Temirov Abdoul-Raouf
Audinot Noah
Thiriet Esteban
Besançon Marcelin

SAE 3.01 – Développement d'une application

1. Fonctionnalités

Idée des fonctionnalités

1. Extraction des classes, attributs, méthodes et relations (héritage, associations) à partir du code source.
2. Ajout, suppression et modification manuelle des éléments (classes, attributs, méthodes, relations).
3. Réorganisation des éléments dans l'interface (drag-and-drop).
4. export diagramme PNG, PDF, JSON
5. Ajout de commentaires ou d'annotations collaboratives.
6. Choix des couleurs et styles pour les classes et relations.
7. Options de filtrage pour afficher ou masquer certains types de relations ou détails (par exemple, afficher uniquement les héritages, adapter le niveau de détail affiché).

Fonctionnalités choisies:

ID	Fonctionnalité
1	Générer le squelette des classes
2	Afficher les classes existantes
3	Afficher les relations entre les classes
4	Déplacer les classes dans l'interface
5	Ajouter ou supprimer des classes manuellement
6	Ajouter des attributs à une classe manuellement
7	Ajouter des méthodes à une classe manuellement
8	Ajouter des constructeurs à une classe manuellement
9	Afficher ou cacher des classes dans le diagramme
10	Afficher ou cacher des méthodes dans le diagramme
11	Afficher ou cacher les interfaces dans le diagramme
12	Importer un fichier ou un dossier pour analyser le code
13	Exporter le diagramme sous forme d'image
14	Supprimer tout ce qui est affiché
15	Ajouter ou supprimer des relations
16	Gestion clic droit

Description textuelle :

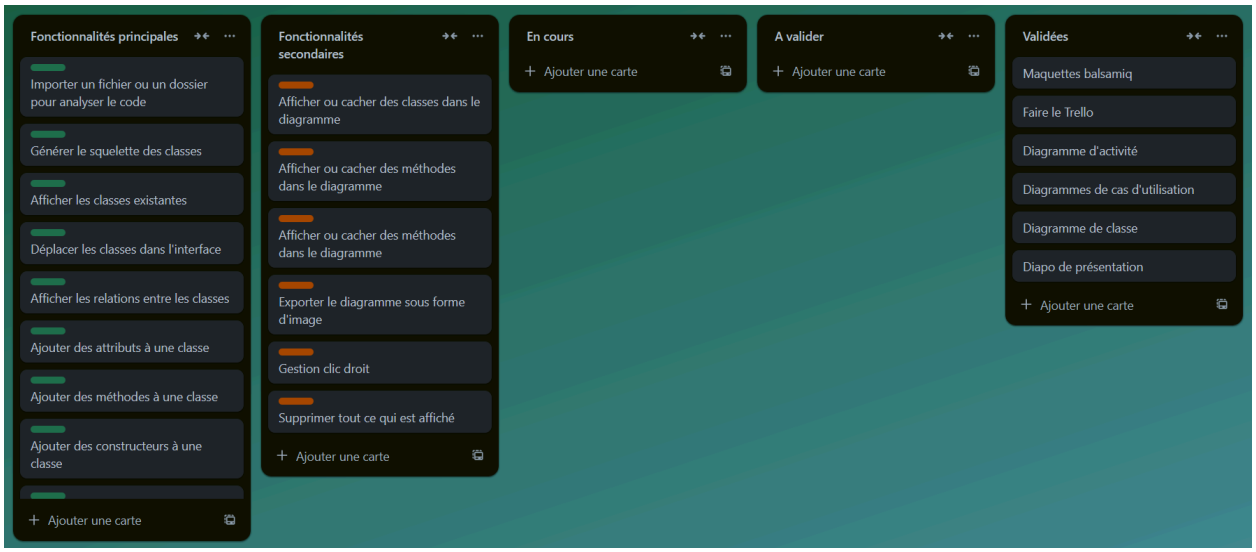
- L'utilisateur peut charger un fichier ou un dossier contenant du code source Java. Le système analysera automatiquement ce code pour identifier les classes, les méthodes, les attributs et les relations, et générera un diagramme UML.
- L'utilisateur peut créer de nouvelles classes ou supprimer des classes existantes directement dans le diagramme. Cela permet de compléter ou de simplifier le modèle généré.
- Il est possible pour l'utilisateur d'ajouter des attributs spécifiques à une classe dans le diagramme. Cela peut être utile pour enrichir les détails ou représenter des modifications futures.
- L'utilisateur peut également compléter une classe en ajoutant des méthodes spécifiques, en définissant leur visibilité (public, privé, protégé) et leurs paramètres.
- L'utilisateur a la possibilité d'insérer des constructeurs dans une classe, permettant une représentation plus complète des fonctionnalités de la classe.
- Pour améliorer la lisibilité, l'utilisateur peut choisir de masquer ou d'afficher certaines classes du diagramme.
- L'utilisateur peut également décider de masquer ou de montrer les interfaces identifiées dans le code ou ajoutées manuellement.

