**TITRE PROFESSIONNEL**

**Concepteur Développeur d’Application**

**Niveau II**

Dossier de projet

Lilian LAYRAC

EPSI

École d'ingénierie informatique

**TABLE DES MATIERES**

[Introduction 5](#_Toc135108173)

[1 - Référentiel de compétences 6](#_Toc135108174)

[2 - Résumé des projets 7](#_Toc135108175)

[2.1 - Introduction 7](#_Toc135108176)

[2.2 – Projet IAPM | StoreSwap 7](#_Toc135108177)

[2.3 – Projet Mobile 7](#_Toc135108178)

[3 – Cahier des charges 8](#_Toc135108179)

[3.1 – Cahier des charges pour IAPM | StoreSwap 8](#_Toc135108180)

[3.1.1 – Description du cahier des charges du projet 8](#_Toc135108181)

[3.1.2 – Utilisateurs des applications 8](#_Toc135108182)

[3.1.3 – Tests et validation 9](#_Toc135108183)

[3.2 – Cahier des charges pour Mobile 9](#_Toc135108184)

[4 – Gestion de projet 10](#_Toc135108185)

[4.1 – Gestion de projet IAPM | StoreSwap 10](#_Toc135108186)

[4.2 – Gestion de projet Mobile 12](#_Toc135108187)

[5 – Spécifications fonctionnelles 13](#_Toc135108188)

[5.1 – Application web StoreSwap 13](#_Toc135108189)

[5.2 – Application desktop IAPM 15](#_Toc135108190)

[6 – Spécifications techniques 16](#_Toc135108191)

[6.1 – Application web StoreSwap 16](#_Toc135108192)

[6.2 – Application desktop IAPM 16](#_Toc135108193)

[2.5 Développer des composants dans le langage d’une base de données 32](#_Toc135108194)

[4.3 Concevoir une base de données 50](#_Toc135108195)

[4.4 Mettre en place une base de données 51](#_Toc135108196)

[4.5 Développer des composants dans le langage d'une base de données 52](#_Toc135108197)

[4.6 Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement 53](#_Toc135108198)

[4.7 Concevoir une application 54](#_Toc135108199)

[4.8 Développer des composants métier 55](#_Toc135108200)

[4.9 Construire une application organisée en couches 56](#_Toc135108201)

[4.9.1 Développer une application mobile 57](#_Toc135108202)

[4.9.2 Préparer et exécuter le déploiement d'une application 58](#_Toc135108203)

[5.1 Développer une interface utilisateur de type desktop 61](#_Toc135108204)

[5.2 Développer la partie back-end d'une interface utilisateur web 62](#_Toc135108205)

[5.3 Préparer et exécuter les plans de tests d'une application 63](#_Toc135108206)

[6.1 Maquetter une application 65](#_Toc135108207)

[6.2 Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement 66](#_Toc135108208)

[7.1 Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement 68](#_Toc135108209)

[7.2 Concevoir une application 69](#_Toc135108210)

[X Conclusion 70](#_Toc135108211)

[X Glossaire 71](#_Toc135108212)

[X Annexes 72](#_Toc135108213)

[Annexes pour IAPM 72](#_Toc135108214)

[Annexe IAPM Maquette 72](#_Toc135108215)

[Annexe IAPM Charte Graphique 73](#_Toc135108216)

[Annexe IAPM Connexion 73](#_Toc135108217)

[Annexe IAPM Procédure Stockée 74](#_Toc135108218)

[Annexe IAPM Trigger 74](#_Toc135108219)

[Annexe IAPM Résultat Trigger 75](#_Toc135108220)

[Annexe IAPM Tests Unitaires 75](#_Toc135108221)

[Annexes pour StoreSwap 75](#_Toc135108222)

[Annexe StoreSwap Accueil 75](#_Toc135108223)

[Annexe StoreSwap Connexion 76](#_Toc135108224)

[Annexe StoreSwap Inscription 76](#_Toc135108225)

[Annexe StoreSwap Boutique 77](#_Toc135108226)

[Annexe StoreSwap Panier 77](#_Toc135108227)

[Annexe StoreSwap Admin Table 78](#_Toc135108228)

[Annexe StoreSwap Admin Table Client 78](#_Toc135108229)

[Annexes pour TentationVoyage 79](#_Toc135108230)

[Annexes pour MSPR A’rosa-je 79](#_Toc135108231)

[Annexes pour PriSearch 79](#_Toc135108232)

[Annexes pour AirSens 79](#_Toc135108233)

[Annexes pour VITAM 79](#_Toc135108234)

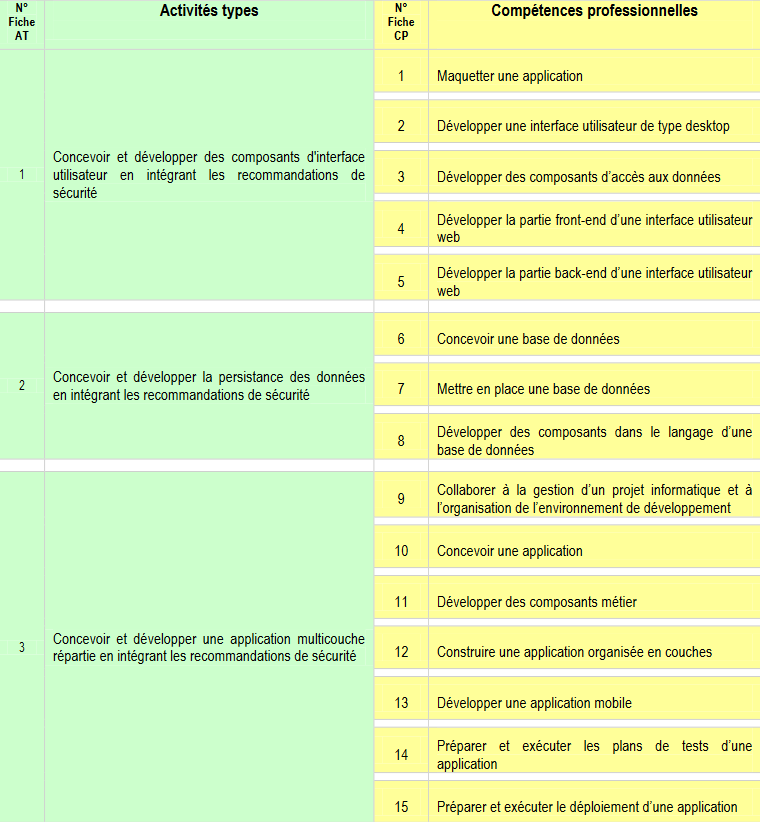
Introduction

Je me présente, Lilian Layrac, étudiant à l’EPSI pour le passage de mon Bachelor CDA (Concepteur développeur d’applications). Mon parcours professionnel se traduit par le passage d’un Baccalauréat STMG avec option SIG (Système d’Information et de Gestion) dans le lycée Marc Bloch, à Sérignan. Je l’ai obtenu avec mention assez-bien. Je me suis ensuite tourné vers quelque chose qui me passionne, l’informatique, en introduisant un cursus BTS SIO (Services informatiques aux organisations), dans ce même lycée. Durant ce cursus de deux ans, j’ai appris les bases du développement informatique, de pars la conception de site web, mais d’autre part, via conceptions de clients lourds comme des applications métiers. Au bout d’un an de formation, j’ai choisi l’option SLAM (Solutions logicielles et applications métiers), car je suis plus attiré par le développement. À la suite de l’obtention de mon BTS, je me suis tourné vers l’EPSI pour effectuer un Bachelor CDA, en suivant un cursus d’alternance.

Mon alternance se déroule dans une petite structure nommée Optim.Solutions (anciennement appelé Algo.Solutions). Nous nous occupons d’intégrer une plateforme open-source nommé VITAM (Programme interministériel archivage numérique). Notre objectif est de déployer cette infrastructure, en proposant le déploiement de cette dernière (qui est très fastidieux), mais aussi de créer des portails de connexions pour y accéder. Nous assurerons également la maintenance de la solution si mise à jour il y a.

Les projets que je vais présenter lors de ce dossier sont des projets personnel et professionnel, ils ont pour but de valider le référentiel de compétences. Malheureusement, les projets en entreprise ne me permettent pas de valider les compétences, c’est pour cela que je me tourne vers des projets personnel et professionnel réalisé en amont.

1 - Référentiel de compétences



2 - Résumé des projets

2.1 - Introduction

Firstly, to validate the title of Designer Application Developer, I need to present you some projects. These projects validate all the competences I showed you above. They are made by different languages of programming, like PHP, HTML/CSS for the front-end, Visual Basic for the back-end, and Dart (with the framework Flutter) for the mobile app.

Our fake company name is Layweb. I’m gona use this name during all the project to contextualize it as a professional project.

2.2 – Projet IAPM | StoreSwap

The first project is called IAPM | Storeswap, it’s a project with a web app (StoreSwap), administered by the desktop app (IAPM). It’s a marketplace web app where you can buy candies. Obviously, I will go deeper into the explanations further into the project. The desktop app IAPM can administrate the database by handle entities and their content. Both apps have the same database. I created a MySQL database managed by a WAMP server, all my work was made in local.

These applications were made by myself, but otherwise, I managed to make them into a professional scope, by creating workflows like Conceptual Data Model, or some Schedule Management.

2.3 – Projet Mobile

The second project is a mobile app ………………………………..

3 – Cahier des charges

3.1 – Cahier des charges pour IAPM | StoreSwap

3.1.1 – Description du cahier des charges du projet

La demande de notre client était de créer une application permettant la vente de bonbons. Cette dernière étant composé d’une interface web, ainsi qu’une interface desktop.

Concernant l’application web StoreSwap, elle est développée en PHP, HTML et CSS, elle gère la vitrine de l’activité Layweb. L’application desktop IAPM**,** quant-à-elle, gère tous les éléments de la base de données, elle gère donc les stocks et les utilisateurs.

Concernant l’entreprise Layweb, elle aimerait expandre son secteur d’activité, actuellement limité à un simple petit magasin de centre-ville, en créant leur propre marketplace, pour ainsi vendre des bonbons dans toute la France, l’Europe, et pourquoi pas le monde entier.

Mais pour développer une activité en ligne florissante, il faut être rigoureux et bien réfléchir à son architecture ou même sa logistique. C’est pour cela que l’entreprise Layweb a fait appel à nos services pour lui créer une interface web StoreSwap, agrémenté de son progiciel de gestion de base de données IAPM, pour faciliter les insertions de nouveaux produits ou la gestion des clients.

Nous nous occupons également des problématiques de sécurité qui ne sont pas à négliger quand nous traitons des données sensibles comme des objets à caractère personnel des comptes utilisateur par exemple.

Les besoins de l’entreprise sont :

- Avoir un site web permettant la navigation des clients

- Avoir une interface progicielle permettant la manipulation des données

- Avoir un contrôle total sur les produits, utilisateurs et commandes

Ce type d’application est totalement adapté à l’activité de l’entreprise Layweb, car elle répond parfaitement à leur attente.

A noter que le client peut ajouter de nouvelles fonctionnalités tout au long du projet, ou même ultérieurement après le livrable de ce dernier (car nous assurons la maintenance de toutes nos applications livrées). Une veille technologie est primordiale en termes de sécurité car il faut constamment corriger des failles.

3.1.2 – Utilisateurs des applications

Les différents utilisateurs des applications sont :

- pour la partie web, les clients, les administrateurs

- pour la partie desktop, les administrateurs

Un client est un utilisateur de l’application, il peut acheter des bonbons, consulter le site, se balader librement dessus mais n’a aucun droit sur l’édition de la base de données, si ce n’est la création, modification ou suppression de son propre compte utilisateur, ou la gestion de son panier. A savoir que le client navigue seulement sur le front-end StoreSwap, et ne touche jamais à l’application IAPM.

Un administrateur, coté web, peut effectuer les mêmes actions que le client, mais il peut également consulter les tables de la base de données sur sa partie administration (intégrée à l’interface web).

Un administrateur, coté desktop, peut consulter les clients, en ajouter, modifier ou supprimer. Il peut également consulter les produits, en ajouter, modifier ou supprimer. Il peut également consulter les commandes, en ajouter, modifier ou supprimer. Il a également une interface dashboard qui permet de consulter le nombre de clients, de produits et de commandes en temps réel.

3.1.3 – Tests et validation

Les spécifications démontrées ci-dessus ont été validée par l’entreprise Layweb, car elles correspondent parfaitement aux attentes du client.

