



# Rapport d'analyse et de conception

---

## Projet Système 2025/2026 – Système de Gestion de Fichiers (SGF)

### Membres de l'équipe :

- CHOISY Alexis
- DEGAT Teddy
- DA SILVA FERREIRA Lucas
- FOURNIE Baptiste
- FATIHI Youssef

**Encadrante :** ABOUDA Dhekra

**Date de remise :** 20 avril 2026

---

## Sommaire

- [I. Analyse des besoins de l'utilisateur](#)
  - [II. Définition du système à réaliser](#)
  - [III. Cahier des charges](#)
  - [IV. Structures de données prévues](#)
  - [V. Liste des fonctions principales](#)
  - [VI. Répartition des tâches](#)
  - [VII. Conclusion](#)
- 

## I. Analyse des besoins de l'utilisateur

### 1. Objectifs fonctionnels

Le programme doit permettre à un utilisateur de :

- Interagir via un shell.
- Gérer une arborescence de fichiers et répertoires (création, suppression, déplacement).
- Manipuler le contenu des fichiers (lecture, écriture) via des primitives spécifiques.
- Assurer la persistance des données par la sauvegarde/recharge du SGF sur le disque physique.

### 2. Contraintes

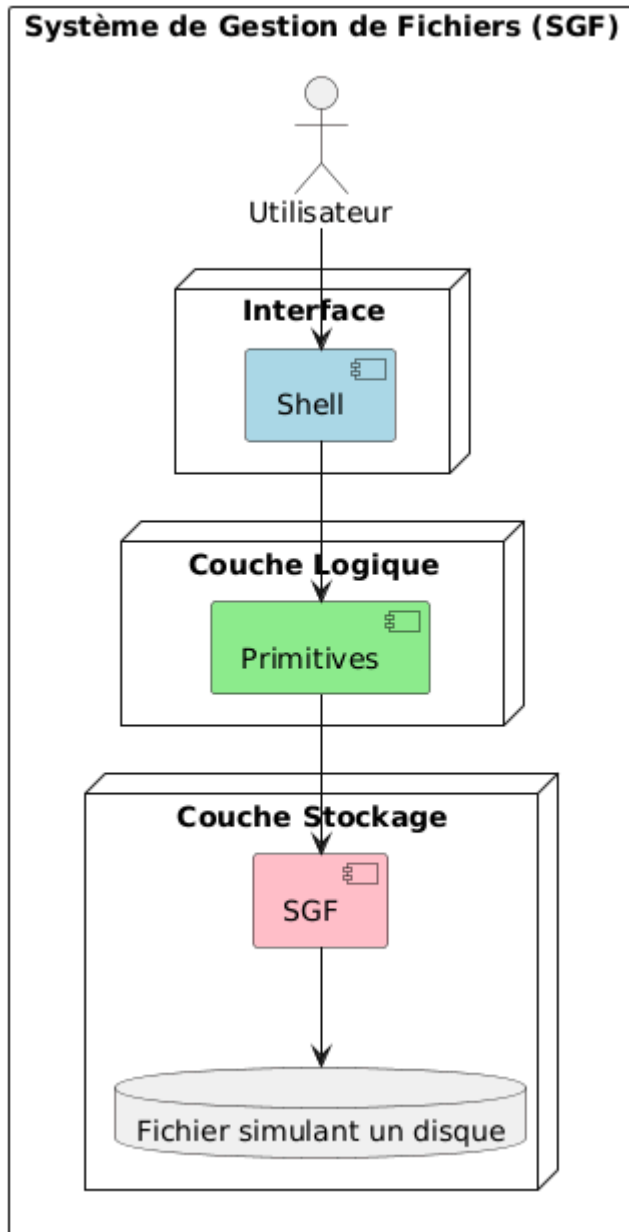
- **Langage :** C uniquement.
  - **Dates limites :** Rendu de l'analyse le 20 avril 2026, conception le 2 mai 2026, et final le 28 mai 2026.
- 

## II. Définition du système à réaliser

## 1. Fonctionnement général

Le système simule une partition de disque dur à travers un fichier. Il s'articule autour de trois couches:

1. **Le Shell** : Analyse les commandes (`ls`, `mkdir`, etc.) et appelle les primitives.
2. **Les Primitives** : Fonctions de bas niveau (`myopen`, `myread`) qui font le lien avec la structure interne.
3. **Le SGF** : Gère l'organisation physique des i-nodes et des blocs de données.



