

GAMO Christian
KAFRAOUI Yassine
SOUCCOUCHETTY Darlène
M2 MIAGE SITN

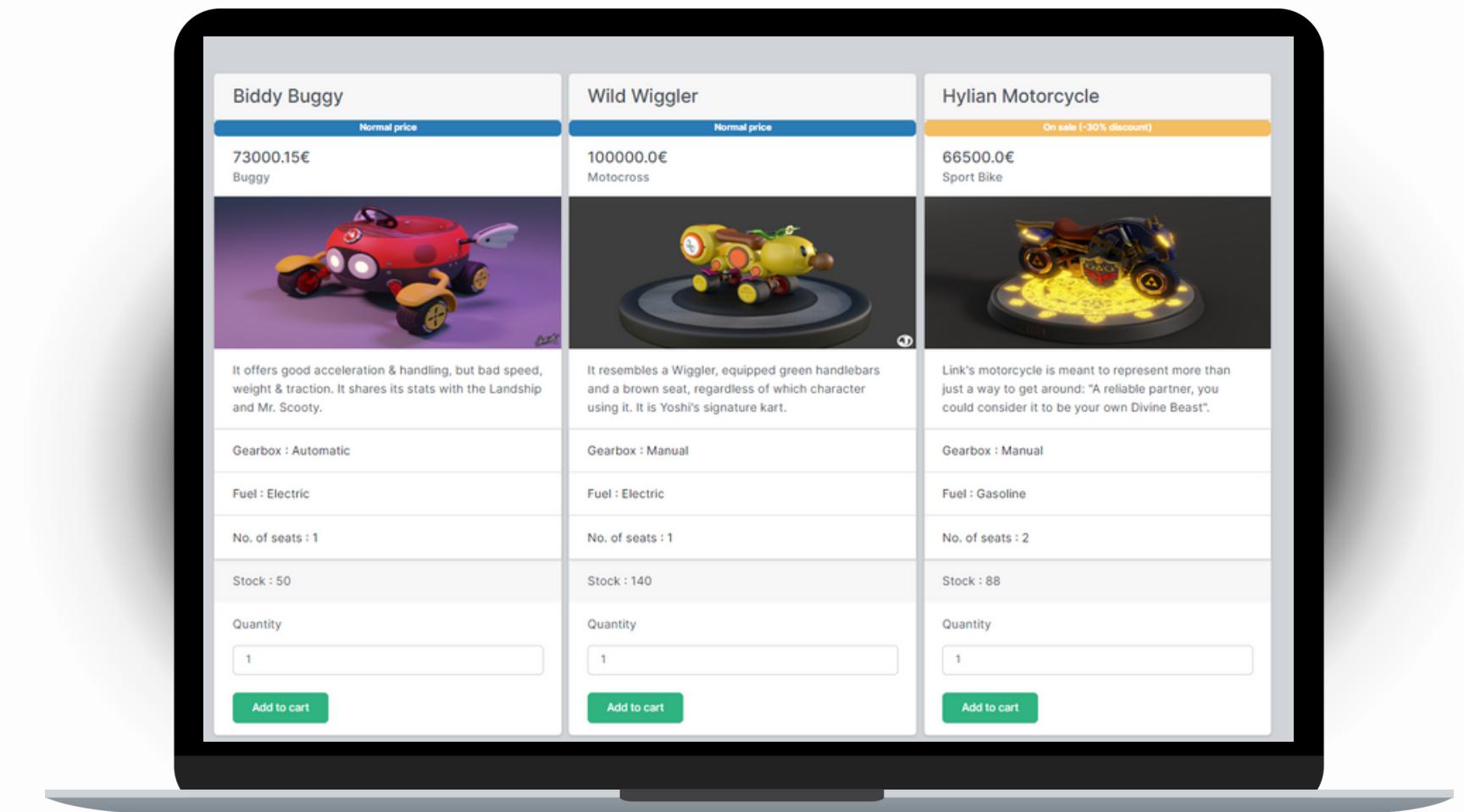


Enseignants : ARAUJO SOBRAL Diogo
BEKHOUCHE Abdesslem

U.E : Infrastructures & Frameworks pour Applications WEB

WigglerKart

Gestionnaire de vente de véhicules



Lien du repository GitHub : https://github.com/christian-gamo/Projet_IFAW

Sommaire



01

Introduction

- Jakarta EE / JSTL en bref
- Différence entre une plateforme et un framework
- L'historique de cette plateforme
- Objectifs de la plateforme Jakarta EE
- Plateformes & Frameworks concurrents
- Licensing et présence sur cloud

02

Caractéristiques techniques

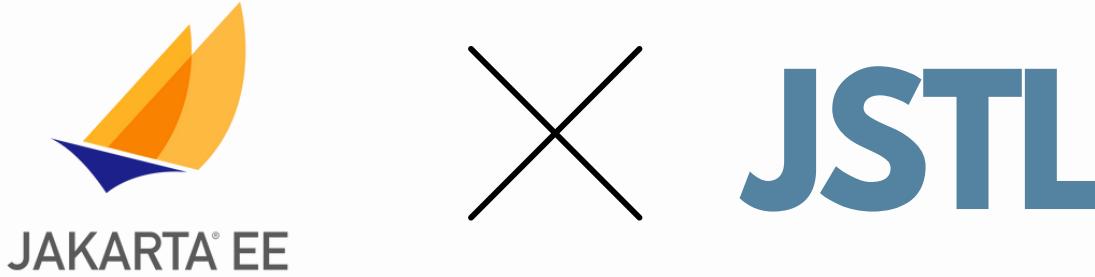
- Fonctionnalités, langages et notion de tags
- Avantages et inconvénients : benchmark par l'exemple
- Architecture et écosystème

03

Présentation de l'application

- Revue détaillée des fonctionnalités
- Architecture fonctionnelle
- Interface et exemple de parcours utilisateur
- Architecture technique
- Structuration de code et concepts de POO (design patterns)
- Gestion de projet

Java EE / JSTL en bref



En quelques mots, JEE / JSTL, c'est quoi ?

- **Jakarta EE (autrefois connue sous l'abréviation J2EE pour Java 2 Platform, Enterprise Edition)**, Jakarta Enterprise Edition n'est pas un framework à proprement dit mais une plateforme orientée serveur pour le développement et l'exécution d'applications WEB distribuées en langage Java. **JEE est plutôt orienté sur des aspects back-end côté serveur** et peut-être utilisé conjointement avec des technologies front-end.
→ **Dans ce contexte, cette plateforme est pertinente pour la création d'applications d'entreprises**

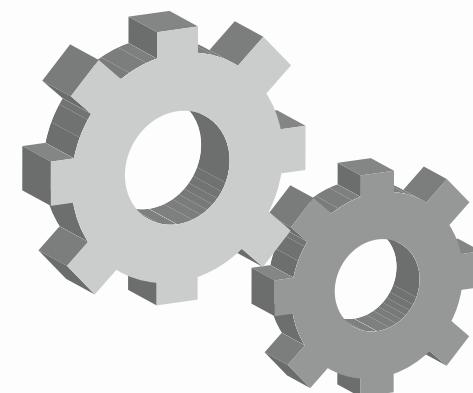
Elle fournit un ensemble de spécifications, de services et de APIs (Application Programming Interfaces) pour simplifier le développement, le déploiement et la gestion de ces applications.

- **JSTL (JavaServer Pages Standard Tag Library)** est une bibliothèque de balises standard utilisées pour la création des pages WEB JSP (JavaServer Pages) dynamiques. Elle fournit des balises prédéfinies qui permettent d'accomplir des tâches courantes dans les pages JSP, sans utiliser directement du code Java.

Différence entre une plateforme et un framework

Plateforme

- Environnement logiciel complet (services et outils pour le développement, le déploiement et l'exécution d'applications).
- Inclut un ensemble de bibliothèques, de compilateurs, de services systèmes, de débogueurs...
- **Exemples** : Android platform, JEE, Node.js

