Rapport de Projet

**Bachelier en informatique de gestion**

**Bloc 2**

**Rapport Liste de voitures**

**Verzwymelen Owen**

**Paquet Benjamin**

**Année académique 2021-2022**

**Table des matières**

1. **Introduction…………………………………………………………………………………………………………**3

Présentation ……………….………………………………………………………………………………..3

Objectifs …….……………….………………………………………………………………………………..3

1. **Mode d’emploi**………………………………………………………………………………………………………..4
2. **Présentation et implémentation du projet……………………………………………………………**5

Schéma de la structure…………………………………………………………………………………..5

Liste des sources et buts………………………………………………………………………………..6

Tableau des fonctions personnelles……………………………………………………………….8

1. **Présentation des algorithmes……………………………………………………………………………..**10
2. **Répartition des tâches………………………………………………………………………………………..**12
3. **Présentation du jeu de tests fonctionnels……………………………………………………………**13
4. **Présentation des améliorations………………………………………………………………………….**14
5. **Mise en évidences des nouvelles matières………………………………………………………….**15

En C………………………………………………………………………………………………………………15

En MPP…………………………………………………………………………………………………………15

En FBD.…………………………………………………………………………………………………………15

1. **Conclusion………………………………………………………………………………………………………….**16

**Introduction**

*Présentation*

Cette année, nous avons développé une application en langage C avec une gestion de base de données et de manipulation Json. Cette application permet d’afficher une liste de voitures sauvés dans une base de données.

L’utilisateur peut donc avoir une liste des marques, des modèles et des versions des voitures.

Le projet est divisée en 2 parties :

Partie Administrateur :

- Un compte administrateur sera défini, il pourra gérer les utilisateurs (ajout, modification et suppression).

Partie Joueur :

- Il devra s’identifier en rentrant son nom d’utilisateur ainsi que son mot de passe.

- Il pourra voir s’afficher les informations des voitures qu’il souhaite

*Objectifs*

- Découper un projet écrit en C en plusieurs fichiers source

- Développer des programmes de tests de ces sources

- Écrire des algorithmes mettant en œuvre les concepts vus en MPP

- Utiliser les notions avancées de la programmation en langage C

- Manipuler le Json avec le C

- Gérer une base de données

- Programmer en équipe en utilisant un système de gestion de version

- Rédiger un rapport

- Présenter et défendre un travail oralement

**Mode d’emploi**

Lors du lancement de l’application, il sera demandé à l’utilisateur d’entrer un identifiant et un mot de passe, si l’utilisateur existe et qu’il a entré le bon de passe, il accède à la suite de l’application, sinon il doit réessayer de s’enregistrer. La suite de l’application dépend de si l’utilisateur est un administrateur ou un simple utilisateur. S’il est administrateur, un menu s’affiche devant lui, lui permettant d’ajouter, modifier ou supprimer des utilisateurs. Si l’utilisateur est un simple utilisateur, il aura devant lui un menu lui permettant de voir différentes informations. Il aura 3 choix différent :

1. Afficher la liste de toutes les marques de voitures de la base de données
2. Afficher la liste des modèles d’une marque demandé, si cette option est choisie, un menu demandant à l’utilisateur d’entrer un nom de marque lui sera demandé préalablement
3. Afficher la liste des versions d’un modèle donné, un menu demandant à l’utilisateur d’entrer un nom de modèle lui sera demandé préalablement

**Présentation et implémentation du projet**

*Schéma de la structure*

Une image contenant texte, écran, capture d’écran

Description générée automatiquement

*Liste des sources et buts*

1. Users.c : fichier qui comporte les fonctions en rapport au joueurs :

* addUser
* modifyUser
* deleteUser
* checkLogin
* checkPwd
* checkExistUser
* checkCorrectPassword
* encrypt
* decrypt

Ces fonctions permettent de travailler avec les utilisateurs, il y a 3 types de fonctions différentes :

1. gestion d’utilisateurs (ajout, modification, suppression)
2. contrôle de données (contrôle du MDP, login, contrôle d’existence de l’utilisateur)
3. cryptage de MDP (cryptage et décryptage)
4. Json.c : fichier qui comportent les fonctions par rapport au json, ces fonctions permettent d’extraire des données d’une chaine json :

* jsonPrimitive
* jsonArray

La première fonction permet d’obtenir la valeur d’une clé d’une chaine json et la deuxième permet d’obtenir un tableau à partir d’une clé de la chaine.

