

Projet Synthèse

PlanMe

Conception

Par :

Jessika Longtin

Finnegan Simpson

Pour :

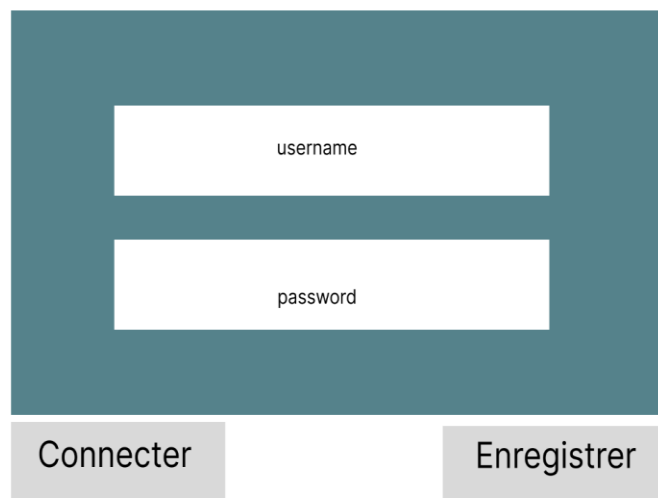
Jean Christophe Demers

Maquettes - interfaces :

Main Color Pallet



Page login:

A login form mockup. It consists of a dark teal rectangular container. Inside this container are two white rectangular input fields. The top field is labeled "username" and the bottom field is labeled "password". Below the teal container are two light gray rectangular buttons. The left button is labeled "Connecter" and the right button is labeled "Enregistrer".


Page Enregistrement:

interessant d'ajout d'un
fond relier au theme de
planifier avec des
"stickers" differents dans
un pattern

Enregistrement

Icon du site
web
a
determiner

Page principale :



Jane Pate's
PlanMe

contact en ligne : 13

▲

Page 1

▶

Page 2

▲

Notes

...

liste a
cocher

évenem
ent

Calandar

liens
page

+

Evenements

Nom	Date	Type	notes

★


+

bloque de texte libre

lorem a lorem qui lorem peut lorem juste un peu.

★

Page principale



Nom's
Prenom

+::

+::

▽ Page 1

▽ Page 2

▷ Page courante

▽ Sous-page

Module texte

+::

12

Comic Sans

Lorem ipsum
- - -
- - -
- - -

Modification module

Titre

nom

Taille :

Petit

Moyen

Grand

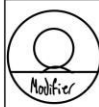
Vue :

tableau

Calendrier

Supprimer le module

Profil



Modifier

Nom :

Prenom :

Courriel :

Date de Naissance: 00/00/00

Theme :

☒


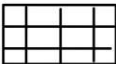
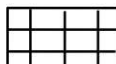



Contacts :

- Jane - w

- w - w

- w - w




Ajout module

	Lise à cacher
	événements
	Calendrier
	lien vers page
	Image
	Budget

Trier dans modules tableaux

trier par	Nom colonne	▼
ordre		▼
+ -		

Filtre modules tableaux


colonne 1	
colonne 2	
colonne 3	

Module image

+ :



Module lien sous page

+::  Sous Page

Module budget

+::

Nom	Date	Revenu	Depense
	10 février 2023	10,00	
	6 avril 2023		20,00
Somme		10,00	20,00 Total: 10,00

Module evenements

+::

Nom	Date	Type	Notes
Sortie Hogworts Legacy	10 février 2023	Jeu	89\$
Fête John	6 avril 2023	Anniversaire	acheter cadeau
~	~	~	~

Conception UML :

