

Compte Rendu

Intégration des compétences

Réalisé par
FERCHICHI Naima
EL ASRI Ayoub

I. Présentation

Avec l'essor des achats en ligne, le passage à une boutique en ligne présente d'innombrables avantages. C'est pour cela qu'on a choisi de créer notre propre site de vente des vêtements en ligne, en utilisant Java Entreprise Edition.

II. Spécifications Fonctionnelles

A. Diagramme de cas d'utilisation

Il s'agit d'un diagramme de cas d'utilisation illustrant les actions qui peuvent être faites par l'utilisateur sur notre site de ventes (voir **figure 1**).

Un utilisateur a la possibilité de :

- S'authentifier
- Parcourir les articles sur la page d'accueil.
- Ajouter une ou plusieurs articles sur son panier.
- Consulter ses commandes.
- Supprimer un article de son panier ou annuler une commande

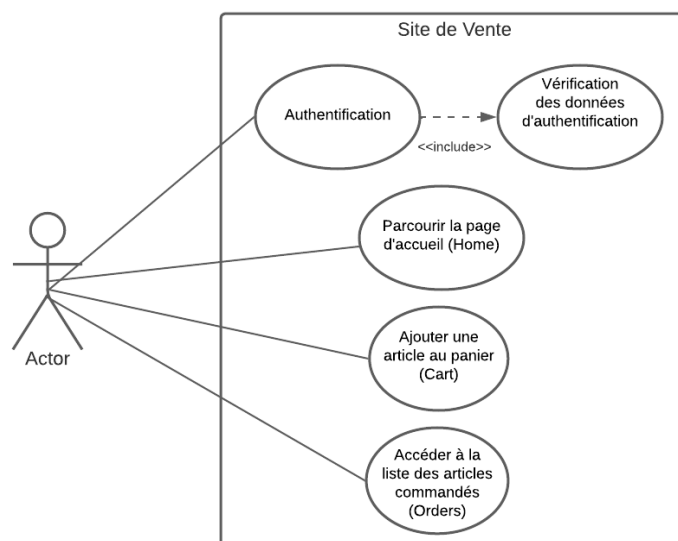


Figure 1 : Diagramme de cas d'utilisation

III. Spécifications Techniques

A. Diagramme de classe

Les diagrammes de classes illustrent trois ensembles de classes :

- ❖ Entities
- ❖ Dao
- ❖ Servlets

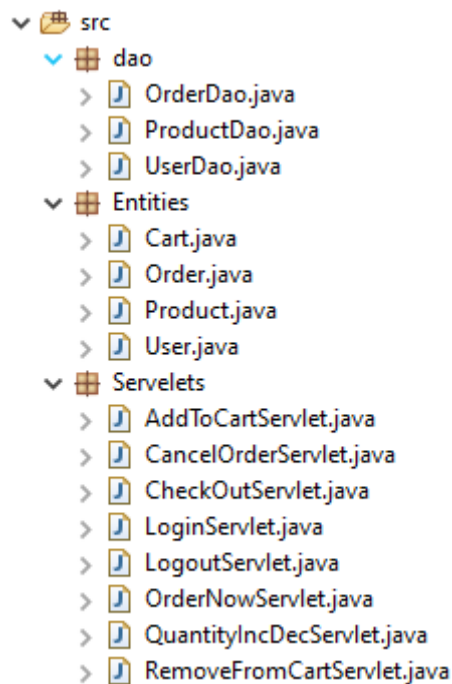


Figure 2 : Ensemble des classes du projet

Le *package* Entities qui contient quatre classes de bases de notre site de vente :

- Utilisateur
- Produit
- Commande
- Panier

Chaque classe est caractérisée par des attributs d'identité, leurs *getters* et *setters*, en plus des annotations pour implémenter le concept de ORM. (voir **figure 3**).

