PIGEAT Amaury

RUFFLE Yanis

GADBIN Nathan

Projet de fin d’année : HTML / CSS / JS / C

Sujet : Créer une filmothèque

Table des matières

Cahier des charges fonctionnel :

A- Reformulation des attendus du projet.

Dans ce projet de fin d'année orienté sur le C et le Web, il nous est demandé de créer une filmothèque à partir d'une base de données de 50000 films.

Pour cela nous allons diviser notre projet en 2 parties :

Une partie client contenant le Frontend (HTML, CSS, JS), nous permettant d'afficher toutes les informations des films recherchées et une partie Serveur (C), qui va comporter toute la partie algorithmique nous permettant la recherche d'un film.

B- Engagement sur la liste des fonctionnalités de l'application.

Un cahier des charges nous a alors été donné, comportant les fonctionnalités minimales à inclure dans notre partie Client, puis dans notre partie Serveur.

Pour la partie Client, nous allons donc implémenter les fonctionnalités suivantes :

Pour ce qui est du design, notre site Web devra tout d'abord contenir plusieurs pages et avoir un design esthétique (grâce à l'HTML et au CSS)

Pour la partie technique, notre site Web devra nous permettre de rechercher des films par réalisateur, par durée, mais aussi de savoir quel est le réalisateur qui a fait le plus grand nombre de films et de connaître ce nombre. On devra aussi afficher le temps d'exécution de toutes ces fonctions une fois la validation du formulaire effectuée.

La dernière fonctionnalité requise est un bouton nous permettant d'effacer la base de données de la filmothèque avant de quitter le site, et donc d'éviter les fuites de mémoires.

Puis, pour la partie Serveur, nous allons implémenter les fonctionnalités suivantes :

Nos fichiers de code source devront être séparés (1 fichier main.c contenant le programme principal puis plusieurs fichiers en .c et .h nous permettant le bon fonctionnement de toutes nos fonctions)

Le temps d'exécution de nos fonctions calculant le résultat de la requête devra être calculées puis affichées lors de l'envoi de la requête par l'utilisateur du site. Et une fonction nous permettant de supprimer toute la filmothèque devra être présente et fonctionnelle.

Des attendus nous ont aussi été demandés quant à la complexité de certaines fonctions, comme la fonction nous permettant d'obtenir tous les films d'un réalisateur, qui devra se faire en O(m) (où m est le nombre de caractères du nom du réalisateur).

Les fonctions nous permettant d'obtenir le nom du réalisateur qui a fait le plus de films et son nombre devra se faire en O(1).

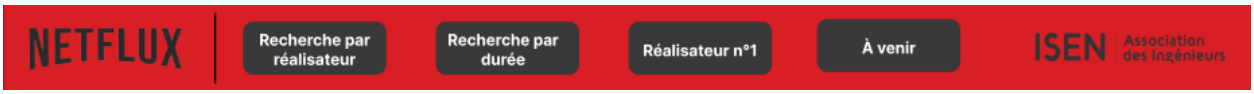
Enfin, la fonction nous permettant d'obtenir tous les films d'une même durée donnée devra elle aussi être en O(1).

Une fois toutes ces fonctionnalités terminées, nous allons essayer d'implémenter de nombreuses fonctionnalités bonus comme chercher un film par catégorie, ou encore ajouter un film directement depuis notre site.

