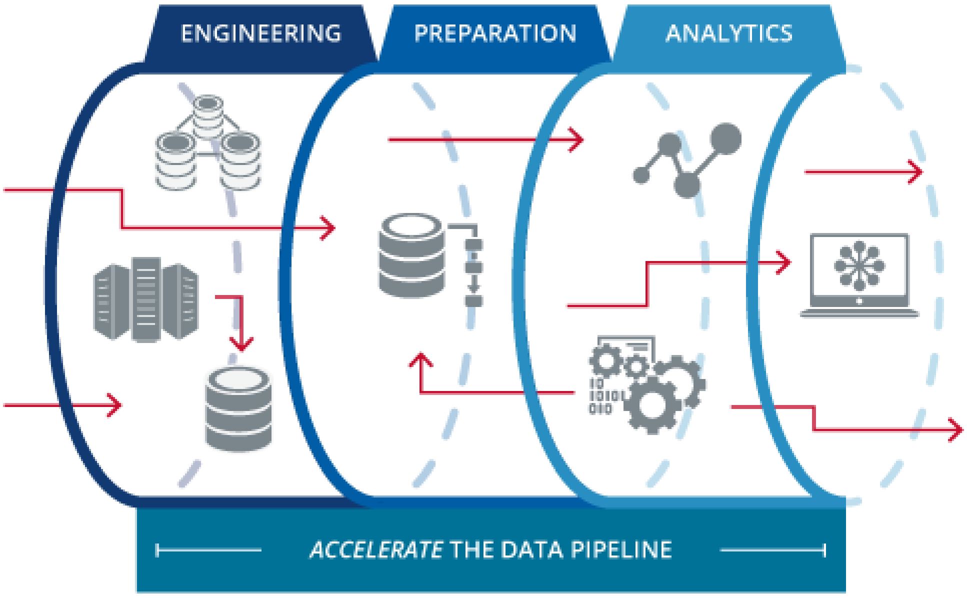
PROJECT REPORT

MODELING OF AN XML DATABASE FOR THE INFORMATION SYSTEM OF A RECIPE BOX COMPANY

Cohort : A23
Course : Data Pipeline - part 1
Instructor : Mrs. Catherine FARON



Team

Ken CHEVALIER,	DA,	Sophia	ken.chevallier@edu.dsti.institute
Archana KHANDELWAL,	DA,	Paris	archana.khandelwal@edu.dsti.institute
Chakib DOUADI	DE,	Paris	chakib.douadi@edu.dsti.institute
Alexandre BENAMENYO	DA	Paris	alexandre.benamenyo@edu.dsti.institute

Table of contents

Introduction 3
1. Workload Distribution 3
2. Working Environment and Tools 4
3. Principles of Modeling and Design Choices 4
4. Scenario Implementations 5
    4.1 Scenario 1 5
    4.2 Scenario 2 5
    4.3 Scenario 3 5
    4.4 Scenario 4 6
    4.5 Scenario 6 6
    4.6 Scenario 7 6
5. Complex Scenario Explanation 7
6. Project Resources 8
    6.1 Github repository 8
    6.2 Web Application 8
Conclusion 9
Appendix : AI assistance 9

Introduction

In this project, we embarked on creating an XML database for a Recipe Box company, aiming to streamline the management of recipes, menus, customer orders, and reviews. Inspired by industry standards and leveraging insights from the BBC Good Food magazine review, our objective was to construct a system that captures the essence of culinary management, incorporating aspects like gastronomy, health,etc. Our approach involved designing a comprehensive XML Schema to structure our database, populating it with a representative dataset and showcasing its functionality through various XSL transformations. Our vision extended beyond the database to the development of a web application designed to bring our XML data to life in a user-friendly interface.

This report outlines our work, detailing the project's inception, the challenges faced, and the methodologies adopted to achieve a cohesive and functional XML database.

1. Workload Distribution

Our team consists of four dedicated students: three Data Analysts and one Data Engineer. Three of us are engaged in an accelerated program, while the fourth member is an alternant.

Together, we bring a diverse set of skills and perspectives to our projects, united by our passion for data and its potential to drive meaningful insights and innovations.

This table summarizes the individual contributions of team members to various aspects of the project, detailing who did what in terms of schema definitions, database development, transformations, and report contributions.