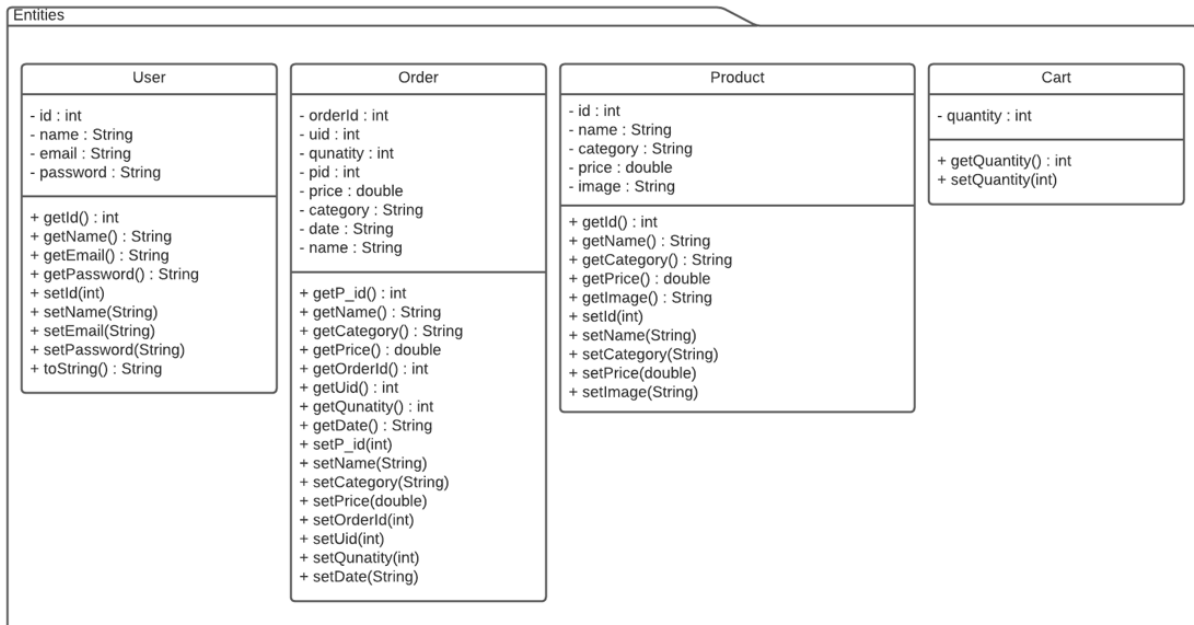


Figure 3 : Diagramme de classes (package Entities)

Le *package* dao qui contient trois classes, permettant de séparer les opérations d'accès de leur mise en œuvre.

- OrderDao
- ProductDao
- UserDao

Chaque classe est caractérisée par un **EntityManager**, permettant de réaliser les opérations CRUD sur une base de données via une unité de persistance, appelée **ecommerce** dans notre cas (voir **figure 4**).

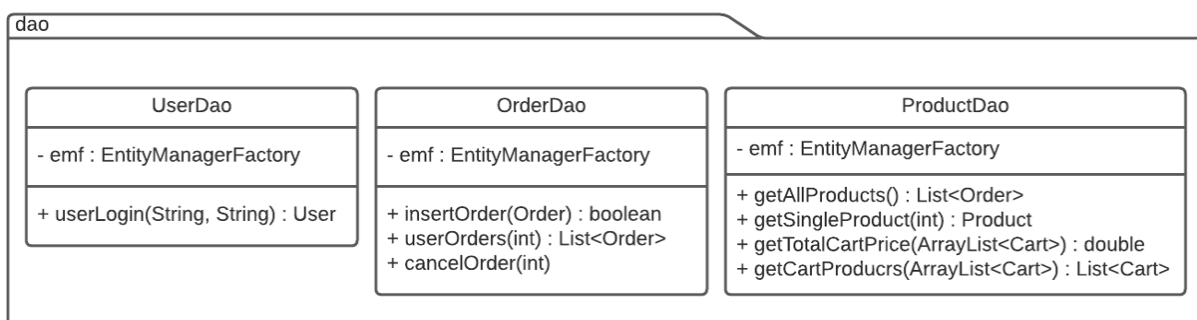


Figure 4 : Diagramme de classe (package dao)

Le *package* servlets qui contient huit servlets, ce sont des classes qui permettent de créer dynamiquement des données au sein d'un serveur HTTP. (voir **figure 5**)

- LoginServlet

- LogoutServlet
- AddToCartServlet
- RemoveFromCartS
- QuantityIncDecServlet
- OrderNowServlet
- CancelOrderServlet
- CheckOutServlet

Chaque servlet, selon les cas, fait une redirection vers une page JSP, pour chaque ressource, on a spécifié une forme unique sous forme d'un fichier jsp :