C- Prototype du site web

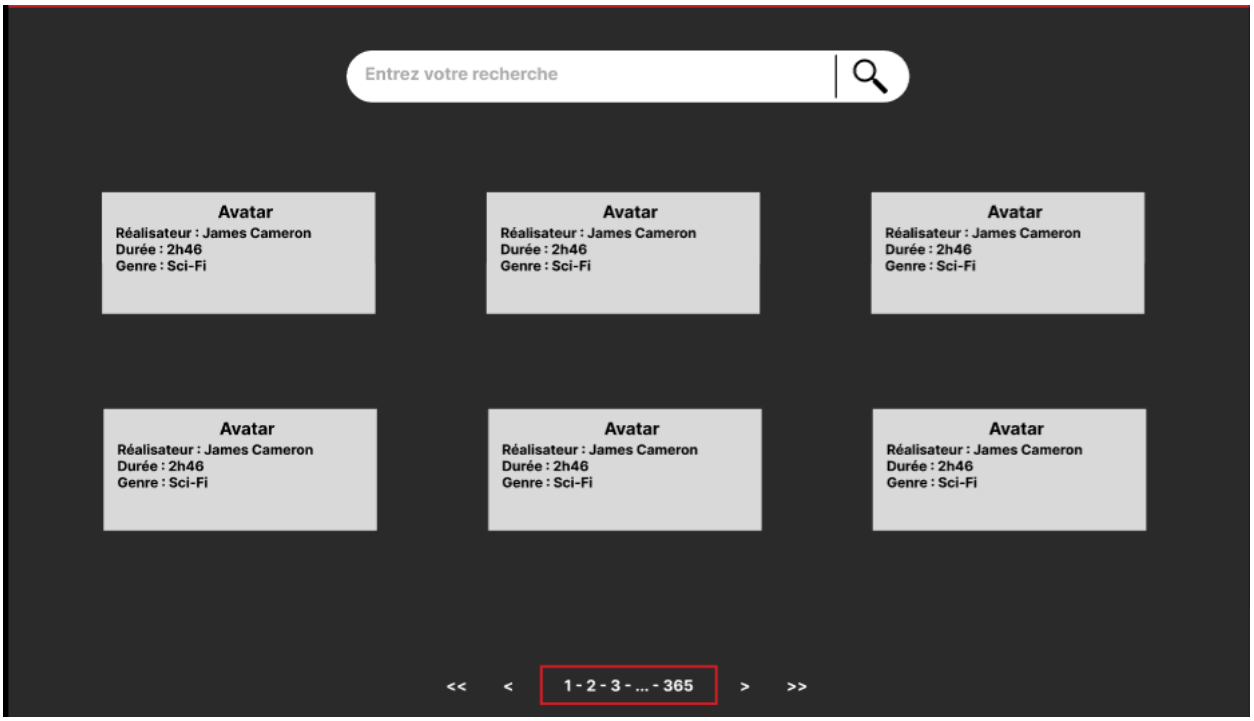
Nous avons réalisé un prototype pour le visuel que l'on souhaiterait obtenir concernant notre site web. Ce prototype a été fait sur Figma, c'est un outil que l'on a découvert en cours de web statique, et qui permet de créer une page et d'y ajouter des choses, des effets et plus, comme on pourrait le faire pour un site web en HTML/CSS/JS.

1- La barre de navigation



Figma permet de créer des rectangles ou l'on peut écrire à l'intérieur. On peut également ajouter des effets comme « onclick » le ferais en JS. Ainsi, lorsque la personne clique sur « recherche par réalisateur », le rectangle s'assombris et la police d'écriture grossis et c'est comme ça que va fonctionner notre filtre. Pareil pour « recherche par durée ». Lorsque la personne clique sur le logo « Netflux », cela réinitialise la recherche, tous les filtres actifs sont désélectionnés et la liste des films complètes s'affiche.

2- Corps du site



Lorsqu'un filtre est sélectionné, une barre de recherche apparait, une fois la recherche effectuée, les fims apparaissent sous le format ci-dessus. Le thème est sombre pour conserver l'intégrité des yeux.

3- Bas de page

Le bas de page contient l'adresse de l'école, ainsi que les noms des 3 développeurs, avec un lien vers leur LinkedIn respectif.

Spécifications techniques :

A- Technologies utilisées

Pour pouvoir travailler sur un espace de travail commun, nous avons créé un repository sur GitHub, qui une fois lié avec Visual studio code, nous permet de pouvoir utiliser les fonctions « commit » et « sync » qui permettent de mettre en liens notre travail. Également, pour pouvoir coder en langage C, nous utilisons CLion, nous développons sur un fichier appart, et une fois le tout fonctionnel, on met les fichiers C dans le dossier cloner avec GitHub, ce qui nous permet de mettre en commun notre travail. Comme dit plus tôt, pour créer la maquette nous avons utilisé Figma. Pour finir, afin d'avoir une feuille de route claire, et un emploi du temps précis, nous avons utilisé Trello. C'est un outil qui permet de créer des catégories, et de pouvoir rajouter des « étiquettes » dans chacune, plus de détails la partie A de planning prévisionnel.

Lien Trello :

https://trello.com/invite/b/4CddTPKr/ATTI2c48a2f89938bfb3ffedc99c7c9c0224D83CE756/projet-fin-dannee-html-css-js-c

B- Choix de structures de données et justification

Premièrement, afin de stocker chaque film, nous avons créé une structure « Movie » :

```
struct Movie {
  unsigned int lengh;
  char* title;
  char* genre;
  char* directorName;
};
```

Lengh prend la durée du film en minutes, title prend le nom du film, dimensionner au plus court, genre prend le style du film, dimensionné au plus court également. Pour finir, directorName prend le nom du réalisateur, avec aucune majuscule, et également dimensionner au plus court.

Pour stocker chaque réalisateur, nous avons créé la structure « Director » :

```
struct Director{
  char* Name;
  struct List* movieList;
};
```

Name prend le nom du réalisateur, dimensionné au plus court, movieList est la liste de film que le directeur à faire, sous forme de liste chainée. Pour obtenir le nombre de film, on prend la taille de la liste.

