# Rapport projet Mini Shell (mini_sh)

Ce document récapitule la logique et la structure implémentées dans le projet mini shell.

## Structure globale :

**doc** : Dossier réceptionnant la documentation doxygen générée. Contient le doxygen.conf.
**include** : Dossier contenant tout les headers du code source. Il est référencé comme tel dans le CMakeList.txt afin de tous les inclure directement lors du build.
**src** : Dossier source contenant la tout les fichiers .c. Contient directement le programme principal (main).
**src/builtins** : Sources des commandes builtins (echo, pwd, exit et cd).
**src/executor** : Source de la partie exécution / interpréteur des commandes.
**src/parser** : Source de la partie Lexer et Parser du shell.
**src/utils** : Source des parties annexes (gestion de l'historique et alias).

## Fonctionnement :

Le shell, une fois exécuté (sans le mode batch) est une boucle qui tourne tant qu'elle n'est pas cassée (exit, extinction manuelle ou crash). Dans cette boucle, le programme attend une saisie de l'utilisateur afin de l'interpréter. Une fois la commande passée, voici le détail de ce qui se passe :

- La commande passe par le **parser**, qui lui même appellera le **lexer** dans un premier temps.
- Le **lexer** va découper la commande en **token**, en utilisant la structure définit dans **typedef.h**.
- Une fois découpé en **token**, le tout est repassé au **parser** qui va construire la commande selon la structure définit dans **typedef.h.**
- La commande construire est ainsi retournée au programme principal, qui lui va appeler **l'interpréteur (executor)** avec la commande passée en paramètre.
- **L'interpréteur** va utiliser la structure construite par le **parser** pour gérer les sorties, l'enchaînement, l'affichage et le retour au programme principal.

## Choix technique : la structure commande

La structure commande (s_command) est construite comme tel :

- **char *name** : Le nom de la commande
- **char **args** : les arguments de la commande
- **char *input_file** : le fichier pour la redirection de l'entrée
- **char *output_file** : le fichier pour la redirection de la sortie
- **int append_output** : mode de sortie -> 0 pour écraser, 1 -> pour rajouter à la fin
- **struct s_command *pipe_next** : commande suivante, dans le cas où plusieurs commandes sont enchainées avec des opérateurs.
- **struct s_token *next_operator** : opérateur suivant dans le cas où plusieurs commandes s'enchaînent.

# Choix technique : les tokens

Pour gérer les tokens identifiés par le lexer, on a utilisé une structure pour les construire, utilisant un enum pour identifier leur type. Voici la liste des type de token que le lexer peut identifier :

- **TOKEN_WORD** : Partie texte, ne correspond à aucuns opérateurs spécial. Peut être le nom de la commande, comme le nom d'un fichier, comme un simple argument.
- **TOKEN_PIPE** : Opérateur pipe
- **TOKEN_REDIRECT_IN** : Opérateur de redirection d'entrée "<"
- **TOKEN_REDIRECT_OUT** : Opérateur de redirection de sortie ">"
- **TOKEN_APPEND** : Opérateur pour rajouter la saisie à la fin, sans écraser (sortie) ">>"
- **TOKEN_OR** : Opérateur OU "||"
- **TOKEN_AND** : Opérateur ET "&&"
- **TOKEN_BACKGROUND** : Opérateur d'exécution en arrière plan "&"
- **TOKEN_EOF** : End of file, fin de la ligne.