1. Marques.c : fichier qui comportent les fonctions en rapport aux marques de voitures et à la DB marques :

* initaliserConnexion
* cloturerConnexion
* créerTables
* remplirTables
* executerSQL
* insérerDonnee
* listMarques

Ces fonctions permettent de gérer la DB marques avec des fonctions qui initie ou clôture la connexion à la DB, créé les tables de la DB et les remplit ainsi qu’une fonction qui permet de faire une liste de toutes les marques

1. Modeles.c : fichier qui comportent les fonctions en rapport aux modèles de voitures et à la DB modèles :

* initaliserConnexion
* cloturerConnexion
* créerTables
* remplirTables
* executerSQL
* insérerDonnee
* listModeles

Ces fonctions permettent de gérer la DB modeles avec des fonctions qui initie ou clôture la connexion à la DB, créé les tables de la DB et les remplit ainsi qu’une fonction qui permet de faire une liste de tous les modèles d’une marque demandé

1. Versions.c : fichier qui comportent les fonctions en rapport aux versions de voitures et à la DB versions :

* initaliserConnexion
* cloturerConnexion
* créerTables
* remplirTables
* executerSQL
* insérerDonnee
* listVersions

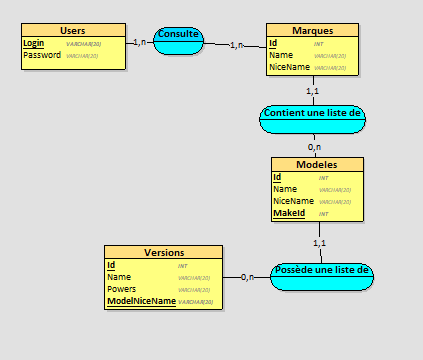
Ces fonctions permettent de gérer la DB versions avec des fonctions qui initie ou clôture la connexion à la DB, créé les tables de la DB et les remplit ainsi qu’une fonction qui permet de faire une liste de toutes les versions d’un modèle demandé

1. main.c : fichier qui comporte l’interface graphique

*Fonctions personnelles*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nom | Paramètres | Type de retour | Utilité | Fichier source |
| checkLogin | struct user user | unsigned int | Vérifie si le login correspond aux critères (nb de caractères) | users.c |
| checkPwd | struct user user | unsigned int | Vérifie si le mdp correspond aux critères (nb de caractères, présence de nombre et de majuscule) | users.c |
| checkExistUser | struct user user | unsigned int | Vérifie que l’utilisateur entré existe dans le fichier | users.c |
| checkCorrectPassword | struct user user | unsigned int | Vérifie que le mdp entré correspond au mdp de l’utilisateur dans le fichier | users.c |
| encrypt | char password[] | char | Crypte le mdp entré | users.c |
| decrypt | char password[] | char | Décrypte le mdp entré | users.c |
| lireJusqueCaractereAttendu | char \*chaineJson  int caractereCourant  int caractereAttendu  char\* messageErreur | void | Place la curseur du flux sur le caractère attendu afin de lire la chaine à partir de ce caractère | json.c |
| lectureChaine | char \*chaineJson  int caractereCourant  char\* chaineLue messageErreur | char | Lis la chaine de caractère et la renvoie | json.c |
| initialiserConnexion | / | void | Initie la connexion avec la DB correspondante au fichier | marques.c  modeles.c  versions.c |
| cloturerConnexion | / | void | Cloture la connexion à la DB correspondante au fichier | marques.c  modeles.c  versions.c |
| creerTables | / | int | Crée la table de données avec les champs et leur taille | marques.c  modeles.c  versions.c |
| remplirTables | / | int | Remplit la DB avec les données des fichiers texte | marques.c  modeles.c  versions.c |

**Schéma des bases de données**



**Présentation des algorithmes**

*Fonction listVersions (suffit de modifier nom des variables pour listMarques et listModeles)*

Fonction listVersions(\*\*listVersions en caractères, niceName en caractères, nbElements en entier, \*erreur en caractères)

constante entiere DIM = 50

Début

i en entier

tableau caractères resultats[DIM][DIM], erreur[DIM]