- ★ Cart.jsp
- ★ Index.jsp
- ★ Login.jsp
- ★ Orders.jsp

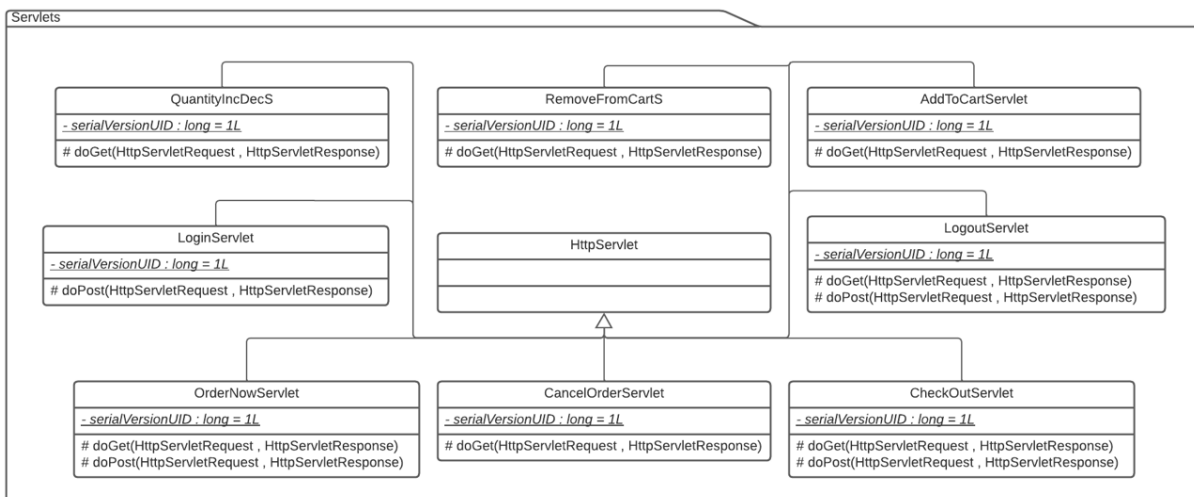


Figure 5 : Diagramme de classe (package servlets)

B. Développement

Notre projet est composé de deux sous-projets, le premier projet “NotreProjetJ2e” est projet qui contient la partie Dao et Rest API en utilisant Hibernate et Jersey

Le deuxième projet “projet2Ecommerce” qui contient la partie Frontend implémenté en jsp en utilisant la rest API déjà implémenté dans le premier projet

Comme première étape, On a commencé par définir les classe du premier projet dans le package “Entities”, on a créé quatres classes

- La classe product: contient toutes les propriétés relatives à un article à savoir le nom,le prix,la catégorie...

- La classe Order: contient toutes les propriétés relatives à une commande inclus les propriétés d'un produit à savoir son pid, le prix, le nom et la catégorie en plus de l'identifiant de la commande, l'identifiant de l'utilisateur et la quantité de la commande
- La classe User: contient toutes les propriétés relatives à un utilisateur à savoir son email, son nom d'utilisateur, son mot de passe...
- La classe Cart: contient la quantité d'un panier.

Toutes les classes contiennent les définitions des getters et setters

Ensuite on est passé à la création de base de donnée MySQL qui a pour nom "ecommerce_cart", ensuite on a créé les table , "orders" qui stocke les commandes la table "User" qui stocke les utilisateurs et la table "cart" qui stock les articles du panier

On a utilisé Hibernate pour la gestion de la partie DAO.Hibernate est un outil de mappage objet-relationnel qui fournit une implémentation de JPA;

Pour la configuration d'Hibernate on a ajouté les fichiers .jar à la librairie du projet ainsi que la création du fichier "Persistence.xml" dans le dossier "\E-Commerce\src\META-INF".Ce fichier XML sert à créer une unité de persistance dans notre exemple c'est

`<persistence-unit name="ecommerce">` , le nom "ecommerce" va être utilisé à chaque création d'une "EntityManagerFactory" dans le DAO.

La définition d'un "Provider" qui fait référence à l'implémentation JPA spécifique utilisée dans notre application pour conserver les objets dans la base de données.

