# **Projet tutoré du semestre 2**

Table des matières

[**Projet tutoré du semestre 2** 1](#_Toc516083533)

[Introduction 2](#_Toc516083534)

[Cadre d’utilisation 2](#_Toc516083535)

[Conception générale 2](#_Toc516083536)

[-Diagramme de cas d’utilisation 2](#_Toc516083537)

[-diagramme de classe 3](#_Toc516083538)

## Introduction

Voici le rapport de projet tutoré du second semestre réalisé par Yanis Levesque et Thomas Vathonne, vous trouverez ci-contre tous les éléments concernant la réalisation de ce projet qui consiste à créer une application permettant la visualisation et création de frise chronologique personnalisée par l’utilisateur.

Note : tous les diagrammes présents dans ce rapport sont aussi dans un fichier à part sur github ouvrable avec le logiciel « StarUml » pour plus de clarté et de lisibilité.

## Cadre d’utilisation

L’application doit permettre à l’utilisateur de créer et d’enregistrer des frises chronologiques qu’il réalise lui-même. A son lancement, l’application doit permettre à l’utilisateur de créer une nouvelle frise ou d’ouvrir une déjà faite auparavant. Quelque soit le choix effectué par l’utilisateur, Il doit donc pouvoir facilement ajouter, modifier ou supprimer à sa guise des évènements liés à la frise à tout moment.

Il est donc indispensable que les frises soient donc enregistrables et ouvrables directement par l’application et par ce fait, les frises doivent être disponible sous le format d’un fichier compréhensible par l’application.

Sous les conseils de nos professeurs et en accord avec le sujet, l’applications est séparée en 4 menus internes : « Création », « Affichage », « Aide » et « Quitter ».

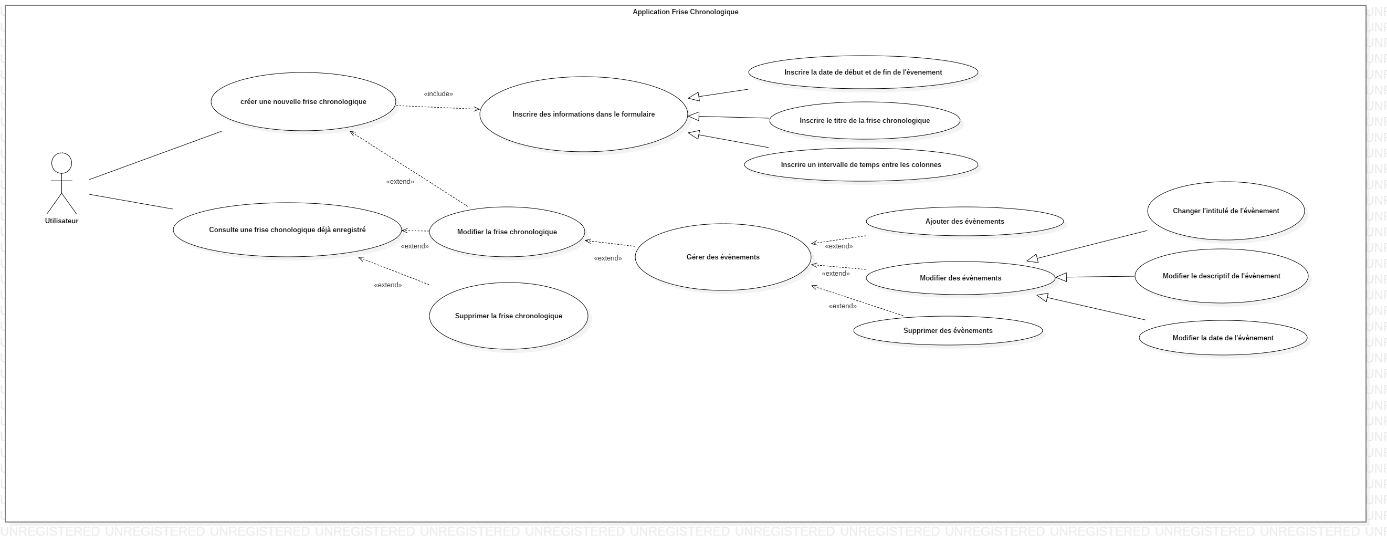
Toutes informations à apporter relatives à la saisie de l’utilisateur sont effectuées a l’aide de formulaires spécialement conçus dans le menu « Création ».

L’affichage du diaporama et de la table d’évènements est situé dans le menu « Affichage ».

Le menu « Aide » indiquera un lien vers le manuel utilisateur et le menu « Quitter » permet juste de quitter l’application.

## Conception générale

### -Diagramme de cas d’utilisation



Voici le diagramme de cas d’utilisation de notre application qui, pour rappel, est aussi donné en annexe pour une meilleure visibilité.

L’utilisateur peut, en premier lieu, créer une frise chronologique ou en modifier une déjà créée.