Member	Task
Ken CHEVALLIER	Schema definition, namespaces declarations
	Database build (3 recipes added)
	1 XSL transformation (html)
	Sharepoint creation and maintenance
	Github repository hosting and maintenance
	Contribution to project report
Archana KHANDELWAL	Research on web resources for data model
	Database model design
	Schema definition
	Database build (18 recipes added)
	3 XSL transformations (2 html + xml)
	Contribution to project report
Chakib DOUADI	Web application design and development
	Project report creation and repository
	Database build (15 recipes)
	3 XSL transformations (2 html + Json), including complex one (3 in 1, see chapter 5).
	Contribution to project report
ALEXANDRE BENAYENMO	Database build (15 recipes)
	1 XSL transformation
	Contribution to project report

2. Working Environment and Tools

The table outlines the working environment and tools utilized in the project, encompassing operating systems, integrated development environments, editors, web application tools, teamwork platforms, and schema validation + data conversion online tools (xml api + python).

OS	ubuntu (virtual machine), windows
IDE	visual studio code
Editor	notepad++ (XML Tools, JStool), sublime
Language	python, xml, json
Web app, container, hosting	streamlit, docker, heroku
Team work, versioning	sharepoint, teams, github
schema validator (xml, json)	https://www.dsti-food.com/XML_Management_Studio
xml to json converter	
domain name, SSL certificate, emails @dsti-food.fr	www.ionos.fr

3. Principles of Modeling and Design Choices

To build the database and define the schema associated with it, we went through several websites of recipe box companies, to get some inspiration for the structure of the dataset and some ideas for the possible transformations in XSLT.

Eventually there were 2 recipe websites that stood out:

- Greenchef.co.uk: recipe box company specifically aimed at “green” categories of meals.
- Pinchofyum.com: more of a blog containing hundreds of recipes to try at home (so not a vendor).

After beginning implementing the database and schema based on both websites, we came to realize that the pinchofyum.com site would be the most adapted to our needs: it contains the most categories we could imagine to use for XSLT (from course type, diet type, season, to method used, and more), and a very complete information for each recipe (ingredients, nutritional score, ratings, FAQ).

The XML database is therefore built from the most interesting recipes found on the website, with about all the information there is to find on each recipe.

The XSD schema has been defined to cope with the variety of categories that define a recipe. We therefore chose a structure of key/keyref for the following elements:

- Diet category (gluten-free, dairy-free, vegetarian, …)
- Cuisine (American cuisine, French cuisine, Indian cuisine, …)
- Course type (Soup, Salad, Sandwich, Dessert, …)
- Meal type (Breakfast, Lunch, Dinner, Snack)
- Season (Winter, Spring, Summer, Fall)
- Method used (Casserole, Bake, Stovetop, …)
- Nutrition value (Total Fat, Sugars, Protein, …)

The rest of the information (ingredients, reviews, FAQ, instructions) were kept as xml schema elements either of simple type (int or string) or of complex type (for ingredients, as a sequence of “title of ingredients” and list of ingredients).

We chose to use a named namespace for the declaration of our database, under the address xmlns:ds=http://www.dsti-food.com/

And this address also hosts the web app that we developed for this project (see chapter 6).

Although the database and schema were already quite complete, we realized that we were lacking a key piece of information on each recipe, which is related to the allergens that could be contained in it. Therefore, after doing some research, we gathered information from https://www.foodallergy.org/living-food-allergies/ and from www.greenchef.co.uk to build a list of allergens (also using key/keyref) that we would arbitrarily put in each recipe.

To keep the separation between the 2 sources of information (1 for recipe, 1 for allergens) and possibly ease the maintenance, we decided to introduce a second namespace specific to the allergies definition (xmlns:al="http://allergens.dsti.com/), and adapt the XSD schemas accordingly, by using a 2nd XSD file thanks to the xs:import instruction.

<xs:import namespace="http://allergens.dsti.com/"

schemaLocation="FoodAllergens.xsd"/).

This results in a mix of default namespace (for the recipes) + named namespace (for the allergies) in the XML database.