\*sqlResult = SELECT \* FROM VERSIONS ORDER BY NAME ASC FOR JSON PATH, ROOT('Version') en requete SQL

si jsonArray(sqlResult,niceName, resultats, nbElements, erreur)

alors

saisir nbElements

pour i variant de 0 à nbElements

listVersions[i][0] 🡨 resultats[i][1]

listVersions[i][1] 🡨 resultats[i][2]

fin pour

fin si

fin function

fonction jsonArray(\*chaineJson en caractères, \*nomAttribut en caractères, nbElements en entier, \*erreur en caractères)

début

i,j=0 , continuer=1 , trouve=1, caractereCourant=0 en entier

messageErreurCarAttendu[DIM],messageErreurLectureChaine[DIM], chaineLue,nomAttri en caractères

tant que nomAttri !=nomAttribut

lireJusqueCaractereAttendu (chaineJson,caractereCourant,'\"',messageErreurCarAttendu)

lireJusqueCaractereAttendu (chaineJson,caractereCourant,':',messageErreurCarAttendu)

lectureChaine(chaineJson, caractereCourant,nomAttri,messageErreurLectureChaine)

fin tant que

lireJusqueCaractereAttendu(chaineJson,caractereCourant,'[',messageErreurCarAttendu)

lireJusqueCaractereAttendu(chaineJson,caractereCourant,':',messageErreurCarAttendu)

tant que continuer <> 1

pour i variant de 0 à DIM

lireJusqueCaractereAttendu (chaineJson, caractereCourant, ':', messageErreurCarAttendu)

si messageCarAttendu[0] != NULL

alors

messageErreur 🡨 messageErreurCarAttendu

sinon

lectureChaine (chaineJson, caractereCourant, chaineLue, messageErreurLectureChaine)

si messageErreurLectureChaine[0]!=NULL

alors

messageErreur 🡨 messageErreurLectureChaine

trouve=0

sinon

resultat[j][i] 🡨 chaineLue

fin si

fin si

fin pour

tant que ((caractereCourant = fgetc(chaineJson)) != EOF) et (caractereCourant != ‘{‘) et (caractereCourant != ‘]’)

fin tant que

si caractereCourant==’]’

alors

continuer=0

fin si

j=j+1

fin tant que

nbElements=j

retourne trouve

fin fonction

Fonction lireJusqueCaractereAttendu(\*chaineJson en caractères,\*caractereCourant en entier, caractereAttendu en entier, \*MessageErreur en caractères)

début

si \*caractereCourant == caractereAttendu

alors

retourne 0

fin si

tant que ((\*caractereCourant = lire caractere de chaineJson) != EOF) et (\*caractereCourant != caractereAttendu) et ((\*caractereCourant == ' ') ou (\*caractereCourant == '\t') ou (\*caractereCourant == '\n'))

fin tant que

si \*caractèreCourant != \*caractèreAttendu

alors

messageErreur 🡨 ‘Erreur : caractère attendu non trouvé’

fin si

fin fonction

fonction lectureChaine( \*chaineJson en caractères, \*caractereCourant en entier, \*chaineLue en caractères, \*messageErreur en caractères)

début

buffer en caractères

si caractereCourant != ‘\"’

alors

messageErreur 🡨 ‘Erreur : ouvrant de chaine non trouvé’

fin si

buffer = (char \*)malloc(100)

positionDansBuffer=0 en entier

tant que ((\*caractereCourant lire caractere de chaineJson) != EOF) et (\*caractereCourant != '\"')

buffer[positionDansBuffer] = (char)\*caractereCourant

positionDansBuffer++

fin tant que

si \*caractereCourant != '\"'

alors

libérer buffer;

messageErreur 🡨 ‘Erreur: fermant de chaine non trouver’

fin si

\*caractereCourant = fgetc(chaineJson)

buffer[positionDansBuffer] = '\0'

positionDansBuffer++

char \*chaine = (char \*)malloc(positionDansBuffer)