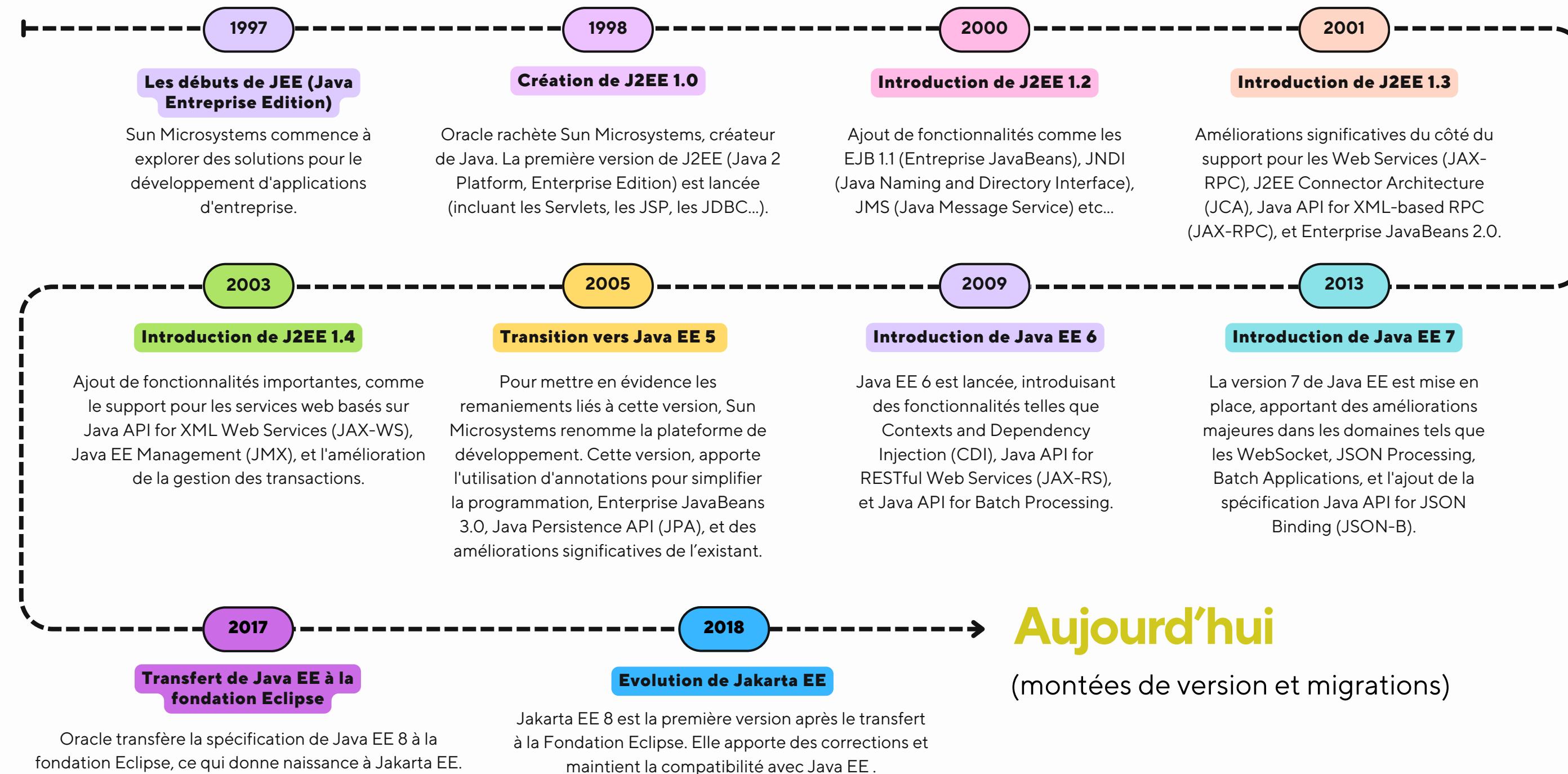


Framework

- Structure logicielle qui fournit un ensemble de bibliothèques et conventions pour faciliter le développement d'applications.
- Peut être spécifique à un domaine applicatif (applications WEB, applications mobiles...).
- **Exemples** : Django, Spring, Ruby on Rails



L'historique de cette plateforme



Objectifs de la plateforme Jakarta EE

Développement d'Applications d'Entreprise

Jakarta EE fournit un ensemble de spécifications et de composants pour faciliter le développement d'applications d'entreprise, telles que les applications de gestion, les systèmes de traitement des transactions, les solutions e-commerce, etc.

Architecture Distribuée

La plateforme prend en charge le développement d'applications distribuées, où les composants peuvent être déployés sur plusieurs machines.

Interopérabilité et Connectivité

Cela inclut la prise en charge des services web, permettant aux applications JEE de communiquer avec d'autres applications, indépendamment de la plateforme ou du langage de programmation utilisé.

Sécurité

La plateforme intègre des mécanismes de sécurité robustes pour protéger les applications d'entreprise contre les menaces. Cela inclut l'authentification, l'autorisation, le chiffrement des données...

Portabilité

Les applications Jakarta EE peuvent être déployées sur différents serveurs d'applications dédiés sans nécessiter de modifications significatives.

Compatibilité ascendante

Jakarta EE maintient la compatibilité ascendante avec Java EE pour assurer une transition numérique de l'existant plus "douce".

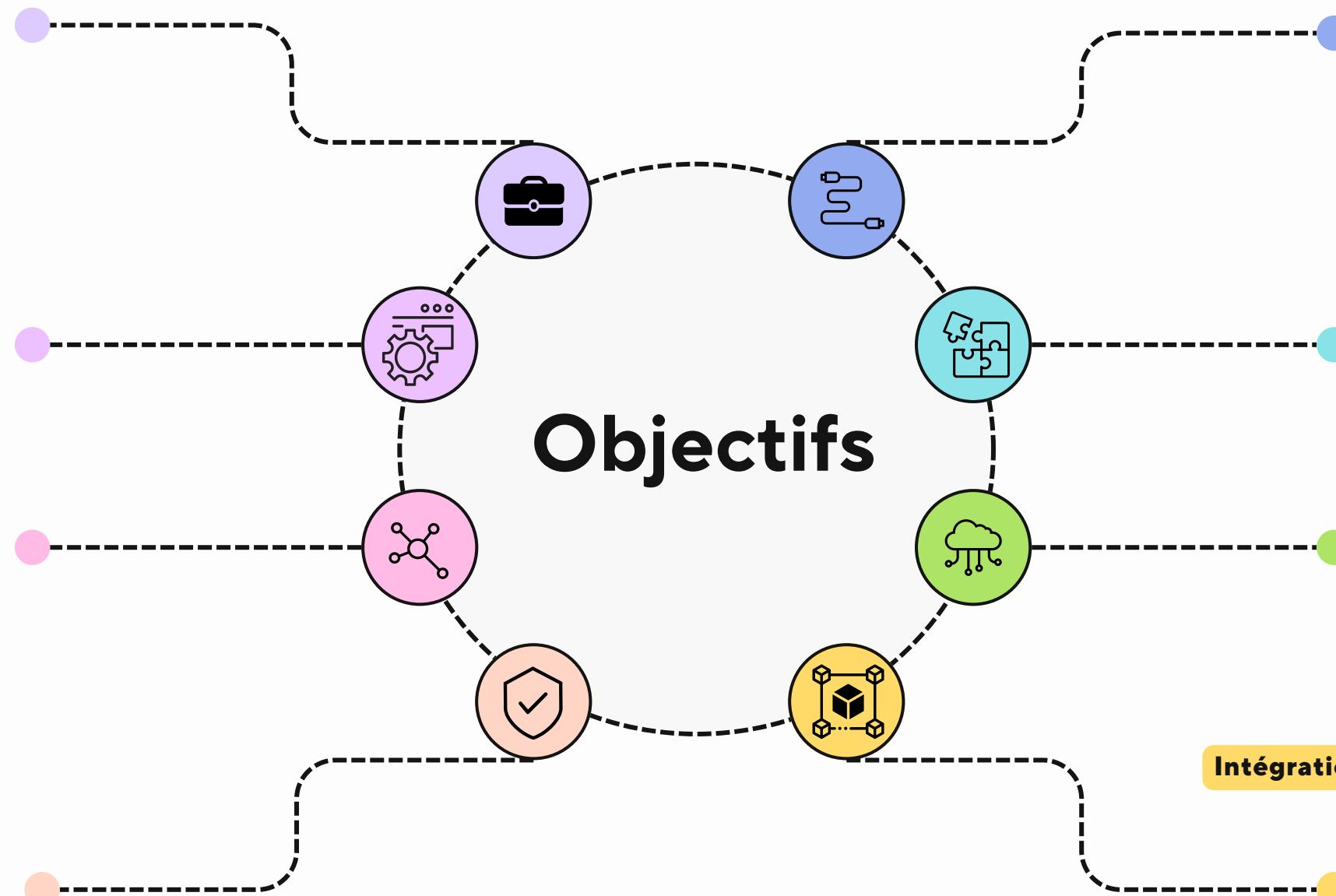
Support de Nouveaux Modèles d'Architecture

Jakarta EE s'adapte pour supporter de nouveaux modèles d'architecture, tels que l'architecture microservices, tout en continuant de fournir des solutions pour les architectures monolithiques plus anciennes.

Intégration avec les Nouvelles Tendances Technologiques

Jakarta EE intègre des technologies émergentes, telles que le cloud, les conteneurs (ex. Docker), les orchestrations de conteneurs (ex. Kubernetes), et les pratiques DevOps.

Objectifs



Plateformes & Frameworks concurrents

Une liste non exhaustive de frameworks concurrents basés sur le langage Java (ou non !)

Framework Spring



- Communauté active et framework populaire
- Framework riche et uniformisant la conception applicative : Spring Boot, Spring Data...
- Injection de dépendances facilitée

Framework Express



- Segmentation logique des fonctionnalités par composants (plus de productivité et moins de redondances)
- Adapté aux applications basées sur des scénarios événementiels (Javascript).

Framework Symfony



- Propose un large éventail de "bundles" prêts à l'emploi et standardisés.
- Intégration avec un ORM (doctrine) pour une gestion transparente des entités via une approche orientée objet des données.

