Ecole Centrale Casablanca

Rapport projet de base des données

House Inventory 

Conception et réalisation dune application de gestion des stocks

---

Réalisé par :
Badr-Eddine Marani
Aymen Lakhyar

Encadrants :
Prof. Mahmoud El Hamlaoui
Prof. Abderrahim Ait Wakrime

Ce rapport est soumis dans le cadre du projet de base de données.

Année académique 2021 – 2022

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

1.1 Présentation de projet

Dans le cadre de la formation dingénieur de lÉcole Centrale Casablanca, les élèves de deuxième année doivent réaliser un projet base de données. Celui-ci consiste sur une durée de deux mois avec une fréquence dune séance par semaine, à conceptualiser et réaliser un système de gestion dune base de données. Cest dans cette optique que nous avons eu le plaisir de choisir le thème : système d'inventaire de maison.

Aujourd'hui, les logiciels sont utilisés dans presque tous les domaines de la vie quotidienne. Les téléphones portables, les télévisions, les appareils de chauffage, les voitures et même certains de nos grille-pains sont dotés de logiciels qui les rendent plus faciles à utiliser et améliorent ainsi la vie des personnes qui les utilisent. Avec l'arrivée de cette nouvelle technologie dans nos voitures et nos téléphones portables, il semble étrange que l'endroit où nous passons le plus de temps soit à la traîne dans cette tendance. Nos maisons sont désormais la prochaine facette de notre vie que nous devrions améliorer.

FIGURE 1.1 – Smart Home

La "Smart Home" est une maison qui utilise les nouvelles technologies pour faciliter la vie de ses occupants. Que vous soyez une personne âgée vivant seule ou un étudiant, chacun peut les utiliser. Avant de pouvoir construire une véritable maison intelligente, nous devons développer

les composants qui constitueront la base du système. L'un de ces composants, d'une importance vitale, est un moyen d'organiser et de garder la trace des objets de la maison. Le système dinventaire de maison est une idée simple qui peut être utilisée comme l'un des composants de la maison intelligente. Imaginez qu'un terrible accident détruise la maison d'un particulier ; il arrive souvent que le compte exact des biens personnels du propriétaire devienne un problème entre lui et la compagnie d'assurance. L'inventaire de la maison résout ce problème en gardant la trace de chaque objet de la maison, ce qui simplifie considérablement le recouvrement des pertes en cas d'accident.

Dans la suite de ce rapport, nous exposerons tous les résultats que nous avons obtenus à lissu de ce projet.

1.2 Plan de travail

1.2.1 Organisation du rapport

Le rapport est divisé en cinq chapitres principaux décrivant le projet, les processus et les résultats, voir figure 3.1. Le premier chapitre est une introduction au projet. La deuxième partie consiste en une étude préalable sur la méthode que nous allons utiliser, les exigences des utilisateurs (les exigences sont en gros divisées en deux groupes : les exigences fonctionnelles et les exigences non fonctionnelles). Le troisième chapitre décrit la première phase du projet qui consiste à réaliser différents diagrammes (dictionnaire des données, MCD, et MLD). Le quatrième chapitre de ce rapport présente le prototype final, une présentation du prototype qui a été développé à partir des résultats identifiés et classés par ordre de priorité. Pour conclure ce rapport, un dernier chapitre représentant la discussion et la conclusion du rapport est présenté.

FIGURE 1.2 – Structure du rapport

1.2.2 Diagramme de Gantt

Tout au long de semestre, notre groupe a organisé des réunions, pendant lesquelles plusieurs aspects de la solution sont discutés. Les différentes composantes de notre produit final ont été choisi par le groupe, après plusieurs recherches. Le tableau et le diagramme Gant ci-dessous montrent les étapes de réalisation du prototype.

FIGURE 1.3 – Diagramme de Gantt

Chapitre 2

Étude préalable

2.1 Méthode danalyse et de conception : Merise

Une méthode est la mise en uvre de plusieurs étapes (méthodologiques), d'une démarche, de principes, d'outils.

La méthode MERISE est pour cela une méthodologie qui propose des étapes pour la mise en place et la conduite de projets informatiques. Le terme MERISE, acronyme signifiant méthode d'études et de réalisations de systèmes informatiques pour les entreprises, est le plus couramment et le plus largement utilisé pour la conception et la réalisation de bases de données en France. Cette méthode vise à remplacer le système de gestion manuelle d'une organisation par un système automatisé de traitement de l'information. Elle vise, d'une part, à démontrer les problèmes potentiels du système existant et, d'autre part, à apporter des améliorations à ce dernier. Les facteurs considérés dans l'étude sont la collecte, le stockage, le traitement et la transmission de l'information.

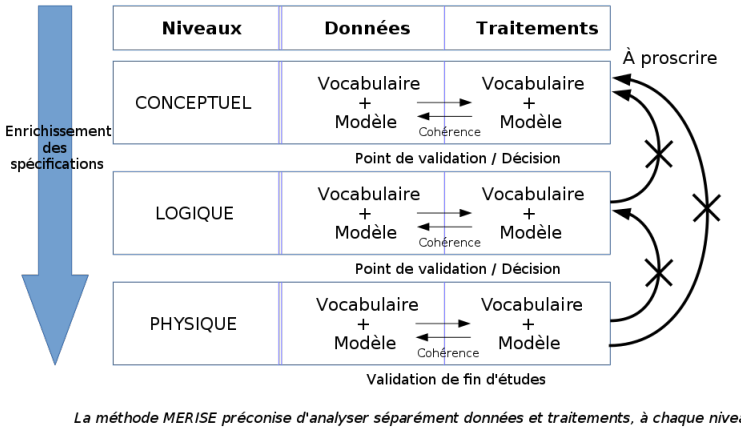

FIGURE 2.1 – Méthode MERISE

Ainsi, MERISE permet d'entreprendre une étude de l'existant et de passer à la conception proprement dite, si cette dernière est la solution, en procédant par exemple comme suit :

Étude de l'existant : Ici on décrit de manière détaillée la structure existante qui décrit la situation réelle de l'organisation et son fonctionnement.

Analyse et évaluation de l'existant : Identification des problèmes et des moyens de les résoudre.

Conception et mise en place du nouveau système : proposer un modèle qui aboutit à la base de données ou encore proposer des interfaces et des programmes en utilisant certains langages de programmation et systèmes de gestion de bases de données.

Toutes ces étapes peuvent être résumées dans la figure 2.1.

2.2 Analyse du problème

2.2.1 Contexte

Le but est d'améliorer la gestion des stocks dune maison. On va créer une application pour gérer efficacement leurs inventaires. Cette application permettra de gérer à la fois les articles achetés, et les produits, mais aussi de gérer l'inventaire hebdomadaire des lots par les dirigeants. Nous avons donc créé le cahier des charges suivant, qui décrit notre démarche.

2.2.2 Analyse et aboutissements

Pour des raisons de simplicité, on considère que les **produits** que nous stockerons dans la base de données sont tous des produits destinés uniquement à être consommés (viande, fruit, etc..).

