

Roberto NIGHTINGALE CASTILLO

Patrice GALLANT

Projet synthèse

420-C61-VM

Groupe 00001

C'est fou. Riez !

Document de conception

Travail présenté à

Monsieur Jean-Christophe Demers

Cégep du Vieux Montréal

Départements d'informatique

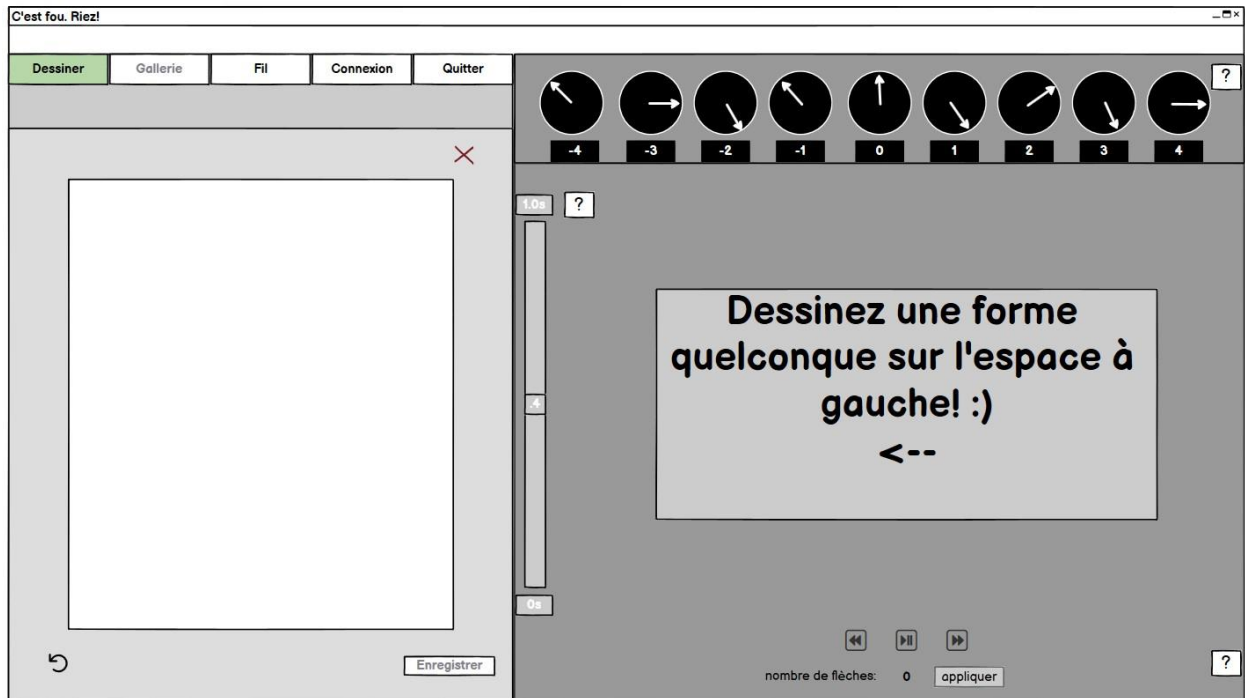
2 mars 2023

Table des matières

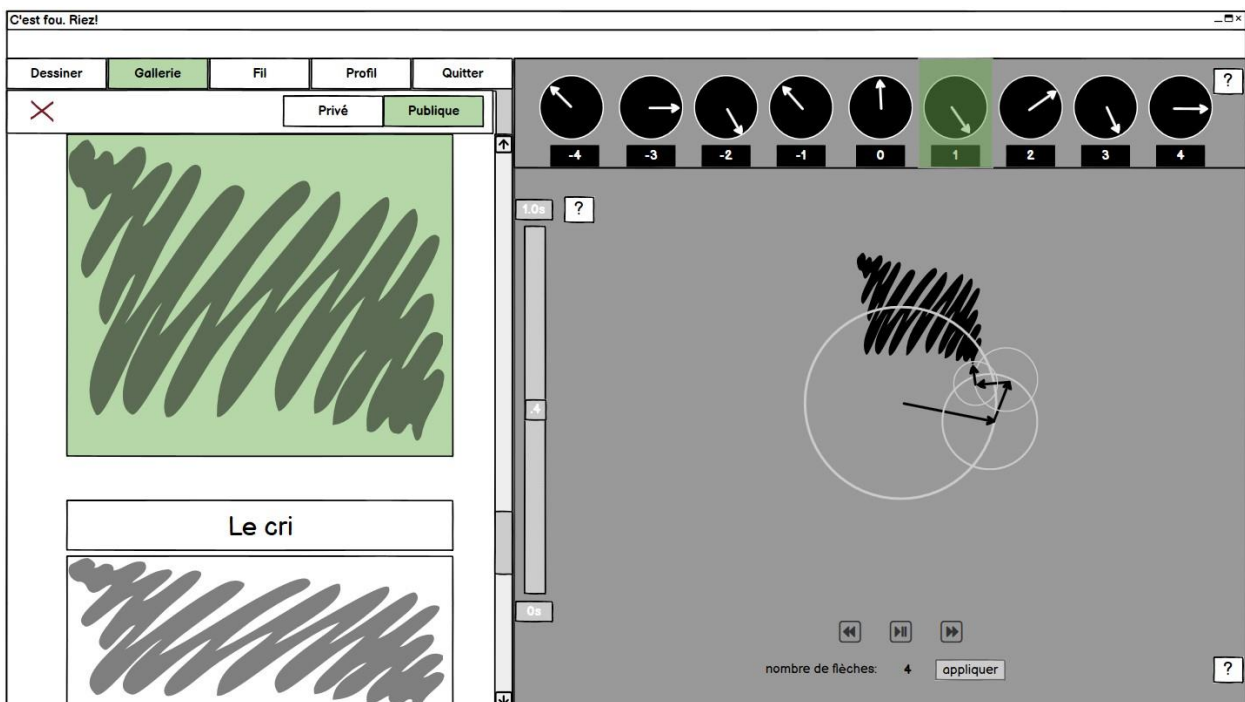
MAQUETTES	3
Fenêtre de dessin.....	3
Fenêtre de changement de mot de passe	5
Fenêtre de profil.....	6
CAS D'USAGE	7
Aire dessin	7
Changer les paramètres et préférences du profile usager.....	7
Fil d'actualité	8
Galerie.....	8
Inscription d'un nouveau compte.....	9
Animation flèches Fourier	9
Menu d'options	10
Modification du mot de passe	10
Panneau de connexion.....	11
DIAGRAMME DE CLASSES	12
Vue.....	12
Modèle	13
Serveur/DAO	13
Structure de données externe	14
Éléments de conception	14