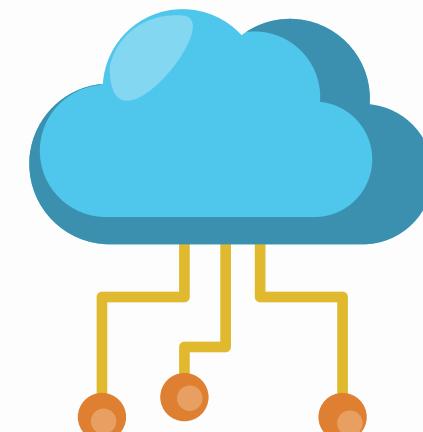


Licensing et présence sur cloud

Une plateforme open-source, communautaire, et basée sur la collaboration

Licensing

- Distribué sous l'Eclipse Public License (une licence open source approuvée par l'Open Source Initiative (OSI)).
- Cette licence est de type “copyleft faible” (plus permissive des conditions sur l'utilisation, la modification et la distribution d'un code source Jakarta EE, même dans des projets propriétaires).
- Les implémentations de JEE doivent suivre les spécifications et normes Java pour assurer l'interopérabilité avec les autres technologies Java.



Présence sur cloud

- Déploiement possible des applications Jakarta EE sur des plateformes PaaS ou SaaS cloud telles que Amazon Web Services, Microsoft Azure, Google Cloud Platform, Red Hat OpenShift, Oracle Cloud, IBM Cloud...
- Prise en charge des architectures de microservices et les applications cloud-native (utilisation possible de conteneurs comme Docker / orchestrateurs de conteneurs comme Kubernetes pour construire des solutions distribuées évolutives).
- Jakarta EE est compatible avec divers services cloud, tels que les bases de données cloud, les services de messagerie cloud, les services de stockage cloud, etc... (l'interaction avec ces services se fait via des spécifications standard Java comme Java Database Connectivity, Java Message Service..)

Fonctionnalités, langages et notion de tags (1)

3 grandes familles d'API - **SDK de base**

Composants

- **Servlets** : Utilisé pour créer des applications web côté serveur. Les Servlets gèrent les requêtes HTTP et génèrent des réponses dynamiques.
- **JSP (JavaServer Pages)** : Utilisé pour créer des pages web dynamiques en intégrant du code Java dans du HTML. JSP simplifie la création d'interfaces utilisateur.
- **EJB (Enterprise JavaBeans)** : Utilisé pour la logique métier dans les applications Jakarta EE. Les EJB offrent des services tels que la gestion des transactions, la sécurité, et la persistance des données.

Services

- **JDBC (Java Database Connectivity)** : Fournit une API pour l'accès aux bases de données relationnelles.
- **JTA/JTS (Java Transaction API/Service)** : Gère les transactions, assurant la cohérence des opérations sur la base de données.
- **JNDI (Java Naming and Directory Interface)** : Permet l'accès aux services de nommage et aux annuaires d'entreprises.
- **JCA (Java Connector Architecture)** : Fournit une architecture pour connecter des applications Jakarta EE à des systèmes d'entreprise externes.
- **JAAS (Java Authentication and Authorization Service)** : Facilite l'échange sécurisé de données.

Communication

- **RMI-IIOP (Remote Method Invocation over Internet Inter-ORB Protocol)** : Permet l'utilisation d'objets Java distribués en utilisant le protocole CORBA.
- **JMS (Java Message Service)** : Facilite la communication asynchrone entre différentes parties d'une application à l'aide de messages orientés middleware.
- **Java Mail** : Fournit une API pour l'envoi et la réception d'e-mails à partir d'applications Jakarta EE.

Fonctionnalités, langages et notion de tags (2)

Modules additionnels - Bibliothèques / spécifications / API

Quelques exemples non présents dans le SDK de base (à déclarer dans le fichier pom.xml (dependencies))

- **Contexts and Dependency Injection (CDI)** : Librairie qui fournit un mécanisme d'injection de dépendances et de gestion des contextes dans une application JEE.
- **Java Persistence API (JPA)** : Librairie qui définit une interface de programmation pour la gestion de la persistance des objets en utilisant les bases de données relationnelles (Hibernate, Eclipse Link).
- **JavaServer Faces (JSF)** : Fournit un modèle de composants réutilisables pour la construction d'interfaces utilisateur riches.
- **Java API for JSON Binding (JSON-B)** : Fournit une API pour la sérialisation / désérialisation d'objets Java en format JSON pour faciliter l'échange de données entre les applications WEB et les clients.
- **JavaServer Pages Standard Tag Library (JSTL)** : Librairie qui fournit une collection de balises prédéfinies utilisées pour simplifier le développement de pages JSP (incorporation de logique Java Servlet et accès aux données "sans écrire de code Java").

Fonctionnalités, langages et notion de tags (3)

Focus sur JSTL

La librairie JSTL est basée sur le **langage EL (Expression Language)**, qui fournit une syntaxe simplifiée pour accéder aux objets Java et aux données dans une page JSP, sans avoir à écrire de code Java “conventionnel”.

→ **En pratique...** - *Exemple de la récupération d'un attribut de session (récupération du modèle d'une voiture)*

Java “Conventionnel”

```
<%= session.getAttribute("car").getModele() %>
```



JSTL

```
 ${sessionScope.personne.nom}
```

Variable

- PageScope
- RequestScope
- SessionScope
- ApplicationScope

Portée

- Page WEB
- Requête HTTP
- Session HTTP
- Application

Avantages et inconvénients : benchmark par l'exemple

Avantages JEE / JSTL



- **Architecture d'entreprise** : Conçu pour le développement d'applications d'entreprise robustes et extensibles.
- **Interopérabilité** : Favorise l'interopérabilité en fournissant des spécifications standardisées pour les services tels que la persistance des données etc...
- **Scalabilité** : Conçues pour être hautement évolutives en utilisant des architectures distribuées et des composants réutilisables.
- **Sécurité** : Propose des mécanismes de sécurité intégrés, ce qui est essentiel dans les applications d'entreprise où la protection des données sensibles est cruciale.
- **Réutilisabilité** : Les balises JSTL favorisent la réutilisation du code en permettant aux développeurs de créer des composants personnalisés pour effectuer des tâches courantes.
- **Lisibilité / Maintenance** : JSTL fournit un ensemble de balises prédéfinies qui encapsulent des fonctionnalités couramment utilisées, telles que les boucles, les conditions, les accès aux collections, etc. Cela permet de réduire la quantité de code Java directement incorporé dans les pages JSP ce qui le rend plus lisible pour des développeurs plus familiers aux langages front-end comme le HTML.

Inconvénients JEE / JSTL



- **Complexité** : En raison de sa nature orientée entreprise, Jakarta EE peut être complexe, surtout pour les petites applications.
- **Ressources système** : Les applications JEE peuvent nécessiter des ressources système importantes et coûteuses en raison de leur architecture robuste.
- **Obsolescence** : Les applications WEB JEE sont peu à peu remplacées par des applications WEB basées sur Sharepoint, Node.Js, Angular, Svelte qui sont plus populaires.

Architecture et écosystème

Servlets et JSP

Couche présentation

Enterprise JavaBeans (EJB), JMS (Java Message Service), JTA (Java Transaction API)

Couche métier

JCA (Java Connector Architecture), JPA (Java Persistence API)

Couche d'intégration

JNDI (Java Naming and Directory Interface), JTA (Java Transaction API), JavaMail

Couche d'infrastructure

Revue détaillée des fonctionnalités

Fonctionnalités

Prise en charge de différentes catégories de véhicules (avec possibilité d'ajout d'une nouvelle catégorie de véhicules)

Affichage des véhicules proposés à la vente au sein d'un catalogue

Effectuer des recherches dans le catalogue à l'aide de mots clés ou opérateurs logique

Passer la commande d'un véhicule

Gestionnaire de commandes permettant de calculer des taxes en fonction du pays de livraison du véhicule, de gérer les commandes payées au comptant ou celles assorties d'une demande de crédit et de gérer les différents états de la commande

Construction d'une liasse de documents au format PDF et HTML (Demande d'immatriculation, certificat de cession et bon de commande)

