**TITRE PROFESSIONNEL**

**Concepteur Développeur d’Application**

**Niveau II**

Dossier de projet

***Lilian LAYRAC***

**EPSI**

*École d'ingénierie informatique*

**TABLE DES MATIERES**

[Présentation personnelle 5](#_Toc135266129)

[1 - Référentiel de compétences 6](#_Toc135266130)

[2 - Résumé des projets 7](#_Toc135266131)

[2.1 - Introduction 7](#_Toc135266132)

[2.2 – Projet IAPM | StoreSwap 7](#_Toc135266133)

[2.3 – Projet Mobile 7](#_Toc135266134)

[3 – Cahier des charges 8](#_Toc135266135)

[3.1 – Cahier des charges pour IAPM | StoreSwap 8](#_Toc135266136)

[3.1.1 – Description du cahier des charges du projet 8](#_Toc135266137)

[3.1.2 – Utilisateurs des applications 8](#_Toc135266138)

[3.1.3 – Tests et validation 9](#_Toc135266139)

[3.2 – Cahier des charges pour Mobile 9](#_Toc135266140)

[4 – Gestion de projet 10](#_Toc135266141)

[4.1 – Gestion de projet IAPM | StoreSwap 10](#_Toc135266142)

[4.2 – Gestion de projet Mobile 12](#_Toc135266143)

[5 – Spécifications fonctionnelles 13](#_Toc135266144)

[5.1 – Application web StoreSwap 13](#_Toc135266145)

[5.2 – Application desktop IAPM 18](#_Toc135266146)

[5.3 – Application Mobile 20](#_Toc135266147)

[6 – Spécifications techniques 21](#_Toc135266148)

[6.1 – Application web StoreSwap 21](#_Toc135266149)

[6.2 – Application desktop IAPM 24](#_Toc135266150)

[6.3 – Application Mobile 25](#_Toc135266151)

[7 – Réalisations du candidat 26](#_Toc135266152)

[7.1 – Application StoreSwap | IAPM 26](#_Toc135266153)

[7.2 – Application Mobile 26](#_Toc135266154)

[X Conclusion 27](#_Toc135266155)

[X Glossaire 28](#_Toc135266156)

[X Annexes 29](#_Toc135266157)

[Annexes pour IAPM 29](#_Toc135266158)

[Annexe IAPM Maquette 29](#_Toc135266159)

[Annexe IAPM Charte Graphique 30](#_Toc135266160)

[Annexe IAPM Connexion 30](#_Toc135266161)

[Annexe IAPM BDD 31](#_Toc135266162)

[Annexe IAPM Résultat Trigger 31](#_Toc135266163)

[Annexe IAPM Tests Unitaires 32](#_Toc135266164)

[Annexe IAPM App Menu 32](#_Toc135266165)

[Annexe IAPM App Ajout Client 32](#_Toc135266166)

[Annexe IAPM App Liste Client 33](#_Toc135266167)

[Annexe IAPM App Ajout Produit 33](#_Toc135266168)

[Annexe IAPM App Liste Produit 34](#_Toc135266169)

[Annexe IAPM Liste Commande 35](#_Toc135266170)

[Annexes pour StoreSwap 36](#_Toc135266171)

[Annexe StoreSwap Accueil 36](#_Toc135266172)

[Annexe StoreSwap Connexion 36](#_Toc135266173)

[Annexe StoreSwap Inscription 37](#_Toc135266174)

[Annexe StoreSwap Boutique 37](#_Toc135266175)

[Annexe StoreSwap Panier 38](#_Toc135266176)

[Annexe StoreSwap Admin Table 38](#_Toc135266177)

[Annexe StoreSwap Admin Table Client 39](#_Toc135266178)

[Annexe StoreSwap Code Connexion 40](#_Toc135266179)

Présentation personnelle

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 introduits dans une thématique professionnelle, 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.

En savoir plus sur moi et l’entreprise ou je travail :

Durant mon année d’alternance chez Optim.Solutions, je me suis formé à Docker et aux technologies qui touchent l’infrastructure système. J’ai acquis des compétences de déploiements et d’optimisation d’architectures qui m’ont beaucoup aidé lors de cette année à l’EPSI. Coté développement informatique, je me suis également auto-formé à des frameworks que je ne connaissais pas, tel que Symfony par exemple (formation sur la plateforme UDEMY). Malgré tout, je suis fier d’avoir réussi à surmonter le manque d’attention de mon tuteur, car cela m’a permis de devenir plus autonome, et surtout d’apprendre davantage en termes de développement informatique.

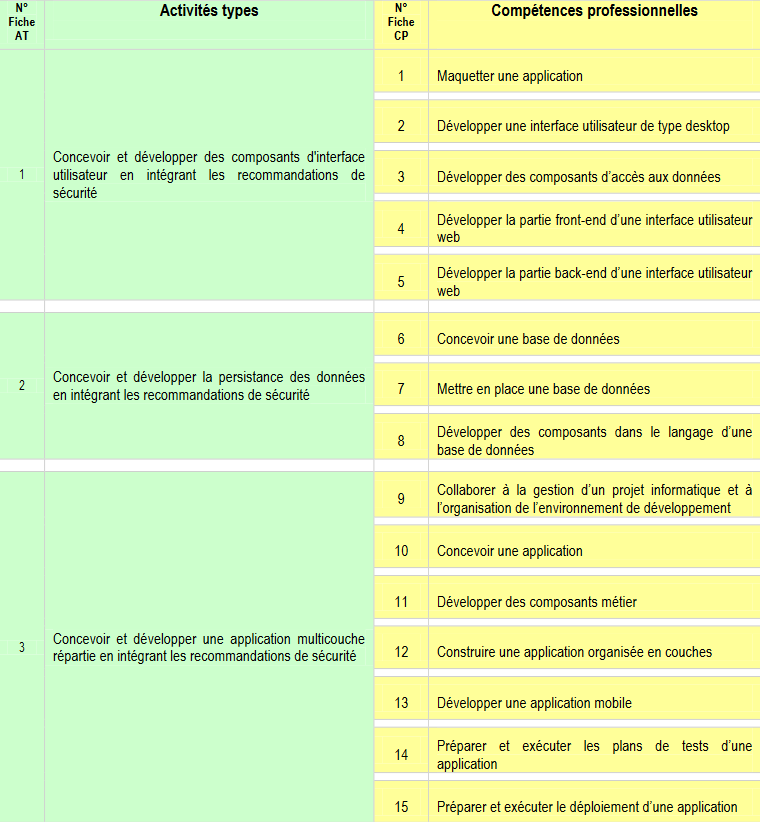
Notre équipe est composée de 4 alternants et d’un tuteur. Nous travaillons ensemble quand le temps nous le permet, car par soucis de calendrier (nous ne sommes pas tous dans la même école), nous devons nous adapter à chacun, et surtout travailler le soir pour mettre en commun.

Notre cycle de travail était basé sur des réunions journalières, pour faire un point tous les matins, nous étions dans un premier espace de co-working. Le temps est passé, mon ancien tuteur à déléguer la gérance de la société à mon tuteur actuel, et nous avons changé d’espace de co-working.

Dans ce nouvel espace, nous y somment seulement le vendredi, le reste du temps, nous sommes en télétravail. Ce n’est pas un cadre de travail idéal, surtout vue la communication qu’il y a dans l’équipe, et surtout entre nous et nos tuteurs (ex et nouveau). Mais nous parvenons à faire avec, et nous essayons tant bien que mal de sortir la tête de l’eau.

Sur ces notes un peu maussades, je vous laisse découvrir mon dossier pour le titre professionnel Concepteur Développeur d’Application, et vous souhaite une bonne lecture.

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. I also created Docker containers on linux operating system, to develop my flutter app (ubuntu).

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.

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 technologique 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 IAPM, 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

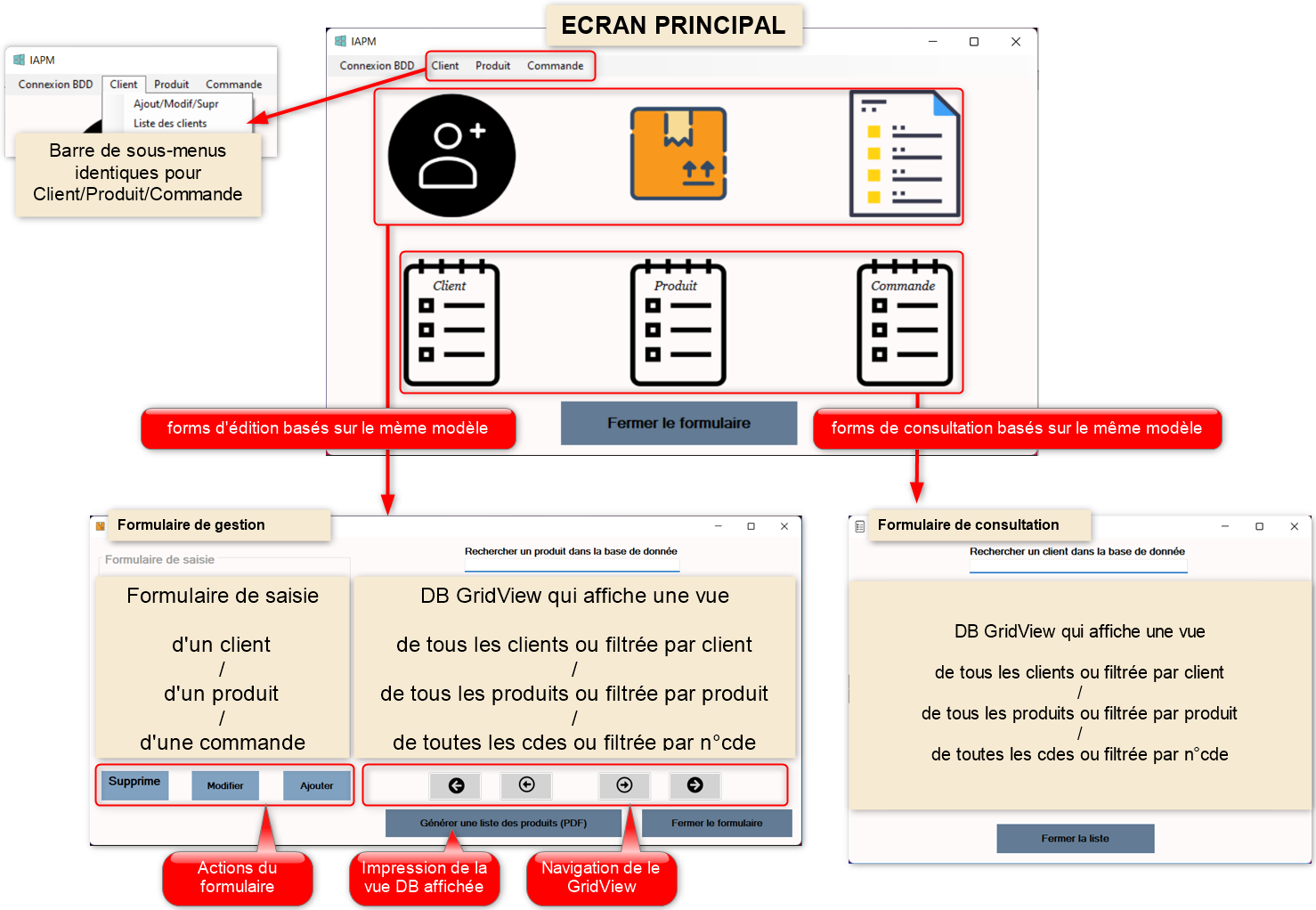
# 4 – Gestion de projet

## 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 organisé et surtout à ne pas me perdre dans la masse de travail importante.