MAQUETTES

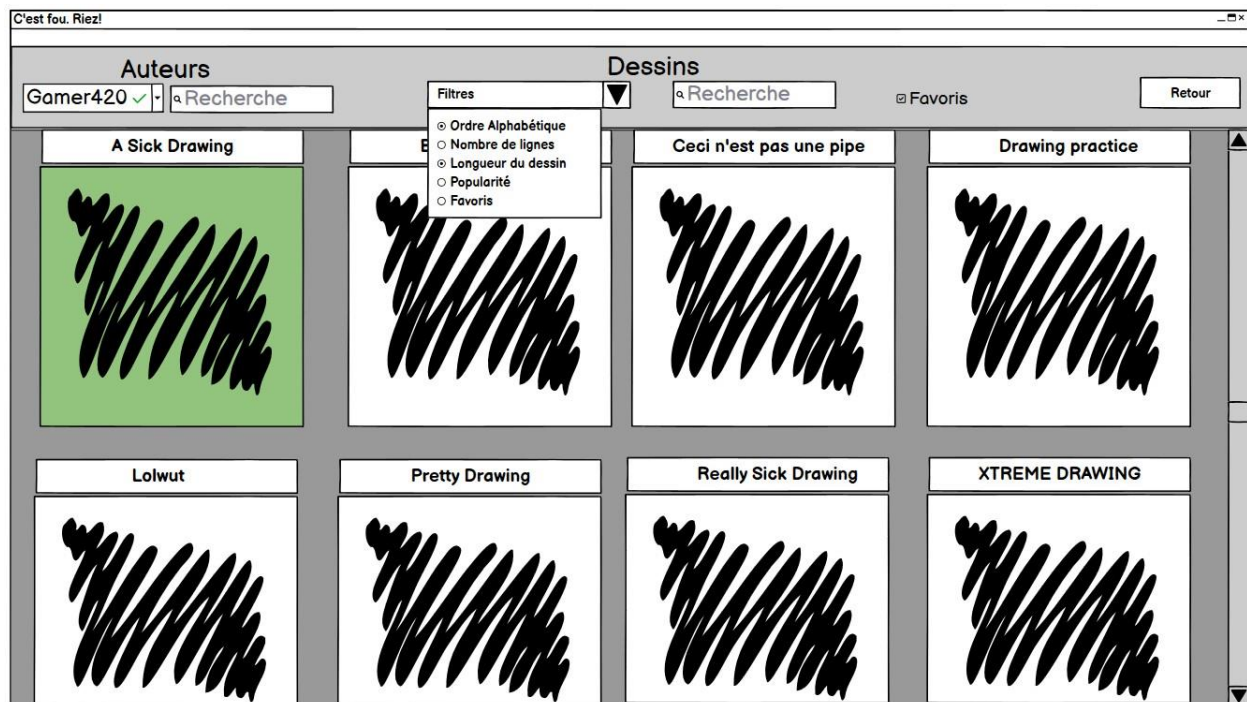
Fenêtre de dessin



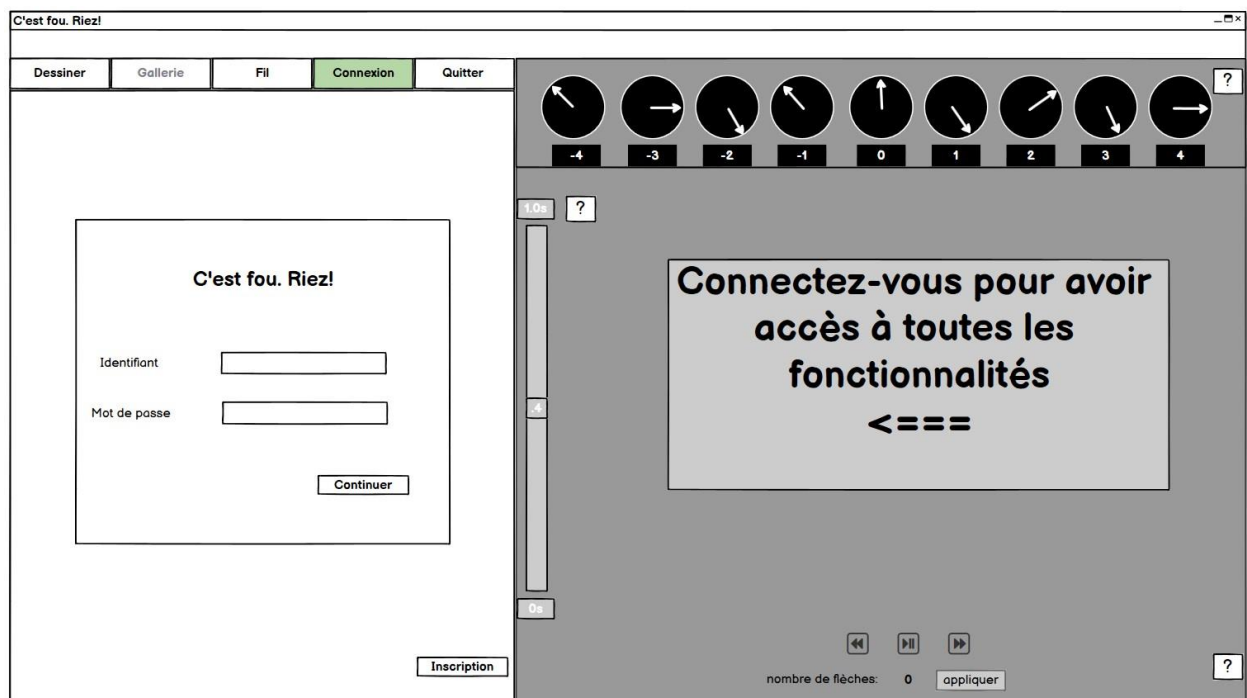
Fenêtre de galerie de dessins



Fenêtre de fil d'actualité



Fenêtre de connexion



Fenêtre de création de comptes

C'est fou. Riez!

Dessiner Galerie Fil **connexion** Quitter

(Espace pour erreur possibles, ex. mot de passes ne coïncident pas)***

Identifiant

Prénom

Nom

Email

Mot de passe

confirmer le mot de passe

S'inscrire

annuler

10s ?

0s

Remplissez la page suivante pour finaliser votre inscription

<===

nombre de flèches: 0 appliquer ?

Fenêtre de changement de mot de passe

C'est fou. Riez!

Dessiner Galerie Fil **profil** Quitter

(Espace pour erreur possibles, ex. mot de passes ne coïncident pas)***

ancien mot de passe

nouveau mot de passe

confirmer le nouveau mot de passe

appliquer

annuler

10s ?

0s

Remplissez la page suivante afin de modifier votre mot de passe

<===

nombre de flèches: 0 appliquer ?

Fenêtre de profil

C'est fou. Riez!

Dessiner


Galerie

Fil

Profil

Quitter

Informations



Identifiant

Prénom

Nom


Email


changer le mot de passe


Supprimer mon compte


Préférences

Couleurs

arrière plan: 

trait du dessin: 

cercles: 


flèches: 

mode sombre:


desactivé

Deconnexion


Appliquer tous les changements




-4




-3




-2




-1




0




1



2



3



4


?


1.0s


?

Voici votre profil! Vous pouvez changer vos préférences!

