CathyMini

Projet d'intégration ECOM

Document De Conception

Auteur	Date	Description
Rassat	28/11/13	Version Initiale

Table des matières

1 Introduction

1.1 Objet Du Document

Ce document décrit l'architecture technique de CathyMini.

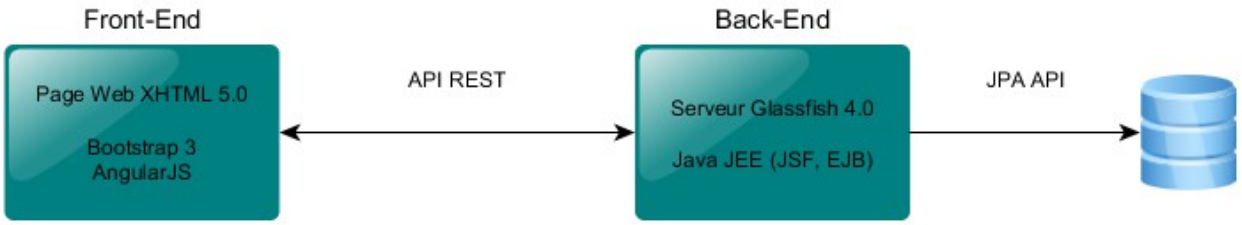
1.2 Découpage technique

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 développé 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

2.1 Notre vision du front end

Dès le début, la vision de l'équipe, a été d'avoir un site « moderne ». Au niveau du design, en respectant les chartes actuelles de graphisme, mais aussi au niveau des interactions Homme-machine, où celles-ci doivent être fluides et dynamiques. Dans cette optique, une communication entre la partie cliente et serveur, basée sur des requêtes Ajax et traitées par une API REST semble la plus adaptée. L'avantage est que les rechargements de page gênant à chaque action sont évités, la fluidité et donc la navigation en sont grandement améliorées.

2.2 L'implémentation des pages Web

En revanche, cette exigence de site « moderne » demande de développer une application cliente plus importante que dans un projet classique. La volonté de qualité du logiciel, nous a donc amené à choisir JSF 2 pour implémenter les pages Web. Les JSF permettent une nette séparation entre la présentation et le contrôleur, contrairement aux servlets seules. Le choix entre JSF 2 et JSP est basé sur la façon de pouvoir découper les pages en composants et gérer des templates. Dans ce domaine JSF 2 avec notamment l'utilisation de Facelets 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 donc divisé en différents composants et templates, permettant une meilleure lisibilité, maintenabilité et réutilisabilité.

2.3 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, le problème est que ceux-ci sont trop rigides est pas assez riches pour couvrir tous nos besoins. L'utilisation de librairies comme PrimeFaces peut améliorer les JSF sur ce point. Dans tous les cas, notre site aura besoin que du code JavaScript spécifique soit écrit, le mélange de ce code avec celui auto-généré par les composants 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. Un 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 JSF seul. Angular étant basé sur MVC, cette logique de séparation est totalement respectée, améliorant la qualité logicielle. Un autre des avantages et qu'on se retrouve un niveau plus bas par rapport aux 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 est notamment d'Angular. Un autre point important dans le choix d'AngularJS est qu'il permet de modulariser la partie cliente, il a également été construit dans le but de favoriser les tests (unitaire et end-2-end), partie de tests qui est malheureusement généralement oublié côté client.

2.4 design

Dans le but de site « moderne » on a choisi d'utiliser Bootstrap 3 avec 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.

