

26/08/2016

Spécifications d'Architecture Logicielle  
Projet Cantine Aston



	<b>Document</b> : <i>Spécifications d'Architecture Logicielle</i> <b>Projet</b> : <i>Cantine Aston</i>	<b>Réf.</b> : <i>SAL-06062016</i> <b>Version</b> : <i>V3.0</i> <b>Date</b> :
	Référentiel documentaire projet SQLI	Page 1 sur 49

Rédaction	
Rédigé par :	
Date	
Vérifié par :	
Date	
Approuvé par :	
Date	

Historique du Document			
Version	Date	Modifiés Par	Actions / Commentaires
1.0			
2.0			
3.0			

	<b>Document</b> : <i>Spécifications d'Architecture Logicielle</i> <b>Projet</b> : <i>Cantine Aston</i>	<b>Réf.</b> : <i>SAL-06062016</i> <b>Version</b> : <i>V3.0</i> <b>Date</b> :
	Référentiel documentaire projet SQLI	Page 2 sur 49

## Table des matières

I.	Introduction.....	3
1.	Objectif du document.....	3
2.	Définitions, acronymes et abréviations.....	4
II.	Spécifications techniques.....	5
1.	Représentation architecturale.....	5
2.	Objectifs architecturaux et contraintes.....	6
3.	Vue des cas d'utilisation.....	8
A.	Réalisation des cas d'utilisation.....	8
B.	Cas d'utilisations de la Cantinière.....	9
C.	Cas d'utilisation de l'utilisateur.....	23
4.	Vue logique.....	34
A.	Base de données.....	34
B.	Procédures stockées.....	36
C.	DAO.....	39
D.	Services.....	40
E.	Web Service.....	42
5.	Taille et performance.....	47
A.	Taille de l'application.....	47
B.	Performance de l'application.....	48

	<b>Document</b> : <i>Spécifications d'Architecture Logicielle</i> <b>Projet</b> : <i>Cantine Aston</i>	<b>Réf.</b> : <i>SAL-06062016</i> <b>Version</b> : <i>V3.0</i> <b>Date</b> :
	Référentiel documentaire projet SQLI	Page 3 sur 49

# I. Introduction

La société Stone Education est un organisme de formation professionnel diplômant reconnu par l'Etat qui accueille de nombreux stagiaires.

Actuellement, l'entreprise prend les commandes de repas du midi de façon manuelle. Les étudiants doivent se déplacer le matin pour passer leur commande. Ou bien, venir directement à l'heure du midi pour récupérer un plat.

Afin de simplifier et améliorer ce système, l'entreprise souhaite se doter d'une application simple de gestion des commandes entièrement informatisée permettant de diminuer le temps d'attente pour chaque utilisateur.

L'application permettra aux utilisateurs de passer commande de leur plat pour le jour même ou pour les autres jours de la semaine, à condition que l'utilisateur soit détenteur d'un compte et que le solde de ce celui-ci permette de régler la commande.

## 1. Objectif du document

Le projet Cantine est destiné à la mise en place d'une application de gestion électronique des commandes pour la cantine de l'école Aston. A ce titre, le présent document a pour but de présenter les aspects techniques de la solution.

Il fera partie des documents de référence lors des problématiques de développement et d'évolution de l'application pour l'architecte, les concepteurs et superviseurs du projet.

L'application Cantine offre à l'utilisateur la possibilité de commander des plats et des formules via un navigateur web.

Elle permettra aussi au personnel de la cantine de gérer une quantité d'articles disponibles à la commande ainsi qu'une heure limite de commande par jour.

Le paiement étant géré par un système de cagnotte qui est approvisionnée par l'utilisateur auprès de la cantinière, il n'est pas prévu de mise en place de système de paiement en ligne d'aucune sorte.

## 2. Définitions, acronymes et abréviations

Sont répertoriés et définis dans cette section les abréviations et acronymes susceptibles d'être rencontrés au long du document.

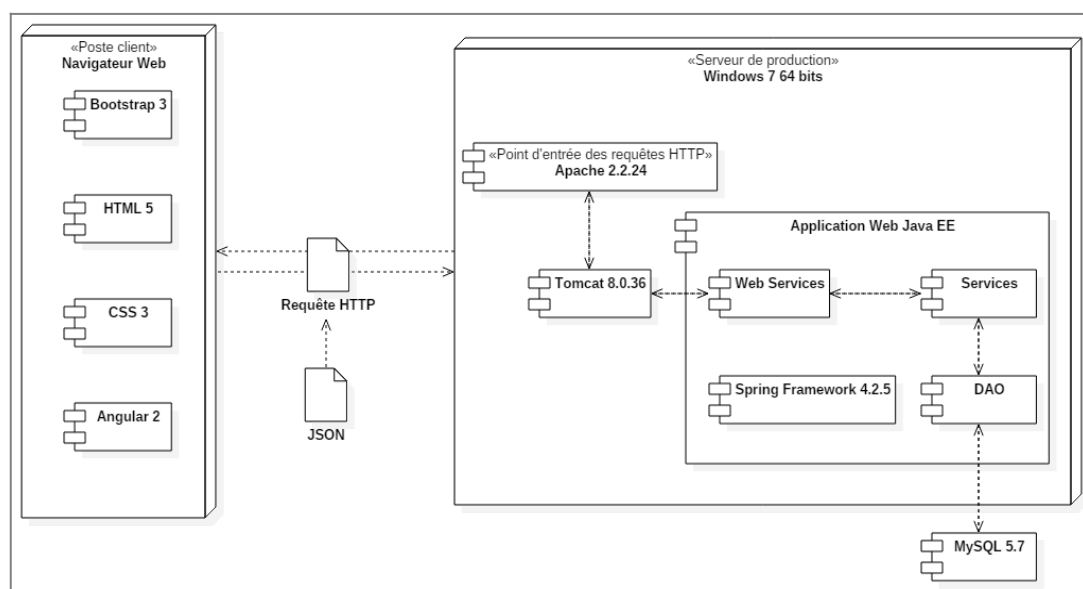
Acronyme/ Abréviation	Signification	
JVM	Java Virtual Machine	Permet au code JAVA d'être interprété convenablement par un ordinateur
DAO	Data Access Object = Objets d'accès aux données	Les objets contenant les méthodes et attributs nécessaires à la communication avec les modèles de données
DTO	Data Transfer Object = Objet de transfert des données (conteneurs)	Les objets contenant les attributs nécessaires au transport des données issues des modèles de données
API	Application Programming Interface = Interface de programmation	Offre au développeur un panel d'éléments utilisables dans le cadre du développement
REST	Representational State Transfer = Processus d'échange de données	Désigne un concept de programmation où les parties impliquées n'ont pas nécessairement connaissance les unes des autres de manière permanente
NPM	Node Package Manager	Un logiciel de gestion de dépendance de bibliothèque JavaScript
UI IHM	User interface = Interface Homme Machine	Les éléments graphiques affichés sur les écrans des terminaux des utilisateurs
URL	Uniform Ressource Locator	Localisateur uniforme de ressource, adresse web vers laquelle des requêtes sont adressées afin de récupérer du contenu
WS	Web Service	Fournisseur de données via une requête web (HTTP/S)
MCD	Modèle Conceptuel de Données	Un schéma représentant l'ensemble des tables de données et des relations
SGBDR	Système de Gestion de Base De Données Relationnelles	
ORM	Object Relationnal Mapping	Librairie permettant de lier les classes d'une application à un schéma de données, simplifiant l'accès aux données

## II. Spécifications techniques

### 1. Représentation architecturale

L'application est basée sur une architecture 3-tiers de type REST :

- La présentation de l'information est dédiée au poste client
  - Application responsive Cross-Platform
- La logique métier contenue dans l'application JAVA est gérée par la couche service qui évolue sur 2 niveaux
  - Les Web Services, points d'accès de l'API renvoyant les résultats au format JSON
  - Les services contenant la majeure partie de la logique applicative
- L'accès aux données, toujours dans l'application JAVA, est pris en charge par les DAO et les DTO



- Poste Client

Le poste client est une plate-forme informatique physique (ordinateur, smartphone ou tablette) d'où les requêtes de ressources sont envoyées au serveur d'application. La présentation des ressources demandées est affichée au travers d'un navigateur web.

- Serveur de production
  - Apache

Il reçoit et résout les requête HTTP, dans notre cas, il les dirige vers le serveur web Tomcat.

- Tomcat

Il reçoit et traite les requêtes transmises par Apache en faisant appel aux applications nécessaires.

- Application JAVA

La solution Cantine Aston développée en JAVA.

- Base de données MySQL

	<b>Document</b> : <i>Spécifications d'Architecture Logicielle</i> <b>Projet</b> : <i>Cantine Aston</i>	<b>Réf.</b> : <i>SAL-06062016</i> <b>Version</b> : <i>V3.0</i> <b>Date</b> :
	Référentiel documentaire projet SQLI	Page 6 sur 49

Il fournit à l'application les données requises.

## 2. Objectifs architecturaux et contraintes

Dans l'objectif de fournir une solution fonctionnelle dans le temps imparti, diverses librairies et logiciels sont utilisés au cours de ce projet afin d'en optimiser le temps de développement.

Nous utilisons Maven afin de faciliter la gestion de dépendance des librairies Java présentent dans le projet. Ce programme nous permet aussi de gérer la compilation et la génération des livrables après s'être assuré que les tests unitaires prévus soient validés.

Le gestionnaire de version Git nous permet de pouvoir travailler de façon organisé en équipe et de gérer un historique de développement du projet.

Le projet est séparé en deux dépôt distincts du côté du gestionnaire version, un projet « Back End » et un projet pour l'interface « Front End ». Chacun disposant d'une branche de développement principale. Chaque membre peut décider de créer et de travailler sur sa propre branche s'il décide de travailler sur une fonctionnalité spécifique pour laquelle il estime intéressant de séparer le code (ex. Création d'une fonctionnalité hors projet, correction de bug signalé)

Librairies utilisées :

- Angular v2.0.0-rc.5 : Création des pages web
- Bootstrap v3.3.6 : Design graphique des pages web
- Log4J v2.6.1 : Log côté JAVA
- Json-simple v1.1.1 : Transformation des données retournées par les services vers le format JSON qui sera retourné par les Web Services
- SpringFramework v4.3.1 : Gestion des dépendances dans le projet
- JUnit v4.12 + JMeter v3.0 : Tests unitaires

Logiciels utilisés :

- Eclipse Mars / Neon : Edition du code Java
- Atom v1.8.0 : Edition du code HTML5 / CSS3 / JavaScript / Typescript
- StarUML : Modélisation UML
- Workbench MySQL v6.3.7 : Modélisation MCD / MLD
- Moqups : Réalisation des maquettes graphiques
- MySQL v5.7 : Gestion des données
- Google drive / BitBucket Wiki : Gestion documentaire
- Git v2.9.0 : Gestion de version du code source
- Node v4.4.7 : Gestion de dépendances des librairies JavaScript
- Typescript 1.8.10 : Compilation du code Typescript vers JavaScript
- Sonar v5.6 : Validation de la qualité du code
- VirtualBox v5.0.20 : Virtualisation de l'environnement de test
- Maven 3.3.9 : Construction du projet
- Apache v2.2 : Serveur frontal
- Apache Tomcat v8.0.36 : Serveur d'application

Le choix de ces librairies et logiciels nous impose un certain nombre de contraintes logicielles.

	<b>Document</b> : <i>Spécifications d'Architecture Logicielle</i> <b>Projet</b> : <i>Cantine Aston</i>	<b>Réf.</b> : <i>SAL-06062016</i> <b>Version</b> : <i>V3.0</i> <b>Date</b> :
	Référentiel documentaire projet SQLI	Page 7 sur 49

Le cahier des charges stipule que le projet doit être implémenté en Java à partir de la version 1.7 minimum. Nous avons décidé d'implémenter notre solution en Java 1.8, celle-ci étant la version release la plus récente.

En conséquence de ce choix nous utiliserons le serveur web Apache Tomcat v8.0, stable et recommandé pour une utilisation avec du code java de version 7 et plus (<http://tomcat.apache.org/whichversion.html>).

Les utilisateurs utilisant une multiplicité de navigateurs internet pour accéder aux applications web, l'application Cantine pourra être utilisée sans difficulté à partir de divers navigateurs (Internet Explorer 9, Firefox v9.0, Chrome v52.0).

Elle sera utilisée sur différents supports utilisant ces navigateurs (Ordinateurs, téléphones mobiles, tablettes tactiles) et bénéficie d'un rendu optimal sur les écrans dont le format s'étend de 5,5 à 21 pouces.

Nous avons choisi d'utiliser le langage de programmation Typescript dans le cadre de la création de composants angular2, aussi nous avons besoin d'un transpiler (aussi appelé compilateur) afin de parser (transformer) notre code Typescript au format JavaScript. Cela nous contraint à installer le logiciel NodeJS, nous donnant accès à NPM, ce dernier nous permettant de télécharger et d'utiliser tsc (Typescript compiler).

Au-delà de ces contraintes logicielles, nous sommes confrontés à des contraintes d'ordre divers.

- **Sûreté**

Afin de s'assurer de l'authenticité de l'utilisateur, un token d'authentification contenant l'identifiant, les informations nominatives et la durée de la session de l'utilisateur est délivré au moment de l'identification ce celui-ci. Ce token sera ensuite échangé lors de chaque requête envoyée depuis le poste client vers le serveur via le HTTP header.

Le token est généré et délivré au poste client à chaque identification, il aura une durée de validité fixée à 1 heure et sera stocké côté client en local storage sous la clé « cantine-token ».

En cas de déconnexion de l'utilisateur, le token préalablement créé et stocké sera détruit.

En cas de d'expiration de la durée de validité du token, celui-ci sera détruit, l'utilisateur sera notifié de l'expiration de sa session et invité à se connecter de nouveau.

- **Confidentialité**

Puisque notre application gère des comptes d'utilisateurs, il est nécessaire de s'assurer que les données sensibles soient difficiles à corrompre. Aussi les données importantes (dans notre cas, le mot de passe) seront cryptées lors des échanges HTTP ainsi qu'en base de données.

En cas d'oubli du mot de passe, l'utilisateur recevra par email un mot de passe généré aléatoirement.

- **Sécurité**

Afin de réduire et de rendre plus difficile le champ des actions malveillantes.

Les données contenues dans le token échangé entre le poste client et le serveur seront cryptées.

Du côté de la base de données, nous mettons en place des procédures stockées, ainsi les appels faits à la base sont limités à des appels de procédures définies.



	<b>Document</b> : <i>Spécifications d'Architecture Logicielle</i> <b>Projet</b> : <i>Cantine Aston</i>	<b>Réf.</b> : <i>SAL-06062016</i> <b>Version</b> : <i>V3.0</i> <b>Date</b> :
	Référentiel documentaire projet SQLI	Page 8 sur 49

Un compte utilisateur MYSQL spécifique avec des droits restreints sera créé dans le cadre de l'application afin d'exécuter les requêtes.

### 3. Vue des cas d'utilisation

#### A. Réalisation des cas d'utilisation

Afin de gérer nos ressources, nous définissons une liste finie de points finaux (end points), vers lesquels nos requêtes seront dirigées.

Le cycle de nos requêtes sera le même pour toutes les actions.

Une requête HTTP est envoyée depuis l'IHM vers un serveur web via un URL, le relais est passé à un WS qui fait intervenir les méthodes des classes de Service appropriées.

