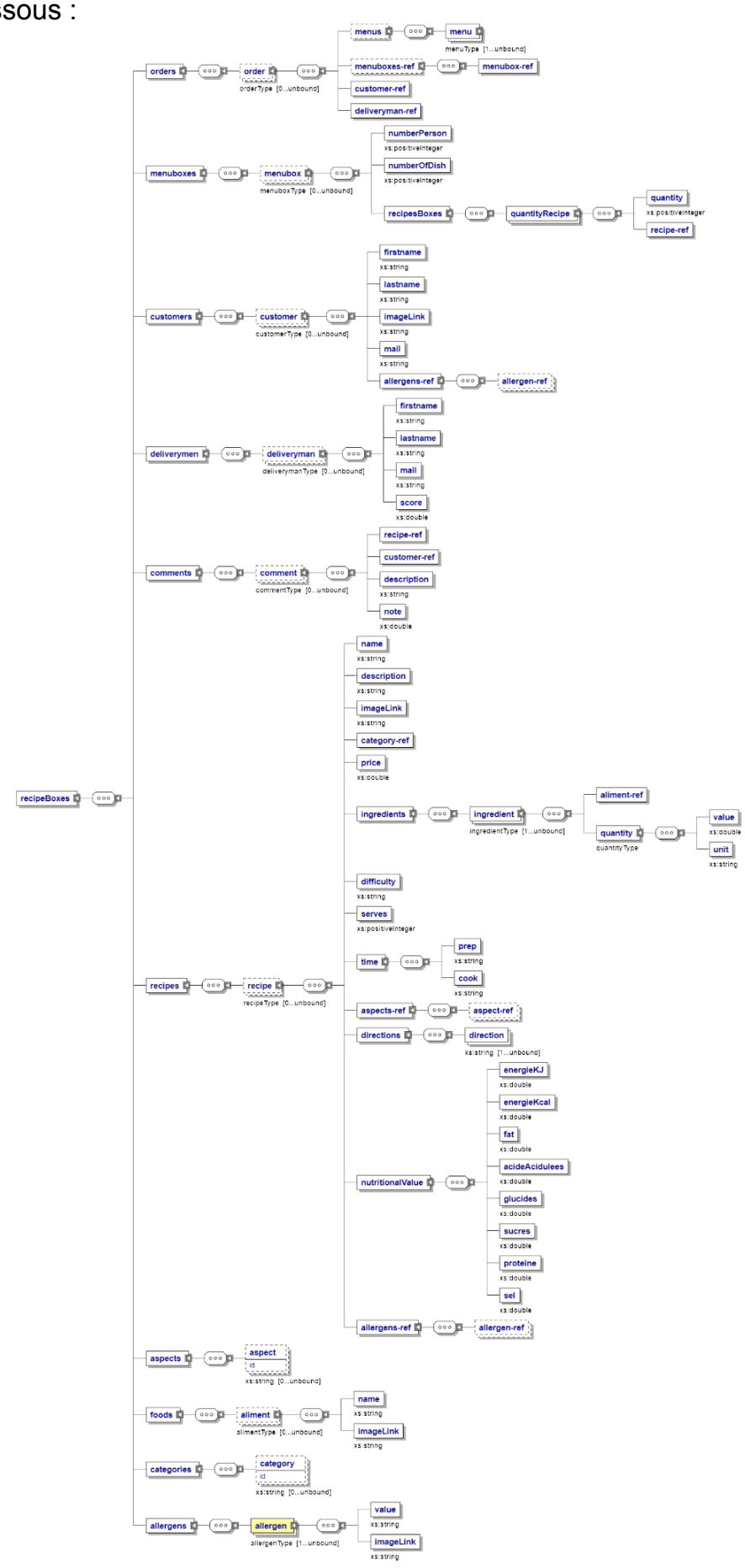
Base de données / Langages du Web

Rapport du Mini-projet

Binôme :
Nicolas ZANIN
Sayf Eddine HALMI

1 - Choix de modélisation :

Nous avons choisi de décrire notre schéma XML comme présenté dans l'arbre ci-dessous :



Notre document XML contient donc, un historique de commande (orders) ainsi qu'un historique de commentaires (comments), en plus de plusieurs listes d'éléments de même type que l'on pourrait aisément assimiler à des bases de données modélisant : les offres possibles de façon simplifiée (recipes et menuboxes), et les acteurs principaux (customers et deliverymen) de l'éventuel système de commande.