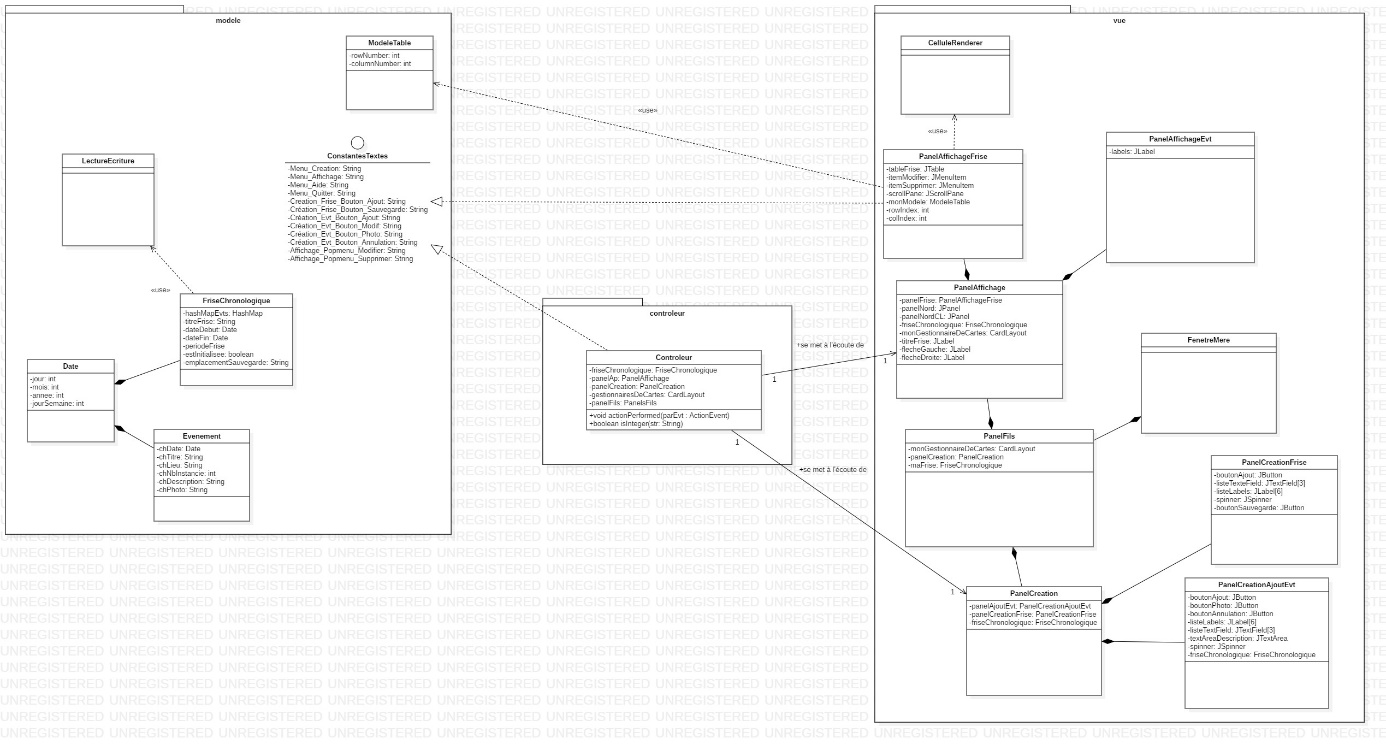
S’il choisit d’en créer une nouvelle, il sera amené à remplir les informations nécessaires à la création de celle-ci dans un formulaire :

* Le titre
* La date de début et de fin
* Un intervalle de temps (une période) pour un meilleur affichage de la frise.

Après cette première étape, la frise est instanciée, l’utilisateur peut alors choisir de l’initialiser et ainsi remplir sa frise d’évènements.

Ces évènements sont définis par l’utilisateur en trois caractéristiques : leur intitulé, leur descriptif et par une date. Qu’il modifie ou crée la frise chronologique, l’utilisateur peut ajouter, modifier, supprimer les évènements et leurs caractéristiques.

### -Diagramme de classe



Le diagramme de classe ci-dessus représente les différentes classes de notre projet sous l’architecture MVC (modèle, vue, contrôleur) vu en cours.

Le paquet « modèle » représente les données essentielles que l’application va utiliser pour remplir son objectif principal. Nous y retrouvons, une classe « Evenement » et une classe « FriseChronologique » qui sont dépendantes toutes les deux d’une classe Date. Ces trois classes représentant les données essentielles afin de pouvoir réaliser une frise chronologique contenant des évènements.

Une classe « LectureEcriture » existe dans l’unique but de pouvoir enregistrer les frises chronologiques constituées par l’utilisateur et de pouvoir les consultées plus tard.

La classe ModeleTable quant à elle pose les bases d’une JTable d’évènement qui sera affiché à l’utilisateur.

Pour finir l’interface « ConstantesTextes » permet de réunir tout les constantes textes qu’on a pu avoir besoin dans l’élaboration de notre programme.

Le paquet « vue » consiste à l’ensemble des classes qui permet l’affichage. Plus simplement, c’est ce que l’utilisateur verra de notre application.