Digramme de cas d'usage

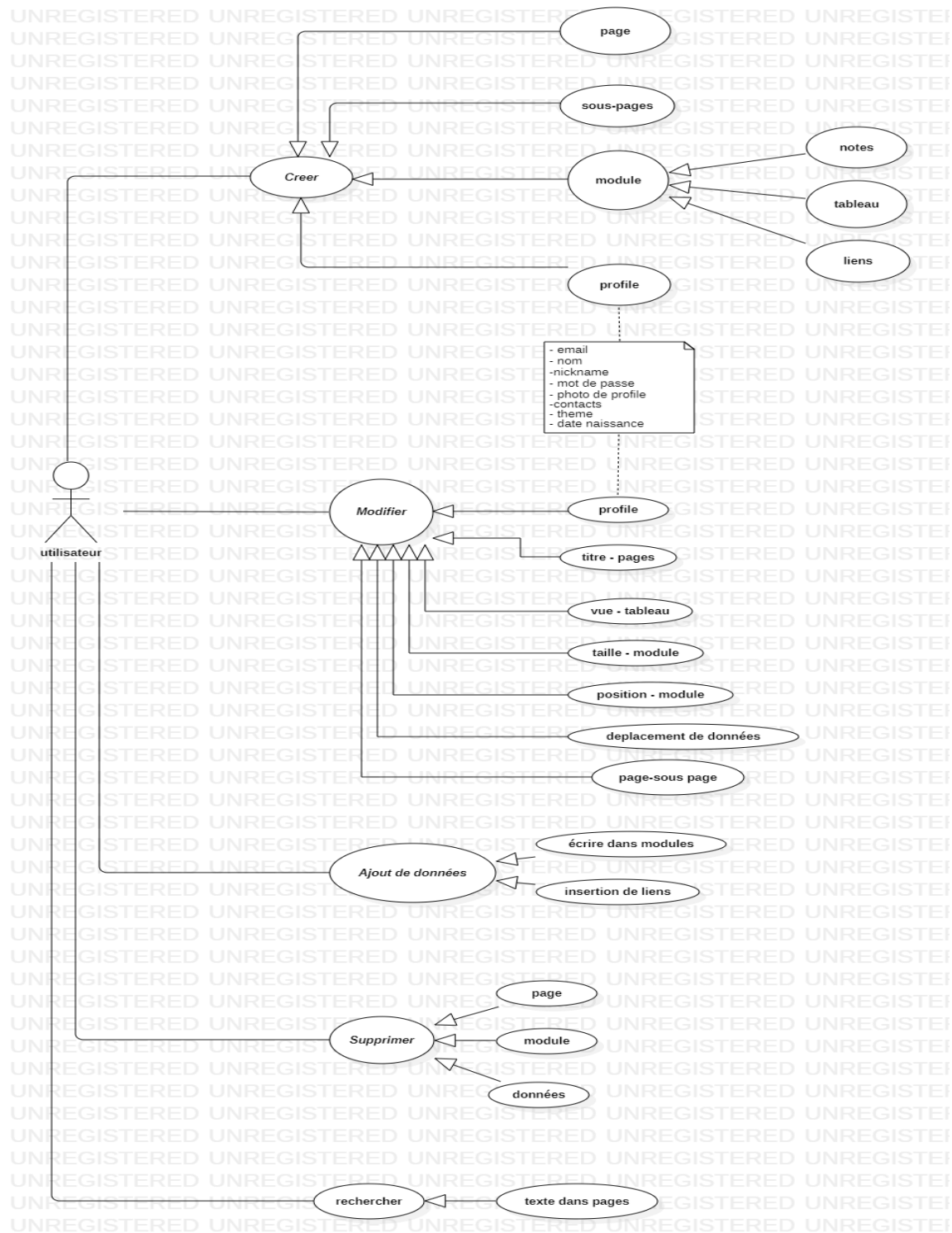