3.2 – Cahier des charges pour Mobile

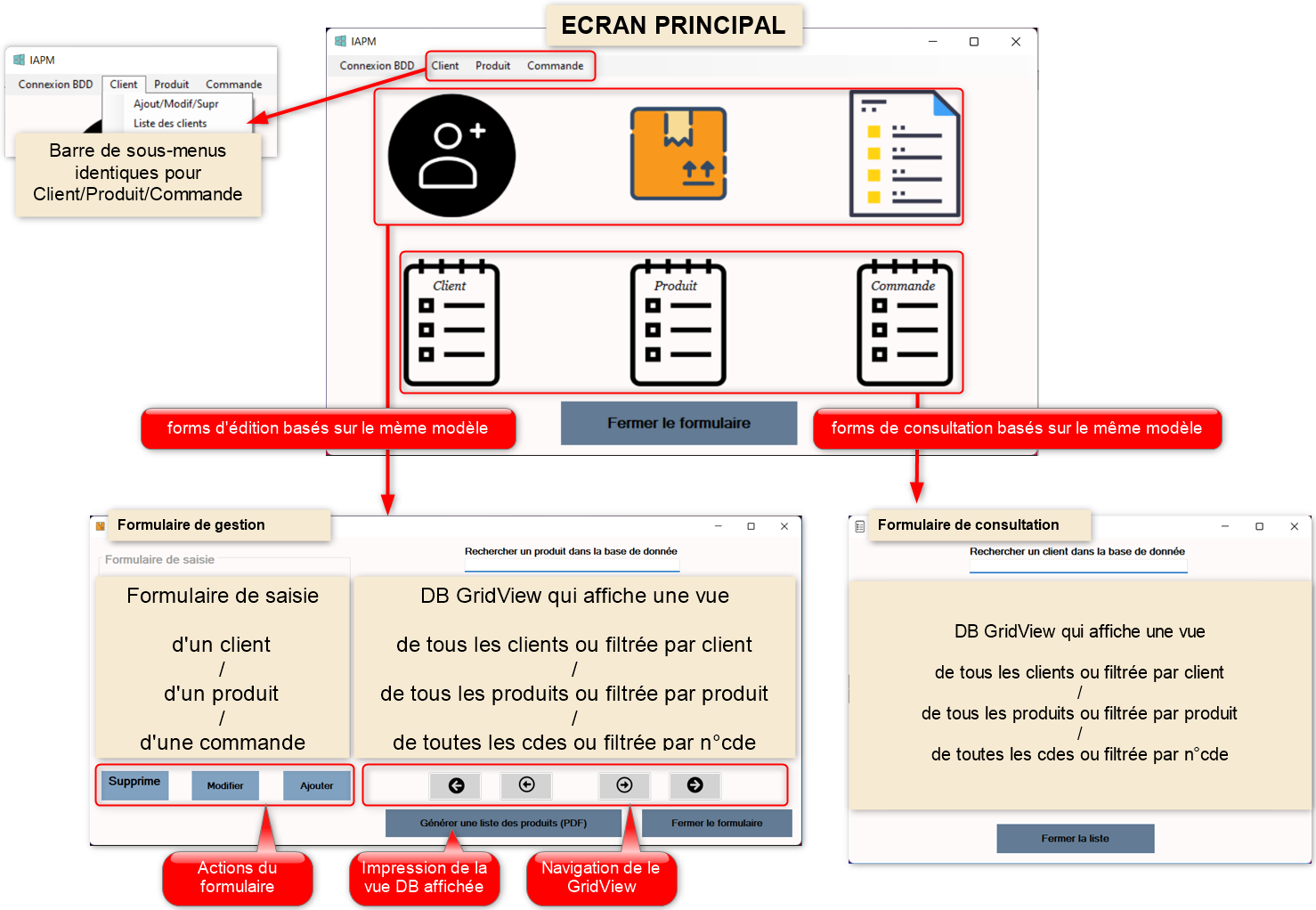
# 4 – Gestion de projet

## 4.1 – Gestion de projet IAPM | StoreSwap

Malgré le cas professionnel fictif de ce projet, j’ai tout de même décidé de travailler avec des **outils collaboratifs** telle que GitHub, pour garder un système de versionning et garder une certaine cohérence dans mes commits et ma gestion du code. Grâce à cela, j’ai réussi à rester organiser et surtout à ne pas me perdre dans la masse de travail importante.

J’ai également réalisé une **maquette** de l’application desktop IAPM via l’application en ligne Figma, pour avoir une idée précise de ce que je voulais comme style d’application avant de me lancer dans la production de cette dernière.

Voici un **StoryBoard** qui explique chaque item de l’application :



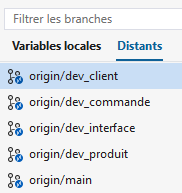
*(Pour plus d’annexes, voir Annexes pour IAPM – Annexe IAPM Maquette)*

J’ai également intégré la notion de **charte graphique** pour avoir des couleurs cohérentes en fonction du style que je voulais donner à l’application. Elle a également été réalisée sur Figma.

*(Voir Annexes pour IAPM – Annexe IAPM Charte Graphiques)*

J’ai utilisé **GitHub** pour ma gestion de projet, et j’ai donc créer différentes branches me permettant de m’organiser lors du développement de l’application IAPM Desktop.

Voici les différentes **branches** créées disponible via l’explorateur de branches sur l’éditeur Microsoft Visual Studio :



Nous avons la branche dev\_client, qui traite tout le développement en rapport avec la gestion des clients, la branche dev\_commande, qui traite tout le développement en rapport avec la gestion des commandes, la branche dev\_interface, qui traite tout le développement en rapport avec la gestion de l’interface graphique, la branche dev\_produit, qui traite tout le développement en rapport avec la gestion des produits, et enfin la branche main ou l’on merge le résultat final et où l’on règle les conflits pour que tout soir réuni sur une seule et même branche.

Dans un cadre professionnel, il aurait été préférable de réaliser un diagramme de Gantt. Je n’ai pas jugé nécessaire d’en créer un, car j’étais seul sur le projet et donc seul à effectuer les tâches.

A noter que si le projet avait été en groupe, il aurait été préférable de faire un diagramme de Gantt, car cela nous permet d’avoir un suivi sur l’avancement du projet, et surtout planifier des correctifs si délais il y a (Il y a d’autres digrammes de gestion de projet comme le diagramme de PERT, qui permet de représenter graphiquement les tâches à accomplir, mais aussi d’y intégrer la notion temporelle, ce qui sert donc à planifier, organiser et contrôler les différentes étapes d’un projet).

J’ai également effectué de tests unitaires pour l’application desktop IAPM pour tester mes méthodes tels que AjouterClient ou ModifierClient.

*(Voir Annexes pour IAPM – Annexe IAPM Tests Unitaires)*

Pour la partie front-end, je n’ai pas réalisé de maquettage, ni de gestion de projet avec GitHub. J’ai simplement posé les idées et coder en fonction de spécificités fonctionnelles et techniques énoncées ci-après dans les rubriques « Spécifications fonctionnelles » et « Spécifications techniques ».

## 4.2 – Gestion de projet Mobile

# 5 – Spécifications fonctionnelles

## 5.1 – Application web StoreSwap

Voici les spécificités de l’application web StoreSwap :

- Le client peut accéder à la page d’accueil

- Le client peut accéder à la page boutique

- Le client peut accéder à son espace personnel via l’onglet Mon Espace (seulement après s’être connecter)

- Le client peut consulter son panier via l’icône associé

- Le client peut consulter les produits les plus commandés

- Le client peut cliquer sur le bouton Votre Feedback

- Le client peut donner un Feedback en remplissant les champs Prénom, Email, Avis

- L’utilisateur peut créer un compte pour devenir client

- Si l’utilisateur n’a pas de compte, il ne peut pas accéder à la boutique

- Si l’utilisateur n’a pas de compte et qu’il tente de se connecter, il est renvoyé vers le formulaire d’inscription

- Le client peut se connecter via le bouton de connexion

- Si un utilisateur s’inscrit sur le site, il devient client

- Le client peut ajouter un article à son panier

- Le client peut ajouter plusieurs articles à son panier

- Le client peut supprimer un article de son panier

- Le client peut supprimer tout son panier d’un seul coup

- Un administrateur peut effectuer toutes les actions d’un client/utilisateur

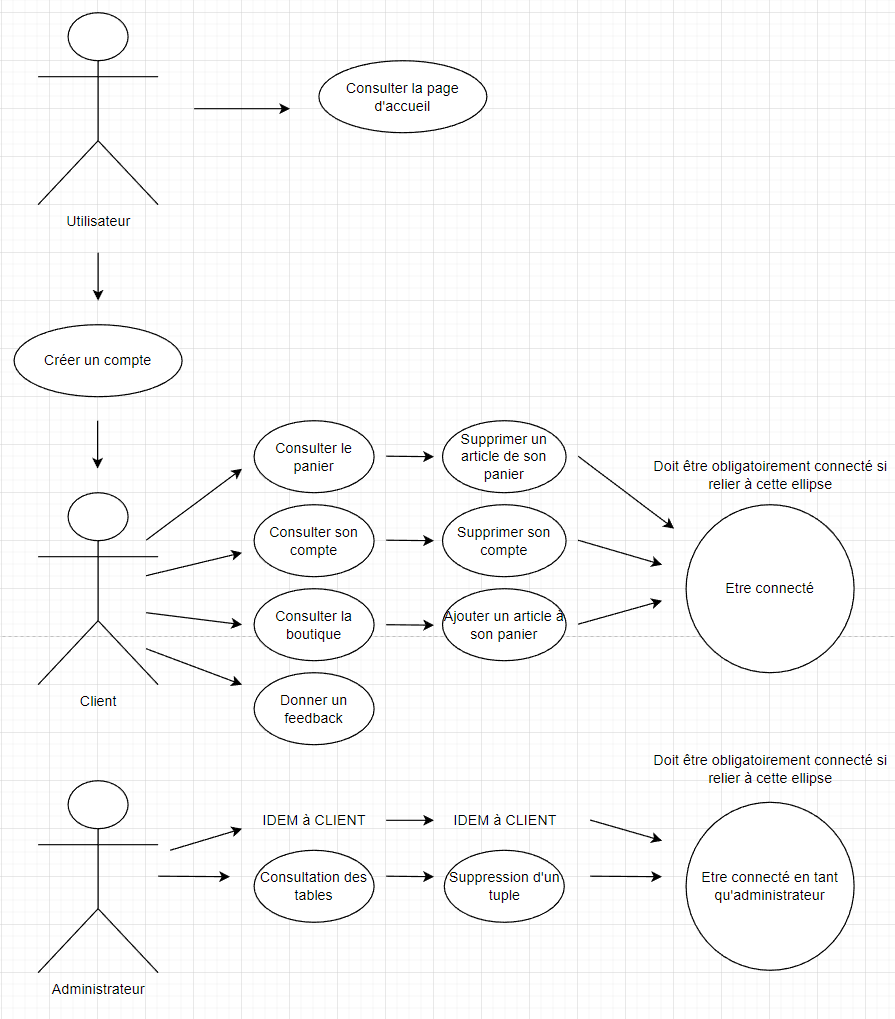
- Un administrateur peut cliquer sur l’icône de connexion pour les administrateurs pour s’y connecter

- Un administrateur peut consulter les différentes tables via l’interface d’administration

- Un administrateur peut supprimer un tuple via l’icône corbeille quand il clique sur une table dans l’interface administration

- Un administrateur peut de nouveau afficher toutes les tables quand il a affiché seulement une seule table

Je vais vous présenter ci-dessous le Diagramme UML (Unified Modeling Language). Ce diagramme constitue une représentation visuelle du système, permettant de mieux comprendre son architecture, ses composants et les interactions entre eux. Grâce à ce diagramme, vous pourrez avoir une vision globale de la structure de l’application, des différentes classes qui la composent, ainsi que des relations et des dépendances entre ces classes. En examinant attentivement le Diagramme UML de l’application StoreSwap, vous serez en mesure de saisir plus facilement les fonctionnalités clés de l’application et d’appréhender la manière dont les différents éléments interagissent pour offrir une expérience utilisateur optimale. A savoir le que le Diagramme UML reprend les spécifications fonctionnelles énoncées ci-dessus.