4. Scenario Implementations

4.1 Scenario 1

The "transfo_01.xsl" XSLT stylesheet is designed to generate a user-friendly recipe list adhering to precise criteria. These criteria encompass recipes with fewer than 300 calories, apt for lunchtime consumption, and ideally suited for the summer season. The stylesheet also provides information on the total cooking time for each recipe. Its primary objective is to offer a selection of healthy and seasonally appropriate meal options tailored specifically for the summer season.

4.2 Scenario 2

The second transformation serves as a valuable tool for individuals with egg and wheat allergies.

By filtering through a collection of recipes, it compiles a comprehensive list of allergy-free recipes. This functionality aids in dietary planning, allergy management, and ensures accurate allergy information for users. It's a powerful resource for creating safe and informed meal options.

4.3 Scenario 3

This XSLT, designed for calculating the average ratings of dishes across different meal types (such as Soup, Salad, Sandwich, Main Dish, and Baking),

The primary goal is to compute and display the average rating for each predefined course type, providing insights into the collective appreciation of recipes within each category.

The stylesheet begins by declaring variables for different course types. This structured approach allows for easy adaptation or expansion to include additional course types without altering the core logic.

For each course type, the stylesheet dynamically selects relevant recipes and calculates the average rating by accumulating the product of the number of reviews and the average rating for each recipe.

This process involves a custom recursive template (accumulateSums) adept at handling this computation across potentially varying numbers of reviews per recipe.

The output is structured in HTML, with headings for each course type followed by the calculated average rating. This format is conducive to easy interpretation and could be directly integrated into our application.

For this, we used ChatGPT to provide help on the calculation, with several non-working iterations, but with a general correct structure on the use of recursive function (see section on AI assistance).

4.4 Scenario 4

This XSLT is designed to list recipes categorized by their course types (e.g., Soup, Salad, Main Dish) and further segmented based on cooking time—specifically, recipes with cooking times less than 30 minutes. It aims to provide a structured overview that helps users quickly identify recipes according to the time they have available for cooking.

The output is structured in HTML, presenting the course name followed by a list of relevant recipes, including their titles and total cooking times. This structure is conducive to web display and user interaction.

4.5 Scenario 6

The XML data produced through the XSLT transformation serves as a valuable resource within our meal planning application project. It furnishes detailed insights into various aspects of recipes, with a specific focus on diet categories and allergens. By integrating this XML data into our recipe database, we augment the depth and breadth of recipe information available to our application.

Through XML parsing, our application efficiently extracts essential details pertaining to diet categories and allergens associated with each recipe. This extracted data empowers our system to effectively filter recipes according to user-specified preferences, especially with regards to excluding allergens such as eggs and wheat. This filtering capability ensures that our application only presents recipes that align with the user's dietary requirements and restrictions.

Consequently, leveraging this XML data enables our application to dynamically generate personalized meal plans tailored precisely to each user's unique dietary needs. By ensuring that all suggested recipes conform to the user's specific allergen restrictions and preferred diet categories, our application enhances user satisfaction and confidence in the meal planning process. This utilization of XML data underscores the effectiveness and adaptability of our meal planning solution in meeting the diverse dietary requirements of our users.

4.6 Scenario 7

This XSLT stylesheet is designed to extract recipes categorized under the “Dinner” meal type (mt2) into a structured JSON format.

For a display purpose, we choose only the first two recipes that match this criteria.

This approach is particularly useful for applications that require a subset of data in a format that's readily consumable by web technologies and APIs.

The transformation organizes recipe data into a structured JSON format, including titles, diet categories, cuisines, ingredients (with name, quantity, and unit), and cooking instructions.

To ensure the integrity of the structured data, it is validated against a predefined JSON schema, affirming that the transformed data adheres to the specified structure and types required.

5. Complex Scenario Explanation

For the complex scenario, we choose another approach to navigate the database within a streamlit application and three transformations (xslt), offering a highly interactive and user-centric experience.

We try to guide the user through the recipe selection process, from meal type to specific recipes and their details.

This setup begins with the categorization of recipes by meal types, proceeds to selection based on a specific meal type, and culminates in the detailed display of a chosen recipe.

First XSLT: Listing Meal Types

This stylesheet extracts and displays all available meal types (e.g., Breakfast, Lunch, Dinner) from the XML data source. It produces an HTML list, providing a straightforward interface for users to view the meal types. This step acts as the entry point in our application, where users begin their journey by selecting a meal type of interest.