- Grâce à l'interface interactive, l'utilisateur peut réorganiser les classes affichées sur le diagramme pour optimiser l'agencement visuel.
- Une fonctionnalité de zoom permet à l'utilisateur d'obtenir une vue d'ensemble complète.
- Une fois le diagramme complété ou modifié, l'utilisateur peut l'exporter dans des formats tels que PNG ou SVG.
- Si nécessaire, l'utilisateur peut réinitialiser l'affichage en supprimant toutes les classes, relations, et autres éléments du diagramme.

2. Planning temporaire des itérations :

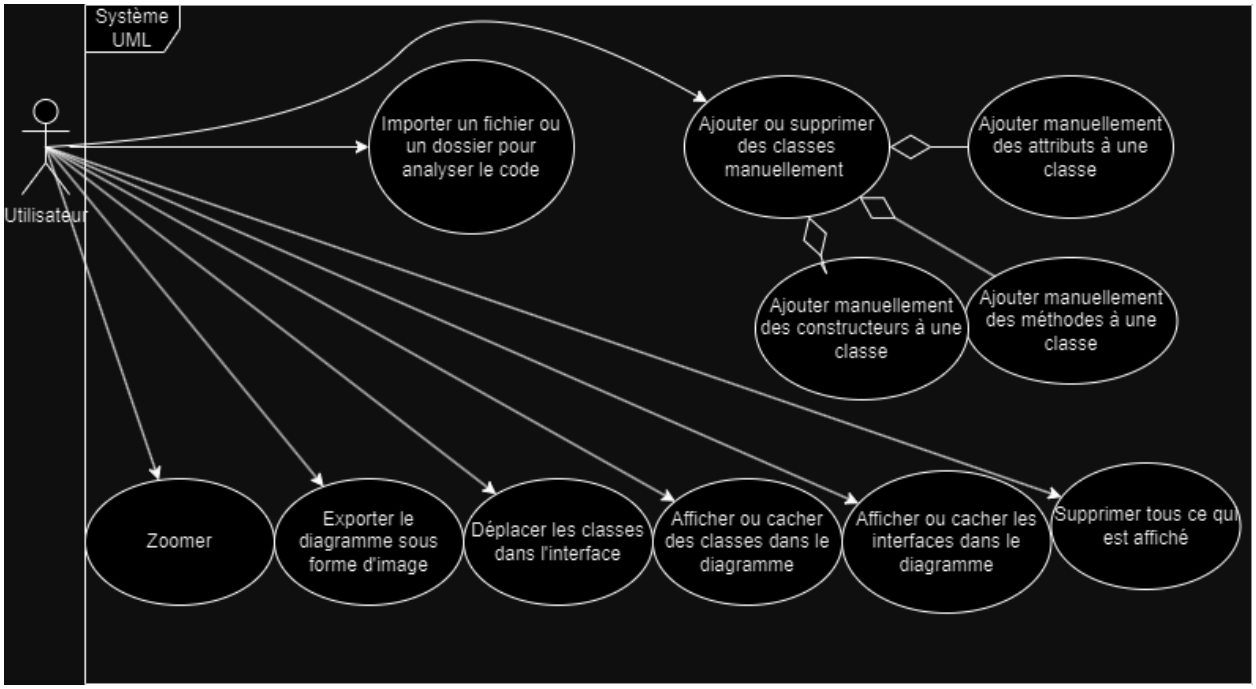
Itération	Objectifs principaux (cas d'utilisation)	Risques identifiés
1	Analyse des besoins, rédaction des cas d'utilisation, création des diagrammes préliminaires (cas d'utilisation, activités).	Mauvaise compréhension des besoins ou couverture insuffisante des cas d'utilisation.
2	- Développement d'une fonctionnalité de base : génération du squelette de classe.	Difficultés à définir un format standard pour représenter les classes et leurs éléments.
3	Extraction automatique des classes, attributs, méthodes et relations. - Tests initiaux pour garantir une extraction fiable.	Extraction incomplète ou incorrecte des relations (relations mal interprétées).
4	- Création de l'interface graphique basique (visualisation des classes et relations).	- Problèmes d'ergonomie ou d'utilisation intuitive.
5	- Ajout, suppression et modification manuelle des éléments (classes, attributs, méthodes, relations).	- Synchronisation incorrecte entre données et interface graphique.
6	- Implémentation du drag-and-drop pour réorganiser les éléments.	- Gestion des collisions ou comportements inattendus dans le repositionnement.
7	- Implémentation des fonctionnalités d'export	- Rendu non conforme ou problèmes de compatibilité des fichiers exportés.
8	- Mise en place des options de filtrage	- Gestion des collisions ou comportements inattendus dans le repositionnement.
9	Tests approfondis, correction des bugs, documentation utilisateur et technique	Temps insuffisant pour couvrir tous les tests ou documentation incomplète.