<===







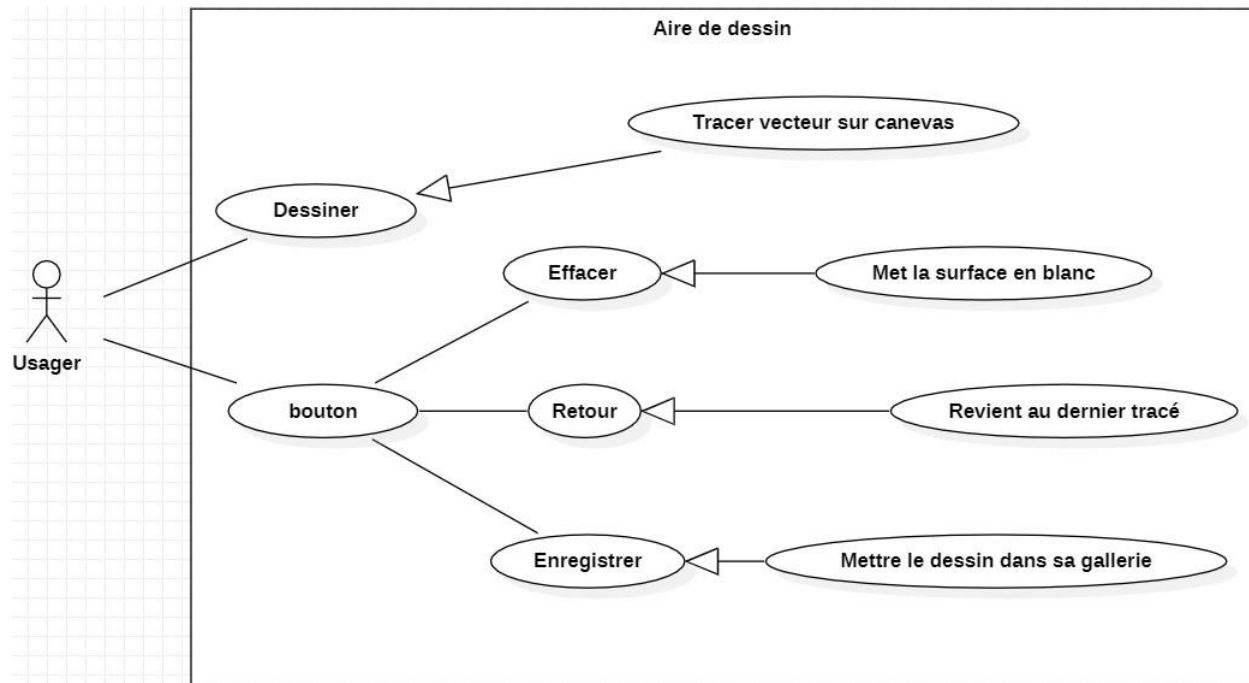
nombre de flèches: 0

appliquer

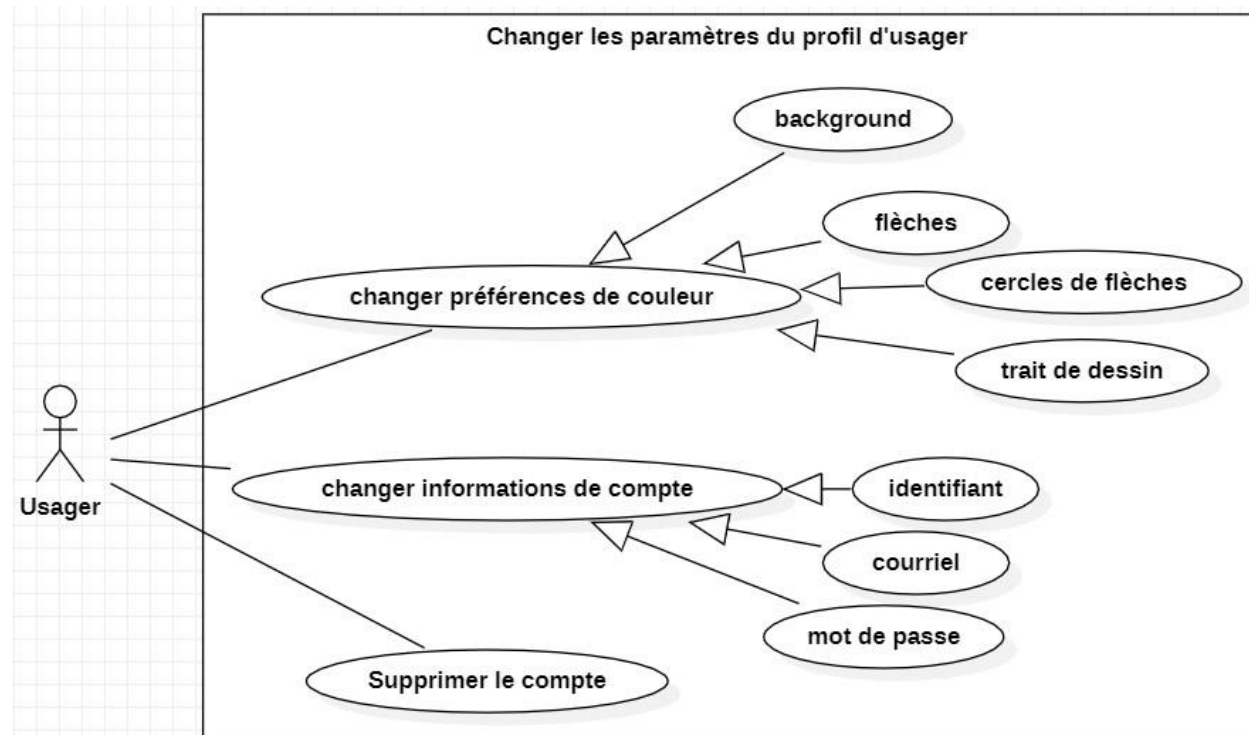
?