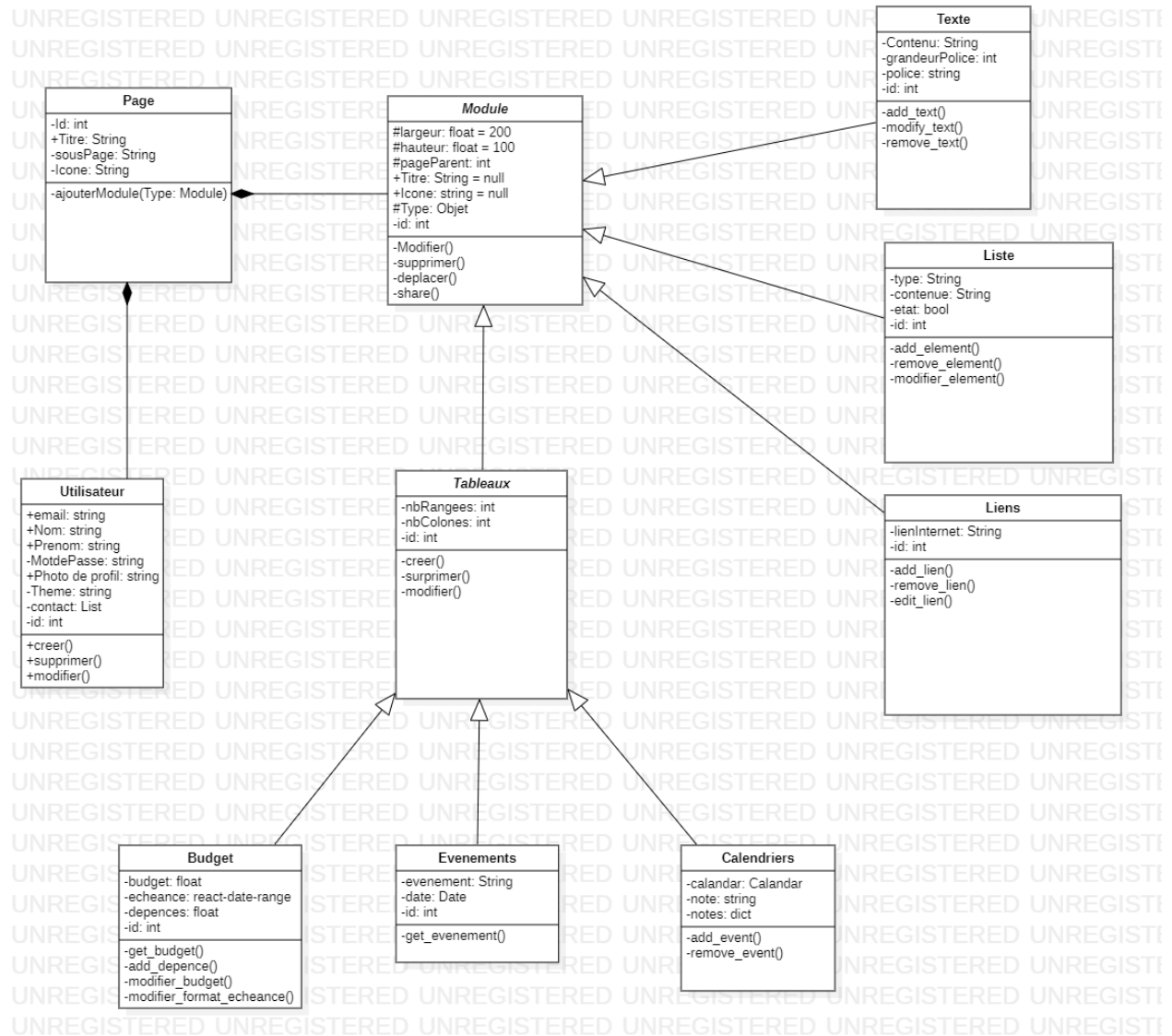