J’ai également réalisé une **maquette** de l’application IAPM via le logiciel en ligne Figma, pour avoir une idée précise de l’état final du design, avant de me lancer dans la production de cette dernière.

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



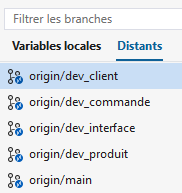
*(Pour plus d’annexes, voir Annexes pour IAPM – Annexe IAPM Maquette)*

J’ai également intégré la notion de **charte graphique** pour avoir une cohérence des effets visuels (graphique, design, forme des boutons, police) dans l’ensemble de 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 soit 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

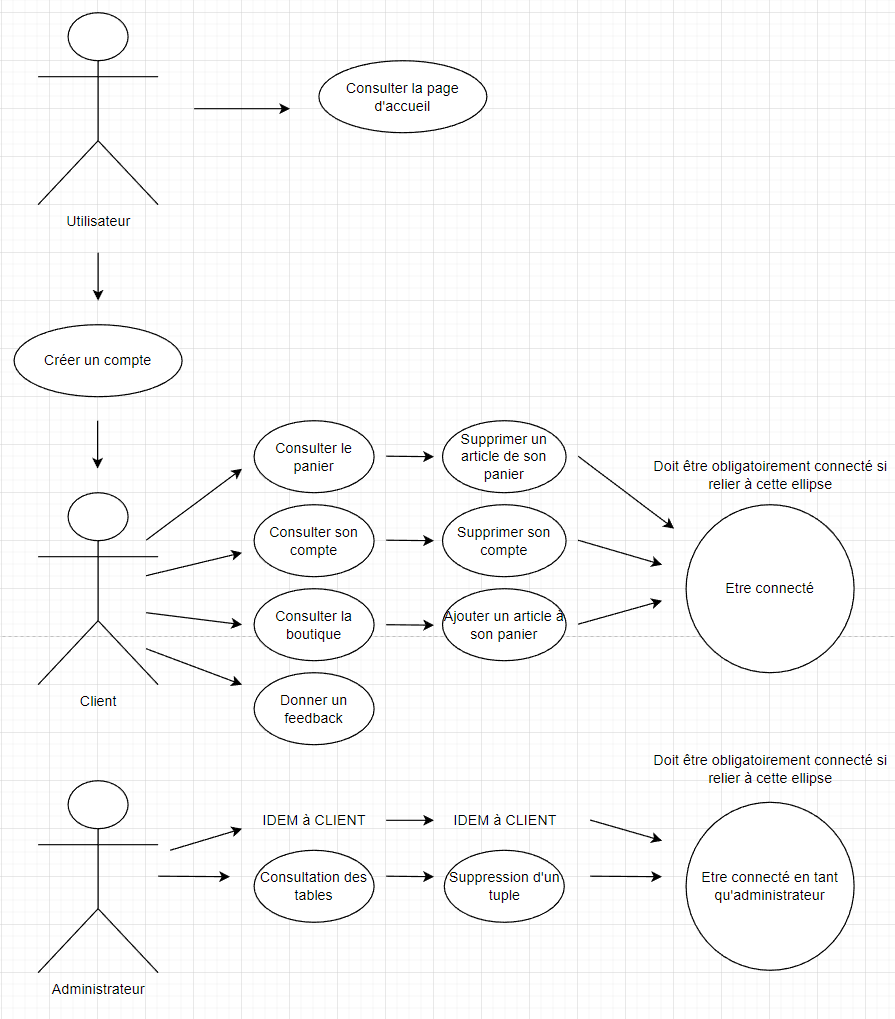
- Un administrateur peut cliquer sur l’icône de connexion pour les administrateurs pour s’y connecter

- 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

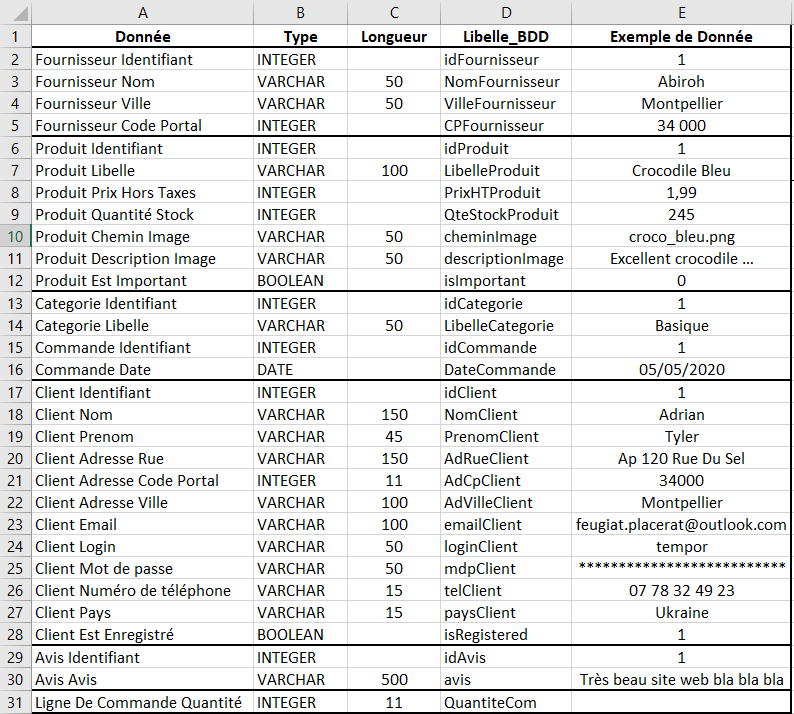
- Un administrateur peut de nouveau afficher toutes les tables quand il a affiché seulement une seule table

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 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.

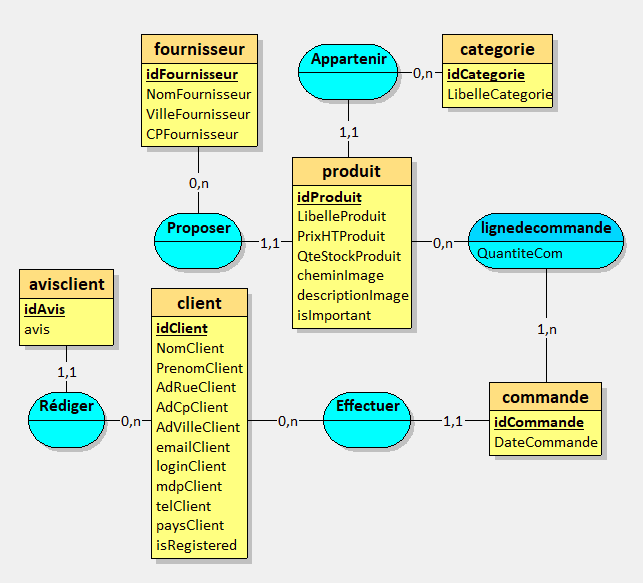
A présent, il est primordial de présenter le dictionnaire de donnée pour avoir une vue précise sur les différentes données manipulées. Ce précieux outil recense et organise toutes les informations qui sont stockés et manipulées par l’application. Le dictionnaire de données constitue une ressource essentielle 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.



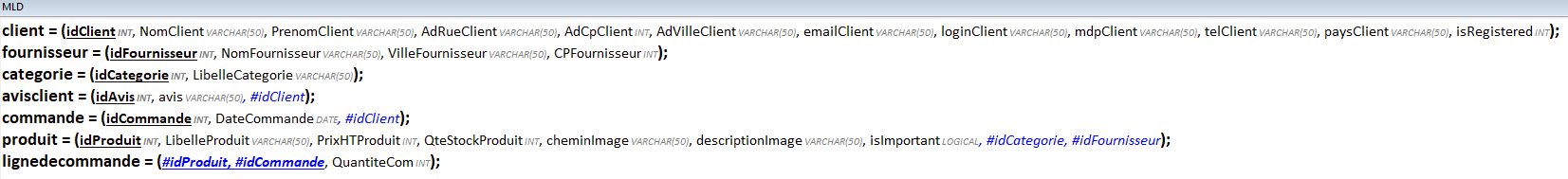
Maintenant, je vais vous présenter le Modèle Conceptuel de Données. Il a été réalisé sur l’éditeur de MCD Looping MCD, cet éditeur nous permet également de générer le MLD (Modèle Logique de Données), mais aussi le script de base de données pour l’intégrer dans l’interpréteur de script MySQL et donc générer 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 🡪 (exemple : produit)
* Les variables sont en lowercase 🡪 (exemple : text)
* Les méthodes adoptent le UpperCamelCase et doivent avoir une signification simple et verbeuse 🡪 (exemple : 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

Dans le cadre de la présentation de mon application desktop IAPM, 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 IAPM et comprendre comment ils interagissent pour offrir des fonctionnalités performantes et intuitives.

DIAGRAMME DES CLASSES de IAPM

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)*

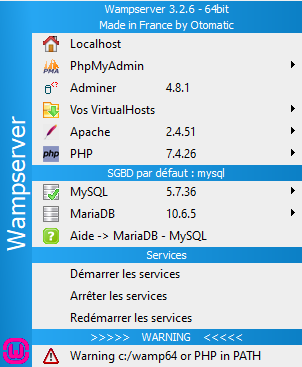
Ces procédures stockées et déclencheurs sont appelés dans l’application desktop IAPM pour récupérer les clients, les produits et les commandes.

## 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 respectées.

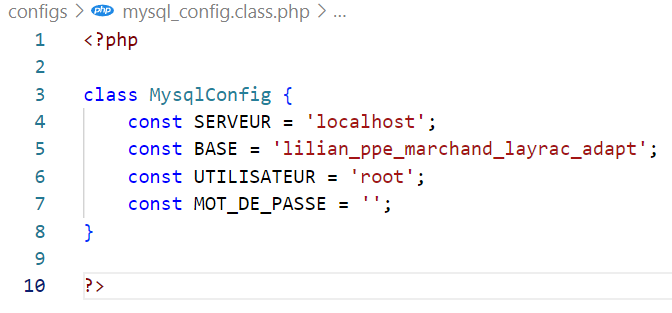
Les exigences de **l’interface utilisateur** sont que l’interface soit intuitive et facile de compréhension pour l’utilisateur. Il doit aller d’un point A à un point B simplement et efficacement sans se perdre sur le site, sinon il risque de le quitter et donc arrêter la navigation.

**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.

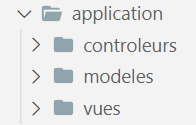
Dans cette classe, je redéfinis les variables stockées dans la classe MysqlConfig, ensuite, dans ma fonction seConnecter, je vérifie si la connexion à déjà été établie, si oui, la méthode ne se réexécute pas, cependant, s’il n’y a pas de connexion établie, j’établie ma chaîne de connexion avec les informations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

L’utilisation de la classe modèle PDO permet de se prémunir d’injections SQL par le biais des requêtes préparées.