CAS D'USAGE

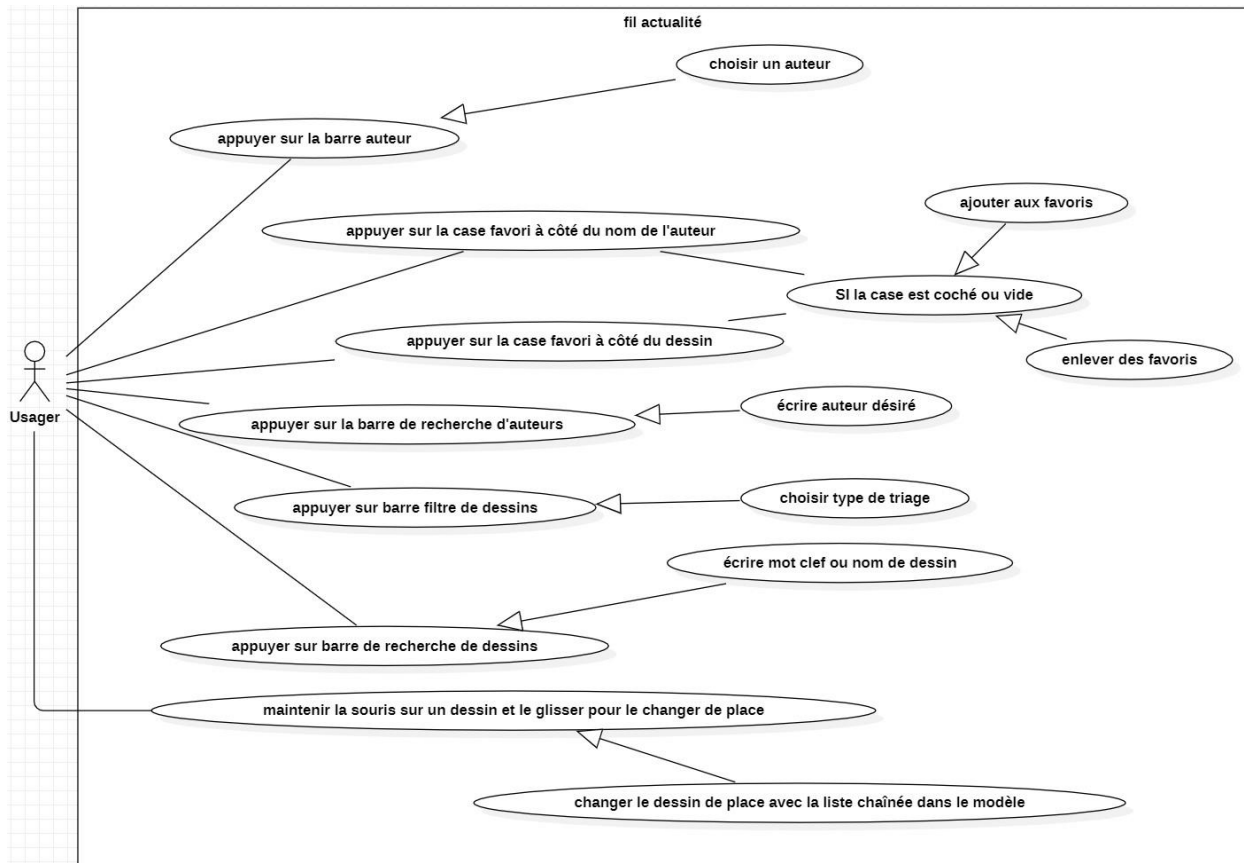
Aire dessin



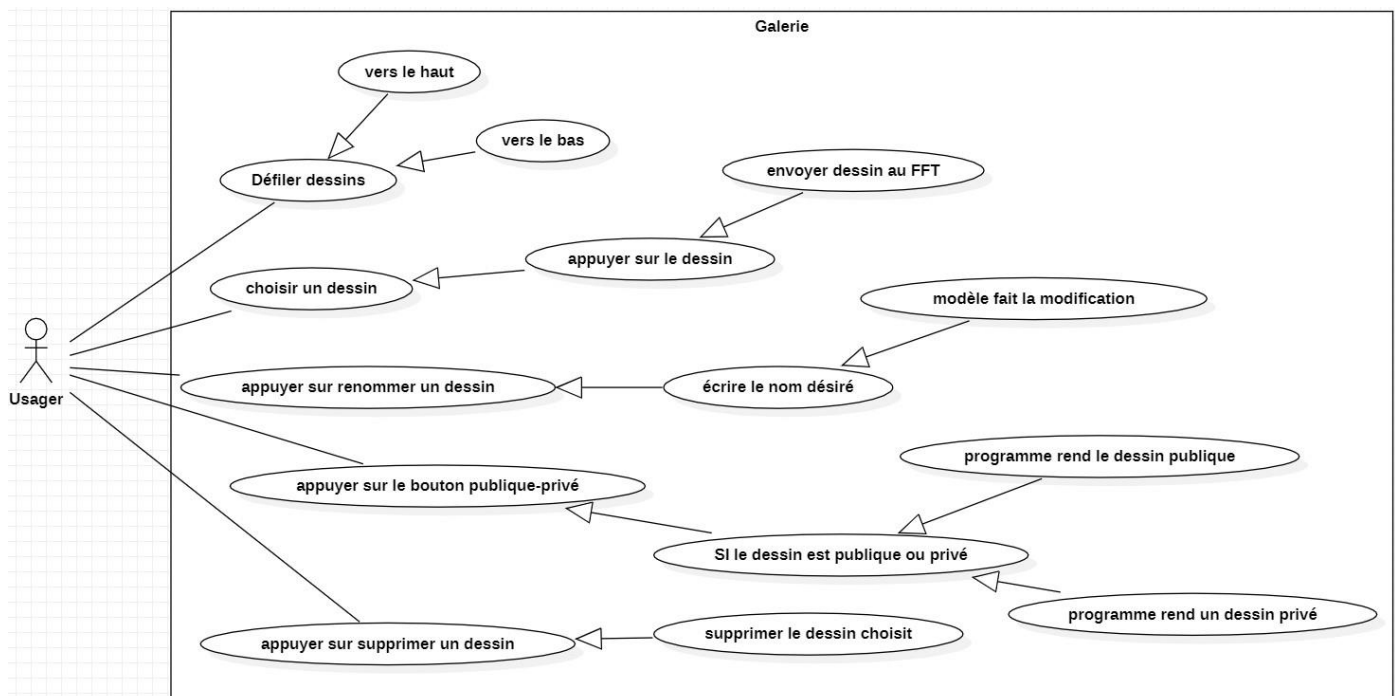
Changer les paramètres et préférences du profil usager