Second XSLT: Selecting Recipes by Meal Type

Upon selecting a meal type (extracted from the xslt output and displayed in a selectbox), this stylesheet filters recipes corresponding to the user's choice, displaying them in a styled HTML table with recipe titles and their associated cuisines. It allows users to visually identify which recipes they might be interested in exploring further based on the meal type and cuisine, enhancing the interactive experience by narrowing down the selection.

Last XSLT: Detailed Recipe Information

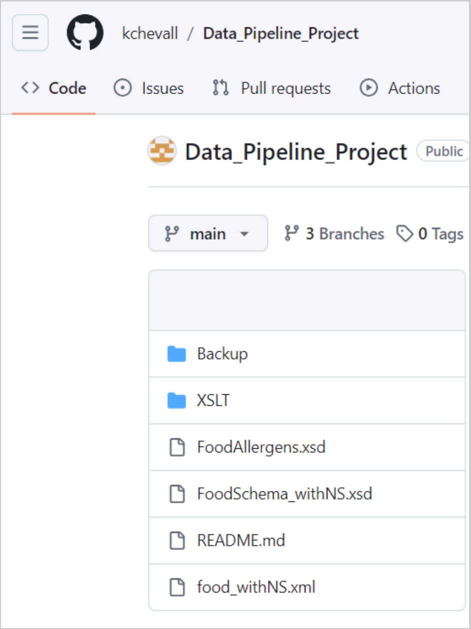
Once a user selects a recipe, this final stylesheet fetches and presents all relevant details about the recipe, including its title, description, ingredients (with quantities and units), and cooking instructions.

The detailed information is formatted in HTML, offering users a comprehensive overview of the recipe. This step provides the depth of information needed for users to understand the recipe fully and, potentially, to prepare it themselves.

The user can rate the recipe and send us a feedback through a form or email feedback@dsti-food.com or contact the administrator : admin@dsti-food.com.

6. Project Resources

6.1 Github repository



kchevall / Data_Pipeline_Project

Data_Pipeline_Project Public

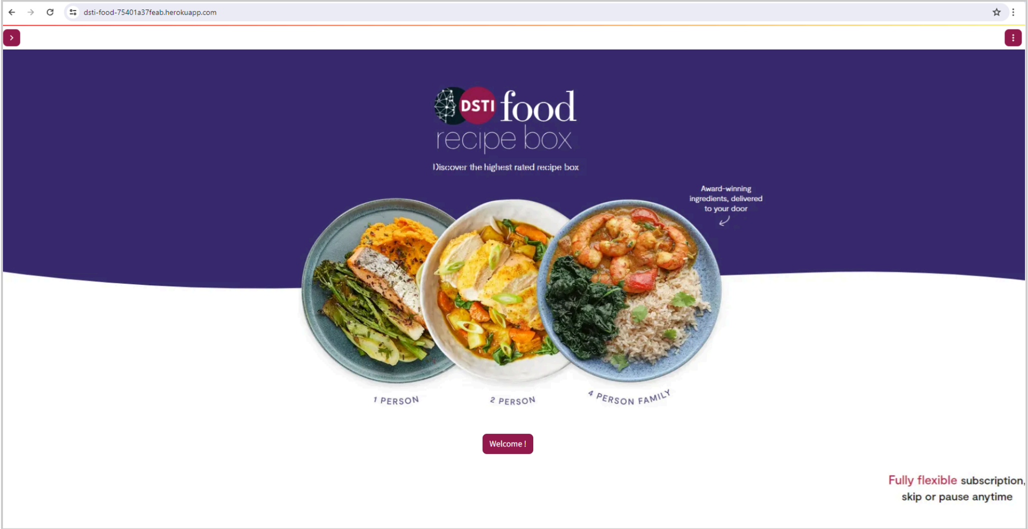
- Backup
- XSLT
- FoodAllergens.xsd
- FoodSchema_withNS.xsd
- README.md
- food_withNS.xml

https://github.com/kchevall/Data_Pipeline_Project

6.2 Web Application

Our vision extended beyond the database to the development of a web application designed to bring our XML data to life in a user-friendly interface. Utilizing Python and Streamlit, we crafted an application that allows users to interact with the data dynamically. This application is hosted on Heroku, providing a reliable and scalable cloud platform for deployment (https://dsti-food-75401a37feab.herokuapp.com/).