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.

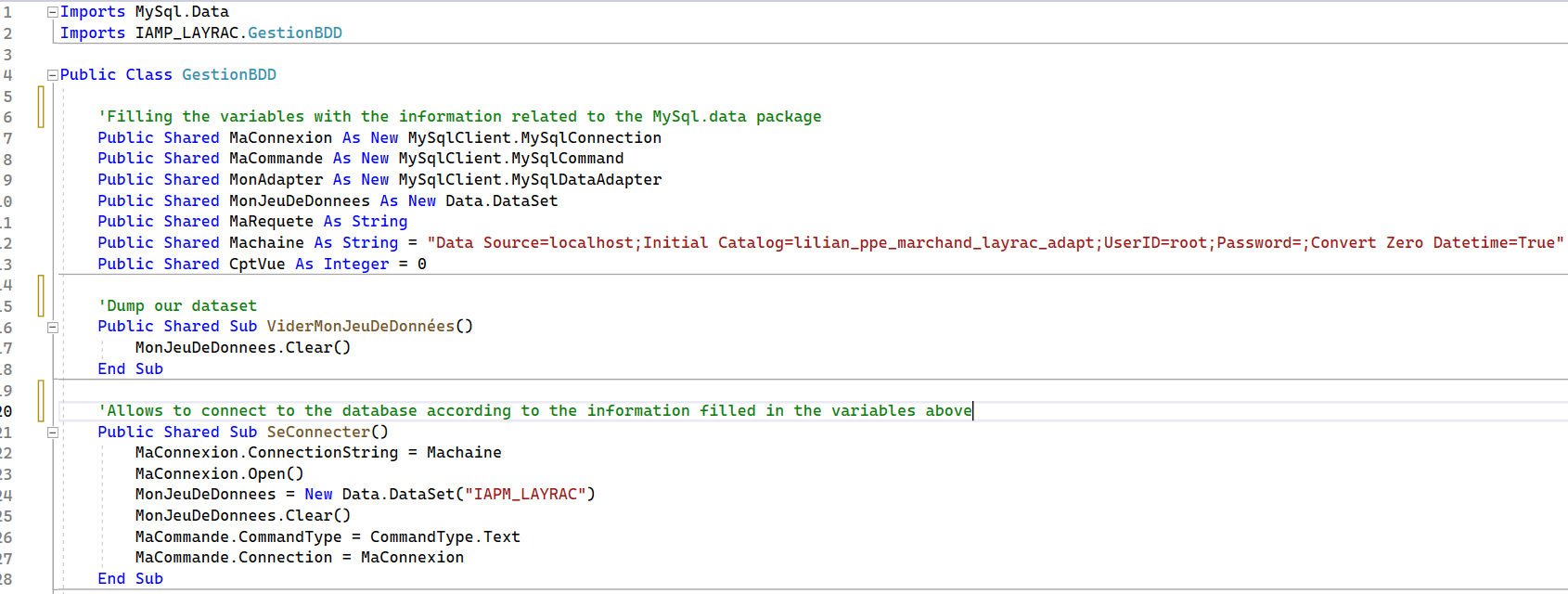
Pour la plateforme, j’ai choisi le langage **Visual Basic** pour le développement. Pour le design, j’ai utilisé **WinForm**, un framework de design proposé dans le package **.NET Core**.

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 continuellement avoir accès aux données.

## 6.3 – Application Mobile

# 7 – Réalisations du candidat

## 7.1 – Application StoreSwap | IAPM

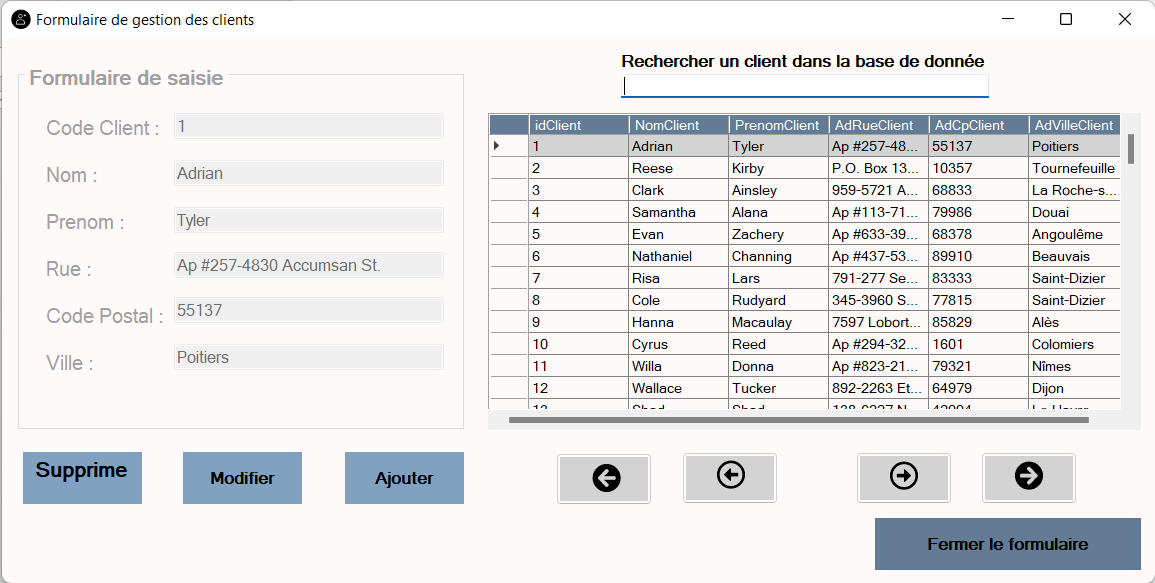
Choix du style pour StoreSwap :

Le choix du style pour cette application web à été très simple et efficace. Partons d’une thématique avec un client, s’il veut une boutique de bonbon, il faut plutôt des tons claires, c’est pour cela que j’ai choisis la prédominance de la couleur saumon. Pour ce qui est du design de l’interface, j’ai essayé de la faire la plus intuitive possible avec le moins de clique à faire pour accéder à l’information attendu par l’utilisateur. L’accueil est simple, épuré, regroupe les meilleurs ventes, les page de connexion et d’inscription sont basiques. La page boutique est simple et compréhensible, nous avons le prix du produit, sa quantité, son image et un bouton pour l’ajouter au panier.

Choix du style pour IAPM :

Le choix du style pour l’application desktop IAPM est totalement contraire au site web, de ce fait, j’ai choisis certes, une palette de couleur pâle, mais plus stylisé pour des applications bureau. J’ai donc utilisé du bleu pour les boutons, et des tons gris claire pour les fonds. Partie graphique, il fallait que toutes les informations soient affiché, par exemple, si l’on veut un client, en tant qu’utilisateur, je veux l’information la plus rapidement possible pour travailler efficacement, c’est pour cela que j’ai rendu disponible toutes les informations des client sur une seule page.

Ici, voici l’interface de gestion de clients :

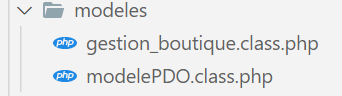


*Pour plus de screenshot de* l’application *(Voir Annexes pour IAPM – Annexe IAPM App « Page »)*

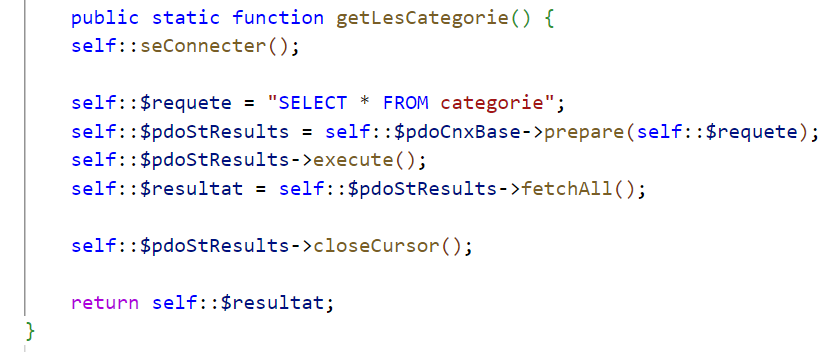
Coté base de données :

Maintenant, nous allons évoquer le sujet de la base de donnée, j’ai réalisé la base de donnée en commençant par le MCD sur le logiciel LoopingMCD, j’ai ensuite exporter le script et l’ai introduit pour créer ma base de données dans le logiciel d’édition de base de données HeidiSQL (C’est un logiciel gratuit). Cette base de donnée à été adaptée tout au long du projet, pour y rajouter des procédures stockées et des déclencheurs. (Toujours dans cet optique d’optimisation du code et effectuer les requêtes coté serveur).

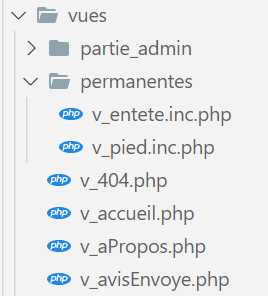
J’évoquerais plus bas le cas d’utilisation d’une procédure stockée dans l’application desktop.

Comment est construite l’application StoreSwap :

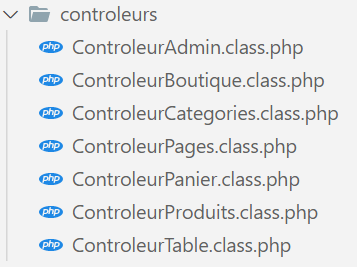
L’application web StoreSwap adopte une architecture MVC, premièrement, nous avons la couche model, c’est la couche qui regroupe les principales classes de notre application. (En l’occurrence, nos classes munis de nos méthodes pour accéder aux données).



Voici une méthode permettant la récupération des catégories stockées en base de données.

Ensuite, nous avons la couche vue, qui regroupe toutes les vues (templates) de notre application, c’est ce qui gère notre affichage.

Ci-dessous, voici un bout de code de la vue permettant l’affichage de la boutique.

Enfin, nous avons la couche contrôleur, il nous permet de gérer les interactions entre le modèle et la vue.

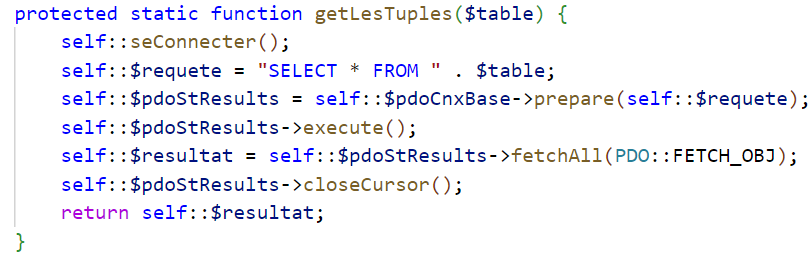
Voici une fonction dans un contrôleur permettant la gestion des erreur (page 404)



*A noter que les dossiers Modèle, Vue et Contrôleur sont tout trois stockés dans le dossier Application.*

Sécurité de l’application StoreSwap :

La classe modèle PDO (PHP Data Objects) est utilisée pour sécuriser les requêtes dans les sites web PHP. PDO fournit une interface uniforme pour accéder à différentes bases de données, ce qui permet de réduire les vulnérabilités potentielles. En utilisant PDO, vous pouvez utiliser des requêtes préparées, qui sont des requêtes SQL avec des paramètres de substitution. Cela empêche les attaques par injection SQL, car les paramètres sont automatiquement échappés et évite les problèmes de concaténation de chaînes. De plus, PDO prend en charge les transactions, ce qui vous permet d'exécuter des opérations de base de données de manière atomique, garantissant ainsi l'intégrité des données. En utilisant PDO, vous pouvez améliorer la sécurité de votre site web PHP en évitant les erreurs courantes liées à la manipulation directe de requêtes SQL.