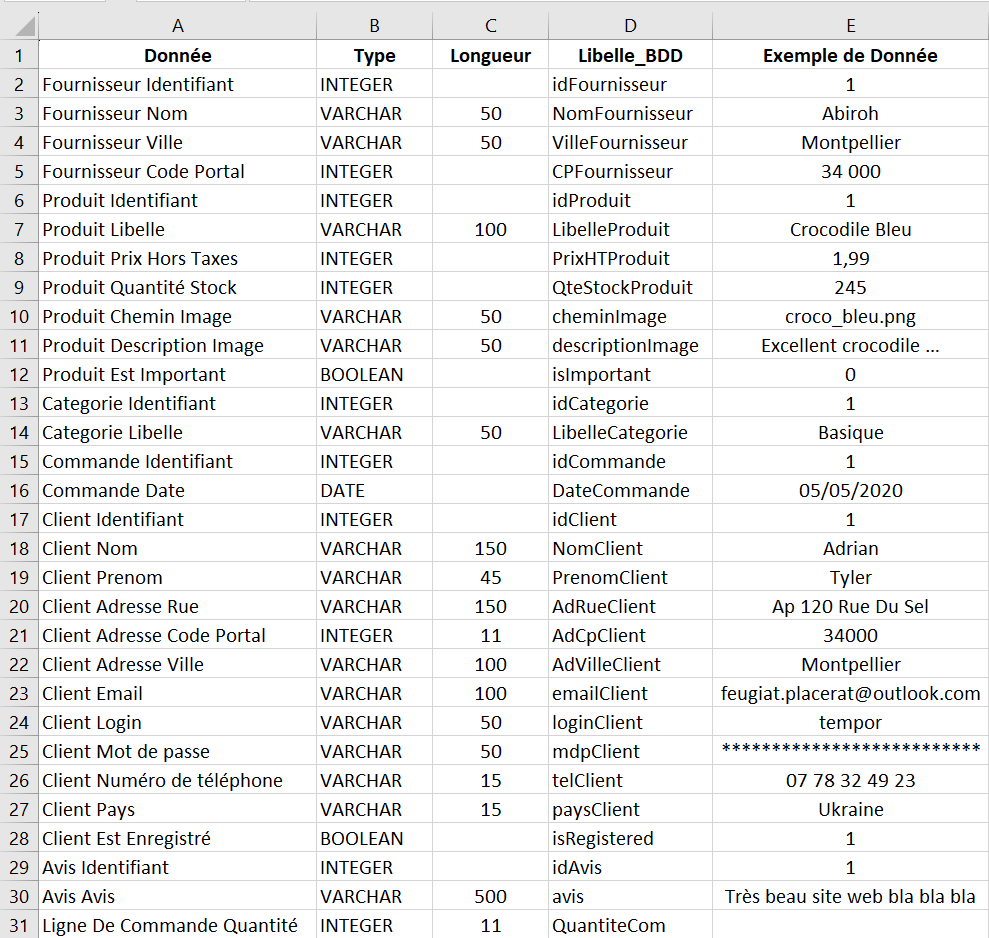


Ce diagramme UML démontre les cas d’utilisation énoncé ci-dessus en fonction de chaque utilisateur pour l’application web StoreSwap.

Dans le cadre de la présentation de mon application web StoreSwap, il est essentiel de comprendre la structure et l’organisation du système. Pour cela, je vais vous présenter le Diagramme des classes, un outil visuel qui représente les différentes classes du programme, ainsi que leurs relations et interactions. Ce diagramme sera un guide précieux pour mieux appréhender la conception et le fonctionnement de l’application, en mettant en évidence les entités principales, les attributs et les méthodes de chaque classe. Grâce à cette représentation claire et synthétique, vous pourrez aisément visualiser les différents composants de l’application StoreSwap et comprendre comment ils interagissent pour offrir des fonctionnalités performantes et intuitives.

DIAGRAMME DES CLASSES

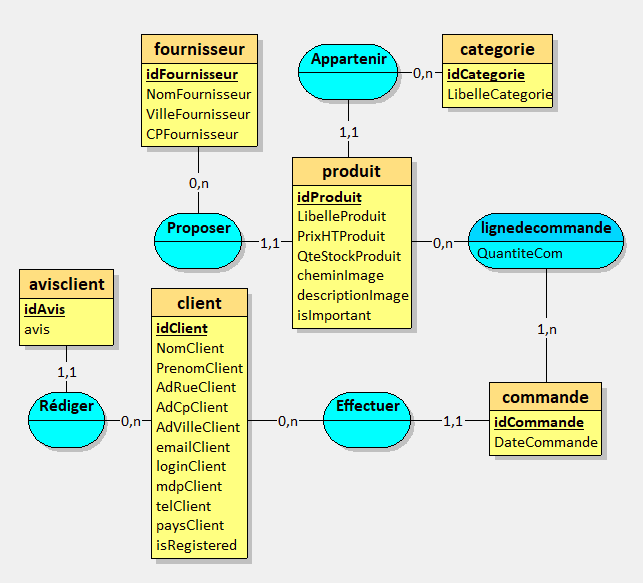
A présent, il est primordial de présenter le dictionnaire de donnée pour avoir une vue précise sur les différentes données manipulées. Ce précieux outil recense et organise toutes les informations qui sont stockés et manipulées par l’application. Le dictionnaire de données constitue une ressource essentielle pour comprendre la structure, le contenu et les relations entre les différentes entités présentes dans l’application StoreSwap. Il vous permettra d’avoir une vue d’ensemble claire et détaillée des différentes tables, champs, types de données et contraintes utilisé dans la gestion des informations. Grâce à cette présentation, vous pourrez apprécier la rigueur et l’efficacité de la gestion de données de StoreSwap, ainsi que la manière dont elles sont organisées pour assurer un fonctionnement optimal de l’application.



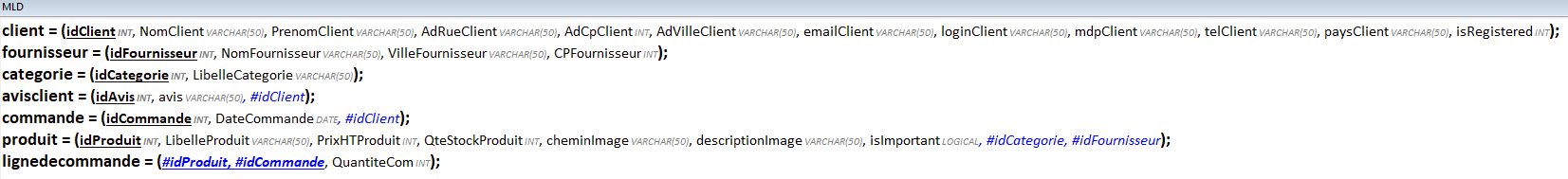
Maintenant, je vais vous présenter le Modèle Conceptuel de Données. Il a été réalisé sur l’éditeur de MCD Looping MCD, cet éditeur nous permet également de générer le MLD (Modèle Logique de Données), mais aussi le script de base de données pour l’intégrer dans l’interpréteur de script MySQL et donc générer notre base de données.

Ce modèle est une représentation abstraite et visuelle des concepts et des relations qui existent au sein de l’application. Le MCD de StoreSwap met en évidence les entités principales, les attributs et les associations entre ces entités, offrant ainsi une vue claire de la structure et du fonctionnement de la base de données. Les entités représentent les objets distincts, les relations indiquent les liens entre ces objets, et les cardinalités spécifient le nombre d’occurrences d’une entité dans une relation.

En examinant attentivement le MCD, vous pourrez comprendre comment les différentes entités sont liées entre elles. Le MCD constitue donc un outil essential pour comprendre la logique sous-jacente de StoreSwap et apprécier la manière dont les données sont modélisées pour répondre aux besoins de l’application.



Avec l’application LoopingMCD, j’ai également généré un Modèle Logique de Données, les clés primaires sont en gras soulignées, les clés étrangères sont précédées d’un # et en bleu, en voici l’image :



Voici les conventions de nommages que j’ai utilisé pour la génération de la base de données, ainsi que le développement de l’application StoreSwap.

* Thème globale 🡪 Commentaires en anglais si possible, ainsi que le code
* Les classes sont en lowercase 🡪 produit
* Les variables sont en lowercase 🡪 text
* Les méthodes adoptent le UpperCamelCase et doivent avoir une signification simple et verbeuse 🡪 DeleteProduit(), GetProduitByCommande()

A savoir que ces conventions son appliquées à la base de données, à l’application web StoreSwap mais aussi à l’application desktop IAPM.

En plus du StoryBoard présenté ci-dessus dans la rubrique Gestion de projet IAPM, j’ai réalisé des maquettes disponibles en annexe.

*(Voir Annexes pour IAPM – Annexe IAPM Maquette)*

## 5.2 – Application desktop IAPM

Voici les spécificités de l’application desktop IAPM

- Un administrateur peut accéder au menu

- Un administrateur peut connecter le progiciel à une base de données précise via un menu déroulant

- Un administrateur peut ouvrir l’onglet Client via un menu déroulant (Idem pour produits et commandes)

- Un administrateur peut accéder à l’administration des clients via le bouton « Ajout/Modif/Supr » dans le menu déroulant Client (Idem pour produits et commandes)

- Un administrateur peut accéder à la visualisation de la liste des clients via le bouton « Liste des clients » dans le menu déroulant Client (Idem pour produits et commandes)

- Un administrateur peut fermer l’application via le bouton « Fermer le formulaire »

- Un administrateur peut ajouter un client via le bouton « Ajouter »

- Un administrateur peut modifier un client via le bouton « Modifier »

- Un administrateur peut supprimer un client via le bouton « Supprimer »

- Un administrateur peut naviguer entre les différents clients à l’aide des boutons fléchés

- Un administrateur peut rechercher un client via la barre de recherche

- Un administrateur peut ajouter un produit via le bouton « Ajouter »

- Un administrateur peut modifier un produit via le bouton « Modifier »

- Un administrateur peut supprimer un produit via le bouton « Supprimer »

- Un administrateur peut naviguer entre les différents produits à l’aide des boutons fléchés

- Un administrateur peut rechercher un produit via la barre de recherche

- Un administrateur peut ajouter une commande via le bouton « Ajouter »

- Un administrateur peut modifier une commande via le bouton « Modifier »

- Un administrateur peut supprimer une commande via le bouton « Supprimer »

- Un administrateur peut naviguer entre les différentes commandes à l’aide des boutons fléchés

Ci-après, je vais présenter la base de données via des captures d’écrans et en y apportant des explications.

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquementVoici l’arborescence de la base de données, on y retrouve les procédures stockées, les déclencheurs ainsi que les tables.

Ci-dessous, voici la table produit :

Une image contenant texte, nombre, Police, logiciel

Description générée automatiquement

J’ai également développé des procédures stockées ainsi que des déclencheurs, vous pouvez les retrouver en annexe.

*(Voir Annexes pour IAPM – Annexe IAPM BDD)*

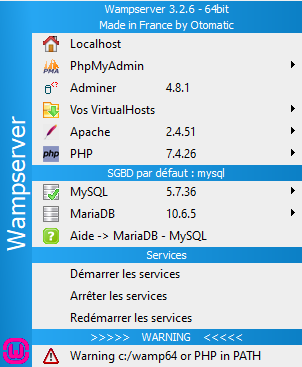
Ces procédures stockés et déclencheurs sont appelé dans l’application desktop IAPM pour récupérer les clients, les produits ou les commandes.

## 5.3 – Application Mobile

# 6 – Spécifications techniques

## 6.1 – Application web StoreSwap

Elaboration des contraintes techniques :

L’éditeur de code utilisé est **Visual Studio Code**, car je le trouve ergonomique et simple d’utilisation. On peut également installer énormément de plugins pour module l’IDE « à sa sauce », et c’est cela qui en fait sa force.

Pour que l’application communique avec la base de données, j’ai utilisé un serveur **WAMP** pour pouvoir travailler en local.

J’ai utilisé la version 7.4.26 de PHP, ainsi que la version 5.7.36 de MySQL.

Pour, la conception des diagrammes UML, j’ai utilisé diagrams.net.

Pour la conception du MCD, j’ai utilisé LoopingMCD, il m’a également généré le MLD.

Pour la plateforme, j’ai choisi le langage **PHP** pour le développement ainsi que **HTML 5** et **CSS 3**. Pour le design, j’ai utilisé le **framework CSS Bootstrap 5**. La base de données est en langage MySQL.

Pour l’architecture, j’ai utilisé le modèle en **couche MVC**, qui permet d’avoir un contrôle total sur les modèles, les vues, ainsi que les contrôleurs de l’application web. Le serveur tourne sur le protocole de base https, qui permet une certaine sécurité lors de la navigation. Les politiques de gestion de session sont respectés.

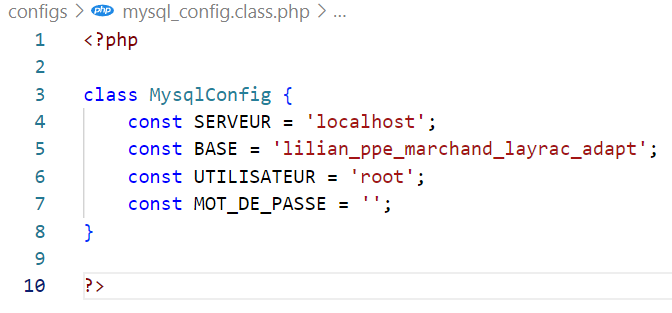
Les exigences de **l’interface utilisateur** sont que l’interface soit intuitive et facile de compréhension pour l’utilisateur. Il doit aller d’un point A à un point B simplement et efficacement sans se perdre sur le site, sinon il risque de le quitter et donc arrêter la navigation.

**L’aspect sécurité** est traité par le biais de hachage du mot de passe lors de l’inscription de l’utilisateur. Un système de cookie de session est également utilisé pour garder la connexion de l’utilisateur, il respecte les spécifications de durée.

Dans le contexte fictif, l’entreprise Layweb assurera la maintenance de l’application si nécessaire et si amélioration est demandé par le client, cependant, elle vous sera facturée.

