MyProfits

Présentation de la solution

MyProfits est une application web permettant d'agréger des dépenses publicitaires (Facebook) et des gains de magasins en ligne (Shopify), afin d'obtenir :

- Des dashboards en temps réel
- Des données analytiques (en ligne ou excel)
- Une gestion des plateformes publicitaires (Facebook)
- Une gestion des magasins en ligne (Shopify)
- Une gestion des produits vendus

La solution intègre une interface d'administration et de gestion des utilisateurs.

Descriptif technique

MyProfits est écrit en :

- **Net Core** pour la partie serveur,
- **ReactJS** pour la partie client

Elle s'appuie sur une **base de données relationnelles (MySQL).**

Cela en fait une application portable sur n'importe quelle plateforme (Linux, Windows, Cloud, …).

L'application est architecturée autour de quatre concepts clés :

- Le Domain Driven Design
- Le pattern Repository
- Code First

La solution est décomposée en **cinq projets** présentés ci-après, chaque projet va jouer un rôle spécifique dans le traitement des données provenant de Shopify, Facebook ou de MyProfits.

MyProfits.Client

L'application Web, le client, est une application SPA développée en **ReactJS / Redux**.

Ce projet intègre aussi un **serveur Web indépendant** en **.Net Core** dont le rôle est de fournir les fichiers de scripts, images, css, html

Les scripts se trouvent dans le répertoire *component*, la partie *reducer* se trouve dans le repertoire *store*.

Principales bibliothèques Javascript

- **Highcart** : Pour le tracé des graphiques (dashboards)
- **ReactStrap** : Bootstrap v4 pour react, pour le design de l'application
- **ReactJS** : Bibliothèque Javascript facilitant le développement des Sites web **monopage** (SPA)
- **Redux** : Bibliothèque de gestion d'état pour React

MyProfits.API

La partie API offre un ensemble de **Web Service** permettant d'accéder aux données et d'effectuer certaines opérations (mise en pause de campagnes Facebook, synthèse des données pour la partie analytics,…)

Les APIs sont accessibles via des URL dont la nomenclature de base est : https://myprofits.osires.fr/api/CONTROLLEUR/METHODE.

Par exemple pour récupérer l'état da la routine de redémarrage des campagnes Facebook, l'URL suivante sera utilisée : https://myprofits.osires.fr/api/Campaign/GetStatut.

Liste et descriptif des API

Campagnes Facebook

- **Campaign** : Ajout/ Suppression/ Mise à jour des campagnes Facebook. Gère les mises en pause, arrêt / relance des campagnes. API aussi utilisée pour la création/ synchronisation des Pixels
- **Platform** : Récupération des Businesses et AdAccount Facebook, utilisés notamment dans l'écran Campaigns, contient la vérification du token Fb.

Shop, Articles & Products

- **Shop** : Ajout/ Suppression/ Mise à jour des products en BDD, l'api va se connecter à l'API du collecteur pour mettre à jour, ajouter ou supprimer des shops qui sont en cours d'utilisation
- **Article** : Ajout/ Suppression/ Mise à jour des articles en BDD
- **ArticleShop** : : Ajout/ Suppression/ Mise à jour des articlesShop en BDD
- **SupplierArticle** : Ajout/ Suppression/ Mise à jour des SupplierArticle
- **SupplierProduct** : Ajout/ Suppression/ Mise à jour desSupplierProduct
- **Order** : Ajout/ Suppression/ Mise à jour des commandes (orders) en BDD
- **OrderLine** : Ajout/ Suppression/ Mise à jour des OrderLine en BDD
- **Product** : Ajout/ Suppression/ Mise à jour des products en BDD

Données et Analytique

- **Analytics** : Récupération des données de l'Analytics stockées en BDD et calculées avec les dépenses des campagnes
- **DataConsolidate :** Données consolidées utilisées pour le dashboard

Utilisateurs

- **User** : Ajout/ Suppression/ Mise à jour des Users
- **Supplier** : Ajout/ Suppression/ Mise à jour des Suppliers
- **Role** : Ajout/ Suppression/ Mise à jour des rôles des utilisateurs

Principales bibliothèques .Net utilisées

- **SwashBuckle** : Adaptation pour .Net Core de swagger, permet, entre autre, de documenter les APIs de l'application
- **Pomelo** : Connecteur MySQL pour .Net Core

MyProfits.Collectors

La collecte des données est gérée par un « collecteur ». Il y a plusieurs collecteurs dans ce projet mais ils fonctionnent globalement de la même manière :

- Un manager est lancé au démarrage de l'application (ShopManager, PlatformManager,…), ce manager va lancer des tâches parallèles (threads) pour chacun des shops ou campagnes.
- Chaque tâche parallèle fonctionne autour d'un minuteur (timer) qui va déclencher périodiquement la récupération des données sur une API cible (celle de Shopify ou Facebook, ou autre dans le futur).
- Les données récupérées sont traitées et écrite en base de données.
- Des web services spécifiques permettent de piloter les collecteurs (arrêt, démarrage, …).