Nom de la fonction et son paramètre associé (Table).

Exécution de la requête.

Récupération des objets avec l’attribut fetchAll.

Affichage du résultat.

Préparation de la requête avec la fonction « prepare ».

Libération des ressources en indiquant à PDO que le travail est finit.

Méthode de connexion à la base de données.

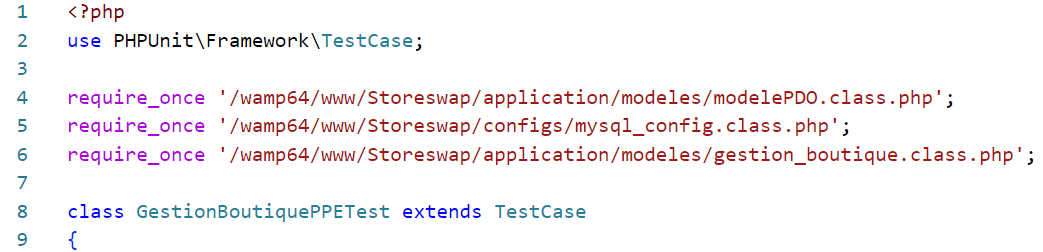
Requête SQL avec la variable $table passée en paramètre.

Tests Unitaires :

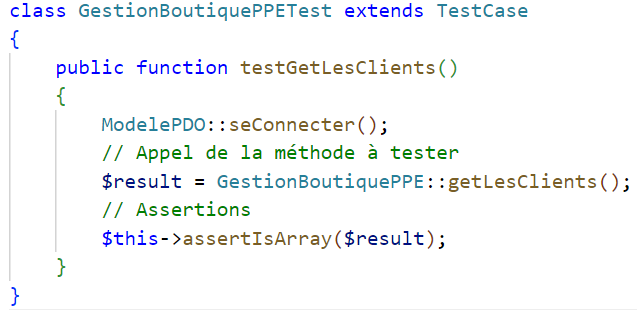
Les **tests unitaires** sont une pratique essentielle dans le développement logiciel pour garantir la qualité et la fiabilité des applications. Ils permettent de vérifier que chaque composant du code fonctionne correctement de manière isolée, en testant individuellement ses fonctionnalités.

Dans notre projet, nous avons utilisé un framework de tests appelé **PHPUnit**, qui est spécialement conçu pour PHP. PHPUnit fournit des fonctionnalités et des outils pour écrire et exécuter facilement des tests unitaires.

Nous avons créé une classe de test appelée **`GestionBoutiquePPETest`**, qui contient des méthodes de test pour vérifier le bon fonctionnement de notre application. Chaque méthode de test correspond à une fonctionnalité spécifique que nous souhaitons tester.



Dans notre classe, nous avons écrit une méthode de test nommée **`testGetLesClients`** pour vérifier la fonction **`getLesClients`** de notre application. Cette fonction est responsable de la récupération des clients.



**3**

**2**

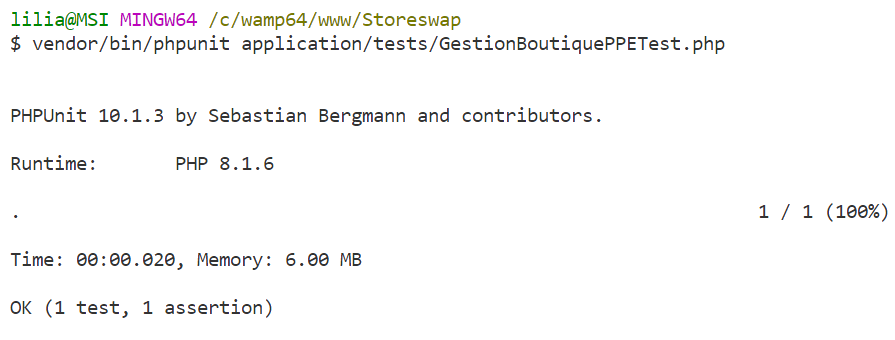
**1**

Je vais vous expliquer chaque composants de la méthode en se référant à l’image ci-dessus.

Pour ce test, nous avons mis en place les préparatifs nécessaires, tels que la connexion à la base de données, afin de s'assurer que le test est exécuté dans un environnement approprié. **(1)**

Ensuite, nous avons appelé la fonction **`getLesClients`** **(2)** et vérifié que le résultat retourné est un tableau à l'aide de l'assertion **`$this->assertIsArray($result)` (3).** Cela nous permet de nous assurer que la fonction renvoie bien les clients sous forme de tableau, comme attendu.

L'exécution des tests unitaires se fait à l'aide d'une commande spéciale, `vendor/bin/phpunit`, qui lance tous les tests définis dans notre application. Nous obtenons un rapport de test détaillé indiquant le nombre de tests exécutés, le nombre d'assertions réalisées et les éventuelles erreurs rencontrées.



Après avoir exécuté les tests, j'ai obtenu un rapport détaillé de PHPUnit. Il m'a indiqué que tous les tests ont été exécutés avec succès, ce qui signifie que mon application a passé avec succès toutes les vérifications que j'ai définies.

Le rapport affiche également des informations utiles, telles que la version de PHPUnit utilisée (ici la version 10.1.3), la version de PHP (8.1.6), le temps d'exécution total des tests (0,019 secondes) et la consommation mémoire (6,00 MB).

En utilisant cette approche de tests unitaires, j’ai pu identifier rapidement d'éventuels problèmes dans mon application et les corriger avant qu'ils ne se propagent à d'autres parties du code. Cela m’a permis de garantir la qualité et la fiabilité de mon application tout au long du processus de développement.

Comment est construite l’application IAPM :

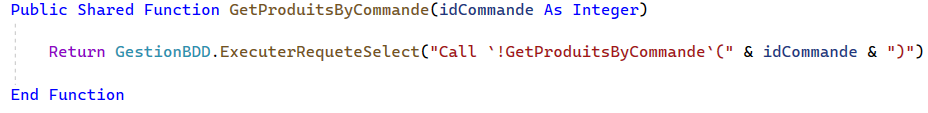
L’application IAPM est composée de classes et de fichiers WinForm permettant la modélisation graphique avec le framework intégré au package .NET.

Vous pouvez retrouver l’arborescence du projet en annexe.

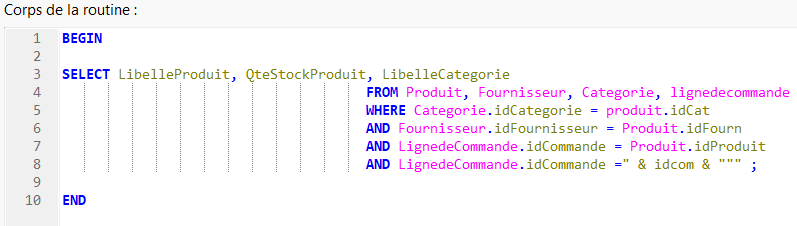
*(Voir Annexes pour IAPM – Annexe IAPM App « Page »)*

Lorsque l’application effectue une requête à la base de donnée, elle appel une requête stockée en base de données, via une procédure stockée, en voici les images :

Voici la procédure stockée appelée dans le code en fonction de l’id de la commande passée en paramètre :

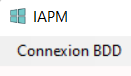


Voici le contenu de la procédure stockée en base de données :



Sécurité de l’application IAPM :

Pour la sécurité de l’accès à la base de donnée pour l’application IAPM, j’ai obligé le fait que l’utilisateur doit se connecter à la base de données avec ses identifiants MySql. Sinon, il n’a pas accès à l’application.