Furthermore, to ensure a professional user experience, we secured a domain name and SSL certificate from www.ionos.fr, guaranteeing our users secure and trustworthy access to our services.



dsti-food-75401a37feab.herokuapp.com

Discover the highest rated recipe box

Award-winning ingredients, delivered to your door

Fully flexible subscription, skip or pause anytime

we let you discover our application : www.dsti-food.com

Conclusion

Our project successfully established a foundational XML database for a Recipe Box company, showcasing its potential through various scenarios and transformations.

Despite achieving our core objectives, time constraints limited our ability to expand the database to include users, deliverers, and orders comprehensively. However, our efforts have laid the groundwork for future enhancements, demonstrating the practical application and adaptability of XML in managing complex data in the culinary domain. This experience has highlighted both our achievements and the areas ripe for further development.

Appendix : AI assistance

For transfo3.xsl: average rating per course type (soup, salad, sandwich, ...)
We used chatGPT to get an inspiration for the recursive function:
You:
With this XML:

```
<?xml version="1.0" encoding="utf-8"?>
<Food xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.dsti-food.com/FoodSchema_withNS.xsd">
<!-- List of recipes -->
<recipe>
<type>lunch</type>
<reviews>
<nbReviews>2</nbReviews>
<note>6</note>
</reviews>
</recipe>
<recipe>
<type>dinner</type>
<reviews>
<nbReviews>2</nbReviews>
<note>8</note>
</reviews>
</recipe>
<recipe>
<type>lunch</type>
<reviews>
<nbReviews>3</nbReviews>
<note>8</note>
</reviews>
</recipe>
<recipe>
<type>dinner</type>
<reviews>
<nbReviews>4</nbReviews>
<note>9</note>
</reviews>
</recipe>
</Food>
```

How do I get the sum of all products of nbReviews and note?

ChatGPT:
To calculate the sum of all products of `nbReviews` and `note` in the given XML, you can use XSLT. Below is an XSLT example that accomplishes this:

~~~
```xml
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text"/>
~~~

```
 <!-- Template to match the root element -->
 <xsl:template match="/Food">
<!-- Variable to store the sum -->
<xsl:variable name="sum">
 <!-- Calculate the sum -->
 <xsl:call-template name="calculateSum">
  <!-- Pass all recipe nodes as parameter -->
  <xsl:with-param name="recipes" select="recipe"/>
 </xsl:call-template>
</xsl:variable>

<!-- Output the sum -->
<xsl:text>Total sum of products: </xsl:text>
<xsl:value-of select="$sum"/>
 </xsl:template>

 <!-- Template to calculate the sum recursively -->
 <xsl:template name="calculateSum">
<!-- Parameters: nodes to sum and current total -->
<xsl:param name="recipes"/>
<xsl:param name="total" select="0"/>

<!-- Base case: if there are no more nodes, return the total -->
<xsl:choose>
<xsl:when test="not($recipes)">
<xsl:value-of select="$total"/>
 </xsl:when>
 <!-- Recursive case: sum the product of nbReviews and note for the current recipe
     and call the template recursively with the next recipe and updated total -->
 <xsl:otherwise>
  <xsl:call-template name="calculateSum">
   <xsl:with-param name="recipes" select="$recipes[position() > 1]"/>
  <xsl:with-param name="total" select="$total + $recipes[1]/reviews/nbReviews * $recipes[1]/reviews/note"/>
  </xsl:call-template>
 </xsl:otherwise>
</xsl:choose>
 </xsl:template>
</xsl:stylesheet>
```

This XSLT calculates the sum recursively by iterating over each `<recipe>` element, multiplying the values of `nbReviews` and `note` within the `<reviews>` element, and adding the result to the total. Finally, it outputs the total sum of all products.

Data Description

We utilized ChatGPT to assist us in describing the data. The cuisine descriptions in the XML database, generated with ChatGPT, offer concise insights into each cuisine's specialties, thereby enriching the recipe dataset.

Time Extraction

We were initially obtaining the output in the format 'PT15M'. However, with the assistance of ChatGPT, we were able to craft a code snippet to convert this format into a more user-friendly representation, such as '15 minutes'. Subsequently, we seamlessly integrated this code into our XSL transformation, ensuring precise showcasing of the cooking times for each recipe.