Trello :


Fonctionnalités principales
Importer un fichier ou un dossier pour analyser le code
Générer le squelette des classes
Afficher les classes existantes
Déplacer les classes dans l'interface
Afficher les relations entre les classes
Ajouter des attributs à une classe
Ajouter des méthodes à une classe
Ajouter des constructeurs à une classe
Ajouter une carte
Fonctionnalités secondaires
Afficher ou cacher des classes dans le diagramme
Afficher ou cacher des méthodes dans le diagramme
Afficher ou cacher des méthodes dans le diagramme
Exporter le diagramme sous forme d'image
Gestion clic droit
Supprimer tout ce qui est affiché
Ajouter une carte
En cours
Ajouter une carte
A valider
Ajouter une carte
Validées
Maquettes balsamiq
Faire le Trello
Diagramme d'activité
Diagrammes de cas d'utilisation
Diagramme de classe
Diapo de présentation
Ajouter une carte

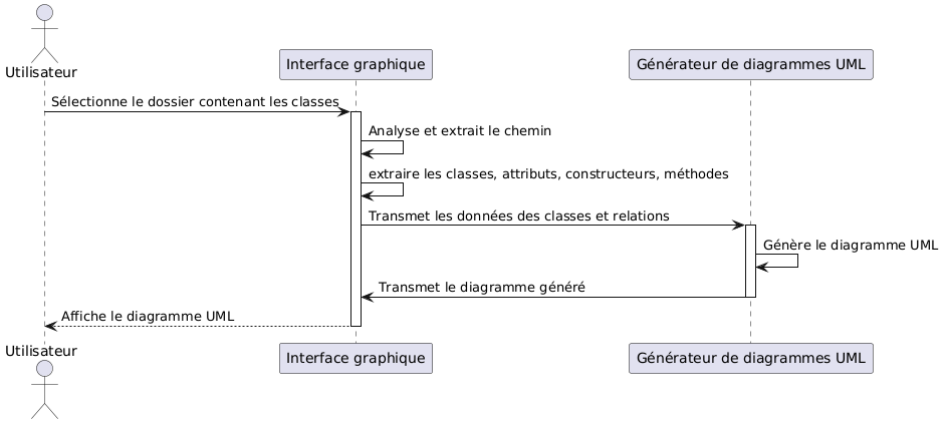
3. Diagrammes :