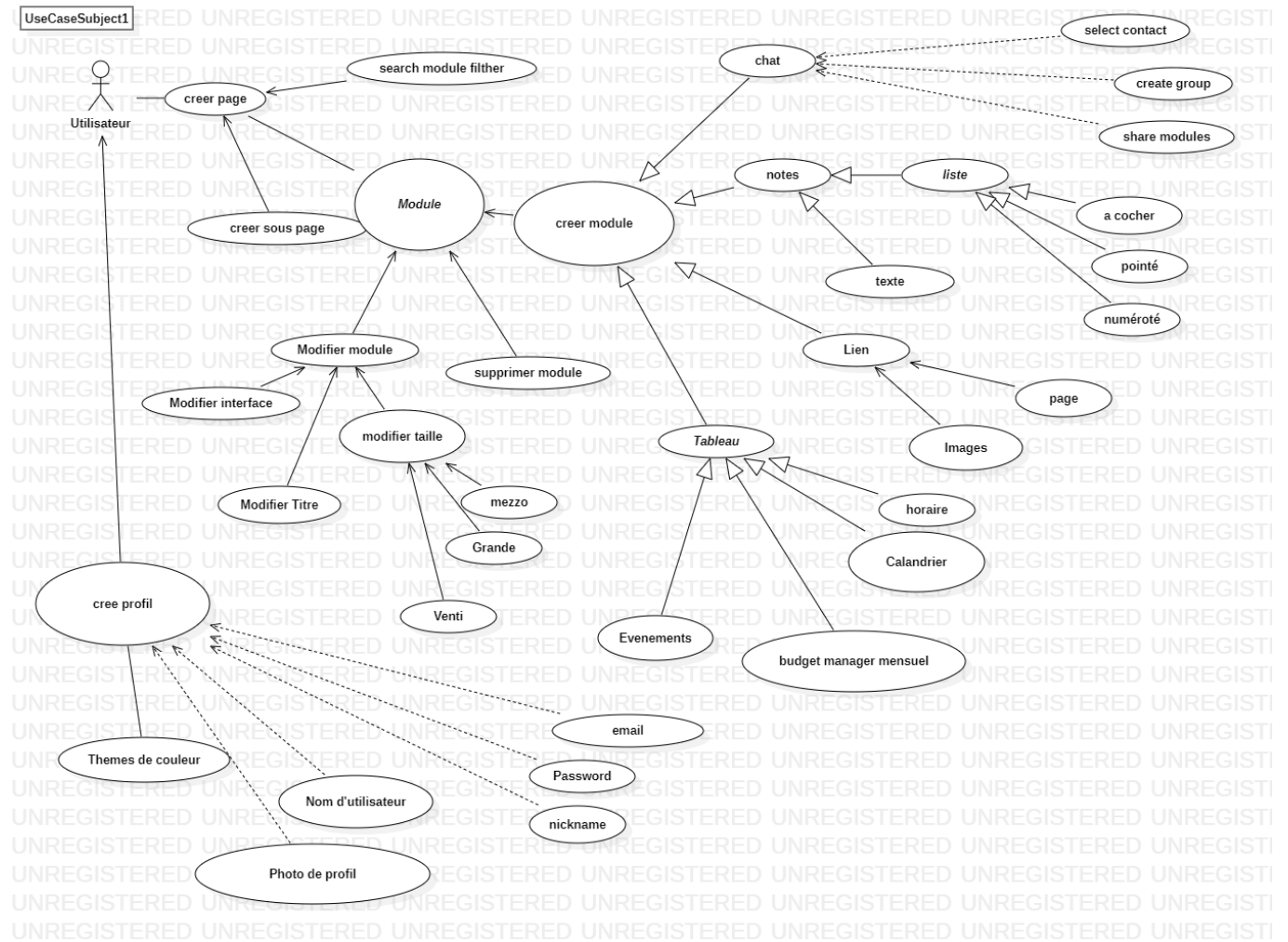


