

Nantes Université - IUT - BUT Informatique

**Rapport de SAE**

Qualité de développement

eq\_02\_01

*OULMAS Thomas, ROUX Célestin, MOREAU–THOMAS Nils, VAN LAERE Pierric*

2ème année de BUT Informatique - SAE du 1er semestre

# 

# 

# Présentation

## Le sujet

une société souhaite une refonte de son système d’information (SI) et l’automatisation

d’une partie du SI, avec principalement la création d’un site (ou application) de présentation et de

ventes de produits en ligne. Le site devra être hébergé sur des serveurs. La société sera

responsable des serveurs et du déploiement de l’application sur ces derniers. L’ensemble devra

répondre aux contraintes de sécurité, de performance et aux aspects juridiques sur la protection

des données.

## Numéro de l’équipe

eq\_02\_01

## Membres de l’équipes

OULMAS Thomas : Recherche des design patterns

ROUX Célestin : Implémentation des design patterns

VAN LAERE Pierric : Implémentation des design patterns

MOREAU–THOMAS Nils : Recherche des design patterns

## Le client

Le client de notre site se nomme Antoine, il possède une graveuse laser avec laquelle il grave des pièces sur mesures en bois. Mais il vend aussi des pièces uniques qu’il produit en parallèle des demandes de ses clients.

Ces pièces sont des dessous de verres, des boules de noël, des portes-clés et autres.

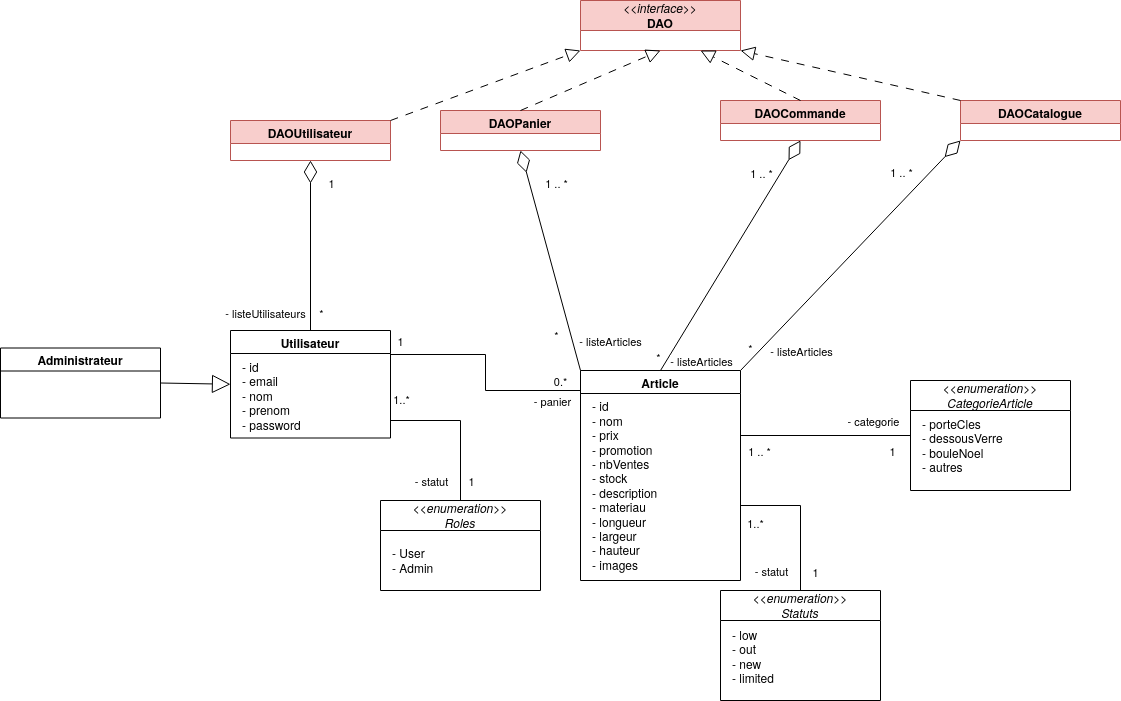
Il possède :

* un instagram : @antoinefactory
* une adresse mail : antoinefactory@gmail.com

# Diagramme de classes niveau analyse

Le diagramme ci-dessous représente les entités composant notre système.

Il a été réalisé lors de notre phase d’analyse.