Chaque produit a un identifiant, un libellé et une brève description, et chacun d'eux appartient à une **catégorie** spécifique. Il est possible de définir des caractéristiques communes entre certains produits. Ces caractéristiques communes peuvent être divisées en deux groupes : le premier est lié à la **forme** du produit (poids, hauteur, volume), et le second contient des **informations** générales telles que la date d'expiration, le prix unitaire, la quantité, et la température (minimale et maximale). Cet inventaire est créé indépendamment pour chaque utilisateur afin que les données de chacun soient séparées de celles des autres.

Chaque produit doit être inscrit à l'inventaire de la maison (**stock**). Il a un identifiant, un nom et la quantité de produit disponible. Il est acheté auprès d'une ou plusieurs entreprises à la fois. Ici, nous considérons que l'**entreprise** ne vend qu'un seul type de produits. Une entreprise peut être présente dans plus d'une **ville**. Ainsi, une ville peut contenir plusieurs entreprises.

Quand un **utilisateur** achète un nouveau produit ou augmente la quantité d'un produit. Nous le mettons dans le stock. Chaque produit comporte la date à laquelle il a été ajouté, ainsi qu'une date à laquelle nous le récupérons, au cas où il serait périmé ou que nous ne l'aurions plus en stock.

Chaque fois qu'un produit est ajouté au stock, l'utilisateur peut effectuer un **contrôle** pour vérifier si le produit est périmé ou non. L'utilisateur peut également vérifier, de temps en temps, la disponibilité d'un certain produit.

Chaque processus de contrôle doit être suivi d'une brève description de ses résultats. Bien évidemment, les résultats de ces contrôles peuvent être variés. Mais, pour simplifier, nous ne considérons dans ce projet que ces résultats typiques : produit périmé, produit non disponible en stock, produit disponible en stock.

2.2.3 Fonctionnalités

Parcourir les articles : L'utilisateur parcourt les articles disponibles dans son inventaire.

Insérer un élément : L'utilisateur insère un élément dans son inventaire.

Supprimer un élément : L'utilisateur supprime un élément de l'inventaire.

Modifier l'élément : L'utilisateur peut modifier les propriétés d'un élément.

Login : Se connecter au système.

Logout : Se déconnecter du système

Modifier l'identifiant et le mot de passe : L'utilisateur peut changer son identifiant et son mot de passe.

2.3 Lidentification des besoins

2.3.1 Les besoins fonctionnels

Le système doit assurer les fonctionnalités suivantes :

1. Le système doit permettre aux utilisateurs de modifier leurs mots de passe par mesure de sécurité.

2. Le système doit permettre aux utilisateurs d'enregistrer toutes les informations relatives aux produits achetés, à ceux qui restent dans le stock.

3. Le système doit être capable d'exporter tous les enregistrements stockés afin de les analyser.

2.3.2 Les besoins non fonctionnels

Ergonomie et Convivialité Lapplication devra offrir aux utilisateurs un cadre de travail organiser et harmonieux. Par des interfaces graphiques claires et lisibles, la possibilité deffectuer des recherches facilement, et de sauvegarder leurs données.

Facilité dutilisation et facteur humain Nous voulons faciliter lutilisation de lapplication en simplifiant dabord sa manipulation, ensuite en proposant un guide dutilisation précis, afin quil soit utilisable par nimporte quel agriculteur. Aucune compétence en informatique ne sera nécessaire pour manipuler ce produit. Lutilisateur aura un accès à un menu "aide" où il pourra sinformer sur lapplication, et communiquer ses difficultés ou préoccupations rencontrées.

Sécurité Chaque utilisateur aura un compte privé et un mot de passe quil devra fournir pour accéder au site, et ce mot de passe pourra au besoin être modifier pour assurer plus de sécurité.

Le système ne doit pas occuper plus que l'espace requis pour un programme normal à l'espace disque.

2.4 Lanalyse des besoins : Identification des acteurs

L'application sera généralement utilisée par deux types d'utilisateurs, un utilisateur à domicile et un administrateur système. Les utilisateurs à domicile sont des personnes qui utilisent le système pour leur inventaire quotidien. Les administrateurs du système constituent un groupe très restreint d'utilisateurs qui ont accès à l'ensemble du système et peuvent en modifier tous ses aspects.

Chapitre 3

Conception

3.1 Dictionnaire de données

	Dictionnaire de données				
	Nom symbolique	**Signification**	**Type**	**Taille**	**Contraintes, règles de calcul**
	user_id	ID d'utilisateur	Numérique	100	Automatique
	user_nom	Nom de l'utilisateur	Alphanumérique	50	Obligatoire
	user_email	Adresse mail utilisateur	Alphanumérique	50	
	user_password	Mot de passe utilisateur	Alphanumérique	50	Obligatoire
Utilisateur	user_derniere_session	Dernière sessios	Date	8	Forme JJ-MM-AAAA
	produit_id	ID produit	Numérique	100	Automatique
	libelle_produit	Libellé produit	Alphanumérique	50	Obligatoire
	produit_temperature_max	Température maximal	Numérique	100	Obligatoire, >0
	produit_temperature_min	Température minimal	Numérique	100	Obligatoire, >0
Produit	produit_description	Description du produit	Alphanumérique	100	Obligatoire
	catégorie_id	ID du catégorie	Numérique	100	Automatique
Catégorie	libelle_catégorie	Libellé catégorie	Alphanumérique	50	Obligatoire
	taille_id	ID taille	Numérique	100	Automatique
	libelle_taille	Libellé de la taille	Alphanumérique	100	Obligatoire
	taille_hauteur	Hauteur du produit	Numérique	100	Obligatoire, >0
	taille_largeur	Largeur du produit	Numérique	100	Obligatoire, >0
Taille	volume	Volume du produit	Numérique	100	Obligatoire, >0
	information_id	ID information	Numérique	100	Automatique
	date_expiration	Date d'expiration du produit	Date	8	Forme JJ-MM-AAAA
	date_d_arrivée	Date d'arrivée du produit	Date	8	Forme JJ-MM-AAAA
Information	prix_unité	Prix unité	Numérique	100	Obligatoire, >0
	entreprise_id	ID entreprise	Numérique	100	Automatique
	entreprise_nom	Nom d'entreprise	Alphanumérique	50	Obligatoire
	entreprise_email	Adresse mail d'entreprise	Alphanumérique	50	
Entreprise	entreprise_description	Description d'entreprise	Alphanumérique	100	
	code_postale	Code postale	Numérique	10	Obligatoire / Contient au mois 6 chiffres
Ville	ville	Nom du ville	Alphanumérique	50	Obligatoire
	contrôle_id	ID du contrôle	Numérique	100	Automatique
	contrôle_date	Date du contrôle	Date	8	Forme JJ-MM-AAAA
Contrôle	contrôle_résultat	Résultat du contrôle	Alphanumérique	100	Obligatoire

TABLE 3.1 – Dictionnaire de données

3.2 Modèle conceptuel de données (MCD)

3.2.1 Modèle EA

A laide de lanalyse effectuée, nous avons pu établir le schéma conceptuel ci-dessous.

