CathyMini

Projet d'intégration ECOM

Document De Conception

Auteur	Date	Description
RASSAT, UZEL	28/11/13	Version Initiale
UZEL	29/11/13	Application d'un modèle
RASSAT	09/12/13	Ajout des fonctionnalités de la façade
BERNARD	13/12/13	Validation du document

/// TODO ///

- modifier les noms des champs de 'subscription'
- verifier les methodes des facades serveur

Table des matières

1 Introduction

1.1 Objet Du Document

Ce document décrit l'architecture technique de CathyMini.

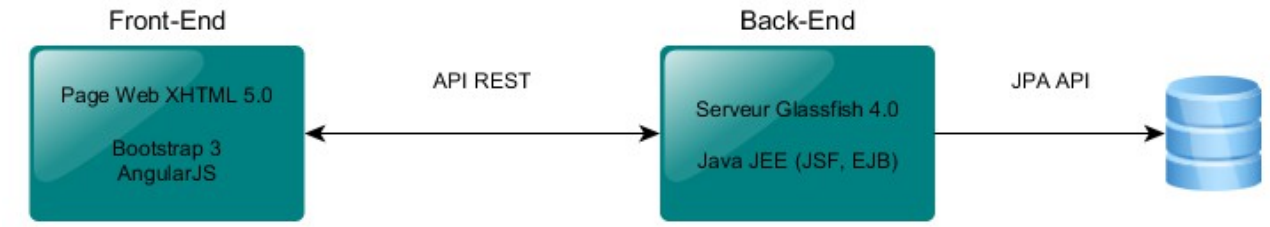
1.2 Découpage technique

CathyMini est un site de E-commerce vendant des produits pour les menstruations, en achat direct ou sous forme d'abonnement.

La partie backend a été développée en Java et est accessible via une API REST.

CathyMini étant un site, la partie frontend est donc visible via un navigateur Web. Elle a été développée avec les technologies standard du Web (XHTML, CSS, JavaScript).

L'application est déployée sur un serveur Glassfish 4.0.



2 Le Front-end

Notre vision du site est celle d'un site *« moderne »*. La conception s'est donc orientée vers un design respectant les chartes graphiques actuelles et des interactions Homme-machine fluides et dynamiques. Nous avons choisi de réaliser la communication entre la partie cliente et la partie serveur grâce à des requêtes Ajax traitées par une API REST. Ces technologies nous paraissaient les plus appropriées car elles permettent d'éviter les rechargements de page à chaque action. Ces chargements se seraient montrés gênants et auraient impactés la fluidité et donc la qualité de la navigation dans notre site.

2.1 L'implémentation des pages Web

Cette exigence de site « *moderne* » demande de développer une application cliente plus importante qu'avec des technologies classiques. La volonté de qualité du logiciel, nous a donc amené à choisir JSF 2 pour implémenter les pages Web. Contrairement aux servlets, les JSF permettent de séparer efficacement la présentation et le contrôleur. Les JSF permettent également de découper les pages en composants et de gérer des templates. Dans ce domaine, notamment avec l'utilisation de Facelets, JSF2 est bien plus riche que les JSP. Une autre solution envisagée a été l'utilisation de GWT, celle-ci a vite été écartée étant donnée que JSF répondait déjà parfaitement à nos besoin et que nos compétences avec JSF étaient plus importantes qu'avec GWT. La partie front-end est ainsi divisée en composants et templates, permettant une meilleure lisibilité, maintenabilité et ré-utilisabilité de ceux-ci.

2.2 L'application cliente

Comme dit précédemment, la communication entre le client et le serveur est essentiellement basée sur des requêtes Ajax. Pour cela, une option possible est d'utiliser les composants de JSF permettant de faire des requêtes Ajax mais qui, dans notre cas, étaient trop rigides et pas assez riches pour couvrir tous nos besoins. Et cela même en utilisant des librairies comme PrimeFaces qui peuvent améliorer les JSF sur ce point. Notre site avait besoin que du code JavaScript spécifique soit écrit et l'utilisation combinée de ce code avec celui auto-généré par les composants aurait nuit de façon importante à la qualité logicielle.

