RAPPORT FINAL

Trello

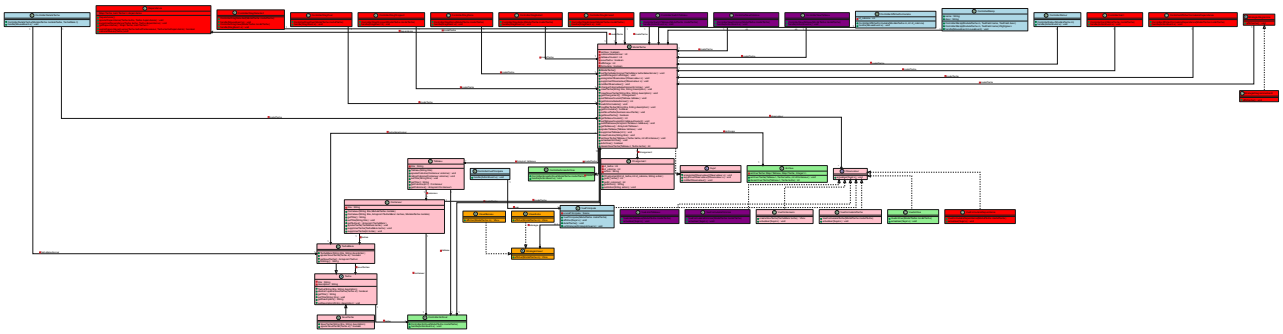
BECHARD Maxime
BENCHERGUI Timothée
KOMODZINSKI Jawad
RYSAK Hugo

TABLE DES MATIERES

LISTE DES FONCTIONNALITÉS

- Créer des tâches
- Afficher des tâches dans les colonnes
- Créer des sous-tâches
- Afficher le détail des tâches
- Modifier les tâches
- Affichage des colonnes sous forme de liste
- Affichage des colonnes sous forme de bureau
- Créer plusieurs tableaux
- Pouvoir changer de tableau
- Ajouter des colonnes
- Pouvoir déplacer des tâches entre les colonnes avec le drag and drop
- Mettre des dépendances aux tâches
- Générer un diagramme de Gantt avec les tâches
- Archivage et désarchivage des tâches

DIAGRAMME DE CLASSE DE L'APPLICATION



En zoomant sur la diagramme on peut tous voir.

Code couleur du diagramme de classe :

Rose : itération 1

Bleu claire : itération 2

Orange : itération 3

Violet : itération 4

Rouge : itération 5

Vert clair : itération 6

REPARTION DU TRAVAIL ENTRE MEMBRE DU GROUPE

ITÉRATION 1 :

Tâche	Membre(s) associé(s)
Diagramme de classe	Timothée BENCHERGUI
Diagramme de séquence affichage	Maxime BECHARD
Diagramme de séquence créer une tâche	Hugo RYSAK
Squelette MVC	Timothée BENCHERGUI
Classe Composite et classe Tache	Jawad KOMODZINSKI
Méthode qui renvoie un tableau	Maxime BECHARD
Afficher la page principale	Timothée BENCHERGUI et Maxime BECHARD
Créer une tâche	Jawad KOMODZINSKI et Hugo RYSAK
Afficher les tâches dans les colonnes	Timothée BENCHERGUI et Maxime BECHARD
Tester les fonctionnalités	Jawad KOMODZINSKI
Rapport de l'itération	Timothée BENCHERGUI

ITÉRATION 2 :

Tâche	Membre(s) associé(s)
Diagramme de classe	Timothée BENCHERGUI
Diagramme de séquence pour modifier une tâche	Hugo RYSAK
Diagramme de séquence pour créer une sous-tâche	Maxime BECHARD
Fonctionnalité Créer sous-tâche	Timothée BENCHERGUI et Jawad KOMODZINSKI
Fonctionnalité modifier une tâche	Hugo RYSAK
Optimisation de l'affichage	Maxime BECHARD
Afficher les sous-tâches	Timothée BENCHERGUI et Jawad KOMODZINSKI
Fonctionnalité afficher une tâche en détail	Timothée BENCHERGUI et Jawad KOMODZINSKI
Tester les fonctionnalités	Hugo RYSAK
Rapport de l'itération	Timothée BENCHERGUI

ITÉRATION 3 :