Composants d’accès aux données :

Pour le projet StoreSwap, pour effectuer la connexion à la base de données, j’ai créé la classe MysqlConfig, qui s’occupe d’affecter les informations de connexion aux variable appelées dans la fonction de connexion seConnecter. Voici le code de la classe MysqlConfig :



La variable **SERVER** correspond à l’IP du serveur MySQL cible,

La variable **BASE** correspond au nom de la base de données cible,

La variable **UTILISATEUR** correspond au nom d’utilisateur permettant de se connecter à la base de données,

La variable **MOT\_DE\_PASSE** correspond au mot de passe permettant de se connecter à la base de données.

Une image contenant texte, capture d’écran, Police, nombre

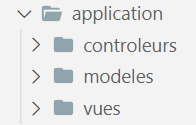
Description générée automatiquementCes variables seront utilisées dans la fonction seConnecter qui se situe dans la classe ModelePDO

Dans cette classe, je redéfinis les variables stockées dans la classe MysqlConfig, ensuite, dans ma fonction seConnecter, je vérifie si la connexion à déjà été établie, si oui, la méthode ne se réexécute pas, cependant, s’il n’y a pas de connexion établie, j’établie ma chaîne de connexion avec les informations stockées dans les variables. La connexion s’effectue à l’aide de l’extension PDO.

Cette fonction sera appelée dans quand vues pour avoir constamment accès aux données.

La fonction seDeconnecter met simplement la variable pdoCnxBase à NULL, ce qui annule donc la connexion à la base de données.

Principe de développement Modèle Vue Contrôleurs :



Le modèle vue contrôleur (MVC) est un principe de conception largement utilisé dans le développement d'applications web, y compris dans le cas de StoreSwap. Il divise l'application en trois composants principaux : le modèle, la vue et le contrôleur.

Le modèle représente les données et la logique métier de l'application. Il est responsable de la manipulation des données, de leur validation et de leur persistance. Dans le cas de StoreSwap, le modèle peut englober les entités telles que les clients, les produits et les commandes, ainsi que les opérations associées à ces entités.

La vue est responsable de l'affichage des données à l'utilisateur. Elle présente les informations de manière conviviale et attractive, en utilisant des éléments tels que les interfaces graphiques et les templates. Dans StoreSwap, la vue peut représenter les différentes pages et interfaces permettant aux utilisateurs de rechercher, afficher et interagir avec les produits et les fonctionnalités de l'application.

Le contrôleur fait le lien entre le modèle et la vue. Il gère les interactions de l'utilisateur, collecte les données saisies et les transmet au modèle pour traitement. Il récupère également les résultats du modèle et les transmet à la vue pour affichage. Dans le contexte de StoreSwap, le contrôleur peut être responsable de gérer les requêtes de recherche, de valider les informations fournies par les utilisateurs lors de la création de leur compte, et de coordonner les actions entre le modèle et la vue.

En adoptant le modèle vue contrôleur, l’application web StoreSwap peut bénéficier d'une séparation claire des responsabilités, ce qui facilite la maintenance, l'évolutivité et la réutilisabilité du code. Il permet également une meilleure organisation du code, ce qui rend le développement et la collaboration plus efficaces. En comprenant ces principes du modèle vue contrôleur, l’application StoreSwap ne peut qu’être un plaisir à développer.

## 6.2 – Application desktop IAPM

Elaboration des contraintes techniques :

L’éditeur de code utilisé est **Microsoft Visual Studio 2022**. Je le trouve ergonomique et facile d’utilisation.

Comme pour l’application StoreSwap, l’application IAPM communique avec une base de données MySQL, hébergée sur un serveur WAMP pour le travail en local.

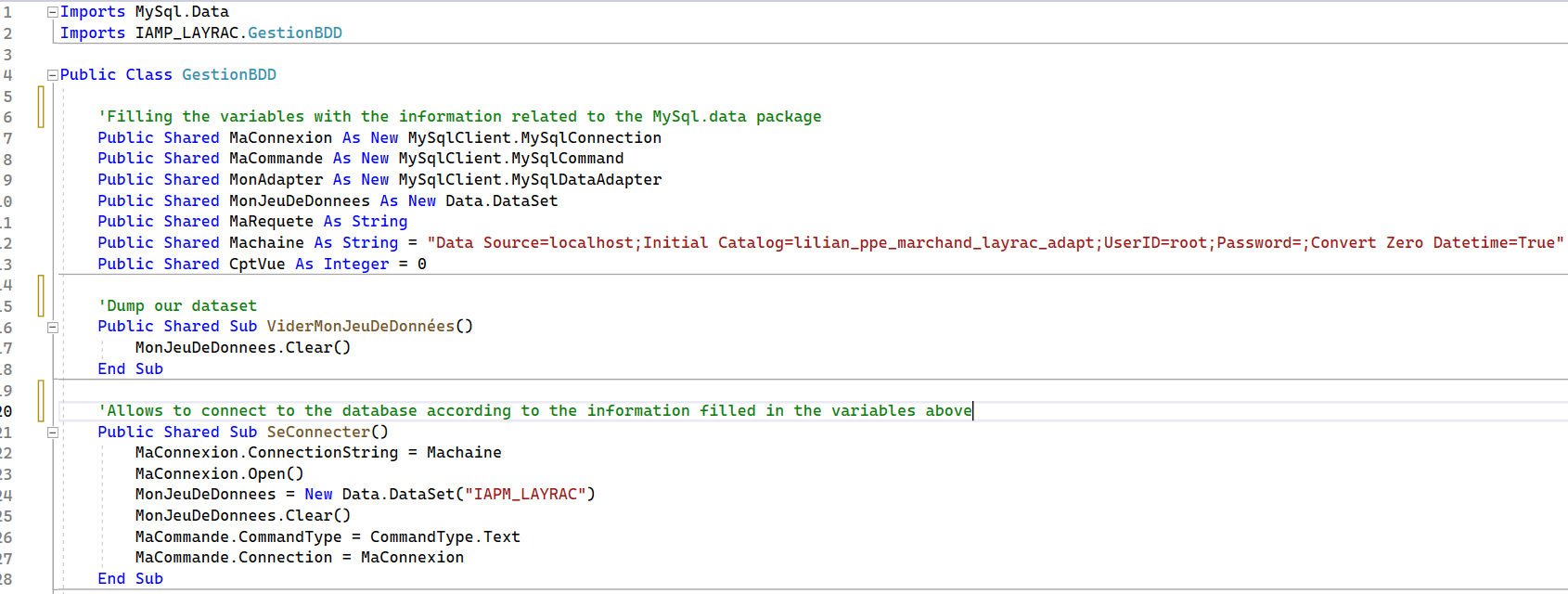
Pour la plateforme, j’ai choisi le langage **Visual Basic** pour le développement. Pour le design, j’ai utilisé **WinForm**, un framework de design proposé dans le package **.NET Core**.

Les exigences de **l’interface administrateur** sont que l’interface soit intuitive et facile de compréhension pour l’administrateur. Il doit aller d’un point A à un point B simplement et efficacement sans se perdre sur le site, sinon il risque de le quitter et donc arrêter la navigation.

Dans le contexte fictif, l’entreprise Layweb assurera la maintenance de l’application si nécessaire et si amélioration est demandé par le client, cependant, elle vous sera facturée.

Composants d’accès aux données :

Pour le projet IAPM, pour effectuer la connexion à la base de données, j’ai créé la classe GestionBDD, qui, comme avec l’application web StoreSwap, s’occupe d’effectuer la connexion à la base de données.



Comme pour StoreSwap, je remplie mes variables avec les informations de connexion pour créer une chaîne propre à MySQL. Je mets donc les informations pertinentes.

Ensuite, la fonction SeConnecter me permet d’utiliser ma chaîne de connexion et donc accéder à mes données stockées dans ma base de données.

Bien évidemment, pour effectuer la connexion et remplir les bonnes variables, je dois importer le module MySql.data, et utiliser les variables associées comme MySqlClient.MySqlConnexion, qui va s’occuper d’établir la connexion à la base de données.

Cette fonction sera appelée dans chaque formulaires pour avoir accès constamment aux données.

## 6.3 – Application Mobile

# 7 – Réalisations du candidat

***IAPM***

***Application Métier Desktop***

1. IAPM (Interface Applicatif de Produits Marchands) / Application Métier Desktop

**Origine**

J’ai créé cette application pour un projet personnel il y a 2 ans. Je l’ai peaufinée au fil du temps pour y intégrer de nouvelles fonctionnalités, améliorer le design, tout en acquérant de nouvelles compétences.

**Objectif/But**

L’objectif principal de cette application est la gestion de clients, produits et commandes. Mais cela ne s’arrête pas là, elle gère également la base de données commune au site web marchand associé.

Le but est de se passer totalement d’un éditeur de base de données traditionnel pour l’insertion de tuples dans la base de données. Ainsi, nous pourrons gérer la base de données du site web marchand, avec l’application desktop IAPM.

**Principe**

IAPM est une application desktop, elle est développée avec le langage objet VB.net. Elle consiste à manipuler une base de données qui est commune à un site web marchand.

Dans l’application IAPM, on peut :

- Ajouter, supprimer ou modifier des clients/utilisateurs

- Ajouter, supprimer ou modifier des produits

- Créer, supprimer ou modifier des commandes de clients/utilisateurs.

- Consulter les tables client, produits

- Performance de l’application avec des requêtes côté serveur (procédures stockées, triggers)

Côté design, l’application est modélisée en WindowsForm, avec l’éditeur Visual Studio 2019 (puis passé à la version 2022 pour y intégrer le style Windows 11).

Côté base de données, j’utilise une base SQL qui tourne sur un serveur local WAMP, l’édition de la base de données s’effectue sur le logiciel HeidiSQL.

* 1. Maquetter une application

Pour la présentation de ce projet IAPM, je vais commencer par vous montrer l’étape de maquettage de l’application, en élaborant tous les aspects relatifs à cette étape de conception qu’il ne faut pas négliger, et qui en somme, est l’une des étapes les plus importantes lors de la conception d’un projet de cette envergure.

Le maquettage de l’application IAPM s’est déroulé en plusieurs étapes, identifier les besoins utilisateur, élaborer les besoins fonctionnels, créer des wireframes pour donner un aspect simplifié de l’interface utilisateur (avec les composants clés), créer des interfaces utilisateur avec un design concret et une charte graphique bien spécifique.

Premièrement, j’ai élaboré les différents aspects que traitent IAPM. Elle traite un besoin d’administration, qui permet la gestion d’une base de données. Cela permet à l’utilisateur de gérer la base de données, sans accéder à un éditeur de base de données classique, en ayant une interface épurée et facile d’utilisation.

IAPM gère la base de données du site web marchand StoreSwap (présenté dans le projet ci-après)

Ensuite, j’ai élaboré les fonctionnalités de l’application qui sont les suivantes :

Une connexion sécurisée est nécessaire pour pouvoir connecter notre application à la base de donnée, via l’onglet « Connexion BDD ».

Coté gestion des clients de la base de données, nous pouvons :

- Créer, modifier ou bien supprimer des clients (clients du site web marchand StoreSwap),

- Nous pouvons également consulter la liste de tous les clients, qui affiche leur identifiant, leur nom, leur prénom, leur mot de passe, et toutes les informations relatives à la table clients.

Coté gestion des produits de la base de données, nous pouvons :

- Créer, modifier, mais aussi supprimer des produits,

- Nous pouvons aussi consulter la liste de tous les produits, qui affiche leur identifiant, leur libelle, leur prix, et toutes les informations relatives à la table produits.

Coté gestion des commandes de la base de données, nous pouvons :

- Créer, modifier ou supprimer des commandes,

- Nous pouvons également consulter toutes les commandes en fonction des clients, des produits, et de la date associé à cette dernière.

À noter que les tables clients, produits et commandes sont consultables dans un formulaire à part, dans une liste de recherche bien spécifique.

Pour la partie wireframes, j’ai créé un maquettage spécifique sur le logiciel Figma pour avoir une vue grossière de l’application (position des boutons, des listes, des formulaires)

Après cette étape, j’ai donc continué sur la réalisation de l’interface utilisateur, en élaborant une charte graphique, intégrant des images/icones/logo pour permettre à l’interface d’être épurée et simple. J’en ai donc déduit que l’application serait composée de couleurs pâles, avec des boutons bleus, rouges, verts, avec un fond beige. La navigation sur l’application s’effectue sur un menu avec des images, et chaque image serait cliquable pour accéder aux gestions des tables spécifiques.

Vous pouvez retrouver des exemples d’interface ici, qui se traduit par la réalisation d’une maquette sur Figma : (Voir Annexe pour IAPM – Annexe IAPM 1)

Enfin, ces interfaces ont été testées auprès d’utilisateurs potentiels, pour en découler des améliorations possibles avec des couleurs à changer, boutons à remplacer, ou même ajouter de nouvelles fonctionnalités.

