Andrieu Paul, Masson-Claudez Aloïs, Ozen Burak, Unternehr David S3D

SAE 3.01

-

Développement d’une application

Table des matières

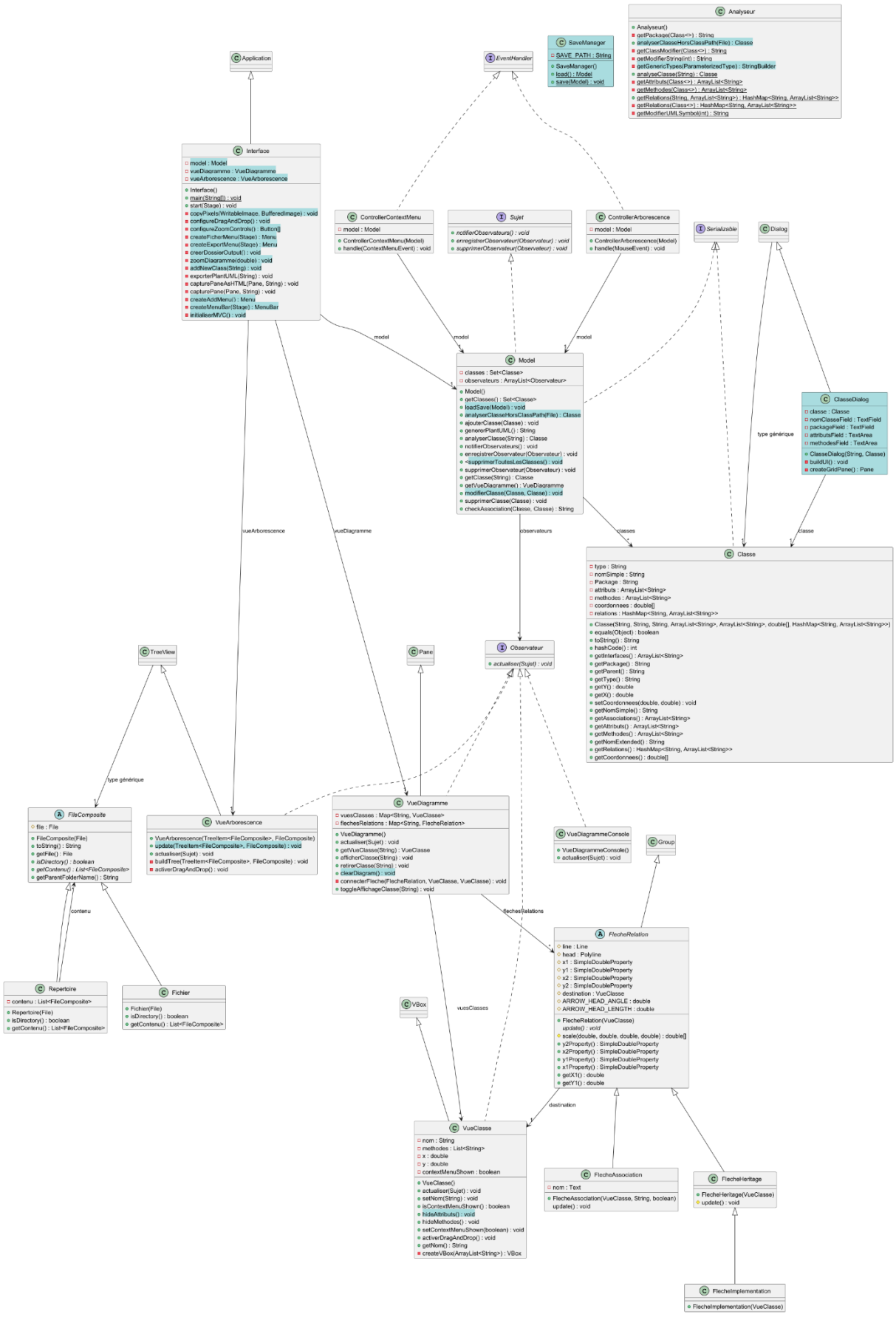
Liste des fonctionnalités réalisées 3
Diagramme de classe finale 4
Répartition du travail 5
Présentation d'un élément original 5
  Paul 5
  Aloïs 5
  Burak 5
  David 5
Modifications par rapport à l'étude préalable 6
Patrons de conception et d'architectures 7
Graphe de scène 8
Mode d'emploi 9