Dans le respect de la qualité logicielle, le choix s'est donc porté vers un framework JavaScript. AngularJS a été choisi, étant celui sur lequel l'équipe est la plus compétente. Ce framework JavaScript permet de bien séparer la partie présentation qui est la structure de la page, de la partie applicative qui va gérer le comportement de la page, ce que ne permet pas la technologie JSF seule. AngularJS étant basé sur un modèle MVC, cette logique de séparation est totalement respectée, améliorant la qualité logicielle. De plus, on se retrouve à un niveau plus bas que celui des JSF ; le mélange entre le code JavaScript spécifique et celui des JSF n'a donc plus lieu. Pour autant, la facilité de développement n'est pas perdu grâce à la richesse des frameworks et notamment d'AngularJS. Un autre point important dans le choix d'AngularJS est qu'il permet de modulariser la partie cliente et de favoriser les tests (unitaires et end-2-end) qui sont malheureusement généralement

oubliés côté client.

2.3 Design

Avec toujours comme objectif un site « moderne », nous avons choisi d'utiliser le framework Bootstrap, combiné à Less pour styliser nos pages. Bootstrap permet de styliser rapidement les pages tout en permettant au site d'être « responsive », ce qui est une nécessité à l'heure actuelle. Less a été choisi dans une but d'avoir du code CSS plus maintenable et lisible, ce qui est généralement un gros point faible de CSS.

2.4 Implantation

Le front-end étant la partie directement accessible par l'utilisateur, il est contenu dans le répertoire « webapp ». Il contient toutes les pages WEB, les composants JSF, les scripts JavaScript, les feuille de style CSS et les images utilisées dans le site.

Les composants JSF sont stockés, selon leur nature, dans les répertoires « *partials* » et « *templates* », partials pour les composants et templates pour les structures. Les pages WEB se trouvent dans le répertoire principal et le répertoire « admin ». Le repertoire « *assets* » contient les les ressources publics (images, scripts, feuilles de style).

La partie JavaScript étant lourde et complexe, un découpage est effectué en respectant le découpage privilégié pour AngularJS. Le découpage est fait en différents modules, directives, contrôleurs, modèles et services :

- Les modules regroupent différents composants, qui peuvent alors être réutilisés dans d'autres modules. Découpant l'application en différents blocs de fonctionnalités.
- Les directives regroupent des composants réutilisables dans l'application, tel que des balises HTML, des attributs ou des classes embarquant du code JavaScript avec eux.
- Les services regroupent le code « métier » de l'application cliente, indépendante de l'interface.
- Les modèles décrivent les différents objets utilisés par l'application.
- Les contrôleurs sont utilisés pour gérer l'affichage et les interactions avec l'affichage. Ils s'appuient sur les services et appartiennent à un module correspondant généralement à une page Web.