Liste et descriptif des collecteurs

Analytics

Fournit les données nécessaires à l'écran Analytics de MyProfits. Son déclenchement quotidien permet de récupérer les dépenses de campagne de Facebook sur la journée ainsi que les recettes des shops Shopify.

Le collecteur Analytics peut être contrôlé via une API et il est possible de récupérer des données sur Facebook et Shopify sur une période spécifiée (nb : tant que le traitement est en cours, le déclenchement quotidien est bloqué).

DataConsolidation

L'objectif est de calculer au fur et à mesure les indicateurs du tableau de bord à partir des données des campagnes Facebook et Shopify préalablement stockées en base de données. Cette partie exploite donc uniquement les données en BDD et fait les calculs (profits, CA, …) à partir de ces derniers.

CollectorShop

Récupère à intervalle régulier les commandes passées sur Shopify et les stocke en BDD le cas échéant. Lors d'un premier lancement ou si l'application s'arrête le Collector Shop possède un mécanisme de rattrapage des commandes (une date de rattrapage max a été paramétrée).

UpdateOrders

Met à jour les commandes en BDD (en particulier les statuts) à partir de l'API Shopify.

Bibliothèques .Net

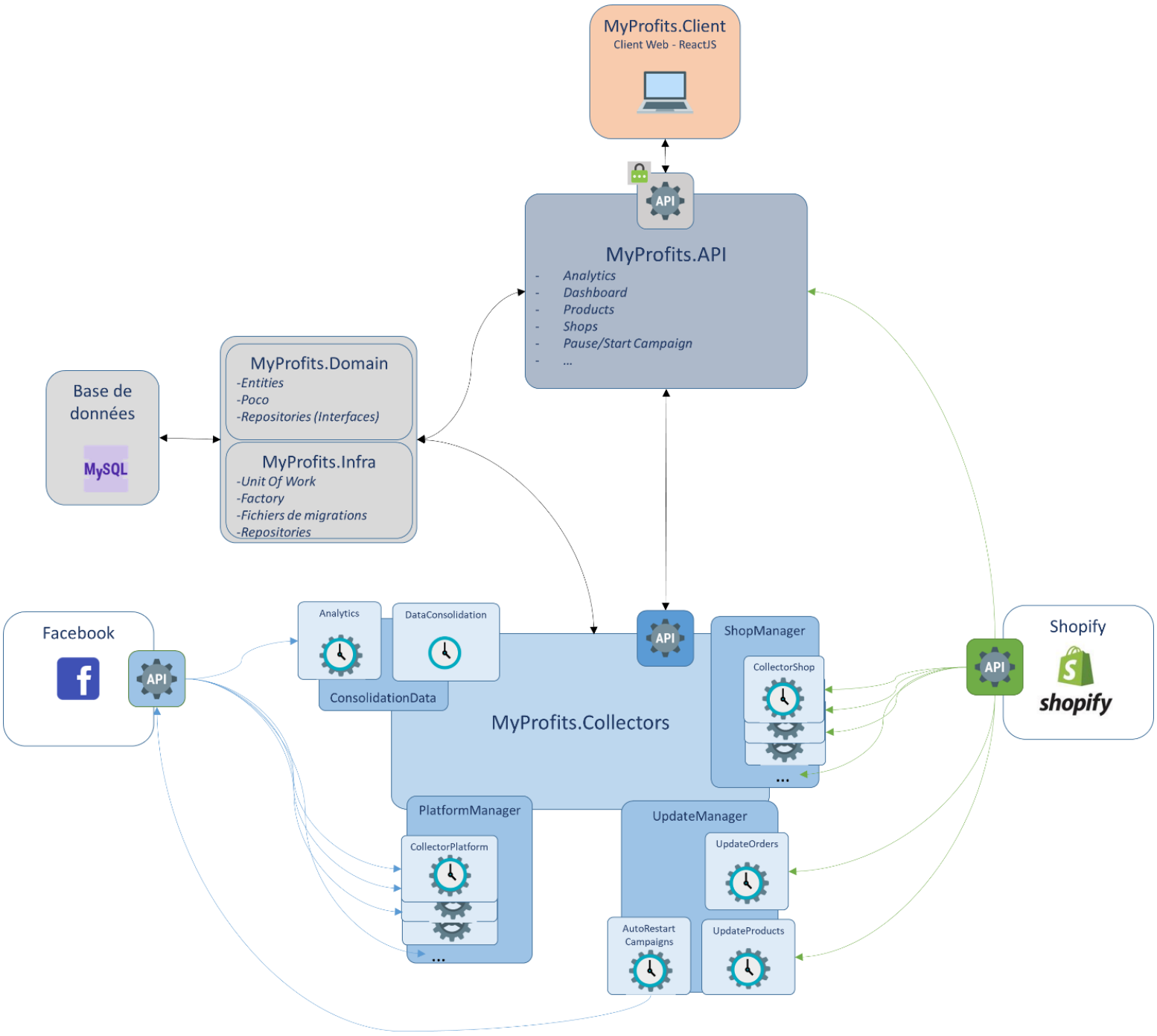
- **Pomelo** : Connecteur MySQL pour .Net Core
- **Log4Net** : Outil de log pour .Net Core
- **ShopifySharp** : Librairie facilitant le management des applications Shopify