Cas d’utilisation global :

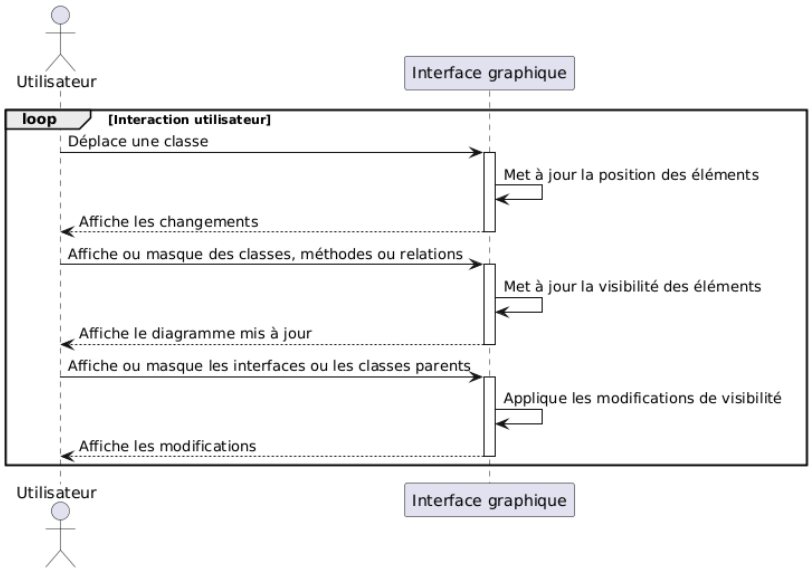


Cas d'utilisation détaillées :

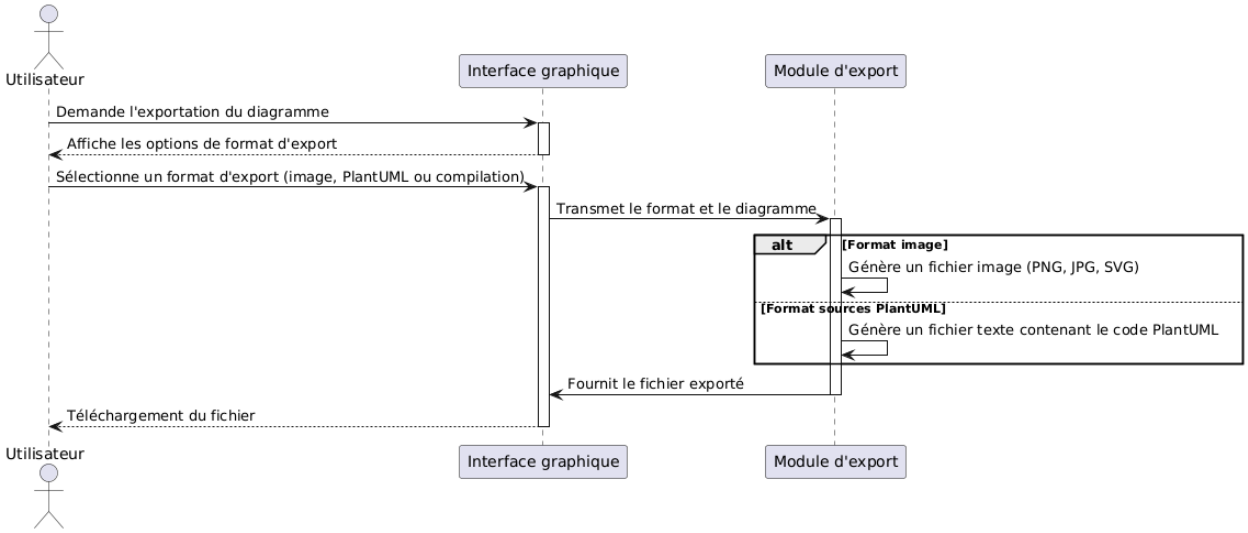
- représenter les classes et les relations entre classes d'un package JAVA en utilisant l'introspection (la capacité qu'a une classe JAVA de connaître ses attributs, ses constructeurs et ses méthodes) ;
- générer les squelettes des classes construites au sein de cette application.