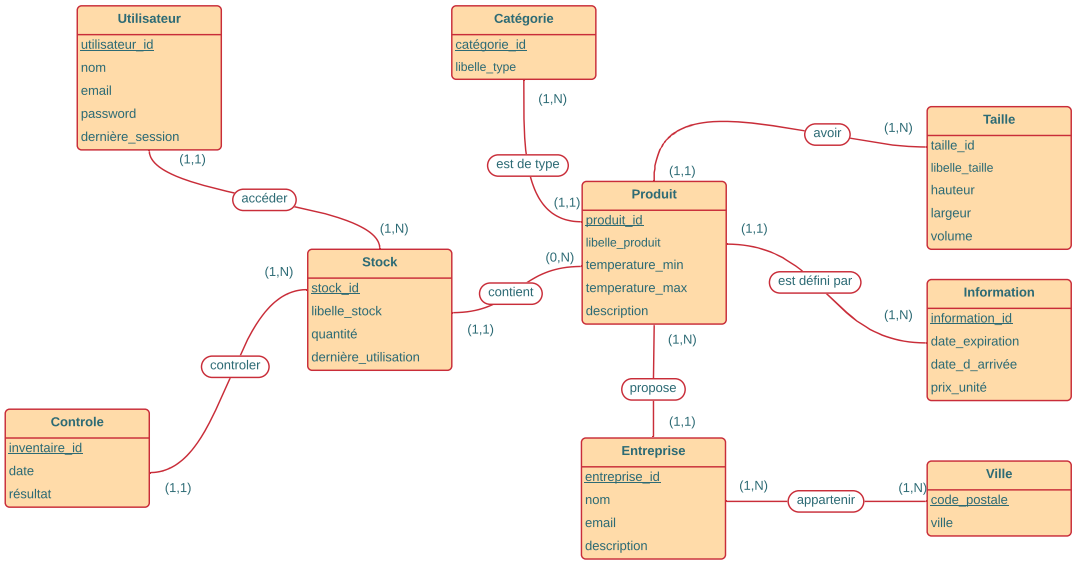

FIGURE 3.1 – Modèle EA

3.2.2 Description des entités

Dans le diagramme entité-association, nous trouvons les entités suivantes :

Ville Nous savons que chaque ville a son propre et unique code postal. C'est pourquoi nous pouvons le considérer ici comme étant une clé primaire.

Catégorie Une catégorie est définie par son identifiant, et son libellé. Nous avons choisi de le considérer comme une entité, au lieu de le mettre comme un attribut dans la table des produits. En effet, il existe de nombreux produits appartenant à la même catégorie (fruits, légumes, viande, etc.).

Entreprise Une entreprise est définie par son identifiant, son nom, email, et une brève description.

Utilisateur Un utilisateur est défini par un identifiant, un nom d'utilisateur et un mot de passe pour accéder à son compte. De plus, pour garder une trace de ses activités, nous avons ajouté la date de la dernière session comme attribut.

Contrôle Un contrôle est définit par un identifiant, un nom, la date de contrôle, ainsi que le résultat de contrôle.

Stock Un stock est associé a un et un seul utilisateur. Il est définit par un identifiant unique, et un libellé.

Information Il contient un certain nombre d'informations génériques et communes à de nombreux objets, comme la température du produit, la date d'expiration, le prix, la quantité, et la date d'entrée.

Taille Elle fonctionne de la même manière que l'entité : "Information". L'entité "Taille" contient des informations relatives à la forme géométrique de l'objet (hauteur, poids, volume).

Produit Chaque produit est acheté auprès une où plusieurs entreprise. Il est définit par un identifiant unique, un libellé, une description, des informations uniques relative au produit comme la température du produit (la plus basse et la plus haute).

3.3 Modèle logique de données (MLD)

3.3.1 Modèle relationnel

Grâce au modèle conceptuel précédent, on a établi le modèle relationnel qui suit :

UTILISATEUR (**utilisateur_id**, utilisateur_nom, utilisateur_email, utilisateur_password, utilisateur_dernière_session, #stock_id)

STOCK (**stock_id**, libellé_stock, #produit_id)

CONTRÔLE (**contrôle_id**, contrôle_date, #stock_id)

PRODUIT (**produit_id**, libellé_produit, température_max, température_min, produit_description, #catégorie_id, #information_id, #taille_id)

CATÉGORIE (**catégorie_id**, libellé_catégorie)

ENTREPRISE (**entreprise_id**, entreprise_nom, entreprise_email, entreprise_description, #produit_id)

VILLE (**code_postale**, ville)

APPARTENIR (**#code_postale, #entreprise_id**)

TAILLE (**taille_id**, libellé_taille, taille_hauteur, taille_largeur, taille_volume)

INFORMATION (**information_id**, date_expiration, date_d_arrivée, prix_unité, quantité)

3.3.2 Dépendance fonctionnelle

utilisateur_id	$\longrightarrow$	#stock_id, utilisateur_nom, utilisateur_email, utilisateur_password, utilisateur_dernière_session
stock_id	$\longrightarrow$	#produit_id, libellé_stock
contrôle_id	$\longrightarrow$	#stock_id, contrôle_date
produit_id	$\longrightarrow$	#catégorie_id, #information_id, #taille_id, libellé_produit, température_max, température_min, produit_description
catégorie_id	$\longrightarrow$	libellé_catégorie
entreprise_id	$\longrightarrow$	#produit_id, entreprise_nom, entreprise_email, entreprise_description
code_postale	$\longrightarrow$	ville
(**#code_postale, #entreprise_id**)	$\longrightarrow$	*évident.*
taille_id	$\longrightarrow$	libellé_taille, taille_hauteur, taille_largeur, taille_volume
information_id	$\longrightarrow$	date_expiration, date_d_arrivée, prix_unité, quantité

3.3.3 Normalisation de la base en 3 forme normale

UTILISATEUR (**utilisateur_id**, utilisateur_nom, utilisateur_email, utilisateur_password, utilisateur_dernière_session, #stock_id)

`utilisateur_id` est le clé. En effet, toutes combinaison des attributs : `utilisateur_nom`, `utilisateur_email` et `utilisateur_password` ne permet pas de retrouver les valeurs des autres attributs.

On a la 1er forme normale (FN1). Car, tous les attributs sont atomiques. De plus, on a dépendance totale de la clé principale `utilisateur_id`, ainsi, elle est aussi atomique. Donc, on a la 2ème forme normale (FN2). Enfin, aucune combinaison entre les attributs ne suffissent pour déduire les autres attributs. On la 3ème forme normale (FN3).

$\Longrightarrow$ ***FN3 est respectée.***

STOCK (**stock_id**, libellé_stock, #produit_id)

`stock_id` est le clé. Car, tous les autres attributs dépendent fonctionnellement du clé `stock_id`. Tous les attributs sont atomiques, on a bien la 1er forme normale (FN1).