Toutes ces étapes m’ont permis d’élaborer le projet et ainsi de partir sur de bonnes bases avant de me lancer dans la réalisation de l’application. Il est primordial de passer par ces étapes avant de développer l’application, car à la manière d’une dissertation, il faut savoir poser les bases et élaborer une bonne stratégie pour ne pas choisir le mauvais chemin lors du lancement en développement.

* 1. Développer une interface utilisateur de type desktop

Je vais à présent vous montrer l’étape du développement de l’interface utilisateur de type desktop.

Dans ce projet IAPM, j’ai respecté les bonnes pratiques de programmation orientée objet, qui sont les suivantes :

L’encapsulation, c’est-à-dire, masquer la complexité interne d'un objet et à fournir une interface simple pour interagir avec cet objet. Elle permet d'améliorer la sécurité, la modularité et la flexibilité du code. En somme, dans mon application IAPM, c’est le fait d’avoir créé des formulaires intuitifs, qui permettent une navigation simple, tout en gardant un aspect de sécurité.

L’abstraction, c’est une technique qui permet de modéliser des concepts du monde réel en utilisant des classes. Elle permet de simplifier la complexité de la réalité en isolant les détails les moins importants et en se concentrant sur les aspects les plus pertinents. Par exemple, dans IAPM, j’ai créé plusieurs classes comme GestionBDD (Ou j’ai créé toutes mes fonctions relatives à la gestion de la base de données), ou encore GestionProduit, (Ou j’ai créé toutes mes fonctions relatives à la gestion de mes produits).

L’héritage, est une technique qui permet de définir une nouvelle classe à partir d’une classe existante. Elle permet de réutiliser le code existant, de minimiser la duplication de code et d’améliorer la modularité du code. Comme par exemple, ma classe FrmAMSClient, qui stock mes fonctions de création de mon formulaire de gestion de client, qui hérite de la classe System.Windows.Forms.Form qui permet à ma classe FrmAMSClient d’hériter des informations de cette dernière.

Le polymorphisme, est ce qui permet à des objets de même type d’avoir des comportements différents. Elle permet d’améliorer la modularité et la flexibilité du code en permettant aux objets d’interagir de manière générique.

Il faut également que nos classes aient une seule responsabilité, et que nos méthodes soient cohérentes.

L’aspect qui est d’utiliser l’interface à disposition est également important, car elle permet par exemple, de faciliter la réalisation de formulaires, ou même de tests. J’ai utilisé le builder Winform disponible sur Visual Studio 2022 pour créer les formulaires d’IAPM.

Et enfin, nous pouvons utiliser la surcharge, c’est ce qui permet de nommer des méthodes avec le même nom, mais d’avoir des signatures différentes. Cela permet de simplifier l’écriture du code et d’améliorer sa lisibilité.

IAPM a été développé dans un style défensif qui permet de minimiser les vulnérabilités, par exemple, sur le formulaire d’ajout de client, un contrôle de saisie a été développé pour éviter toutes les erreurs lors de l’insertion (lettre au lieu de chiffres, champs vides…). Ces erreurs sont gérées correctement pour éviter tout plantage ou mauvaise insertion dans la base de données.

Le code source de l’application IAPM est commenté de façon à ce que le code soit maintenable par une autre personne. Les commentaires sont concis et clairs, avec l’essentiel.

Voici un exemple d’une fonction commentée : (Voir Annexe pour IAPM – Annexe IAPM 2)

La documentation du projet (sous forme de maquette technique), est rédigée de façon compréhensible pour des personnes lambda, qui ne connaîtrait pas les termes techniques.

Lors de la conception de mon application IAPM, j’ai utilisé des outils de débogage comme par exemple, le point d’arrêt, qui m’a permis d’effectuer du débogage pas à pas et de trouver les erreurs dans mes fonctions.

* 1. Développer des composants d’accès aux données

Tout d’abord, les données de l’application IAPM sont stockés en local sur un serveur WAMP. J’utilise une base de données en langage MySQL. Pour accéder à ces données, j’utilise une interface de connexion à la base de données qui s’appelle HeidiSQL, c’est un EBD (éditeur de base de données).

Pour le développement des composants d’accès aux données, j’ai choisi de développer une classe nommée GestionBDD, la réflexion a été de créer cette classe pour y mettre toutes les fonctions permettant la discussion avec la base de données.

Premièrement, j’ai importé le module MySql.data disponible dans le framework .NET Core, ensuite, j’ai rempli la classe en instanciant des variables utilisant les propriétés du module MySql.data, comme MySqlClient.MySqlConnexion qui permet d’effectuer la connexion à la base de données avec une chaîne de caractère spécifique, contenant le nom de la base de données, son utilisateur, son mot de passe.

Après avoir rempli mes variables avec les méthodes de MySql.data associés, je construis ma fonction qui se nomme SeConnecter() pour instancier la connexion. À savoir que la connexion à la base s’effectue au lancement de l’application, et renvoie un message d’erreur si elle ne fonctionne pas (La connexion peut également s’effectuer via le bouton Connexion BDD sur l’interface utilisateur dans le menu principal).

Vous pouvez retrouver la méthode SeConnecter ici : (Voir Annexe pour IAPM – Annexe IAPM 2)

Ensuite, coté applicatif, j’utilise des procédures stockées en base de données, ainsi que des triggers (déclencheurs), pour effectuer les requêtes coté serveur et donc éviter une surcharge coté client. C’est essentiellement une bonne pratique.

Voici la procédure stockée RechercherClient, qui permet d’effectuer la recherche d’un client en fonction de la chaîne de caractère entrée dans la section rechercher sur le formulaire de gestion des clients : (Voir Annexe pour IAPM – Annexe IAPM 3)

J’ai également développé des triggers (déclencheurs), pour effectuer des automatismes coté base de données. Comme le trigger GestionStats (Voir Annexe pour IAPM – Annexe IAPM 4). Il permet de récupérer le nombre de commandes, le nombre de clients ainsi que le nombre de produits. Il s’actualise toutes les 10 secondes et affiche sur l’application une liste contenant le nombre de commandes, de clients, de produits ainsi que la date ou le trigger a été déclenché. (Voir Annexe pour IAPM – Annexe IAPM 5).

* 1. Développer la partie back-end d’une interface utilisateur web

Maintenant, je vais présenter la partie back-end de mon application StoreSwap, qui est gérer dans l’application IAPM.

* 1. Concevoir une base de données
  2. Mettre en place une base de données
  3. Développer des composants dans le langage d’une base de données
  4. Développer des composants métiers

***StoreSwap***

***Application Web***

1. StoreSwap / Application Web

**Origine**

Ce projet personnel a été créé, comme IAPM, il y a deux ans. Je l’ai également peaufinée au fil du temps pour y intégrer de nouvelles fonctionnalités, améliorer le design, tout en acquérant de nouvelles compétences.

**Objectif/But**

L’objectif principal de ce projet est de vendre des bonbons, c’est basiquement un site web marchand, avec une boutique et un panier.

Le but de ce site web marchand, est la création de compte utilisateurs (clients), gérer un panier en fonction des articles dans la boutique, et également la gestion de la base de données dans la section administration. (La base de données est tout de même gérée par l’application IAPM).

**Principe**

StoreSwap est une application web, elle est développée en html/css, et php. Pour le côté style, j’ai utilisé le framework Bootstrap 5 pour le CSS. L’application web manipule la même base de données que IAPM.

Dans l’application web StoreSwap, nous pouvons :

- Gérer un panier en fonction des articles dans la boutique

- Créer des utilisateurs

- Se connecter avec son compte utilisateur

- Afficher les données utilisateurs sur une page Profil

- Consulter la base de données sur la partie administration

- Gestion des cookies de session pour rester connecter avec son compte

Côté design, l’application utilise le framework Bootstrap 5 (combiné avec du CSS personnalisé). L’éditeur de cote utilisé est Visual Studio Code.

Côté base de données, j’utilise une base SQL (MySQL) qui tourne sur un serveur local WAMP, l’édition de la base de données s’effectue sur le logiciel HeidiSQL.

Côté exécution du code, avec WAMP, je fais tourner un serveur Apache 2 pour interpréter le php.

* 1. Développer des composants d’accès aux données
  2. Développer la partie front-end d’une interface utilisateur web
  3. Concevoir une base de données
  4. Mettre en place une base de données

2.5 Développer des composants dans le langage d’une base de données

* 1. Construire une application organisée en couches
  2. Préparer et exécuter le déploiement d’une application

***TentationVoyage***

***Application Web***

1. TentationVoyage / Application Web

**Origine**

TentationVoyage est un projet professionnel. Le travail a été effectué par une équipe de 3 développeurs, dont moi. Nous avons fait la refonte totale du site de base, en y refaisant une refonte totale du CSS, ainsi que la gestion interne du site (Nous avons gardé la même charte graphique).

**Objectif/But**

L’objectif principal de cette mission était de donner un « coup de jeune » au site web. Et ça a été chose faite. Le site est méconnaissable, et cette version est plus au goût du jour, et est surtout plus au goût de son administrateur.

**Principe**

TentationVoyage est un site web développé en html/css et php. Pour le style, nous avons utilisé Bootstrap 5. La base de donnée est en MySQL, gérer avec l’interface PhpMyAdmin.

Cette application permet d’organiser des voyages avec des widgets (Kayak, Skyscanner…). Nous pouvons commander des billets pour des vols, trains, bus, bateaux.

Sur l’interface utilisateur, vous pouvez trier les widgets avec un système de favoris, et être abonné à une Newsletter pour recevoir des bons plans définit par l’administrateur du site (en fonction des destinations préférées définis sur votre compte utilisateur).

Dans l’application web TentationVoyage, nous pouvons :

- Commander des billets avec des widgets affiliés

- Planifier des voyages

- Mettre des widget favoris

- Créer des comptes utilisateurs

- S’abonner à une Newsletter

Pour le coté administration :

- Gérer les utilisateurs (Ajout, suppression, modification)

- Gérer les widgets (Intégrer des widget en ajoutant directement le code Html/js), pour que l’administrateur puisse administrer son site sans même toucher directement le code.

- Publier des « Bon plans » (qui seront publié sur le site, mais également sous forme de Newsletter).

Côté design, l’application utilise le framework Bootstrap 5 (combiné avec du CSS personnalisé). Nous avons utilisé VisualStudio Code, avec des extensions comme LiveShare pour travailler en équipe et à distance.

Côté base de données, nous avons utilisé une base SQL (MySQL), qui tournait sur un serveur OVH Cloud (ou le site est également hébergé.)

* 1. Développer des composants d’accès aux données
  2. Développer la partie front-end d’une interface utilisateur web
  3. Concevoir une base de données
  4. Mettre en place une base de données
  5. Développer des composants dans le langage d’une base de données
  6. Collaborer à la gestion d’un projet informatique et à l’organisation de l’environnement de développement
  7. Concevoir une application
  8. Développer des composants métier
  9. Construire une application organisée en couches
     1. Préparer et exécuter le déploiement d’une application

***MSPR A’rosa-je***

***Application Web, Desktop, Mobile***

1. MSPR A’rosa-je / Application Web, Desktop, Mobile

**Origine**

MSPR A’rosa-je est un projet d’étude supérieure par groupe, en appliquant un contexte et une démarche professionnelle.

Nous avons eu un groupe de 4, avec des tâches bien réparties en fonction des compétences et aisances de chacun.

La mission a été réalisée avec succès, en répondant à toutes les problématiques énoncées dans le référentiel d’objectifs en créant ces applications.

**Objectif/But**

L’objectif de ce contexte est de créer 3 applications, web, mobile et desktop. Elles doivent avoir une base de données commune, et communiquer entre elles.

Le but de l’application web est de gérer les comptes utilisateurs, gérer l’ajout de plantes, interagir via une messagerie avec des botanistes, accéder à la carte pour afficher les botanistes pouvant garder vos plantes.

Le but de l’application mobile est la consultation de l’état de votre plante, gardée par un botaniste en particulier.

Le but de l’application desktop est le même que l’application web.

**Principe**

MSPR A’rosa-je est un projet qui met en relation des utilisateurs et des botanistes, les 3 applications permettent une communication claire et précise entre l’utilisateur et le botaniste, pour avoir toutes les informations essentielles et nécessaires au bien-être de la plante.

L’application web est réalisée avec le framework Angular (TypeScript), pour le CSS, nous avons développé un CSS personnalisé.