## 7.2 – Application Mobile

X Conclusion

X Glossaire

IAPM : Interface Applicatif de Produits Marchands

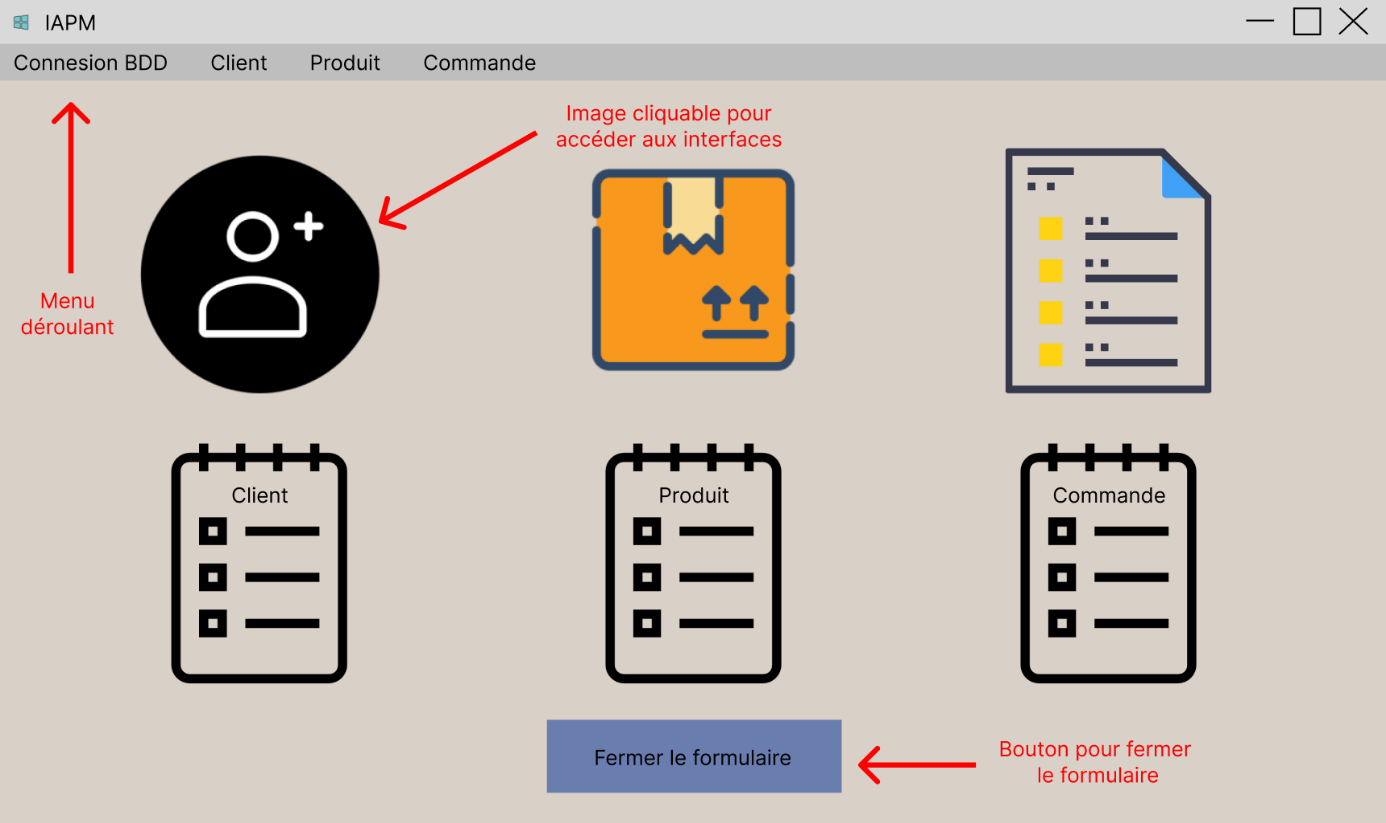
MVC : Model-View-Controller

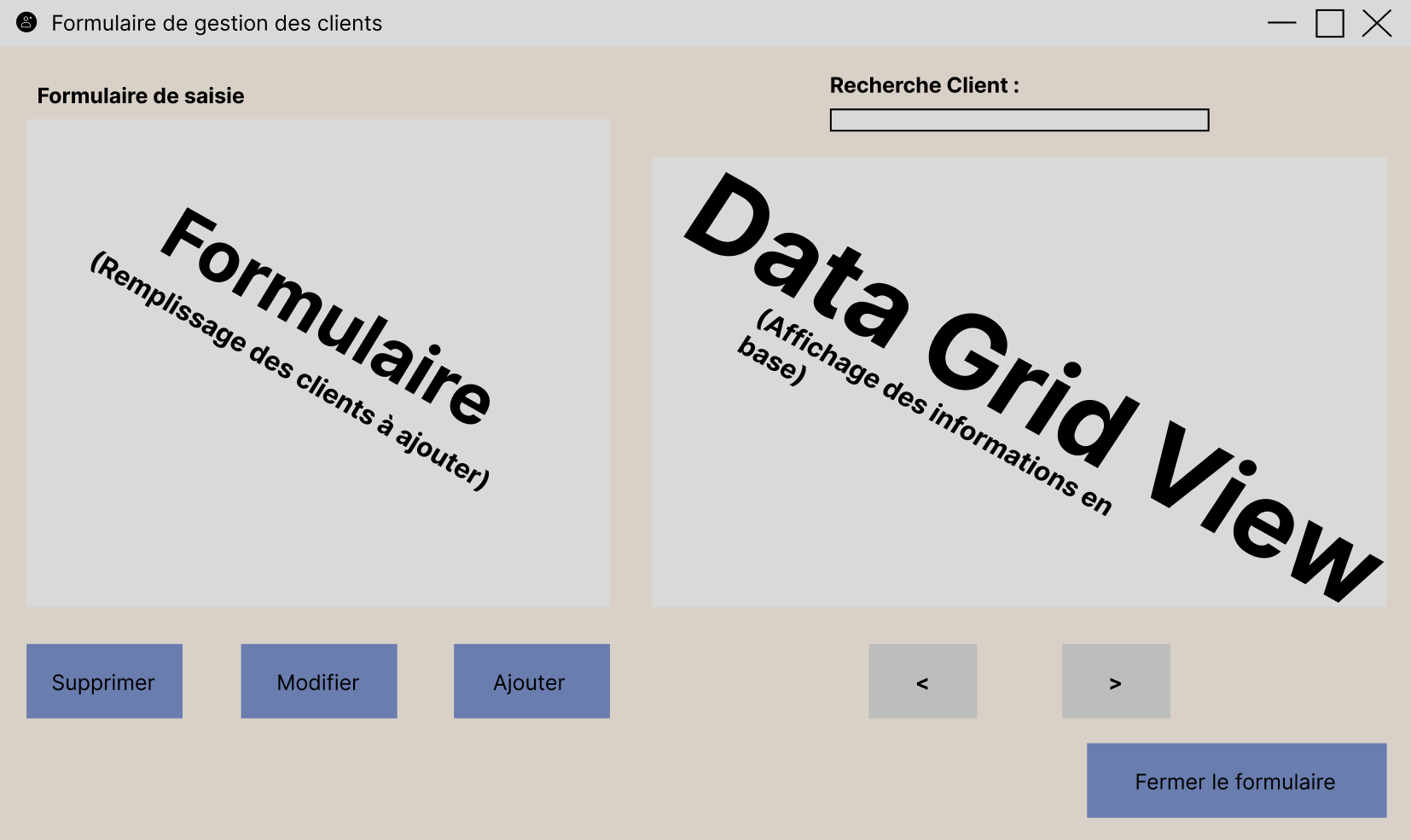
MVVM : Model-View-ViewModel

X Annexes

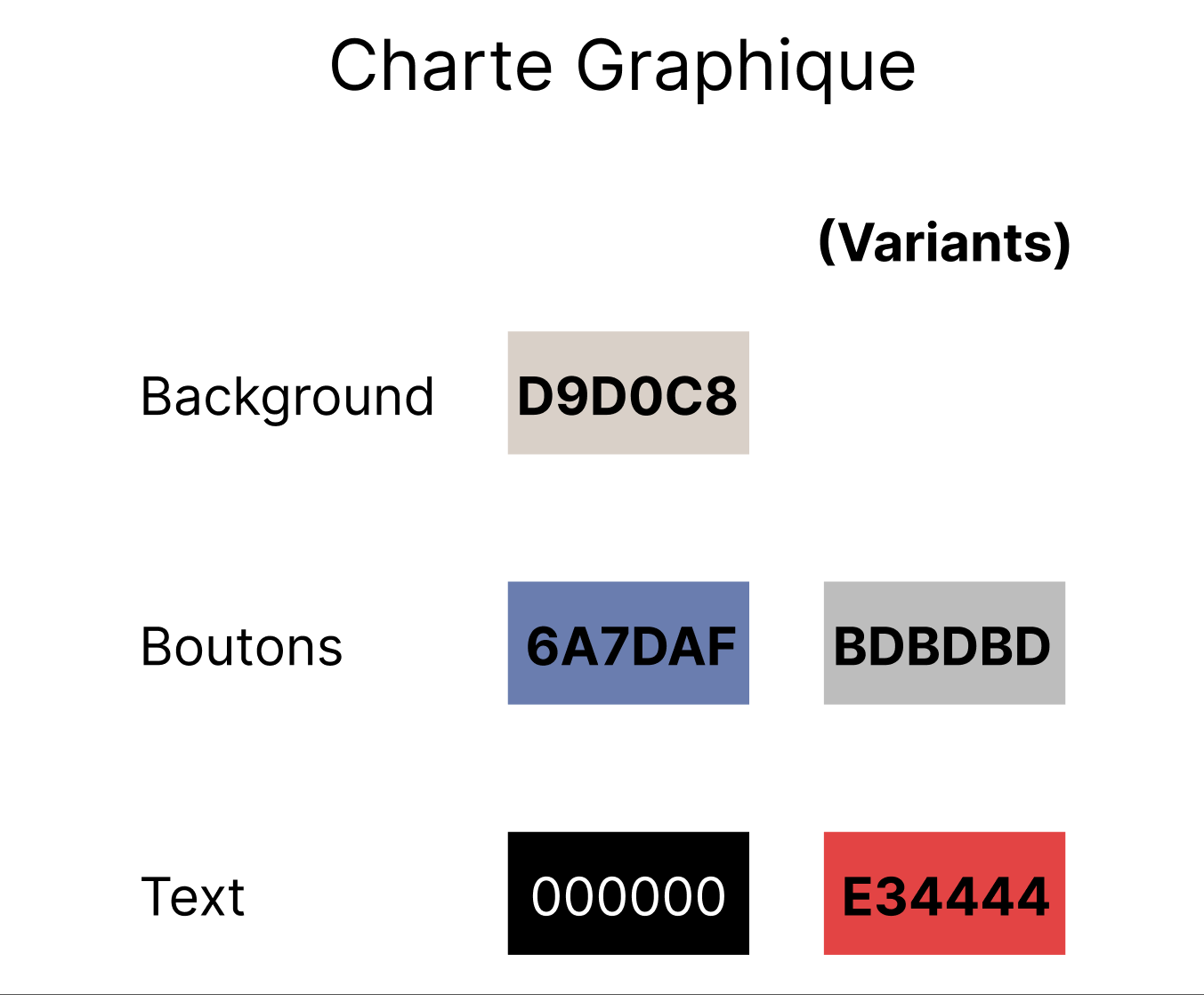
Annexes pour IAPM

Annexe IAPM Maquette

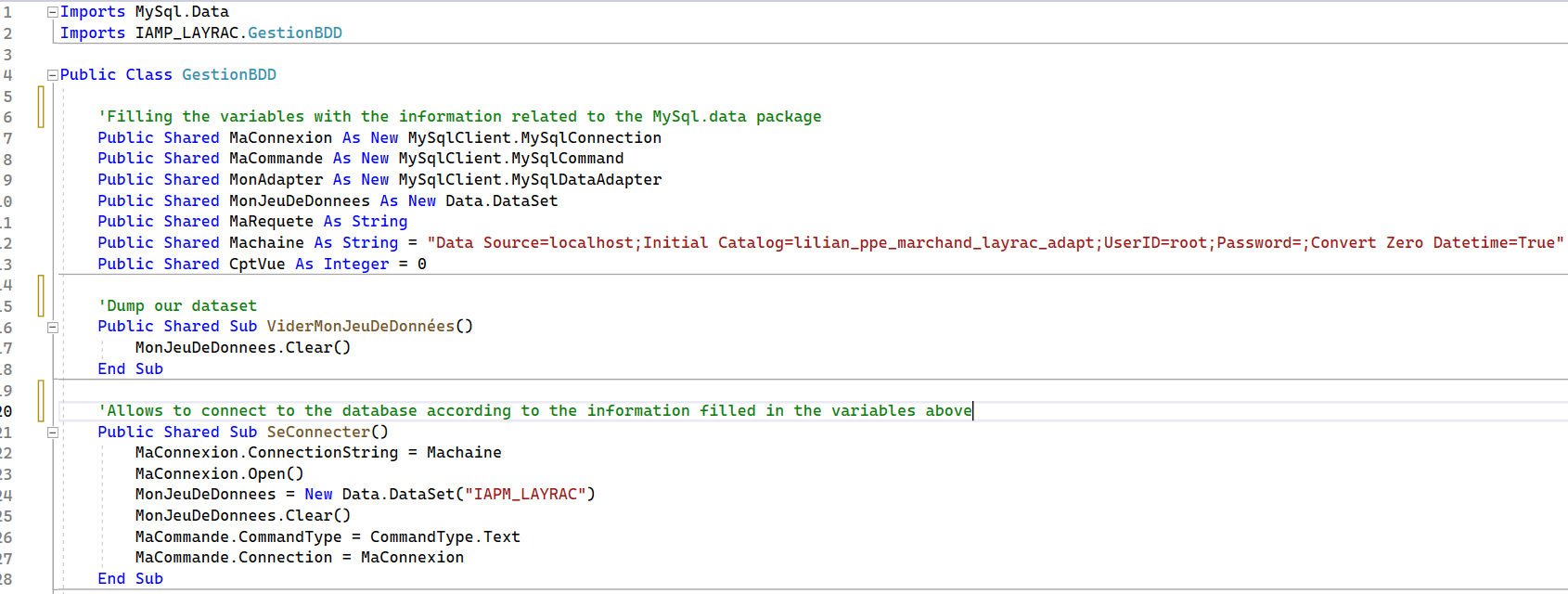




### Annexe IAPM Charte Graphique

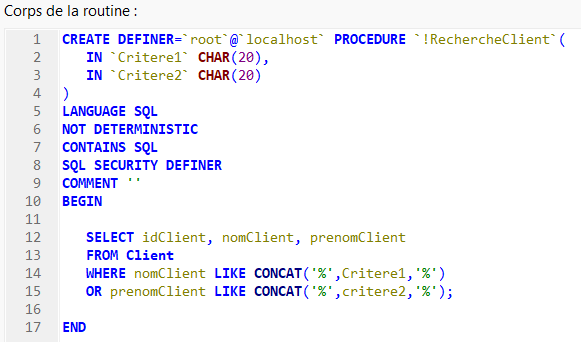


Annexe IAPM Connexion

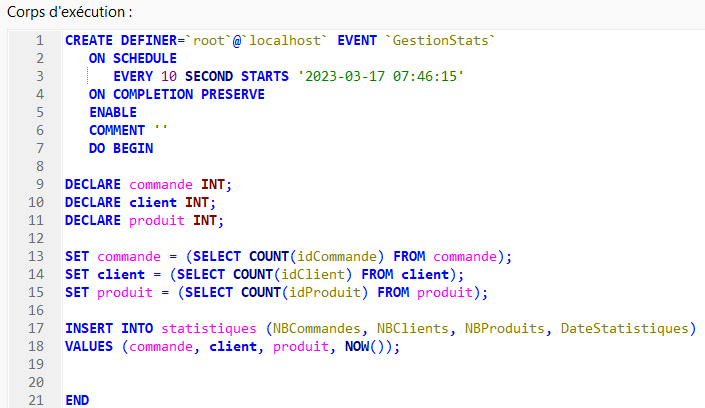


Annexe IAPM BDD

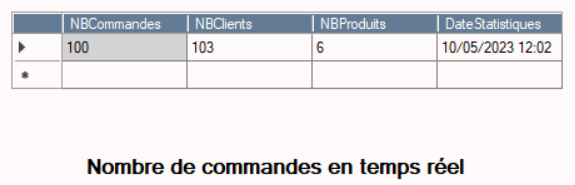
Une procedure stockée qui effectue la recherche d’un client :



Un déclencheur qui s’occupe de la gestion des statistiques :

