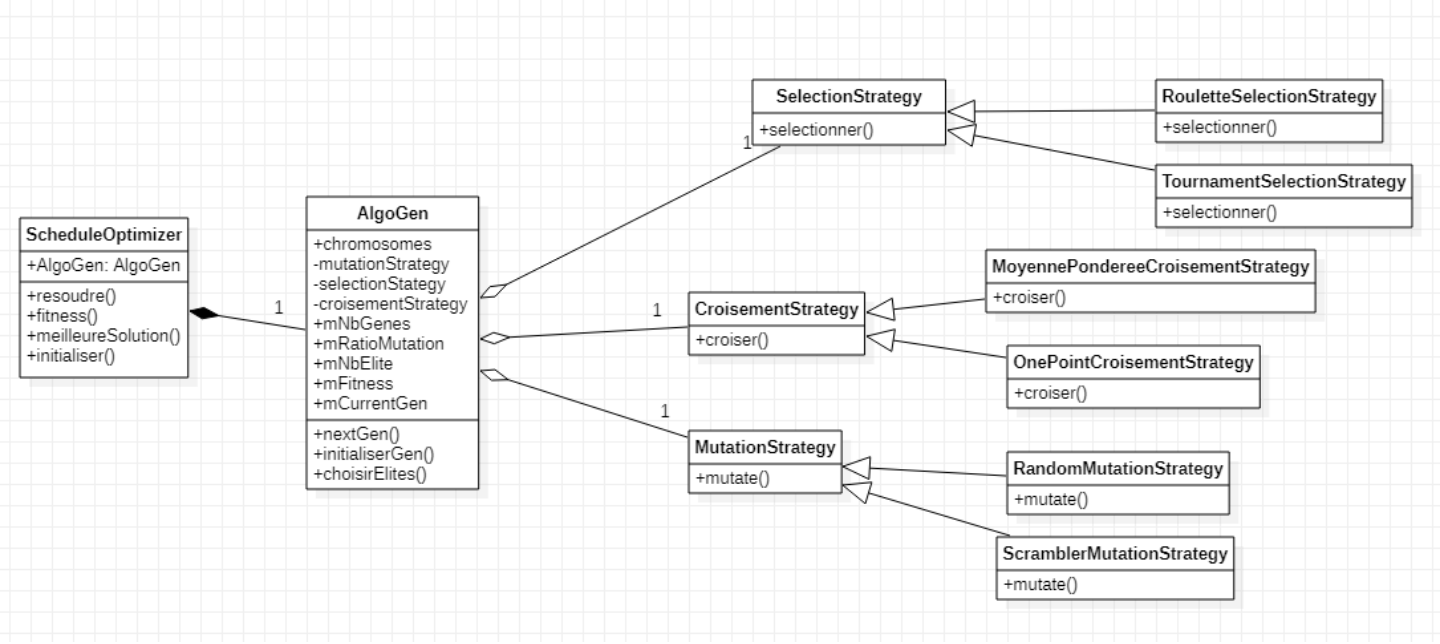


Figure XIII : Diagramme de classe : algorithme génétique

Diagramme de séquences

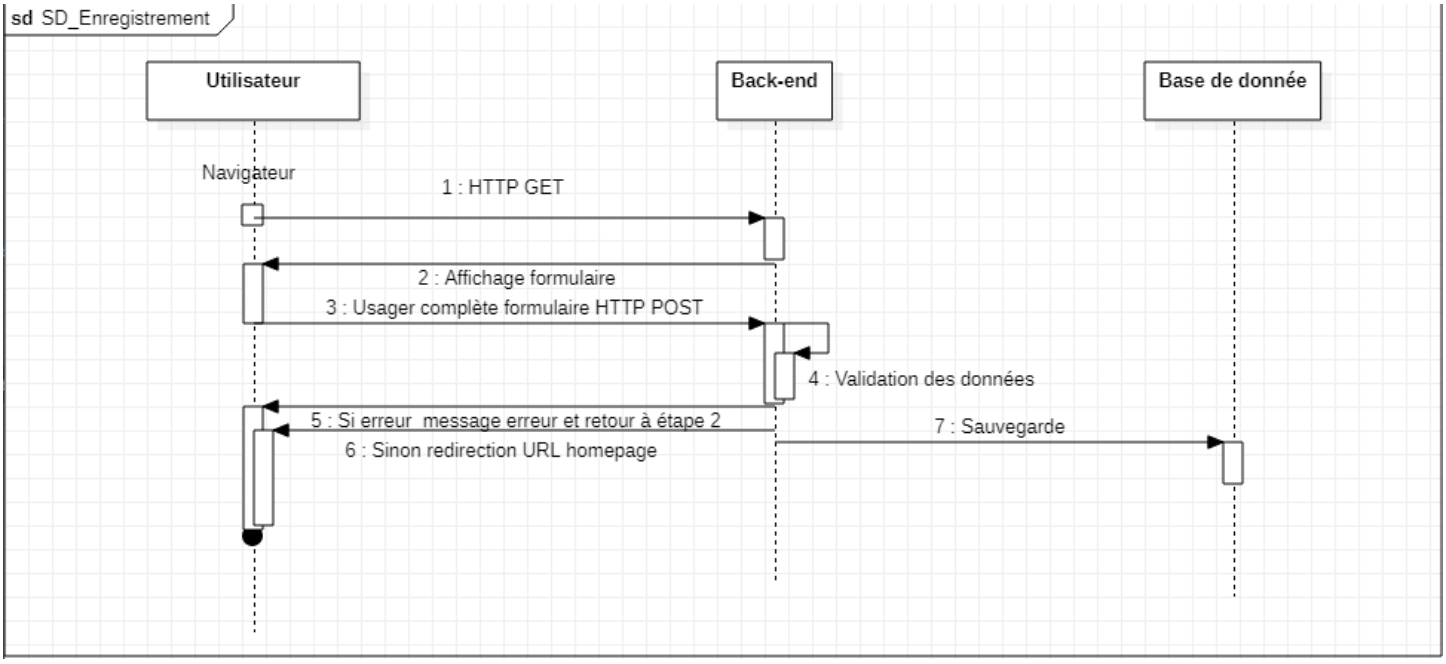


Figure XIV : Diagramme de séquences : enregistrement

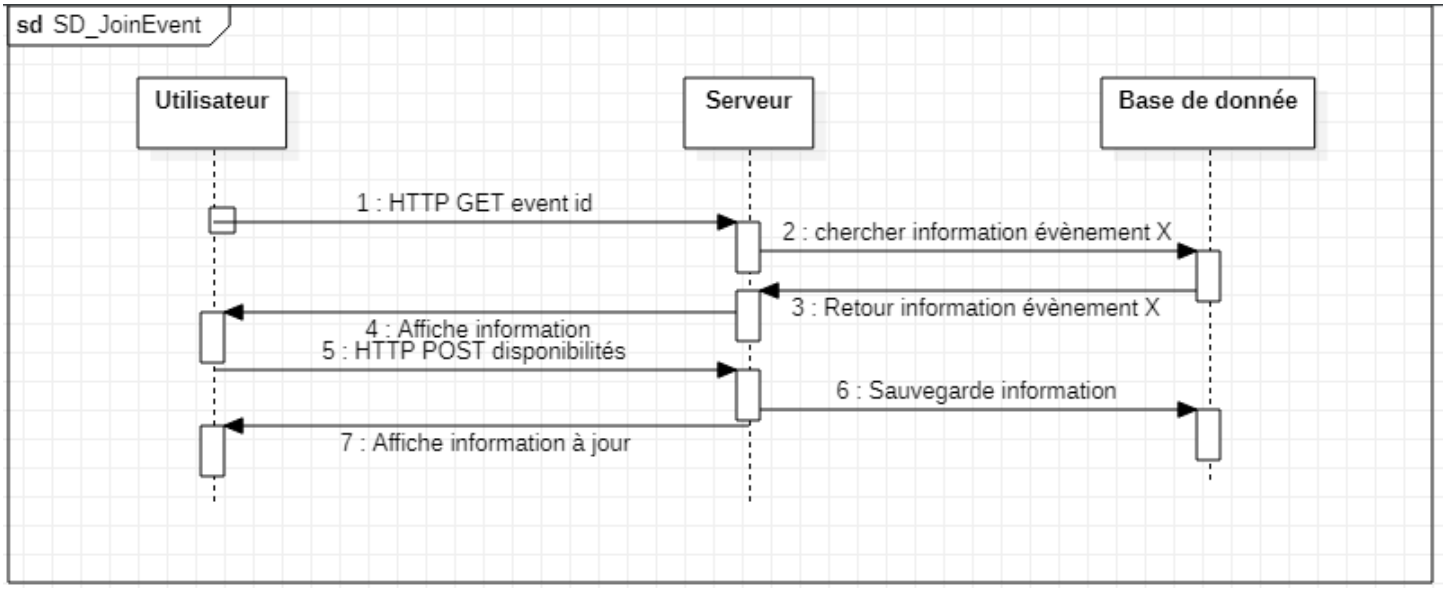


Figure XV : Diagramme de classe : joindre évènement