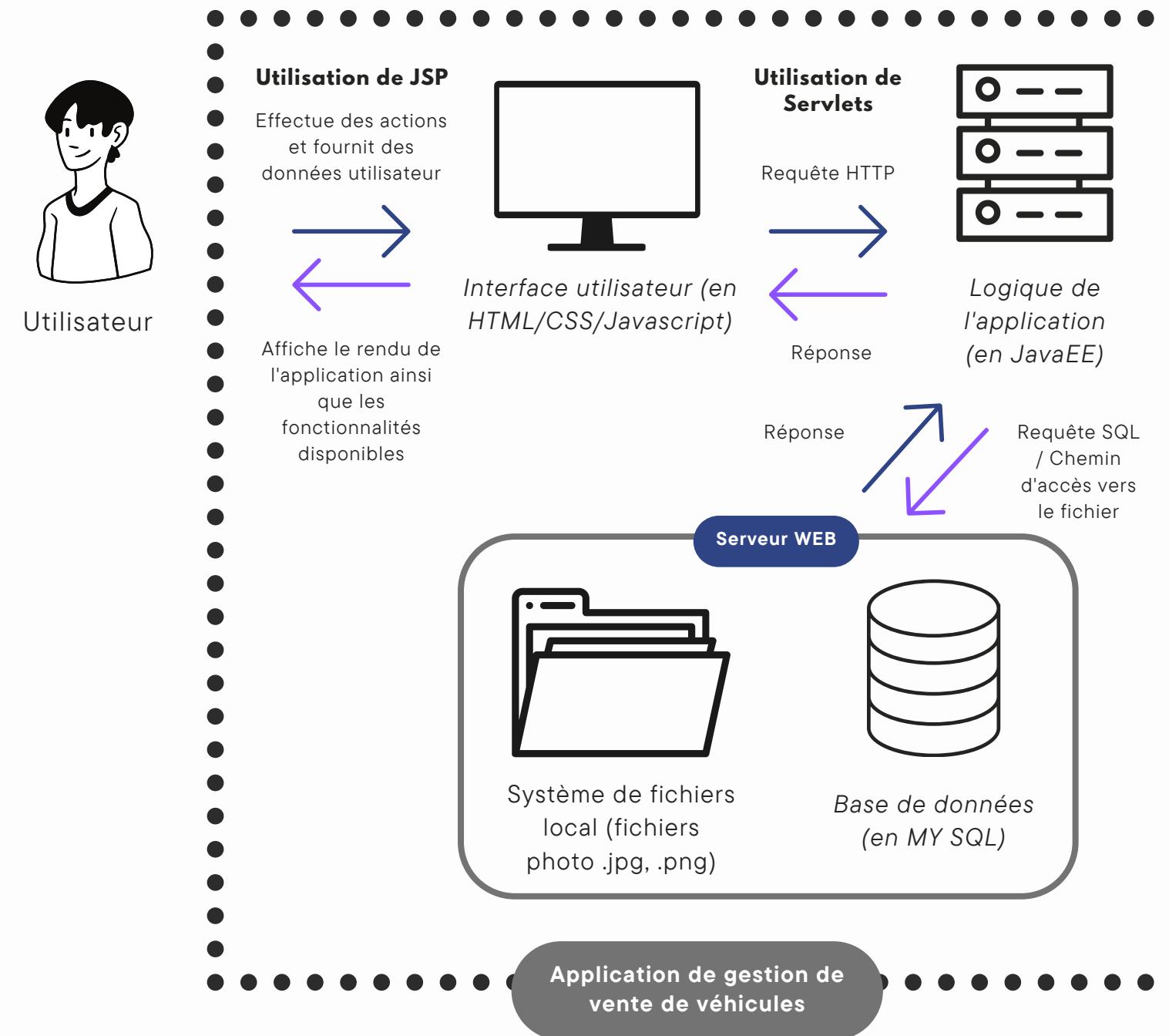
Système permettant de solder les véhicules difficiles à vendre (ceux qui sont dans le stock depuis longtemps)

Gestion des différents types de clients (Sociétés, Particuliers, Administrateur)

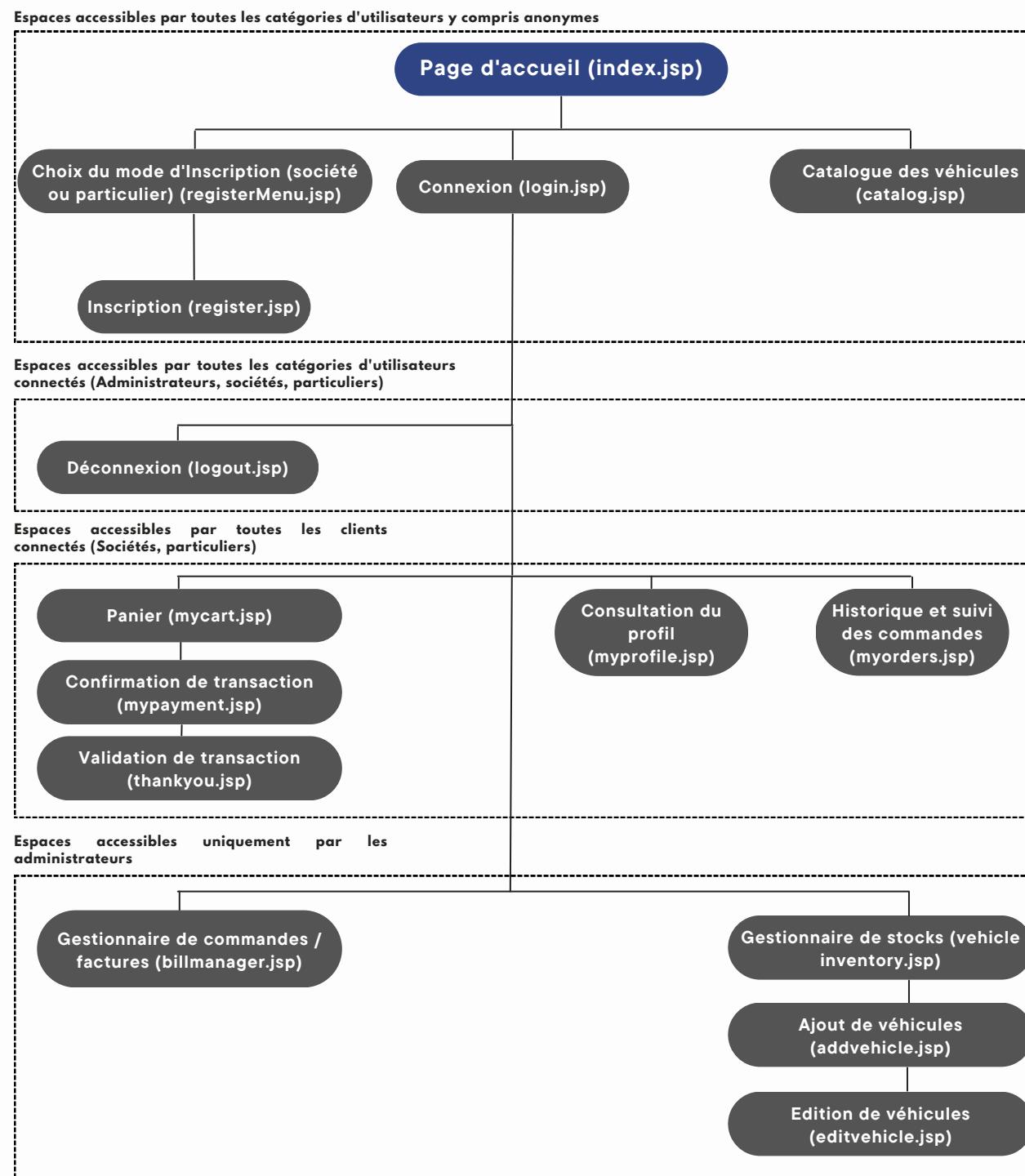
Inscription / Connexion / Déconnexion / Espace utilisateur

Gestionnaire de stocks.

Architecture fonctionnelle (1)