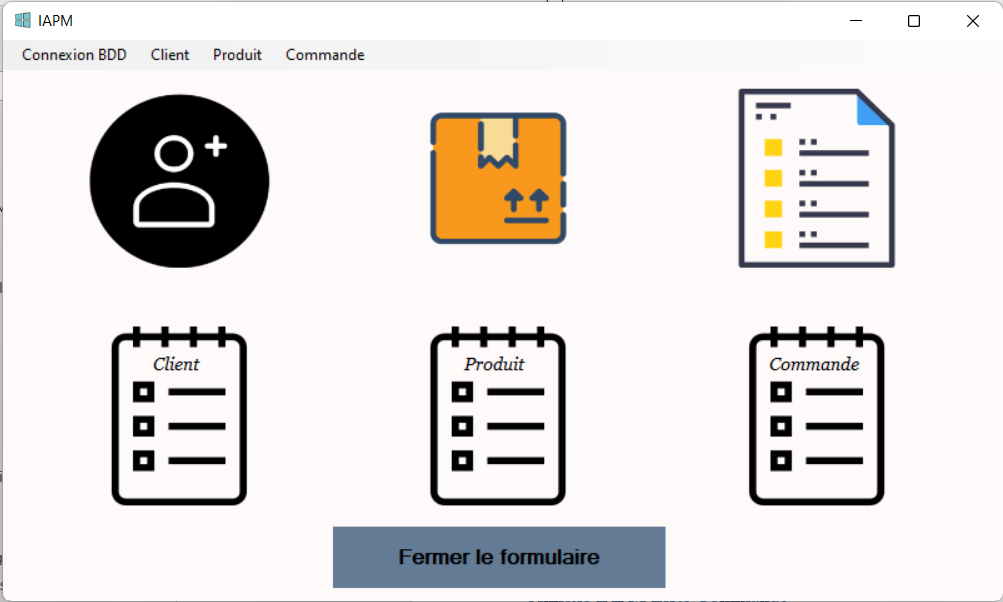


Annexe IAPM Résultat Trigger

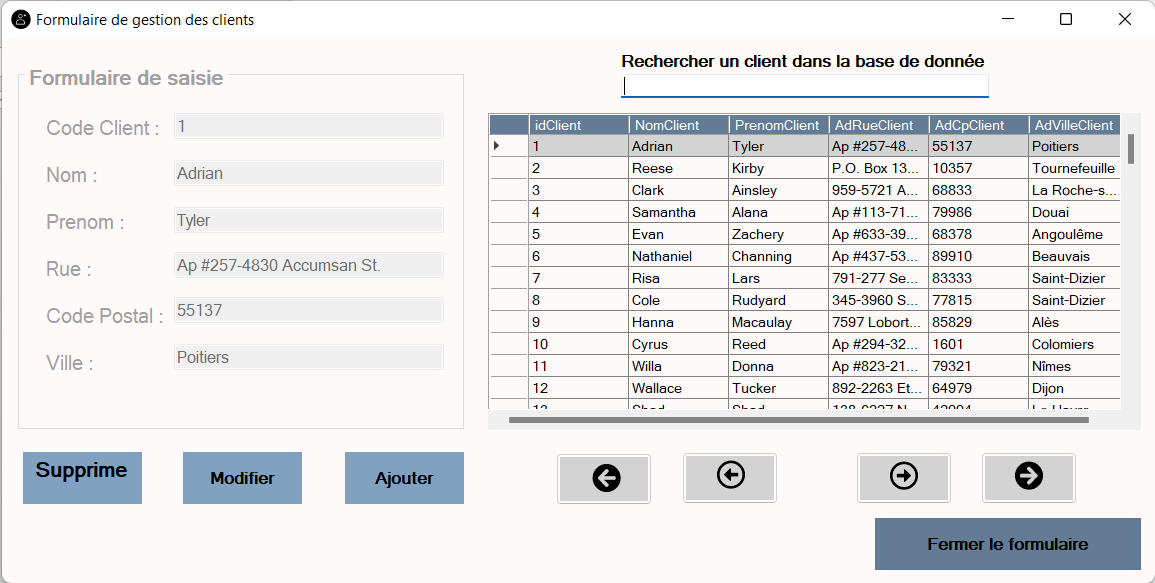


### Annexe IAPM Tests Unitaires

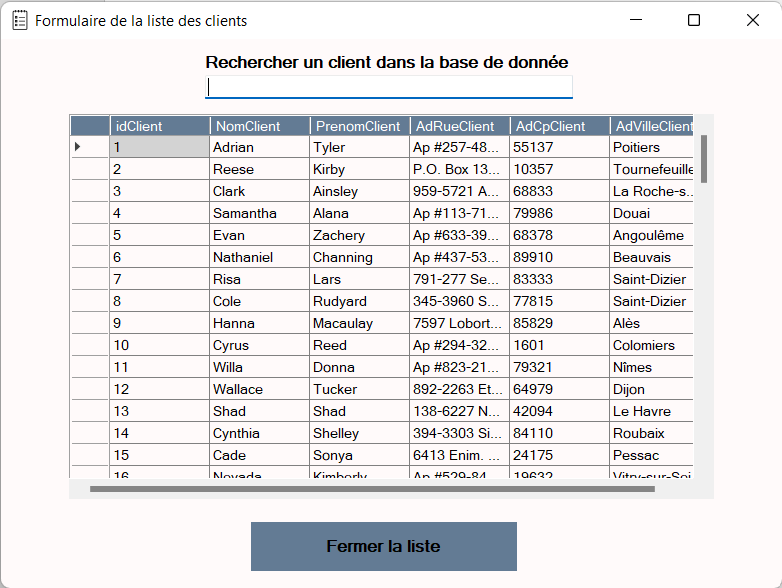
### Annexe IAPM App Menu



### Annexe IAPM App Ajout Client



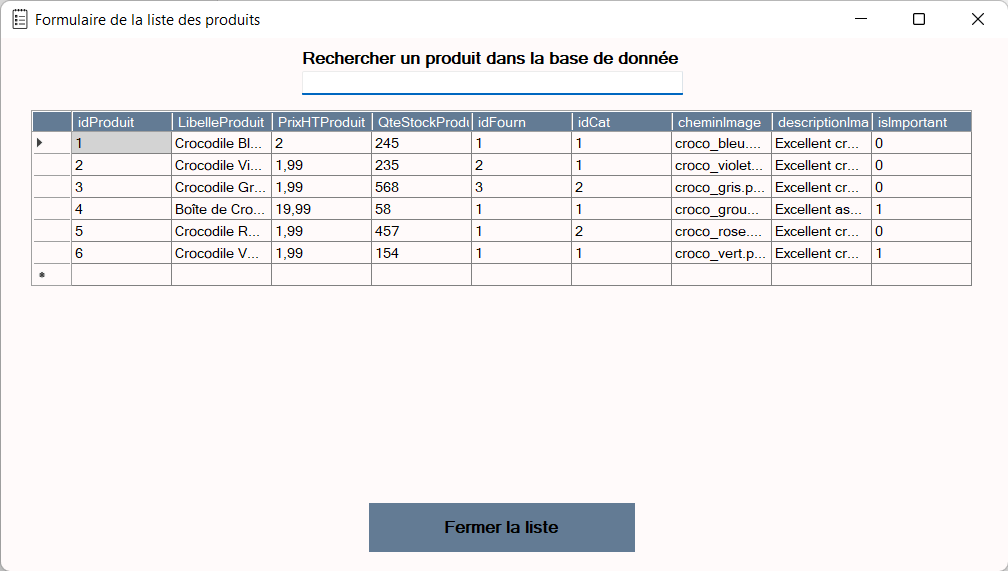
### Annexe IAPM App Liste Client



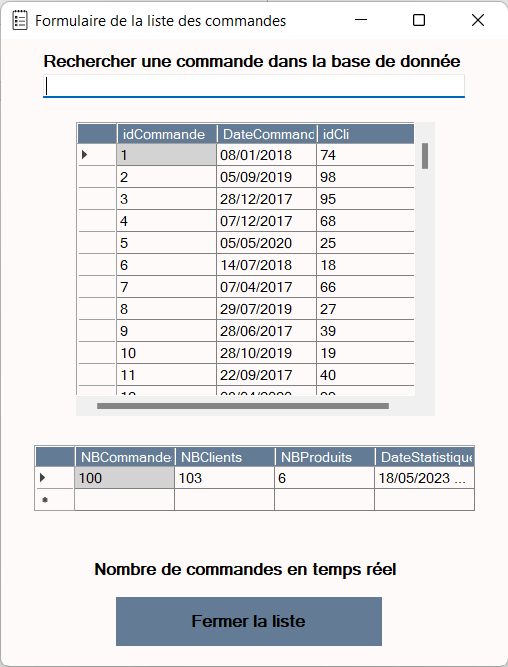