L’application mobile est réalisée avec le framework Flutter (Dart)

L’application desktop est une retranscription de l’application web avec Electron

Avec ces 3 applications, vous pourrez :

- Gérer les utilisateurs/botanistes (Ajout, modification, suppression)

- Ajouter des plantes (les classer par familles)

- Visualiser, via une carte interactive, les botanistes et les plantes

- Échanger entre utilisateurs et botanistes via un chat (messagerie)

* 1. Développer la partie front-end d’une interface utilisateur web
  2. Développer la partie back-end d’une interface utilisateur web

4.3 Concevoir une base de données

4.4 Mettre en place une base de données

4.5 Développer des composants dans le langage d'une base de données

4.6 Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement

4.7 Concevoir une application

4.8 Développer des composants métier

4.9 Construire une application organisée en couches

4.9.1 Développer une application mobile

4.9.2 Préparer et exécuter le déploiement d'une application

***Prisearch***

***Application CLI***

1. PriSearch / Application CLI

**Origine**

**Objectif/But**

**Principe**

5.1 Développer une interface utilisateur de type desktop

5.2 Développer la partie back-end d'une interface utilisateur web

5.3 Préparer et exécuter les plans de tests d'une application

1. AirSens / WorkShop

**Origine**

**Objectif/But**

**Principe**

6.1 Maquetter une application

6.2 Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement

1. VITAM (Programme Interministériel d’Archivage Numérique) / Projet Professionnel

**Origine**

**Objectif/But**

**Principe**

7.1 Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement

7.2 Concevoir une application

X Conclusion

X Glossaire

IAPM : Interface Applicatif de Produits Marchands

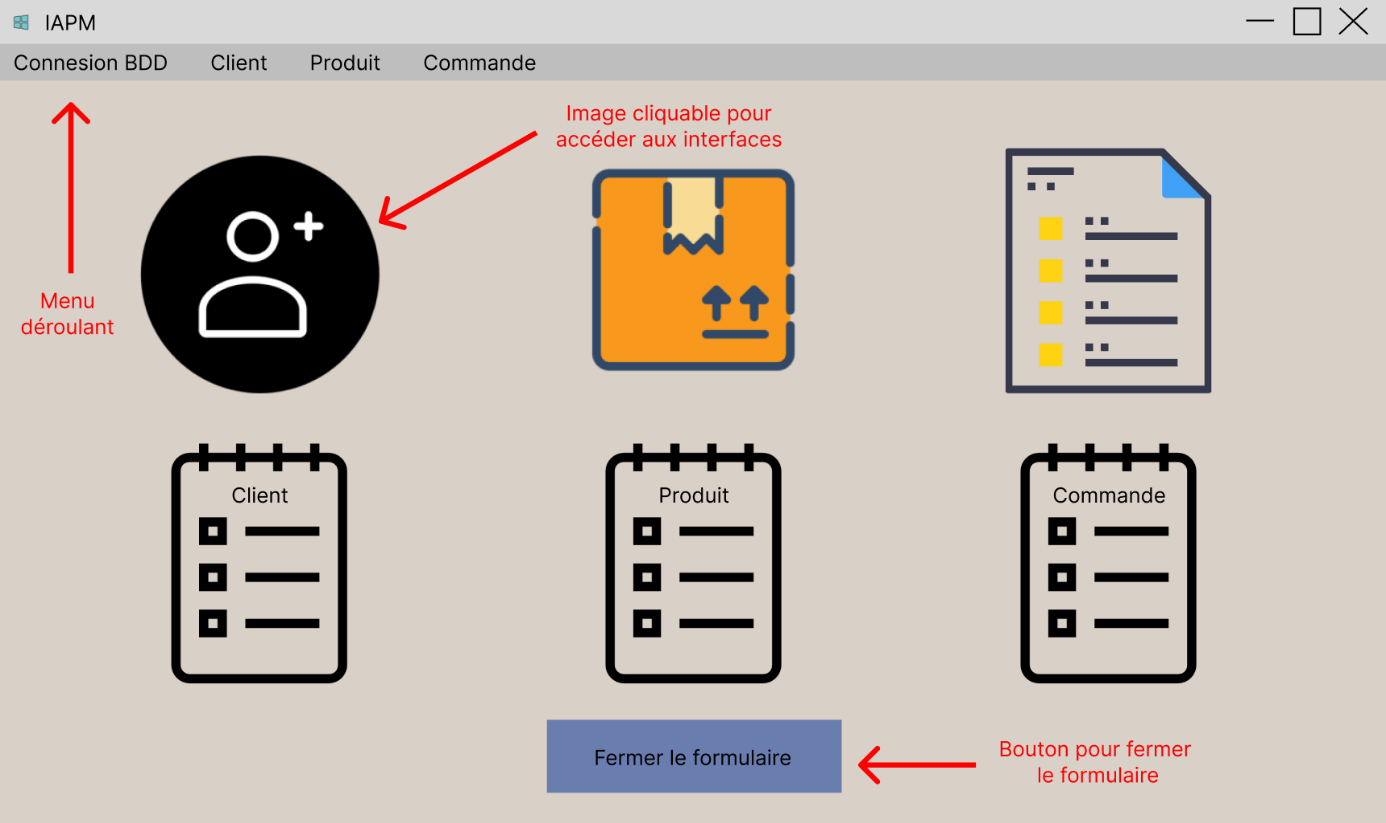
MVC : Model-View-Controller

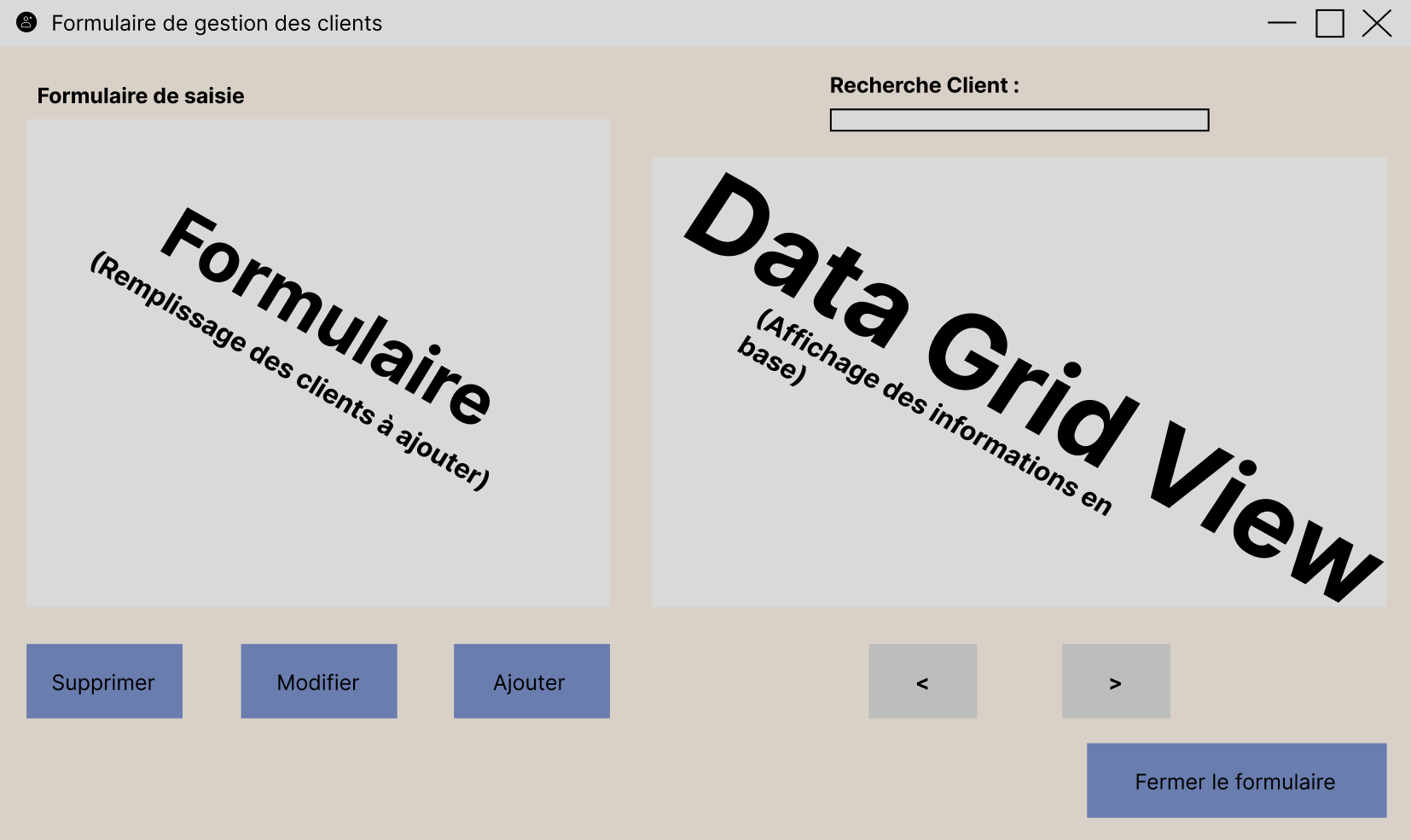
MVVM : Model-View-ViewModel

X Annexes

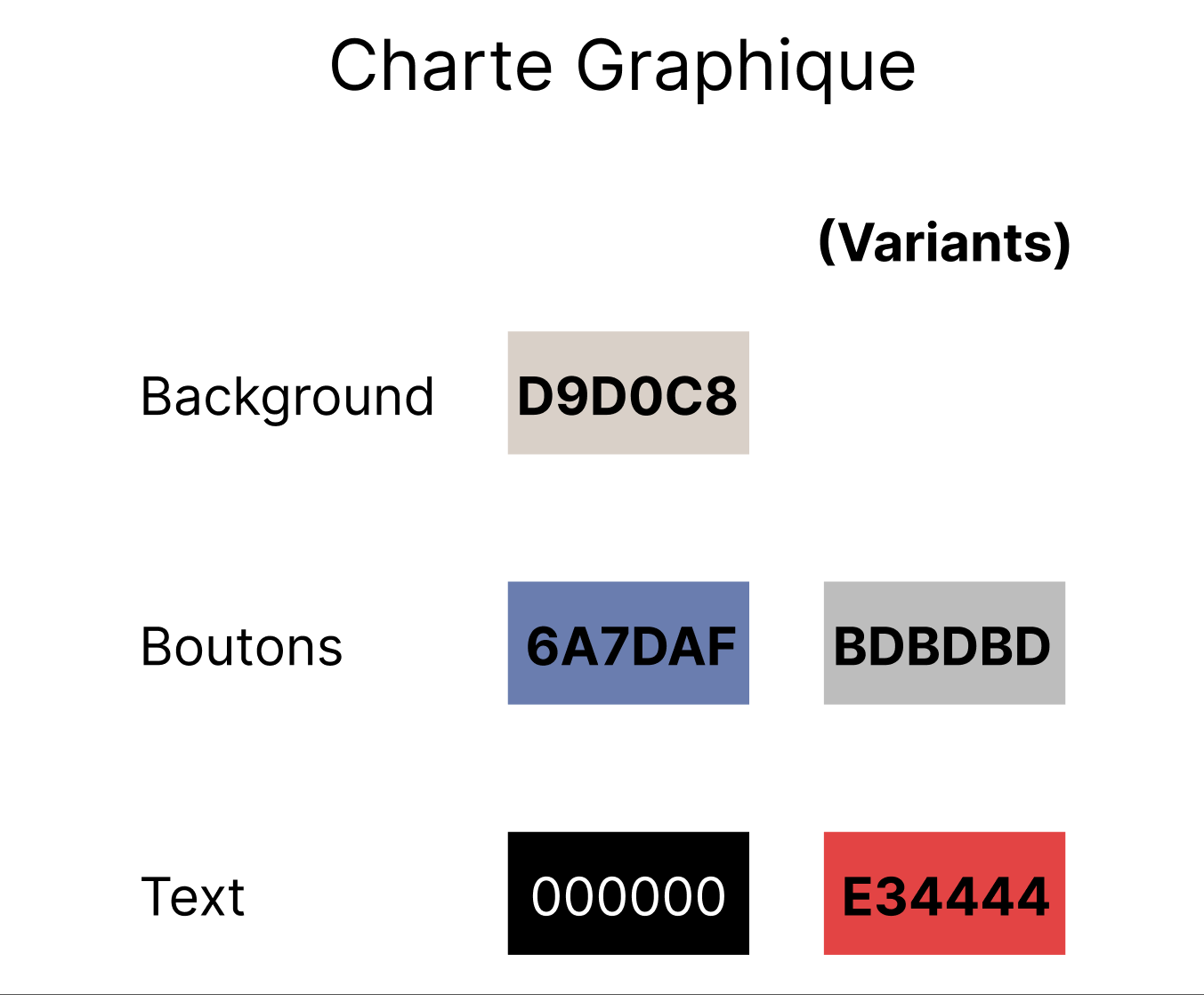
Annexes pour IAPM

Annexe IAPM Maquette

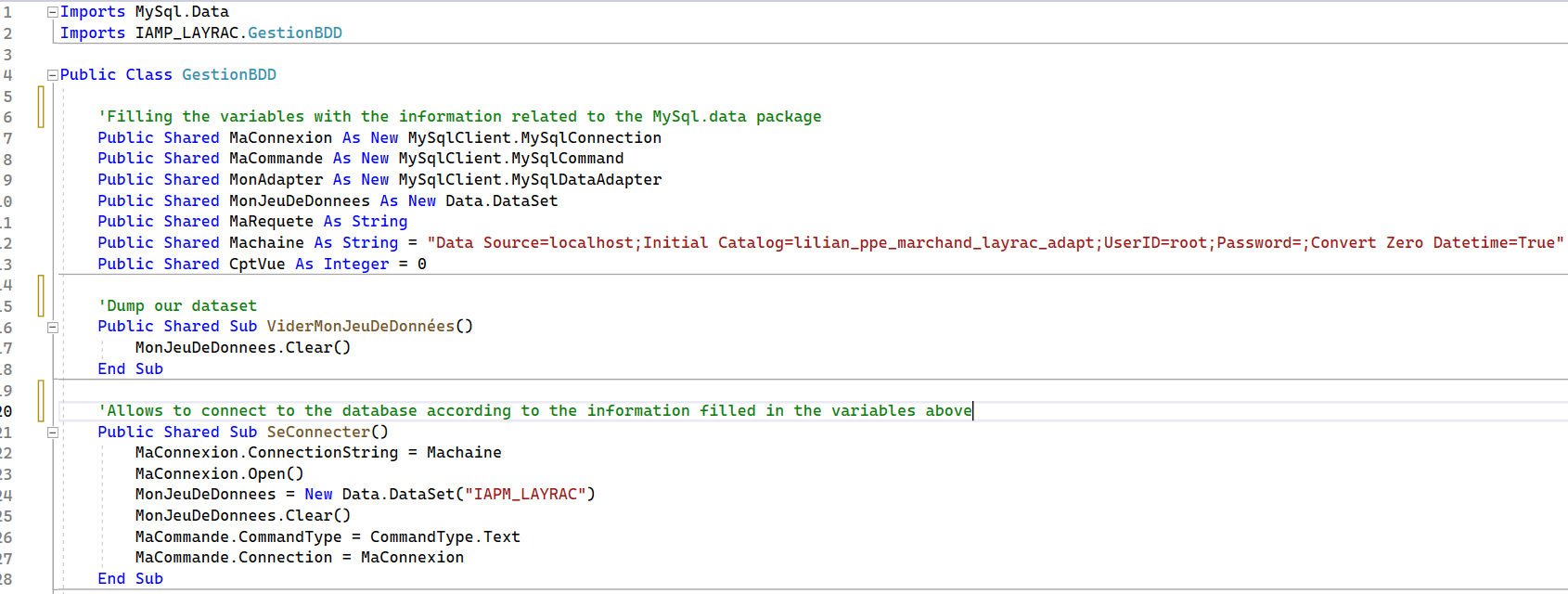




### Annexe IAPM Charte Graphique

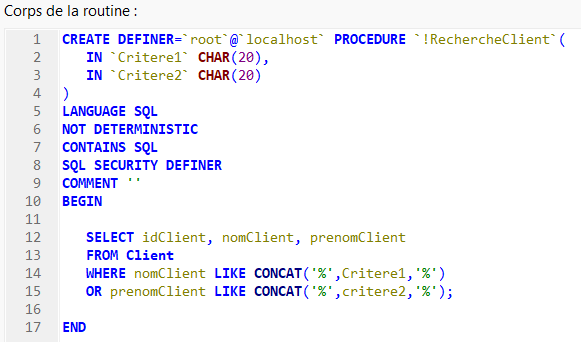


Annexe IAPM Connexion

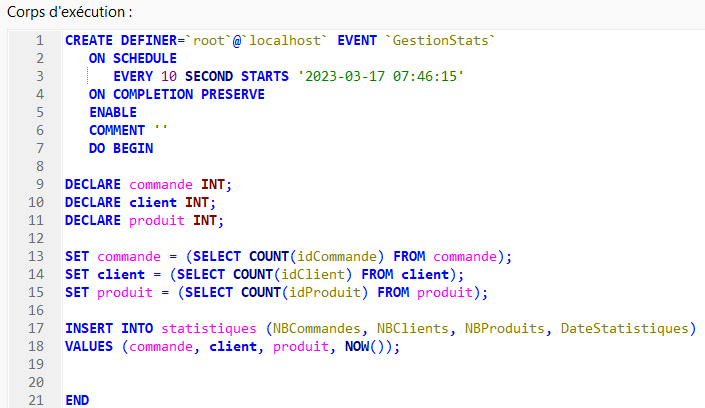


Annexe IAPM BDD

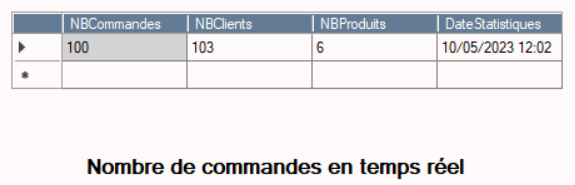
Procédure stockée :



Déclencheur :



