



MASTER DEGREE IN COMPUTER SCIENCE
Knowledge Engineering and Business Intelligence

Project "Personalized Wine Menu"

Group members

Clarissa Albanese

clarissa.albanese@studenti.unicam.it

Greta Sorritelli

greta.sorritelli@studenti.unicam.it

Professors

Knut Hinkelmann

Holger Wache

Abstract

The project aims to address the challenges faced by restaurants with digital menus due to COVID-1. While QR codes allow guests to access menus on their smartphones, the small screen size makes it difficult for them to get a full overview, especially when dealing with extensive wine menus. In order to meet consumer preferences, which may include specific countries, regions, or grape preferences, as well as their desired taste profiles, a system needs to be developed to ease the selection of wines that match both guest preferences and menu.

The objective of this project is to build a knowledge-based system that represents information about wines typically found in international restaurants. The system utilizes input such as menus and guest preferences to support the wine selection process. To accomplish this, several knowledge-based solutions are implemented, based on decision tables, Prolog, and a knowledge graph. These solutions enable the system to provide tailored recommendations based on user input.

In addition, a graphical modeling language is designed to allow chefs to visually represent meals and wines, adding all relevant information required for customers to make informed choices based on their preferences. By combining the knowledge-based system with the graphical modeling language, this project aims to enhance the wine selection experience for guests in restaurants, improving customer satisfaction and engagement.

Contents

1	Introduction	8
1.1	Project Description	8
1.2	The Scope	8
2	Technologies	9
2.1	Camunda Modeler	9
2.2	Camunda Simulator	9
2.3	Prolog Language	10
2.4	Protégé	11
2.5	ADOxx	12
3	Knowledge Base	14
3.1	The inputs of the Knowledge Base	14
3.1.1	Guest preferences	14
3.1.2	Restaurant's menu	16
3.2	The outputs of the Knowledge Base	17
3.2.1	Wine List	18
4	Decision Tables	30
4.1	Decision Model and Notation	30
4.2	The DMN Diagram	31
4.2.1	The "Country preferences" table	32
4.2.2	The "Region preferences" table	33
4.2.3	The "Taste preferences" table	34
4.2.4	The "Grape preferences" table	35
4.2.5	The "Menu meal" table	36
4.2.6	The "Wine Selection" table	37
4.3	Camunda DMN-Simulator	38
5	Rule-Based Systems: Logic Programming and Prolog	39
5.1	Introduction	39
5.1.1	Rule-Based Systems	39
5.1.2	Logic programming	39
5.1.3	Usages	39
5.2	Prolog	40
5.2.1	Facts and Rules	40
5.2.2	SWISH	40
5.2.3	Examples	41

6	Knowledge Graph	43
6.1	Ontologies in Protégé	43
6.2	Class Hierarchy	43
6.3	Object Property Hierarchy	44
6.4	Data Property Hierarchy	45
6.5	Resulting Knowledge Graph	46
6.6	SPARQL Query Plugin and queries	47
6.6.1	Queries implemented	48
7	Graph Modeling Language	52
7.1	Introduction to a graphical modeling language	52
7.2	User and Library Management	52
7.3	Class hierarchy	53
7.3.1	Classes	53
7.3.2	Relation Classes	56
7.4	The special attribute GraphRep	57
7.4.1	GraphRep of Wine class	57
7.4.2	GraphRep of Meal class	59
7.4.3	GraphRep of Country, Region and Grape classes	59
7.5	Model result	60
8	Albanese Clarissa's Conclusions	61
8.1	Introduction	61
8.2	The aim	61
8.3	Advantages and disadvantages	61
8.3.1	Decision Tables	61
8.3.2	Prolog	62
8.3.3	Knowledge Graph	62
8.3.4	Graphical Modeling Language	63
8.4	Final Conclusion	63
9	Sorritelli Greta's Conclusions	64
9.1	Introduction	64
9.2	Implications and Applications	64
9.3	Advantages and Disadvantages of Each Task	64
9.3.1	Decision Tables	64
9.3.2	Prolog	65
9.3.3	Knowledge Graphs	65
9.3.4	Graphical Modeling Language	65
9.4	Final Conclusion	66
9.4.1	Further Work	66