Tâche	Membre(s) associé(s)
Diagramme de classe	Timothée BENCHERGUI
Diagramme de séquence sur le changement du visuel	Hugo RYSAK
Affichage en liste	Maxime BECHARD
Patron stratégie de l'affichage	Jawad KOMODZINSKI
Contrôleur pour changer l'affichage	Jawad KOMODZINSKI
Visuel / CSS de l'application	Maxime BECHARD et Hugo RYSAK
Rapport de l'itération	Timothée BENCHERGUI

ITÉRATION 4 :

Tâche	Membre(s) associé(s)
Diagramme de classe	Timothée BENCHERGUI
Diagramme de séquence pour créer une colonne	Hugo RYSAK
Diagramme de séquence pour créer un tableau	Maxime BECHARD
Pouvoir créer des tableaux	Timothée BENCHERGUI et Maxime BECHARD
Pouvoir créer des colonnes	Jawad KOMODZINSKI
Afficher les colonnes	Jawad KOMODZINSKI
Pouvoir naviguer entre les tableaux	Maxime BECHARD et Timothée BENCHERGUI
Teste de la fonctionnalité d'ajout des colonnes	Jawad KOMODZINSKI
Teste de la création d'un tableau	Timothée BENCHERGUI
Rapport de l'itération	Timothée BENCHERGUI

ITÉRATION 5 :

Tâche	Membre associé(s)
Diagramme de classe	Timothée BENCHERGUI
Diagramme de séquence fonctionnement du drag and drop	Maxime BECHARD
Diagramme de séquence pour la dépendance des tâches	Hugo RYSAK
Diagramme de séquence pour la génération de Gantt	Hugo RYSAK

Ajouter des dépendances aux tâches	Hugo RYSAK et Jawad KOMODZINSKI
Gérer les dépendances des tâches	Hugo RYSAK et Jawad KOMODZINSKI
Fonctionnalité du drag and drop	Maxime BECHARD et Timothée BENCHERGUI
Création du formulaire pour les dépendances	Hugo RYSAK
Visuel Gantt	Hugo RYSAK et Jawad KOMODZINSKI
Rapport de l'itération	Timothée BENCHERGUI

ITÉRATION 6 :

Tâche	**Membre(s) associé(s)**
Diagramme de classe	Timothée BENCHERGUI
Diagramme de séquence pour archiver une tâche	
Archiver une tâche	Jawad KOMODZINSKI
Visuel des tâches archiver	Jawad KOMODZINSKI
Teste des fonctionnalités de l'itération 5	Jawad KOMODZINSKI et Hugo RYSAK
Teste de l'archivage des tâches	Jawad KOMODZINSKI
Rapport de l'itération	Timothée BENCHERGUI

PRESENTATION D'UN ELEMENT DONT ON EST FIERS

MAXIME BÉCHARD:

Durant la conception de la partie visuelle de l'application, j'ai été amené à concevoir une méthode permettant d'optimiser les opérations nécessaire à l'affichage lors de petites opérations sur la fenêtre. Les informations relatives à l'opération effectuée tel que l'ajout de tache, de colonne, ou l'archivage de tâche, est stockée dans le modèle, pour informer les vues des changement à apporter aux visuel, sans pour autant recréer l'entièreté du visuel de toute la page. Par exemple, la création d'une tache entraînera pour le visuel l'ajout simple de la box représentant la tache ajoutée, dans la colonne correspondante.

TIMOTHÉE BENCHERGUI:

Élément dont je suis le plus fier est le drag and drop. C'est une fonctionnalité qui a été très compliqué à réaliser car il fallait regarder différents tutos sur YouTube ou même regarder la documentation pour voir comment marcher le fonctionnement du drag and drop. Pour intégrer le système au projet nous avons dû créer 6 différentes classe contrôleur pour gérer le drag detected, done, dropped, Entered, Existed et Over. Les six différentes classe détectent quand il y a une action sur la tâche et montre ou es qu'on peut la déposer. Je sais que ce n'est pas très original mais j'ai préféré parler de cela car c'est la fonctionnalité qui m'a donné le plus de fils à retordre.