chaine 🡨 buffer

liberer buffer

chaineLue=chaine

**Répartition des taches**

|  |  |
| --- | --- |
| **Itération** | **Auteur** |
| ***Partie 1*** | |
| Itération 1 (Json) | Verzwymelen Owen |
| Itération 2 (User) | Paquet Benjamin |
| Itération 3 (Looping et DB) | Verzwymelen Owen |
| ***Partie 2*** | |
| Itération 4 (Marques) | Verzwymelen Owen |
| Itération 5 (Modèles) | Paquet Benjamin |
| Itération 6 (Versions) | Paquet Benjamin |
| ***Partie 3*** | |
| Itération 7 (Liste Marques) | Verzwymelen Owen |
| Itération 8 (Liste Modèles) | Paquet Benjamin |
| Itération 9 (Liste Versions) | Verzwymelen Owen |
| ***Partie 4*** | |
| Itération 10 (Liste voitures) | Paquet Benjamin |
| Itération 11 (Interface ) | Paquet Benjamin |
| Itération 12 (Rapport) | Verzwymelen Owen |

**Présentation du jeu de tests fonctionnels**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | UC1 : Créer un compte | Création d’un compte (correcte) | - Entrer le chiffre 1  - Entrer un login faisant entre 8 et 20 caractères et n’existant pas  - Entrer un mot de passe contenant une majuscule, un chiffre, et faisant entre 8 et 20 caractères. | Création d’un utilisateur | Création d’un utilisateur |
| 2 | UC1 : Créer un compte | Création d’un compte (login incorrecte) | - Entrer le chiffre 1  - Entrer un login ne faisant pas entre 8 et 20 caractères  - Entrer un mot de passe contenant une majuscule, un chiffre, et faisant entre 8 et 20 caractères. | Message d’erreur indiquant que le login n’est pas correct | Message d’erreur indiquant que le login n’est pas correct |
| 3 | UC1 : Créer un compte | Création d’un compte (mot de passe incorrecte) | - Entrer le chiffre 1  - Entrer un login faisant entre 8 et 20 caractères  - Entrer un mot de passe soit a qui n’a pas de majuscule, soit pas de chiffre, soit ne faisant pas entre 8 et 20 caractères | Message d’erreur indiquant que le mot de passe n’est pas correct | Message d’erreur indiquant que le mot de passe n’est pas correct |
| 4 | UC2 : Connexion | Connexion à un compte existant | - Entrer le chiffre 2  - Entrer un login existant  - Entrer le mot de passe correspondant au login | Message « connexion réussie » et affichage du menu | Message « connexion réussie » et affichage du menu |
| 5 | UC2 : Connexion | Connexion à un compte (login inexistant) | - Entrer le chiffre 2  - Entrer un login inexistant  - Entrer un mot de passe | Message « login incorrecte » | Message « login incorrecte » |
| 6 | UC2 : Connexion | Connexion à un compte (mot de passe incorrect) | - Entrer le chiffre 2  - Entrer un login existant  - Entrer un mot de passe incorrect | Message « mot de passe incorrecte » | Message « mot de passe incorrecte » |
| 7 | UC2 : Connexion | Connexion à un compte admin | - Entrer le chiffre 2  - Entrer le login « adminlog »  - Entrer le mot de passe correspondant | Message « Connecter à un compte administrateur » et affichage du menu » | Message « Connecter à un compte administrateur » et affichage du menu » |
| 8 | UC3 : Ajouter utilisateur | Créer utilisateur depuis compte admin | - Entrer le chiffre 1  - Entrer un login faisant entre 8 et 20 caractères et n’existant pas  - Entrer un mot de passe contenant une majuscule, un chiffre, et faisant entre 8 et 20 caractères. | Création d’un utilisateur | Création d’un utilisateur |
| 9 | UC3 : Ajouter utilisateur | Créer utilisateur depuis compte admin (login incorrecte) | - Entrer le chiffre 1  - Entrer un login ne faisant pas entre 8 et 20 caractères  - Entrer un mot de passe contenant une majuscule, un chiffre, et faisant entre 8 et 20 caractères. | Message d’erreur indiquant que le login n’est pas correct | Message d’erreur indiquant que le login n’est pas correct |
| 10 | UC3 : Ajouter utilisateur | Créer utilisateur depuis compte admin (mot de passe incorrecte) | - Entrer le chiffre 1  - Entrer un login faisant entre 8 et 20 caractères  - Entrer un mot de passe soit a qui n’a pas de majuscule, soit pas de chiffre, soit ne faisant pas entre 8 et 20 caractères | Message d’erreur indiquant que le mot de passe n’est pas correct | Message d’erreur indiquant que le mot de passe n’est pas correct |
| 11 | UC4 : Modifier utilisateur | Modifier utilisateur depuis compte admin (correcte) | - Entrer 2  - Entrer login existant  - Entrer le mot de passe du log à modifier  - Entrer un login faisant entre 8 et 20 caractères et n’existant pas  - Entrer un mot de passe contenant une majuscule, un chiffre, et faisant entre 8 et 20 caractères. | Modification de l’utilisateur | Modification de l’utilisateur |
| 12 | UC4 : Modifier utilisateur | Modifier utilisateur depuis compte admin (login incorrecte) | - Entrer 2  - Entrer login existant  - Entrer le mot de passe du log à modifier  - Entrer un login ne faisant pas entre 8 et 20 caractères et n’existant pas  - Entrer un mot de passe contenant une majuscule, un chiffre, et faisant entre 8 et 20 caractères. | Message d’erreur indiquant que le login n’est pas correct | Message d’erreur indiquant que le login n’est pas correct |
| 13 | UC4 : Modifier utilisateur | Modifier utilisateur depuis compte admin (login incorrecte) | - Entrer 2  - Entrer login existant  - Entrer le mot de passe du log à modifier  - Entrer un login faisant entre 8 et 20 caractères et n’existant pas  - Entrer un mot de passe soit a qui n’a pas de majuscule, soit pas de chiffre, soit ne faisant pas entre 8 et 20 caractères | Message d’erreur indiquant que le mot de passe n’est pas correct | Message d’erreur indiquant que le mot de passe n’est pas correct |
| 14 | UC4 : Supprimer utilisateur | Supprimer utilisateur depuis compte admin (correct) | -Entrer 3  - Entrer login de l’utilisateur à supprimer | Message « utilisateur supprimer » et suppression | Message « utilisateur supprimer » et suppression |
| 15 | UC4 : Supprimer utilisateur | Supprimer utilisateur depuis compte admin (incorrect) | - Entrer 3  - Entrer login n’existant pas | Message d’erreur « ce login n’existe pas » | Message d’erreur « ce login n’existe pas » |
| 16 | UC5 : Afficher liste | Afficher liste marque depuis compte client | - Entrer 1 | Affichage des marques |  |
| 17 | UC5 : Afficher liste | Afficher liste modèles depuis compte client | - Entrer 2 | Affichage des modèles |  |
| 18 | UC5 : Afficher liste | Afficher liste version depuis compte client | - Entrer 3 | Affichage des versions |  |