- générer à partir de cette représentation des diagrammes de classe à l'aide d'une interface
  graphique simple d'utilisation (déplacer les classes sur l'écran, afficher / masquer certaines
  classes ou méthodes, afficher / masquer la classe parent et les interfaces d'une classe, …) ;



- exporter les diagrammes de classe sous différents formats (image, sources plantUML, résultat d'une compilation plantUML) ;



- modifier le diagramme résultant en ajoutant de nouvelles classes ou des méthodes à des classes existantes ;

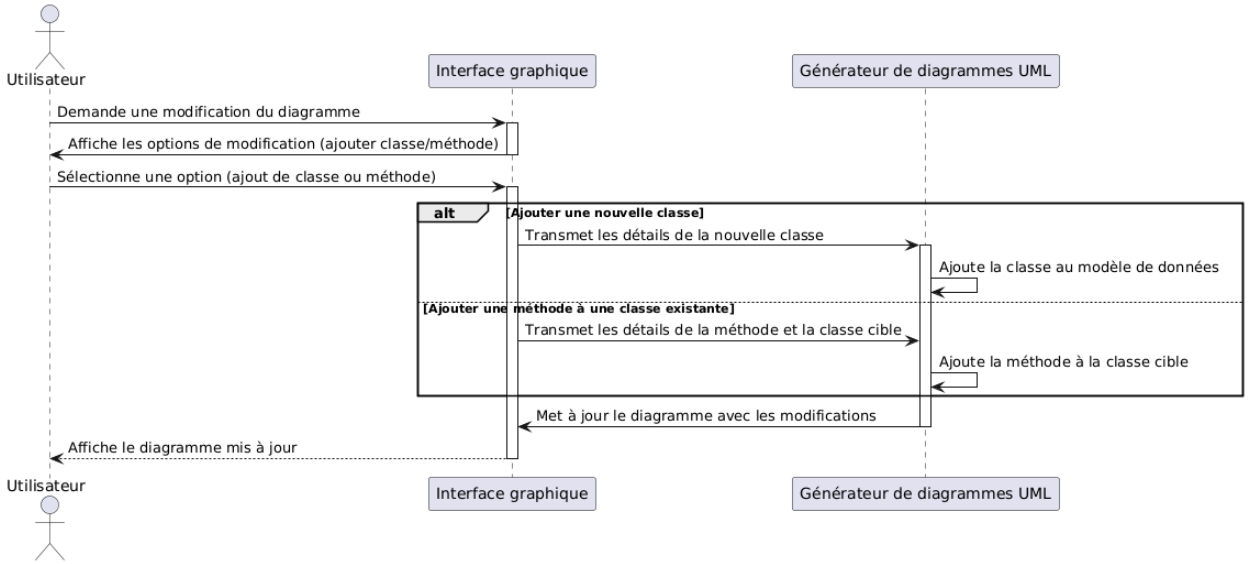


Diagramme de classe :