### Annexe IAPM App Ajout Produit



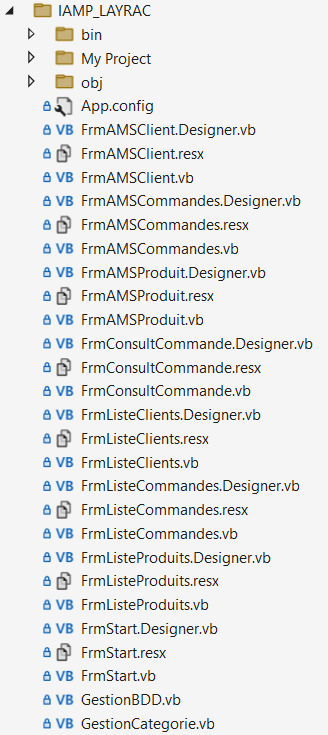
### Annexe IAPM App Liste Produit



### Annexe IAPM App Liste Commande



### Annexe IAPM Arborescence

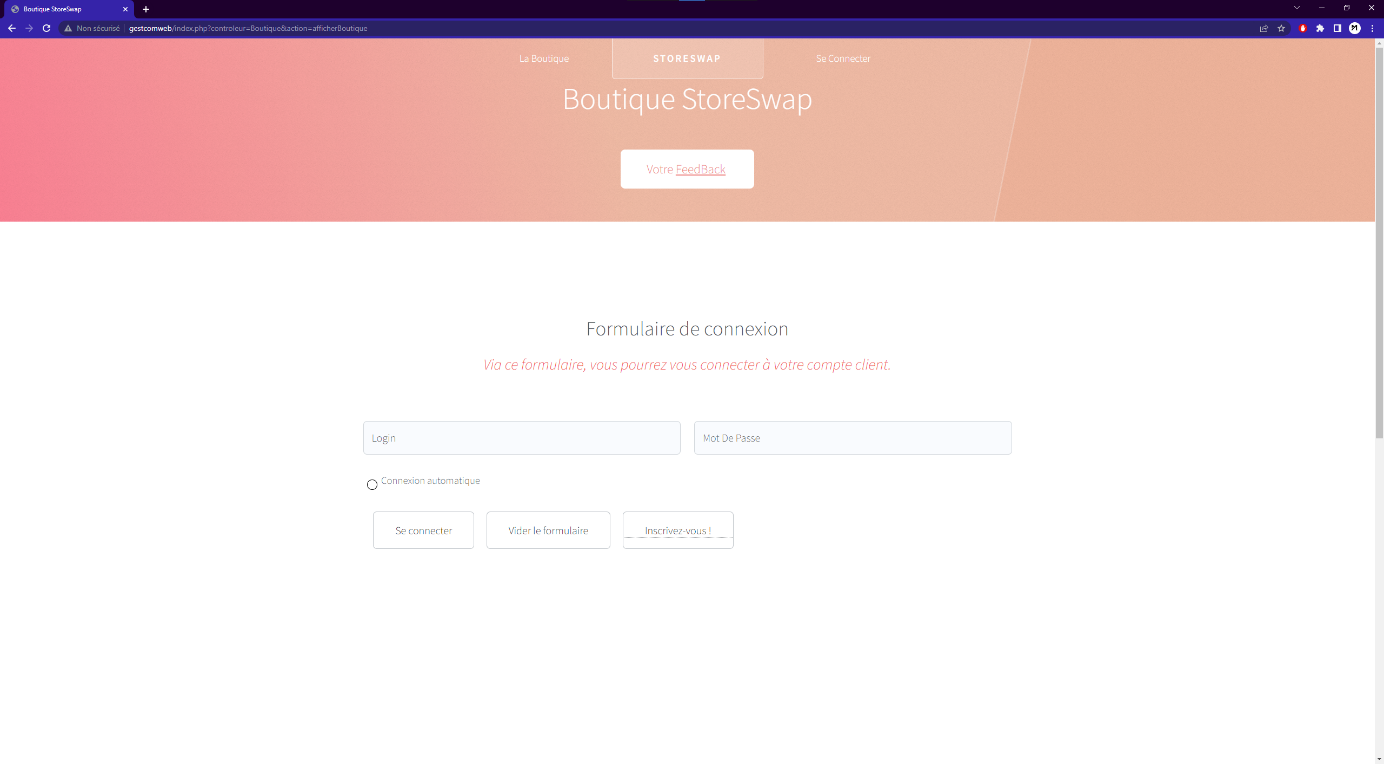


Annexes pour StoreSwap

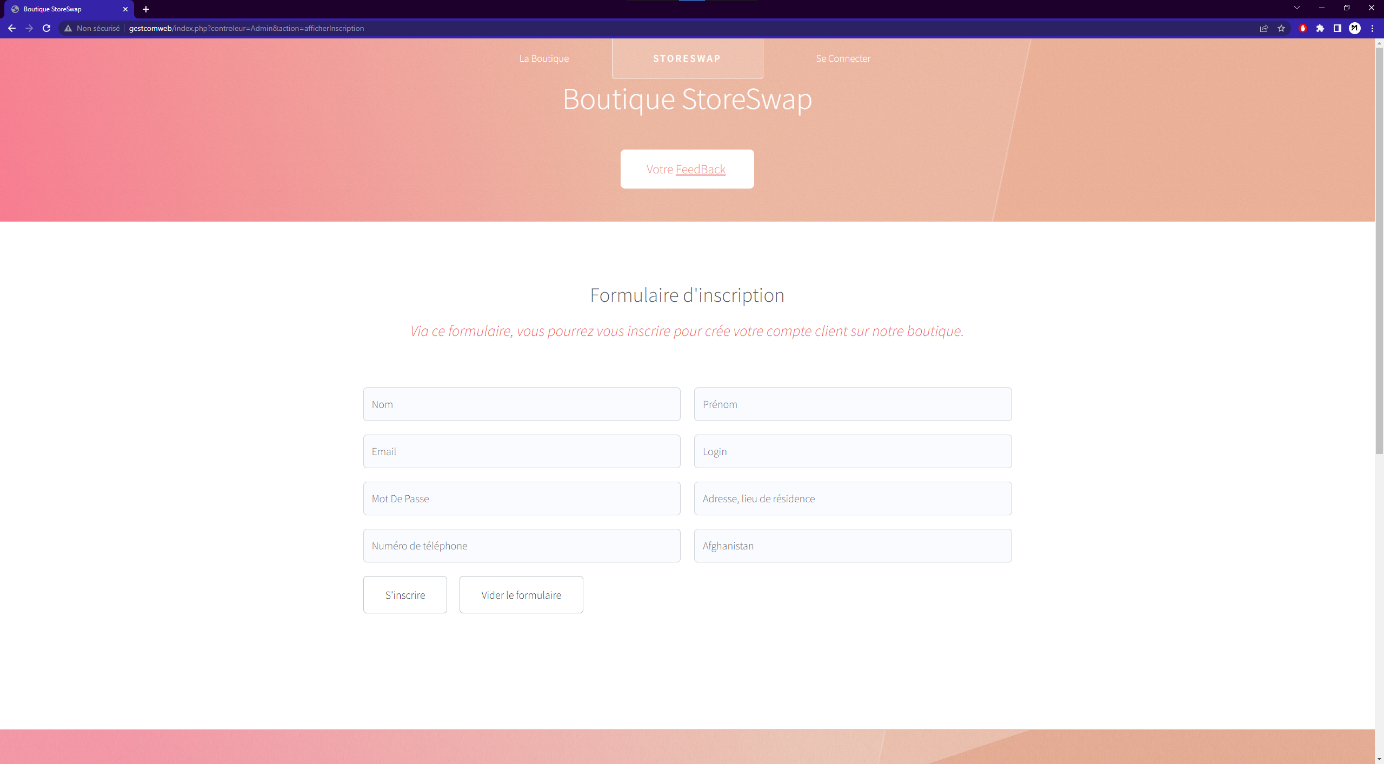
### Annexe StoreSwap Accueil



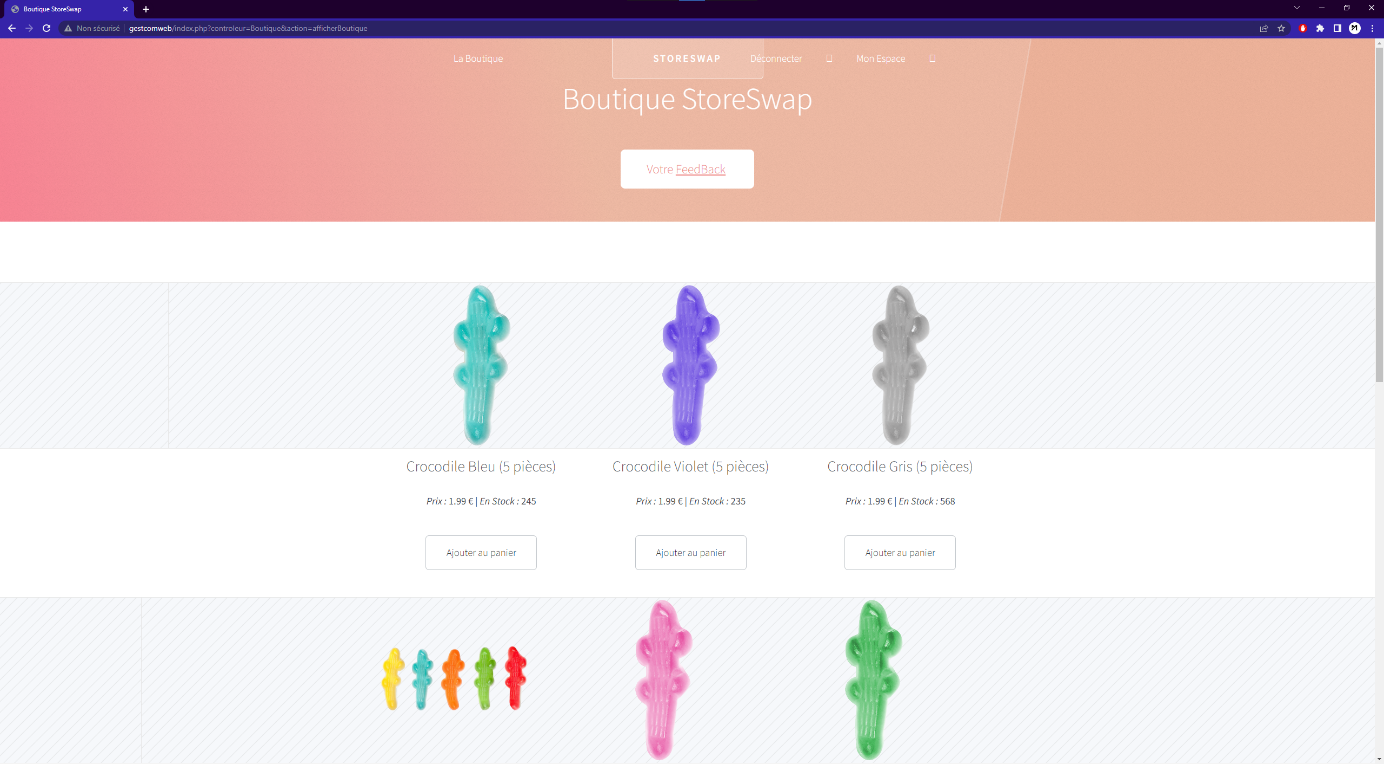
