ID CARD

PROJET J2E

CAHIER DES CHARGES
25/01/17
-
03/02/17

NUMAN GEZGIN
RICHARD HADDAD

Document rédigé par Richard Haddad

IUT Paris Descartes
Licence Professionnelle Génie logiciel
2017

CONTEXTE

Projet à rendre pour le 31 Mars 2017.

Sujet libre.

Langage mis en avant : J2E.
Technologies imposées: Servlet, JSP.
Optionnel : sauvegarde des données dans une base de données.

TABLE DES MATIÈRES

PROBLÉMATIQUE

Un internaute habitué à parcourir internet a tendance à s'inscrire à de nombreux sites. Dans la majorité des cas celui-ci rentrera à chaque fois les mêmes informations : pseudo, mot de passe, nom, prénom, etc.

Ces répétitions sont lassantes, tant côté utilisateur que côté développeur, avec des formulaires qui sont parfois identiques d'un site à l'autre. Refaire la roue en somme.

N'est-ce pas un des buts de l'informatique que d'éviter ce type de redondance ?

CONCEPT

L'application vise deux publics : l'utilisateur, et le développeur.

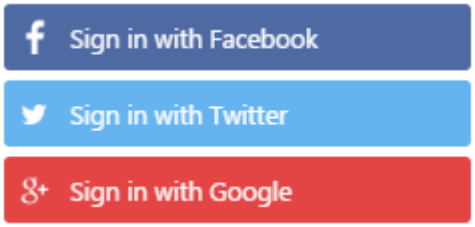
Du point de vue utilisateur, le but de l'application est de lui permettre d'entrer ses informations personnelles et de connexions (pseudo, nom, etc) une seule et unique fois, pour ensuite pouvoir s'inscrire à un maximum de sites web sans avoir à rentrer une seconde fois ces mêmes informations.
Un slogan peu original pourrait être : *write once, sign in everywhere*.

Du point de vue développeur, l'application pourrait grandement lui faciliter la tâche. Le développeur n'aurait qu'à demander à l'utilisateur un identifiant unique (et mot de passe) après quoi il lui suffirait d'effectuer les requêtes des données souhaitées à ladite application.

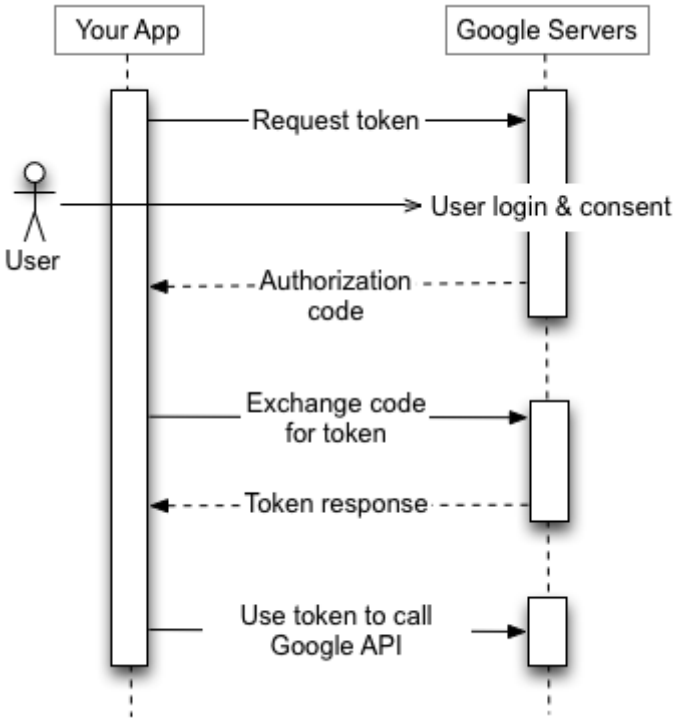
Nous porterons une attention particulière à la sécurité, notamment en permettant à l'utilisateur de définir quels sites peuvent avoir accès à quelles informations, et sous quelles conditions.

L'EXISTANT

Le concept de l'application se base sur des APIs existantes créées par Google, Facebook et Twitter.



Prenons l'exemple de Google.