3 Le Back-End

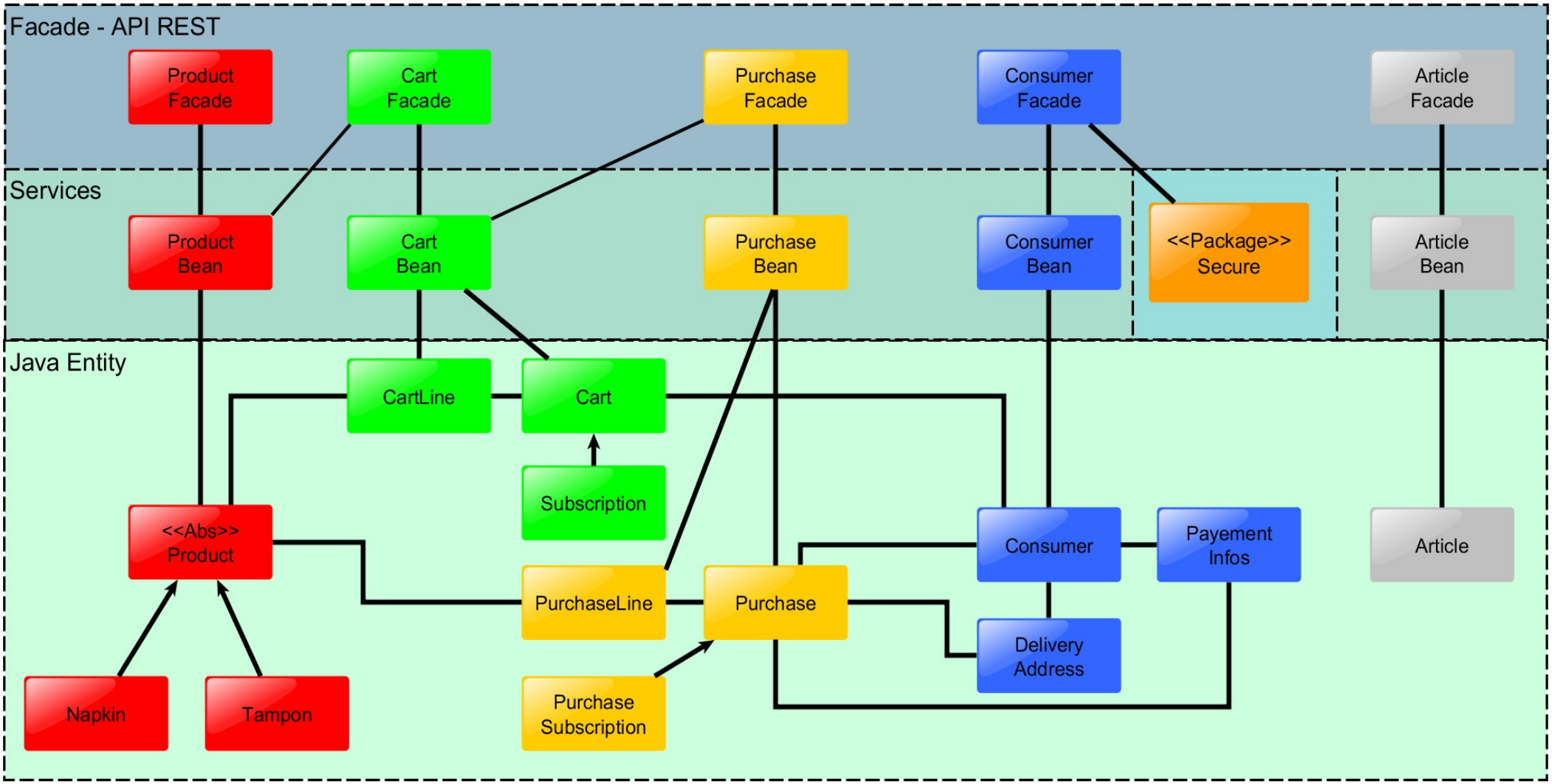
3.1 Architecture logicielle

L'architecture de la partie Back-end est découpée en 3-tiers :

- Le premier tiers est l'interface du serveur ; les différentes façades contactées par le Front-end via l'API REST. Elles reçoivent, traduisent et propagent les requêtes à la logique métier.
- Le deuxième tiers implémente la logique métier grâce à des Java Session Bean. Il traite les requêtes reçues sur la façade et réalise la persistance des données.
- Le dernier tiers implémente des Java Entity Bean. Ils permettent de réaliser le mapping des données à l'aide d'annotations et d'être utilisé comme modèle par la logique métier.