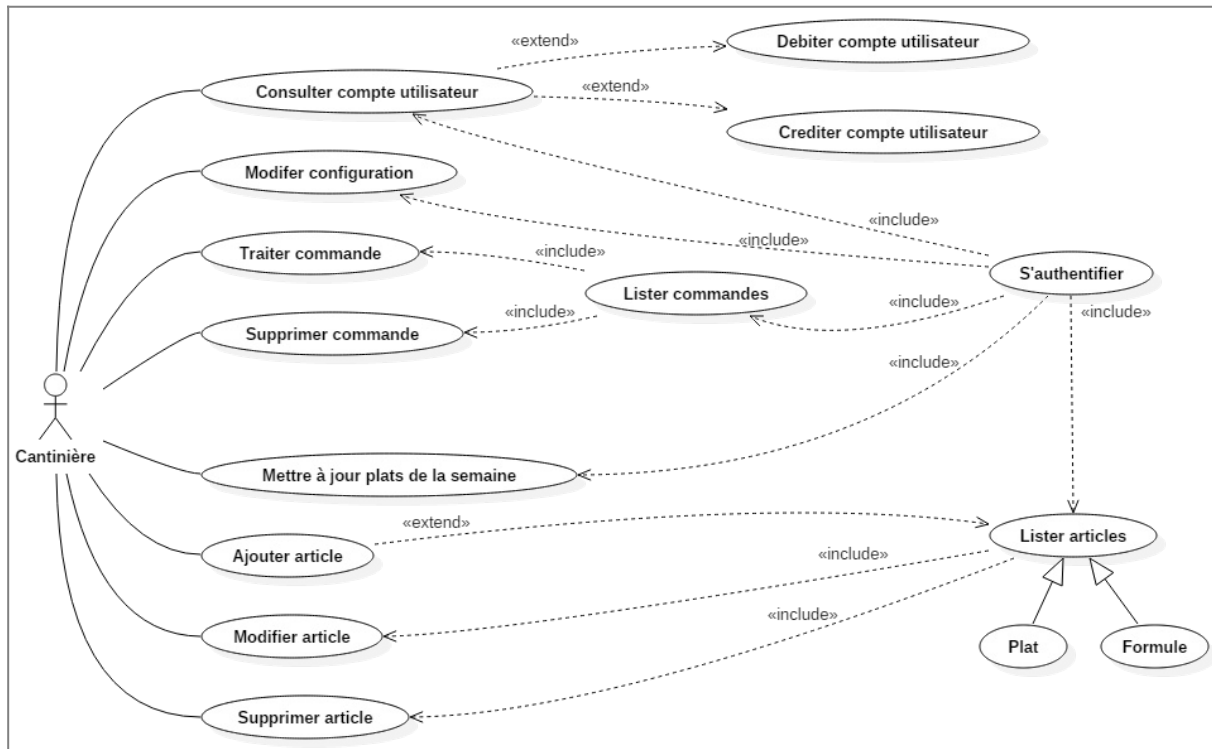
Ces classes de services font alors appel aux classes de DAO qui envoient des requêtes vers la base de données et retournent un état, en fonction duquel la classe de service retournera un résultat au WS qui le délivrera à l'IHM selon un format défini (ici le JSON).

L'affichage est mis à jour après confirmation des actions, en cas d'échec les causes seront affichées à l'utilisateur.


Nous utiliserons les requêtes HTTP basées sur l'url [http\[s\]://www.cantine-aston.fr/api/v1](http[s]://www.cantine-aston.fr/api/v1) afin de gérer nos ressources :

- GET
  - Récupération de liste de ressources (Ex : `/ressource/list[?options]`)
  - Récupération d'une ressource ciblée (Ex : `/ressource/{ressource_id}`)
  - Code de retour 200 dans le cas d'une requête réussie
  - Code de retour 400 dans le cas d'une requête erronée / invalide
  - Code de retour 404 en cas d'échec de la requête
- PUT
  - Création de ressources (Ex : `/ressource/create?options`)
  - Code de retour 200 dans le cas d'une requête réussie
  - Code de retour 404 en cas d'échec de la requête
- PATCH
  - Modification de ressources (Ex : `/ressource/update/{ressource_id}[?options]`)
  - Code de retour 200 dans le cas d'une requête réussie
  - Code de retour 404 en cas d'échec de la requête
- DELETE
  - Suppression d'une ressource ciblée (Ex : `/ressource/delete/{ressource_id}`)
  - Code de retour 200 dans le cas d'une requête réussie
  - Code de retour 404 en cas d'échec de la requête

## B. Cas d'utilisations de la Cantinière



➤ *S\'authentifier :*

Plats de la Semaine
Carte
Se Connecter


Login :

Password :

Se connecter
Créer un compte

Annuler
Mot de passe oublié

Recap des Commandes
Plats de la Semaine
Carte
Compte Utilisateur
Configuration

Bonjour Françoise

Lundi

Mardi

Mercredi

Jeudi

Vendredi

▼ Client	▼ Commande	▼ Emporté	▼ Annuler
Kevin Lorem	Plat 1	<input type="checkbox"/>	
Stéphanie Briere	Plat 2	<input checked="" type="checkbox"/>	
Lorem Bah	Plat 1	<input checked="" type="checkbox"/>	

Quantités du jour :

Plat 1 - 15

Plat 2 - 5

L'utilisateur non connecté, depuis n'importe quelle page accessible, clique sur le bouton « Se Connecter », une fenêtre modale s'affiche.

Après avoir renseigné ses identifiants, l'utilisateur clique sur le bouton « Se Connecter », un appel à la méthode « UserWS.authenticate (String email, String password) » est envoyé, elle invoque à son tour la méthode « IUserService.authenticate (String email, String password) » qui fait appel à la méthode « IUserDAO.selectByEmailPassword (String email, String password) » qui retourne l'utilisateur correspondant aux critères si celui-ci existe.

Si la valeur de l'attribut « isCooker » de l'utilisateur est FALSE l'utilisateur sera connecté en tant que client (Cf. : Cas d'utilisation client, « S'authentifier »), dans le cas où « isCooker » vaut TRUE il s'agit d'une cantinière, elle est alors dirigée vers la page « Récap des commandes ».


Dans l'interface, le bouton de connexion est remplacé par les informations nominatives de l'utilisateur.



Dans le cas de données de connexion erronées, aucune donnée d'utilisateur ne sera extraite de la base de données. Un message d'échec de connexion est affiché pour l'utilisateur en l'invitant à réessayer de se connecter.

➤ *Consulter, créditer, débiter un compte utilisateur :*

Recap des Commandes	Plats de la Semaine	Carte	Compte Utilisateur	Configuration	Bonjour Françoise
---------------------	---------------------	-------	--------------------	---------------	-------------------

Rechercher un compte Utilisateur :

Votre recherche :  

▼ Prénom	▼ Nom	▼ Email	▼
Kevin	Lorem	kevin.lorem@test.com	
Lorem	Bah	lorem.bah@test.com	

Recap des Commandes	Plats de la Semaine	Carte	Compte Utilisateur	Configuration	Bonjour Françoise
---------------------	---------------------	-------	--------------------	---------------	-------------------

Prénom : Kevin

Nom : Lorem

Adresse mail : kevin.lorem@test.com

Commande en cours :

Récapitulatif de la commande

Cagnotte : X €

Créditer

Soldier le compte

	<b>Document</b> : <i>Spécifications d'Architecture Logicielle</i> <b>Projet</b> : <i>Cantine Aston</i>	<b>Réf.</b> : <i>SAL-06062016</i> <b>Version</b> : <i>V3.0</i> <b>Date</b> :
	Référentiel documentaire projet SQLI	Page 12 sur 49

Lors de la consultation du compte d'un utilisateur la cuisinière peut créditer ou solder la cagnotte de cet utilisateur.

Au chargement de la page « Compte utilisateur », les informations de l'utilisateur souhaité sont affichées après appel à la méthode « UserWS.getAccount (Integer userId) », qui retourne l'utilisateur récupéré par la méthode « IUserService.getByld (Integer userId) » par l'appel à la méthode « IUserDAO.selectByld (Integer userId) ».

✓ *Consultation du compte d'un utilisateur :*

La cantinière, connectée suivant la procédure expliquée ci-dessus, sur la barre de navigation elle clique sur le lien « Compte Utilisateur »; l'écran « Compte Utilisateur » s'affiche pour la cantinière.

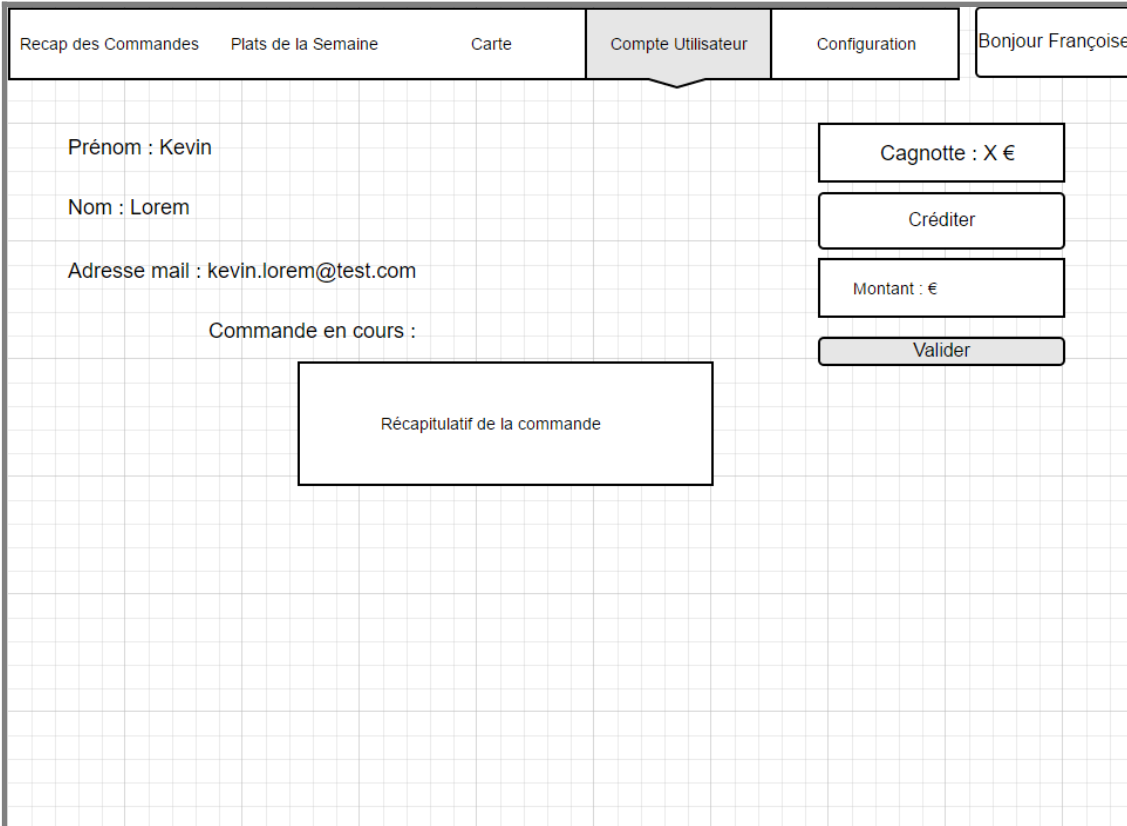
La cantinière peut alors saisir un texte dans le champ de recherche et lancer la recherche

Une requête HTTP GET contenant les données saisies est envoyée au web service « searchUser », celui-ci invoque la méthode « UserWS.search (String value) » qui fait appel au service métier « IUserService.search (String value) », qui à son tour invoque la méthode « IUserDAO.searchAllByValue (String value) ».

Cette méthode envoie une requête à la base de données qui lui retourne la liste de tous les utilisateurs contenant le texte d'entrée dans les champs nom, prénom ou mail.

La cantinière peut alors choisir l'utilisateur souhaité en cliquant sur l'image du lien de consultation de compte. Les informations relatives à cet utilisateur sont affichés sur l'écran « Compte Utilisateur ».

✓ *Créditer le compte d'un utilisateur :*



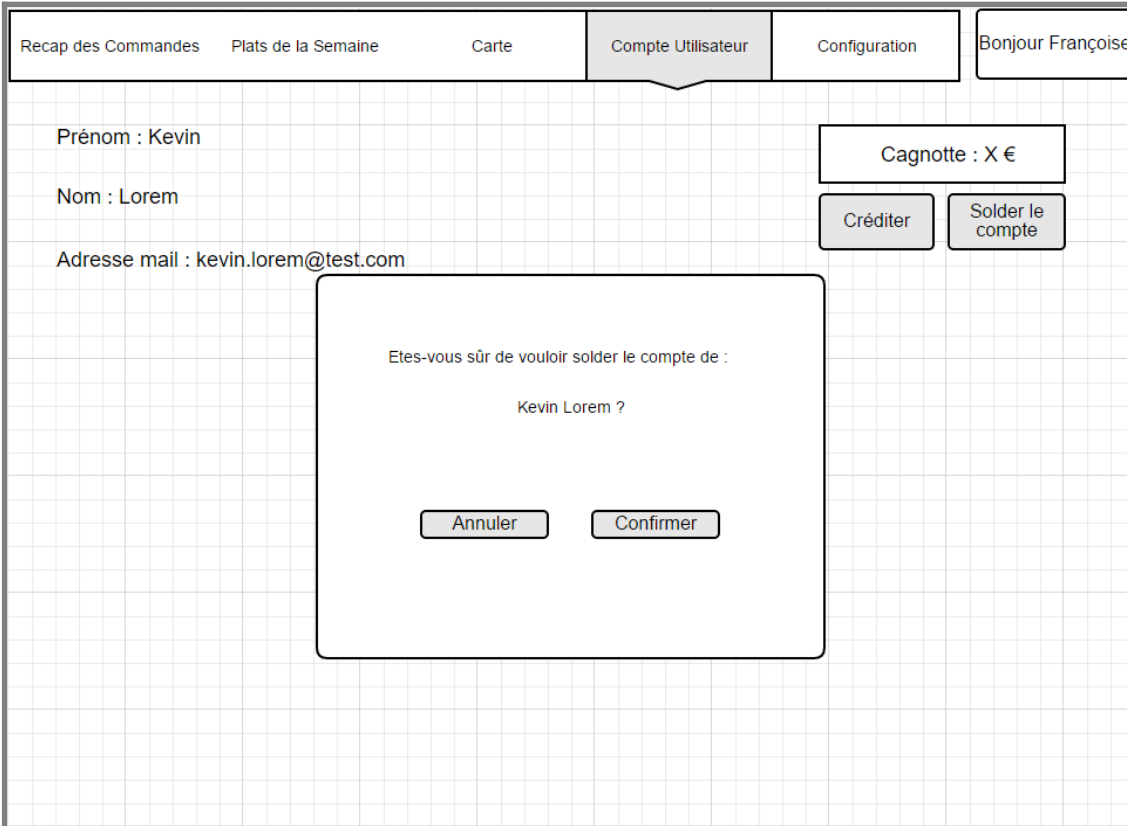
The screenshot shows a web application interface for a canteen. The top navigation bar includes links for 'Recap des Commandes', 'Plats de la Semaine', 'Carte', 'Compte Utilisateur' (which is the active page), 'Configuration', and a user greeting 'Bonjour Françoise'. The main content area is divided into two columns. The left column displays user information: 'Prénom : Kevin', 'Nom : Lorem', and 'Adresse mail : kevin.lorem@test.com'. Below this, it says 'Commande en cours :'. The right column contains a 'Cagnotte : X €' label, a 'Créditer' button, a 'Montant : €' input field, and a 'Valider' button. A 'Récapitulatif de la commande' box is positioned below the user information on the left.

Depuis l'écran de détail d'un compte utilisateur, la cantinière clique sur le bouton « Créditer », un formulaire de saisie s'affiche.

La cantinière saisit le montant souhaité et valide sa saisie à l'aide du bouton « Valider ».