Dans l'objectif de maximiser la clarté du document ainsi que la réutilisabilité des éléments et informations, on peut remarquer l'usage très fréquent d'idref permettant de définir les éléments partagés à une seule reprise et les appeler dans d'autres composants. On citera à titre d'exemple les allergènes pour lesquels une base de données a été définie explicitement et l'usage d'allergens-ref pour les customers et les recipes.

On notera enfin que, dans le but de ne pas surcharger le document et faciliter sa lecture, nous avons omis de déclarer certains éléments qui, hors du contexte d'un projet scolaire, pourraient s'avérer importants (ex: catégories de menuboxes, ustensiles, dates…).

2 - Avantages & Inconvénients :

Avantages :

Notre modélisation permet de garder plusieurs types de données séparément des autres données, avec plusieurs identifiants ce qui permet de pouvoir changer rapidement ce type de données sans changer les valeurs à plusieurs endroits dans le XML.
De plus, pour créer un objet qui contient tous ces types de données sera plus simple, puisque via le identifiants de chaque objets, on peut créer un objet contenant tous ces objets.
Ainsi, ça permet de ne pas créer de la redondance parmi ces objets.
Nous stockons le type de catégorie d'une recette dans une recette et comme un menu contient uniquement une recette alors pour afficher la catégorie d'un menu, il faut regarder la valeur de recette (l'enfant de menu) et l'afficher.

Inconvénient :

L'inconvénient de ce type de modélisation est que la profondeur du xml se crée vite, et donc lors de certains scénarios, la profondeur de ces balises xml devenait importante et donc difficilement manipulable et nécessitant de plusieurs 'key' pour obtenir certaines données.

3 - Scénarios et leurs implémentations :

Scénario Basique (1, 2 et 3):

Pour ce premier scénario, nous affichons les commentaires de chaque client sur une recette ainsi que la note que le client à ajouter sur la recette.
De plus, les commentaires sont triés par note, puis par nom de recette puis par nom de client.

Donc pour implémenter ceci, on à utiliser le xsl:sort pour trier les données, et utilisation de la fonction key dans le select pour récupérer les valeurs des objets référencées par leur identifiant.

Pour le deuxième scénario, nous affichons les recettes, avec leur noms, description et instruction avec leur ingrédients et quantité.

Donc pour implémenter ce deuxième scénario, il suffisait de passer par les key pour récupérer les valeurs des objets référencées.

Pour le troisième scénario, nous affichons le nombre de menus livrées pour chaque commande ainsi que le nom et prénom du livreur et du client.

Donc pour implémenter ce troisième scénario, on a utiliser la fonction count pour compter le nombre de menu et les key pour récupérer les valeurs des données du client et du livreur

Scénario 4:

Le scénario consiste à recréer une liste de recipes pour un client en particulier de façon à ce qu'aucune d'elle ne contienne d'allergène dont il pourrait souffrir, tout en les triant par ordre de prix et par ordre alphabétique afin de simplifier la recherche d'une recette en particulier si besoin.
Nous avons pu réaliser cela en XSL à l'aide d'un paramètre représentant le client en particulier (qui est donc "hardcodé") ainsi que l'usage du mot clé d'une condition dans la balise apply-templates en prenant garde à correctement éliminer les recettes dont au moins l'un des idref de leurs fils allergen-ref correspond à un idref du fils allergen-ref du client en question.
Quand à la récupération des autres informations relié aux recettes par référence il s'agit d'utiliser de rechercher les valeurs dans leurs bases de donnée de bases à l'aide du mot clé select dans la balise value-of et l'usage de la condition "/type/d/element/recherché[@id = current()/@idref]" ou autre similaire

Scénario 5:

Pour ce 5 ème scénario, nous pouvons afficher l'historique des commandes via un json généré depuis un fichier xml.

Pour effectuer ceci, on doit utiliser plusieurs keys et template pour pouvoir récupérer toutes les valeurs de chaque commande passée.
Donc pour palier à l'utilisation de plusieurs keys dans une même requête, nous avons utilisé des 'condition' qui permet de rechercher les recette en fonction de l'identifiant courant et l'identifiant faisant référence puis grâce à cette méthode, on a pu passer les requêtes de 2 à 1 key dans un template et donc c'est plus facile à récupérer toutes les valeurs.

Scénario 6:

Pour ce dernier scénario, on affiche le nombre moyen de commentaires par client.
Puis le nombre de commentaires par client et par recette.
Ensuite le nombre moyen de commentaire,
Affichage des aliments et des clients.

Pour obtenir ces données, on utilise la librairie dom4j, et on utilise donc le XPATH, pour sélectionner les données à récupérer.