Diagramme de classe:



Flowchart:

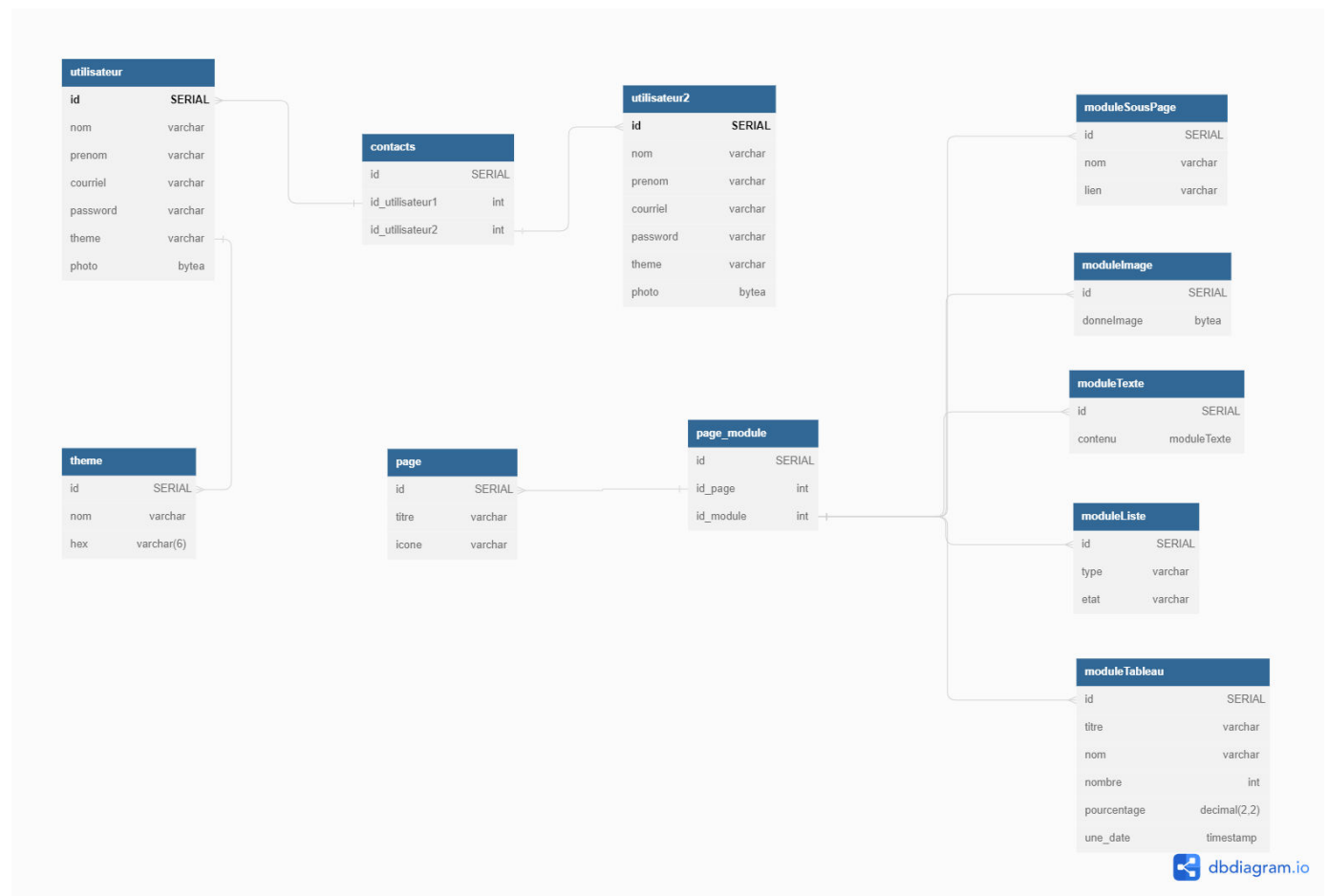


Schéma(s) de la structure de données externe – détaillé(s)

La base de données utilisée est **PostgreSQL**. Nous allons utiliser cette base de données notamment pour représenter notre module de calendrier et événements qui sont littéralement sous forme de table.

C'est donc le modèle de données le plus proche de la réalité. De plus, PostgreSQL est parfait pour gérer des utilisateurs et peut même stocker des fichiers d'image bytes et répond donc très bien à tous nos besoins de stockage externe autant plus que c'est le type de base de données avec laquelle nous avons le plus d'aisance jusqu'à présent.

Schéma des tables:



Éléments de conception :

Structures de données :

Pour la structure de donnée à construire nous-même, nous allons faire une structure de graph pour la gestion des relations entre les utilisateurs. Le graph contiendra des nœuds utilisateurs avec des relations contacts avec les autres utilisateurs. Nous allons chercher tous les utilisateurs avec leur liste de contacts respective de la base de données pour commencer. Ensuite, ont créé un nœud pour chaque utilisateur avec les attributs suivants: nom, date de naissance et modules les plus utilisés. Pour chaque nœud, la relation "est en contact avec" est créée avec chacun de ses contacts. Cette implémentation servira à notre algorithme de suggestion de contact car on peut utiliser le graph pour déterminer la proximité, notamment avec des méthodes comme des matrice de contiguïté , entre les utilisateurs

Trois utilisées:

Calendrier/événements: Nous allons sauvegarder les calendriers et événements sous forme d'objets contenant la date, description participant et tous autres attributs que l'utilisateur ajoute ou retire. Cette structure est la même pour le calendrier et les événements puisque ce sont les mêmes données seulement affichées différemment. Le fait que ce soit un objet nous permettra de le modifier aisément avec des méthode de celles-ci

Json: Le json servira à stocker les différents types de modules. Chaque module contiendra un id, son type de module et une liste de contenu: string pour le texte, liste de strings pour les listes et objets pour les calendriers et événements.

