Recipe-Company-RDF-Graph

EXPLANATATION – VINZENT ASCHIR AND BALAJI GOVINDARAJAN

RECIPE LEVEL – VINZENT

We implemented a *Recipe* class with nine properties, four of which store general information as strings. Each recipe is assigned a unique ID (string) to distinguish between different recipes for the same dish.

CUISINE

Gastronomies are represented by subclasses of *Cuisine*, such as *ItalianCuisine*. Recipes or menus are linked to the appropriate cuisine using the *hasCuisine* property.

Health

To account for health concerns, we created instances of the Allergen class (e.g., Gluten) and linked them to recipes or ingredients using a dedicated property to indicate potential allergic reactions. Recipes can also be categorized as Vegan or Vegetarian by making them instances of these subclasses.

Course Classification

The Class *Courses* (another subclass of Recipe) organizes recipes into different courses. Recipes can be instances of *MainDish*, *Appetizer*, or *Dessert*.

Time

durations are represented using ISO 8601 duration format, allowing for flexible specification of periods ranging from a few minutes to multiple days. This approach also facilitates integration with external ontologies, such as Schema.org.

Ingredients

To provide a machine-readable ingredient list and avoid duplication, recipes are linked to an IngredientBundle instance via the hasIngredientList property. Each bundle contains triples representing:

- the ingredient
- its amount (decimal)
- the unit

Both amount and unit are optional; for example, spices like sugar may not have a quantity or unit and are included only for flavour.

Subclasses of Ingredients

Ingredients have subclasses that specify their type, such as *Spice* or *Drink*. These subclasses can be linked to allergens (e.g., *DairyProduct* to *Milk*-allergy). The property indicating *Allergen* is inherited by *Ingredient*. Instead of using *rdf:label*, we use *rbox:hasName* to assign names to *Ingredients*-instances.

Units
The Unit class has several subclasses, such as *VolumeUnit* or *CountUnit*. Specific units (e.g., *grams*, *inches*) are instances of these subclasses.

Menu
Different recipes can be grouped together within an instance of the *Menu* class. For demonstration, we created instances representing German, French, and Italian menus. Each menu instance uses the same core identifying and descriptive attributes as individual recipes, such as *hasID* and *hasDescription*.

COMPANY LEVEL – BALAJI

How it Works:
Customers subscribe to a *SubscriptionPlan*. After the selection among various subscriptionPlans, the included menu had been chosen. The recipes that belong to this ordered menu will be ordered by an *OrderLine*.

Subscription plan:
The Subscription is divided into categories such as 3 recipes x 2 people and 4 recipes x 4 people. Based on the customers requirements a plan will be chosen. An instance of the class *SubscriptionPlan* hold the properties (*planOrderCount*, *planReviewCount*, *planAverageRating*, *planPopularityScore*) to help evaluate popularity. With the help of the popularity the customer can choose the plan.

Order:
An instance of the class Order holds an *OrderLine* with the Property *hasorderLine.* The *OrderLine* instance, belonging to an order, is summarizing to the customer all the ordered recipes from different menus or single chosen recipes.

Courier:
The instance *DeliveryPerson* of the class *Courier* has a rating (string) given by the customers. The Attributes *deliveryStart* / *deliveryEnd* inform you whether your delivery is delayed. An instance of *Courier* has also properties indicating the customers details and the destination of the order.

Review:
A Review is has a review body and review ratings. The review body contains the actual review as an text (string), where the review ratings are an integer. Based on the review, a popularity score is calculated.

When properties range to an integer string or any other literal I decided to use rdfs:DataProperty instead of rdf:Propery.

Coding
All class definitions at the recipe-level are labeled and commented in German and English. Some instances and properties are also labeled; however, full labeling and commenting were omitted due to time constraints. GraphDB Demo Version was used for visualization. We used ChatGPT, GitHub Copilot, Google, and other search engines to assist with debugging and coding. The design of the RDF graph, ontology decisions, and overall implementation were performed independently by the team.

1643 triples.