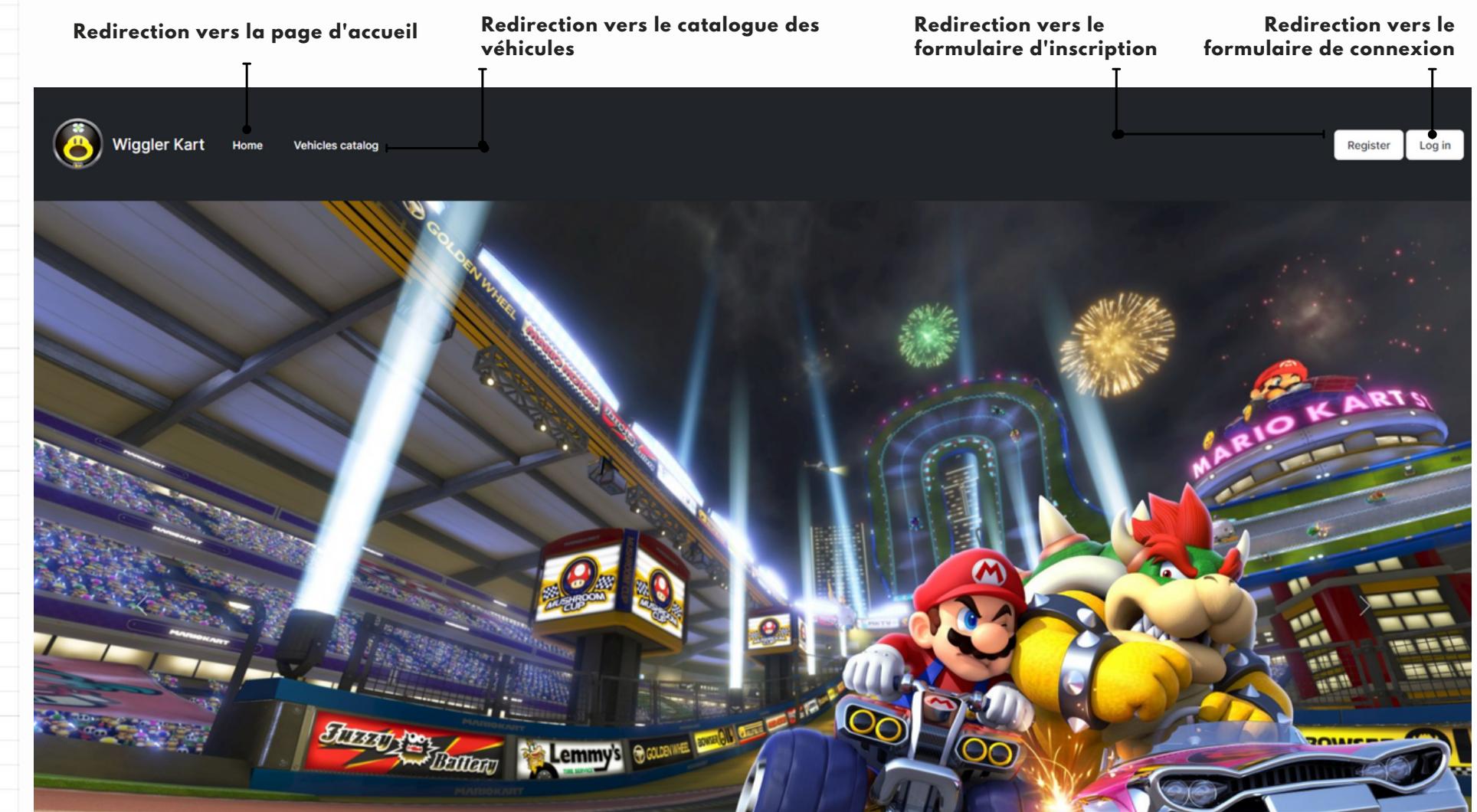
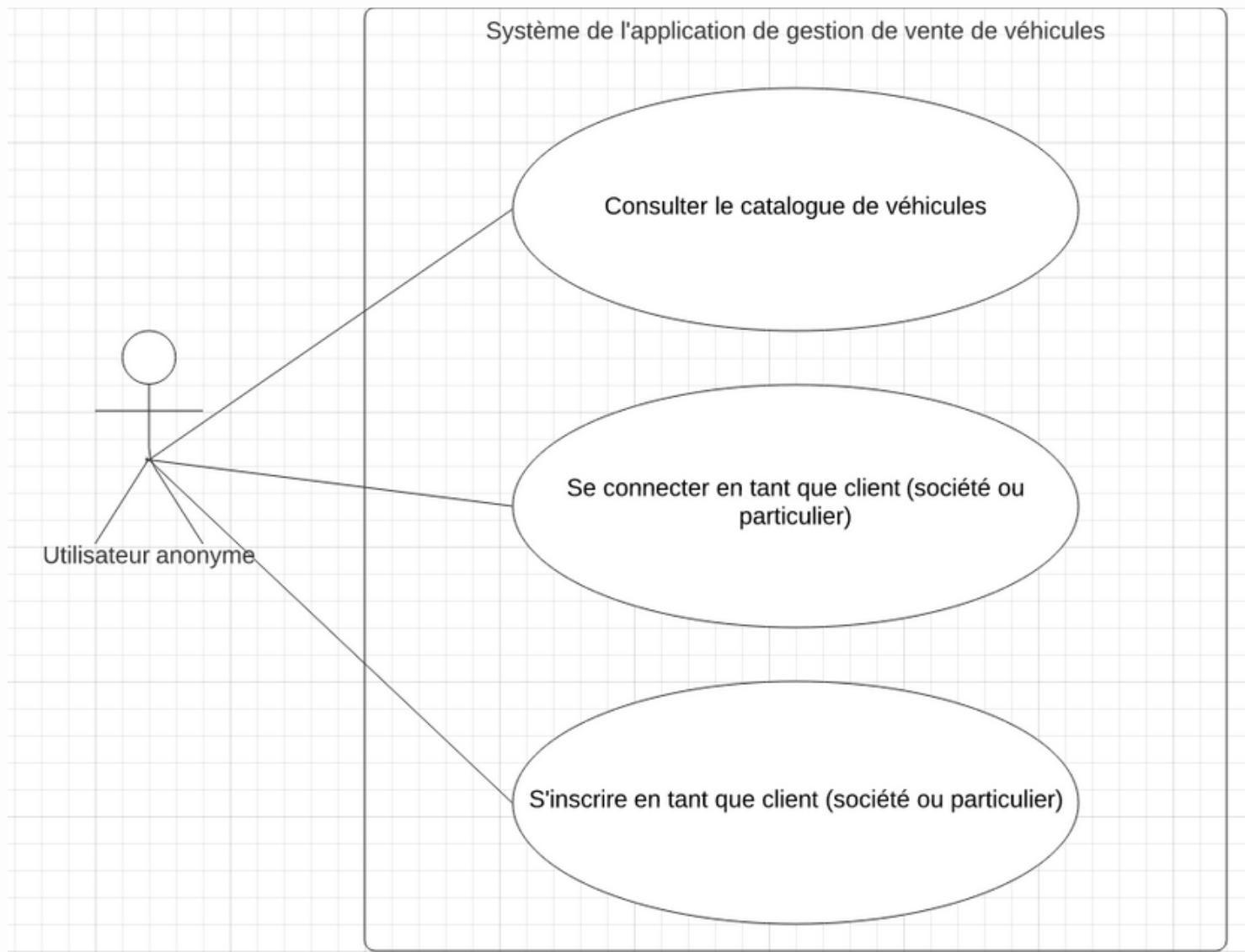


Architecture fonctionnelle (2)



Interface et exemple de parcours utilisateur (1)

Parcours d'un utilisateur anonyme dans l'application



Interface et exemple de parcours utilisateur (2)

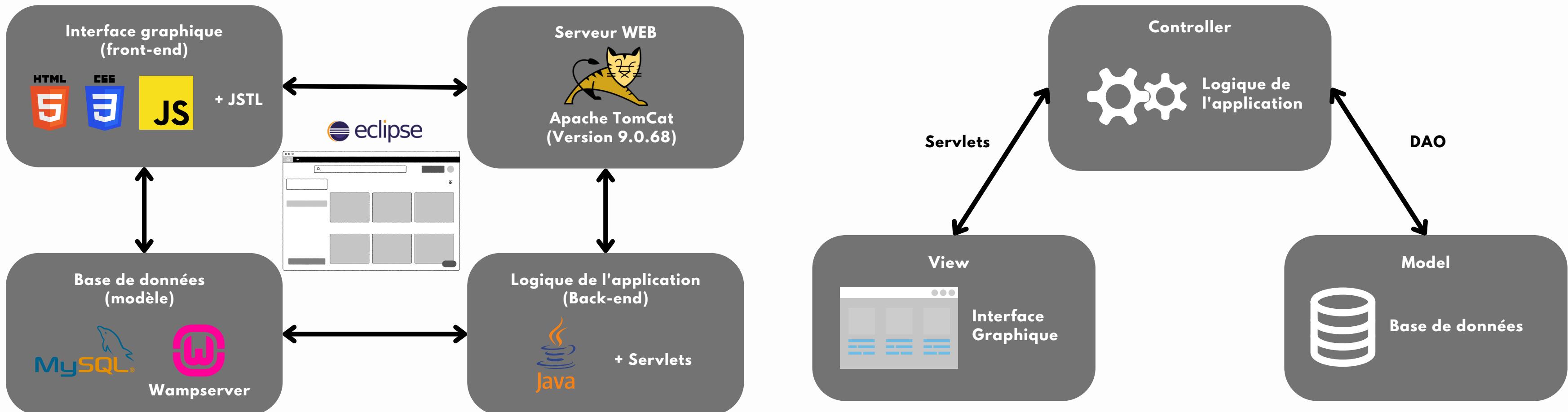
Parcours d'un utilisateur anonyme dans l'application

The diagram illustrates the user flow for an anonymous user in the application, divided into three main sections:

- Barre de filtres (Filter Bar):** Located on the left, this section contains dropdown menus for "Type of vehicle", "Type of fuel", "Type of gearbox", and "Number of seats" (set to 4), along with a "Special offers" dropdown and a "Price" dropdown. A "Search" button is at the bottom.
- Fiche produit d'un véhicule (Vehicle Product Detail):** This section displays three vehicle models: Biddy Buggy, Wild Wiggler, and Hylian Motorcycle. Each card includes the vehicle name, price (Normal price or On sale), type, and a small image. Below each card is a detailed description and technical specifications (Gearbox, Fuel, Seats, Stock). At the bottom of each card is a "Login as a customer to add a vehicle to your cart" button.
- Redirection vers le formulaire d'inscription en tant que particulier (Redirection to the individual registration form):** This section shows a registration dialog with two buttons: "Register as a simple user" (highlighted in black) and "Register as a company". Arrows point from the "Register as a simple user" button to the "Redirection vers le formulaire d'inscription en tant que particulier" label and from the "Register as a company" button to the "Redirection vers le formulaire d'inscription en tant que société" label.
- Saisie de l'email (Email Input):** This section shows an input field labeled "Email address" with placeholder text "Enter your email".
- Saisie du mot de passe (Password Input):** This section shows an input field labeled "Password" with placeholder text "Enter your password".
- Connexion (avec vérification des identifiants et du respect de la casse exigée) (Login (with identifier verification and case sensitivity requirement)):** This section shows a "Log in" button.