La clé est simple et non composite, et tous les autres attributs dépendent de la clé. Donc, on a la 2ème forme normale (FN2). Ainsi, aucune combinaison entre les attributs ne suffissent pour déduire les autres attributs. On la 3ème forme normale (FN3).

$\Longrightarrow$ ***FN3 est respectée.***

CONTRÔLE (**contrôle_id**, contrôle_date, #stock_id)

$\Longrightarrow$ ***FN3 est respectée.***

PRODUIT (**produit_id**, libellé_produit, température_max, température_min, produit_description, #catégorie_id, #information_id, #taille_id)

`produit_id` est le clé. Car, tous les autres attributs dépendent fonctionnellement du clé `produit_id`. Tous les attributs sont atomiques, on a bien la 1er forme normale (FN1).

La clé est simple et non composite, et tous les autres attributs dépendent de la clé. Donc, on a la 2ème forme normale (FN2). Ainsi, aucune combinaison entre les attributs ne suffissent pour déduire les autres attributs. On la 3ème forme normale (FN3).

$\Longrightarrow$ ***FN3 est respectée.***

CATÉGORIE (**catégorie_id**, libellé_catégorie)

$\Longrightarrow$ ***FN3 est respectée.***

ENTREPRISE (**entreprise_id**, entreprise_nom, entreprise_email, entreprise_description, #produit_id)

`entreprise_id` est le clé. Car, tous les autres attributs dépendent fonctionnellement du clé `entreprise_id`. Tous les attributs sont atomiques, on a bien la 1er forme normale (FN1). De plus, on a dépendance totale de la clé principale. En effet, le clé est simple (non composite) et tous les autres attributs dépendent de la clé. On a donc le 2ème forme normale (FN2). Enfin, on a 3ème forme normale (FN3). Car, aucune combinaison entre les attributs ne suffissent pour déduire les autres attributs.

$\Longrightarrow$ ***FN3 est respectée.***

VILLE (**code_postale**, ville)

On a la 1er forme normale (FN1). En effet, tous les attributs sont atomiques, et l'identifiant `code_postale` est unique. De plus, on a dépendance totale de la clé principale. En effet, le clé est simple (non composite) et l'attribut `ville` dépend de la clé. On a donc le 2ème forme normale (FN2). On a 3ème forme normale (FN3). Car, l'attribut `ville` dépend de la clé.

$\Longrightarrow$ ***FN3 est respectée.***

APPARTENIR (**#code_postale, #entreprise_id**)

Les deux clés sont atomiques.

$\Longrightarrow$ ***FN3 est respectée.***é

TAILLE (**taille_id**, libellé_taille, taille_hauteur, taille_largeur, taille_volume)

$\Longrightarrow$ ***FN3 est respectée.***

INFORMATION (**information_id**, date_expiration, date_d_arrivée, prix_unité, quantité)

$\Longrightarrow$ ***FN3 est respectée.***

Chapitre 4

Réalisation

4.1 Introduction

4.2 Environnement de développement

Il sagit dans cette partie didentifier les différentes caractéristiques de lenvironnement matériel et logiciel qui nous ont servi à limplémentation de notre application.

4.2.1 Environnement matériel

La machine utilisée pour réaliser ce site est un ordinateur portable Lenovo qui a la configuration suivante :

Système dexploitation Windows 10

Processeur Intel(R) Core(TM) i7-8550U CPU

Mémoire RAM installée 8,00 Go

4.2.2 Environnement logiciel

Nous allons maintenant donner une brève définition de chaque langage et bibliothèque utilisés.

Git

Git est un système de contrôle de version. Il conserve un historique de toutes les modifications apportées au code. Les modifications sont stockées dans une base de données spéciale appelée "repository", également connue sous le nom de "repo".

FIGURE 4.1 – Git

L'utilisation de Git pour le développement de logiciels présente deux avantages principaux :

— Le suivi des modifications et des mises à jour. On est capable de voir qui a fait quels changements. Git permet également de savoir quand et pourquoi un changement a été effectué.

— Permettre de travailler en collaboration. Les projets de développement de logiciels nécessitent généralement que de nombreuses personnes travaillent ensemble. Git fournit aux développeurs un moyen systématique de le faire. Ainsi, les développeurs se concentrent sur le projet plutôt que sur les longues sessions de communication entre les autres développeurs.
— Il existe d'autres systèmes de contrôle de version, tels que Subversion, Mercurial, etc. Cependant, Git est le plus populaire.

Pendant le processus de réalisation, nous utiliserons la plateforme GiHub.

Python

Python est un langage de programmation open-source, interprété, orienté objet, de haut niveau avec une sémantique dynamique, avec des applications dans de nombreux domaines, notamment la programmation web, les scripts, le calcul scientifique et l'intelligence artificielle. Il est bien structuré et nombre de ses fonctionnalités prennent en charge la programmation fonctionnelle, la méta programmation et les méta-objets.

FIGURE 4.2 – Python

Il est très populaire et largement utilisé par des organisations connues telles que la NASA et Google, entre autres.

La bibliothèque `mysql-connector-python`

MySQL est un système de gestion de bases de données relationnelles (SGBDR) open source et gratuit qui utilise le langage de requête structuré (SQL). Dans cet article, nous allons partager une explication complète sur MySQL. Il s'agit d'un système de gestion de base de données relationnelle (SGBDR) open-source avec un modèle client-serveur. Un SGBDR est un logiciel ou un service utilisé pour créer et gérer des bases de données basées sur un modèle relationnel.

FIGURE 4.3 – mysql-connector-python

Nous avons décidé dans ce projet d'utiliser python au lieu du langage Java, pour la simplicité et la disponibilité de la bibliothèque `mysql-connector-python`.

4.3 Création de la base de données

```
1 --
2 -- Database : Home
3 --
4 DROP DATABASE IF EXISTS Home;
5 CREATE DATABASE IF NOT EXISTS Home;
```

```
USE Home;
```

4.3.1 Création des tables de la base de données

Utilisateur

```
--
-- Structure de la table : utilisateur
--
DROP TABLE IF EXISTS user;
CREATE TABLE IF NOT EXISTS user (
    user_id INT NOT NULL AUTO_INCREMENT,
    user_name VARCHAR(50) NOT NULL,
    user_password VARCHAR(50) NOT NULL,
    user_last_session DATETIME NOT NULL,

    PRIMARY KEY (user_id)
);
```

Ville

```
--
-- Structure de la table : City
--
DROP TABLE IF EXISTS City;
CREATE TABLE IF NOT EXISTS City (
    zip_code INT NOT NULL AUTO_INCREMENT,

    city_name VARCHAR(50) NOT NULL,

    PRIMARY KEY (zip_code)
);
```

