

Spring 2 : Répartition des tâches

Tâche " Affichage de l'Espace de Stockage" - Imane

BACKEND – Calcul & API

Objectif : exposer l'espace utilisé par l'utilisateur

1.1. Créer un service `StorageService` :

☐ `getTotalStorageUsed(User user)` :

- Somme des tailles (`size`) de tous les fichiers de l'utilisateur
- En octets, puis converti en Mo (ou Go)


2. FRONTEND – Visualisation

Objectif : afficher la donnée dans une interface claire

2.1. Créer une section `StorageUsageCard` sur la page `/dashboard` :

☐ Appeler `GET /storage/usage`

☐ Afficher :

- `15.2 MB utilisés`
- Bar progressif :  30%
- Texte : "15 Mo sur 50 Mo" (si tu gères un quota)

2.2. Composants utiles :

☐ `ProgressBar` (librairie ou composant Tailwind)

☐ Conversion dynamique (`bytes` → `MB` , `GB`)

☐ Couleur de la barre selon usage :

-  < 50%

- 🟡 50–80%
- 🔴 > 80%

Tâche "Folder" - Souhail

. ENDPOINTS API REST

Backend – FolderController

- ☐ GET /folders
→ Récupérer tous les dossiers **racine** de l'utilisateur
- ☐ GET /folders/{parentId}
→ Récupérer les sous-dossiers d'un dossier parent
- ☐ POST /folders
→ Créer un nouveau dossier (optionnellement avec un parent)
- ☐ PUT /folders/{id}
→ Renommer un dossier
- ☐ DELETE /folders/{id}
→ Supprimer un dossier et (optionnellement) ses sous-dossiers

. LOGIQUE MÉTIER (SERVICE)

Backend – FolderService

- ☐ createFolder(name, parentId, user)
 - ☐ getRootFolders(user)
 - ☐ getSubFolders(parentId, user)
 - ☐ renameFolder(folderId, newName, user)
 - ☐ deleteFolder(folderId, user)
→ avec suppression récursive si sous-dossiers
-

. FRONTEND (Next.js)

Pages/Composants

- ☐ Page `/dashboard` : ajouter une **section Dossiers**
 - ☐ Afficher la liste des dossiers racine
 - ☐ Lorsqu'on clique sur un dossier → naviguer dans les sous-dossiers (structure arborescente ou breadcrumb)

 - ☐ Bouton **"Créer dossier"** :
 - Affiche un petit formulaire (nom, dossier parent optionnel)
 - ☐ Action **"Renommer"** → inline edit ou popup
 - ☐ Action **"Supprimer"** → avec confirmation
 - ☐ Affichage responsive avec icônes
-

. SÉCURITÉ & VALIDATION

Backend

- ☐ Vérifier que :
 - Le nom est unique dans le même dossier parent
 - Le nom n'est pas vide
 - L'utilisateur a le droit d'accéder à ce dossier

Frontend

- ☐ Ne pas afficher les actions (supprimer, renommer) si non connecté
 - ☐ Gérer les erreurs API (ex : dossier introuvable, accès refusé)
 - ☐ Afficher des messages de succès/erreur clairs
-

Tâche "File" - Yassine Ouabou - Mourad - Zakaria

1. Téléversement de fichiers (upload)

Frontend

- ☐ Composant drag & drop (`react-dropzone`) ou `<input type="file">`
 - ☐ Upload vers `/files/upload` avec `axios` + `FormData`
 - ☐ Affichage progressif (`progress bar`)
-

2. Liste et navigation des fichiers

Backend

- ☐ `GET /files` → liste tous les fichiers racine de l'utilisateur
- ☐ `GET /files?folderId=X` → liste fichiers dans un dossier donné

Frontend

- ☐ Afficher les fichiers d'un dossier actif
 - ☐ Support **hiérarchie** : **breadcrumb** ou navigation arborescente
 - ☐ Icône personnalisée selon extension (`.pdf` , `.jpg` , `.zip` , etc.)
 - ☐ Format de taille lisible : `120 KB` , `1.5 MB` ...
-

3. Renommer, Supprimer, Télécharger

Frontend

- ☐ Bouton "Renommer" (popup ou inline edit)
 - ☐ Bouton "Supprimer" avec confirmation
 - ☐ Bouton "Télécharger" → utilise `/files/{id}`
 - ☐ Afficher messages de succès/erreur avec `toast` (ex: `react-toastify`)
-