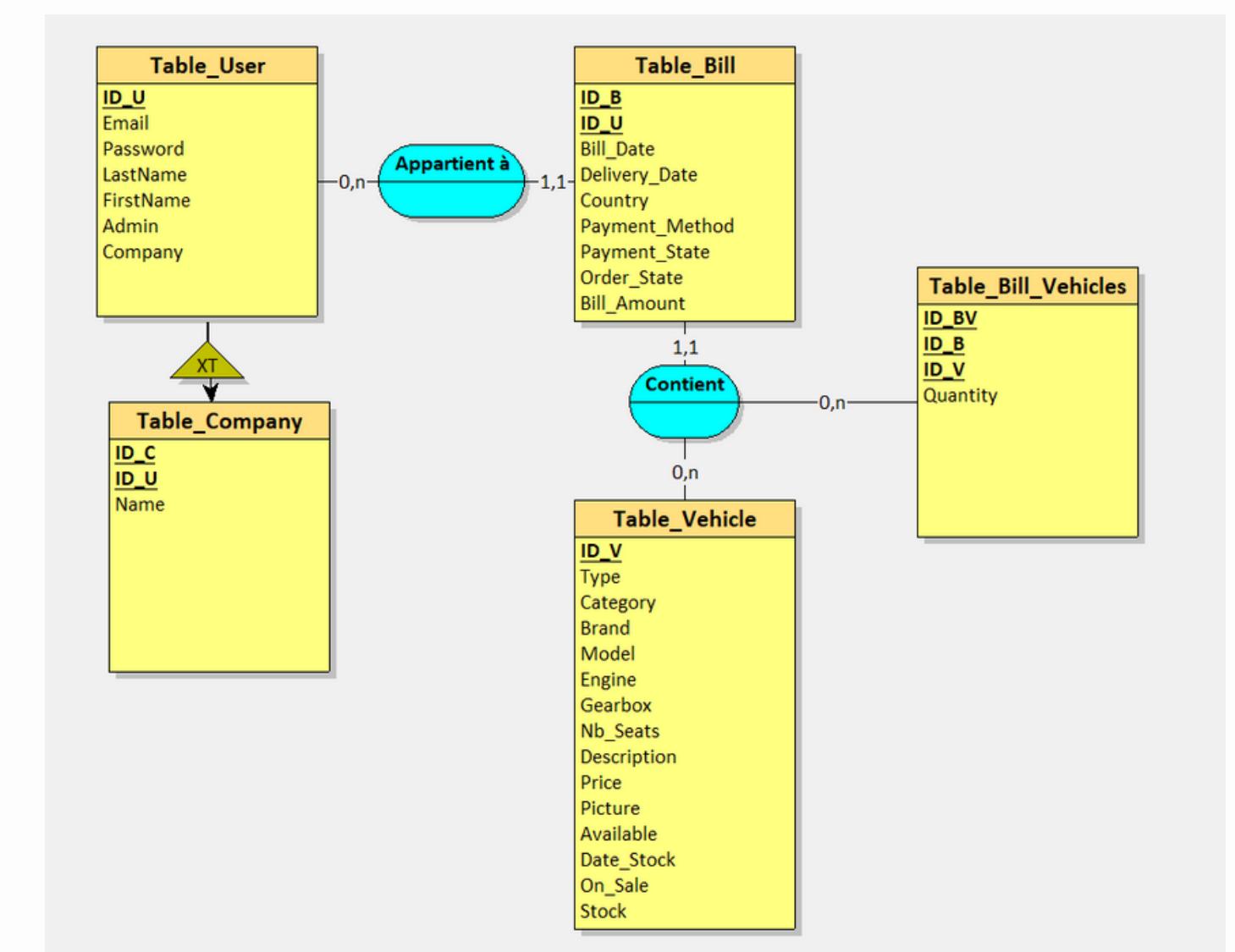
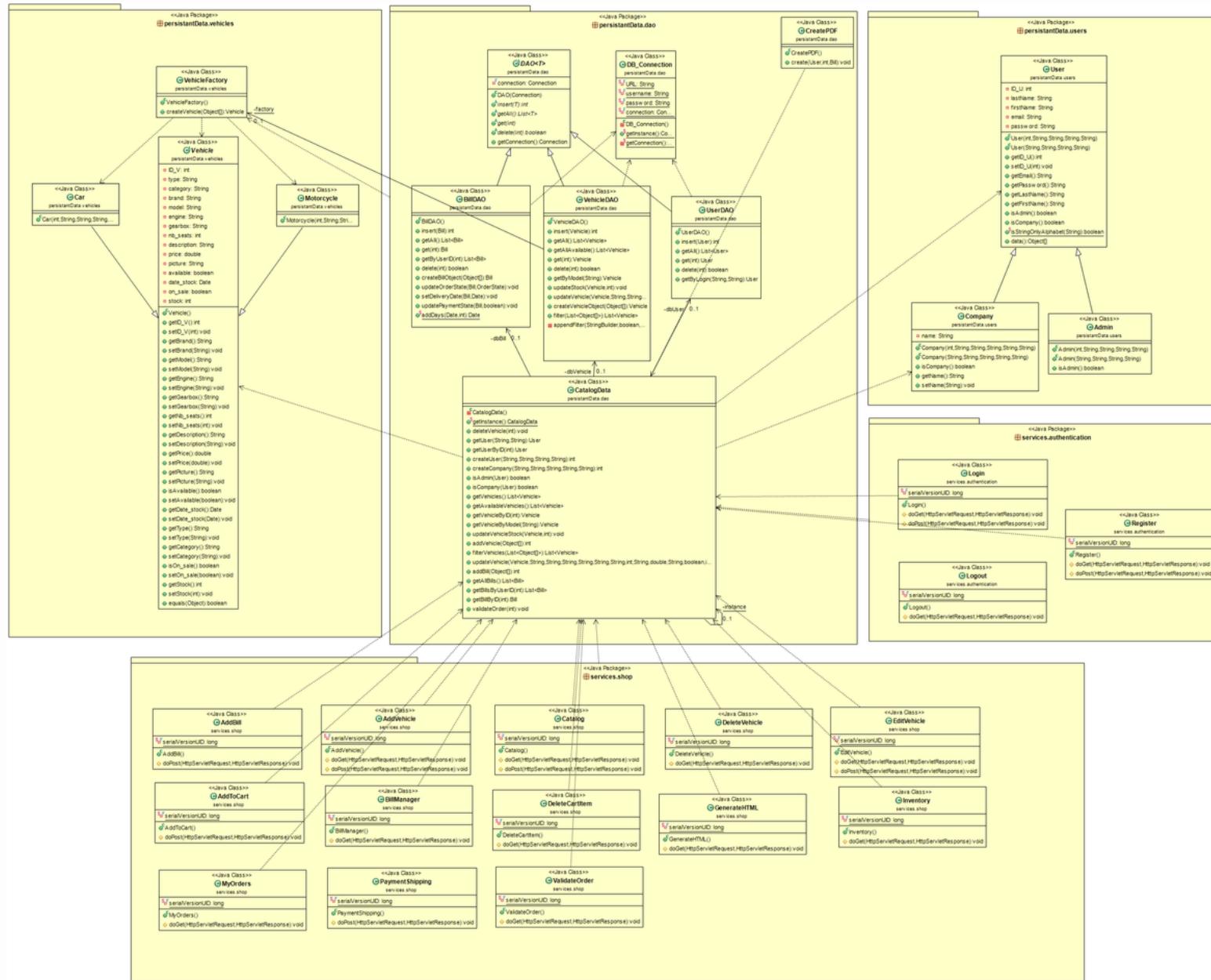
Architecture technique (1)

Technologies utilisées et application du modèle MVC



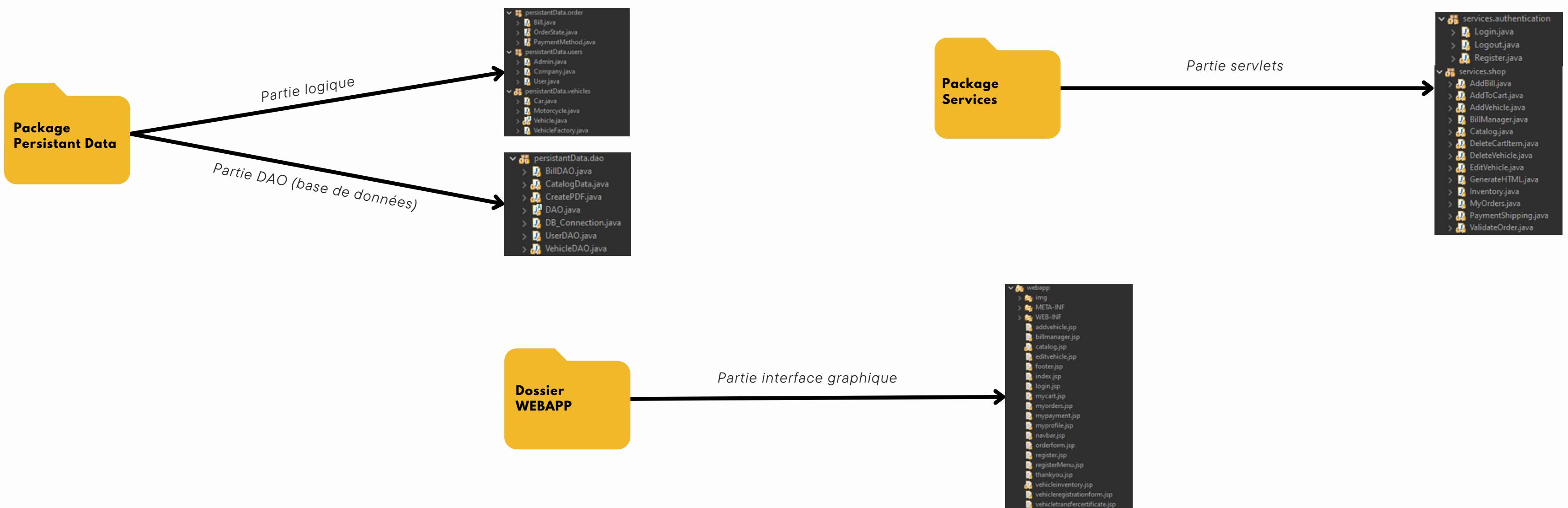
Architecture technique (2)