Liste des fonctionnalités réalisées

- Introspection (David)
- Affichage d'un diagramme de classe complet avec les relations (David)
- Supprimer des classes (Burak)
- Masquer des classes (Paul)
- Masquer des méthodes (Aloïs)
- Masquer des attributs (Aloïs)
- Drag and drop depuis l'arborescence (David)
- Déplacer des classes dans le diagramme (Burak)
- Exportation en PNG, JPG, HTML, PUML (Paul, Burak)
- Zoom / Dézoom le diagramme (Paul)
- Création de classe depuis l'interface graphique (Aloïs)
- Modifier les classes existantes (Paul)
- Sauvegarder l'état d'un diagramme, pour le rouvrir à l'ouverture de l'application (David)
- Choix du dossier de travail, d'arborescence, d'export par l'utilisateur (David)

Andrieu Paul, Masson-Claudez Aloïs, Ozen Burak, Unternehr David S3D

Diagramme de classe finale



Andrieu Paul, Masson-Claudez Aloïs, Ozen Burak, Unternehr David S3D

Répartition du travail

Voir fonctionnalités réalisés et Trello : https://trello.com/b/MDXGQbNn/t-iut-sae301-s3d-equipe5

Présentation d’un élément original

Paul

La modification d'une classe existante via le menu clic droit puis avec l'apparition d'une nouvelle fenêtre et la sauvegarde des informations rentrés dans cette fenêtre

Aloïs

La fonctionnalité de création de classe à partir de zéro était très intéressante à réaliser. J'ai dû lire les données entrées par l'utilisateur et les traiter pour qu'elles soient valides (notamment avec des expressions régulières).

Burak

Le drag & drop car je trouve que c'est l'une des fonctionnalités la plus intéressante à faire

David

Affichage des relations sur le diagramme :

Une flèche est représentée par une classe FlecheRelation, cette classe forme une flèche via une Line pour le corps de la flèche, et une PolyLine pour la tête. Les coordonnées sont représentées par des SimpleDoubleProperty pour pouvoir attacher les flèches aux VueClasse en utilisant la Property de l’attributs.

Des listeners sont ajoutés aux coordonnées pour pouvoir changer dynamiquement les coordonnées.

Une méhtode scale(...) s'occupe de calculer la longueur de la flèche. Etant donné que les coordonnées des flèches sont reliées au centre de chaque VueClasse, on cherche à trouver les coordonnées du point d'intersection entre le corps de la fleche, et la bordure de la VueClasse. Pour cela, on calcul ce point grâce à l'angle entre les deux centres des VueClasses, et leur longueur et largeur. Une fois ces coordonnées calculées, étant donné qu'on obtient les coordonnées du point du point de vue de la VueClasse, il faut ajouter les coordonnées du parent.

La tête de flèche est calculé en utilisant aussi l'angle, et est créé au fur et à mesure en partant du bout du corps de la flèche.

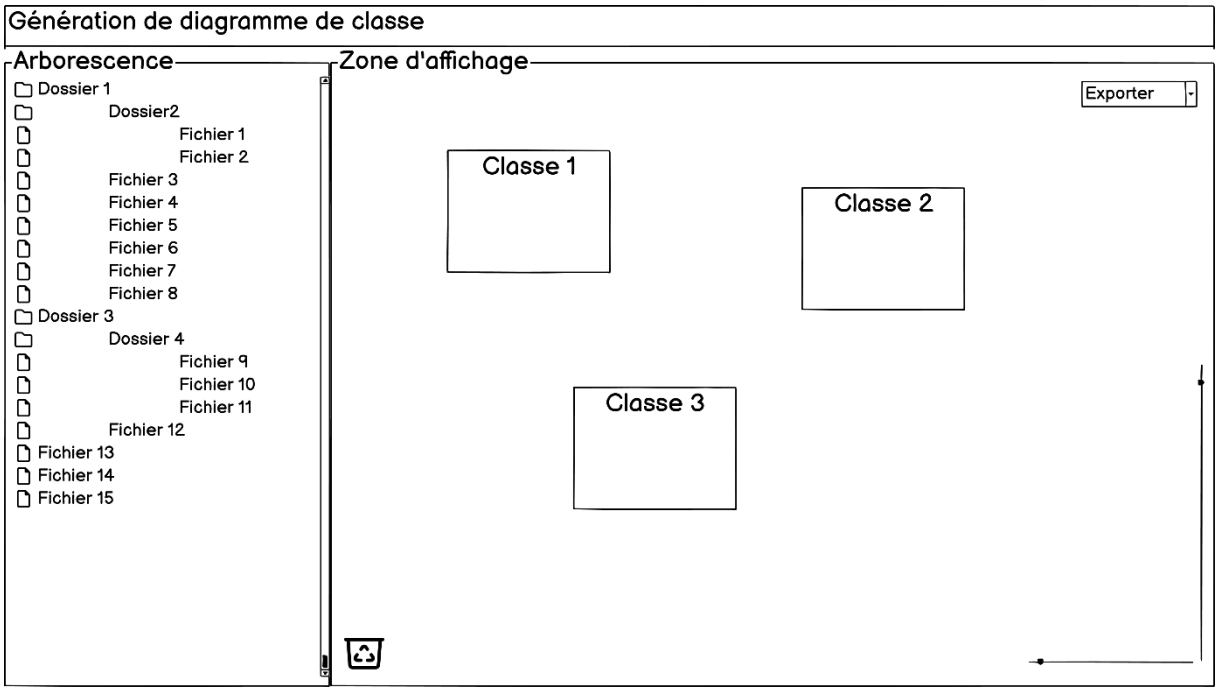
Des classes filles sont utilisés pour styliser la flèche en fonction du type de relation