3 Le Back-End

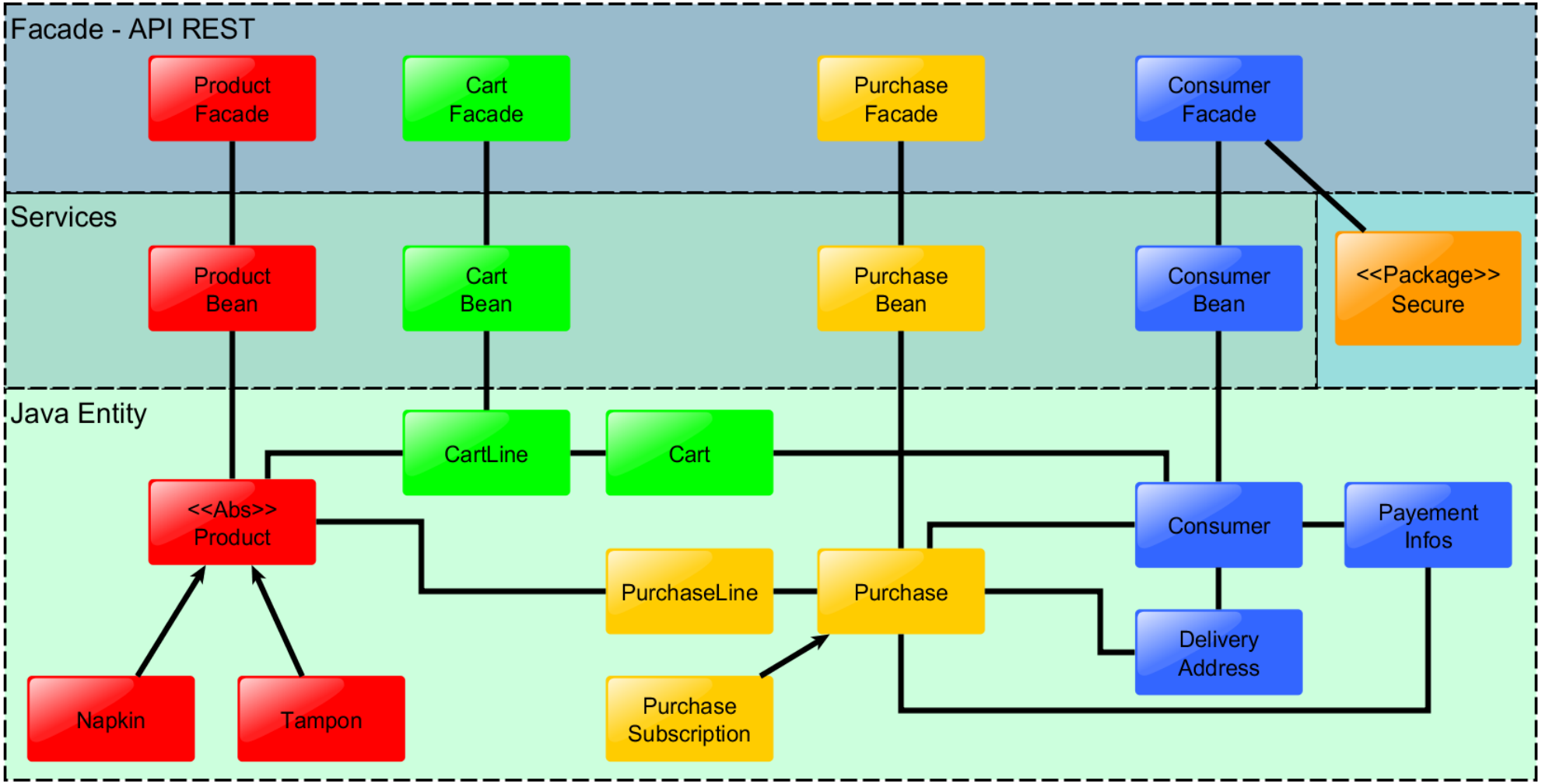
3.1 Architecture logicielle

L'architecture de la partie Back-end est découpé 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 implémentant 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. Ils permettent de réaliser le mapping des données directement dans les Java Entity 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 le caddie, géré 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
Services
Product Bean
Cart Bean
Purchase Bean
Consumer Bean
<<Package>> Secure
Java Entity
CartLine
Cart
<<Abs>> Product
Consumer
Payement Infos
PurchaseLine
Purchase
Delivery Address
Napkin
Tampon
Purchase Subscription

3.2 Les Java Entity