Diagramme de classe & Modèle Conceptuel de Données (MCD)



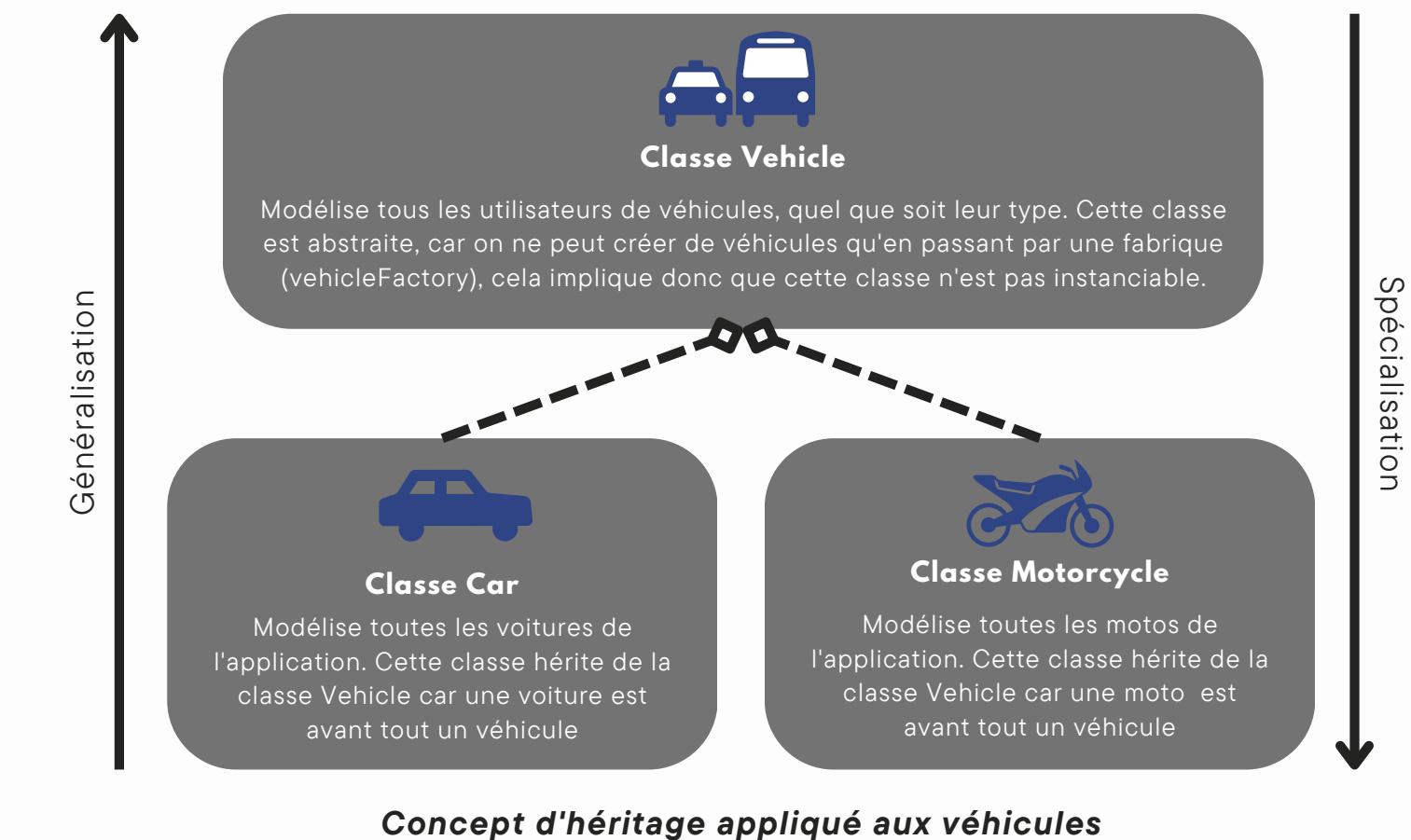
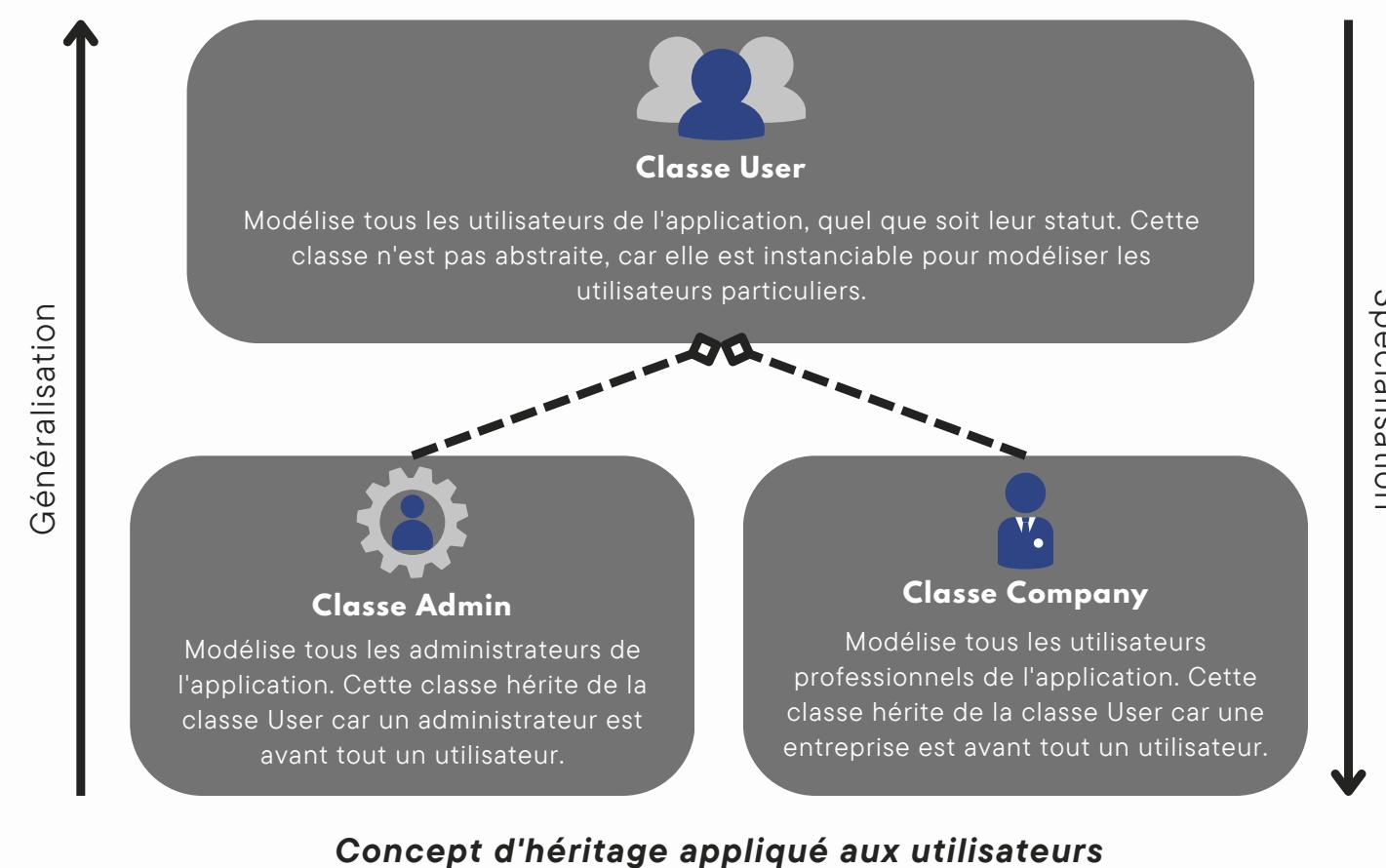
Architecture technique (2)