JAWAD KOMODZINSKI:

Gérer les dépendances est la chose dont je suis le plus fière, elle est indispensable à la génération du diagramme de Gantt. J'ai utilisé une Map pour gérer efficacement les dépendances. De ce fait, chaque tache a une liste de dépendance. Par exemple, A -> {B, C, D} signifie qu'il faut faire les taches B, C et D avant de pouvoir faire la tache A. Cela en tête, j'ai dû gérer l'ajout des dépendances et le calcul de leur niveau. Ce fut un défi, il fallait empêcher les erreurs comme les doubles dépendances (A->B et B->A) ou les cycles (A->B, B->C, C->A). Ce fut intéressant de coder la méthode pour empêcher les cycles qui est une méthode récursive qui utilise une approche de recherche en profondeur. Il fallait aussi mettre les niveaux à jour dynamiquement dans le bon ordre, donc on met à jour en commençant par le niveau 0 et on augmente. Dû à ses défis, cette fonctionnalité est celle qui m'a le plus marqué.

HUGO RYSAK:

Une fonctionnalité dont je suis fière est l'affichage en liste. Premièrement parce que le design que j'ai créé me semble plutôt intuitif et ergonomique et assez agréable à regarder et à interagir avec (cliquer sur les listes pour que celles-ci révèlent toutes les taches qu'elles contiennent). Deuxièmement parce que j'ai dû faire de nombreuse recherche pour modifier le visuel avec css et trouver une apparence qui me convenait vraiment. Par la suite, j'ai dû utiliser plusieurs parties de code que je n'avais pas coder moi-même (l'affichage des tâches par exemple), cela a était un vrai défi de comprendre et d'utiliser le code fait par quelqu'un d'autre mais j'ai réussi à comprendre et réutiliser et modifier le code mis à ma disposition. Enfin, mon code fait preuve d'originalité avec l'utilisation de classe anonyme pour afficher ou non les listes en mode étendu.

COMPARAISON ENTRE LE PROJET ET L'ETUDE PREALABLE

Étude préalable	Projet final
- *Créer des tâches* - *Créer des sous-tâches* - *Créer des tâches dépendantes* - *Archiver des tâches* - *Visualisation bureau* - *Visualisation listes et sous listes* - *Générer un diagramme Gantt*	*- Créer des Tâches* *- Créer des Sous Tâches* *- Créer des Tâches dépendantes* *- Modifier des Tâches* *- Créer des Colonnes* *- Créer des Tableaux* *- Archiver des Tâches* *- Désarchiver des Tâches* *- Visualisation Bureau* *- Visualisation Listes et Sous Listes* *- Visualiser Différents Tableau* *-Générer Diagramme Gant*

PATRONS DE CONCEPTION ET ARCHITECTURE UTILISES

ARCHITECTURE :

Modèle Vue Contrôleur :
L'architecture MVC est utilisée dans tous le projet, c'est en quelque sorte le squelette du projet, les données peuvent agir sur les vues (l'affichage) et les contrôleurs agissent sur les données. Grâce à cela l'utilisateur interagi seulement avec les contrôleurs.

PATRONS DE CONCEPTION :

Observateur :
Ce patron de conception est étroitement lié à l'architecture MVC car on utilise un modèle qui implémente une interface Sujet qui elle contient une liste d'observateurs qui sont en fait les différentes vues de l'application.

Composite :
Le patron composite quant à lui est utilisé pour les tâches et les sous-tâches. Nous avons une classe abstraite « Tâche » qui représente une Tâche puis nous avons une classe « Tâche mère » qui hérite de la classe « Tâche » et qui contient aussi une liste de Tâche qui sont enfaite ces sous-tâches. Pour finir il y a aussi une classe « Sous-Tâche » qui hérite de la classe tâche.

Stratégie :

Le patron Stratégie a été utilisé à plusieurs endroits. Le premier est pour le choix du visuel de l'application, car elle peut s'afficher sous forme de liste ou de bureau, il y a donc une classe abstraite « Stratégie Visuel » avec deux autres

classes « Visuel Bureau » et Visuel Liste » qui hérite toutes les deux de le première classe.

Le deuxième endroit est pour la génération du diagramme de Gantt avec une classe abstraite « Stratégie Diagramme » et une autre classe « Stratégie Diagramme Gantt » qui hérite de la première classe et grâce à ce système il est possible de faire des générateurs de d’autre diagrammes.