La configuration d'Hibernate est faite pour les trois classe du package "Entities" en utilisant la syntaxe suivante :

```
<class>Entities.Order</class>
<class>Entities.Product</class>
<class>Entities.User</class>
```

En fin on doit préciser les propriétés suivante le

"driver",

"l'URL de la base de donnée",

Le nom de l'utilisateur,

Le mot de passe

```
<properties>
  <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
  <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/ecommerce_cart?serverTimezone=UTC" />
  <property name="javax.persistence.jdbc.user" value="root" />
  <property name="javax.persistence.jdbc.password" value="" />
  <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect" />
</properties>
```

Une fois la configuration réussie on a procédé à la définition de la classe Dao C'est la classe repertoire qui implémente des méthodes et qui fait appel aux données stockés dans la base de données grâce à Hibernate comme suit :

```
@PersistenceContext(unitName="ecommerce")
private EntityManagerFactory emf;

public OrderDao() {
    emf = Persistence.createEntityManagerFactory("ecommerce");
}
```

La création d'une "PersistenceContext" qui est un ensemble d'instances d'entité dans lequel, pour toute identité d'entité persistante, il existe une instance d'entité unique grâce à l'annotation java "@PersistenceContext(unitName=le_nom_de_l'unité_de_persistence"

Ensuite La définition d'un attribut de type EntityManagerFactory dans la classe qui va être initialisé dans le constructeur de la classe .

Pour l'insertion des données on a utilisé la méthode merge d'Hibernate utilisée dans une transaction qui doit obligatoirement finir par un commit pour garantir l'apparition des données dans la base de données comme suit :

```
public boolean insertOrder(Order model)
{
    boolean result = false;
    EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();
    em.merge(model);
    em.getTransaction().commit();
    result = true;
    return result;
}
```

Pour la récupération des données on a utilisé la syntaxe suivante

```
List<Order> list = new ArrayList<>();
EntityManager em = emf.createEntityManager();
list= em.createQuery("From Order",Order.class).getResultList();
for (Order o : list)
```

Pour la recherche d'un article par son identifiant en utilise la fonction "find" d'Hibernate comme suit:

```

public Product getSingleProduct(int id) {
    EntityManager em = emf.createEntityManager();
    return em.find(Product.class, id);
}

```

Finalement on a implémenté la classe service c'est la classe qui sert comme une rest api qui fournit des URL grâce aux méthodes déjà implémentées dans la classe repertoire. Prenons un exemple :

```

@Path("/getSingleProduct/{id}")
@GET
@Produces(MediaType.APPLICATION_JSON)
public Product getProduct(@PathParam("id") int id){
    return dao.getSingleProduct(id);
}

```

C'est la méthode qui a pour URL "<http://localhost:8080/NotreProjetJ2e/api/shop/getSingleProduct/{id}>" C'est une méthode qui appelle la méthode getSingleProduct(id) déjà implémentée dans la classe repertoire et c'est une requête de type GET qui retourne le produit ayant pour identifiant l'id passé en paramètre

La deuxième Partie c'est la création du deuxième projet qui contient le frontend et fait appel à la rest api implémenté dans le premier projet .

La récréation des classes dans le package Entities .

La création de la classe principale qui crée un lien entre le premier et le deuxième projet. Elle permet de reprendre les lien du Rest API et les transformer en méthodes qu'on peut utiliser dans le deuxième projet

prenons un exemple d'une fonction dans la classe principale

```

public static void deleteOrder(int id) {
    WebTarget target = getWebTarget();
    String URL = "/cancelOrder/" + id;
    Response reponse = target.path(URL).request().delete(Response.class);
    System.out.println(reponse);
}

```

Cette fonction permet la suppression d'une commande. Dans cette méthode on a fait appel à l'interface WebTarget qui permet d'exécuter un lien d'une Rest api de type POST, GET, DELETE ... et le transforme en une méthode qui s'appelle deleteOrder(id).

En Fin lieu on a implémenté les servlets et jsp pour la partie front

On a créé une servlet qui est classe Java qui permet de créer dynamiquement des données au sein d'un serveur HTTP. Une servlet pour chaque fonctionnalité à savoir l'ajout au panier, la suppression d'un article, la connexion et la déconnexion...