Catégorie

```
--
-- Structure de la table : Category
--
DROP TABLE IF EXISTS Category;
CREATE TABLE IF NOT EXISTS Category (
    category_id INT NOT NULL AUTO_INCREMENT,

    category_nom VARCHAR(50) NOT NULL,

    PRIMARY KEY (category_id)
);
```

Entreprise

```
--
-- Structure de la table : Company
--
DROP TABLE IF EXISTS Company;
CREATE TABLE IF NOT EXISTS Company (
    company_id INT NOT NULL AUTO_INCREMENT,

    product_id INT NOT NULL,

    company_name VARCHAR(50) NOT NULL,
    company_email VARCHAR(50) NOT NULL,
    company_description VARCHAR(50) NOT NULL,

    PRIMARY KEY (company_id),
    FOREIGN KEY (product_id) REFERENCES Product(product_id)
);
```

Contrôle

```
--
-- Structure de la table : Control
--
DROP TABLE IF EXISTS Control;
CREATE TABLE IF NOT EXISTS Control (
    control_id INT NOT NULL AUTO_INCREMENT,

    stock_id INT NOT NULL,

    control_date DATETIME NOT NULL,

    PRIMARY KEY (control_id),
    FOREIGN KEY (stock_id) REFERENCES Stock(stock_id)
);
```

Stock

```
--
-- Structure de la table : Stock
--
DROP TABLE IF EXISTS Stock;
CREATE TABLE IF NOT EXISTS Stock (
    stock_id INT NOT NULL AUTO_INCREMENT,

    product_id INT NOT NULL,

```

```
    stock_name VARCHAR(50) NOT NULL,

    PRIMARY KEY (stock_id),
    FOREIGN KEY (product_id) REFERENCES Product(product_id)
);
```

Information

```
--
-- Structure de la table : Information
--
DROP TABLE IF EXISTS Information;
CREATE TABLE IF NOT EXISTS Information (
    information_id INT NOT NULL AUTO_INCREMENT,

    information_expiration_date DATETIME NOT NULL,
    information_entry_date DATETIME NOT NULL,
    information_unit_price INT NOT NULL,
    information_quantity INT NOT NULL,

    PRIMARY KEY (information_id)
);
```

Taille

```
--
-- Structure de la table : Shape
--
DROP TABLE IF EXISTS Shape;
CREATE TABLE IF NOT EXISTS Shape (
    shape_id INT NOT NULL AUTO_INCREMENT,

    shape_name VARCHAR(50) NOT NULL,
    shape_height VARCHAR(50) NOT NULL,
    shape_weight VARCHAR(50) NOT NULL,
    shape_volume VARCHAR(50) NOT NULL,

    PRIMARY KEY (shape_id)
);
```

Produit

```
--
-- Structure de la table : Product
--
DROP TABLE IF EXISTS Product;
```

```
CREATE TABLE IF NOT EXISTS Product (
    product_id INT NOT NULL AUTO_INCREMENT,

    shape_id INT NOT NULL,
    information_id INT NOT NULL,
    category_id INT NOT NULL,

    product_name VARCHAR(50) NOT NULL,
    product_temperature_max VARCHAR(50) NOT NULL,
    product_temperature_min VARCHAR(50) NOT NULL,
    product_description VARCHAR(50) NOT NULL,

    PRIMARY KEY (product_id),
    FOREIGN KEY (shape_id) REFERENCES Shape(shape_id),
    FOREIGN KEY (information_id) REFERENCES Information(information_id),
    FOREIGN KEY (category_id) REFERENCES Category(category_id)
);
```

Appartenir

```
--
-- Structure de la table : BELONGING
--
DROP TABLE IF EXISTS BELONGING;
CREATE TABLE IF NOT EXISTS BELONGING (
    zip_code INT NOT NULL,
    company_id INT NOT NULL,

    FOREIGN KEY (zip_code) REFERENCES City(zip_code),
    FOREIGN KEY (company_id) REFERENCES Company(company_id),
    CONSTRAINT PRIMARY KEY (
        zip_code,
        company_id
    )
);
```

4.3.2 Contraintes d'intégrité

```
--
-- Contraintes d'intégrité
--

ALTER TABLE Product
ADD CONSTRAINT
    CHECK (
        product_temperature_max > 0 AND
```

```
9           product_temperature_min > 0
10      );
11
12  ALTER TABLE Shape
13  ADD CONSTRAINT
14      CHECK (
15          shape_height > 0 AND
16          shape_weight > 0 AND
17          shape_volume > 0 AND
18      );
19
20  ALTER TABLE Information
21  ADD CONSTRAINT
22      CHECK (
23          information_quantity > 0 AND
24          information_entry_date <= information_expiration_date AND
25          information_unit_price > 0
26      );
27
28  ALTER TABLE Control
29  MODIFY control_results ENUM (
30      'Produit␣est␣périmé',
31      'Produit␣est␣disponible',
32      'Produit␣n␣est␣pas␣disponile'
33  );
```

4.3.3 Commandes SQL

Cette partie a pour objectif : tester toutes les requêtes avant de les ajouter dans le code permettant de contrôler lapplication.

Requêtes concernant l'affichage d'un produit

```
1  SELECT product_name, category_name
2  FROM Product, Category
3  WHERE Product.category_id = Category.category_id;
```

Requêtes concernant l'affichage de derniers résultats de contrôle des produits

```
1  SELECT product_name, control_results
2  FROM Product, Control, Stock
3  Where
4          Control.stock_id = Stock.stock_id AND
5      Stock.product_id = Product.product_id
6  GROUP by product_name;
```

Requêtes concernant modification de mot de passe

```
UPDATE User
SET
    user_password = password
WHERE
    user_name = name;
```

Requêtes concernant la suppression d'un produit

```
DELETE FROM User
WHERE product_name = name
```