Listings

6.1	Prefixes required in SPARQL Queries	47
6.2	SPARQL Query filtering grape preferred	48
6.3	SPARQL Query filtering grape to exclude	48
6.4	SPARQL Query filtering grape preferred and grape to exclude	48
6.5	SPARQL Query filtering country preferred	49
6.6	SPARQL Query filtering region preferred	49
6.7	SPARQL Query filtering pairing with the meal chosen	49
6.8	SPARQL Query filtering region, taste, grape preferred and grape to exclude	50
6.9	SPARQL Query filtering region, taste, grape preferred, grape to exclude and pairing with meal chosen	50
6.10	SPARQL Query filtering country, taste, grape preferred, grape to exclude and pairing with meal chosen	51
6.11	SPARQL Query showing all wines and their characteristics	51
7.1	Wine's GraphRep	58
7.2	Meal's GraphRep	59
7.3	Country's GraphRep	59
7.4	Region's GraphRep	59
7.5	Grape's GraphRep	59

List of Figures

2.1	Camunda Modeler Logo	9
2.2	Prolog Logo	10
2.3	Protégé Logo	11
2.4	ADOxx Logo	12
3.1	Masseto taste	18
3.2	Masseto food pairing	19
3.3	The Pastoralist taste	19
3.4	The Pastoralist food pairing	20
3.5	Clos De Los Siete taste	20
3.6	Clos De Los Siete food pairing	21
3.7	Bourgogne Pinot Noir taste	22
3.8	Bourgogne Pinot Noir food pairing	22
3.9	Fontanafredda Moscato Asti taste	23
3.10	Fontanafredda Moscato Asti food pairing	23
3.11	Mionetto Prosecco Brut taste	24
3.12	Mionetto Prosecco Brut food pairing	25
3.13	Montrachet Grand Cru taste	25
3.14	Montrachet Grand Cru food pairing	26
3.15	Coteau De Vernon taste	26
3.16	Coteau De Vernon food pairing	27
3.17	Malene Rosé taste	27
3.18	Malene Rosé food pairing	28
3.19	Whispering Angel Rosé taste	29
3.20	Whispering Angel Rosé food pairing	29
4.1	Decision Model and Notation Logo	30
4.2	DMN Diagram	31
4.3	Country preferences table	32
4.4	Region preferences table	33
4.5	Taste preferences table	34
4.6	Grape preferences table	35
4.7	Menu meal table	36
4.8	Wine Selection table	37
4.9	Simulation of DMN Diagram	38
5.1	Prolog Choice Menu	41
5.2	First example in Prolog	41
5.3	Second example in Prolog	42
5.4	Third example in Prolog	42

6.1	Class Hierarchy in Protégé	44
6.2	Object Property	45
6.3	Data Property	45
6.4	Knowledge Graph	46
6.5	SPARQL Query for listing all wines	47
7.1	User management	52
7.2	Library management	53
7.3	Wine attributes	53
7.4	Values of wine attributes	54
7.5	Wine attributes icons example	54
7.6	Meal type attribute	54
7.7	Meal attribute icons example	55
7.8	Country class example	55
7.9	Region class example	55
7.10	Grape class example	55
7.11	"Comes from" relation class example	56
7.12	"Is a part of" relation class example	56
7.13	"Is made of" relation class example	57
7.14	"Pairs with" relation class example	57
7.15	Model	60

1. Introduction

This introduction chapter will speak about the given Project Description and the Scope of the project itself.