Une requête HTTP PATCH contenant le montant saisi est envoyée au web service « UserWS.creditJackpot (Integer userId, Double amount) », celui-ci invoque la méthode « IUserService.creditJackpot (Integer userId, Double amount) », qui à son tour invoque la méthode « IUserDAO.updateJackpot (Integer userId, Double amount) ». Cette méthode envoie à la base de données une requête qui met à jour le champ « jackpot » de la table User.

✓ *Débiter le compte d'un utilisateur*



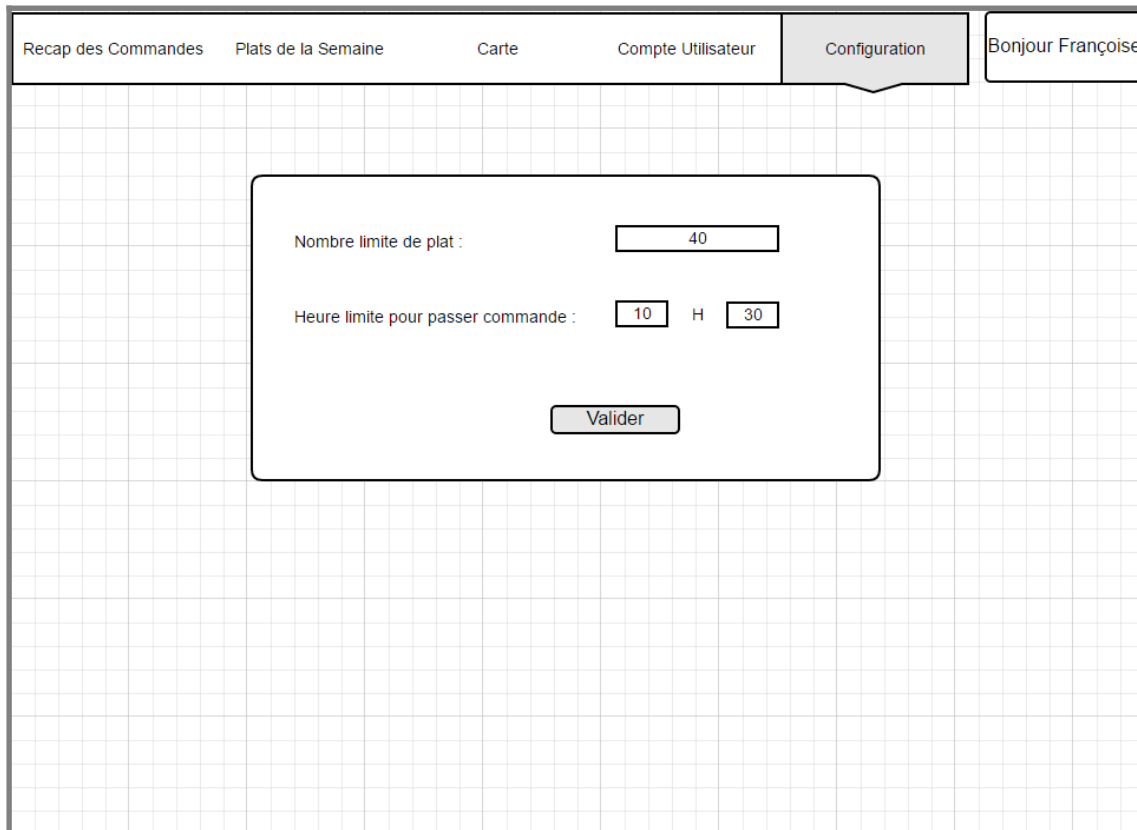
The screenshot shows a web application interface with a navigation bar at the top containing links: 'Recap des Commandes', 'Plats de la Semaine', 'Carte', 'Compte Utilisateur' (active), 'Configuration', and a user greeting 'Bonjour Françoise'. Below the navigation bar, the user's details are displayed: 'Prénom : Kevin', 'Nom : Lorem', and 'Adresse mail : kevin.lorem@test.com'. To the right of these details, there is a 'Cagnotte : X €' field and two buttons: 'Créditer' and 'Soldier le compte'. A modal dialog box is open in the center of the screen, asking for confirmation: 'Etes-vous sûr de vouloir solder le compte de : Kevin Lorem ?'. The dialog has two buttons: 'Annuler' and 'Confirmer'.

Sur l'écran « Compte Utilisateur » la cantinière clique sur le bouton « Soldier le compte », par suite l'écran suivant lui est présenté.

Pour solder le compte, la cantinière doit cliquer sur le bouton « Confirmer ».

Par suite, une requête HTTP PATCH contenant la valeur de la cagnotte et l'identifiant du compte est envoyée via le web service « UserWS.debitJackpot (Integer userId) » à la méthode « IUserService.debitJackpot (Integer userId) », qui à son tour invoque la méthode « IUserDAO.updateJackpot (Integer userId, Double amount) ». Cette méthode envoie à la base de données une requête qui met à jour le champ jackpot de la table User (initialisation à 0 de ce champ).

➤ *Modifier configuration*



La cantinière peut modifier la configuration des commandes. Elle peut mettre à jour le nombre limite de commandes ainsi que l'heure limite pour passer commande.

Pour modifier la configuration la cantinière clique sur le lien Configuration de la barre de navigation, l'écran de configuration lui est alors présenté :

Les données de la configuration sont récupérées par la méthode « ConfigurationWS.fetch () » qui appelle la méthode « IConfigurationService.getConfiguration () » qui invoque la méthode « IConfigurationDAO.select (Integer configurationId) ».

La modification de la configuration suit la procédure suivante :

- La cantinière saisit dans les champs de saisie appropriés les valeurs souhaitées pour le nombre limite de plats et l'heure limite pour passer commande, puis elle appuie sur le bouton « Valider ».
- Une requête HTTP PATCH contenant les valeurs saisies est envoyée au web service « ConfigurationWS.update (LocalTime limitHour, Integer limitQuantity) » qui invoque la méthode « IConfigurationService.update (LocalTime limitHour, Integer limitQuantity) », qui à son tour invoque la méthode « IConfigurationDAO.update (LocalTime limitHour, Integer limitQuantity) ». Elle envoie en base une requête pour mettre à jour la table « Configuration ».



➤ *Traiter commande*

Recap des Commandes
Plats de la Semaine
Carte
Compte Utilisateur
Configuration

Bonjour Françoise

Lundi
Mardi
Mercredi
Jeudi
Vendredi

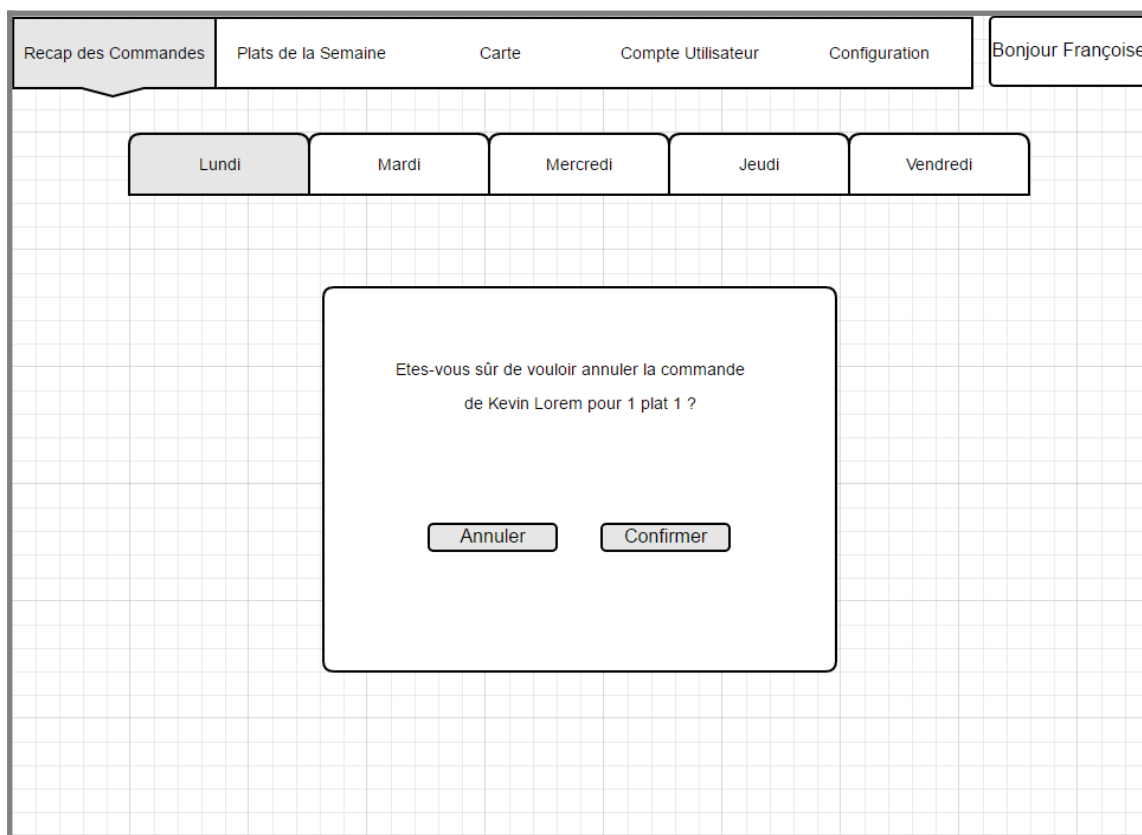
▼ Client	▼ Commande	▼ Emporté	▼ Annuler
Kevin Lorem	Plat 1	<input type="checkbox"/>	
Stéphanie Brierre	Plat 2	<input checked="" type="checkbox"/>	
Lorem Bah	Plat 1	<input checked="" type="checkbox"/>	

Quantités du jour :
Plat 1 - 15
Plat 2 - 5

Le traitement d'une commande par la cantinière connectée suit la procédure suivante :

- La cantinière clique sur le lien « Récap des Commandes » de la barre de navigation. L'écran suivant lui est alors présenté qui montre la liste des commandes passées par les clients
- La cantinière clique sur l'icône de la colonne « Emporté » pour la commande en attente de traitement
- Cette action déclenche un événement JavaScript qui va appeler le web service « OrderWS.lift (Integer orderId) ». Celui-ci invoque la méthode du service métier « IOrderService.lift (Integer orderId) » pour mettre à true la valeur du champ « delivered » via la méthode « IOrderDAO.updateOrderStatus (Integer orderId, Boolean delivered, Boolean cancel) ».

➤ *Annulation commande*

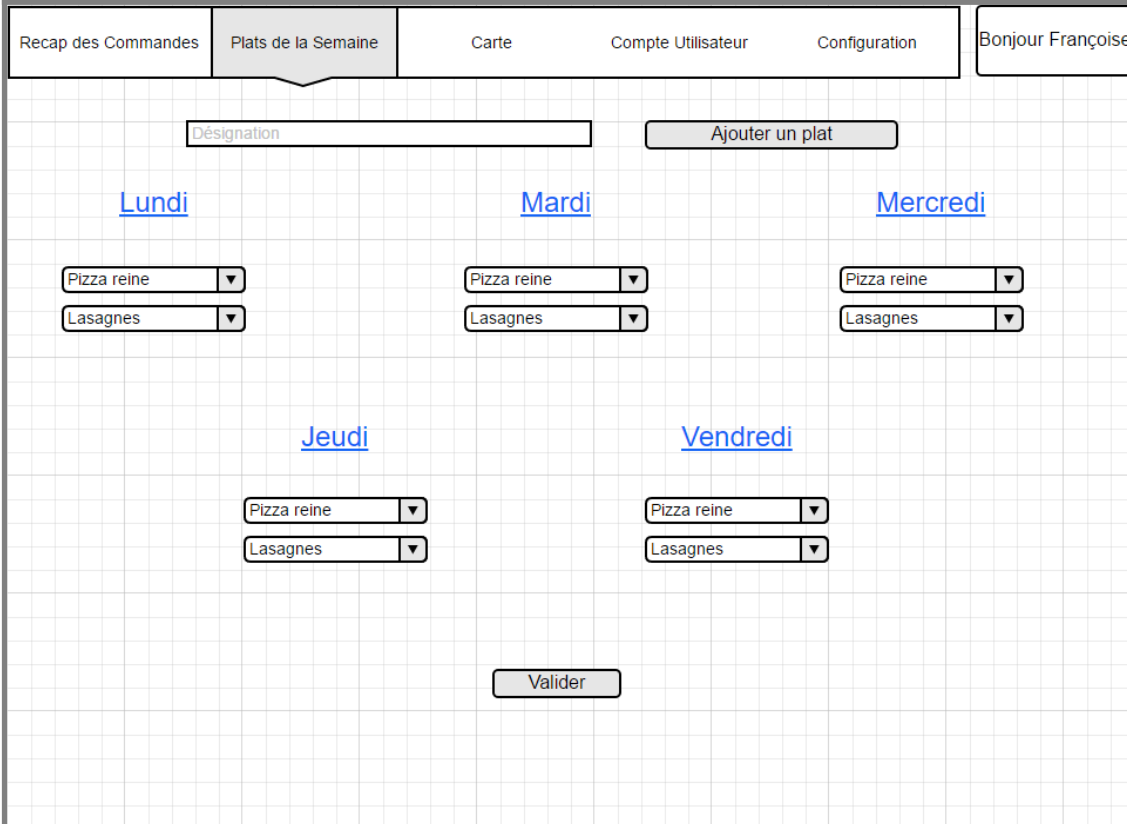


The screenshot shows a web application interface. At the top, there is a navigation bar with links: 'Recap des Commandes', 'Plats de la Semaine', 'Carte', 'Compte Utilisateur', and 'Configuration'. A user greeting 'Bonjour Françoise' is displayed on the right. Below the navigation bar, there are five buttons representing days of the week: 'Lundi', 'Mardi', 'Mercredi', 'Jeudi', and 'Vendredi'. In the center of the page, a modal dialog box is open with the text: 'Etes-vous sûr de vouloir annuler la commande de Kevin Lorem pour 1 plat 1 ?'. At the bottom of the dialog, there are two buttons: 'Annuler' and 'Confirmer'.

L'annulation d'une commande par la cantinière connectée suit la procédure suivante :

- Sur la barre de navigation la cantinière clique sur le lien « Récap des commandes », l'écran de récapitulation des commandes lui est alors présenté.
- La cantinière clique sur l'icône de suppression de commande de la colonne « Annulé » pour la commande à supprimer
- Après validation par la cantinière de l'annulation de la commande par clic sur l'icône, un appel au web service « OrderWS.cancel (Integer orderId) » invoque la méthode « IOrderService.cancel (Integer orderId) » du service métier, qui à son tour invoque la méthode « IOrderDAO.updateOrderStatus (Integer orderId, Boolean delivered, Boolean cancel) » avec le paramètre « delivered » initialisé à false et le paramètre « canceled » à true. La procédure liée à cette action annule la commande et crédite le compte de l'utilisateur concerné.
- La commande qui vient d'être annulée disparaît de l'affichage des commandes en cours.

➤ *Mettre à jour plats de la semaine*



La mise à jour des plats de la semaine par la cantinière suit la procédure suivante :











- Sur la barre de navigation la cantinière connectée clique sur le lien Plats de la semaine, l'écran des plats de la semaine lui est alors présenté :
- La cantinière sélectionne les plats de la semaine via des listes déroulantes pour chaque jour. Si un plat est inexistant, elle saisit le nom de celui-ci dans le champs prévu puis clique sur le bouton « Ajouter un plat », cette action déclenche une requête HTTP PUT vers la méthode « ArticleWS.createMeal (String designation, Double price) » puis vers la méthode « IArticleService.create (IArticleEntity article) » qui fait appel à la méthode « IArticleDAO.insert (IArticleEntity article) ». Ce qui a pour effet de créer un nouveau plat dans la base de données (le plat est inséré avec « IArticleEntity.categoryId » = identifiant des plats).
- Après avoir sélectionné ses plats du jour un clic sur le bouton « Valider » déclenche un appel à la méthode « WeeklyMealWS.update (String meals) » qui invoque la méthode « IWeeklyMealService.update (String meals) », qui enfin invoque la méthode « IWeeklyMealDAO.update (String meals) » pour mettre à jour la table WeeklyMeal.

