Projet Synthèse

PlanMe

# Conception

Par :

Jessika Longtin

Finnegan Simpson

Pour :

Jean Christophe Demers

# Maquettes - interfaces :

# Conception UML :

Digramme de cas d’usage

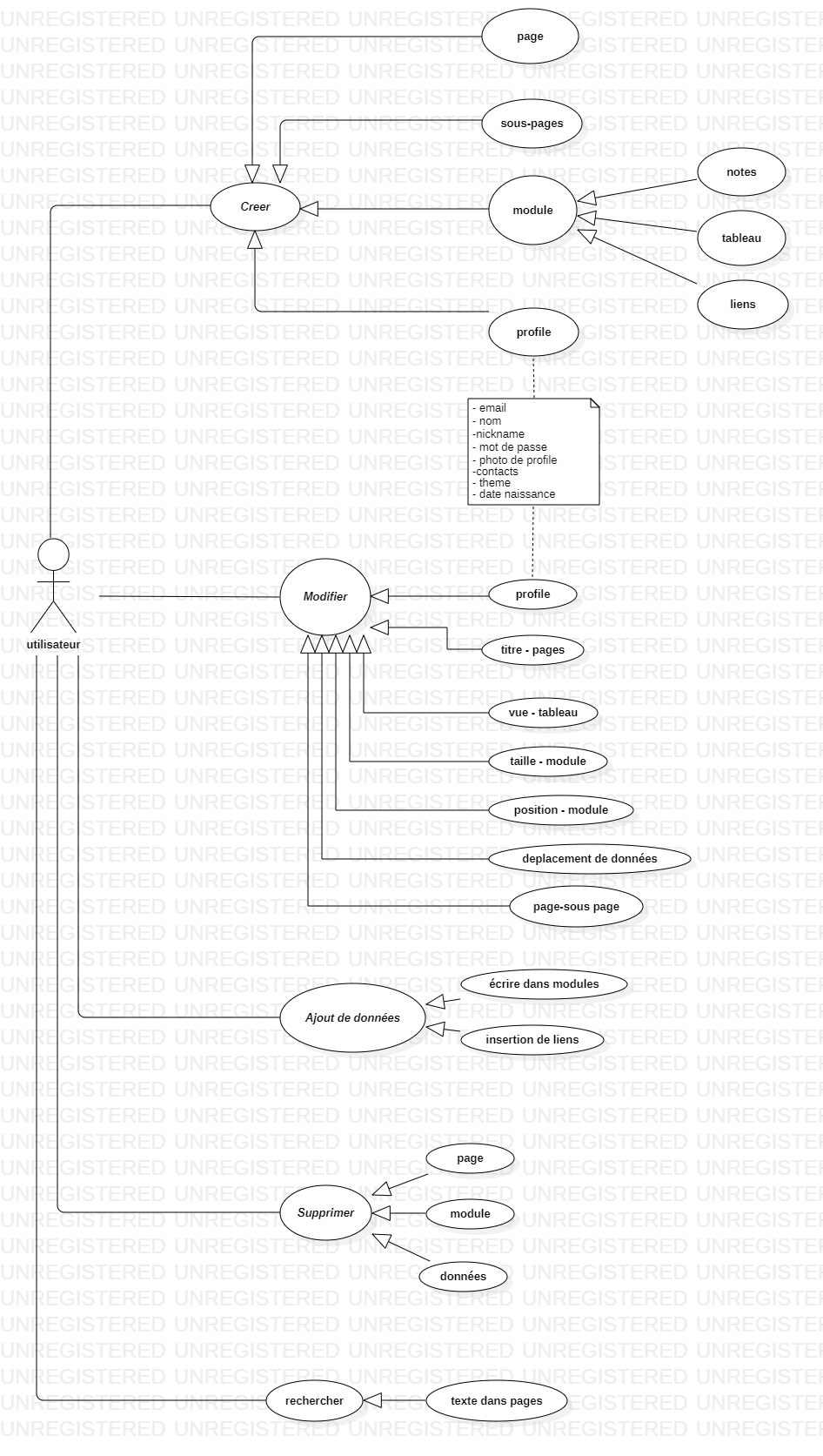
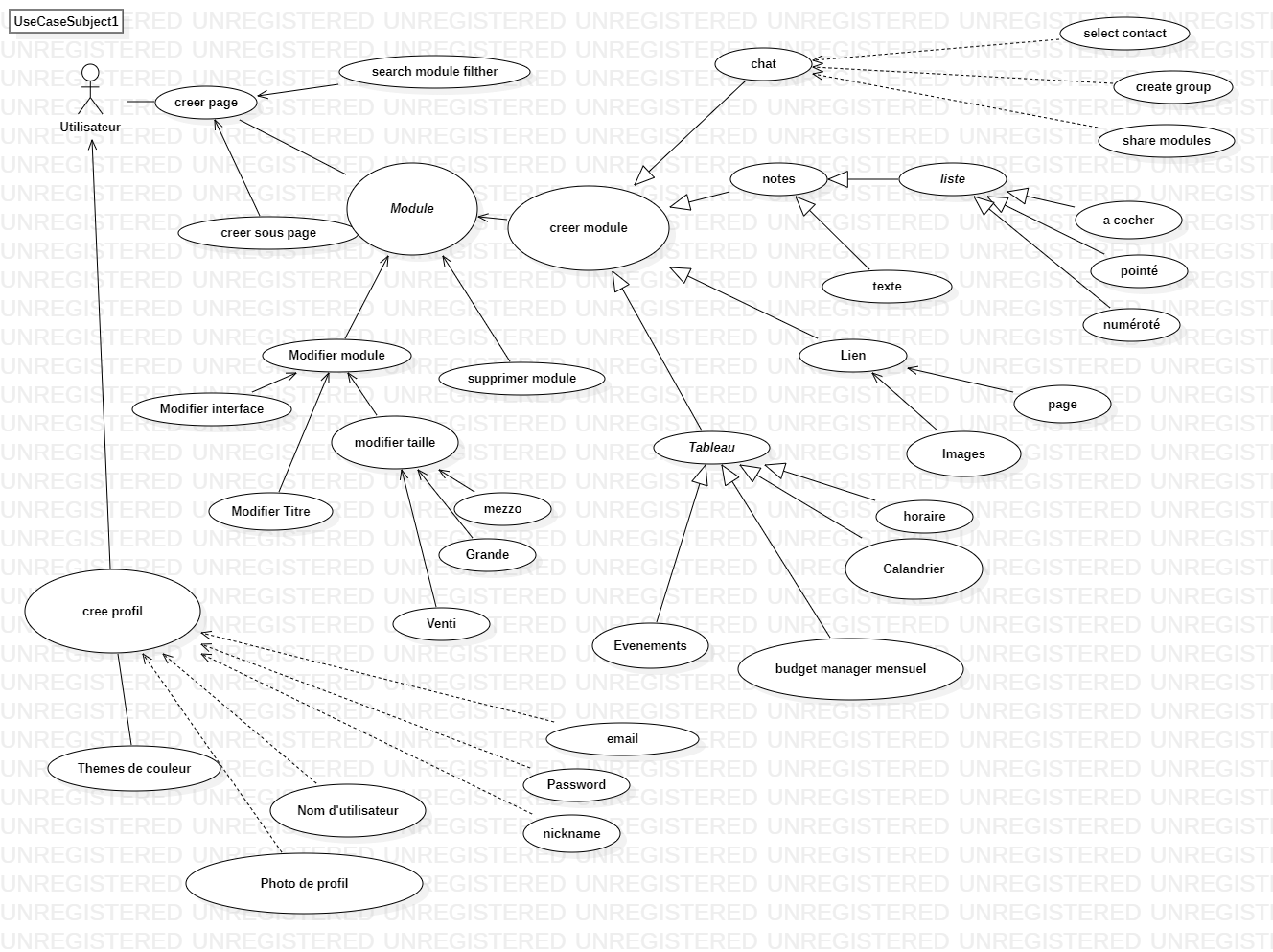