Exemples de json:

```
"Liste" : {
  "id": 0,
  "type": "liste to do",
  "contenu": {
    "ligne 1" : "bla",
    "ligne 2": "bla bla",
    etc...
  }
}

"Bloc de texte": {
  "id": 0,
  "type": "texte",
  "contenu": {
    "police" : "comic sans",
    "taille de texte": 12,
    "texte": "lorem ..."
  }
}
```

Arbre n-ary: L'arbre binaire représente la hiérarchie de pages et ses modules. Le nœud à la racine est l'utilisateur et le premier étage sont les pages parent. Sous celles-ci sont les modules et les sous-pages. Nous avons choisi l'arbre binaire pour cette application puisqu'on a besoin d'avoir accès aux sous modules profond fréquemment pour modifier, créer et supprimer. Un arbre n-ary est plus approprié qu'un arbre binaire pour ce contexte puisque les pages peuvent avoir n-modules et n-sous-page, donc un nombre d'enfant différents.

Patrons de conception :

« Strategy » :

« Strategy » est un modèle de conception qui permet de créer des groupes d'algorithmes séparés en classes et de rendre leurs objets commutables.

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 sélectionné 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 ajoutés sans avoir à reformater le code existant.

Factory méthode:

'Factory' est un modèle de programmation technique qui est sous le paravent de "pattern créationnel". Il donne la possibilité de créer des objets dans une superclasse et permet aux sous classes de celle-ci d'altérer le type des objets créés et d'instancier les classes au besoin du fonctionnement de notre application.

Dans le cadre de notre projet web planner, nous allons utiliser le pattern « factory » dans le but de créer une class mère de module de même que de créer différentes classes enfants. Par exemple : la classe tableau qui sera la classe mère de plusieurs classes, comme : la classe événement, qui héritera des caractéristiques de la classe tableau et en plus d'autres éléments spécifiques à la classe événement comme une date, un nom etc...

Il serait intéressant d'utiliser ce pattern en créant différents types d'événements en plus, dans le cadre de partages de module ajouter différents utilisateurs avec différentes permissions comme lecture écriture etc...

Composite :

'Composite' est un modèle de conception qui permet de regrouper plusieurs objets et d'apporter des modifications à chacun d'eux d'un seul coup. Ce pattern est très utile car il permet la hiérarchisation de notre structure d'objets, permettant à l'utilisateur d'interagir avec ces objets individuellement et en groupe.

Pour notre application PlanMe, il serait intéressant d'utiliser le pattern composite dans le but d'affecter les enfants de pages et modules en les regroupant, permettant de les modifier tous ensemble au besoin. Par exemple pour l'ajout ou la suppression de données.

Bref, en utilisant ce pattern, cela nous permet de créer une application flexible et modulable en apportant aux utilisateurs un niveau d'organisation face aux objets et leur permet d'interagir avec ces objets individuels ou en groupe.

Expression régulière :

L'expression régulière que nous avons choisi d'identifier va être pour la recherche dans les documents. Il y a une icône de loupe à côté de l'icône de profil qui permet de taper un mot à rechercher et qui a chaque lettre tapée va rechercher cette sous-chaîne dans toutes les pages et afficher les chaînes la contenant avec ou elle se situe dans quelle page et module. D'autres expressions régulières que nous allons utiliser vont être avec la création de mots de passe pour la longueur du mot de passe et si elle contient par exemple une majuscule, un nombre et un symbole.

Exemple de regex pour la recherche de mots dans les pages :

```
- '\\b${(mot recherche)}\\w*' , 'gi'
```

g : permet la recherche globale sans arrêter à la première correspondance du String.

i : Permet la recherche d'être case insensitive

Exemple de regex pour les mots de passes :

```
- '^(?=.*?[A-Za-z])(?=.*?[0-9]).{6,}$'
```

Exemple de regex pour Vérification de courriels :

```
- '^[a-zA-Z0-9._:$!%-]+@[a-zA-Z0-9.-]+.[a-zA-Z]$'
```

Algorithme :

Pour l'implémentation d'algorithmes dans notre « planner », nous allons intégrer un algorithme de suggestion de contacts. Celle-ci se base sur le degré de connaissance entre les utilisateurs. Exemple l'utilisateur 1 a un ami d'un ami, puisqu'il y a deux bonds entre l'utilisateur 1 et celui cible c'est de degrés 2. C'est le nombre d'amis entre. Nous allons prendre les exemple 10 personnes les plus proches (le plus petit degrés) pour la première sélection. Ensuite, celles-ci seront jugées selon la différence d'âge et les modules plus utilisés avec l'utilisateur. Finalement, nous prenons les trois meilleurs pour les suggérer.