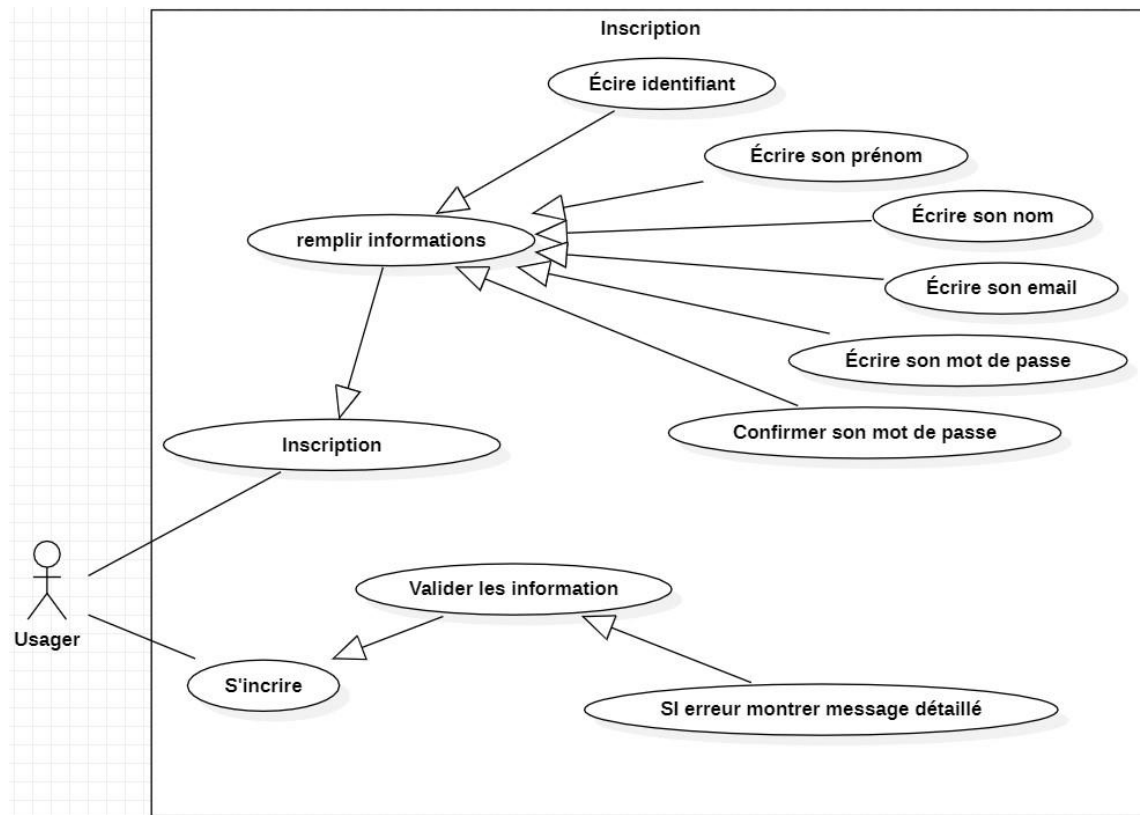
Fil d'actualité



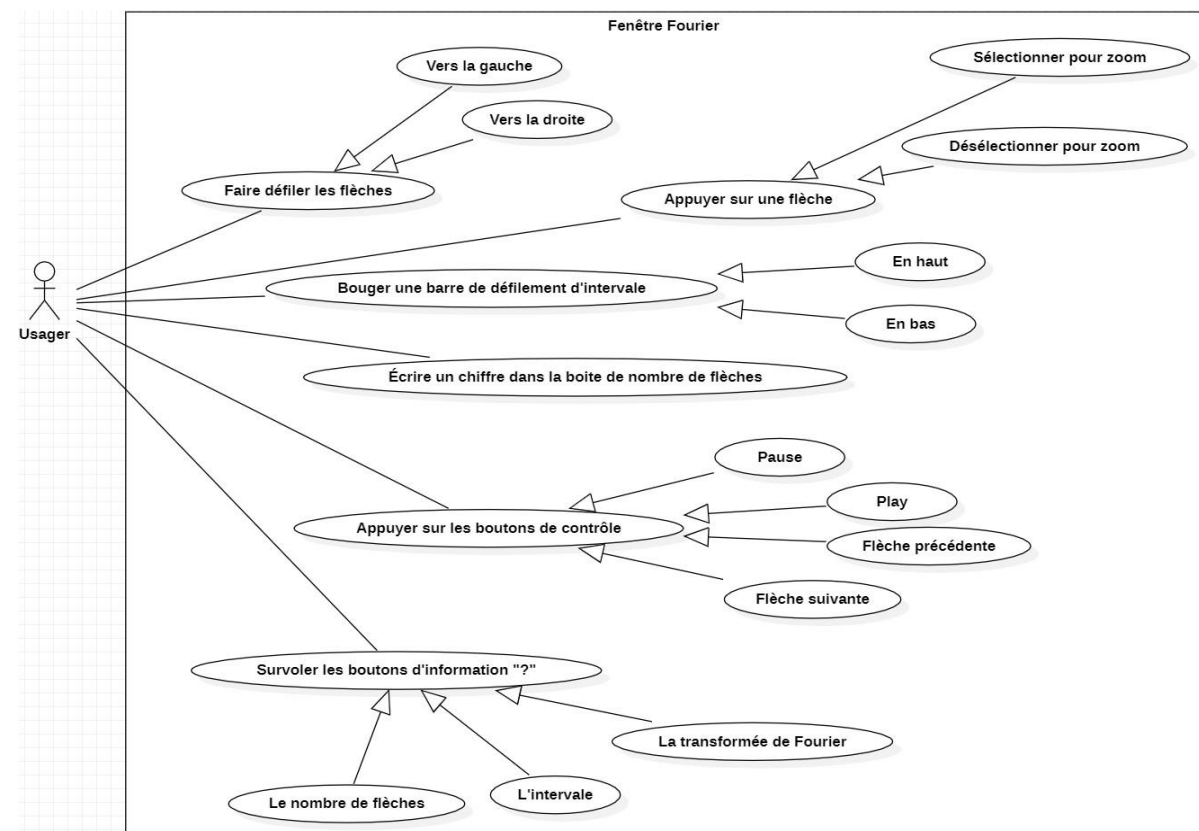
Galerie



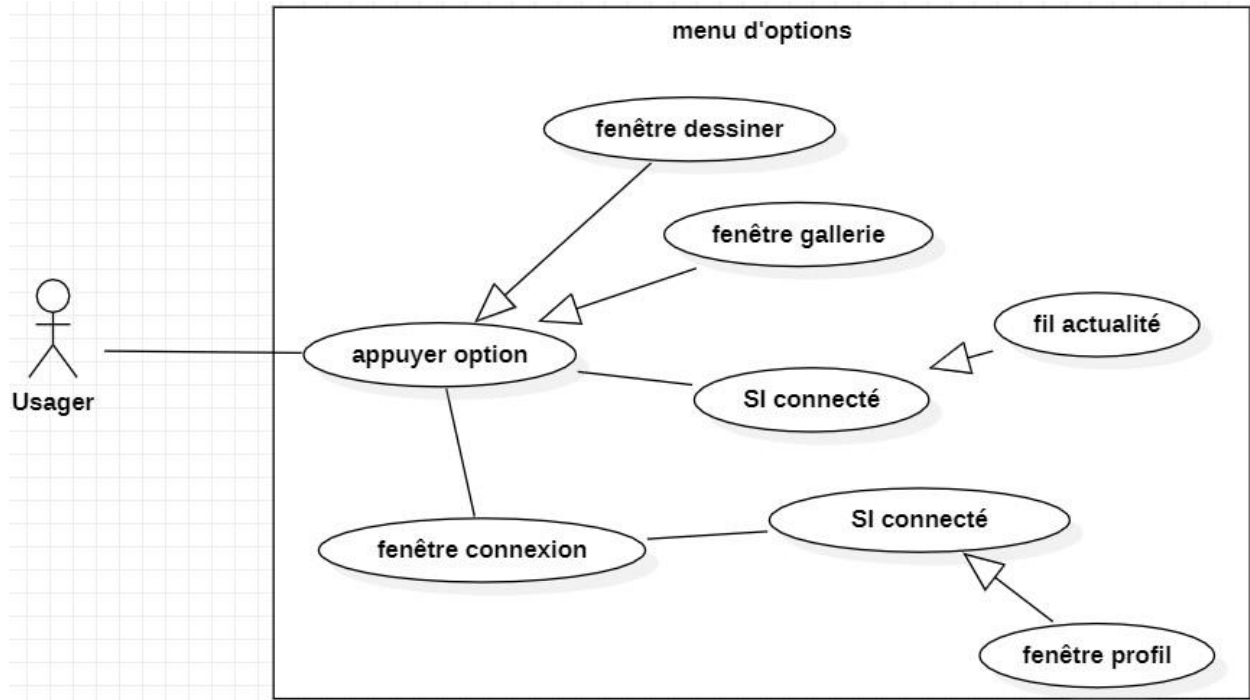
Inscription d'un nouveau compte



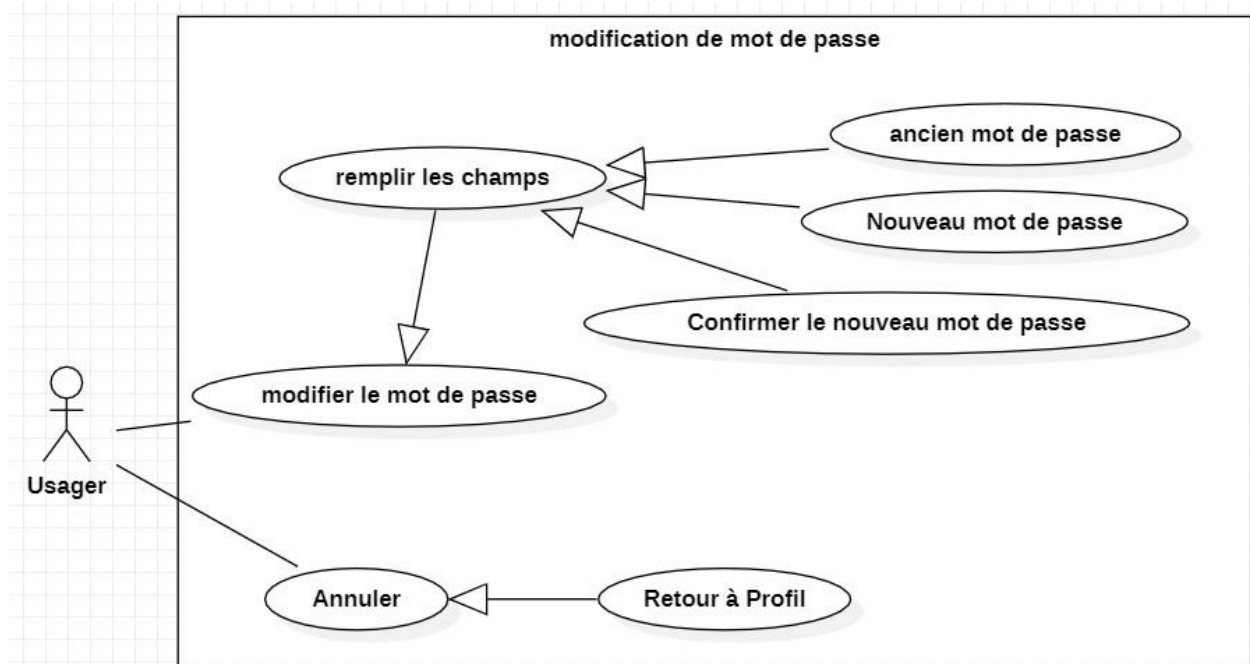