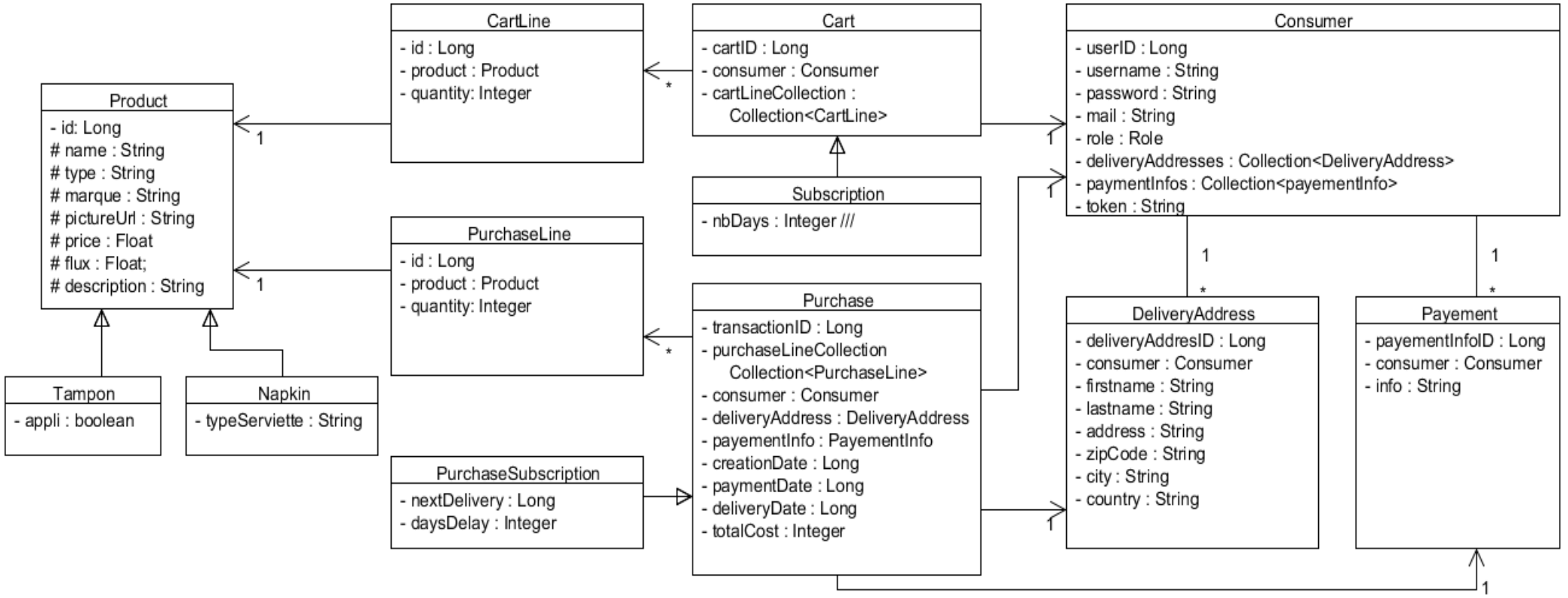
Un composant supplémentaire - le composant *Secure* - gère la sécurité de l'application. Lorsqu'un utilisateur essaie d'accéder à une ressource, un intercepteur vérifie que ses droits sont suffisants pour y accéder. Lorsqu'un utilisateur se connecte, il génère aléatoirement et insère dans une table une chaîne de caractère qui servira d'identifiant pour la connexion de l'utilisateur.

Nous avons choisi de n'utiliser que des Java Session Bean stateless car les technologies que nous utilisions ne nous permettaient pas de faire fonctionner les Java Session Bean stateful simplement. Les données de session de l'utilisateur lui sont transmises par des modifications explicites de sa session HTTP. Les données de session sont les caddies (panier et abonnements), gérés par la façade *CartFacade,* et la connexion au compte de l'utilisateur grâce à la façade *ConsumerFacade* et le package *Secure*.


Facade - API REST
Product Facade
Cart Facade
Purchase Facade
Consumer Facade
Article Facade
Services
Product Bean
Cart Bean
Purchase Bean
Consumer Bean
<<Package>> Secure
Article Bean
Java Entity
CartLine
Cart
Subscription
<<Abs>> Product
PurchaseLine
Purchase
Consumer
Payement Infos
Article
Delivery Address
Napkin
Tampon
Purchase Subscription

3.2 Les Java Entity

Les Java Entity Bean forment une représentation de la base de données.



3.3 Détail des façades et de la logique métier

Les fonctionnalités fournies par le serveur sont uniquement visibles et accessibles au niveau des façades qui s'appuient ensuite sur les services pour réaliser la logique métier. Les façades contiennent donc des attributs @EJB leur permettant de faire appel aux méthodes que les services fournissent. Pour rappel, les services sont implantés à l'aide de Java Session Bean stateless.

3.3.1 ProductFacade

La façade ProductFacade traite les requêtes centrées sur les produits.