On retrouve alors une structure « List » et une structure « Cell » qui lui est liée :

```
struct Cell {
  struct Movie* movie;
  struct Cell* next;
};
```

```
struct List {
  struct Cell* head;
  unsigned int size;
};
```

C'est une liste chainée, dans la cellule on retrouve un pointeur vers le film, et un pointeur vers la cellule suivante. Elle permet d'ajouter les films à « l'infini » lorsqu'on initialise le site en O(1).

Afin de stocker tous les réalisateurs, on utilise la structure « NodeTrie » :

```
struct NodeTrie {
  struct NodeTrie* next[27];
  struct Director* director;
};
```

La seule particularité est que on utilise un tableau de taille 27 car on prend tout l'alphabet, et on rajoute le ' - '. Le pointeur « director » est initialisé à NULL, s'il n'est pas nul cela veut dire que la suite de charactère est le nom d'un réalisateur.

Pour finir, on retrouve la structure « Netflux » qui rassemble le tout, et qui est la structure principale :

```
struct Netflux {
  struct NodeTrie* $[27];
  struct List* lenghSort[400];
  struct producer* biggest;
};
```

On prend le stockage est réalisateur « NodeTrie » que l'on nomme $ car l'est la base de l'arborescence. On utilise un tableau de liste afin de chercher en O(1) tous les films ayant la même durée, la durée étant l'indice dans le tableau. Enfin, on retrouve un pointeur vers le réalisateur ayant le plus grand nombre de film sur la plateforme pour y accéder en O(1). A chaque fois qu'on ajoute un film, on doit vérifier qu'il ne change pas, avec un simple « if ».

Planning prévisionnel :

A- Identification des tâches

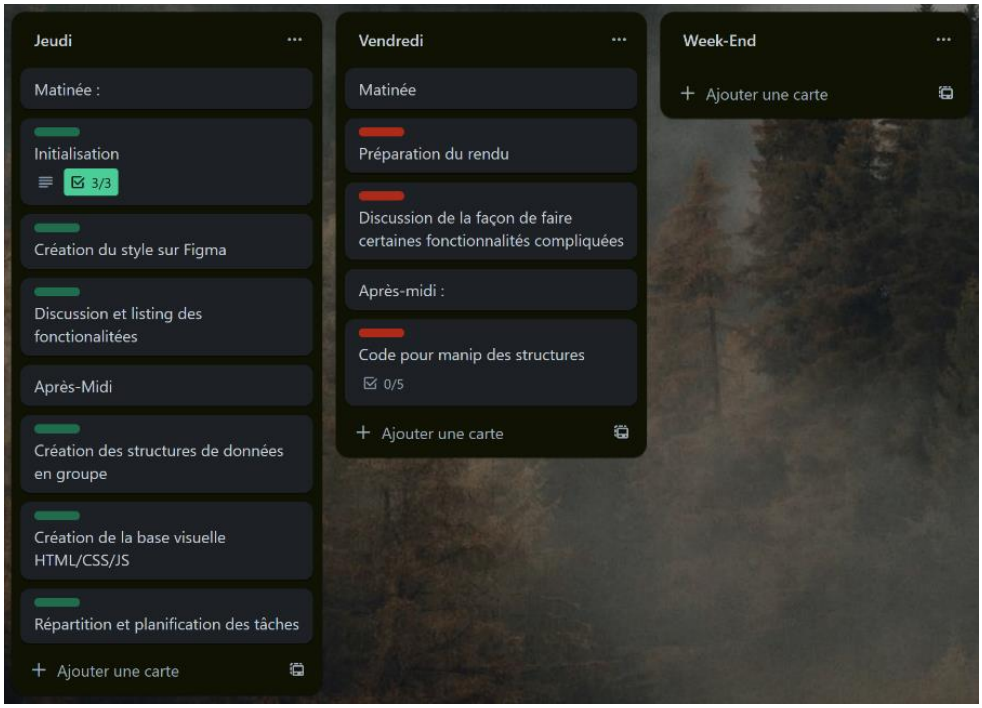
À la suite de la lecture du diaporama contenant les consignes du projet, nous avons commencé par faire plusieurs tâches comme :