Lorsque l'utilisateur souhaite s'inscrire au site web du développeur, un token est demandé au serveur Google. Celui-ci requiert alors une connexion manuelle de l'utilisateur, sans passer par le site web !
Une fois la connexion effectuée, ainsi que l'autorisation donnée par l'utilisateur, le serveur Google envoie un code au site web, lui permettant de l'échanger contre un token.
C'est avec ce token que le site web peut alors récupérer les données personnelles de l'utilisateur, à condition qu'il l'ait autorisé.

ID Card utilisera un principe similaire.

FONCTIONNALITÉS

CÔTÉ UTILISATEUR

SITE DE L'APPLICATION

L'utilisateur peut, en accédant directement au site de l'application ID Card :

- s'inscrire
- se connecter (si inscrit)
- gérer sa configuration (si connecté)

INSCRIPTION

L'utilisateur peut se créer un compte via un formulaire d'inscription classique. Il lui sera demandé :

- un pseudonyme (unique, longueur 3 – 32)
- une adresse mail (unique, longueur 6 – 64)
- un mot de passe (longueur 6 – 32)

Une fois l'inscription achevée est confirmée, l'application connecte automatique l'utilisateur.

CONNEXION

Si préalablement inscrit et non connecté, l'utilisateur peut se connecter à l'application via un formulaire d'inscription classique. Il lui sera demandé son pseudonyme (ou mail, au choix) et son mot de passe.

En cas de multiples échecs un captcha Google pourra être requis.

Une fois la connexion achevée et confirmée, l'application redirige l'utilisateur sur la page de configuration.

PROPRIÉTÉS ET CONFIGURATION

Lors de l'inscription, l'utilisateur possède alors 4 propriétés :

- un identifiant (unique, constant, visible)
  Différencie chaque utilisateur de part son caractère unique. Utilisé par les applications externes.
- un pseudonyme (unique, constant, visible)
  Utilisé par la connexion et les applications externes
- une ou plusieurs adresses mail (modifiable)
  Au moins une adresse doit être définie comme principale. Utilisé par la connexion et en cas de perte de mot de passe (via la principale). Peut être utilisé par les applications externes si permit.
- un mot de passe (chiffré, modifiable, non visible)
  Utilisé par la connexion. Inaccessible par les applications externes.

En plus de ces 4 propriétés, l'utilisateur peut en créer autant qu'il le souhaite.

Une propriété est caractérisée par :

- un nom
- aucune, une, ou plusieurs valeurs

Chaque valeur possède une visibilité. La visibilité définie si une application externe peut accéder à la valeur en question.
L'utilisateur doit définir pour chaque valeur quelles applications peuvent y accéder. Par défaut une valeur n'est accessible par aucune application externe. Une application est définie via l'adresse IP de son serveur.

A noter que ce principe de valeur et de modification de la visibilité ne s'applique pas aux 4 propriétés citées plus haut, à l'exception des adresses mail.

L'utilisateur a également accès à toutes les sessions ouvertes depuis des applications externes, avec diverses informations (site web, date de connexion, IP, propriétés demandées). L'utilisateur peut à tout moment fermer une ou plusieurs de ses sessions.

Sans session ouverte, une application externe ne peut accéder à aucune propriété.

Toutes ces actions peuvent être effectuées depuis la page de configuration.

INSCRIPTION À UNE APPLICATION EXTERNE

Lors de l'inscription à une application externe utilisant l'API de l'application, il sera demandé à l'utilisateur, via une fenêtre indépendante, son pseudonyme (ou adresse mail) et son mot de passe afin de se connecter à l'application ID Card.

Une fois la connexion achevée et confirmée, la fenêtre se ferme et l'application est informée de la connexion. L'utilisateur est alors inscrit.

CÔTÉ DÉVELOPPEUR

Du point de vue du développeur ID Card permet de simplifier la gestion de la connexion et de l’inscription, ainsi qu’alléger le stockage de la base de donnée.

L'utilisation de l'application se fait en requêtes Ajax.

INSCRIPTION ET CONNEXION

Lorsque le développeur propose à l'utilisateur de s'inscrire ou se connecter, il doit indiquer à l'API quelles propriétés souhaite t-il pouvoir y accéder en précisant pour chaque propriété voulue :

- le nom de la propriété
- son type (string, entier, double, mail, date, …)
- son importance (requis, facultatif, ...)

Le développeur peut proposer à l'utilisateur de s'inscrire ou se connecter via l'API de l'application en

redirigeant l'utilisateur vers une adresse url spécifique (que ce soit via un lien, bouton, ou action javascript) avec les données en GET demandées (voir au dessus) qui ouvrira alors une fenêtre indépendante proposant à l'utilisateur de se connecter avec ses identifiants ID Card.