Méthode	Droit d'accès	Description
create(...)	Administrateur	Crée un nouveau produit dans la base de donnée
edit(...)	Administrateur	Modifie les informations associées à un produit
delete(...)	Administrateur	Détruit un produit de la base de donnée et les informations qui lui sont associées
all(...)	Libre	Renvoie la liste triée de tous les produits satisfaisant un ensemble de critères de recherche et de tri donnée en paramètre
populate(...)	Administrateur	Peuple la base de données à partir d'un fichier XML

3.3.2 PurchaseFacade

La façade *PurchaseFacade* traite les requêtes centrées les achats des utilisateurs.

Méthode	Droit d'accès	Description
createPurchase(...)	Membre	Finalise le nouvel achat d'un utilisateur
createSubscrption(...)	Membre	Finalise le nouvel abonnement d'un utilisateur
editSubscrption(...)	Membre	Modifie les informations liées à l'abonnement d'un utilisateur
stopSubscrption(...)	Membre	Arrête l'abonnement en cours d'un utilisateur
getSubscriptions(...)	Membre	Renvoie la liste des abonnements de l'utilisateur connecté
getPurchases(...)	Membre	Renvoie la liste des achats de l'utilisateur connecté
purgePurchases(...)	Administrateur	Supprime tous les achats de plus de 2 ans et les abonnements stoppés depuis plus de 2 ans de la base de données

3.3.3 ConsumerFacade

La façade ConsumerFacade traite les requêtes centrées sur les utilisateurs.

Méthode	Droit d'accès	Description
suscribe(...)	Anonyme	Crée un nouveau compte utilisateur
connect(...)	Anonyme	Connecte un utilisateur à son compte
logout(...)	Membre	Déconnecte un utilisateur de son compte
delete(...)	Membre	Supprime un compte utilisateur
seeCurrent(...)	Libre	Renvoie les informations générales du compte utilisateur auquel l'utilisateur est connecté. Le mot de passe étant une information critique, il n'est pas renvoyé
UpdateConsumer(...)	Membre	Modifie les informations générales d'un compte utilisateur
addAddress(...)	Membre	Ajoute une adresse de livraison au compte utilisateur connecté
deleteAddress(...)	Membre	Supprime une adresse de livraison au compte utilisateur connecté
getAddresses(...)	Membre	Renvoie la liste de toutes les adresses de livraison du compte utilisateur connecté
addPaymentInfos(...)	Membre	Ajoute une information de payement au compte utilisateur connecté
deletePaymentInfos (...)	Membre	Supprime une information de payement au compte utilisateur connecté
getPaymentInfos(...)	Membre	Renvoie la liste de toutes les informations de payement du compte utilisateur connecté. Ces informations étant critiques, elles ne seront renvoyées que de manière partielle
getConsumers(...)	Administrateur	Renvoie la liste de tous les utilisateurs. Les informations critiques tel que les informations de payement et le mot de passe ne sont pas renvoyées

3.3.4 CartFacade

La façade *CartFacade* traite les requêtes centrées le caddie.

Méthode	Droit d'accès	Description
addProductToCart(...)	Libre	Ajoute un produit au caddie et, si l'utilisateur est connecté, conserve les informations du caddie dans la base de données
deleteProductToCart (...)	Libre	Supprime un produit du caddie et, si l'utilisateur est connecté, conserve les informations du caddie dans la base de données
ChangeQuantityTo Cart(...)	Libre	Change la quantité d'un produit du caddie et, si l'utilisateur est connecté, conserve les informations du caddie dans la base de données
getCart(...)	Libre	Renvoie le caddie de l'utilisateur
AddProductTo Subscription(...)	Libre	Ajoute un produit au caddie '*abonnement*' et, si l'utilisateur est connecté, conserve les informations du caddie dans la base de données
deleteProductTo Subscription(...)	Libre	Supprime un produit du caddie '*abonnement*' et, si l'utilisateur est connecté, conserve les informations du caddie dans la base de données
ChangeQuantityTo Subscription(...)	Libre	Change la quantité d'un produit du caddie '*abonnement*' et, si l'utilisateur est connecté, conserve les informations du caddie dans la base de données
getSubscription(...)	Libre	Renvoie le caddie '*abonnement*' de l'utilisateur