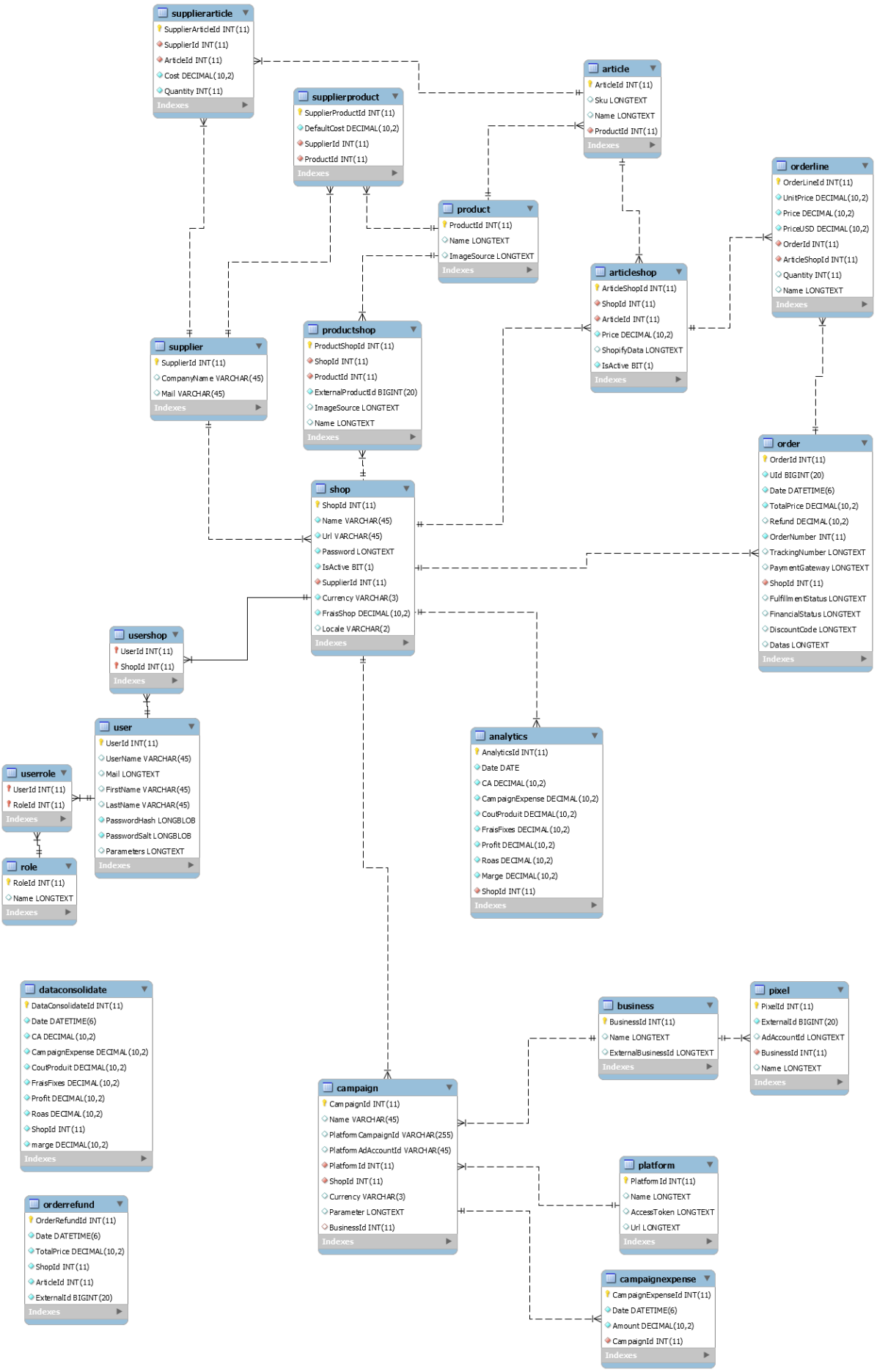
MyProfits.Domain et MyProfits.Infra

Le domaine et l'Infra permettent de désolidariser la base de l'applicatif. Les Entitées de la base sont dans le Domaine, leurs méthodes génériques ou spécifiques, le contexte de la base de données et les Unit Of Works sont présents dans l'Infra.