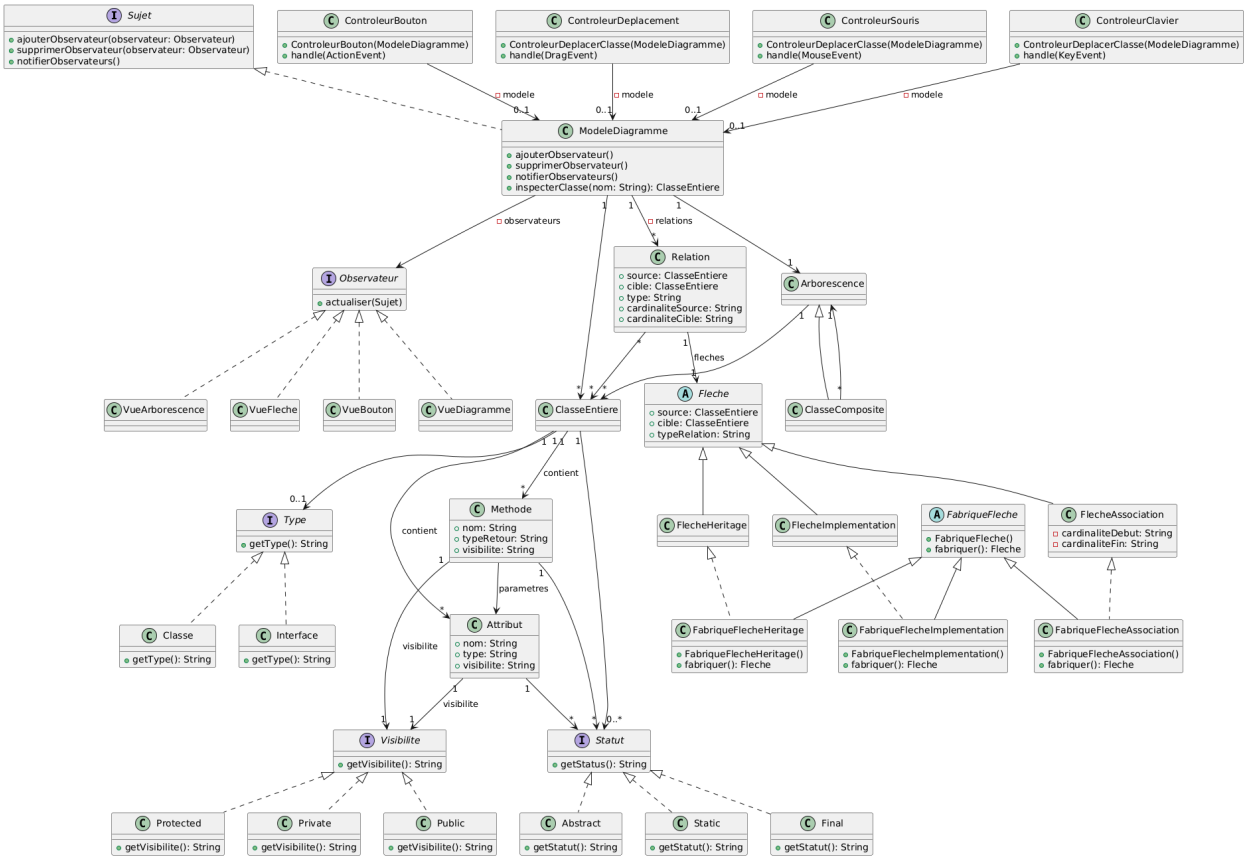
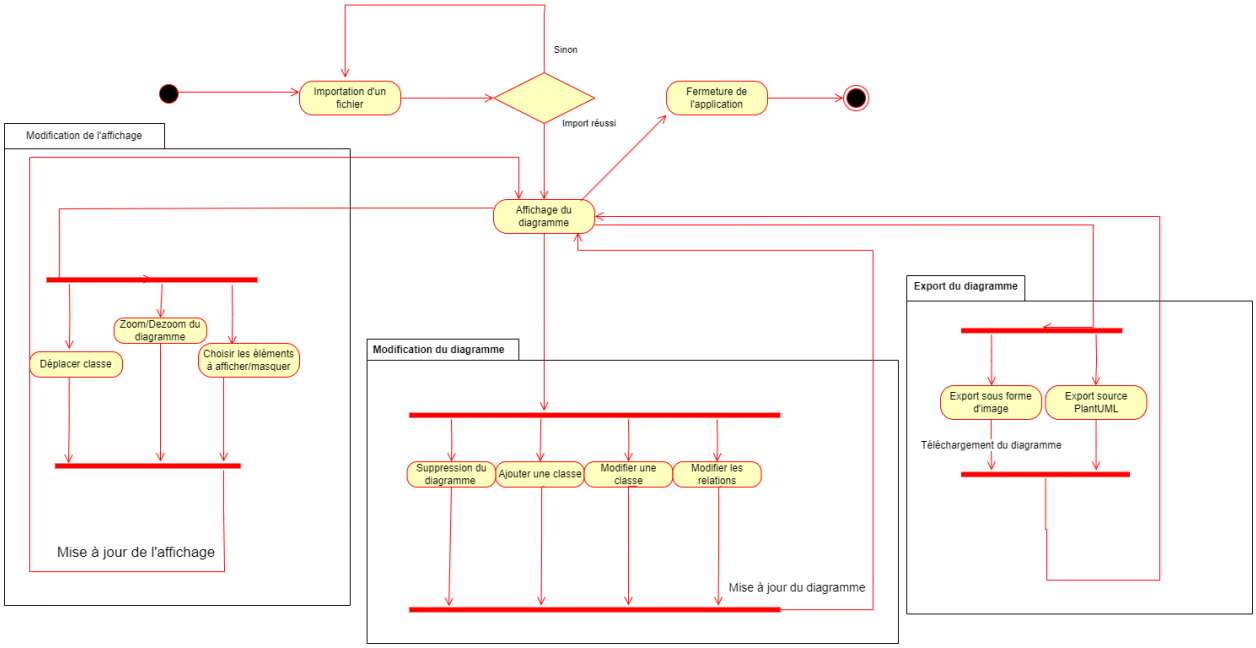
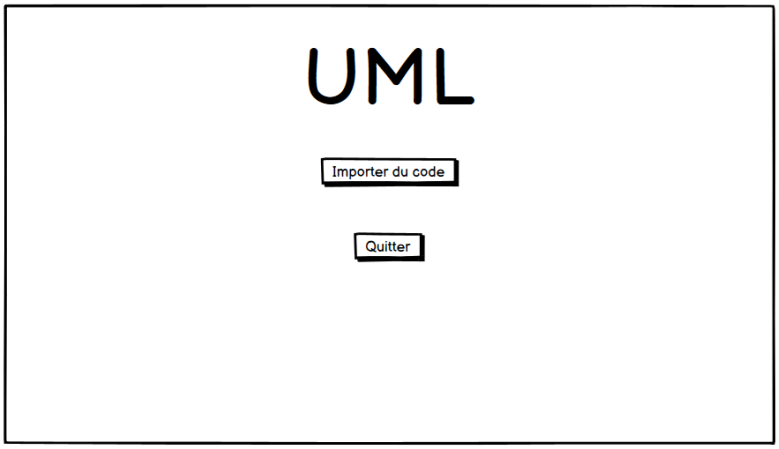