### Annexe StoreSwap Connexion



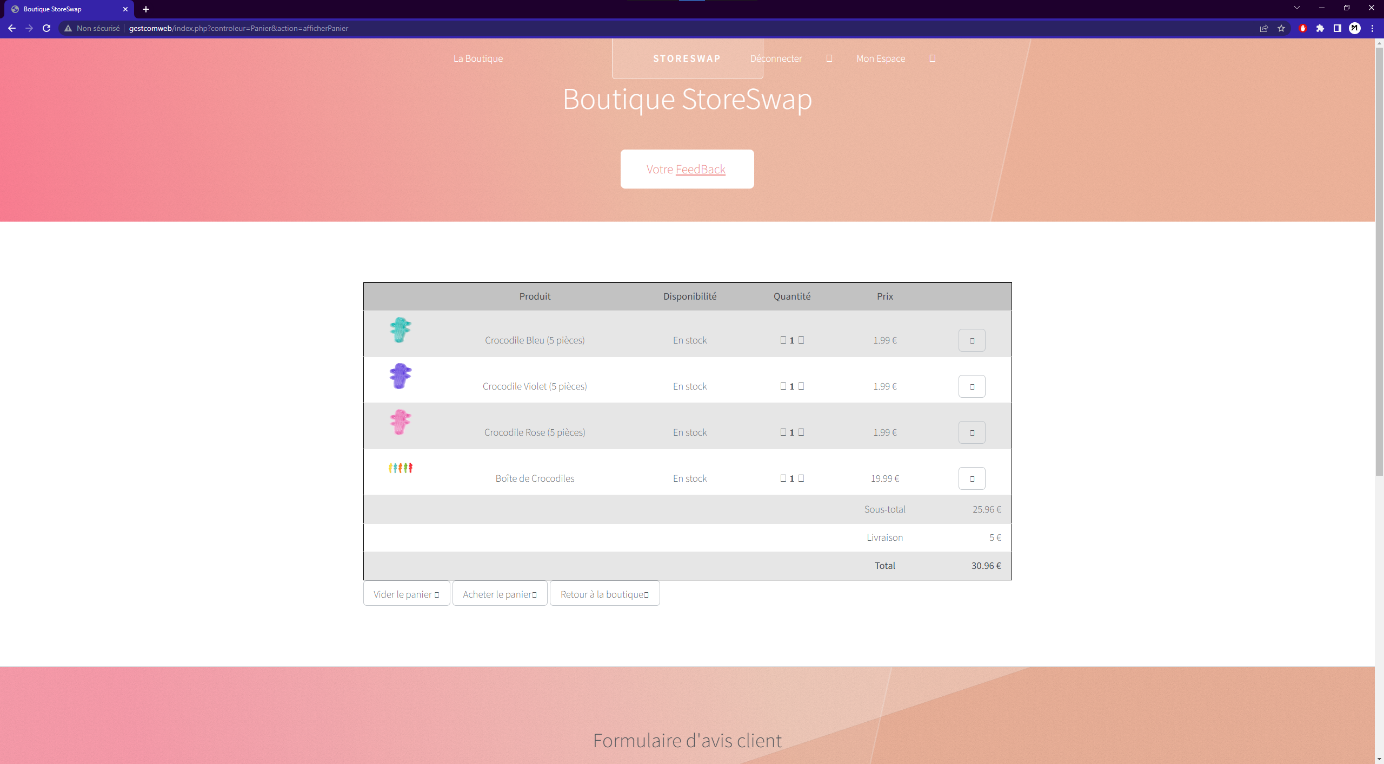
### Annexe StoreSwap Inscription



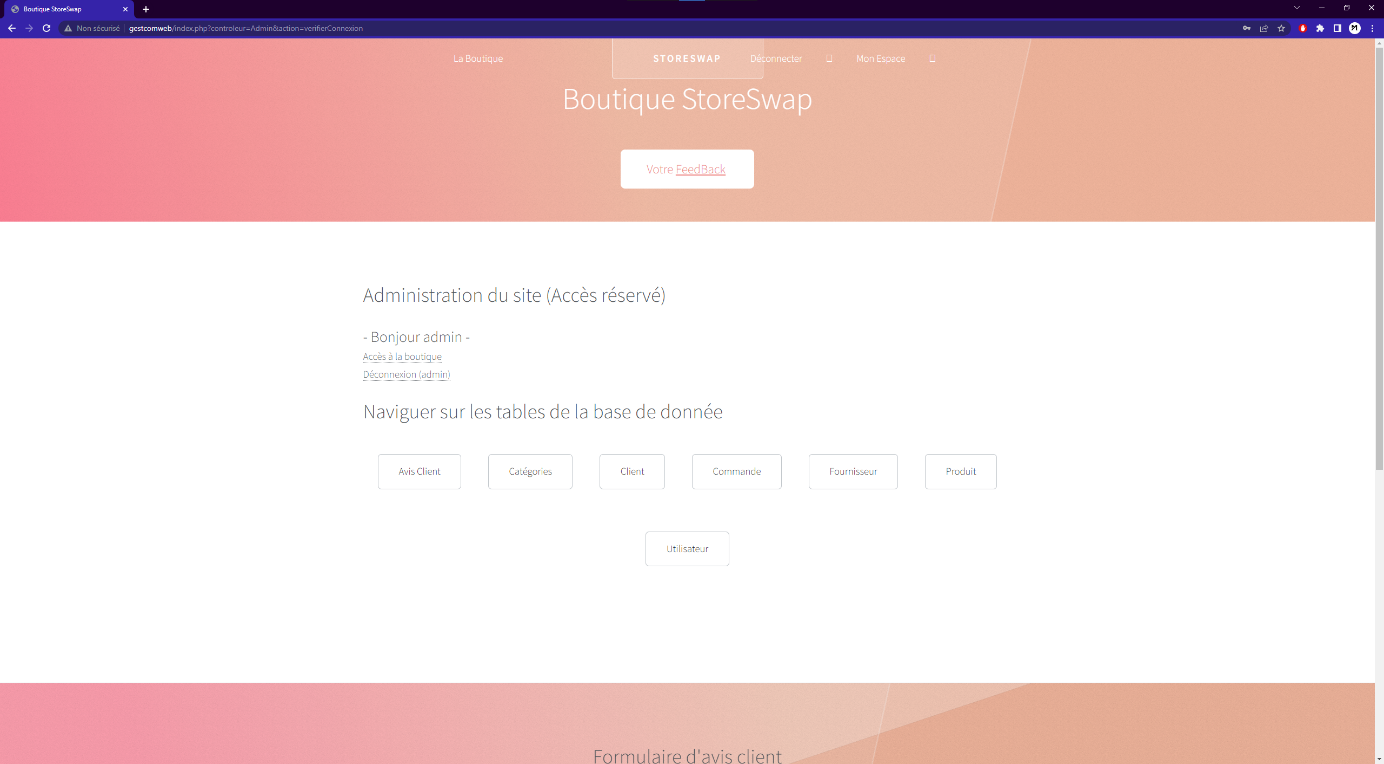
### Annexe StoreSwap Boutique



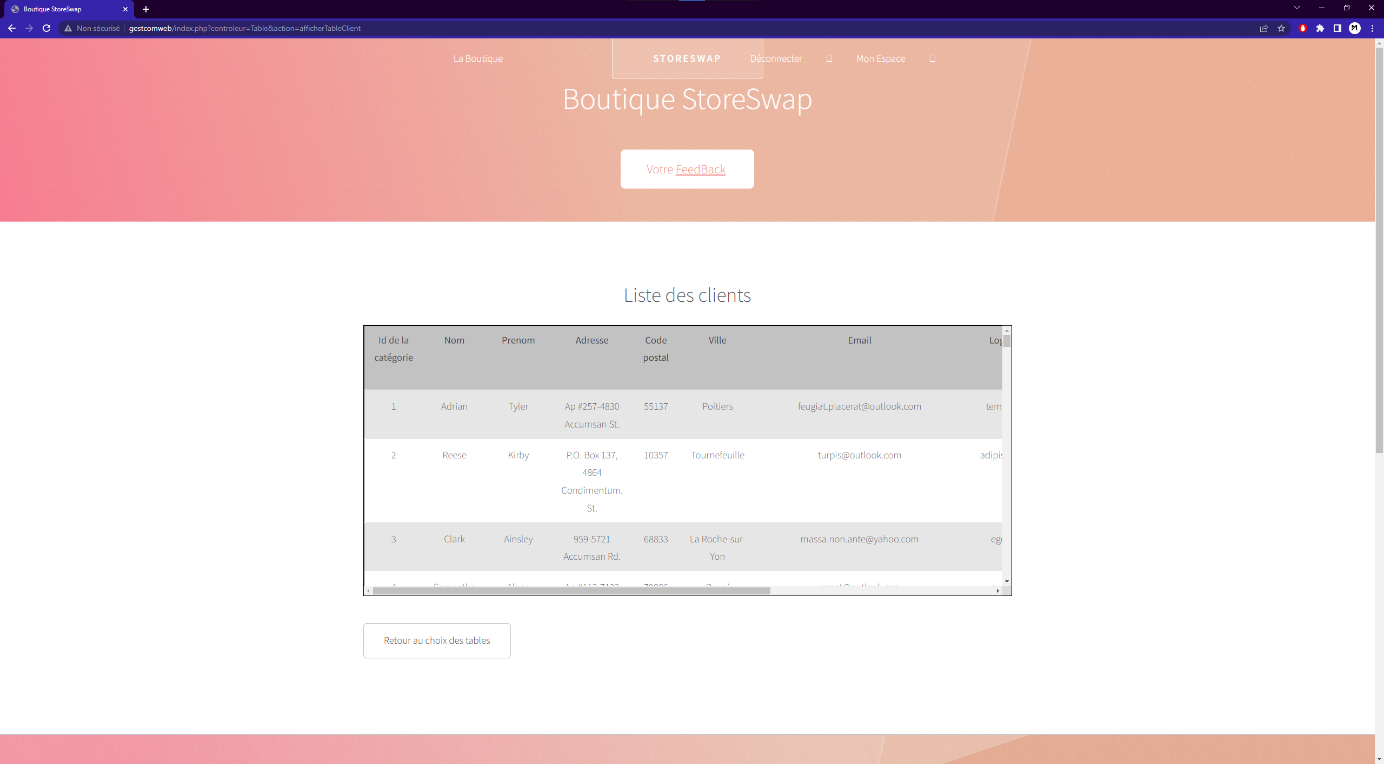
### Annexe StoreSwap Panier



### Annexe StoreSwap Admin Table



### Annexe StoreSwap Admin Table Client



### Annexe StoreSwap Code Connexion