Animation flèches Fourier



Menu d'options



Modification du mot de passe



Panneau de connexion

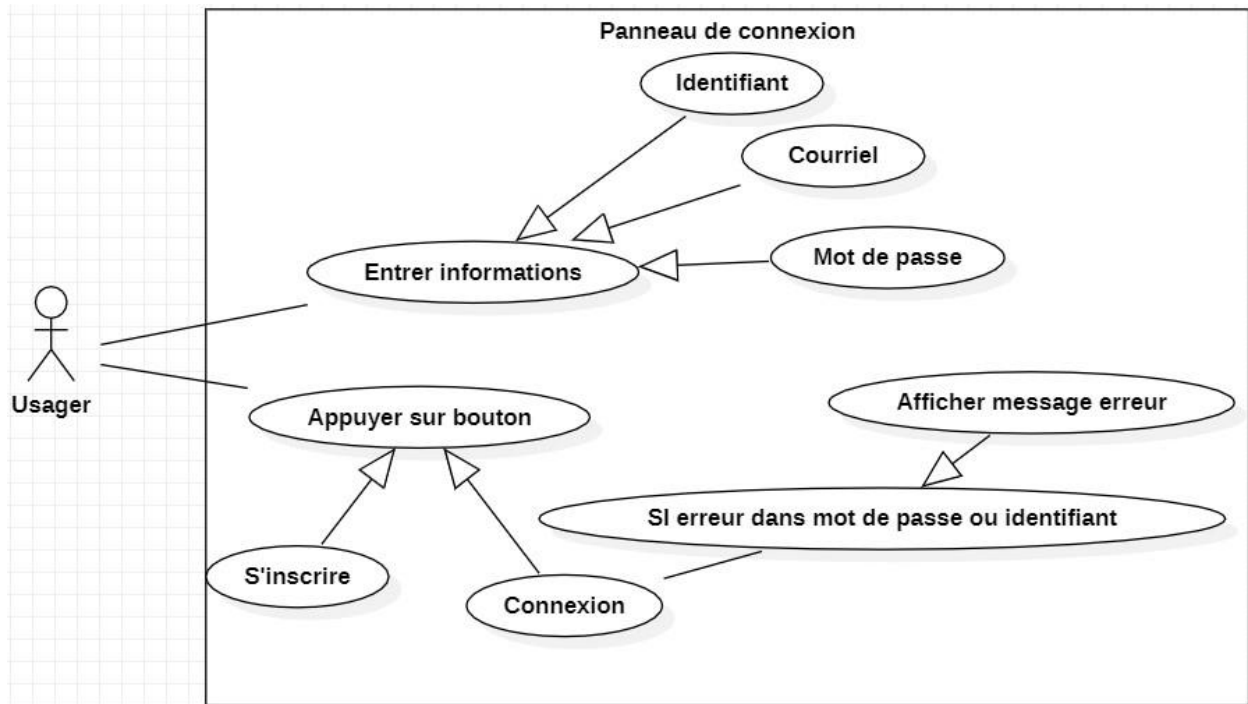
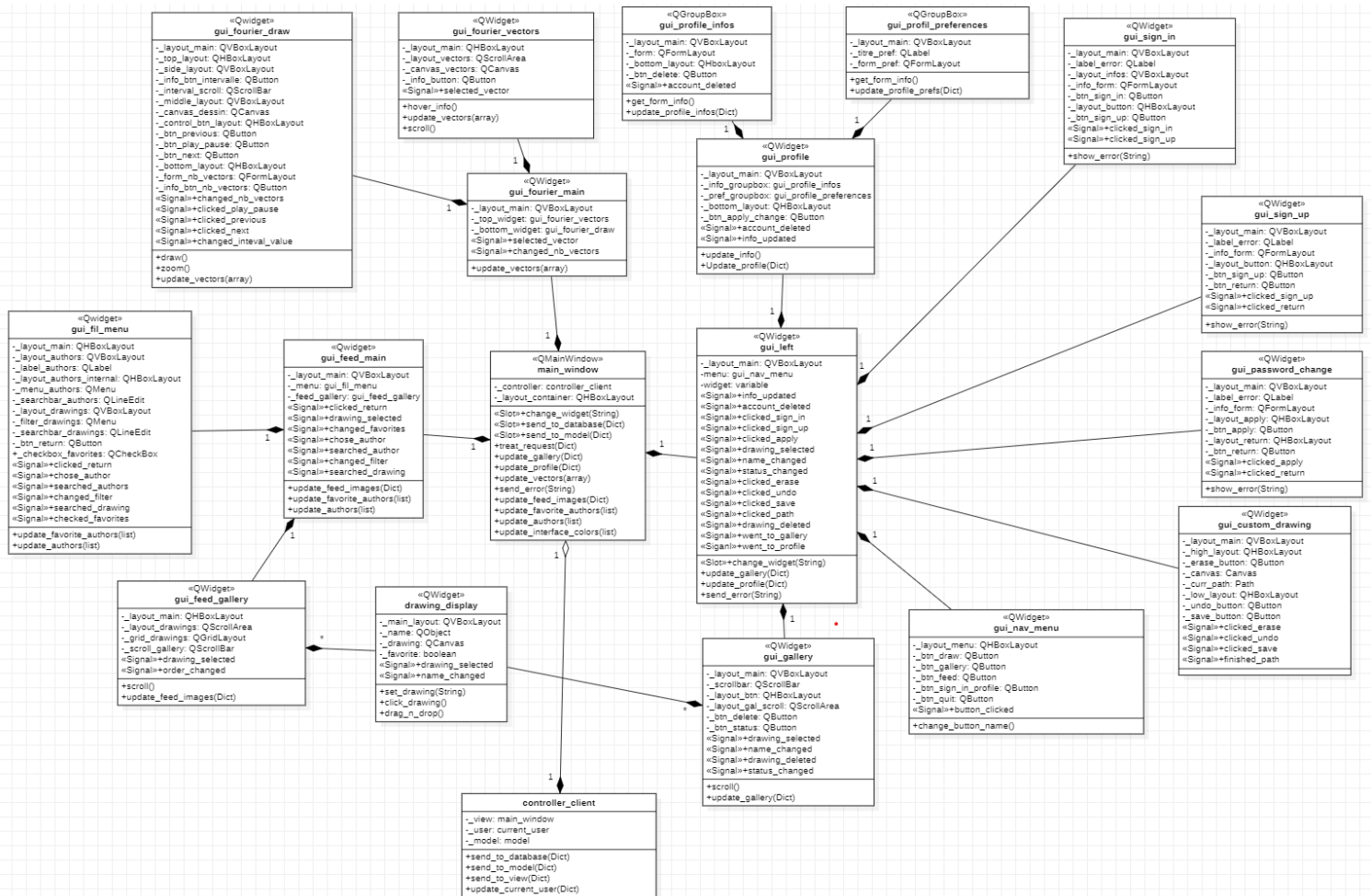
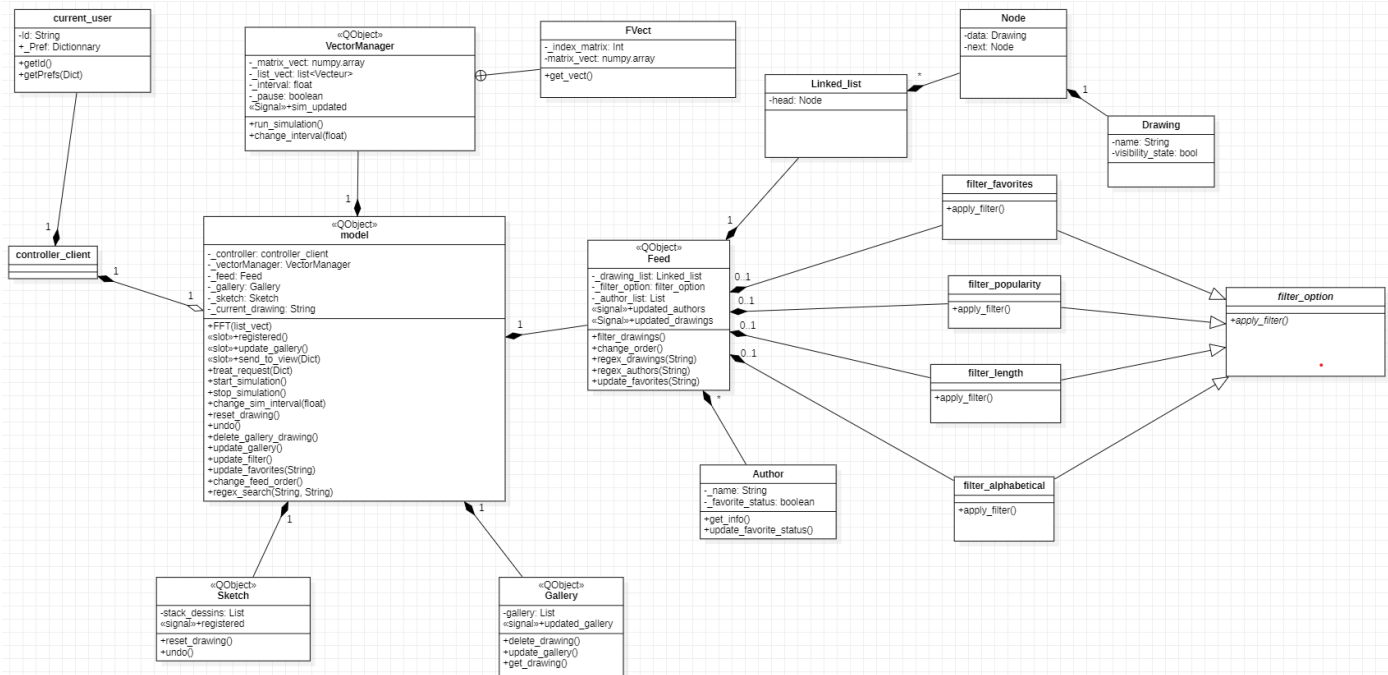