Annexe IAPM Résultat Trigger



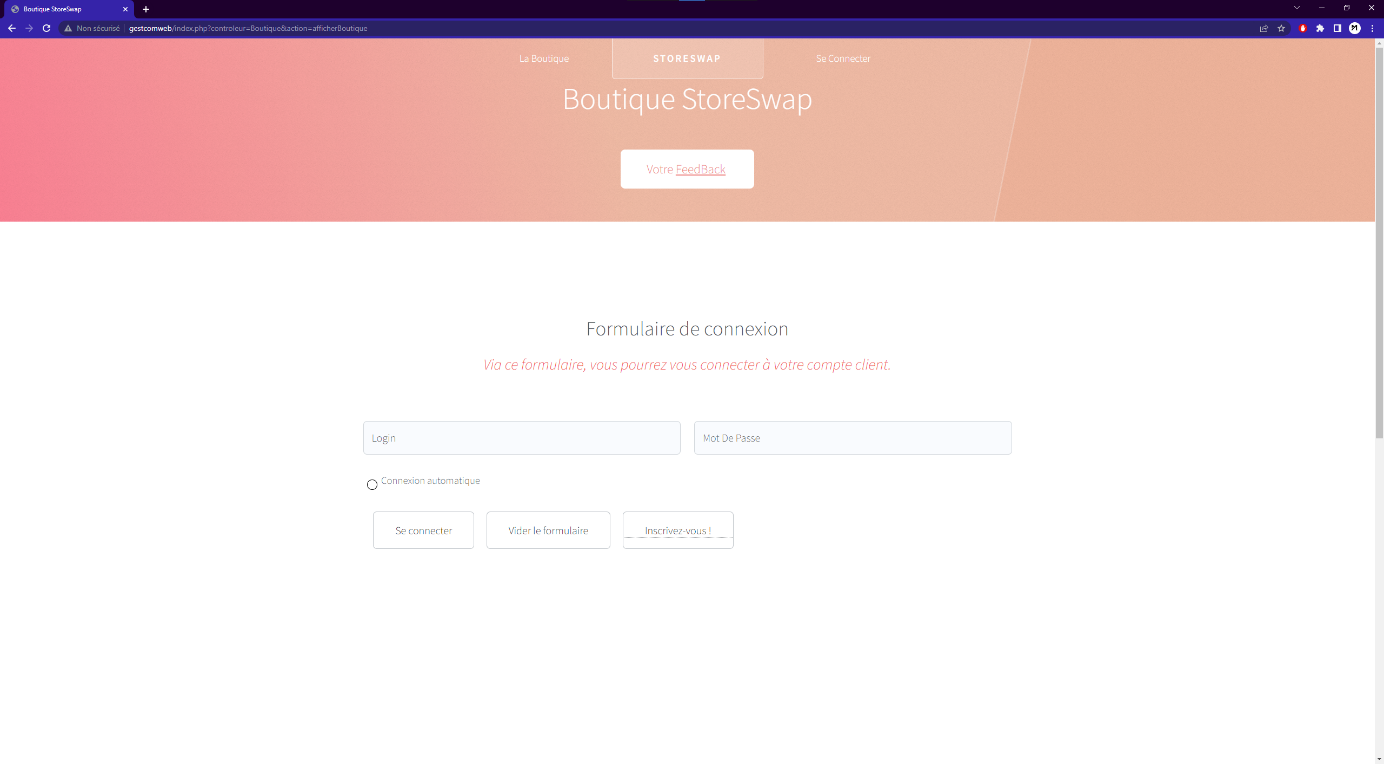
### Annexe IAPM Tests Unitaires

Annexes pour StoreSwap

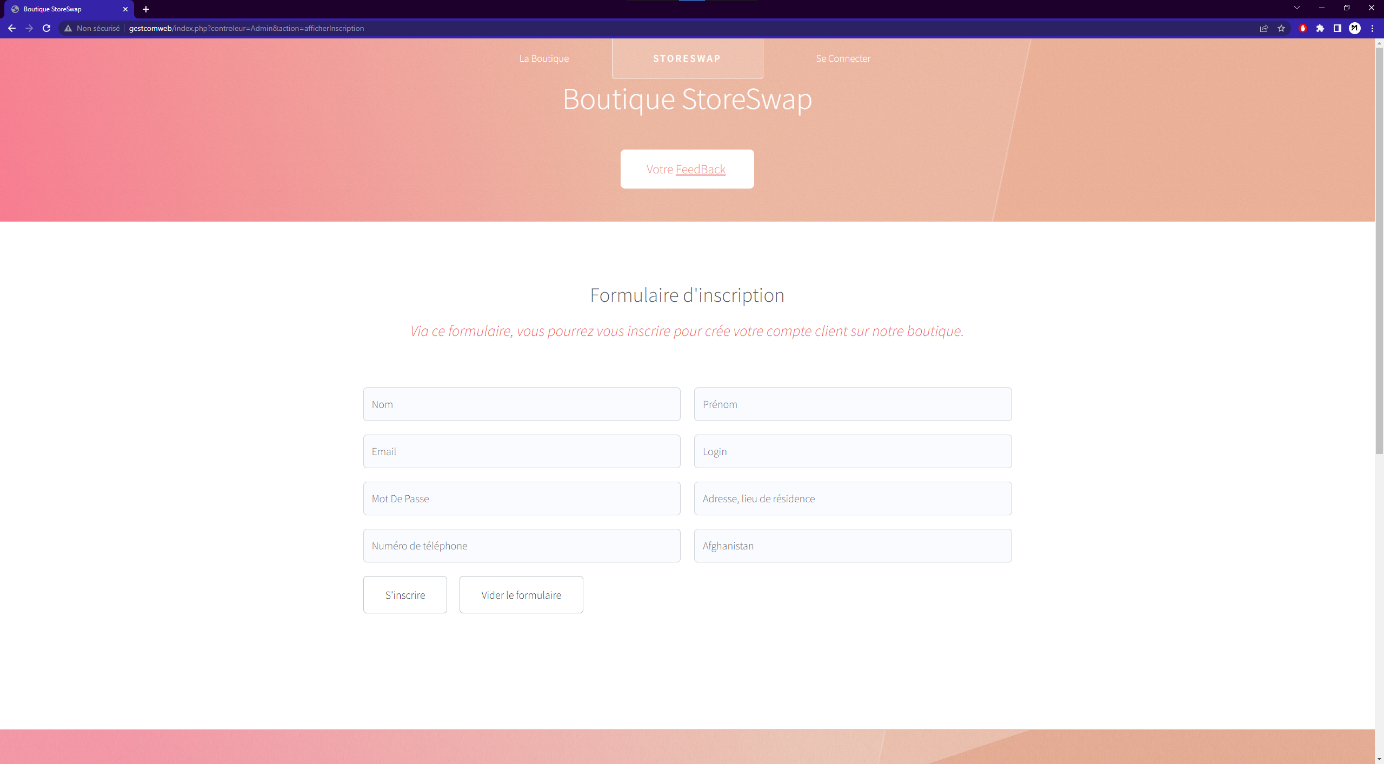
### Annexe StoreSwap Accueil



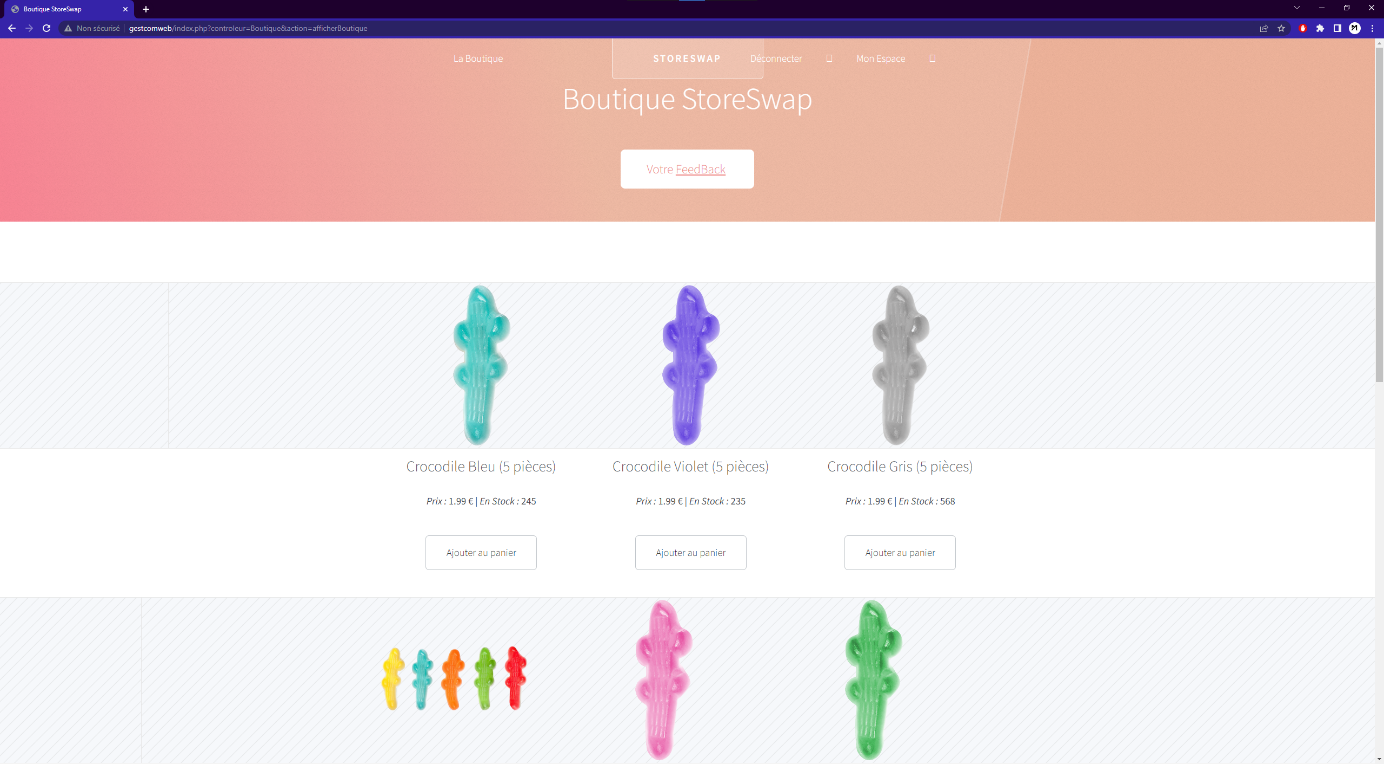
### Annexe StoreSwap Connexion



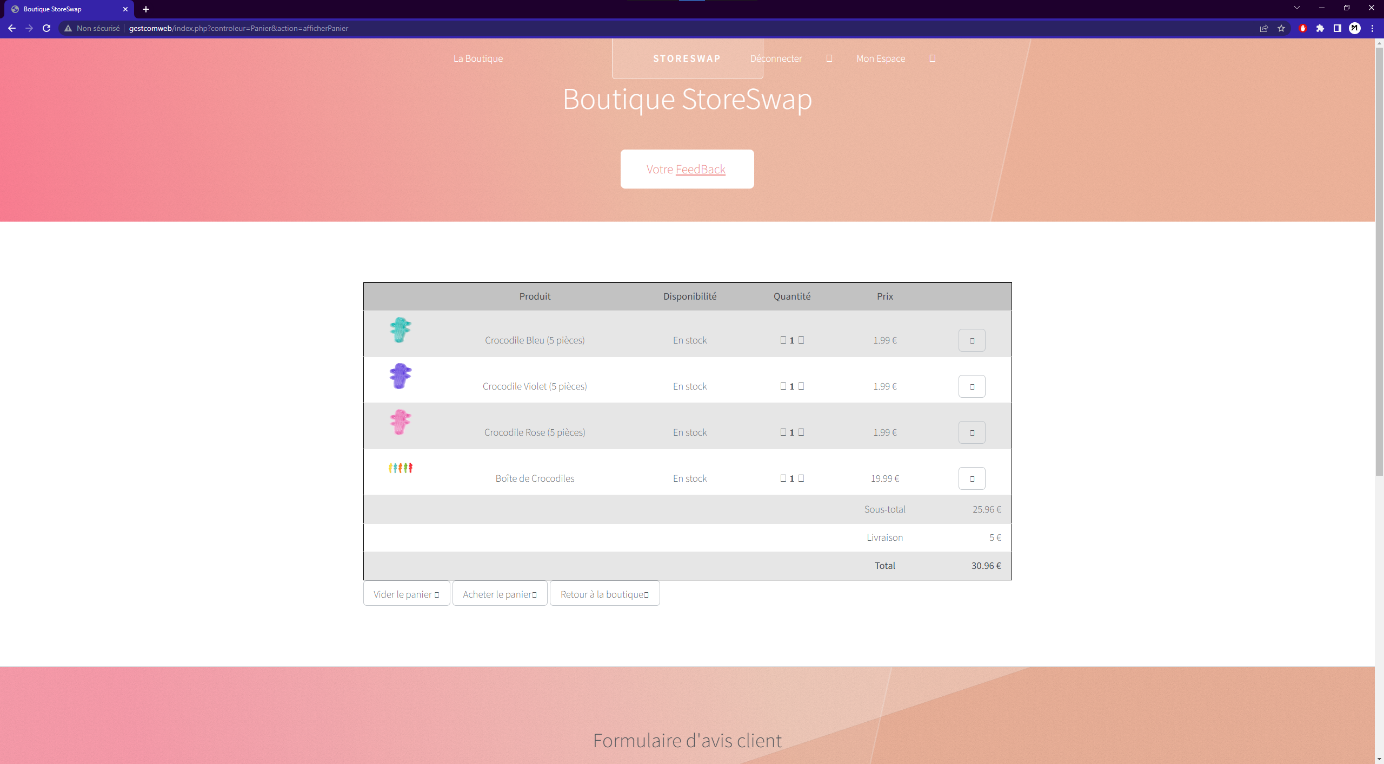
### Annexe StoreSwap Inscription



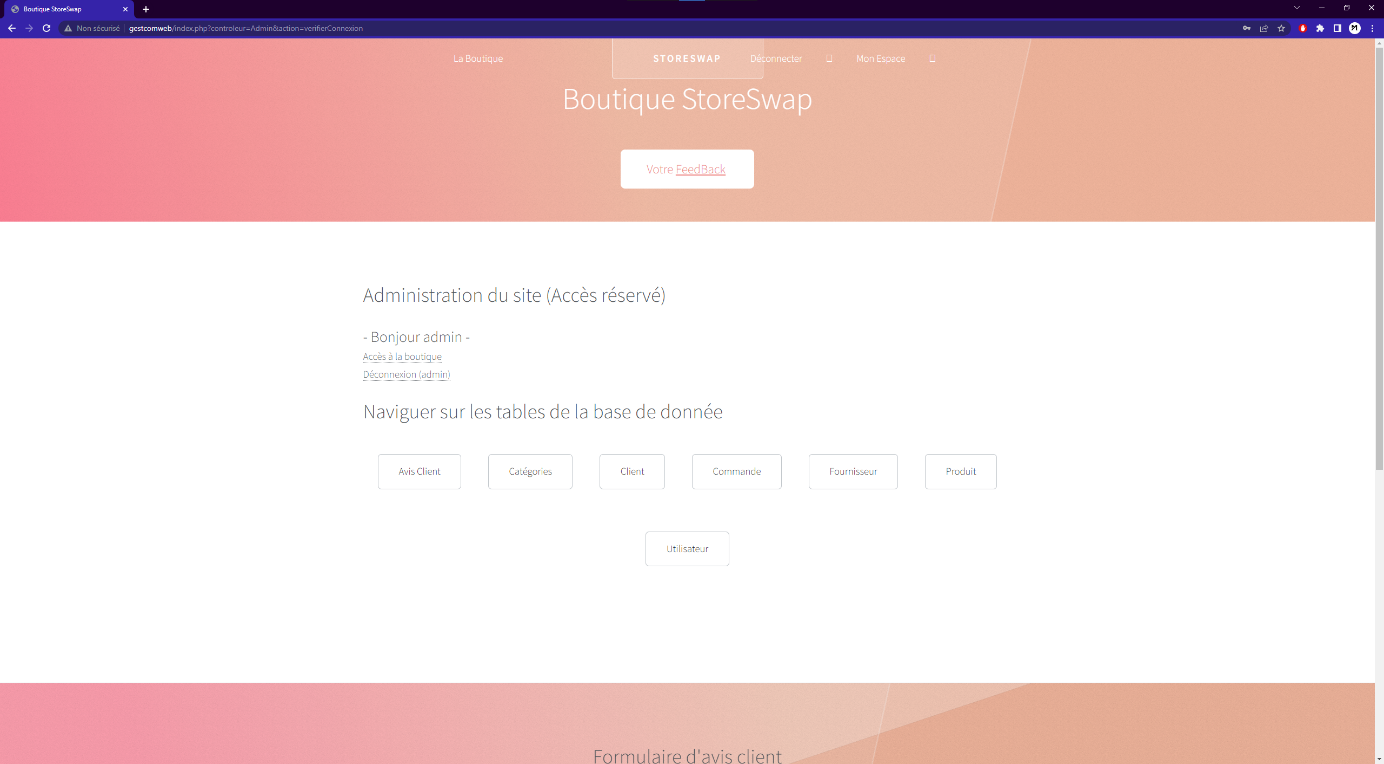