4.4 Interfaces

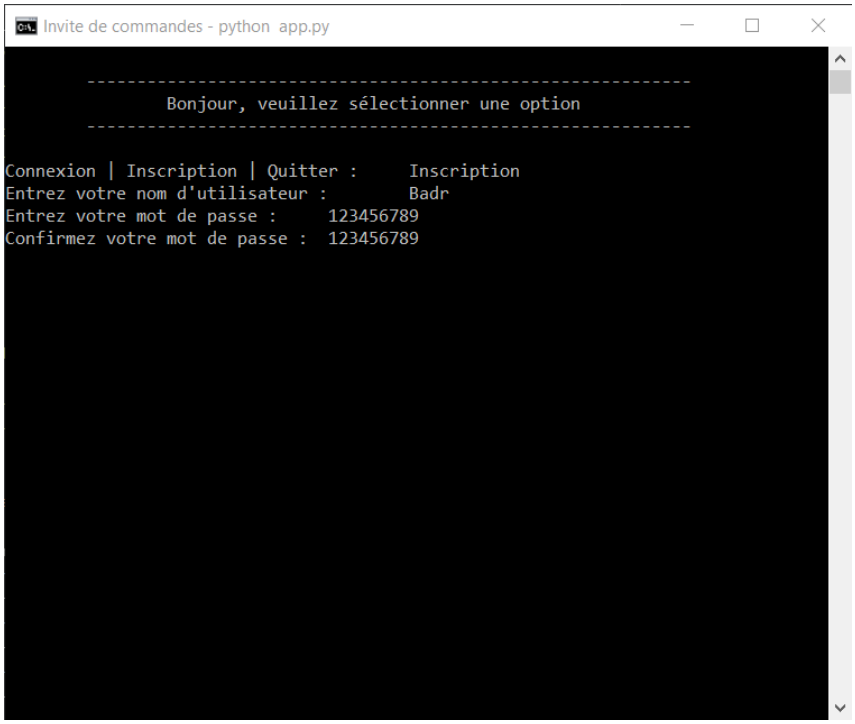

FIGURE 4.4 – Ouvrir un compte

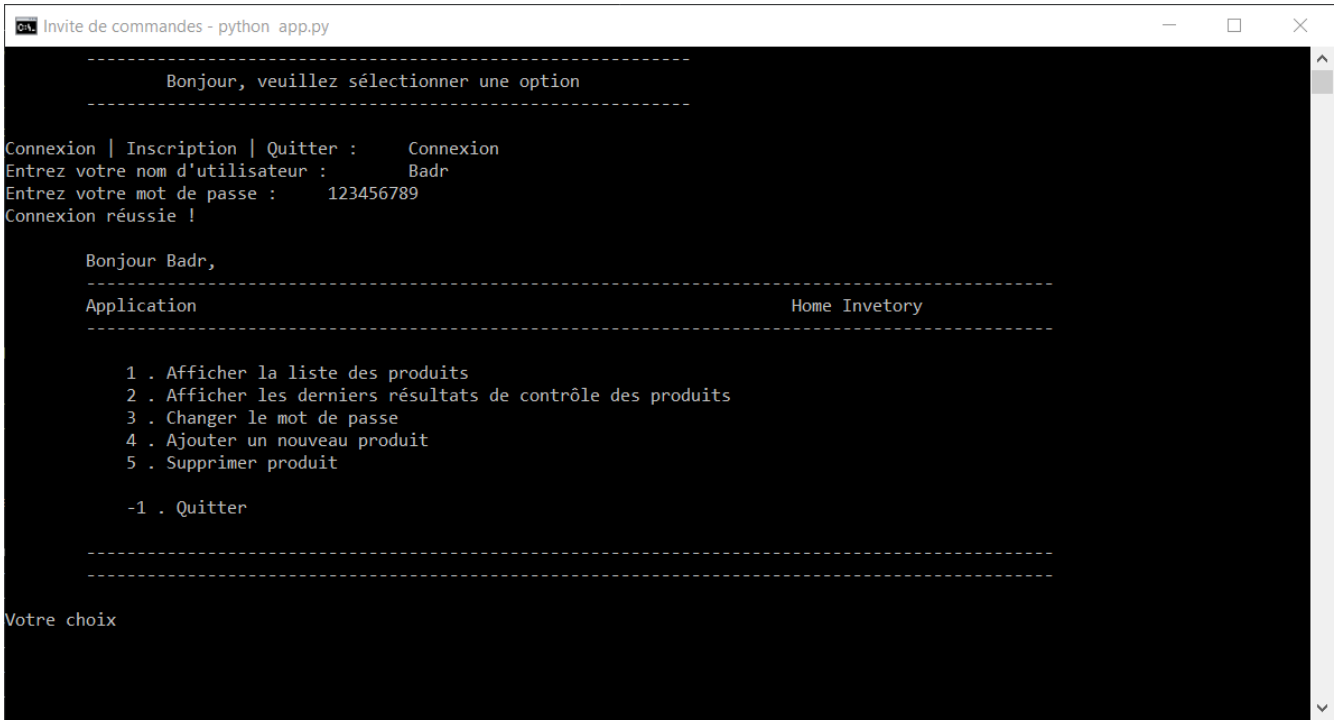

FIGURE 4.5 – Page d'accueil

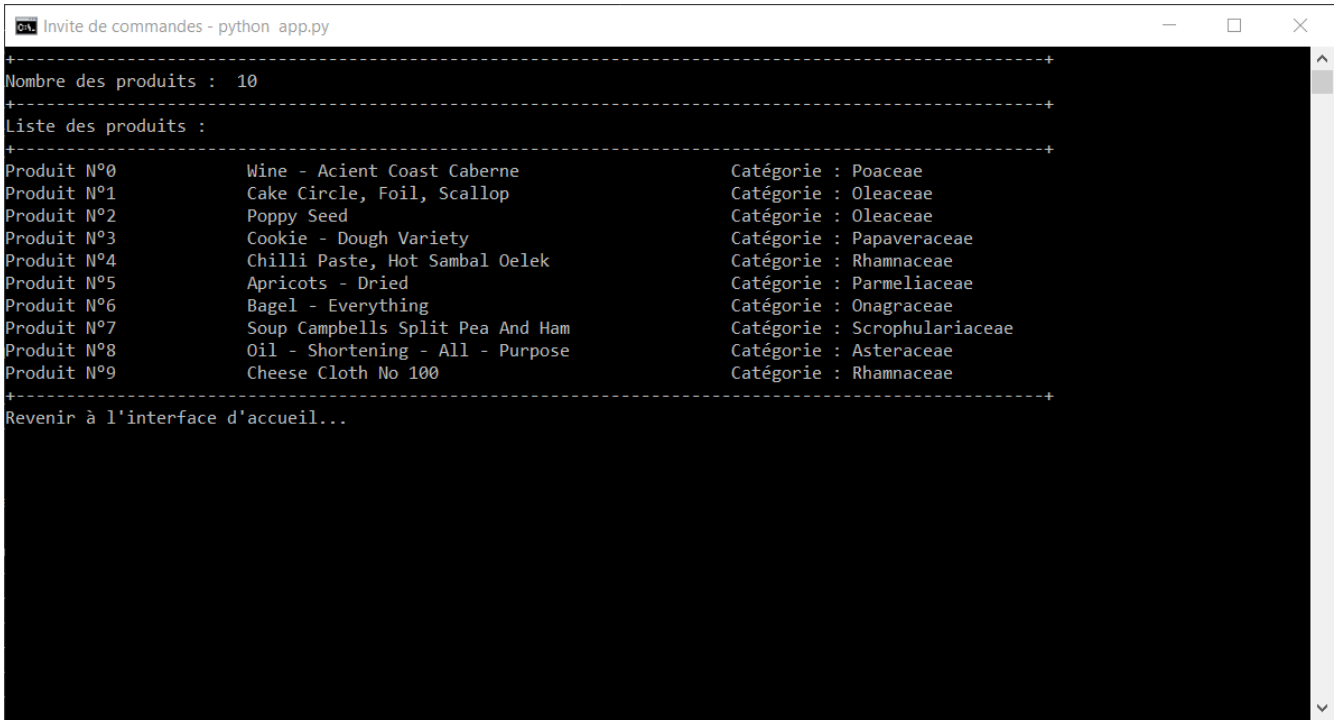

FIGURE 4.6 – Liste des produits

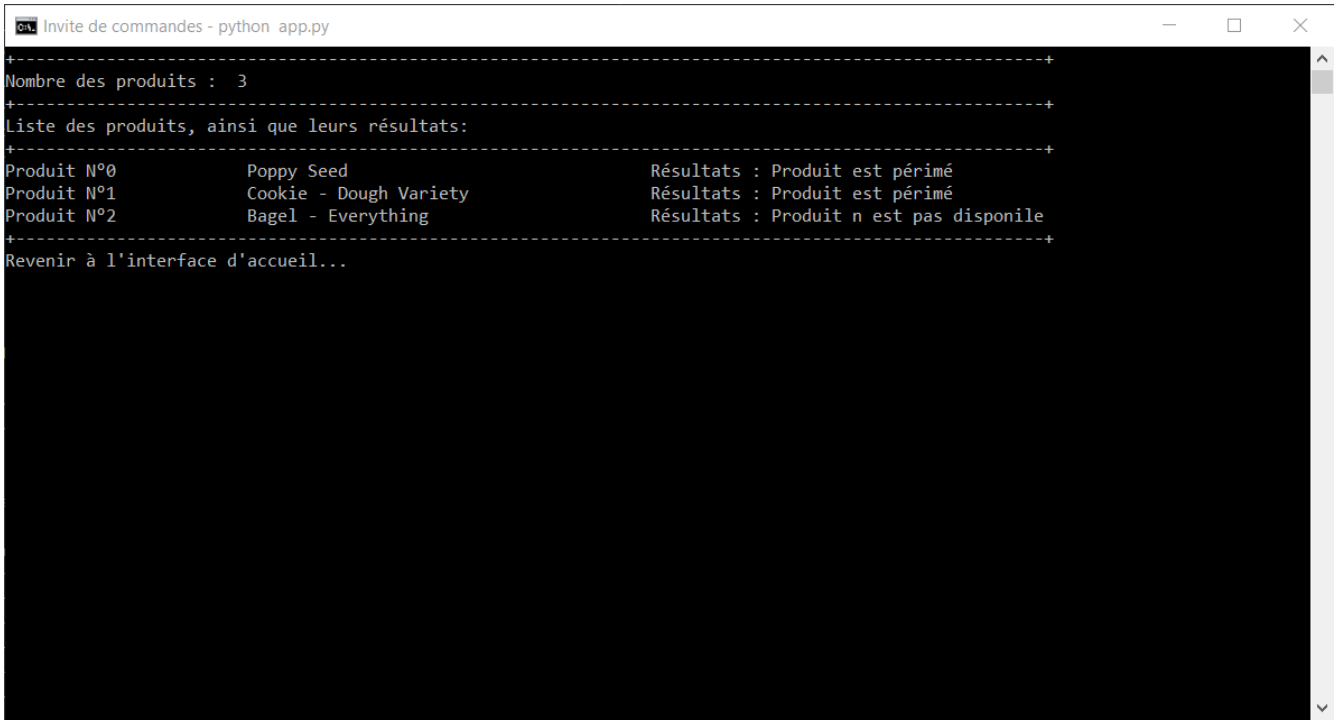

FIGURE 4.7 – Liste des résultats de contrôle

Chapitre 5

Conclusion générale

Les cours de base de données et ce projet nous a été très utile pour bien comprendre tout les notions vues ce semestre : conception et analyse avec création du modèle conceptuel puis traduction en modèle relationnel, et ensuite la réalisation avec la création de la base de données.

Lexpérience était réussie pour tous les membres de léquipe parce que nous étions obligés daller chercher et explorer dautres méthodes et outils pour résoudre notre problème daller plus loin dans lanalyse.

Annexe A

Code Python

Tous les programmes sont disponibles sur la plateforme github.

A.1 Fichier `auth.py`

```
1  import mysql.connector
2  import os
3  import time
4
5  from app.interface import home
6
7
8  connection = mysql.connector.connect(
9      host = 'localhost',
10     port = '3306',
11     database = 'Home',
12     user = 'root',
13     password = 'toor'
14 )
15 cursor = connection.cursor()
16
17 def gain_access(Username=None, Password=None):
18
19     Username = str(input("Entrez votre nom d'utilisateur :\t"))
20     Password = str(input("Entrez votre mot de passe :\t"))
21
22     if not len(Username or Password) < 1:
23         if True:
24             cursor.execute('SELECT user_name, user_password FROM User')
25             data = cursor.fetchall()
26             if (Username, Password) in data :
27                 print("Connexion réussie !")