1.1 Project Description

With COVID-19 many restaurants have their menus digitized. Guests can scan a QR code and have the menu presented on their smartphones. A disadvantage is that the screen is very small, and it is difficult to get an overview. This is especially true if a restaurant offers a big wine menu. Usually customers have preferences. Some prefer wine only from a specific country, e.g. Italy, or a specific region like Lombardy. Others exclude or prefer specific grapes, e.g. Some don't like Pinot Noir. Most people are not wine experts and would like to select their wine by describing the taste (i.e. dry/not-dry, tannin/less-tannin) but a very prominent decision influencer is the meal. Red wine usually is offered to meat dishes; white wine usually to fish. But there are exceptions, e.g. white wine with chicken.

The objective of the project is to represent the knowledge about wines. Menus and guest preferences are needed to support the selection process (i.e. they are input). The aim is to create a system that allows to select those wines that fit the guest preferences and the menu. The knowledge base contains information about typical wines (of an international restaurant) with wines from different regions and countries.

1.2 The Scope

The objective of this project is to build a knowledge-based system that represents information about wines typically found in international restaurants, with inputs and outputs. The system utilize input such as menus and guest preferences to support the wine selection process.

Three main knowledge-based solutions are implemented based on:

- Decision Tables with Camunda Modeler;
- Prolog Algorithm;
- Knowledge Graph in Protégé.

These solutions enable the system to provide tailored recommendations based on user input.

In addition, a graphical modeling language, through the ADOxx Metamodelling Platform, is designed to allow chefs to visually represent meals and wines, adding all relevant information required for customers to make informed choices based on their preferences.

2. Technologies

The purpose of this chapter is presenting the technologies and frameworks studied and used during the realization of the project, briefly introducing their functionalities and employment.

2.1 Camunda Modeler¹

Camunda Modeler is a powerful visual modeling tool used for designing and editing process models in the context of Business Process Model and Notation (BPMN), Case Management Model and Notation (CMMN) and Decision Model and Notation (DMN). It is a part of the Camunda BPM platform, which is an open-source workflow and decision automation framework.

In this project the Camunda Modeler was chosen because is a desktop application that can be installed and used locally, all while integrating the local development environment. Furthermore, Camunda Modeler provides a user-friendly interface that allows to create and modify process models easily.



Figure 2.1: Camunda Modeler Logo

With Camunda Modeler, it is possible to design complex workflows by defining user tasks, service tasks, script tasks, and other elements required to orchestrate processes. The tool allows to define process execution paths, decision points, and exception handling, enabling to create robust and flexible process models.

Overall, Camunda Modeler is a comprehensive and user-friendly tool for modeling and designing business processes. It empowers organizations to efficiently create, modify, and deploy process models, enabling them to achieve process automation and optimization goals effectively.

2.2 Camunda Simulator²

The Camunda DMN Simulator is a tool provided by Camunda's consulting division. It is designed to help users validate and test DMN decision models in a standalone

¹<https://camunda.com/platform/modeler/>

²<https://consulting.camunda.com/dmn-simulator/>

environment. The DMN Simulator allows interactive simulations of DMN decision tables, enabling users to observe and verify outcomes based on different input data.

The simulator offers a user-friendly interface for inputting test data and running simulations against DMN decision models. It facilitates exploration of different scenarios and variations, allowing users to understand decision table behavior and evaluate correctness and effectiveness.

The DMN Simulator is particularly useful during the design and testing phases of decision modeling processes. It provides insights into real-world performance without complex workflow configurations or BPMN processes.

2.3 Prolog Language

Prolog (Programming in Logic) is a declarative programming language based on formal logic and used for knowledge representation and logical inference.

In Prolog, programs are defined in terms of facts and rules that represent knowledge and logical relationships. The language follows a logical programming paradigm, where programs are written as a series of logical statements, called clauses, and the execution of a program involves logical inference and pattern matching.

The key components of Prolog programs are:

Facts: These are statements that define relationships between objects or describe properties of objects. Facts are typically written as predicates, which consist of a predicate name followed by a list of arguments.

Rules: Rules define logical relationships and can be used to derive new facts based on existing facts and logical conditions. Rules are written as logical implications using the `:-` operator.