Prenons l'exemple du servlet "AddToCartServlet"

`@WebServlet(name = "AddToCartServlet", urlPatterns = "/add-to-cart")` . Cette notation c'est pour créer une servlet qui a pour nom "AddToCartServlet" et pour lien "/add-to-cart" qu'on va l'utiliser dans le bouton "AjouterAuPanier" dans le fichier jsp

```
try (PrintWriter out = response.getWriter()) {
    out.print("add to cart servlet");

    ArrayList<Cart> cartList = new ArrayList<>();
    int id = Integer.parseInt(request.getParameter("id"));
    Cart cm = new Cart();
    cm.setId(id);
    cm.setQuantity(1);
    HttpSession session = request.getSession();
    ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute("cart-list");
    if (cart_list == null) {
        cartList.add(cm);
        session.setAttribute("cart-list", cartList);
        response.sendRedirect("index.jsp");
    } else {
        cartList = cart_list;

        boolean exist = false;
        for (Cart c : cart_list) {
            if (c.getId() == id) {
                exist = true;
                out.println("<h3 style='color:crimson; text-align: center'>Item Already in Cart. <a href='\"";
            }
        }
    }
}
```

Dans cette servlet on a implémenté la fonction doGet(), Cette servlet permet d'ajouter un article au panier (Quantité=1). Elle fait appel à la session là où on a enregistré le panier du client pour récupérer les articles précédemment ajoutés s'il existe, puis l'ajout des articles choisis au panier s'il n'existent pas. Si l'article choisi existe déjà le programme affiche un message d'erreur pour informer que l'article existe déjà il suffit juste de changer la quantité au panier directement .

Finalement l'implémentation des fichiers jsp qui servent comme frontend donc l'utilisation des balises html et d'une bar de navigation ainsi que des boutons qui exécutent le code implémenté par les servlets ou qui servent pour la redirection . Prenons comme exemple le fichier

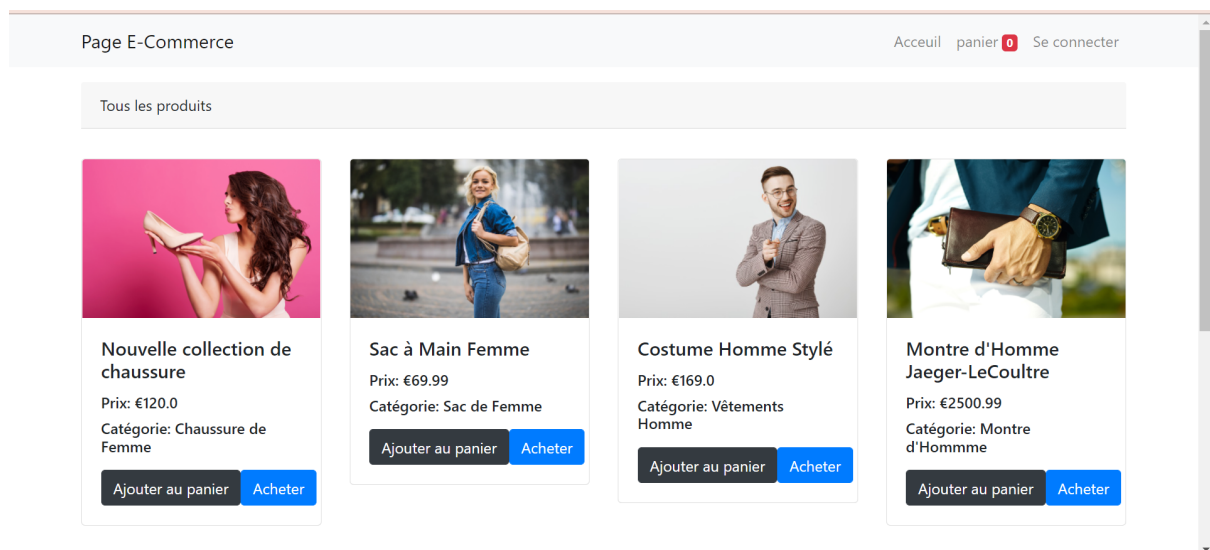

```

<head>
<body>
<%
if (cart_list != null) {
for (Cart c : cartProduct) {
%>
<tr>
<td><%=c.getName()%></td>
<td><%=c.getCategory()%></td>
<td><%= dcf.format(c.getPrice())%></td>
<td>
<form action="order-now" method="post" class="form-inline">
<input type="hidden" name="id" value="<%= c.getId()%>" class="form-input">
<div class="form-group d-flex justify-content-between">
<a class="btn btn-sm btn-incre" href="quantity-inc-dec?action=inc&id=<%=c.getId()%>"><i class="fas fa-pl
<input type="text" name="quantity" class="form-control" value="<%=c.getQuantity()%>" readonly>
<a class="btn btn-sm btn-decre" href="quantity-inc-dec?action=dec&id=<%=c.getId()%>"><i class="fas fa-mi
</div>
<button type="submit" class="btn btn-primary btn-sm">Acheter</button>
</form>
</td>
<td><a href="remove-from-cart?id=<%=c.getId() %>" class="btn btn-sm btn-danger">Supprimer</a></td>
</tr>
<%
}}%>
</body>

```

Les différentes interfaces de notre projet se présentent comme suit .