28                 time.sleep(1)
29                 home(Username)
30             else :
31                 print("Le mot de passe ou le nom d'utilisateur n'existe pas")
32         else:
33             print("Erreur de connexion au système")
34
35     else:
36         print("Veuillez essayer de vous connecter à nouveau.")
37         os.system('cls'); gain_access()
38
39 def register(Username=None, Password1=None, Password2=None):
40     Username = input("Entrez votre nom d'utilisateur :\t")
41     Password1 = input("Entrez votre mot de passe :\t")
42     Password2 = input("Confirmez votre mot de passe :\t")
43
44
45     if not len(Password1) <= 6:
46         if not Username == None:
47             if len(Username) < 1:
48                 print("Veuillez fournir un nom d'utilisateur")
49                 time.sleep(1); os.system('cls');
50                 register()
51             else:
52                 if Password1 == Password2:
53
54                     cursor.execute("insert into user (user_name, user_password, stock_id) values (%s, %s, 2)", (Username, Password1))
55                     connection.commit()
56
57                     print("Utilisateur créé avec succès !")
```

```
58                      print("Veuillez vous connecter pour continuer...")
59                      time.sleep(1); os.system('cls');
60                      gain_access()
61
62                  else:
63                      print("Les mots de passe ne sont pas identiques")
64                      time.sleep(1); os.system('cls');
65                      register()
66      else:
67          print("Mot de passe trop court")
68          time.sleep(1); os.system('cls');
69          register()
```