Queries: Users can interact with a Prolog program by posing queries, which are logical statements that seek to find solutions or verify facts based on the defined knowledge. Prolog uses a process called backtracking to find all possible solutions to a query.

Prolog employs a unification mechanism to match queries with facts and rules, allowing for powerful pattern matching and logical inference. It supports logical operations such as conjunction (`;`) and disjunction (`;`) to combine multiple conditions in a query.



Figure 2.2: Prolog Logo

Prolog’s declarative nature and pattern matching capabilities make it well-suited for tasks that involve symbolic reasoning and knowledge-based systems. It allows for the efficient representation and manipulation of complex knowledge structures, making it a popular choice in various domains where logical reasoning and inference are crucial.

2.4 Protégé³

Protégé is a widely used open-source ontology development and knowledge management tool. It provides a user-friendly environment for creating, editing, and managing ontologies, which are formal representations of knowledge in a specific domain. Protégé is commonly used in the field of semantic web technologies, knowledge engineering, and artificial intelligence.

Key features of Protégé include:

Ontology Editing: Protégé offers a rich set of editing tools and capabilities for creating and modifying ontologies. Users can define classes, properties, and relationships to represent concepts and their interconnections within a domain.

Graphical User Interface (GUI): Protégé provides a user-friendly and intuitive GUI that enables users to navigate and visualize ontologies easily.

Reasoning and Inference: Protégé integrates with reasoners, which are computational engines capable of performing logical inference over ontologies. This allows users to validate the consistency of ontologies, detect logical errors, and derive new knowledge based on defined rules and axioms.

Collaboration and Versioning: Protégé supports collaboration among multiple users working on the same ontology project.

Plugin Architecture: Protégé offers a plugin architecture that allows users to extend its functionality.

Integration and Interoperability: Protégé supports importing and exporting ontologies in various standard formats and integration with semantic web technologies, for linking and sharing knowledge resources.



Figure 2.3: Protégé Logo

Protégé has a large user community and extensive documentation, making it a popular choice for ontology development and knowledge representation tasks. Its user-friendly interface, powerful reasoning capabilities, and extensibility make it a valuable tool.

³<https://protege.stanford.edu/>

2.5 ADOxx⁴

ADOxx is a comprehensive and extensible software development platform that focuses on modeling and executing business processes and information systems. It provides a set of tools and methodologies to design, implement, and manage complex enterprise solutions.

The ADOxx platform is based on the concept of **metamodeling**, which means that it allows users to define their own modeling languages and methodologies tailored to their specific needs. This flexibility enables organizations to create domain-specific modeling languages (DSMLs) that capture the unique aspects of their business processes and information systems.

ADOxx consists of:

- **ADOxx Development Toolkit** to define Modelling languages, for Library Management and administration of users, models and components.
- **ADOxx Modelling Toolkit** to create Models.

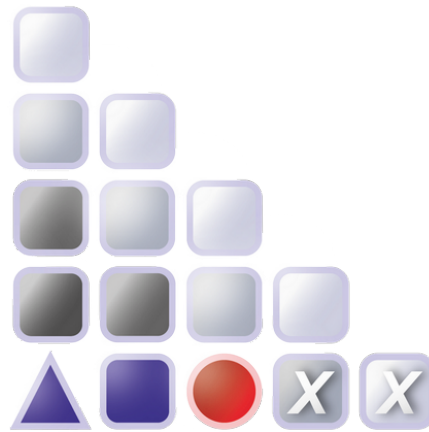


Figure 2.4: ADOxx Logo

Key components of ADOxx include:

MetaEdit+: A graphical modeling tool used to define and create DSMLs. It allows users to create models using predefined or customized graphical notations, define their semantics and constraints, and generate executable code from the models.

ADOxx Metamodel Repositor: A central repository that stores the definitions of DSMLs created using MetaEdit+. It provides a structured environment to manage and organize the modeling artifacts.