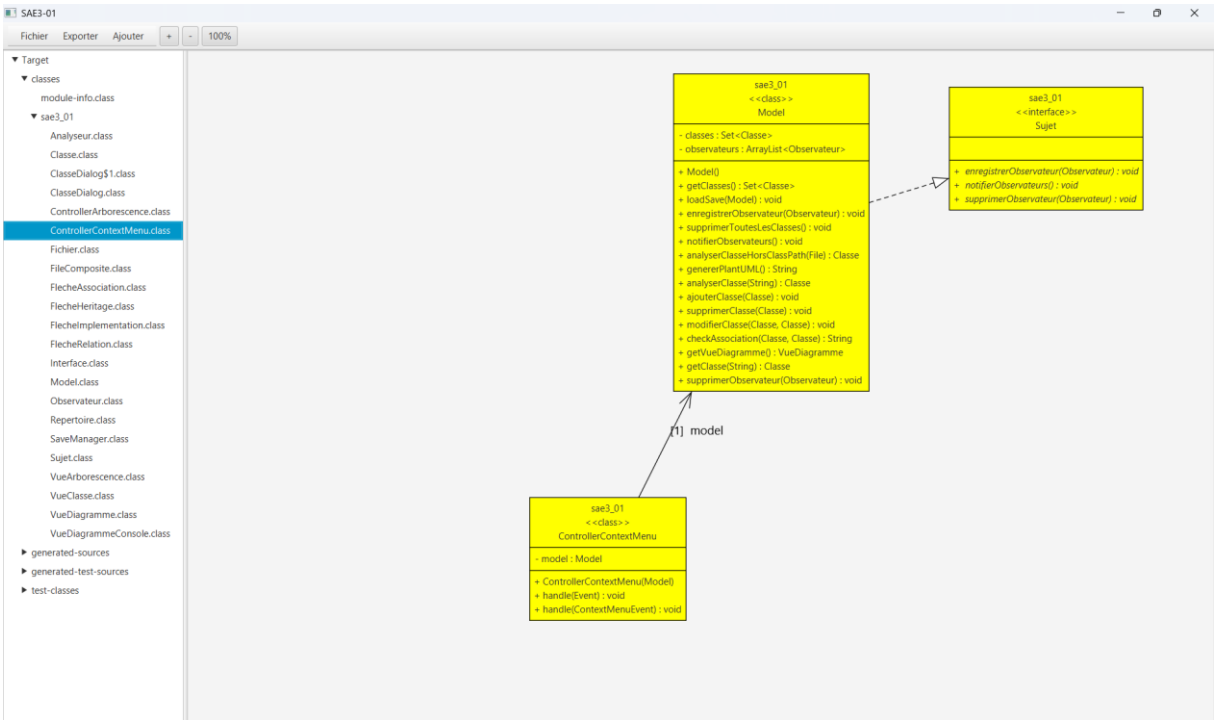
Modifications par rapport à l'étude préalable

Toutes les fonctionnalités de l'étude préalable ont été ajouté, sauf la génération d'un squelette de classe.

Certaines fonctionnalités supplémentaires ont été ajoutés, comme la sauvegarde d'un diagramme ou la création de classe à partir de l'interface graphique.

L'interface est elle aussi un peu différente des maquettes originales, notamment l'emplacement des boutons et leur nombre.