DIAGRAMME DE CLASSES

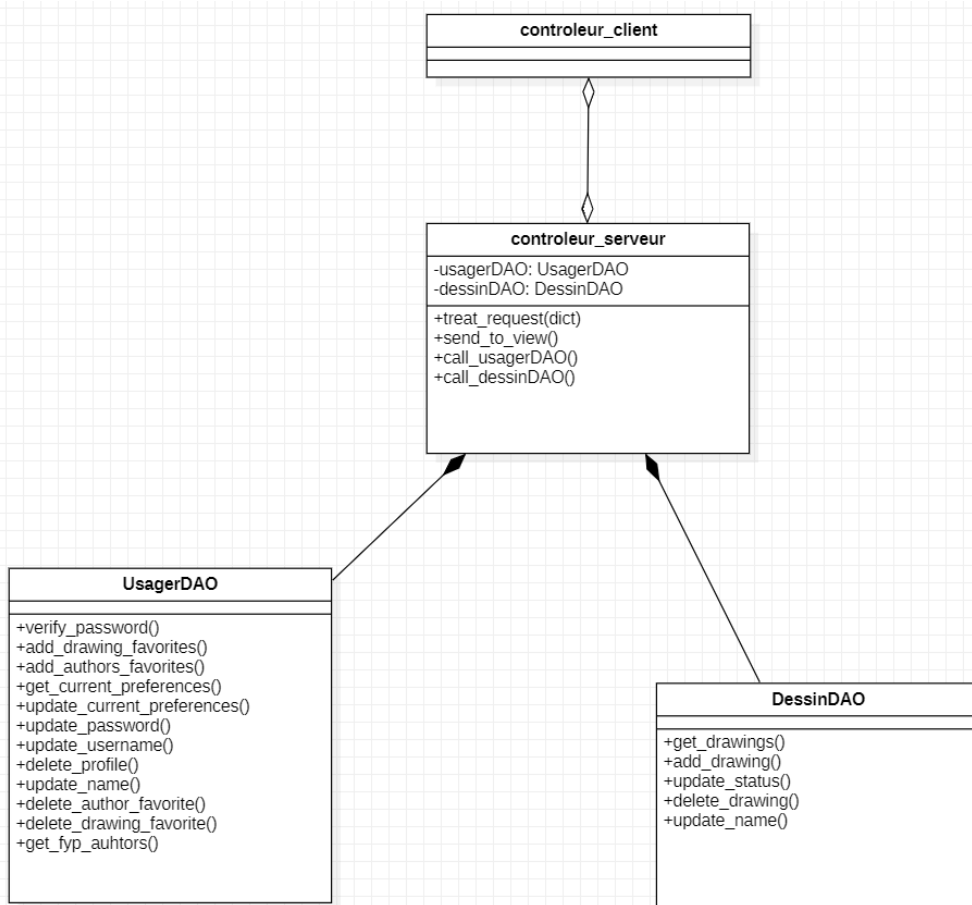
Vue



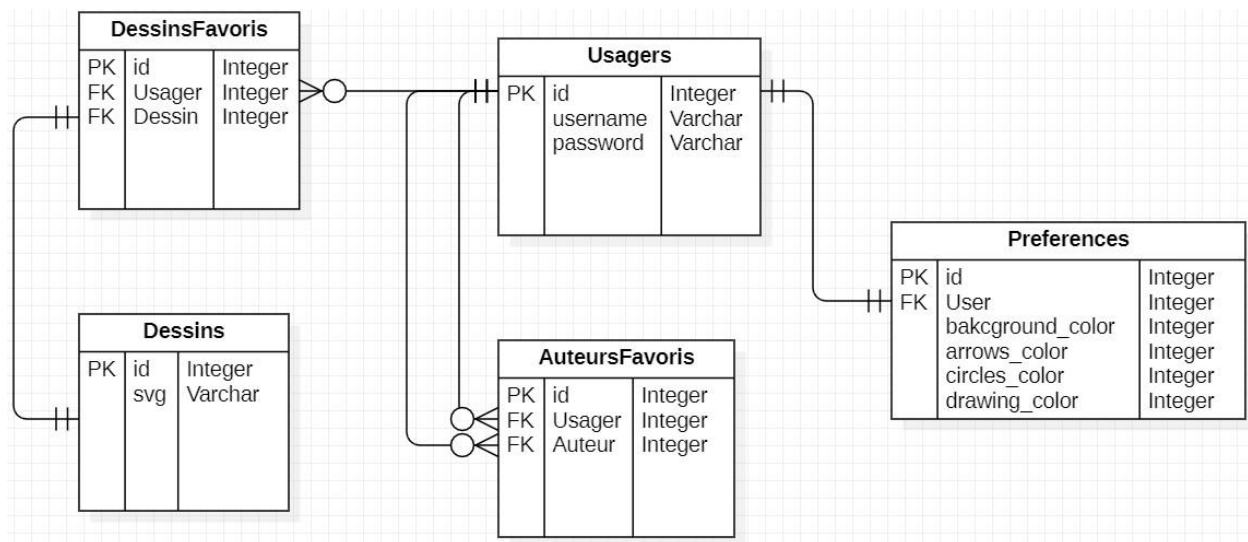
Modèle



Serveur/DAO



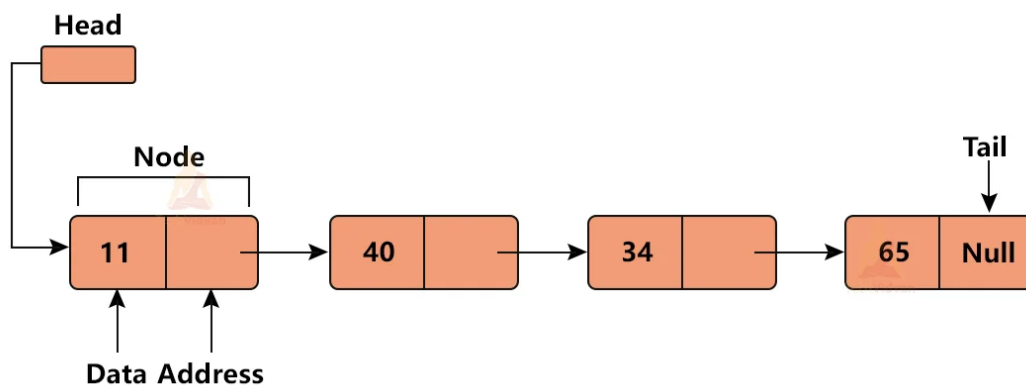
Structure de données externe



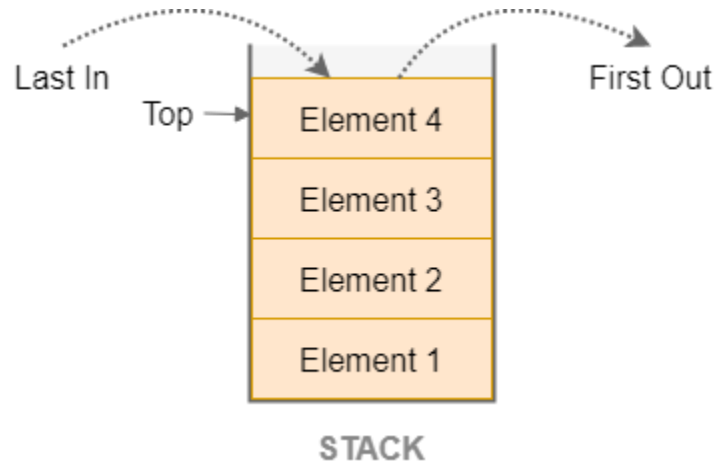
Éléments de conception

Structures de données (*Data Structures*)

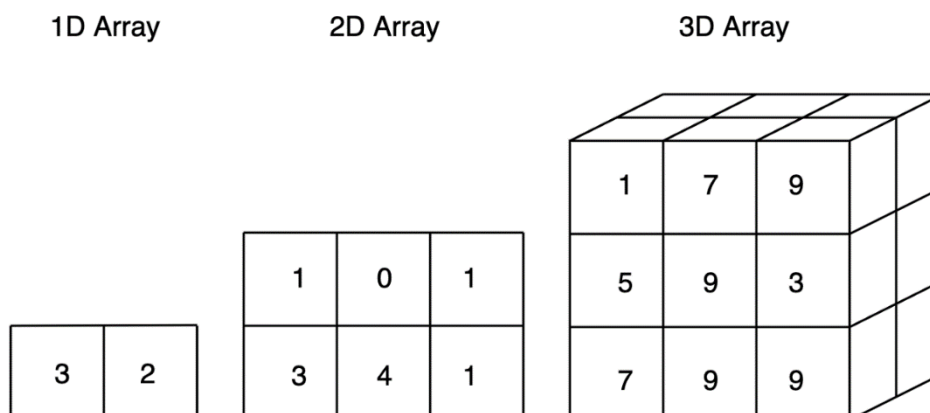
Dans le contexte de notre projet, nous allons avoir une galerie d'images exposant les dessins des différents usagers de l'application. Ces derniers pourront faire glisser des images d'un endroit à un autre en gardant la souris appuyée sur l'élément, cela afin d'organiser la galerie à leur manière. Pour accomplir cette tâche de manière optimale, nous allons faire l'usage d'une liste chaînée (*Linked List*). Cette structure de donnée (*Data Structure*) est munie d'une tête (*head*) étant le premier élément de la liste et d'une queue (*tail*) étant le dernier. À l'intérieur de la structure de données, il y aura des nœuds constitués de 2 parties. La première fait référence à l'objet venant de la classe: Dessin, qui contient les informations de celui-ci et la deuxième est une référence au nœud suivant. Dans le cas de la queue, son deuxième morceau ne pointera sur rien (*None*). Ayant maintenant ce concept en tête, il est plus simple de réordonner la liste en modifiant uniquement les pointeurs des nœuds. Cette structure de données est celle que nous comptons programmer de A à Z.