**Présentation des améliorations**

1. Pouvoir quitter l’application à tout moment.
2. Faire en sorte que lorsqu’on écrit son mot de passe/ s’inscrit, on puisse revenir en arrière.
3. Annuler une action.
4. Redesigner l’interface pour qu’elle soit plus attractive et attrayante pour l’utilisateur
5. Possibilité d’afficher les versions des modèles de la marque entrés avec une option après la demande d’affichage de la liste des modèles

**Mise en évidence des matières**

*En C*

- Manipulations binaires

- Librairies

- Chaînes de caractères

- Expressions régulières avec regex

- L’allocation dynamique par ‘malloc’

- Fonctions ‘strcpy’, ‘strcat’, ‘strcmp’,’strstr’

- utilisation des fonctions de Mysql.h pour la db

- manipulation de chaine Json

*En MPP*

- Logique sur l’ouverture, l’écriture, la lecture et la fermeture dans les fichiers

- Logique sur la déclaration et l’utilisation d’une structure

- Logique des tris

- L’allocation dynamique

- Modélisation d’un modèle entité-association via looping

*En FBD*

- Manipulation de champs par l’opérateur.

- Les différents modes d’ouverture des fichiers

- Gestion de flux

- Les nouvelles structures en C (struct)

- Manipulation d’une base de données

**Conclusion**

En conclusion, ce projet nous a permis d’approfondir la matière abordée pendant toute l’année que ce soit dans les cours théoriques et pratiques. Aussi, le fait de travailler à plusieurs sur un seul projet et de le diviser en plusieurs fichiers nous a appris à collaborer, communiquer avec le groupe et à travailler de manière structurée. Ce projet nous a mis à l’épreuve et nous a demandé de la patience ainsi que de la concentration pour le réaliser au mieux et fournir un travail acceptable malgré quelques problèmes rencontrés pendant le développement du jeu.