---

## III. Cahier des charges

### 1. Description globale des fonctions

Fonctionnalité	Description
Shell	Gère les commandes <code>ls</code> , <code>mkdir</code> , <code>rmdir</code> , <code>cat</code> , <code>cp</code> , <code>rm</code> , <code>mv</code> .
Gestion Inodes	Attribution et libération d'i-nodes pour fichiers et répertoires.

Fonctionnalité	Description
<b>Gestion Blocs</b>	Allocation de blocs de données (contigus ou non) pour le stockage.
<b>Arborescence</b>	Gestion des répertoires comme des fichiers spéciaux listant des i-nodes.
<b>Persistance</b>	Sauvegarde de l'état complet du disque virtuel dans un fichier binaire.
<b>Superbloc (df)</b>	Fournit les infos sur les blocs/inodes libres et l'espace disque.

## 2. Fonctions bonus

- Gestion de plusieurs utilisateurs simultanés.
- Implémentation des filtres **grep** et **find**.
- Gestion des droits d'accès et dates de modification sur les inodes.

## IV. Structures de données prévues

Le SGF utilise des structures à taille fixe pour simuler la partition .

```
/**
 * @struct inode
 * @brief Un inode est un fichier, il possède des permissions, un type
 * (répertoire par exemple) et pointe sur des blocs de données
 */
typedef struct inode {
    unsigned short perms; // rwxrwxrwx
    char filetype;
    int blocs[MAX_BLOCS];
} inode;

/**
 * @struct bloc
 * @brief Un bloc possède un tableau de données brut concernant des inodes
 */
typedef struct bloc {
    char datas[MAX_BYTES_PER_BLOC];
} bloc;

/**
 * @struct disk
 * @brief Un disque est une liste d'inodes qui pointent sur des blocs de
 * donnée
 */
typedef struct disk {
    char owned_blocs[MAX_BLOCS]; // 1 si possédé par un inode, 0 si libre
    inode inodes[MAX_INODE];
    bloc blocs[MAX_BLOCS];
} disk;

// pour 10 inode qui a 30 blocs de chacun 1024 octets, on a 30720 octets,
```

```
soit
// 30,7 Ko sur le disque
```

---

## V. Liste des fonctions principales

### Primitives Système

- **int create\_inode(disk \*disk, inode \*parent, char inode\_type, char \*name) :** Crée un fichier et retourne son inode.
- **int myread(int inode, char \*buffer, int nombre) :** Lit n octets depuis l'inode.

### Commandes Shell

- **do\_ls()** : Lit le répertoire actuel et affiche les noms et inodes associés.
- **do\_mkdir(char \*nom)** : Crée un i-node de type répertoire et initialise les entrées . et ..

---

## VI. Répartition des tâches

L'équipe est divisée en groupe pour assurer l'intégration finale.

### 1. SGF & Stockage (Alexis et Youssef)

- Initialisation et formatage du disque virtuel (inodes à 0).
- Fonctions de sauvegarde/rechargement du SGF dans le fichier image.

### 2. Primitives & I-nodes (Teddy et Baptiste)

- Gestion de l'allocation des blocs et des i-nodes.
- Développement des primitives mycreat, myopen, myread, mywrite.

### 3. Shell & Commandes (Lucas et Alexis)

- Création de l'interpréteur de commandes (processus fils et exec).
- Implémentation des commandes ls, mkdir, cat, rm.

### Planning prévisionnel

Phase	Description	Échéance
Phase 1	Analyse des besoins et structures de données	20 avril 2026
Phase 2	Conception des algorithmes principaux	2 mai 2026
Phase 3	Développement du SGF et des primitives	Mai 2026
Phase 4	Finalisation du Shell et tests de validation	28 mai 2026
Phase 5	Soutenance et remise du code source	29 mai 2026

---

## VII. Conclusion

Ce rapport constitue le socle de développement de notre Mini SGF. En respectant une séparation nette entre le stockage, les primitives et l'interpréteur, nous visons une architecture robuste conforme aux attentes. La prochaine étape sera la validation des algorithmes de lecture/écriture.