Mathématique :

Au niveau de fonction(s) mathématique, nous allons utiliser de la projection linéaire pour projeter et prédire les tendances de l'utilisateur en matière de gestion monétaire dans le module de budget après deux mois de données. En appliquant cette fonction sur les données de l'utilisateur en fonction de son historique de dépenses, nous pourrions projeter ses tendances de dépenses futures en fonction de son budget présent.

exemple de formule pour une projection linéaire sur un budget:

Projection de dépenses = $a + bx$

Où:

Dépenses prévues représentent le montant que l'on prévoit que l'étudiant dépensera

a : l'ordonnée à l'origine, c'est-à-dire la valeur de y lorsque x est égal à zéro

b : la pente de la ligne de régression, c'est-à-dire le changement de y pour chaque unité de changement de x

x : le budget présent

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.

Bibliography

Correia, E. (2023, 02 29). *medium*. Retrieved from graph data structure implementation in javascript: <https://medium.com/before-semicolon/graph-data-structure-implementation-in-javascript-668f291a8a16>

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

factory method. (2023, 02 16). Retrieved from refactoring guru: <https://refactoring.guru/design-patterns/factory-method>

FireShip. (2023, 02 16). *Node.js ultimate beginner'S guide in 7 easy steps* . Retrieved from youtube: https://www.youtube.com/watch?v=ENrzd9HAZK4&ab_channel=FireShip

FireShip. (2023, 02 16). *Which js Framework is best?* Retrieved from Youtube: https://www.youtube.com/watch?t=262&v=cuHDQhDhvPE&feature=youtu.be&ab_channel=FireShip

ForrestKnight. (2023, 02 20). *how to build a minimal website with react + tailwind + vite*. Retrieved from youtube: https://www.youtube.com/watch?v=b0pkpcD8Ms4&ab_channel=ForrestKnight

Generic Trees(N-ary Trees). (2023, 02 29). Retrieved from geeksforgeeks: <https://www.geeksforgeeks.org/generic-treesn-array-trees/>

HEUSSLER, J. (2023, 02 22). *Design Patterns : à quoi ça sert et comment les utiliser*. Retrieved from ADIMEO DIGITAL TRANSFORMER: <https://www.adimeo.com/blog-technique/design-patterns-a-quoi-ca-sert-etcomment->

Liu, A. (2023, 02 16). *Creating the Notion API*. Retrieved from Notion: <https://www.notion.so/blog/creating-the-notion-api>

Microsoft. (2023). *OpenAI*. Retrieved from ChatGPT: <https://chat.openai.com/>

Mojeed, I. (2023, 02 22). *what is the virtual DOM in React?* Retrieved from LogRocket frontend Analytics: <https://blog.logrocket.com/virtual-domreact/#>

Mosh, p. w. (2023, 02 17). *React JS - React Tutorial for Beginners*. Retrieved from youtube:
https://www.youtube.com/watch?v=Ke90Tje7VS0&ab_channel=ProgrammingwithMosh

proxy. (2023, 02 16). Retrieved from refactoring guru:
<https://refactoring.guru/design-patterns/proxy>

strategy. (2023, 02 16). Retrieved from refactoring guru:
<https://refactoring.guru/design-patterns/strategy>

Teton-Landis, J. (2023, 02 16). *The data model behind Notion's flexibility*. Retrieved from Notion: <https://www.notion.so/blog/data-model-behind-notion>

Wathan, A. (2023, 02 16). *Rapidly build modern websites without ever leaving your HTML*. Retrieved from tailwindcss:
<https://tailwindcss.com/>

You, E. (2023, 02 20). *Learn Vite with Evan You*. Retrieved from Youtube:
https://www.youtube.com/watch?v=DkGV5F4XnfQ&ab_channel=VueMastery