Model Execution Engine: A runtime environment that allows the execution of models created using ADOxx. It enables the simulation and execution of business processes, workflow management, and the automation of tasks based on the defined models.

Model Transformation Engine: ADOxx supports model transformations, allowing users to convert models from one DSML to another. This feature enables the integration of different modeling languages and the exchange of information between various modeling tools.

⁴<https://www.adoxx.org/live/home>

ADOxx Solution Builder: This tool assists in building complete solutions based on the models created using ADOxx. It supports the integration of external systems, database connectivity, and the development of user interfaces.

ADOxx promotes the **model-driven development (MDD)** approach, where models play a central role in software development. By using models as the primary artifacts, organizations can improve the efficiency, maintainability, and reusability of their software systems.

ADOxx is often used in domains such as **business process management (BPM)**, **enterprise architecture (EA)**, information systems development, and other areas where customized modeling languages and process automation are required.

Overall, ADOxx provides a powerful platform for creating and managing domain-specific modeling languages, facilitating the design and implementation of complex enterprise solutions, and supporting model-driven development practices.

3. Knowledge Base

In this chapter we focus on the Knowledge Base. A knowledge base is a structured collection of information, facts, rules, and concepts that is organized and stored for easy access and retrieval. It serves as a repository of knowledge and acts as a central source of information for an individual, organization, or system.

A knowledge base contains knowledge about a specific domain or subject area, and its purpose is to capture, organize, and represent that knowledge in a way that can be utilized for various purposes, such as decision making, problem solving, and information retrieval. It can include both explicit knowledge, which is information that is easily codified and documented, and tacit knowledge, which is more experiential and context-specific.

The information in a knowledge base is typically structured and organized using a knowledge representation language or framework, such as ontologies or semantic networks. This allows for logical relationships and connections to be defined between different pieces of knowledge, enabling efficient retrieval and inference.

3.1 The inputs of the Knowledge Base

The inputs of the Knowledge Base are the **restaurant's menu** and the **guest preferences** and they are needed to support the selection process.

The menu has been chosen to standardize the dishes based on five categories of food: red meat, fish, chicken, vegetables, and desserts.

The guest preferences, on the other hand, are divided into country, region, taste and grapes. The guest can express preferences regarding the wine that will accompany his dish, one or more.

In the knowledge base there are also exceptions in terms of guest preferences that are described in food pairing between dishes and wines.

The knowledge-based system will utilize these inputs to recommend wines that align with the guest's preferences and the menu, taking into account factors like country and region preference, grape preference, taste preferences, and the **compatibility of wines with different meal categories**.

3.1.1 Guest preferences

Country

To ensure a consistent selection of wines based on customer preferences regarding country, the following have been chosen:

1. France;
2. Italy;

3. Argentina;
4. United States;
5. Australia.

Region

The guest can also express a preference regarding the region to which the chosen wine belongs. Based on the selected countries and the search for suitable wines, the selected regions are:

- Tuscany
- Burgundy;
- Provence;
- Piedmont;
- Veneto;
- Tuscany;
- Mendoza;
- California,
- South Australia.

Grape

Each wine selected for the knowledge base is composed by one or more grape varieties. Every type of grape is produced in a specific country. Here are the selected grapes:

- Merlot (France, Italy);
- Cabernet Sauvignon (France);
- Malbec (France);
- Petit Verdot (France);
- Cabernet Franc (France);
- Pinot Noir (France);
- Moscato (Italy);
- Glera (Italy);
- Chardonnay (France);
- Viognier (France, Croatia);
- Grenache (Spain);
- Rolle (Italy);

- Syrah (France);
- Cinsault (France);
- Tibouren (France).

Taste

The five taste commonly found in wine, which the guest can choose as his preferences are:

1. acid;
2. bitter;
3. spice;
4. sweet;
5. salt.