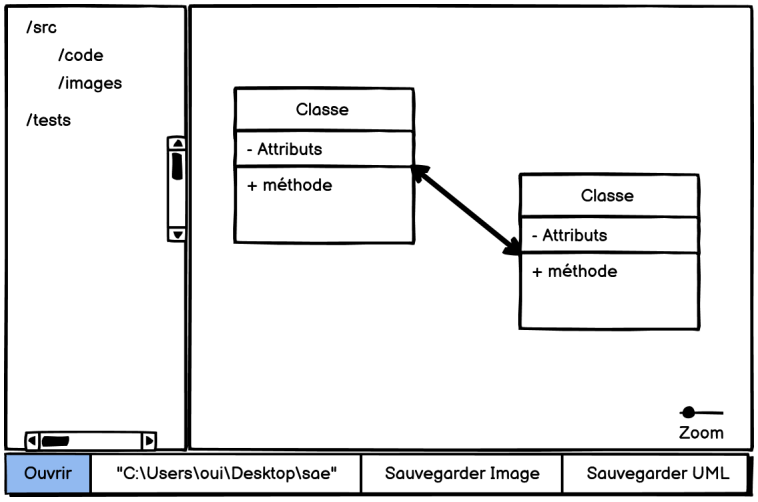
Diagramme d’activité :