Les modifications apportées à l'Infra seront automatiquement répercutées sur la Base (utilisation du code first).

Architecture technique de la solution




supplierarticle
SupplierArticleId INT(11)
SupplierId INT(11)
ArticleId INT(11)
Cost DECIMAL(10,2)
Quantity INT(11)
Indexes
supplierproduct
SupplierProductId INT(11)
DefaultCost DECIMAL(10,2)
SupplierId INT(11)
ProductId INT(11)
Indexes
article
ArticleId INT(11)
Sku LONGTEXT
Name LONGTEXT
ProductId INT(11)
Indexes
orderline
OrderLineId INT(11)
UnitPrice DECIMAL(10,2)
Price DECIMAL(10,2)
PriceUSD DECIMAL(10,2)
OrderId INT(11)
ArticleShopId INT(11)
Quantity INT(11)
Name LONGTEXT
Indexes
product
ProductId INT(11)
Name LONGTEXT
ImageSource LONGTEXT
Indexes
articleshop
ArticleShopId INT(11)
ShopId INT(11)
ArticleId INT(11)
Price DECIMAL(10,2)
ShopifyData LONGTEXT
IsActive BIT(1)
Indexes
supplier
SupplierId INT(11)
CompanyName VARCHAR(45)
Mail VARCHAR(45)
Indexes
productshop
ProductShopId INT(11)
ShopId INT(11)
ProductId INT(11)
ExternalProductId BIGINT(20)
ImageSource LONGTEXT
Name LONGTEXT
Indexes
order
OrderId INT(11)
Uid BIGINT(20)
Date DATETIME(6)
TotalPrice DECIMAL(10,2)
Refund DECIMAL(10,2)
OrderNumber INT(11)
TrackingNumber LONGTEXT
PaymentGateway LONGTEXT
ShopId INT(11)
FulfillmentStatus LONGTEXT
FinancialStatus LONGTEXT
DiscountCode LONGTEXT
Datas LONGTEXT
Indexes
shop
ShopId INT(11)
Name VARCHAR(45)
Url VARCHAR(45)
Password LONGTEXT
IsActive BIT(1)
SupplierId INT(11)
Currency VARCHAR(3)
FraisShop DECIMAL(10,2)
Locale VARCHAR(2)
Indexes
usershop
UserId INT(11)
ShopId INT(11)
Indexes
user
UserId INT(11)
UserName VARCHAR(45)
Mail LONGTEXT
FirstName VARCHAR(45)
LastName VARCHAR(45)
PasswordHash LONGBLOB
PasswordSalt LONGBLOB
Parameters LONGTEXT
Indexes
userrole
UserId INT(11)
RoleId INT(11)
Indexes
role
RoleId INT(11)
Name LONGTEXT
Indexes
analytics
AnalyticsId INT(11)
Date DATE
CA DECIMAL(10,2)
CampaignExpense DECIMAL(10,2)
CoutProduit DECIMAL(10,2)
FraisFixes DECIMAL(10,2)
Profit DECIMAL(10,2)
Roas DECIMAL(10,2)
Marge DECIMAL(10,2)
ShopId INT(11)
Indexes
dataconsolidate
DataConsolidateId INT(11)
Date DATETIME(6)
CA DECIMAL(10,2)
CampaignExpense DECIMAL(10,2)
CoutProduit DECIMAL(10,2)
FraisFixes DECIMAL(10,2)
Profit DECIMAL(10,2)
Roas DECIMAL(10,2)
ShopId INT(11)
marge DECIMAL(10,2)
Indexes
business
BusinessId INT(11)
Name LONGTEXT
ExternalBusinessId LONGTEXT
Indexes
pixel
PixelId INT(11)
ExternalId BIGINT(20)
AdAccountId LONGTEXT
BusinessId INT(11)
Name LONGTEXT
Indexes
campaign
CampaignId INT(11)
Name VARCHAR(45)
Platform CampaignId VARCHAR(255)
Platform AdAccountId VARCHAR(45)
Platform Id INT(11)
ShopId INT(11)
Currency VARCHAR(3)
Parameter LONGTEXT
BusinessId INT(11)
Indexes
platform
Platform Id INT(11)
Name LONGTEXT
AccessToken LONGTEXT
Url LONGTEXT
Indexes
orderrefund
OrderRefundId INT(11)
Date DATETIME(6)
TotalPrice DECIMAL(10,2)
ShopId INT(11)
ArticleId INT(11)
ExternalId BIGINT(20)
Indexes
campaignexpense
CampaignExpenseId INT(11)
Date DATETIME(6)
Amount DECIMAL(10,2)
CampaignId INT(11)
Indexes