Une classe « FenetreMere » a pour contentPane un « PanelFils » qui contiendra tout nos éléments graphiques :

* Le « PanelCreation » ayant pour layout un borderLayout qui est lui-même composé de « PanelCreationFrise » et « PanelCreationAjoutEvt » situés à gauche et à droite respectivement pour le menu « Creation »,
* Le « PanelAffichage » ayant pour layout un borderLayout qui est lui-même composé de « PanelAffichageFrise » (ayant pour layout un borderLayout, avec la frise placé en son centre) et « PanelAffichageEvt » (ayant pour layout un gridBagLayout, avec le titre, la date, et la description de l’évènement).

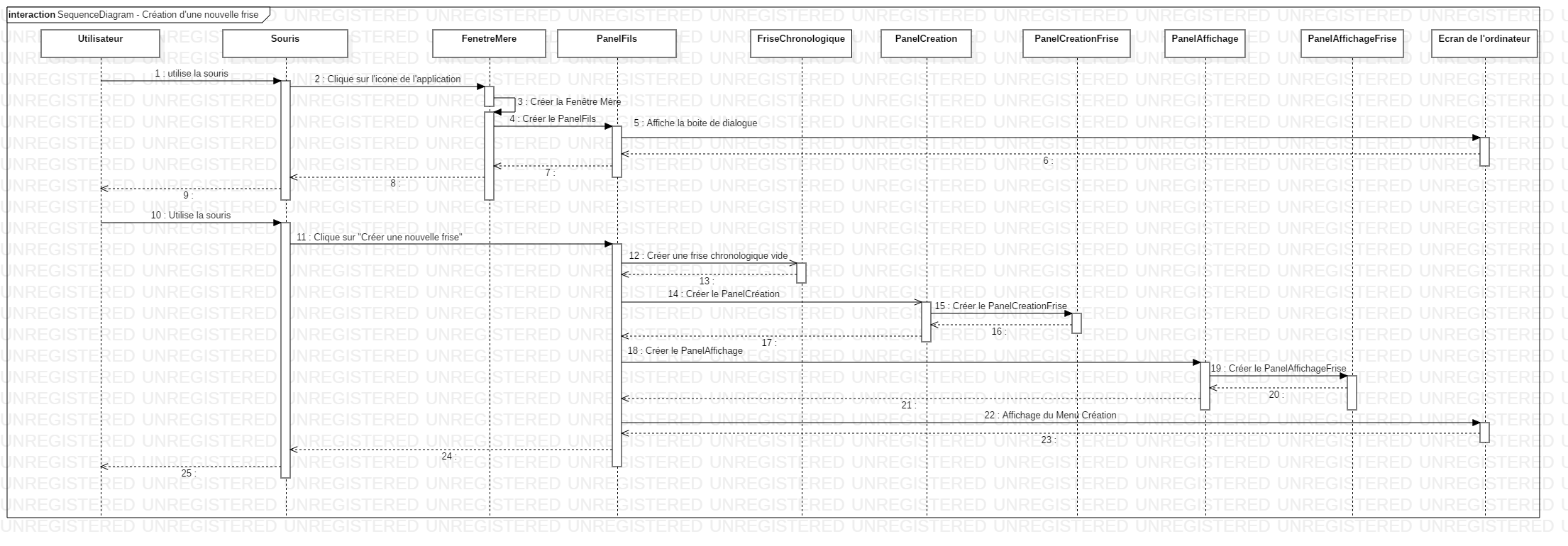
PanelAffichageFrise est placé dans la partie sud du borderLayout de PanelAffichage, tandis que la partie nord est constitué d’un borderLayout qui contient un CardLayout et permet ainsi l’affichage du diaporama.

PanelAffichageFrise utilise les services de « CelluleRender » pour un meilleur affichage de la JTable.

Le paquet controleur ne contient qu’une seule classe nommée « Controleur », qui permet d’effectuer la synchronisation entre les données du modèle et la vue présentée à l’utilisateur. IL se met à l’écoute de « PanelAffichage » et de « PanelCreation » et utilise l’interface « ConstantesTextes »

### -Diagrammes séquentiels

#### Diagramme séquentiel – Création d’une nouvelle frise



Voici le diagramme séquentiel qui montre tout ce que déclenche la création d’une nouvelle frise.

L’utilisateur utilise sa souris pour cliquer sur l’extension .jar de notre application, cette action entrainera en chaîne la création de la « FenetreMere », du « PanelFils » et affichera une boite de dialogue lui demande s’il veut ouvrir une frise chronologique déjà existante ou au contraire en créer une nouvelle.

Si l’utilisateur clique sur « créer une nouvelle frise », la frise sera créer ainsi que le « PanelCreation », le « PanelCreationFrise », le « PanelAffichage », le « PanelAffichageFrise ». Le menu création s’affichera ensuite sous les yeux de l’utilisateur

#### Diagramme séquentiel – Ajout d’un évènement