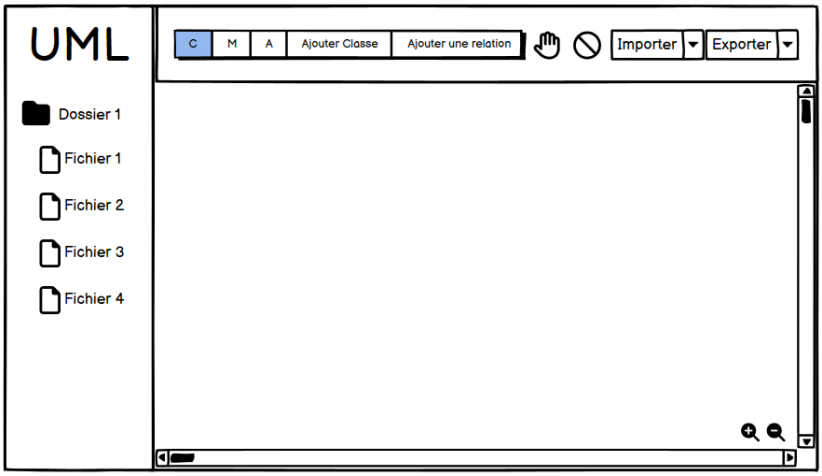
4. Maquettes :

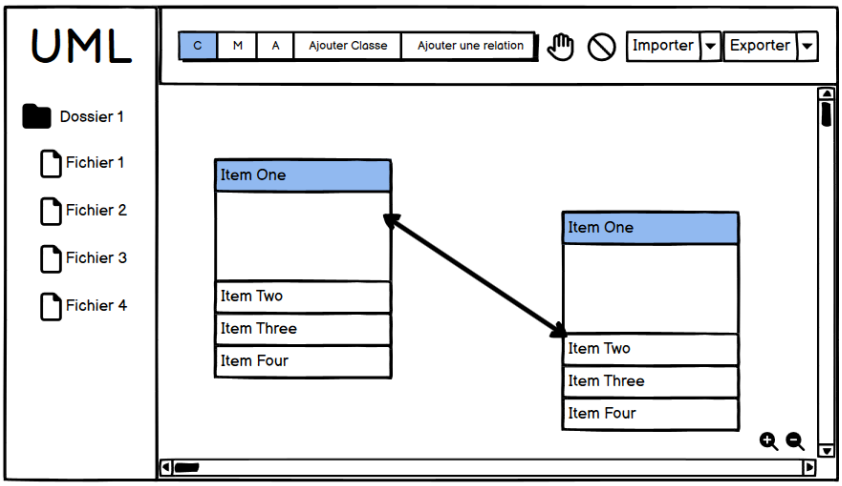


Exemple maquette 1:



Exemple maquette 2 :


UML
Dossier 1
Fichier 1
Fichier 2
Fichier 3
Fichier 4
C
M
A
Ajouter Classe
Ajouter une relation
Importer
Exporter


UML
Dossier 1
Fichier 1
Fichier 2
Fichier 3
Fichier 4
C
M
A
Ajouter Classe
Ajouter une relation
Importer
Exporter
Item One
Item Two
Item Three
Item Four
Item One
Item Two
Item Three
Item Four