Projet Synthèse

PlanMe

# Conception

Par :

Jessika Longtin et Finnegan Simpson

Pour :

Jean Christophe Demers

# Maquettes - interfaces :Diagram Description automatically generated with low confidence

# Conception UML :

# Schéma(s) de la structure de données externe – détaillé(s)

# Éléments de conception :

## Structures de données :

## Patrons de conception :

« Strategy » :

« Strategy » est un modèle de conception qui permet de créer une famille d’algorithmes sépares en classes et de rendre leurs objets interchangeables.

Dans le contexte de notre application, ce modèle de conception de stratégies nous sera utile pour permettre aux utilisateurs de visualiser, par exemple, leur emploi du temps en mode calendrier ou en mode liste en créant différentes classes implémentant ces différentes interfaces et tout cela peut être sélectionner par l’utilisateur avec des options rendant l’application dynamique dépendamment du choix de l’utilisateur.

Ainsi, ce patron de conception permet une flexibilité et l’expansion future de nouveaux modules optionnels pouvant être ajoutées sans avoir à reformater le code existant.

Factory :

Composite :

## Expression régulière :

## Algorithme :

Pour l’implémentation d’algorithmes dans notre « planner », nous pensons mettre en place un processus de tri, qui peut, par exemple, être utilisé pour trier des taches en ordre d’importance d’après l’utilisateur, de dates de création ou d’échéances, en implémentant des méthodes de tri approprié. Celles-ci seront incorporer dans un menu d’options ou des « check boxes » dans la « Nav bar » du module.

Additionnellement, il serait intéressant d’ajouter un system de recherche qui a pour effet une mise à jour automatiquement, dépendant des lettres inscrites dans la barre de recherche du module, conséquent l’affiche de pages, modules ou notes spécifique à la recherche de l’utilisateur.

## Mathématique :

Veille technologies:

### React:

Dans le cadre du cours de veille technologie, nous avons choisis l’exploration de la librairie javascript react.js.

Étant utilisée majoritairement pour la construction d’interface utilisateur, React est une librairie superbe pour les besoins de notre application. Celle-ci, étant un outil de planification permettant d’organiser facilement son emploi du temps, de prendre des notes et de faire le suivi de tâches à accomplir, utilisable par plusieurs types de personnes, comme les étudiants pour gérer leurs travaux, la planification de devoirs, examens ou la gestion de budgets etc. De-même que toutes autres personnes qui souhaite organiser différents aspects de sa vie au même endroit etc. Tout cela demande l’implémentation d’interface d’enregistrement de compte utilisateur, des tables et modules graphique interactives et modifiable par l’utilisateur, de la recherche de données de l’utilisateur, l’éventuelle implémentation de partage et gestion de contacts etc.

De-plus, react nous permet de créer des composants réutilisables à l’intérieur de notre application, ce qui permet non seulement de gagner du temps, mais aussi de rendre le code maintenable.

Sur le point de performance, l’utilisation d’un DOM « Document Object Model » virtuel par react, permet la mise à jour efficaces de l’interface utilisateur sans avoir besoin de rafraichir la page en entier, ce qui rend le chargement de pages plus fluide qui delà porte une meilleure expérience de notre application pour les utilisateurs.

# References

*Design patterns strategy*. (2023, 02 26). Retrieved from refactoring guru: https://www.uxpin.com/studio/blog/react-design-patterns/

https://www.uxpin.com/studio/blog/react-design-patterns/ (2/16/2023)

<https://refactoring.guru/design-patterns/strategy> (2/16/2023)

<https://refactoring.guru/design-patterns/factory-method> (2/16/2023)

<https://refactoring.guru/design-patterns/proxy> (2/16/2023)