A.2 Fichier home_app.py

```
1   import mysql.connector
2   import time
3   import os
4
5   connection = mysql.connector.connect(
6       host = 'localhost',
7       port = '3306',
8       database = 'Home',
9       user = 'root',
10      password = 'toor'
11  )
12  cursor = connection.cursor()
13
14  def print_products() :
15      cursor.execute("""
16          SELECT product_name, category_name
17          FROM Product, Category
18          Where Product.category_id = Category.category_id
19      """)
20      data = cursor.fetchall()
21
22      print("Nombre des produits : ", len(data))
23      print("Liste des produits :")
24      i = 0
25      for d in data :
26          print("Produit N°{:<12}\t{:<35}\t\tCatégorie : {:<12}".format(i, d[0], d[1]), end='\n'); i += 1
27
28      if input("Revenir à l'interface d'accueil...") != '' :
29          os.system('cls');
30
31
32  def print_results() :
33      cursor.execute("""
34          SELECT product_name, control_results
35          FROM Product, Control, Stock
36          Where
37                  Control.stock_id = Stock.stock_id AND
38              Stock.product_id = Product.product_id
39          GROUP by product_name
40      """)
41      data = cursor.fetchall()
42
43      print("Nombre des produits : ", len(data))
44      print("Liste des produits, ainsi que leurs résultats:")
45      i = 0
46      for d in data :
47          print("Produit N°{:<12}\t{:<30}\t\tRésultats : {:<12}".format(i, d[0], d[1]), end='\n'); i += 1
48
49      if input("Revenir à l'interface d'accueil...") != '' :
50          os.system('cls');
51
52  def change_password(Username = None) :
53      Password1 = input("Entrez votre mot de passe :\t")
54      Password2 = input("Confirmez votre mot de passe :\t")
55
56      if Password1 == Password2 :
57
58          cursor.execute("""
59              UPDATE User
60              SET
61                  user_password = {}
62              WHERE
63                  user_name = {};
64          """.format(Username, Password1))
65          connection.commit()
66
67          print("Mot de passe changée avec succès !")
68
69      if input("Revenir à l'interface d'accueil...") != '' :
70          os.system('cls');
```

```
71
72  def delete_product() :
73      name = input("Entrez le nom de produit :\t")
74
75      cursor.execute("""
76          DELETE FROM User
77          WHERE product_name = {}
78      """.format(name))
79      connection.commit()
80
81      print("Produit supprimée avec succès !")
82
83
84  def add_new_product() :
85      name = input("Entrez le nom de produit :\t")
86      taille = input("Entrez le libellé de sa taille :\t")
87      temp_min = input("Entrez sa temperature minimal :\t")
88      temp_max = input("Entrez sa temperature maximal :\t")
89      desc = input("Entrez une description de produit :\t")
90
91      cursor.execute("""
92          insert into Product
93              (product_name, product_temperature_max, product_temperature_min, product_description, shape_id, information_id, category_id) values
94              ({}, {}, {}, {}, {}, {}, {});
95      """.format(name, taille, temp_min, temp_max, desc, 1, 1, 1))
96      connection.commit()
97
98
99      print("Mot de passe ajoutée avec succès !")
100
101
102     if input("Revenir à l'interface d'accueil...") != '' :
103         os.system('cls');
```

Annexe B

Structure des tables en MySQL

```
mysql> describe User;
+------------------+-------------+------+-----+---------+----------------+
| Field            | Type        | Null | Key | Default | Extra          |
+------------------+-------------+------+-----+---------+----------------+
| user_id          | int         | NO   | PRI | NULL    | auto_increment |
| stock_id         | int         | NO   | MUL | NULL    |                |
| user_name        | varchar(50) | NO   |     | NULL    |                |
| user_password    | varchar(50) | NO   |     | NULL    |                |
| user_last_session | datetime    | YES  |     | NULL    |                |
+------------------+-------------+------+-----+---------+----------------+
5 rows in set (0.08 sec)

mysql> describe product;
+-------------------------+-------------+------+-----+---------+----------------+
| Field                   | Type        | Null | Key | Default | Extra          |
+-------------------------+-------------+------+-----+---------+----------------+
| product_id              | int         | NO   | PRI | NULL    | auto_increment |
| shape_id                | int         | NO   | MUL | NULL    |                |
| information_id          | int         | NO   | MUL | NULL    |                |
| category_id             | int         | NO   | MUL | NULL    |                |
| product_name            | varchar(50) | NO   |     | NULL    |                |
| product_temperature_max | varchar(50) | NO   |     | NULL    |                |
| product_temperature_min | varchar(50) | NO   |     | NULL    |                |
| product_description     | varchar(50) | NO   |     | NULL    |                |
+-------------------------+-------------+------+-----+---------+----------------+
8 rows in set (0.24 sec)

mysql> describe category;
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra          |
+---------------+-------------+------+-----+---------+----------------+
| category_id   | int         | NO   | PRI | NULL    | auto_increment |
| category_name | varchar(50) | NO   |     | NULL    |                |
+---------------+-------------+------+-----+---------+----------------+
2 rows in set (0.11 sec)

mysql> describe shape;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| shape_id     | int         | NO   | PRI | NULL    | auto_increment |
| shape_name   | varchar(50) | NO   |     | NULL    |                |
| shape_height | varchar(50) | NO   |     | NULL    |                |
| shape_weight | varchar(50) | NO   |     | NULL    |                |
| shape_volume | varchar(50) | NO   |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
5 rows in set (0.04 sec)

mysql> describe information;
+-----------------------------+----------+------+-----+---------+----------------+
| Field                       | Type     | Null | Key | Default | Extra          |
+-----------------------------+----------+------+-----+---------+----------------+
| information_id              | int      | NO   | PRI | NULL    | auto_increment |
| information_expiration_date | datetime | NO   |     | NULL    |                |
| information_entry_date      | datetime | NO   |     | NULL    |                |
| information_unit_price      | int      | NO   |     | NULL    |                |
| information_quantity        | int      | NO   |     | NULL    |                |
+-----------------------------+----------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql> describe company;
+--------------------+-------------+------+-----+---------+----------------+
| Field              | Type        | Null | Key | Default | Extra          |
+--------------------+-------------+------+-----+---------+----------------+
```

```
| company_id          | int         | NO   | PRI | NULL    | auto_increment |
| product_id          | int         | NO   | MUL | NULL    |                |
| company_name        | varchar(50) | NO   |     | NULL    |                |
| company_email       | varchar(50) | NO   |     | NULL    |                |
| company_description | varchar(50) | NO   |     | NULL    |                |
+---------------------+-------------+------+-----+---------+----------------+
5 rows in set (0.02 sec)

mysql> describe city;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| zip_code  | int         | NO   | PRI | NULL    | auto_increment |
| city_name | varchar(50) | NO   |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)

mysql> describe control;
+----------------+-----------------------------------------------------------------------------------------------+------+-----+---------+----------------+
| Field          | Type                                                                                          | Null | Key | Default | Extra          |
+----------------+-----------------------------------------------------------------------------------------------+------+-----+---------+----------------+
| control_id     | int                                                                                           | NO   | PRI | NULL    | auto_increment |
| stock_id       | int                                                                                           | NO   | MUL | NULL    |                |
| control_date   | datetime                                                                                      | NO   |     | NULL    |                |
| control_results | enum('Produit est périmé','Produit est disponible','Produit n est pas disponile') | YES  |     | NULL    |                |
+----------------+-----------------------------------------------------------------------------------------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> describe belonging;
+------------+------+------+-----+---------+-------+
| Field      | Type | Null | Key | Default | Extra |
+------------+------+------+-----+---------+-------+
| zip_code   | int  | NO   | PRI | NULL    |       |
| company_id | int  | NO   | PRI | NULL    |       |
+------------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```