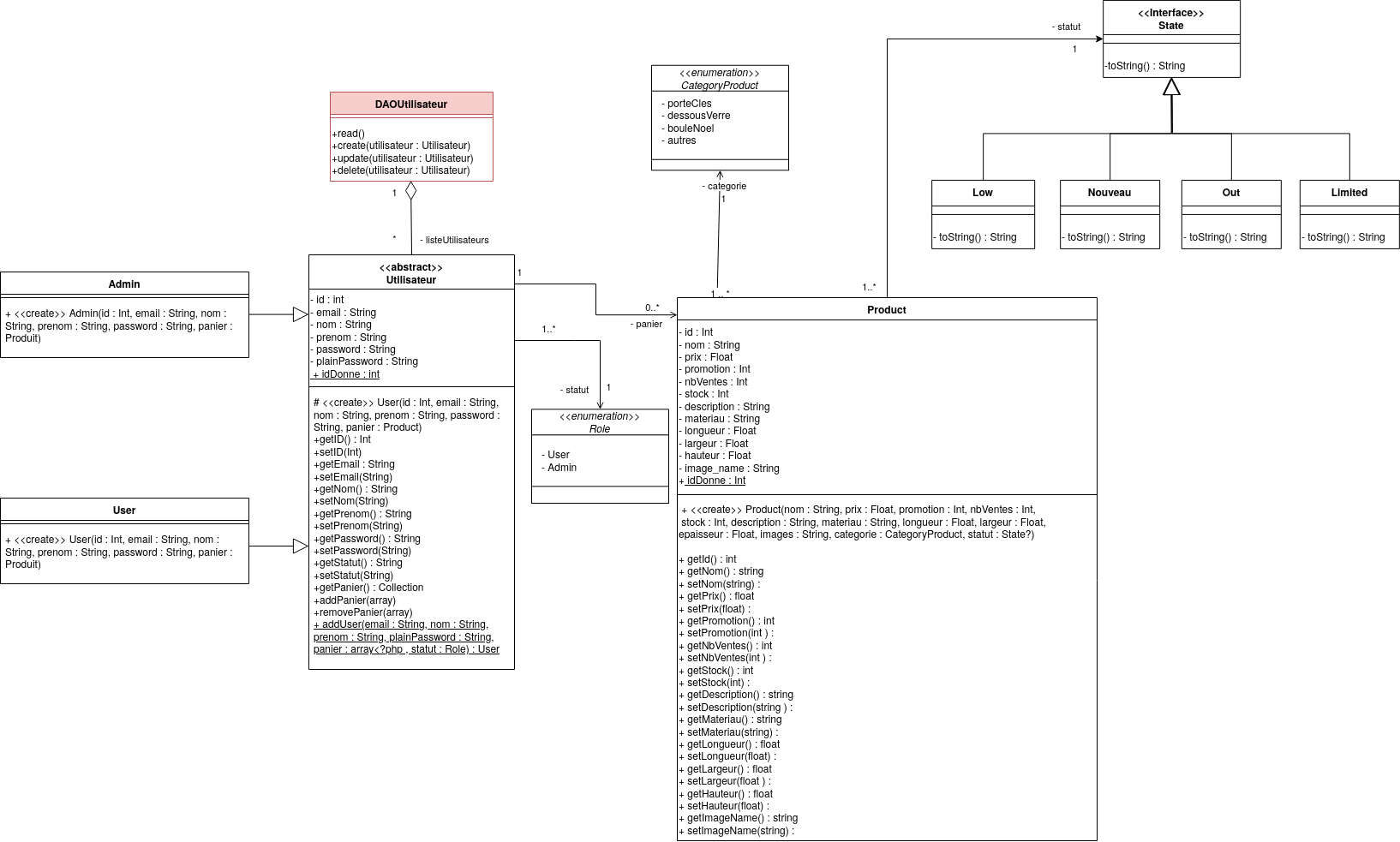
*Accessible ici :*

[*https://drive.google.com/drive/folders/1gYN1hFHo6SKa-QVgsPzaAWxHS1y6ZvUn?usp=sharing*](https://drive.google.com/drive/folders/1gYN1hFHo6SKa-QVgsPzaAWxHS1y6ZvUn?usp=sharing)

# Diagramme de classes niveau conception détaillée

La conception des design patterns sur notre site étant rendue plus compliqué à cause de Symfony qui en implémente une bonne partie par lui-même, nous avons décidé de faire un fonctionnement à échelle réduite du site, nous permettant d’économiser du temps ainsi que d'être plus confiant sur leurs implémentations.

Le diagramme ci-dessous représente les entités composant notre système à échelle réduite, avec la visibilité, les méthodes.



*Accessible ici :*

[*https://drive.google.com/drive/folders/1atqD63N45Wprfzj661242qnFIUnAuL3E?usp=sharing*](https://drive.google.com/drive/folders/1atqD63N45Wprfzj661242qnFIUnAuL3E?usp=sharing)

# Patrons de conceptions

Les implémentations des patrons de conceptions sont disponible dans le dossier /ressources/designsPatterns du git

## DAO

Dans notre implémentation, le DAO permet la liaison avec la table utilisateur de notre base de données, permettant de gérer l’ajout et les modifications de comptes.

## Factory

Notre site avait besoin de créer des utilisateurs, mais notre client voulait des comptes administrateur et utilisateur. Mais comment savoir lequel créer selon le contexte ?

Dans notre implémentation, la factory nous permet de générer des types d’utilisateurs : User et Admin. Le choix étant déterminé par le rôle donné en paramètre de la factory, si le rôle correspond à celui qu’un admin devrait avoir, alors un Admin est créé, si le rôle correspond à celui d’un Utilisateur classique (User) alors un User est créé.

## State

Notre site avait besoin de statut d’article, mais notre client voulait un texte différent selon le statut de l’article. Mais comment faire cette association entre un texte et statut ?

Dans notre implémentation, le State permet d’associer un statut pour un article avec un string, par exemple la classe implémentant State, Low, est associé au string “Bientôt Épuisé” via la méthode toString().

Voici les statuts “Out”, “Low”, “New”, “Limited” avec leurs strings associés : “épuisé” (mis en majuscule avec du CSS), “Bientôt épuisé”, “Nouveau”, “Limited”.



L’intérêt de cette implémentation c’est qu’elle permet d’ajouter, sur l’affichage des articles, du texte selon leurs statuts (s'ils en possèdent un), ce design pattern nous permettant de faire cela en écrivant seulement product.statut.toString(), on pourrait également ajouter de nouveaux statuts dans le futur plus facilement en ajoutant une nouvelle classe implémentant State.

Nous n’avons pas implémenter ce design pattern pour la catégorie car selon les choix du client dans le futur elles peuvent être amenées à être complètement modifiées.

Nous n’avons également pas implémenter ce design pattern pour les rôles puisque Symfony les gère automatiquement donc il n’aurait pas d’intérêt à notre site.