➤ *Lister articles*

Recap des Commandes
Plats de la Semaine
Carte
Compte Utilisateur
Configuration

Bonjour Françoise

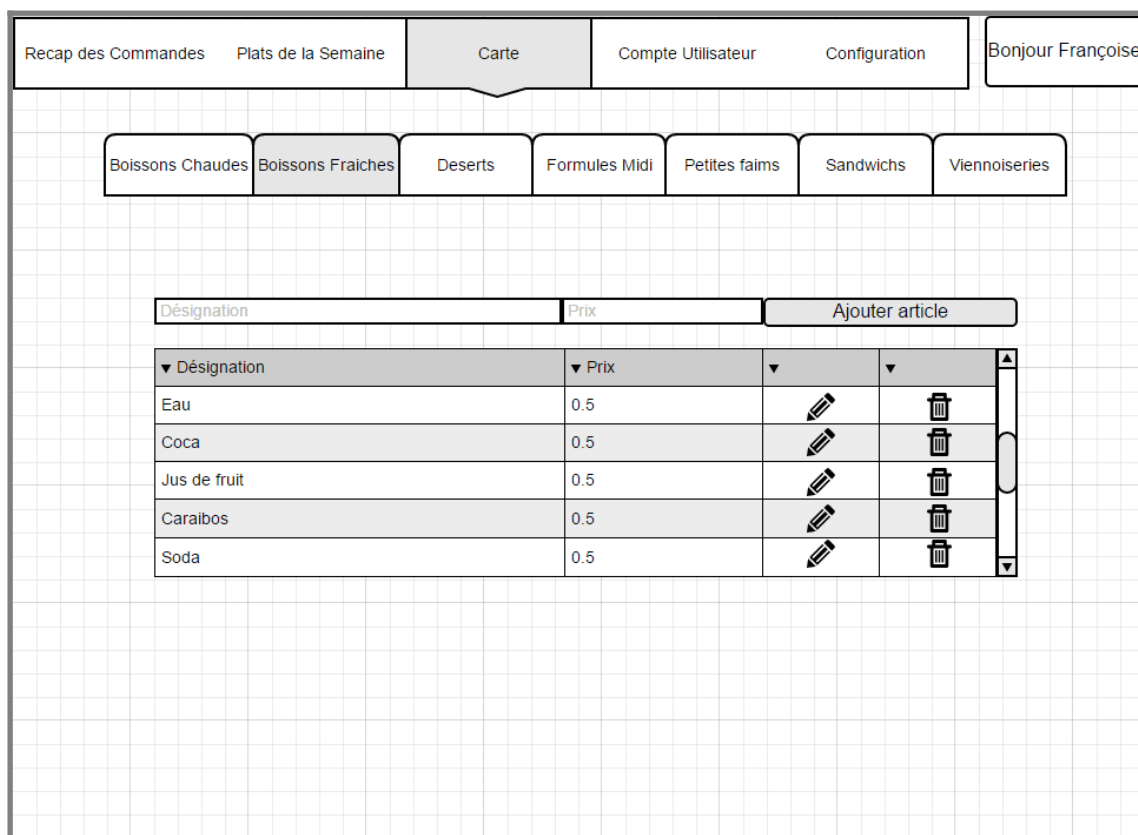
Boissons Chaudes
Boissons Fraiches
Deserts
Formules Midi
Petites faims
Sandwichs
Viennoiseries

Désignation	Prix	Ajouter article	
▼ Désignation	▼ Prix	▼	▼
Eau	0.5		
Coca	0.5		
Jus de fruit	0.5		
Caraibos	0.5		
Soda	0.5		

La visualisation des articles par la cantinière connectée est mise en œuvre selon le schéma suivant :

- Au chargement de la page « Carte » l'écran « Carte » ci-dessous est présenté à la cantinière.
- En cliquant sur chaque lien de la barre de menus, la liste des articles de la catégorie correspondante s'affiche en suivant le cheminement ci-après. Un événement JavaScript est déclenché qui va appeler la méthode « ArticleWS.getArticles () » qui fait appel à son tour à la méthode « IArticleService.getAll () » qui invoque la méthode « IArticleDAO.selectAll () ». Un tri des données est effectué afin d'afficher les articles des catégories souhaités en excluant plats et formules.

➤ *Ajouter article*













Recap des Commandes	Plats de la Semaine	<b>Carte</b>	Compte Utilisateur	Configuration	Bonjour Françoise
---------------------	---------------------	--------------	--------------------	---------------	-------------------

Boissons Chaudes	<b>Boissons Fraîches</b>	Deserts	Formules Midi	Petites faims	Sandwichs	Viennoiseries
------------------	--------------------------	---------	---------------	---------------	-----------	---------------

Désignation	Prix	Ajouter article	
▼ Désignation	▼ Prix	▼	▼
Eau	0.5		
Coca	0.5		
Jus de fruit	0.5		
Carabos	0.5		
Soda	0.5		

L'ajout d'un nouvel article à la carte est opéré à partir de la page « Carte » de la manière suivante :









- La cantinière saisit la désignation et le prix dans les champs prévus à cet effet, puis clique sur le bouton « Ajouter article ». Cela déclenche un événement JavaScript qui va appeler le web service « ArticleWS.createArticle (String designation, Double price) ». Celui-ci crée une nouvelle instance d'une entité « Article » et invoque la méthode « IArticleService.create (IArticleEntity article) », qui à son tour invoque la méthode « IArticleDAO.insert (IArticle article) » pour insérer le nouvel article en base.
- Finalement l'écran de la carte est mis à jour en affichant le nouvel article dans la liste des articles disponibles.

➤ *Modifier article*

Recap des Commandes
Plats de la Semaine
Carte
Compte Utilisateur
Configuration

Bonjour Françoise

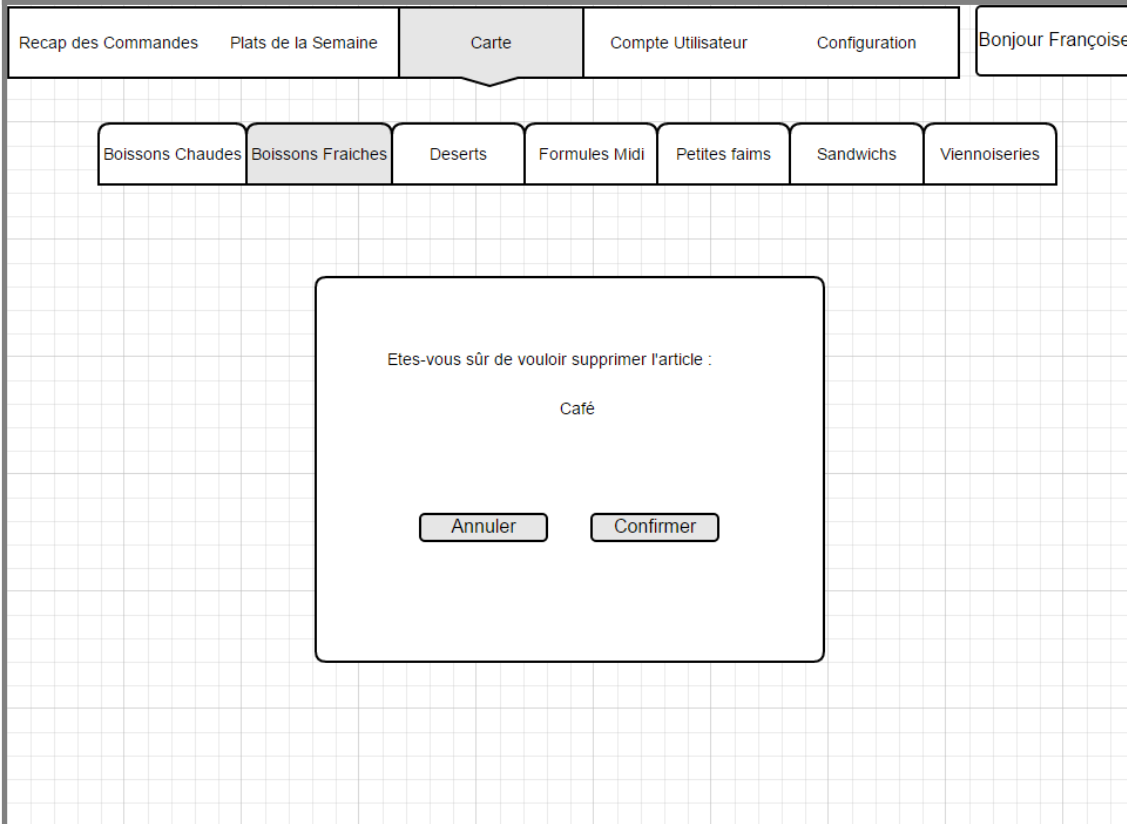
Boissons Chaudes
Boissons Fraîches
Deserts
Formules Midi
Petites faims
Sandwichs
Viennoiseries

Désignation	Prix	Ajouter article	
▼ Désignation	▼ Prix	▼	▼
Eau I	0.5I	✓	
Coca	0.5		
Jus de fruit	0.5		
Caraibos	0.5		
Soda	0.5		

La modification de la désignation et/ou le prix d'un article de la carte est faite sur la page « Carte » de la manière suivante :

- La cantinière clique sur l'icône du crayon située sur la ligne de l'article souhaité. Alors l'affichage se modifie, les zones modifiables sont débloquées, la cantinière modifie la désignation et/ou le prix et clique sur l'icône « coche verte ».
- Un événement JavaScript est déclenché qui va appeler la méthode « ArticleWS.update (Integer id, String designation, Double price) ». Cette méthode invoque la méthode « IArticleService.update (IArticleEntity article) », qui à son tour fait appel à la méthode « IArticleDAO.update (IArticleEntity article) » pour mettre à jour en base l'article considéré dans la table « Article ».
- Finalement l'écran de la carte est mis à jour avec les nouvelles valeurs des champs de l'article.

➤ *Supprimer article*

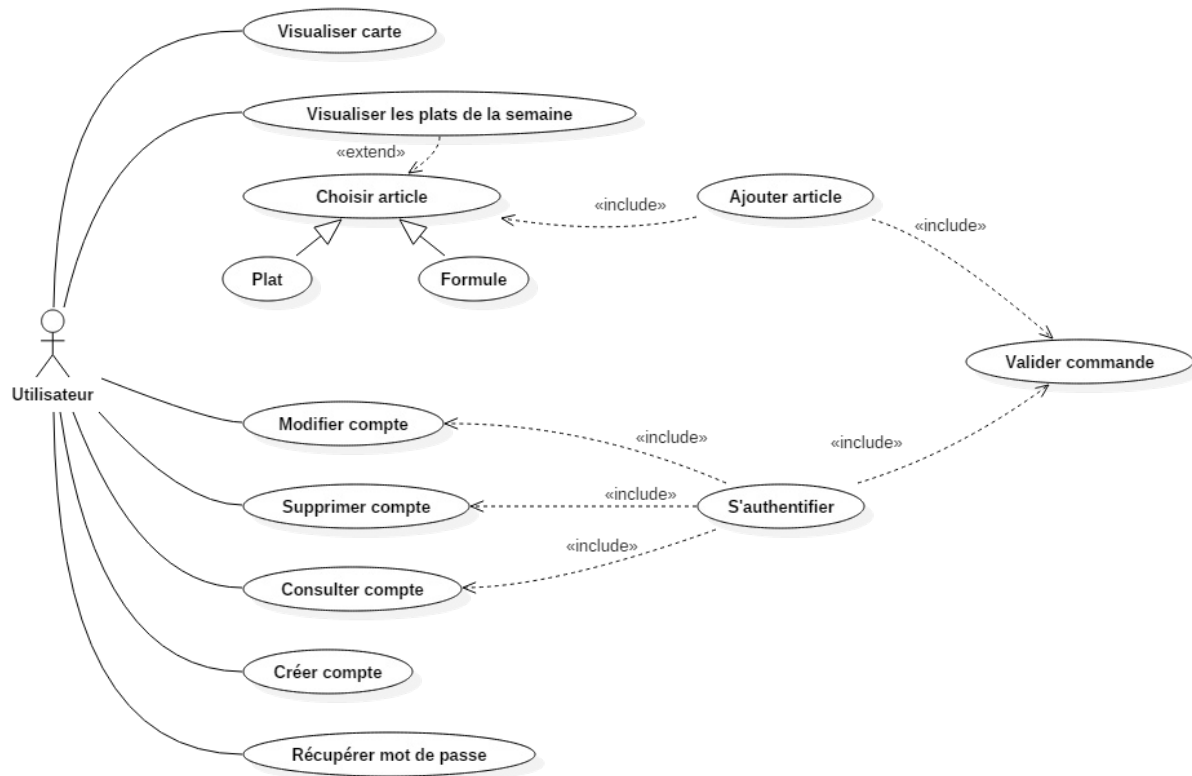


The screenshot shows a web application interface. At the top, there is a navigation bar with links: 'Recap des Commandes', 'Plats de la Semaine', 'Carte' (highlighted), 'Compte Utilisateur', and 'Configuration'. To the right of these links is a user greeting: 'Bonjour Françoise'. Below the navigation bar is a horizontal menu with categories: 'Boissons Chaudes', 'Boissons Fraîches' (highlighted), 'Deserts', 'Formules Midi', 'Petites faims', 'Sandwichs', and 'Viennoiseries'. The main content area is a large grid. In the center of the grid, there is a modal dialog box with the text: 'Etes-vous sûr de vouloir supprimer l'article :', followed by 'Café'. At the bottom of the dialog are two buttons: 'Annuler' and 'Confirmer'.