Diagramme de classe:

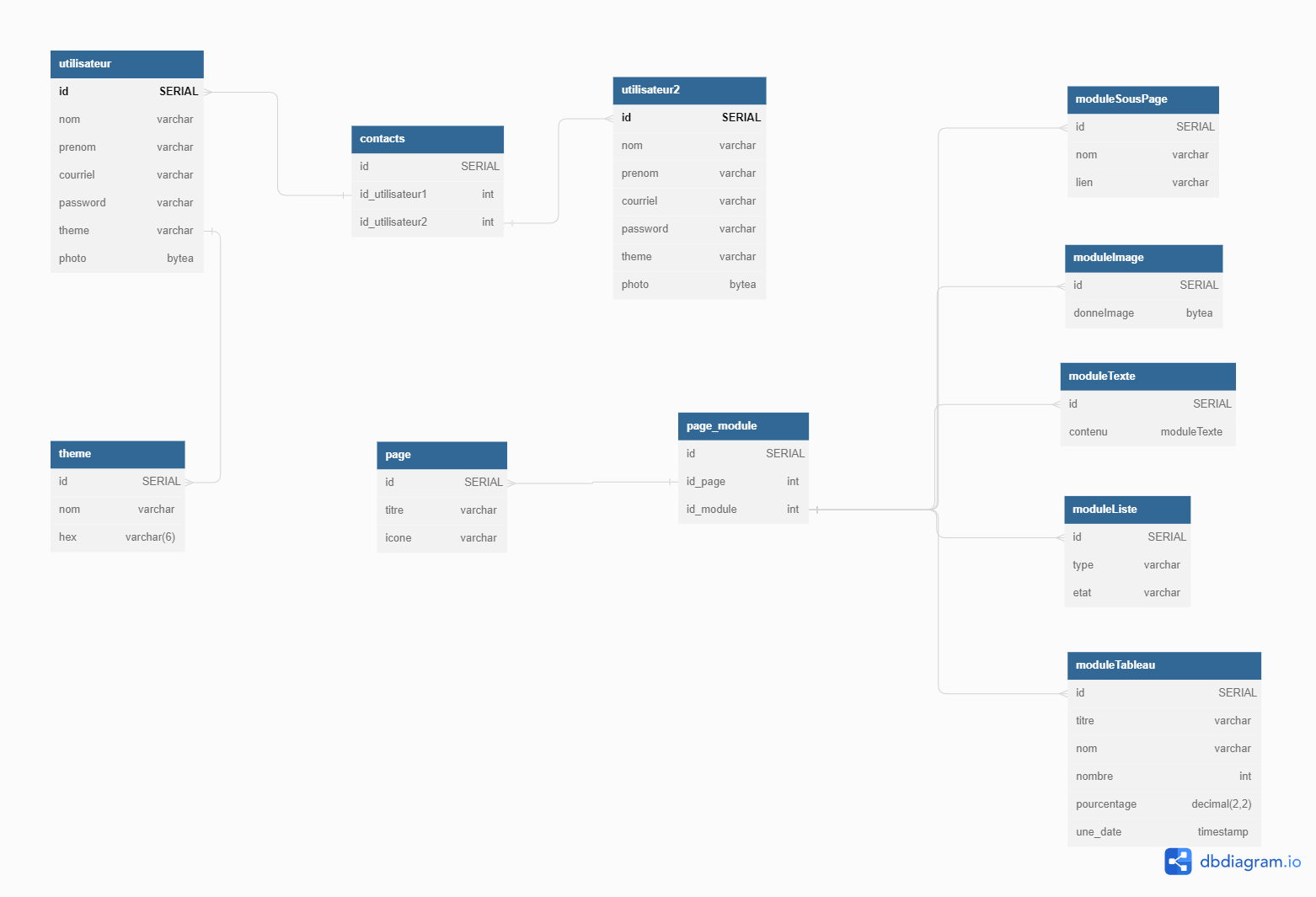
Flowchart:



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

La base de données utilisée est PostgreSQL.

Schéma des tables:



# Éléments de conception :

## Structures de données :

Pour la structure de donnée a 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 a notre algorithme de suggestion de contact car ont 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 tout autres attributs que l’utilisateur ajoute ou retire. Cette structure est la même pour le calendrier et les événements puisque c’est les mêmes données seulement affichées différemment. Le fait que ce soit un objet nous permettra de le modifier aisément avec des methode 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 a la racine est l’utilisateur et le premier étage sont les pages parent . Sous celles-ci sont les modules et les sous-pages.Ont 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 sépares 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 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 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 créer 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 différents classes enfants, par exemple : et il serait intéressant d’utiliser ce pattern en créant différents types d’évènements en plus, dans le cadre de partages de module ajouter différents utilisateurs avec différents permission 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 ensembles 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 :

* Formater les colonnes comme date ou pourcentage
* Un système de recherche qui a pour effet, 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.
* Pour les mots de passe

## 

## 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 linaire pour projeter et prédire les tendances de l’utilisateur dans le module de budget après deux mois de données.

## 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.

# Reference (Mojeed, 2023)s

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

HEUSSLER, J. (2023, 2 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-et-comment-les-utiliser

Mojeed, I. (2023, 2 22). *what is the virtual DOM in React?* Retrieved from LogRocket frontend Analytics: https://blog.logrocket.com/virtual-dom-react/#concept-review-what-is-dom

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)

[Structure derriere notion] https://www.notion.so/blog/data-model-behind-notion

[API notion]https://www.notion.so/blog/creating-the-notion-api

https://www.youtube.com/watch?v=Ke90Tje7VS0&ab\_channel=ProgrammingwithMosh

interesting for building a custumizable interface

https://tailwindcss.com/

vite project guides:

https://www.youtube.com/watch?v=b0pkpcD8Ms4&ab\_channel=ForrestKnight

video by the creator:

https://www.youtube.com/watch?v=DkGV5F4XnfQ&ab\_channel=VueMastery

exploring diferent frameworks :

link at timestamp on react :

https://youtu.be/cuHDQhDhvPE?t=262

Node.js Ultimate Beginner’s Guide in 7 Easy Steps :

https://www.youtube.com/watch?v=ENrzD9HAZK4

calandar:

<https://codesandbox.io/s/xmt6r?file=/src/Details.js>

Correia, E. (2022, 29 novembre). *Graph Data Structure Implementation in JavaScript - Before Semicolon*. Medium. Consulté le 28 février 2023, à l’adresse https://medium.com/before-semicolon/graph-data-structure-implementation-in-javascript-668f291a8a16