Patrons de conception et d’architectures

Nous avons mis en place le patron d’architecture MVC, et les patrons de conception composite et observateur

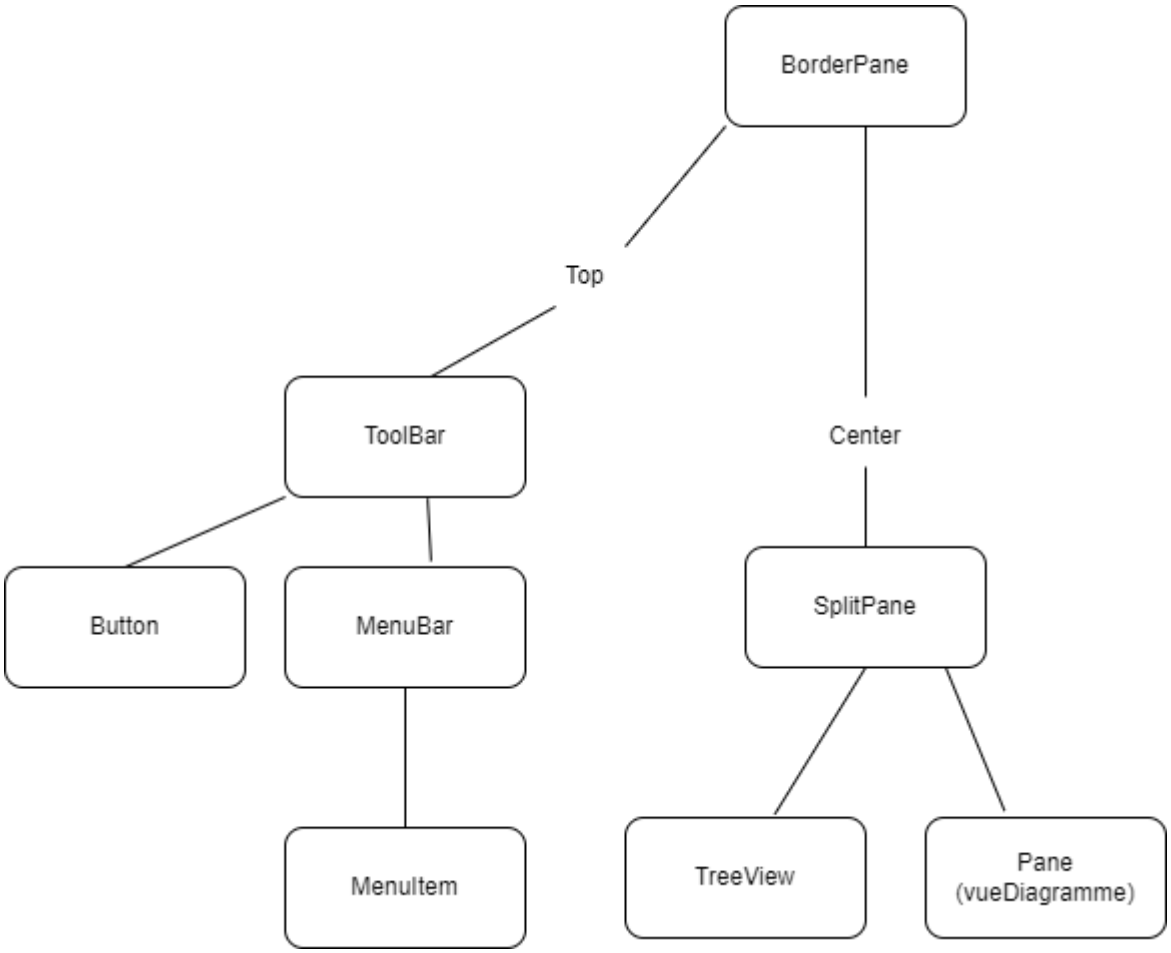
MVC :

- Modèle :
  - Model
- Vues :
  - VueArborescence (TreeView<FileComposite>)
  - VueClasse (VBox)
  - VueDiagramme (Pane)
  - VueDiagrammeConsole
- Contrôleurs :
  - ControllerArborescence
  - ControllerContextMenu

Composite :

- FileComposite (classe mère) :
  - Repertoire (Classe fille, contient des FileComposite)
  - Fichier (Classe fille)