La suppression d'un article de la carte est faite sur la page « Carte » de la manière suivante :

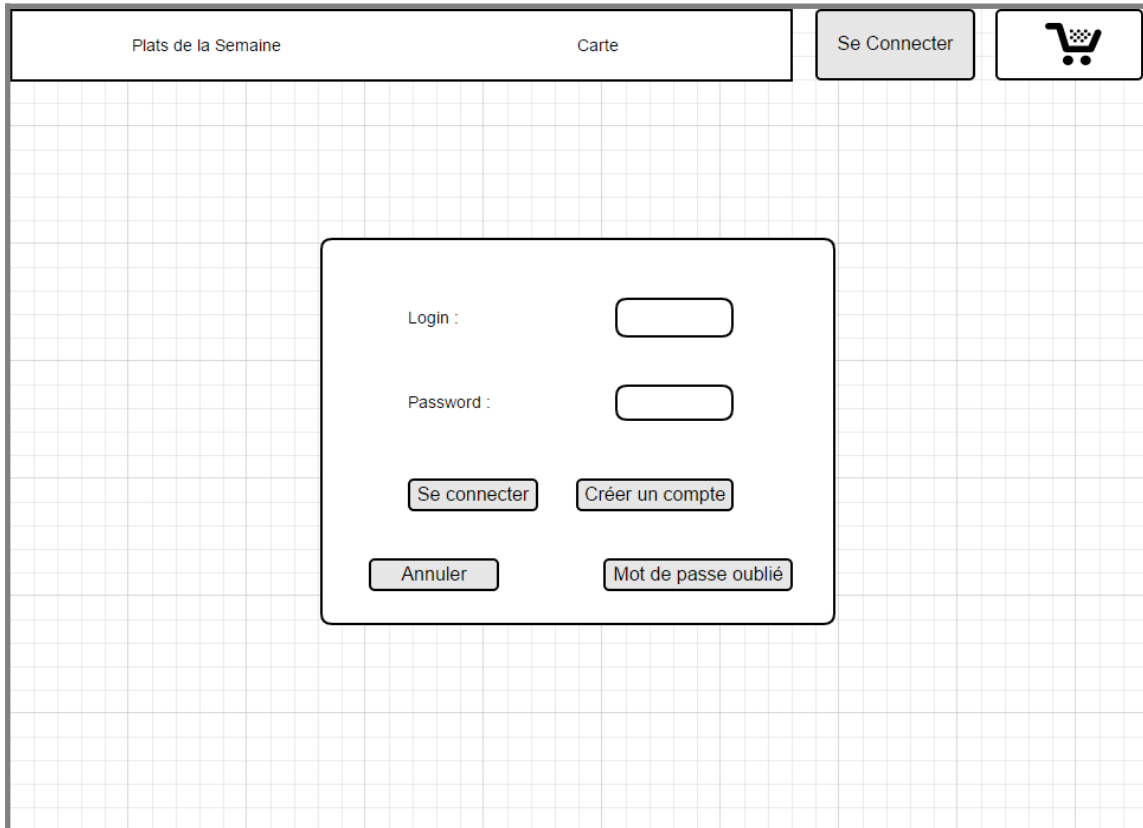
- La cantinière clique sur l'icône de la poubelle située sur la ligne de l'article qu'elle souhaite supprimer de la carte.
- Après validation par la cantinière de la suppression de l'article, un événement JavaScript est déclenché qui va appeler la méthode « ArticleWS.delete (Integer articleId) ». Cette méthode invoque la méthode « IArticleService.delete (Integer articleId) », qui à son tour fait appel à la méthode « IArticleDAO.delete (Integer articleId) » pour supprimer en base l'article considéré de la table « Article ».
- Finalement l'écran de la carte est mis à jour en faisant disparaître de la liste des articles l'article qui vient d'être supprimé.

## C. Cas d'utilisation de l'utilisateur





➤ *S'authentifier*



L'utilisateur non connecté, depuis n'importe quelle page accessible clique sur le bouton « Se Connecter », une fenêtre modale s'affiche.

Après avoir renseigné ses identifiants, l'utilisateur clique sur le bouton 'se connecter', un appel à la méthode `UserWS.authenticate (String email, String password)` est envoyée, elle invoque à son tour la méthode `IUserService.authenticate (String email, String password)` qui fait appel à la méthode `IUserDAO.selectByEmailPassword (String email, String password)` qui retourne l'utilisateur correspondant aux critères si celui-ci existe.

Dans l'interface, le bouton de connexion est remplacé par les informations nominatives de l'utilisateur.

Dans le cas de données de connexion erronées, aucune donnée d'utilisateur ne sera extraite de la base de données. Un message d'échec de connexion est affiché pour l'utilisateur en l'invitant à réessayer de se connecter.

➤ Visualiser carte

Plats de la Semaine	Carte	Se Connecter
<div> <div> <h3>Boissons Chaudes</h3> <p>Café..... 1 €</p> <p>Chocolat..... 1 €</p> </div> <div> <h3>Boissons Fraiches</h3> <p>Coca..... 1 €</p> <p>Orangina ..... 1 €</p> </div> <div> <h3>Viennoiseries</h3> <p>Pain au Chocolat..... 1 €</p> <p>Croissant..... 1 €</p> </div> <div> <h3>Les Desserts</h3> </div> <div> <h3>Sandwiches</h3> </div> <div> <h3>Les Petites Faims</h3> </div> <div> <h3>Les Formules Midi</h3> </div> </div>		

Au chargement de la page « Carte », la méthode « ArticleWS.getArticles () » récupère l'intégralité des articles par la méthode « IArticleService.selectAll () » qui fait appel à la méthode « IArticleDAO.selectAll () ».

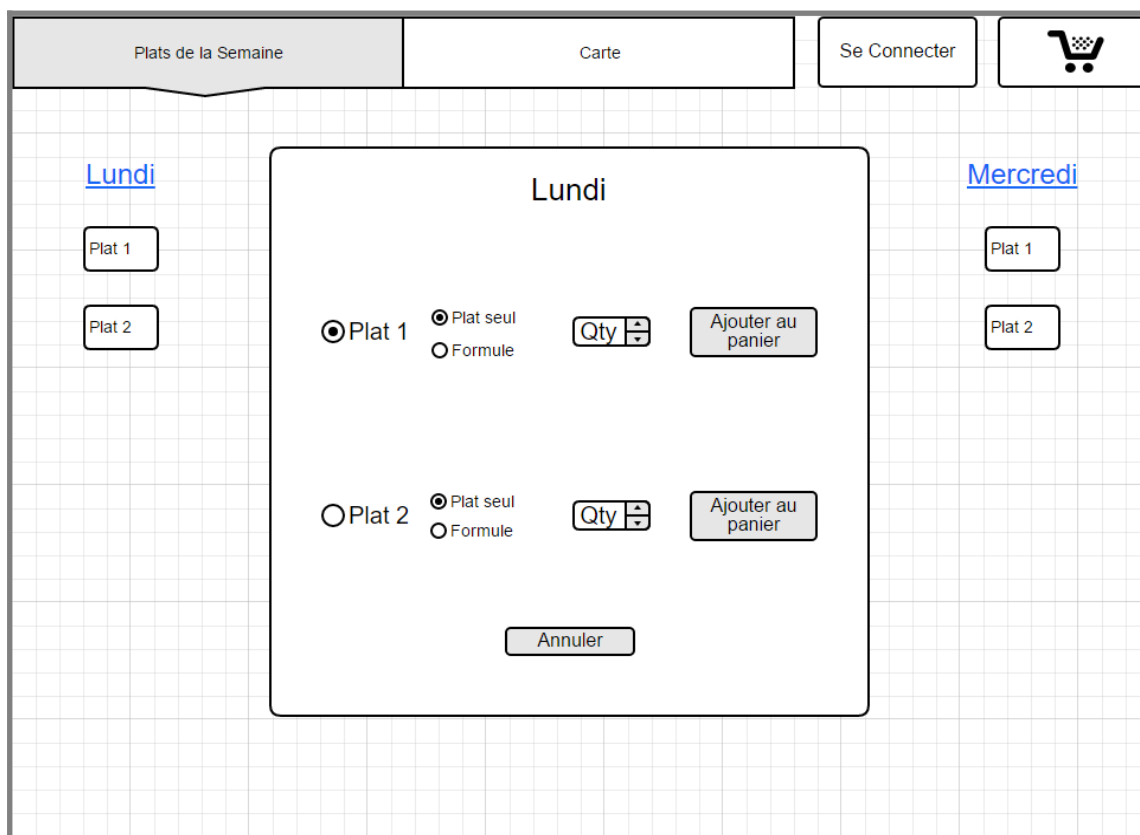
Un tri est effectué à l'affichage pour n'afficher que les articles des catégories souhaitées.

➤ *Visualiser les plats de la semaine*

Plats de la Semaine		Carte	Se Connecter						
<table border="0"> <tr> <td data-bbox="226 396 703 672"> <p><a href="#">Lundi</a></p> <div data-bbox="300 524 378 568">Plat 1</div> <div data-bbox="300 602 378 647">Plat 2</div> </td> <td data-bbox="708 396 1177 672"> <p><a href="#">Mardi</a></p> <div data-bbox="758 524 836 568">Plat 1</div> <div data-bbox="758 602 836 647">Plat 2</div> </td> <td data-bbox="1182 396 1362 672"> <p><a href="#">Mercredi</a></p> <div data-bbox="1198 524 1276 568">Plat 1</div> <div data-bbox="1198 602 1276 647">Plat 2</div> </td> </tr> <tr> <td data-bbox="226 678 703 1126"> <p><a href="#">Jeudi</a></p> <div data-bbox="517 777 595 822">Plat 1</div> <div data-bbox="517 855 595 900">Plat 2</div> </td> <td data-bbox="708 678 1177 1126"> <p><a href="#">Vendredi</a></p> <div data-bbox="984 777 1062 822">Plat 1</div> <div data-bbox="984 855 1062 900">Plat 2</div> </td> <td data-bbox="1182 678 1362 1126"></td> </tr> </table>				<p><a href="#">Lundi</a></p> <div data-bbox="300 524 378 568">Plat 1</div> <div data-bbox="300 602 378 647">Plat 2</div>	<p><a href="#">Mardi</a></p> <div data-bbox="758 524 836 568">Plat 1</div> <div data-bbox="758 602 836 647">Plat 2</div>	<p><a href="#">Mercredi</a></p> <div data-bbox="1198 524 1276 568">Plat 1</div> <div data-bbox="1198 602 1276 647">Plat 2</div>	<p><a href="#">Jeudi</a></p> <div data-bbox="517 777 595 822">Plat 1</div> <div data-bbox="517 855 595 900">Plat 2</div>	<p><a href="#">Vendredi</a></p> <div data-bbox="984 777 1062 822">Plat 1</div> <div data-bbox="984 855 1062 900">Plat 2</div>	
<p><a href="#">Lundi</a></p> <div data-bbox="300 524 378 568">Plat 1</div> <div data-bbox="300 602 378 647">Plat 2</div>	<p><a href="#">Mardi</a></p> <div data-bbox="758 524 836 568">Plat 1</div> <div data-bbox="758 602 836 647">Plat 2</div>	<p><a href="#">Mercredi</a></p> <div data-bbox="1198 524 1276 568">Plat 1</div> <div data-bbox="1198 602 1276 647">Plat 2</div>							
<p><a href="#">Jeudi</a></p> <div data-bbox="517 777 595 822">Plat 1</div> <div data-bbox="517 855 595 900">Plat 2</div>	<p><a href="#">Vendredi</a></p> <div data-bbox="984 777 1062 822">Plat 1</div> <div data-bbox="984 855 1062 900">Plat 2</div>								

Au chargement de la page « Plats de la semaine », la méthode « WeeklyMealWS.selectAll () » récupère l'intégralité des plats de la semaine par la méthode « IWeeklyMealService.selectAll () » qui fait appel à la méthode « IWeeklyMealDAO.selectAll () ».

➤ *Ajouter article*




The screenshot shows a web application interface for 'Plats de la Semaine' (Dishes of the Week). At the top, there are navigation tabs: 'Plats de la Semaine' (active), 'Carte', 'Se Connecter', and a shopping cart icon. Below the tabs, there are links for 'Lundi' and 'Mercredi'. The main content area is a modal titled 'Lundi' that appears when a dish is selected. Inside the modal, there are two sections for 'Plat 1' and 'Plat 2'. Each section has radio buttons for 'Plat seul' (selected) and 'Formule', a quantity input field labeled 'Qty', and an 'Ajouter au panier' button. At the bottom of the modal is an 'Annuler' button. On the left and right sides of the modal, there are buttons for 'Plat 1' and 'Plat 2' to select the dish.

Depuis la page de consultation des plats de la semaine, en présence de la fenêtre modale de sélection de plat/ formule en tant que client.

Cette partie de l'application est gérée indépendamment des WS, c'est Angular qui est en charge de la persistance temporaire de ces données.

Au clic sur le bouton « Ajouter au panier », la catégorie (formule, plat), l'identifiant et la quantité d'articles de la ligne correspondante est sauvegardée afin d'être affichée plus tard sur l'écran de récapitulation des commande, avant d'être potentiellement validée.

➤ *Valider commande*

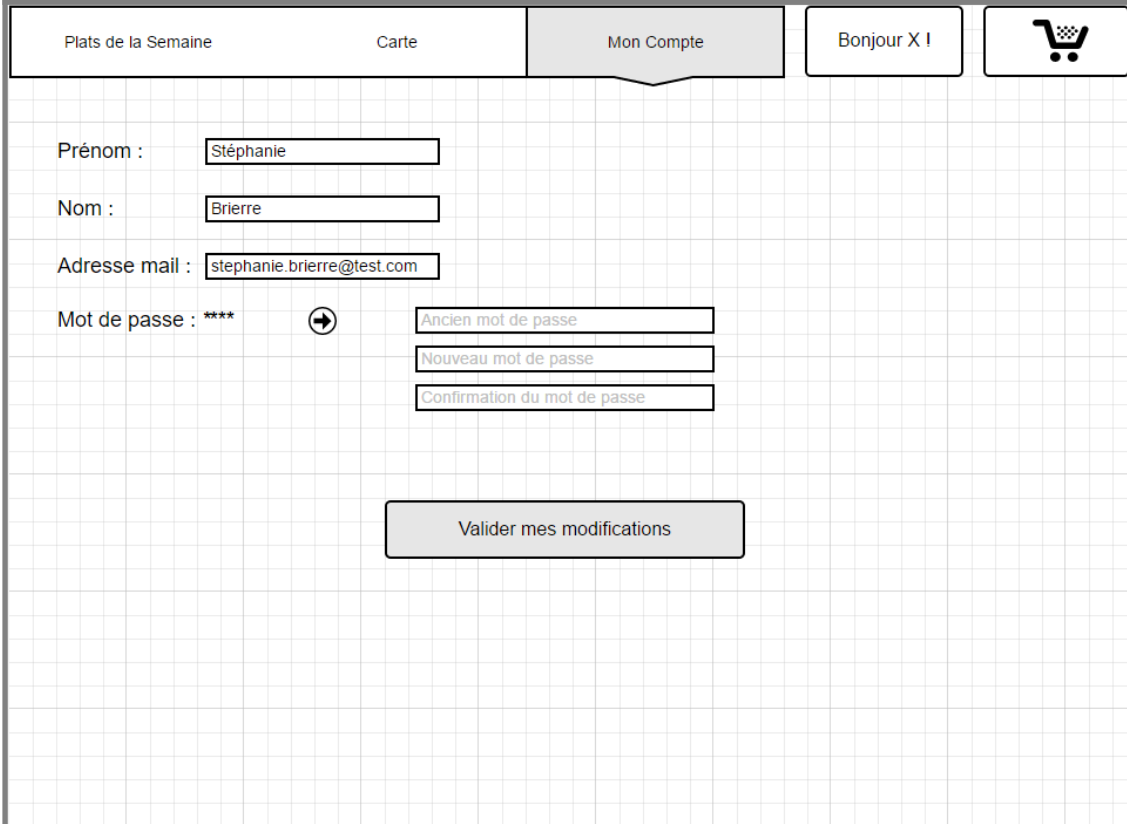
Plats de la Semaine		Carte	Se Connecter	
Pour Lundi				
Plat 1	Qty : 1	Sous total : 5€	<input type="button" value="Supprimer"/>	
Pour Mercredi				
Plat 1	Qty : 1	Sous total : 5€	<input type="button" value="Supprimer"/>	
Pour Vendredi				
Plat 1	Qty : 1	Sous total : 5€	<input type="button" value="Supprimer"/>	
Total : 15 €				

Depuis la page de consultation du panier connecté en tant que client.

Au clic sur le bouton « Valider la commande », la méthode « OrderWS.order (String order) » récupère une chaîne de caractères qui contient une liste d'identifiants d'articles liés à une quantité pour une date de livraison. Une « Map<LocalDate, IOrderHasArticle> » est envoyée à la méthode « IOrderService.create (Map<LocalDate, IOrderHasArticle> order) » qui retourne un objet « IOrderEntity » envoyé par la méthode « IOrderDAO.create (Map<LocalDate, IOrderHasArticle> order) ».

Les articles commandés sont passés en paramètres optionnels de la requête, dans le cas où ces paramètres seraient inexistants, aucune requête ne sera lancée.

➤ *Modifier compte*




Depuis la page de consultation de compte connecté en tant que client.