Structuration du code



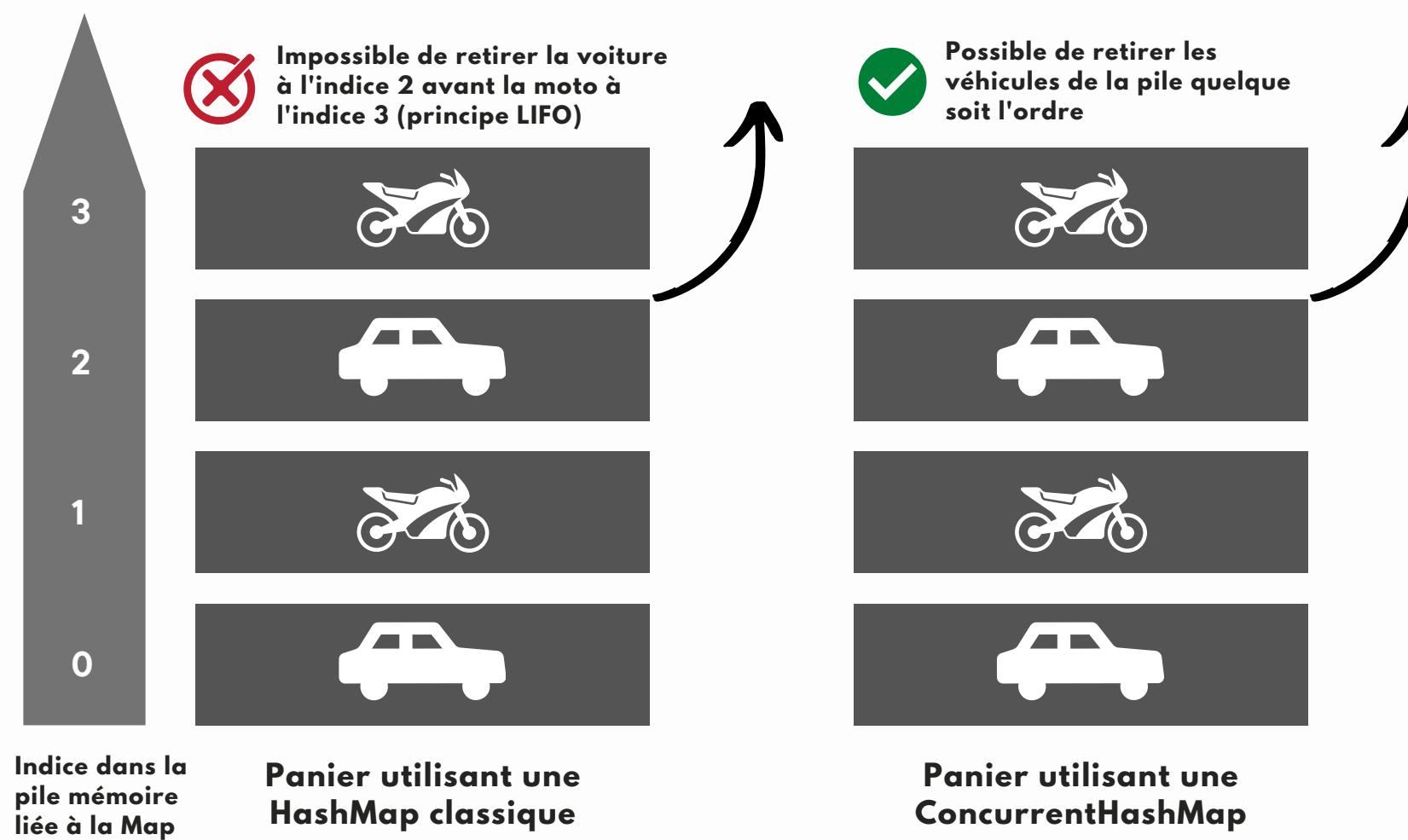
Structuration de code et concepts de POO (design patterns) (1)

Les bases du langage Java, aussi applicables dans un contexte d'entreprise !



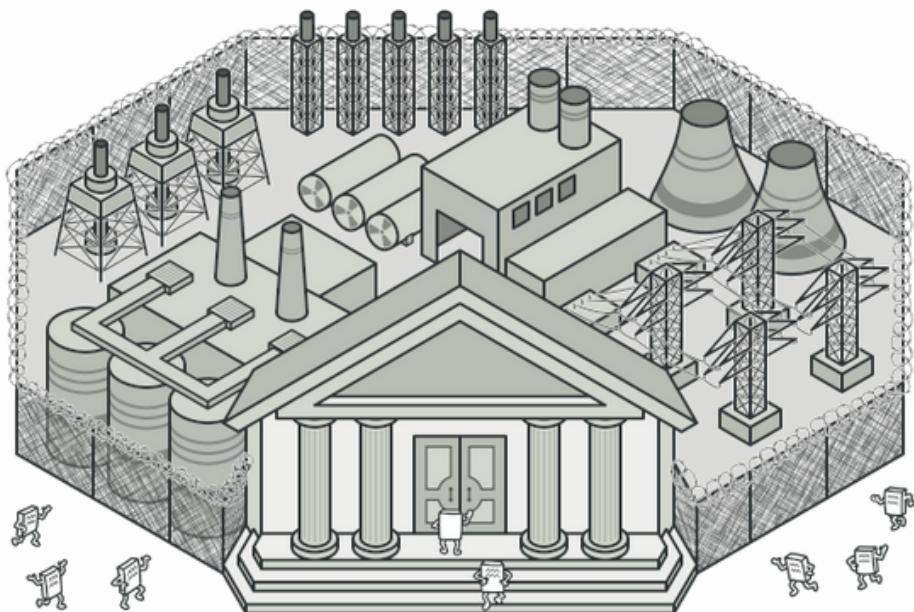
Structuration de code et concepts de POO (design patterns) (2)

L'importance du choix de structures de données adéquates en POO

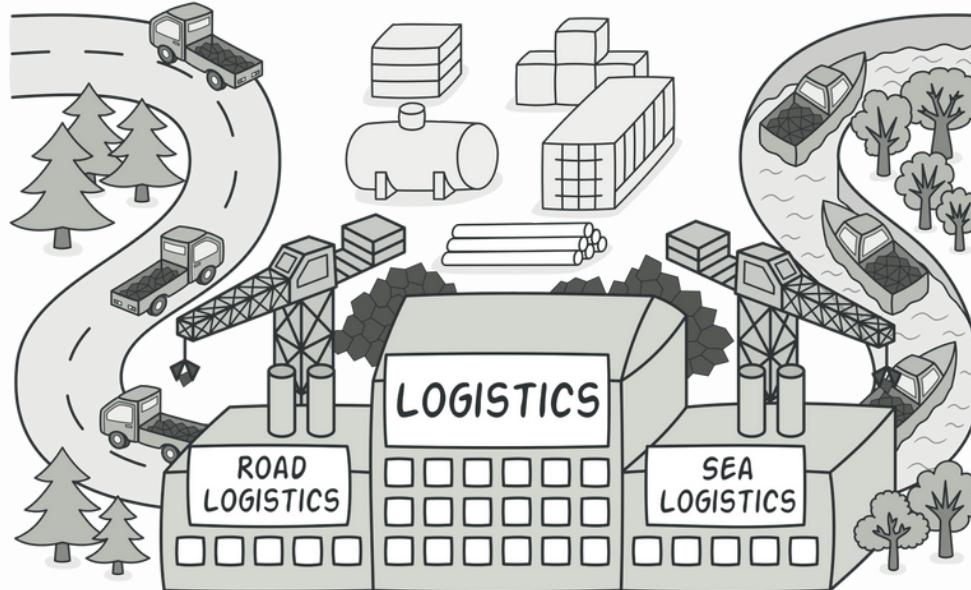


Structuration de code et concepts de POO (design patterns) (3)

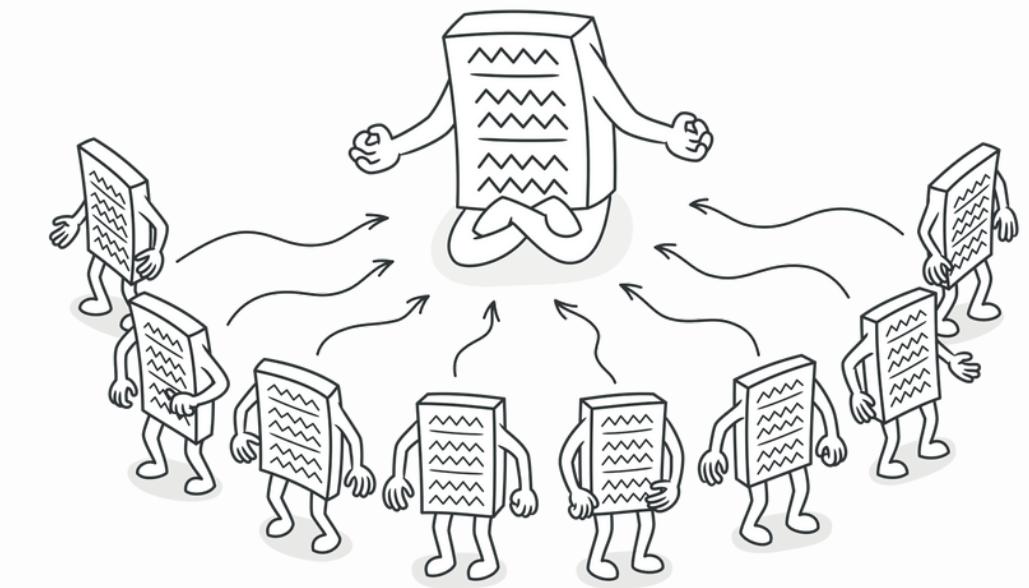
Des designs patterns pour anticiper des potentiels remaniements / évolutions de l'application



Façade



Factory Method



Singleton

Gestion de projet



2 semaines de développement dont
• 3/4 logique applicative + servlets
• 1/4 JSP

Codage en sessions collaboratives



1 semaine de code review
• Test JUnit basiques
• Utilisation de Sonarlint