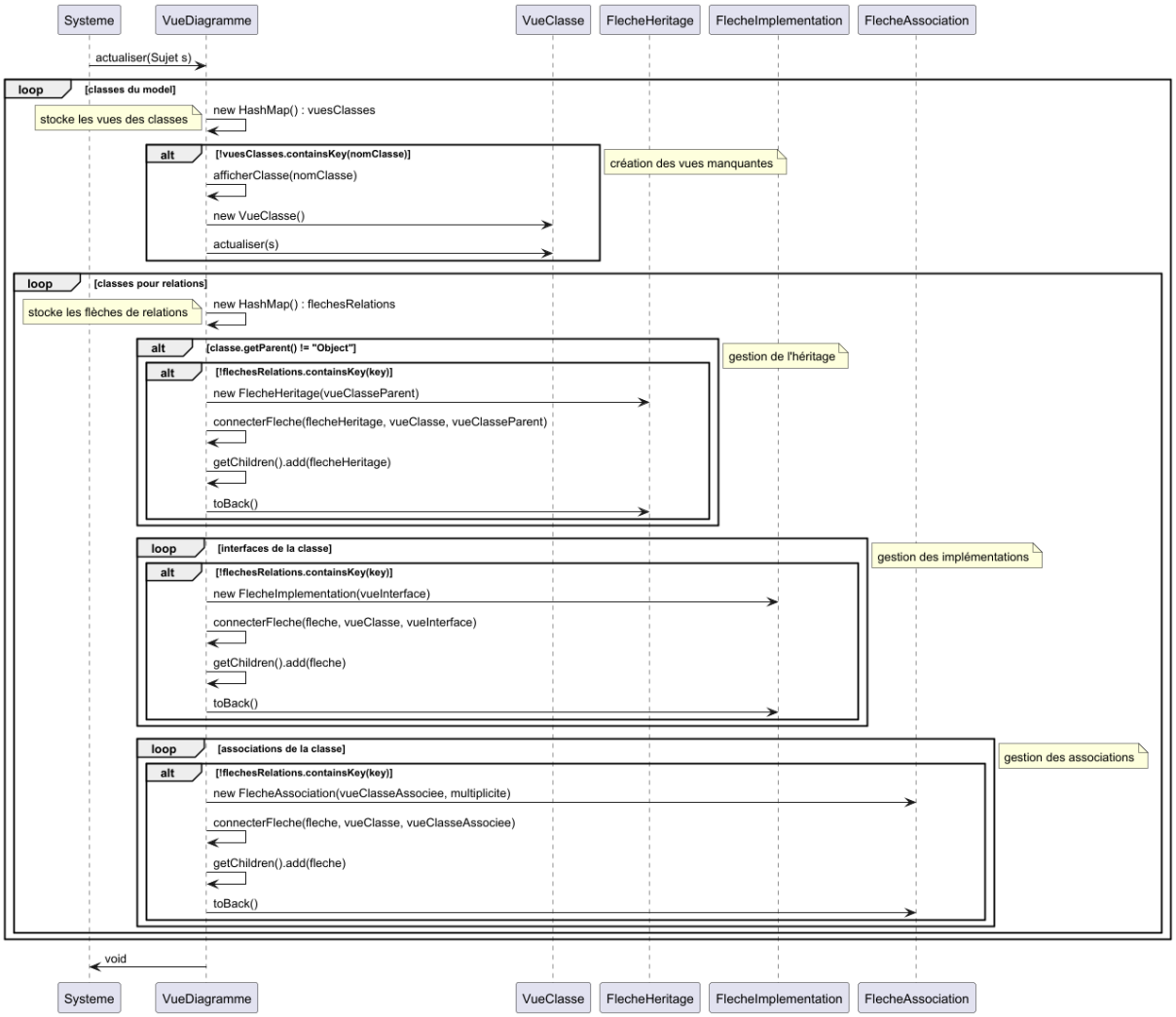
Graphe de scène



En haut de l'interface se trouve une ToolBar qui contient des options comme sauvegarder, réinitialiser, changer de dossier, créer une classe…

Au centre se trouve un SplitPane qui contient un TreeView (VueArborescence) et un Pane (VueDiagramme). C'est dans ce Pane que l'ont créé des VuesClasse (VBox) représentant des classes. Le TreeView affiche une arborescence de FileComposite, correspondant aux fichiers d'un dossier

Lorsqu'on ajoute des classes au diagramme, on ajoute des VueClasse comme enfant de VueDiagramme. Ce processus se fait à chaque appel de la méthode actualiser() de vueDiagramme, dont voici le diagramme de séquence :



Mode d’emploi

Notre projet ne contient aucun module supplémentaire. Nous utilisons seulement Maven et JavaFX. Pour utiliser l’application, il faut cloner le dépôt git dans un dossier, puis ouvrir ce dossier en tant que projet IntelliJ. Pour lancer l’application, il faut exécuter le fichier Interface.java.