Au clic sur le bouton « Modifier mes infos », la méthode « `UserWS.update(String email, String lastName, String firstName, String oldPassword, String newPassword)` » récupère l'id de l'utilisateur courant dans le token situé dans le header de la requête afin de reconstruire un objet « `IUserEntity` » pour l'utiliser en tant que paramètre de la méthode « `IUserService.update(IUserEntity user)` » qui retourne un objet « `IUserEntity` » avec les modifications effectuées en base de données par la méthode « `IUserDAO.update(IUserEntity user)` ».

Si cette phase se déroule correctement, la méthode « `IUserService.updatePassword(Integer userId, String oldPassword, String newPassword)` » est lancée, faisant appel à la méthode « `IUserDAO.updatePassword(Integer userId, String oldPassword, String newPassword)` » qui retourne un objet « `IUserEntity` » avec les modifications effectuées.

➤ *Supprimer compte*

Plats de la Semaine	Carte	<b>Mon Compte</b>	Bonjour X !	
---------------------	-------	-------------------	-------------	---

Prénom : Stéphanie	Cagnotte : 20 €
Nom : Brierre	
Adresse mail : stephanie.brierre@test.com	

Commande en cours :

Récapitulatif de la commande

Modifier mes infos


Supprimer mon compte

Depuis la page de consultation de compte connecté en tant que client.

Au clic sur le bouton « Supprimer mon compte », la méthode « UserWS.deleteAccount () » récupère l'id de l'utilisateur courant dans le token situé dans le header de la requête afin de l'utiliser en paramètre de la méthode « IUserService.delete (Integer userId) » qui retourne l'état de la requête faite en base de données par la méthode « IUserDAO.delete (Integer userId) ». Soit TRUE en cas de succès, dans le cas contraire la valeur retournée sera FALSE.

L'utilisateur est déconnecté puis redirigé vers la page d'accueil « Plats de la semaine ».

➤ *Consulter compte*

Plats de la Semaine	Carte	<b>Mon Compte</b>	Bonjour X !	
---------------------	-------	-------------------	-------------	---

Prénom : Stéphanie	Cagnotte : 20 €
Nom : Brierre	
Adresse mail : stephanie.brierre@test.com	

Commande en cours :

Récapitulatif de la commande

Modifier mes infos


Supprimer mon compte

Au chargement de la page « Mon compte », la méthode « `UserWS.getCurrentAccount ()` » récupère l'id de l'utilisateur courant dans le token situé dans le header de la requête afin de l'utiliser en paramètre de la méthode « `IUserService.getByld (Integer userId)` » qui retourne l'utilisateur correspondant en base de données par la méthode « `IUserDAO.selectByld (Integer userId)` ».

Dans le même temps, à partir du même id extrait et de la date du jour, la méthode « `IOrderService.selectAtDate (LocalDate date)` » nous retourne une liste des commandes du jour via la méthode « `IOrderDAO.selectAllAtDate (LocalDate date)` », liste sur laquelle nous devons effectuer un tri en fonction de l'attribut « `IOrderEntity.userId` ».




➤ *Créer compte*

Plats de la Semaine	Carte	Se Connecter	
---------------------	-------	--------------	---

Login :

Password :

Plats de la Semaine	Carte	Se Connecter	
---------------------	-------	--------------	---

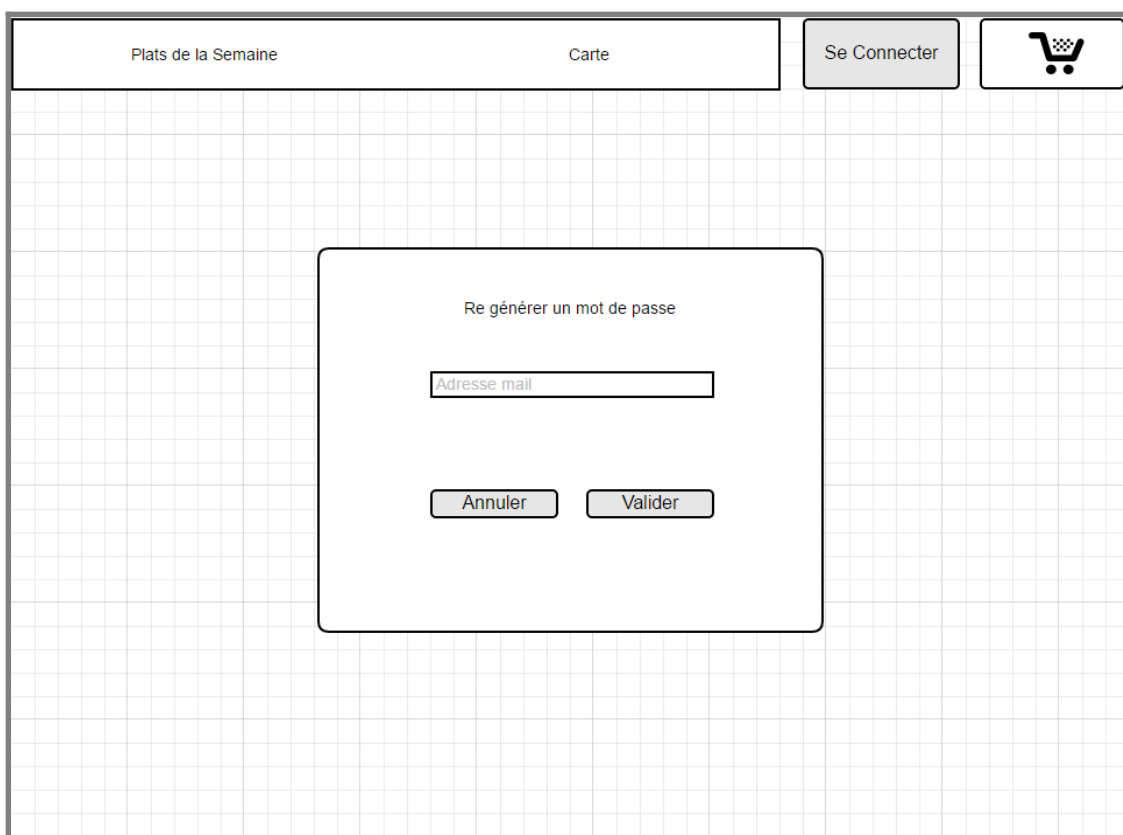
	<b>Document</b> : <i>Spécifications d'Architecture Logicielle</i> <b>Projet</b> : <i>Cantine Aston</i>	<b>Réf.</b> : <i>SAL-06062016</i> <b>Version</b> : <i>V3.0</i> <b>Date</b> :
	Référentiel documentaire projet SQLI	Page 33 sur 49

L'utilisateur non connecté, depuis n'importe quel page accessible, clique sur le bouton « Se connecter », une fenêtre modale s'affiche, clique sur le lien « Créer un compte », la fenêtre modale contient un formulaire de saisie.

Après avoir entré l'email, mot de passe, nom, prénom et cliqué sur le bouton de validation, la méthode « UserWS.create (String email, String password, String lastName, Sting firstName) » construit un objet « IUserEntity » à partir des paramètres, la vérification de la cohérence des données est effectuée par la méthode « IUserService.create (IUserEntity user) » qui retourne l'utilisateur nouvellement inséré dans la base de données par la méthode « IUserDAO.insert (IUserEntity user) ».

En cas de succès de la création de compte l'utilisateur est invité à se connecter.

### ➤ Récupérer mot de passe



L'utilisateur non connecté, depuis n'importe quelle page accessible clique sur le bouton « Se connecter », une fenêtre modale s'affiche.

Clique sur le lien « Récupérer mot de passe », la fenêtre modale contient maintenant un champ de saisie pour l'adresse email.

Après avoir entré l'adresse email et cliqué sur le bouton de validation, la méthode « UserWS.recoverPassword (String email) » est appelée, la génération du mot de passe crypté et l'envoi par mail sont effectués par la méthode service « IUserService.recoverPassword (String email) » après modification dans la base de données du nouveau mot de passe de l'utilisateur « IUserDAO.resetPasswordForEmail (String email, String password) ».

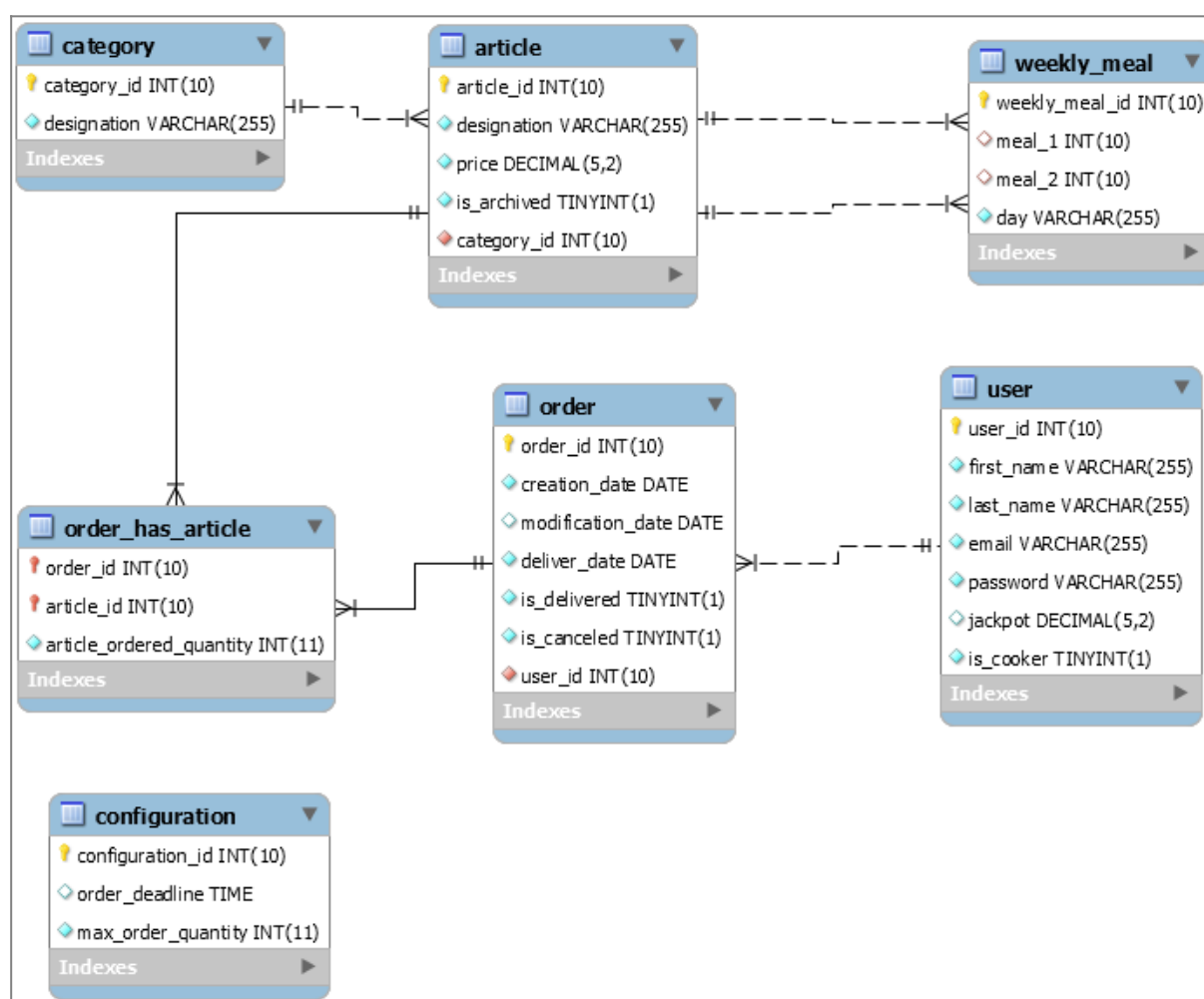
## 4. Vue logique

### A. Base de données

Le format des données que nous avons besoin de faire persister peut être considéré comme un schéma relationnel, dans ce sens nous avons décidé d'utiliser le système de gestion de bases de données relationnelles MySQL afin de stocker les données pour une durée indéterminée.

Ce gestionnaire nous permet d'utiliser toutes les fonctionnalités nécessaires à la mise en œuvre de notre projet (procédures stockés, transactions).

Ce choix a aussi été défini en accord avec le niveau de compétence des membres de l'équipe, un autre SGBDR tel que PostGreSQL ou MariaDB aurait tout aussi bien pu être mis en place.



Libellé	Description	Relation	Clé primaire	Clé étrangère
Category	Catégories auxquelles les articles peuvent appartenir	- Type 1..N avec la table « article »	- Simple « category_id »	Sans
Article	Articles de la carte	- Type 1..1 avec la table « category » - Type 0..N avec la table « order » - Type 0..N avec la table « weekly_meal »	- Simple « article_id »	- « category_id »
Weekly_meal	Plats à la commande pour la semaine courante	- Type 0..1 avec la table « article »	- Simple « weekly_meal_id »	- « meal_1 » - « meal_2 »
Order	Commandes passées par les utilisateurs	- Type 1..1 avec la table « user » - Type 1..N avec la table « article »	- Simple « order_id »	- « user_id »
Order_has_article	Articles contenus dans les commandes passées par les utilisateurs	Table issue de la relation entre les tables « order » et « article »	- Composée de «order_id» et de «article_id»	- «order_id» - «article_id»
User	Données des utilisateurs	- Type 0..N avec la table « order »	- Simple « user_id »	Sans
Configuration	Données concernant la configuration de l'application	Aucune	- Simple « configuration_id »	Sans

## B. Procédures stockées

Notre schéma de bases de données implique l'utilisation de clé étrangères multiples au niveau de la table « order\_has\_article », ou encore « weekly\_meal ». Ce type de relation implique de se conformer à des exigences spécifiques du point de vue d'un ORM. C'est pourquoi nous avons décidé de mettre en place un système de requête par procédures stockées.