4. Recherche & filtres

Backend

- ☐ `GET /files/search?q=` → retourne les fichiers dont `originalName` contient le mot-clé

Frontend

- ☐ Barre de recherche au-dessus de la liste
 - ☐ Filtres :
 - par dossier
 - par extension (`.pdf` , `.jpg` , etc.)
 - par date (si tu veux aller plus loin)
-

5. Partage de fichiers et dossiers

Backend

- ☐ `POST /files/{id}/share` → partager un fichier/dossier avec un autre utilisateur par email
 - Payload : `{ targetUserEmail: string }`
- ☐ `GET /shared` → lister les fichiers/dossiers partagés **avec l'utilisateur**
- ☐ `GET /shared/by-me` → lister les fichiers/dossiers que l'utilisateur a partagés
- ☐ (optionnel) `DELETE /files/{id}/share/{sharedUserId}` → retirer l'accès

Frontend

- ☐ Bouton "Partager" → ouvre un **popup** pour saisir un email
-

Tâche: Gestion des Utilisateurs && Authentification - Youssef

AUTHENTIFICATION (Register / Login / JWT)

Backend (Spring Boot)

- ☐ Ajouter validation : mot de passe > 6 caractères
- ☐ Hasher le mot de passe avec BCrypt (`PasswordEncoder`)
- ☐ Créer endpoints :
 - ☐ `POST /auth/register` → crée un user
 - ☐ `POST /auth/login` → renvoie `access token` + `refresh token`
- ☐ Implémenter service JWT (sign, verify, decode)
- ☐ Ajouter `JwtAuthenticationFilter` + config `SecurityFilterChain`

Frontend (Next.js) (pas pour le mement)

- ☐ Créer pages : `/register` et `/login`
- ☐ Formulaires avec validation (React Hook Form, par exemple)
- ☐ Envoyer les données à l'API backend (`fetch` ou `axios`)
- ☐ Stocker le `access token` et `refresh token` (cookie ou localStorage)
- ☐ Rediriger vers `/dashboard` après login

GESTION DES TOKENS (Refresh + Logout)

Backend

- ☐ Créer table `Token` (stocke refresh tokens valides)
- ☐ Endpoint : `POST /auth/refresh` → renvoie un nouveau access token
- ☐ Endpoint : `POST /auth/logout` → invalide le refresh token (soft delete)

Frontend

- ☐ Appeler `refresh` automatiquement si `access token` expire
- ☐ Ajouter bouton "Logout" → supprime tokens côté client + appelle `/logout`

FORGOT PASSWORD (avec email)

Backend

- ☐ Endpoint : `POST /auth/forgot-password` → génère token & envoie email
- ☐ Endpoint : `POST /auth/reset-password?token=xxx` → change mot de passe
- ☐ Stocker token de reset temporairement (DB ou en mémoire)
- ☐ Configurer envoi email via SMTP ou Mailtrap

Frontend

- ☐ Page `/forgot-password` avec champ email
 - ☐ Page `/reset-password?token=xxx` pour entrer le nouveau mot de passe
 - ☐ Validation et messages d'erreurs
-

PROFIL UTILISATEUR CONNECTÉ

Backend

- ☐ Endpoint : `GET /user/me` → retourne infos de l'utilisateur connecté
- ☐ Endpoint : `PATCH /user/update` → modifie nom, email, etc.
- ☐ Protection : uniquement accessible à un utilisateur connecté

Frontend

- ☐ Page `/profile`
 - ☐ Afficher données de l'utilisateur (`/me`)
 - ☐ Formulaire pour modifier les données (nom, email...)
-

PROTECTION DES ROUTES

Frontend (Next.js)

- ☐ Créer un **middleware** ou **HOC** pour vérifier que l'utilisateur est connecté
- ☐ Rediriger vers `/login` si le token est absent ou expiré
- ☐ Protéger toutes les pages privées : `/dashboard` , `/profile` , etc.

Backend (Spring Boot)

- ☐ Dans `SecurityConfig` , permettre :

- `/auth/**`, `/forgot-password/**` → `permitAll()`
- Toutes les autres routes → `authenticated()`

☐ Injecter l'utilisateur connecté avec `@AuthenticationPrincipal`