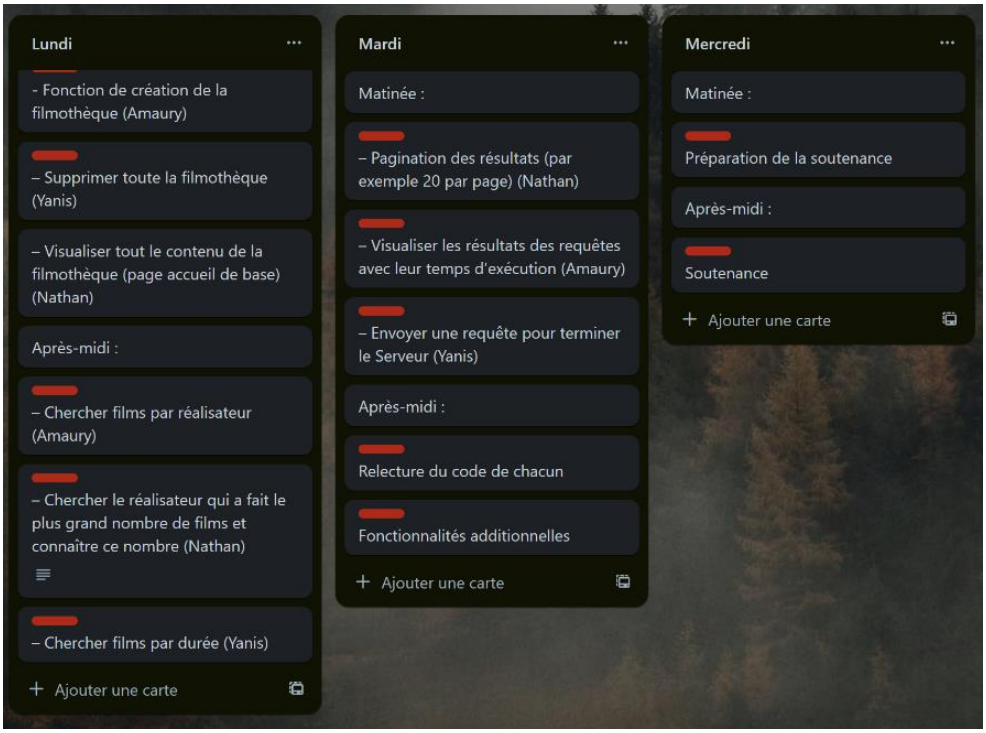
- La création d'une maquette via Figma
- Création du style et de l'HTML coté client
- Choix des structures de données pour le coté Serveur

Ensuite, nous avons identifié les tâches les plus importantes que nous devions faire :

- Coder les différentes Structures de données
- Faire le système de validation des formulaires en JS coté Client
- Coder la fonction de création de la filmothèque coté Serveur
- Implémenter les fonctions de suppression de la filmothèque coté Serveur
- Créer les fonctions permettant de visualiser la filmothèque coté Client
- Faire un programme permettant de chercher un film grâce à son réalisateur coté Serveur
- Faire une fonction permettant de trouver le réalisateur ayant fait le plus de film (+ combien de films) coté Serveur
- Ecrire un programme permettant de chercher un film grâce à sa durée coté Serveur
- Coder une fonction permettant de paginer les résultat (21 films par pages) coté Client
- Réaliser une fonction permettant de visualiser le temps d'exécution et le résultats coté Serveur
- Implémenter une fonction permettant d'envoyer une requête pour terminer le serveur
- Relecture de l'intégralité de notre code
- Faire des fonctions additionnelles
- Préparer la soutenance

B- Planification dans le temps





Pour le Trello, nous avons dressé une liste par jours et chacun des jours sont séparés en deux partie : Matin et Après-midi. Nous avons ensuite, à coté de chacune des tâches, indiqué la personne responsable de celle-ci. De plus, si nous avons fini la tâche, nous mettons son étiquette à vert et si le travail reste à faire, nous mettons son étiquette à rouge. Nous avons aussi dressé

une liste Week-end étant vide et correspondant à une marge. En effet, si le vendredi, nous n'avons pas le temps de finir tout ce que nous avions à faire, on inscrit cela dans la partie Week-end et cela devra être fait durant celui-ci.

C- Affectation de personnes

Pour ce qui est d'Amaury, son travail sera de :

- Coder certaines structures de données
- Coder la fonction de création de la filmothèque coté Serveur
- Faire un programme permettant de chercher un film grâce à son réalisateur coté Serveur
- Réaliser une fonction permettant de visualiser le temps d'exécution et le résultats coté Serveur
- Relire l'intégralité du code
- Faire des fonctions additionnelles
- Préparer la soutenance

Pour ce qui est de Nathan, son travail sera de :

- Coder certaines structures de données
- Faire le système de validation des formulaires en JS coté Client
- Créer les fonctions permettant de visualiser la filmothèque coté Client
- Faire une fonction permettant de trouver le réalisateur ayant fait le plus de film (+ combien de films) coté Serveur
- Coder une fonction permettant de paginer les résultat (21 films par pages) coté Client
- Relire l'intégralité du code
- Faire des fonctions additionnelles
- Préparer la soutenance

Pour ce qui est de Yanis, son travail sera de :

- Coder certaines structures de données
- Implémenter les fonctions de suppression de la filmothèque coté Serveur
- Ecrire un programme permettant de chercher un film grâce à sa durée coté Serveur
- Implémenter une fonction permettant d'envoyer une requête pour terminer le serveur
- Relire l'intégralité du code
- Faire des fonctions additionnelles
- Préparer la soutenance