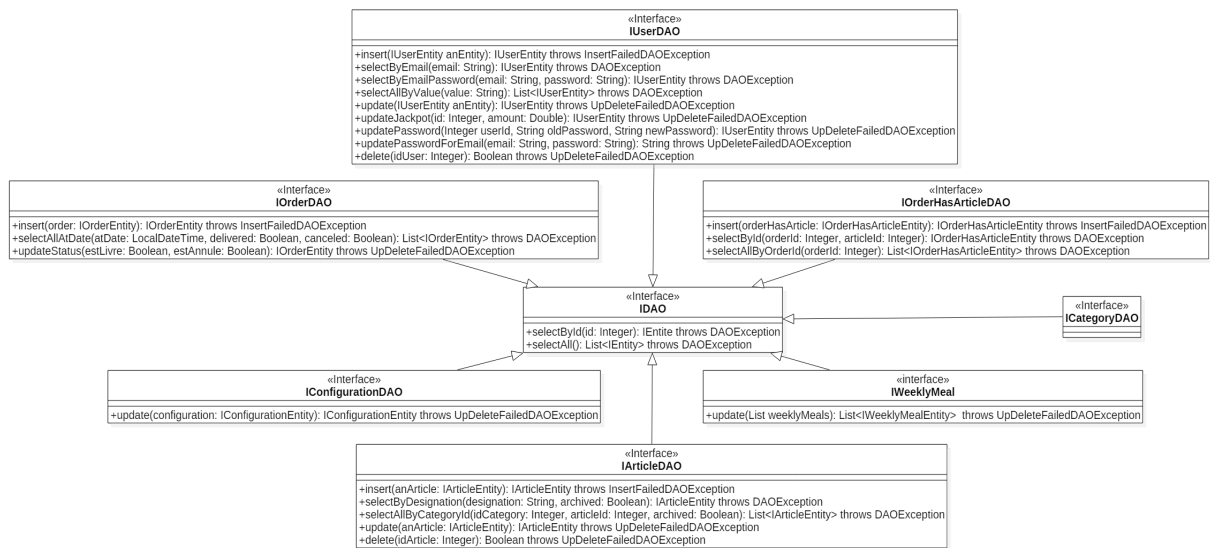
Table concernée	Libellé de la procédure	Description
Weekly_meal	SELECT_ALL_WEEKLY_MEALS ()	Sélectionne tous les plats de la semaine
	UPDATE_WEEKLY_MEAL (IN `meal_1_id` INT(10), IN `meal_2_id` INT(10), IN `id` INT)	Met à jour les plats de la semaine Prend en paramètre une chaîne de caractères contenant une liste d'articles liés aux jours de la semaine
Article	INSERT_ARTICLE (IN `articleDesignation` VARCHAR(255), IN `articlePrice` DECIMAL(5,2), IN `articleCategoryId` INT)	Insère un nouvel article après avoir vérifié qu'aucun article possédant la même désignation et catégorie ne soit existant Retourne l'article nouvellement inséré
	SELECT_ARTICLE_BY_ID (IN `id` INT)	Sélectionne un article par son identifiant
	SELECT_ALL_ARTICLES ()	Sélectionne tous les articles non archivés
	SELECT_ALL_ARTICLES_BY_CATEGORY_ID` (IN `orderBy` VARCHAR(255), IN `sortDir` TINYINT(1), IN `articleCategoryId` INT)	Sélectionne une liste d'articles non archivés correspondant à un identifiant de catégorie reçu en paramètre
	UPDATE_ARTICLE (IN `articleDesignation` VARCHAR(255), IN `articlePrice` DECIMAL(5,2), IN `articleCategoryId` INT, IN `id` INT)	Met à jour les données d'un article après avoir vérifié qu'aucun article possédant la même désignation et catégorie ne soit existant. Retourne l'article modifié
	ARCHIVE_ARTICLE (IN `id` INT)	Définit un article comme étant archivé. Retourne l'entrée modifiée
	DELETE_ARTICLE (IN `id` INT)	Supprime un article. Retourne le nombre de lignes affecté
Category	SELECT_ALL_CATEGORIES ()	Sélectionne toutes les catégories

Table concernée	Libellé de la procédure	Description
Order_has_article	INSERT_ORDER_HAS_ARTICLE (IN `orderId` INT, IN `articleId` INT, IN `orderQuantity` INT)	Insère un lien entre une commande et un article pour une quantité donnée
	SELECT_ORDER_HAS_ARTICLE_BY_ID (IN `orderId` INT, IN `articleId` INT)	Sélectionne un article appartenant à une commande
	SELECT_ALL_ORDER_HAS_ARTICLES_BY_ORDER_ID (IN `orderId` INT)	Sélectionne une liste d'articles contenus dans une commande
Order	INSERT_ORDER (IN `deliverDateOrder` DATE, IN `userIdOrder` INT)	Insère une nouvelle commande. Retourne la commande créée
	SELECT_ORDER_BY_ID (IN `id` INT)	Sélectionne une commande par son identifiant
	SELECT_ALL_ORDERS_BY_DELIVER_DATE (IN `atDateOrder` DATE)	Sélectionne une liste de commande non livrées, non annulées à la date visée
	UPDATE_ORDER_STATUS (IN `isDeliveredOrder` TINYINT(1), IN `isCanceledOrder` TINYINT(1), IN `id` INT)	Modifie l'état d'une commande (livrée ou annulée). Retourne la commande visée
Configuration	SELECT_CONFIGURATION_BY_ID (IN `id` INT)	Sélectionne une configuration à partir d'un identifiant
	UPDATE_CONFIGURATION (IN `orderDeadlineConfiguration` TIME, IN `maxOrderQuantityConfiguration` INT, IN `id` INT)	Met à jour les données de la configuration. Retourne la configuration modifiée
User	INSERT_USER (IN `userFirstname` VARCHAR(255), IN `userLastname` VARCHAR(255), IN `userEmail` VARCHAR(255), IN `userPassword` VARCHAR(255))	Insère un nouvel utilisateur. Retourne l'utilisateur nouvellement créé
	SELECT_USER_BY_ID (IN `id` INT)	Sélectionne un utilisateur à partir de son identifiant

Table concernée	Libellé de la procédure	Description
	SELECT_USER_BY_EMAIL (IN `userEmail` VARCHAR(255))	Sélectionne un utilisateur à partir de son email
	SELECT_USER_BY_EMAIL_PASSWORD (IN `userEmail` VARCHAR(255), IN `userPassword` VARCHAR(255))	Sélectionne un utilisateur à partir de son email et de son mot de passe
	SEARCH_ALL_USERS_BY_VALUE (IN `searchValue` VARCHAR(255))	Sélectionne une liste d'utilisateurs à partir d'une valeur de recherche sur les champs «first_name», «last_name», «email»
	RESET_USER_PASSWORD_FOR_EMAIL (IN `userEmail` VARCHAR(255), IN `userPassword` VARCHAR(255))	Met à jour le mot de passe d'un utilisateur
	UPDATE_USER (IN `userFirstname` VARCHAR(255), IN `userLastname` VARCHAR(255), IN `userEmail` VARCHAR(255), IN `id` INT)	Met à jour les données d'un utilisateur. Retourne l'utilisateur mis à jour
	UPDATE_USER_JACKPOT (IN `userJackpot` DECIMAL(5,2), IN `id` INT)	Met à jour la cagnotte d'un utilisateur. Retourne l'utilisateur visé
	UPDATE_USER_PASSWORD (IN `oldUserPassword` VARCHAR(255), IN `newUserPassword` VARCHAR(255), IN `id` INT)	Met à jour le mot de passe d'un utilisateur. Retourne l'utilisateur visé
	DELETE_USER (IN `id` INT)	Supprime un utilisateur après avoir vérifié que ce dernier n'a aucune commande en cours et que la valeur de sa cagnotte est à 0.00. Retourne le nombre de lignes affectées

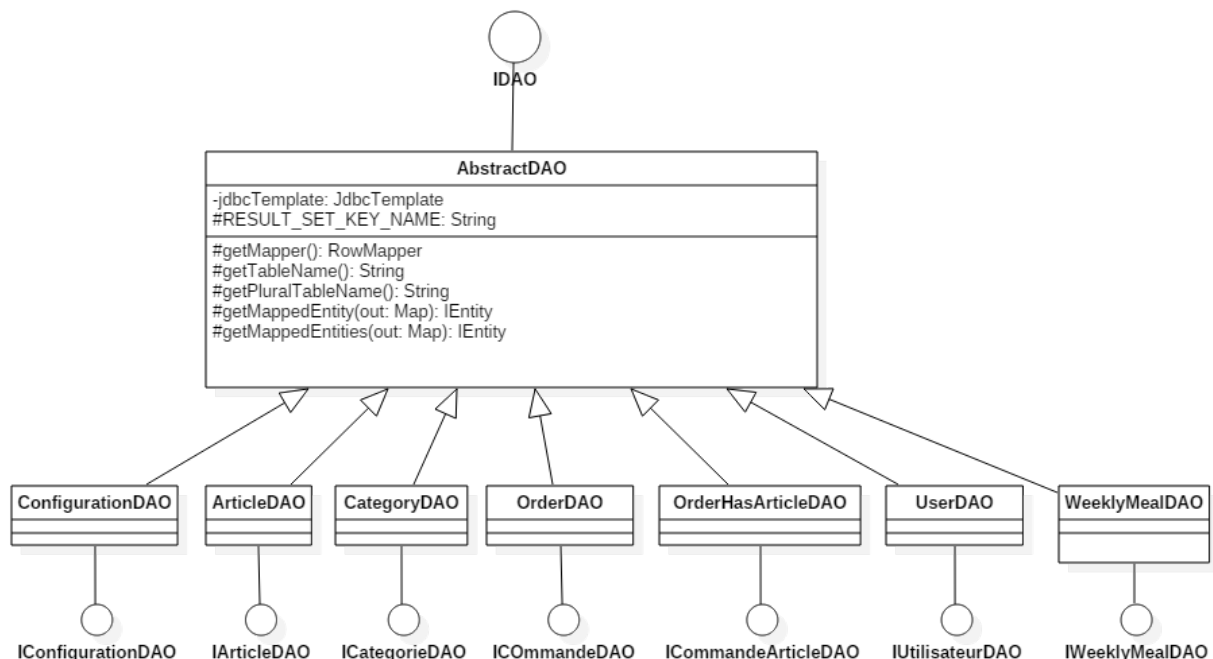
## C. DAO

### ➤ Diagramme de classe des interfaces DAO



Nos interfaces DAO décrivent les contrats que les classes concrètes devront implémenter afin d'assurer un fonctionnement cohérent de l'application.

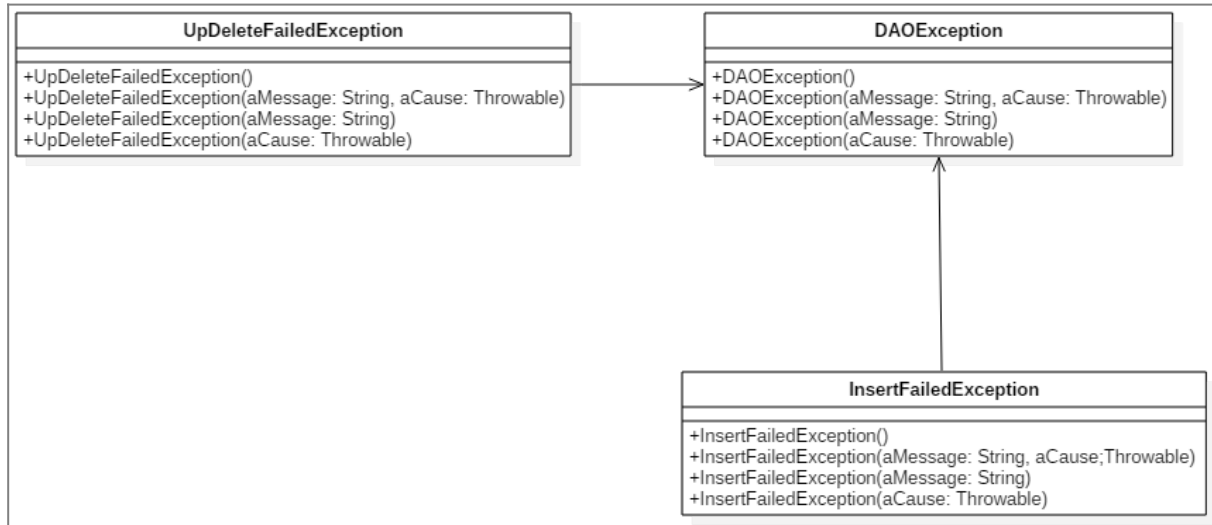
### ➤ Diagramme de classe des interfaces implémentées



Toutes nos classes concrètes de DAO hériteront d'une classe Abstraite «AbstractDAO» afin de partager les mêmes méthodes et attributs.