### Annexe StoreSwap Boutique



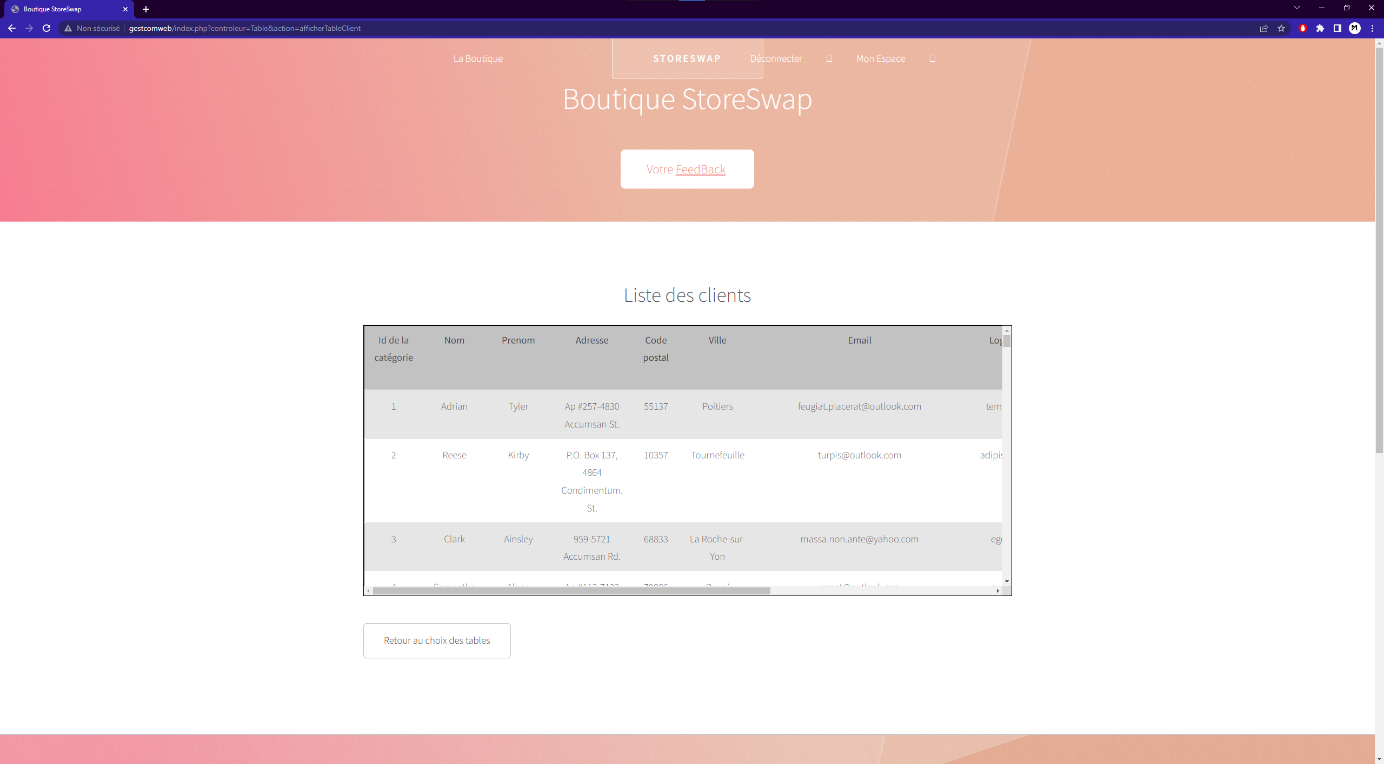
### Annexe StoreSwap Panier



### Annexe StoreSwap Admin Table



### Annexe StoreSwap Admin Table Client



### Annexe StoreSwap Code Connexion



Annexes pour TentationVoyage

Annexes pour MSPR A’rosa-je

Annexes pour PriSearch

Annexes pour AirSens

Annexes pour VITAM