C'est la page d'accueil qui a pour lien "<http://localhost:8080/projet2Ecommerce/index.jsp>"
C'est la page qui contient tous les articles précédemment ajoutés dans une base de données MySQL, chaque article contient une description, le prix, la catégorie et une photo ainsi que deux boutons pour ajouter au panier qui incrémente l'indice du panier ou acheter qui va nous diriger directement vers la page de connexion si on est pas connecté et la liste des commande si on est en mode connecté




Cette page montre le changement d'indice du panier après l'ajout d'un article au panier


Page E-Commerce

Accueil panier 1 Commandes Se déconnecter


Tous les produits




Nouvelle collection de chaussure
Prix: €120.0
Catégorie: Chaussure de Femme
Ajouter au panier Acheter



Sac à Main Femme
Prix: €69.99
Catégorie: Sac de Femme
Ajouter au panier Acheter



Costume Homme Stylé
Prix: €169.0
Catégorie: Vêtements Homme
Ajouter au panier Acheter



Montre d'Homme Jaeger-LeCoultre
Prix: €2500.99
Catégorie: Montre d'Homme
Ajouter au panier Acheter

Cette page est le panier qui a pour lien "<http://localhost:8080/projet2Ecommerce/cart.jsp>"
C'est la page qui affiche tous les articles mis dans le panier pour un utilisateur donné. Dans cette page on peut modifier la quantité et/ou supprimer l'article du panier. Cette page contient le prix total du panier qui est implémenté dans la classe "calcul"

Page E-Commerce

Accueil panier 1 Se connecter

Prix Total: € 120

Nom	Catégorie	Prix	Buy Now	Cancel
Nouvelle collection de chaussure	Chaussure de Femme	120	<input type="button" value="+"/> <input type="text" value="1"/> <input type="button" value="-"/> <input type="button" value="Acheter"/>	<input type="button" value="Supprimer"/>

Cette page est la page de connexion qui a pour lien ["http://localhost:8080/projet2Ecommerce/login.jsp"](http://localhost:8080/projet2Ecommerce/login.jsp) cet interface permet de saisir les données de connexion qui sont déjà enregistré dans la base de données .

Exemple de mail: projet@gmail.com

Mot de passe : 123456

Le bouton de Login permet la redirection vers la page d'accueil

Page E-Commerce
Accueil panier 1 Se connecter

address Email

projet@gmail.com

Mot de passe

.....

Login

Cette page a pour lien ["http://localhost:8080/projet2Ecommerce/orders.jsp"](http://localhost:8080/projet2Ecommerce/orders.jsp) elle contient les commandes d'un utilisateur donné. On peut également annuler une commande en la supprimant.

Page E-Commerce
Accueil panier 0 Commandes Se déconnecter

Toutes les commandes

Date	Nom	Catégorie	Quantité	Prix	Annuler
2022-04-25	Nouvelle collection chaussure	Chaussure de Femme	1	120	Annuler Commande

IV. Conclusion

La création d'un site e-commerce pourra se faire de plusieurs manières et technologies, en utilisant la technologie J2EE avec tous les outils tels que JSP, Servlet, REST et autres, nous permettent de combiner beaucoup de compétences soit en Front-end ou en Back-end dans un seul IDE, cela est très pratique et robuste.

Notre solution peut être améliorée en ajoutant l'option d'ajouter des nouveaux articles, donc on ajoutera un autre type d'authentification qui auront ce droit.

[Lien vers le projet :](#)

https://www.mediafire.com/file/nfge7zshwbji1os/Projet_J2EE_-_Projet_-_EL_ASRI_-_FERCHICHI.zip/file

[Lien de la vidéo explicative :](#)

<https://youtu.be/cXJnz0I3WLU>