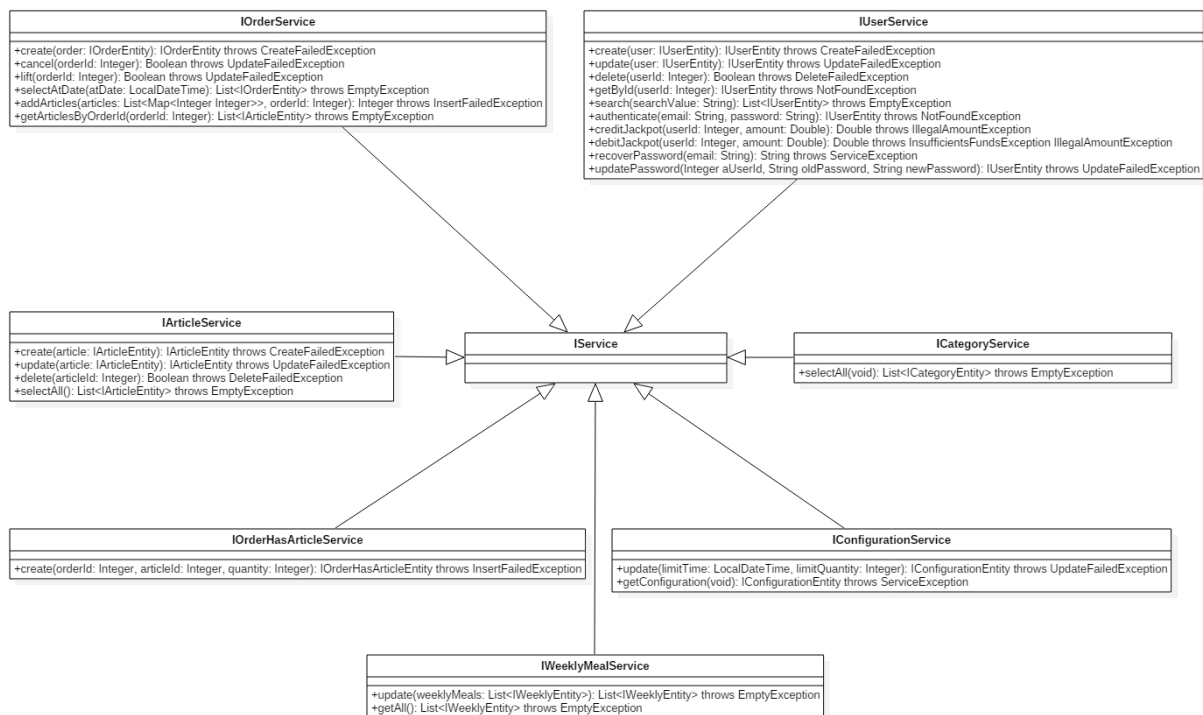


➤ *Diagramme de classe des exceptions liées aux DAO.*



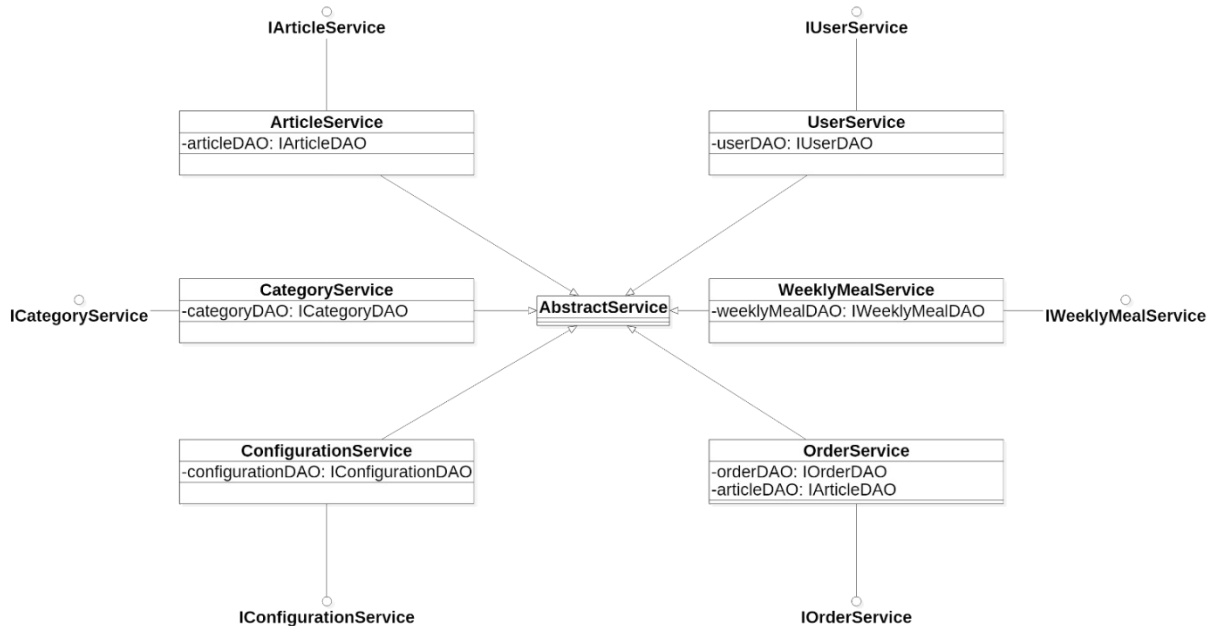
## D. Services

➤ *Diagramme de classe des interfaces Service*



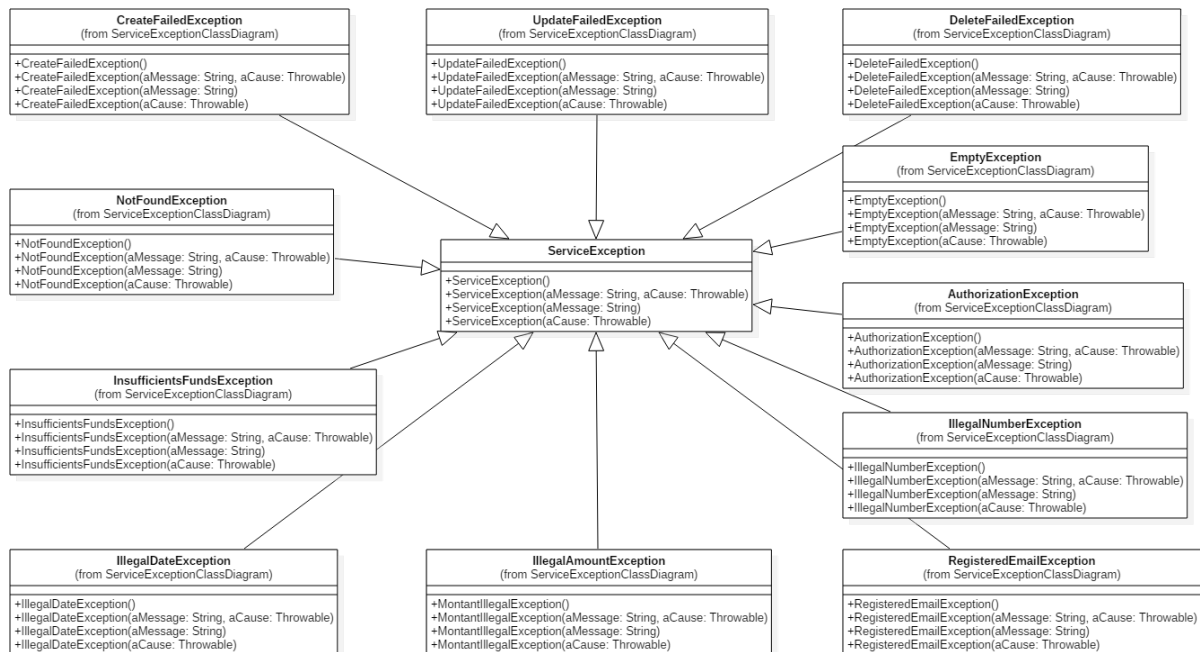
Nos interfaces Service décrivent les contrats que les classes concrètes devront implémenter afin d'assurer un fonctionnement cohérent de l'application.

➤ *Diagramme de classe des services.*



Toutes nos classes concrètes de Service hériteront d'une classe Abstraite «AbstractService» afin de partager les mêmes méthodes et attributs.

➤ *Diagramme de classe des exceptions des classes services.*



## E. Web Service

<b>WeeklyMealWebService</b> +weeklyMealService: IWeeklyMealService +list(): JSONResponseObject +update(meals: String): JSONResponseObject	<b>ArticleWebService</b> +articleService: IArticleService +create(designation: String, price: Double): JSONResponseObject +list(categories: String): JSONResponseObject +update(articleId: Integer, designation: String, price: Double): JSONResponseObject +delete(articleId: Integer): JSONResponseObject
<b>UserWebService</b> +userService: IUserService +create(email: String, password: String, lastName: String, firstName: String): JSONResponseObject +authenticate(email: String, password: String): JSONResponseObject +getAccount(userId: Integer): JSONResponseObject +getCurrentAccount(): JSONResponseObject +search(value: String): JSONResponseObject +recoverPassword(email: String): JSONResponseObject +update(email: String, lastName: String, firstName: String, oldPassword: String, newPassword: String): JSONResponseObject +creditJackpot(userId: Integer, amount: Double): JSONResponseObject +debitJackpot(userId: Integer): JSONResponseObject +deleteAccount(): JSONResponseObject	
<b>ConfigurationWebService</b> +configurationService: IConfigurationService +get(configurationId: Integer): JSONResponseObject +update(limitHour: LocalTime, limitQuantity: Integer): JSONResponseObject	<b>OrderWebService</b> +orderService: IOrderService +selectAtDate(date: LocalDate): JSONResponseObject +order(articles: String): JSONResponseObject +lift(orderId: Integer): JSONResponseObject +cancel(orderId: Integer): JSONResponseObject

Les web services fournissent des données au format JSON à partir d'un url d'accès dont la base est [www.cantine-aston.fr/api/v1](http://www.cantine-aston.fr/api/v1) et sera complété par le chemin d'accès adéquat.


Le serveur renverra toujours un STATUS 200 afin de pouvoir toujours faire circuler un objet JSON dans le corps de la réponse. C'est cet objet JSON qui permettra à l'application côté client, de gérer les états.

L'objet JSON retourné dans le corps de la requête HTTP sera au format suivant :

```
{
  status : 200 | 400,
  message_count : integer,
  data_count : integer,
  messages : [ ],
  data : [ ]
}
```

- La clé « status » contient l'état de la requête. Les valeurs utilisées sont 200 (OK) en cas de requête réussie ou 400 (Bad request) en cas de requête mal formée.
- La clé « message\_count » contient le nombre de messages que contient l'objet.
- La clé « data\_count » contient le nombre de données que contient l'objet.
- La clé « messages » contient un tableau de chaînes de caractères. Ces messages pourront décrire des erreurs en cas d'échec ou être simplement informatif en cas de réussite.
- La clé « data » contient un tableau d'éléments.

Les chemins d'accès aux ressources sont décrits ci-après.

	<b>Document</b> : <i>Spécifications d'Architecture Logicielle</i> <b>Projet</b> : <i>Cantine Aston</i>	<b>Réf.</b> : <i>SAL-06062016</i> <b>Version</b> : <i>V3.0</i> <b>Date</b> :
	Référentiel documentaire projet SQLI	Page 43 sur 49

➤ *Chemins des plats de la semaine :*

Fin de l'URL	Méthode appelée	Valeur passée en paramètre	Valeur de retour
GET/weekly_meal/list	WeeklyMeal WebService.list()	Aucune	Retourne l'ensemble des plats de la semaine
PATCH/ weekly_meal/ update? meals=""	WeeklyMeal WebService. update (String meals)	La chaîne de caractères passée en paramètre et formatée comme suit : «idArtX [, idArtY];idArtX [, idArtY];idArtX [, idArtY];idArtX [, idArtY];idArtX [, idArtY)» Chaque tuple d'entiers idArtX, idArtY représente une combinaison de plat pour un jour Cinq tuples doivent être fournis en paramètre, cependant chaque tuple dispose d'une valeur optionnelle	

➤ *Chemins de configuration :*

Fin de l'URL	Méthode appelée	Valeur passée en paramètre	Valeur de retour
PATCH /configuration/ update/1? limit_quantity="" &limit_time=""	Configuration WebService. update (LocalTime limitHour, Integer limiQuantity)	Le paramètre « limit_quantity » est un entier Le paramètre « limit_time » est formaté «hh:mm» Les paramètres sont obligatoires	
GET /configuration/ configuration_id	Configuration WebService.get (Integer configurationId)	Le paramètre « configuration_id » est un entier	

➤ *Chemins de commande :*

Fin de l'URL	Méthode appelée	Valeur passée en paramètre	Valeur de retour
GET/order/list?date="	OrderWebService .getAtDate (LocalDate date)	Le paramètre « date » est formaté «jj/mm/aaaa» et est obligatoire	Retourne une liste de commandes non livrées, non annulées pour la date indiquée en cas de réussite
PUT/order/create?articles='articleId1,articleQuantity1;articleIdN,articleQuantityN'	OrderWebService .order (String articles)	Pour chaque article un tuple «articleId, quantity» est formé La liste des tuples est transmise sous la forme d'une chaîne de caractères obligatoire	
PATCH/order/lift/{order_id}?action='lift'	OrderWebService .lift (Integer orderId)		
PATCH/order/cancel/{order_id}?action='cancel'	OrderWebService .cancel (Integer orderId)		

	<b>Document</b> : <i>Spécifications d'Architecture Logicielle</i> <b>Projet</b> : <i>Cantine Aston</i>	<b>Réf.</b> : <i>SAL-06062016</i> <b>Version</b> : <i>V3.0</i> <b>Date</b> :
	Référentiel documentaire projet SQLI	Page 45 sur 49

➤ *Chemins d'article :*

Fin de l'URL	Méthode appelée	Valeur passée en paramètre	Valeur de retour
PUT/article/ create? designation=" "& price=" "&is_meal=" "	ArticleWebService. create (String designation, Double price, Integer category)	Le paramètre « designation » est une chaîne de caractères obligatoire Le paramètre « price » est un nombre décimal optionnel, auquel cas, la valeur de l'article sera automatiquement 7.5, soit le prix par défaut d'une formule Le paramètre « category » désigne la catégorie de l'article ajouté	
GET/article/List? Catégories=	ArticleWebService. list (String categories)	Le paramètre « categories » contient les identifiants des catégories souhaitées séparés par des « ; »	
PATCH/article/ update/ {article_id}? designation=" "& price=" "	ArticleWebService. update (Integer articleId, String designation, Double price)	Le paramètre « designation » est une chaîne de caractères Le paramètre « price » est un nombre décimal Tous les paramètres sont obligatoires	
DELETE/articles/ delete/ {article_id}	ArticleWebService. delete (Integer articleId)		

	<b>Document</b> : <i>Spécifications d'Architecture Logicielle</i> <b>Projet</b> : <i>Cantine Aston</i>	<b>Réf.</b> : <i>SAL-06062016</i> <b>Version</b> : <i>V3.0</i> <b>Date</b> :
	Référentiel documentaire projet SQLI	Page 46 sur 49

➤ *Chemins utilisateur :*

Fin de l'URL	Méthode appelée	Valeur passée en paramètre	Valeur de retour
PUT/user/create? email="&password ="&lastname="&fir stname="	UserWebService. create (String email, String password, String lastName, String firstName)	Tous les paramètres sont obligatoires. Ce sont des chaînes de caractères	
GET/user/ authenticate? email="&password ="	UserWebService. authenticate (String email, String password)	Tous les paramètres sont obligatoires. Ce sont des chaînes de caractères	
PATCH/user/ recover/ {user_email}	UserWebService. recoverPassword (String email)		
GET/user/{user_ id}	UserWebService. getAccount (Integer userId)		
GET/user/current	UserWebService. getCurrentAccount ( )		
GET/user/search? value="	UserWebService. search()	Le paramètre « value » est obligatoire. C'est une chaîne de caractères	
PATCH/user/ update/ {user_id}? email="&lastname ="&firstname="& last_password="& new_password="	UserWebService. update (String email, String lastName, String firstName, String oldPassword, String newPassword)	Tous les paramètres sont des chaînes de caractères et optionnels. Cependant au moins un paramètre doit être renseigné Si le paramètre « last_password » est renseigné alors le paramètre « new_password » doit l'être aussi et inversement, sans quoi le mot de passe ne sera pas modifié	
PATCH/user/ credit/{user_id}? amount="	UserWebService. creditJackpot (Integer userId, Double amount)	Le paramètre « amount » est obligatoire. C'est un nombre décimal strictement positif	
PATCH/users/ sold/{user_id}	UserWebService. debitJackpot (Integer userId)		
DELETE/user/ delete/current	UserWebService. deleteAccount ( )		

	<b>Document</b> : <i>Spécifications d'Architecture Logicielle</i> <b>Projet</b> : <i>Cantine Aston</i>	<b>Réf.</b> : <i>SAL-06062016</i> <b>Version</b> : <i>V3.0</i> <b>Date</b> :
	Référentiel documentaire projet SQLI	Page 47 sur 49

## 5. Taille et performance

### A. Taille de l'application

Dans son format de test, la base de données représente un volume de 1Mo.

Elle contient :

- 4 utilisateurs
- 54 articles
- 5 plats du jour
- 8 catégories
- 1 configuration
- 15 commandes
- 15 liaisons d'articles vers commande

Les données qui évolueront dans le temps concernent :

- La table `user`
- La table `article`
- La table `order`
- La table `order\_has\_article`

L'école Aston accueille environ 300 personnes (stagiaires + personnel) par an, chacune pouvant passer jusqu'à cinq commandes d'un article par semaine. Ces informations nous permettent de calculer une augmentation potentielle du volume de la base de données.

Nous estimons le nombre d'utilisateurs de l'application à 50% de l'ensemble des personnes fréquentant l'établissement.

Afin de faciliter les calculs, certains détails sont volontairement mis de côté et les valeurs sont arrondies à l'entier supérieur.

Taille de la base = 1Mo

Espace d'une donnée =  $1 / 102 = 0.01\text{Mo}$

Espace utilisé par la table `user` =  $4 * 0.01 = 0.04\text{Mo}$

Espace utilisé par la table `article` =  $54 * 0.01 = 0.54\text{Mo}$

Espace utilisé par la table `order` =  $15 * 0.01 = 0.15\text{Mo}$

Espace utilisé par la table `order\_has\_article` =  $15 * 0.01 = 0.15\text{Mo}$

Nous constatons que le volume des tables « order\_has\_article » et « order » évoluent au même rythme, pour simplifier nous multiplions la taille « order » par 2.

Chaque jour le volume de données que consomme un utilisateur en commande serait de :

$0.01 * 2 = 0.02 \text{ Mo}$ .

Donc l'augmentation prévisionnelle du volume par an par utilisateur serait de :

$260 * 0.02 = 5.2 \text{ Mo}$ .



	<b>Document</b> : <i>Spécifications d'Architecture Logicielle</i> <b>Projet</b> : <i>Cantine Aston</i>	<b>Réf.</b> : <i>SAL-06062016</i> <b>Version</b> : <i>V3.0</i> <b>Date</b> :
	Référentiel documentaire projet SQLI	Page 48 sur 49

L'application prévoit de pouvoir commander jusqu'à deux plats différents chaque jour, la cantinière peut donc en conséquence créer deux nouveaux articles par jour.

Espace consommé par jour en création d'article :

$$0.01 * 2 = 0.02 \text{ Mo.}$$

Donc l'augmentation prévisionnelle du volume par an pour la création d'articles serait de :

$$260 * 0.02 = 5.2 \text{ Mo.}$$

Chaque année l'école Aston accueille jusqu'à 150 personnes (stagiaires + employés) susceptibles d'utiliser l'application.

Donc l'augmentation prévisionnelle du volume par an pour la création d'utilisateur serait de :

$$150 * 0.01 = 1.5 \text{ Mo.}$$

Par conséquent, l'augmentation globale du volume de la base de donnée par an serait de :

$$150 * 5.2 + 5.2 + 1.5 = 78.67 \text{ Mo.}$$

## B. Performance de l'application

Cf. : [BahBrierreCoulibaly\\_CantineAston\\_PlanDeTestsEtDeValidation\\_V3\\_2016.docx](#)

### IV. Tests de charge

1. Connexion simultanée de plusieurs utilisateurs