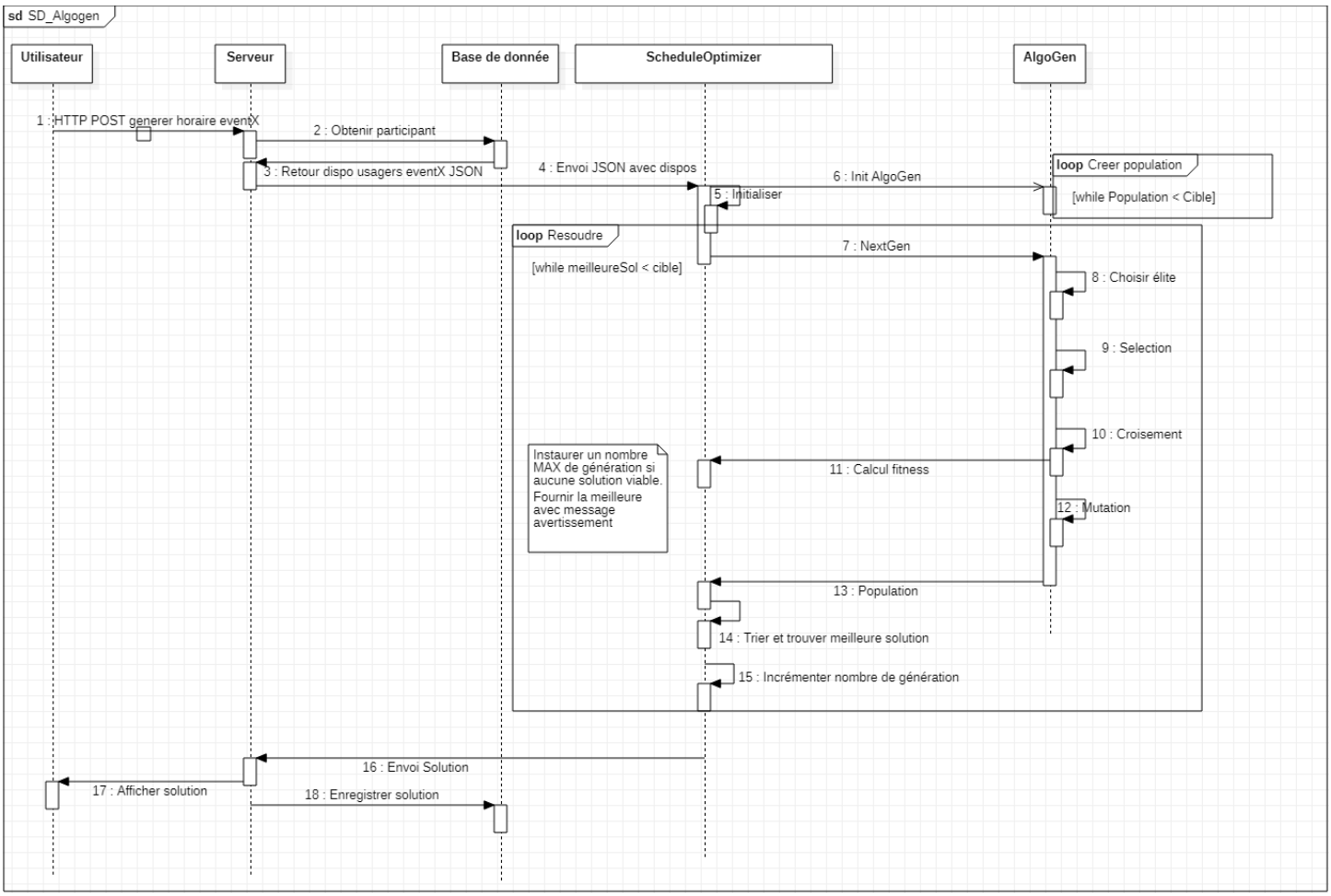


Figure XVI : Diagramme de classe : générer horaire

Schéma de données externes

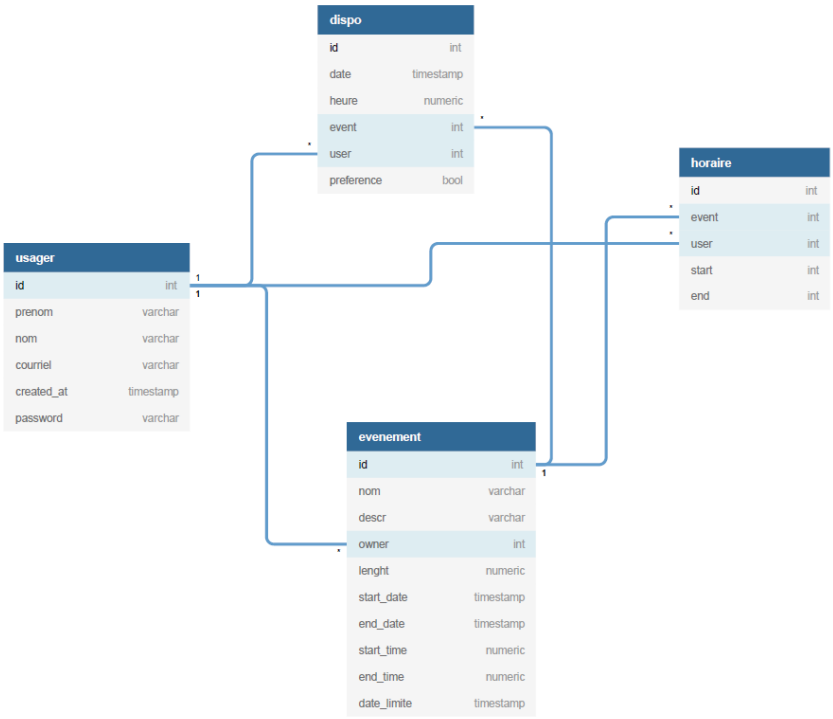


Figure XVII : Schéma de la base de donnée

Une copie du schéma, ainsi qu'un lien vers l'outil de dessin utilisé est disponible en annexe.

Le choix définitif de la technologie n'est pas encore choisi au moment de remettre ce document. Le language choisi sera soit PosgreSQL ou bien MongoDB. Ayant moins d'expérience dans l'utilisation de MongoDB, nous avons une préférence vers cette technologie dans le but d'améliorer notre connaissance d'une technologie.

Dans les deux solutions envisagées, il est possible d'être directement les données sous format JSON. Il s'agit du standard qui sera utilisé pour communiquer entre le *front-end*, le *back-end* et le script d'algorithme génétique.