Documentation technique - Projet Design Patterns

Romain NICOLAS & Juliette SCHILLING & Joseph Schlesinger

Octobre-Décembre 2023

Documentation technique du projet de tableur de Design Patterns
M1 Informatique, année universitaire 2023-2024

1 Hébergement du code

L'entiéreté de notre code est disponible depuis notre répertoire public Github à l'adresse suivante :

`https://github.com/JulietteSCHILLING/projetDesignPatterns`

2 Guide d'installation, lancement du serveur et connexion

Pour installer et lancer le serveur, il faut récupérer le code depuis le Github du projet puis il suffit d'une unique commande, lancée depuis le répertoire principal projetDesignPatterns :

```
./startServer.sh
```

ou bien directement :

```
npm run devStart
```

Ensuite, pour se connecter au serveur, il faut ouvrir la page :

`http://localhost:3000/`

3 Description des routes mises en place

Pour mieux gérer l'organisation de l'ensemble des routes, nous les avons réparties sur plusieurs fichiers, situés dans le répertoire views.

3.1 users.js

Ce fichiers gère les routes en rapport avec les données utilisateurs (extraction / ajout / modification).

— '/docsPartages/ :idDocument', méthode GET : Récupère la liste des utilisateurs qui n'ont pas accès au document :idDocument dont l'identifiant est passé dans la route :

```
let id = req.params.idDocument;
```

— '/new', méthode POST : Créé un nouvel utilisateur à partir des données passées en req.
Format d'envoi des données depuis le formulaire :

```
FormData(5)  nomCompte → "Pierre", prenomCompte → "Paul", mail →
    "paul@mail.com", login → "loginPaul", mdp → "mdpPaul"
```

Appel à la route :

```
fetch("/users/new",  method : 'POST', body : formData )
```

— '/getNomPrenom', méthode GET : Récupère le nom et prénom de l'utilisateur connecté en récupérant son identifiant dans les cookies.
Format de retour des données :

```
Array [ ... ] 0: Object  nomCompte: "Schilling", prenomCompte: "Juliette"
                    length: 1 <prototype>: Array []
```

Les informations sont donc récupérables de cette manière :

```
const username = data[0].nomCompte; const userFirstname =
data[0].prenomCompte;
```

— '/getNomPrenomId/ :id', méthode GET : Même principe et même format de retour que 'getNomPrenom' mais on récupère le nom et prénom de l'utilisateur dont l'identifiant :id est passé en paramètre.
Appel à la route, avec idUser l'identifiant de l'utilisateur dont on souhaite récupérer le nom et prénom :

```
fetch('/users/getNomPrenomId/'+idUser,  method: 'GET' )
```

— '/delete', méthode POST : Suppression de l'utilisateur dont l'identifiant est passé en req :

```
const userId = req.body.userId;
```

Code de retour : 500 dans le cas où une erreur se produit, 404 dans le cas où l'utilisateur n'est pas trouvé, 200 si tout se passe bien.

— '/connexion', méthode POST : Connecte un utilisateur dont le login et le mot de passe sont passés en req :

```
const login = req.body.login; const mdp = req.body.mdp;
```

Code de retour : 500 dans le cas où une erreur se produit, 404 dans le cas où l'utilisateur n'est pas trouvé, 200 si tout se passe bien.
Connexion de l'utilisateur : son identifiant est stocké dans les cookies.

3.2 document.js

Ce fichiers gère les routes en rapport avec les données des documents (extraction / ajout / modification).

— '/new', méthode POST : Insère un nouveau document dans la base de données, dont les informations sont passées en req, (le créateur est l'utilisateur connecté) :

```
const idCompte = req.cookies.user; const titre = req.body.titre;
```

Cette route va donc créé le document ainsi que toutes les cases.

— '/partagerDocument', méthode POST : Permet de partager un document à un utilisateur via leurs identifiants passés en req :

```
let idDocument = req.body.documentId; let idCompte = req.body.compteId;
```

— '/updateCase', méthode POST : Modifie la valeur d'une case via son identifiant :

```
const caseId = req.body.caseId; const newText = req.body.newText;
```

— '/renommer', méthode POST : Renomme le titre d'un document :

```
const docId = req.body.documentId; const newName =
req.body.newDocumentName;
```

— '/supprimer', méthode DELETE : Supprime un document via son identifiant :

```
const docId = req.body.documentId;
```

3.3 renderUsers.js

Ce fichier gère toutes les routes en rapport avec la récupération des pages liées avec les utilisateurs.

— '/renderNew', méthode GET : Render la page d'inscription d'un utilisateur, fichier users/-new.ejs.

— '/renderConnexion', méthode GET : Render la page de connexion d'un utilisateur, fichier users/connexion.ejs.

— '/renderDelete', méthode GET : Render la page de suppression d'un utilisateur en lui donnant la liste de tous les utilisateurs, fichier users/delete.ejs.

```
res.render('users/delete',  users: users );
```

3.4 renderDocument.js

Ce fichier gère toutes les routes en rapport avec la récupération des pages liées avec les documents.

— '/renderDocument/ :idDoc', méthode GET : Render la page d'affichage du document d'identifiant :idDoc en lui donnant toutes les données du document, fichier document/document.ejs.

```
res.render('document/document',  document: document );
```

— '/renderCreateDocument', méthode GET : Render la page de création d'un document, fichier document/new.ejs.