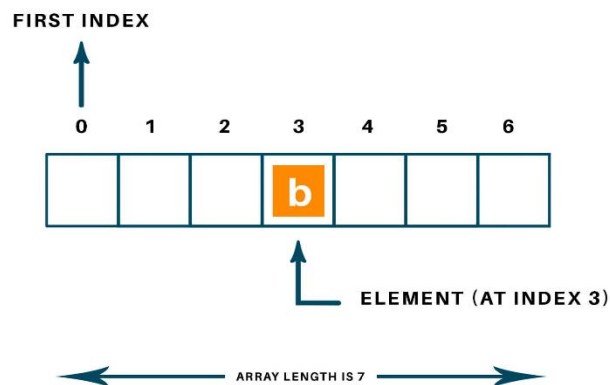
Considérant que le programme comprend une zone de dessin, nous voulons donner la possibilité à l'utilisateur de retourner à son dernier tracé au cas où il le juge nécessaire. Logiquement, il faut tenir compte de tous les traits que l'utilisateur fait s'il veut revenir en arrière plusieurs fois. La structure de données qui nous sera utile dans ce cas est la pile (*stack*). Cette dernière suit le principe du « dernier entré, premier sorti » (*Last In First Out*). Elle nous permet d'empiler chaque tracé par-dessus un autre et d'enlever le premier qui se trouve sur la pile, étant le dernier à avoir été empilé. Nous allons procéder de la manière suivante : lorsque l'utilisateur va appuyer sur la souris et la glissera pour dessiner, le chemin (*path*) de son tracé sera stocké au moment où il cesse d'appuyer. Quand l'utilisateur va appuyer sur retour, il suffira de retirer le dernier chemin dessiné et de dire à l'interface de refaire le dessin formé de ces chemins.



Au sujet de l'animation, nous voulons faire tourner les vecteurs autour de leur centre de rotation, tous en même temps. Pour que ça soit possible, les coefficients des vecteurs seront calculés à l'aide de l'algorithme de transformation de Fourier rapide (*Fast Fourier Transform*). Nous allons faire usage des listes : « numpy.ndarray » de la librairie NumPy pour conserver l'angle de départ du vecteur, ainsi que sa taille. Cette structure de données est optimale pour les opérations sur des listes, qui dans ce cas, seront des matrices à deux dimensions. Dans cette situation, il faut multiplier chaque objet de la liste, étant des listes contenant les informations mentionnées précédemment, par une matrice de sinus et de cosinus. Cela nous permettra de créer un effet de rotation dans l'interface graphique.



Pour avoir une meilleure séparation de la vue et du modèle, nous allons implémenter le patron de conception: procuration (*Proxy*) pour les vecteurs tournants. Dans notre cas, les informations de chaque vecteur seront contenues dans un objet de la classe *Fvect*. Cet objet est un des objets « proxy » qui va aller porter les données nécessaires pour que la vue puisse faire l'animation. Étant donné que chaque animation peut avoir plusieurs flèches, le même nombre de « proxys » doivent être instanciés. Pour passer au travers de tous les vecteurs, nous allons nous servir de la liste (*List*) de python. Cette structure de données va nous permettre de parcourir et mettre à jour les vecteurs à mesure que leurs angles de rotation changent. De plus, la liste est dynamique, alors sa grandeur change si on met ou on en enlève des éléments. Exemple : Si un utilisateur veut voir à quoi ressemblerait son dessin avec moins de flèches.



Patrons de conception

Notre interface graphique sera organisée selon le patron de conception du composite. Elle sera faite avec PySide, une librairie Python faite pour la création d'interfaces graphiques, et sera faite d'un élément principal qui sera subdivisé en plusieurs widgets, eux-mêmes subdivisés en d'autres widgets. Ces widgets seront des objets créés à partir de classes que nous aurons programmées, et donneront une structure en forme d'arbre. Le fonctionnement de la librairie de PySide se compose déjà avec ce patron de conception. Le composite est très utile, car il donne une meilleure organisation à notre vue et il est facile à comprendre, car tout s'emboîte bien.

Pour que nos widgets puissent envoyer des informations et des demandes d'action aux autres parties de notre programme, nous utiliserons des signaux. Ces signaux seront implémentés selon le patron de conception de chaîne de responsabilité, où chaque parent de l'émetteur de signaux le recevra et décidera de faire quelque chose avec ou de le lancer plus haut dans la hiérarchie. Par exemple, un widget qui reçoit un signal de changement d'interface d'un sous-widget pourra faire l'action de changer de sous-widget, et s'il reçoit un signal de sauvegarde, il l'enverra plus haut vers le contrôleur qui l'enverra ensuite au modèle.

Nous implémenterons le patron de méthode (*Template Method*) pour ce qui est des options de tri qui se trouvent dans la galerie d'image des autres usagers. Ces tris ont des fonctions très similaires, mais avec une variation de comment ils font le tri des images. En les traitant comme des « templates » d'une classe abstraite, nous avons moins de duplication inutile de code, et ce sera plus facile d'ajouter des options de tri supplémentaires si nous le désirons.

Expression régulière

La galerie de dessins de tous les usagers de notre projet va avoir deux barres de recherche, une pour les auteurs et une pour les dessins. Ces barres de recherche vont transformer ce qui sera écrit par l'utilisateur en expression régulière et l'appliquera soit aux identifiants d'utilisateur ou aux noms de dessin, afin de n'afficher que ceux qui contiennent la séquence entrée.

Algorithme et Mathématique

Le concept au cœur de notre projet, les séries complexes de Fourier vont être principalement exécutés par l'algorithme de transformation de Fourier rapide (*Fast Fourier Transform*) que nous allons implémenter. Ce dernier va prendre une série de coordonnées dérivée d'un fichier d'image de type SVG et va traiter ces séries de points comme des ondes. Essentiellement, c'est comme si des ondes avaient été enroulées autour d'un point sur le chemin d'un cercle. Il traitera ensuite ces ondes pour obtenir une série de nombres complexes que nous pourrions utiliser pour déterminer le rayon et l'angle de départ des vecteurs dont nous aurons besoin pour recréer le dessin. Pour bien implémenter le FFT, il nous faudra un niveau raisonnable de compréhension des intervalles, des nombres complexes, de l'utilisation des sinus et cosinus dans des équations, dont la rotation de vecteurs, ainsi que de concepts plus simple comme les exposants.

Veille technologique

Le cours de veille technologique nous permet d'approfondir nos connaissances en mathématiques pour aller de l'avant dans le projet. Étant donné que Joseph Fourier était un grand mathématicien, les notions qu'il utilisait sont encore aujourd'hui considérées comme des mathématiques d'études supérieures. N'ayant pas eu la chance d'explorer les aspects du calcul différentiel et du calcul intégral en informatique, le cours de veille technologique est pour nous du temps précieux pour aller effectuer des recherches à ces sujets. Nous utilisons les séries de Fourier dans l'algorithme qui est au cœur de notre projet et afin de bien l'intégrer, il est indispensable de comprendre en surface les mathématiques qui sont à la base de son raisonnement.