Une fois la connexion préliminaire effectuée, ID Card alertera l'utilisateur sur les propriétés accessibles par l'application externes. Ce n'est que lorsque l'utilisateur confirmera explicitement rendre accessible ces propriétés qu'il sera considéré comme réellement connecté. L'application externe obtiendra alors l'identifiant unique de l'utilisateur qui lui servira à récupérer les propriétés.

Si jamais l'utilisateur ne permet pas l'accès à une propriété requise par l'application externe, la connexion est rendue impossible tant que la visibilité de la propriété n'est pas modifiée, ou que l'application externe ne réduit pas ses contrainte. La visibilité des propriétés requises pouvant être modifiée depuis la même fenêtre avant la finalisation de la connexion.

RÉCUPÉRATION DES PROPRIÉTÉS

Une fois la connexion finalisée, l'application externe peut accéder en Ajax à l'ensemble des propriétés répondant aux conditions suivantes :

- La propriété a été demandée lors de la connexion
- La propriété est rendue public par l'utilisateur pour cette application externe

Le développeur doit pour cela fournir l'identifiant du compte et le nom de la propriété voulue. L'API renvoit alors deux informations :

- un booléen définissant si la requête est un succès ou non (booléen)
- un code d'erreur rendant le booléen plus explicite (entier)
- la propriété souhaitée
  composée de :
  - son nom (string)
  - une liste de valeurs (type des valeurs variable, liste pouvant être vide)

TECHNOLOGIES

L'application est composée d'une partie serveur (avec base de donnée) et une partie client.

A noter que la priorité sera mise sur l'aspect serveur (notamment J2E) de part la nature du projet.

Côté serveur :

- J2E (Java 8)
  Requit de part la nature même du projet.
- jOOQ (https://www.jooq.org/)
  Ne voulant pas manipuler du SQL directement dans le code Java, nous avons pensé en premier lieu aux ORM. Ces derniers ayant des défauts pouvant être rédhibitoires à terme (optimisation, débogage) nous avons souhaité partir sur une solution plus simple à utiliser, légère, performante, et proche de la syntaxe SQL tout en profitant de la sécurité de la POO.
- PostgreSQL
  Mise en place rapide, performance et fiabilité connues.

Côté client :

- HTML 5, CSS 3, JavaScript ES6 (Ajax via jQuery)
  Technos web de base.
- Bootstrap
  Pour une mise en place rapide de la vue.
- ReactJS, TypeScript
  La partie client étant « one-page », React facilitant et sécurisant la conception de la vue. Typescript permettant une syntaxe typée et plus stricte, donc plus fiable.
- SASS
  Facilite la conception du CSS via des instructions.

STRUCTURE DU CODE

SITE WEB DE L'APPLICATION

Lors de l'accès par l'utilisateur au site de l'application, une page HTML lui sera renvoyée. Après quoi les actions utilisateur provoqueront des appels Ajax au serveur. La gestion de la vue étant faite via le JavaScript, avec React notamment.

CÔTÉ SERVEUR

Le serveur utilise les Servlets.

La première requête utilisateur (lors de l'arrivée sur le site) renverra une page HTML.
Hormis ce cas-là, chaque requête utilisateur renverra, via les Servlets correspondant, des données JSON.

La structure du serveur suit le pattern MVC.

La bibliothèque jOOQ est utilisée dans le modèle afin d'effectuer les requêtes à la base de donnée PostgreSQL.

Lors d'une requête utilisateur, le fichier « web.xml » appelle le Servlet correspondant, ce dernier étant un contrôleur. Il appellera alors le modèle pour les requêtes à la base de donnée, et la vue pour le renvoi des données JSON.

CÔTÉ CLIENT

Une fois la page HTML récupérée par le client, le JavaScript s'initialise. Ce dernier utilise le pattern MVC, la vue étant entièrement gérée par React, et le modèle utilisant Ajax.

La première page étant celle de connexion & inscription, toutes les actions de l'utilisateur provoqueront des requêtes Ajax au serveur. Ces dernières renvoyant des données JSON permettant au contrôleur de mettre à jour la vue (donc les composants React).

Les composants React utiliseront les classes Bootstrap, et adapteront leur structure en fonction de ce dernier.