3.1.2 Restaurant's menu

The menu chosen for the International Restaurant includes dishes that are commonly found in this type of restaurant and has been divided into five categories of food: red meat, fish, chicken, vegetables and desserts. There are six dishes, one for red meat, one for fish, one for chicken, one for vegetables and two for desserts:

- **Beef Bourguignon** - *Red meat*: A classic French dish consisting of tender beef braised in red wine, with mushrooms, onions, and aromatic herbs. It is typically served with mashed potatoes or crusty bread.
- **Fish and Chips** - *Fish*: A popular British dish featuring battered and deep-fried white fish (typically cod or haddock) served with thick-cut potato chips (fries) and tartar sauce. It is often enjoyed with mushy peas.
- **Chicken Caesar Salad** - *Chicken*: A classic salad featuring grilled or roasted chicken, crisp romaine lettuce, Parmesan cheese, croutons, and Caesar dressing. It is a popular choice for a lighter chicken meal.
- **Ratatouille** - *Vegetables*: A traditional French Provençal dish made with a medley of sautéed vegetables such as eggplant, zucchini, bell peppers, and tomatoes, seasoned with herbs like thyme and basil.
- **Tiramisù** - *Desserts*: An Italian classic made with layers of ladyfingers dipped in coffee and layered with a creamy mixture of mascarpone cheese, eggs, and sugar. It is often dusted with cocoa powder and served chilled.
- **Fresh fruit Salad** - *Desserts*: Fresh fruit salads are versatile desserts found in many restaurants worldwide. They typically feature a medley of seasonal fruits like watermelon, cantaloupe, pineapple, grapes, berries, and citrus segments.

3.2 The outputs of the Knowledge Base

As output of the knowledge-based system there are the **recommended wines**, based on the guest preferences and the restaurant menu, among a list of five types of wines, that are been chosen to ensure a well-rounded selection, as these are renowned wines commonly found in international restaurants: *four red wines, four white wines and two rose wine*:

- **Masseto:** Masseto is a highly acclaimed Italian red wine from the Tuscany region. It is made exclusively from Merlot grapes and is known for its exceptional quality, concentration, and elegance. Masseto offers rich flavors of dark fruits, spice, chocolate, and velvety tannins, with a long and harmonious finish.
- **The Pastoralist:** The Pastoralist is a red wine produced by the d'Arenberg winery in McLaren Vale, Australia. It is a blend of Grenache, Syrah, and Mourvèdre grapes. The wine showcases vibrant fruit flavors, such as blackberries and cherries, along with hints of spice and earthiness. It has a medium to full body, smooth tannins, and a lingering finish.
- **Clos De Los Siete:** Clos De Los Siete is a red wine blend from the Mendoza region of Argentina. It is produced by a group of renowned winemakers, and the blend typically includes Malbec, Merlot, Cabernet Sauvignon, Syrah, and Petit Verdot. The wine offers ripe fruit flavors, well-integrated tannins, and a harmonious balance between fruitiness, structure, and acidity.
- **Bourgogne Pinot Noir:** Bourgogne Pinot Noir refers to Pinot Noir wines from the Burgundy region of France. These wines showcase the classic characteristics of the Pinot Noir grape, with delicate aromas of red berries, cherries, and floral notes. They often exhibit a medium body, silky tannins, and a vibrant acidity, making them versatile and food-friendly.
- **Fontanafredda Moscato Asti:** Fontanafredda Moscato Asti is a sparkling wine from the Asti region of Piedmont, Italy. It is made from the Moscato Bianco grape and is known for its aromatic profile and sweetness. This Moscato Asti offers intense aromas of ripe peaches, apricots, and orange blossoms, with a light and refreshing effervescence.
- **Mionetto Prosecco Brut:** Mionetto Prosecco Brut is a sparkling wine from the Prosecco region of Veneto, Italy. It is made from the Glera grape and is known for its lively bubbles and crisp, dry character. This Prosecco offers flavors of green apple, citrus, and floral notes, with a refreshing acidity and a clean, effervescent finish.
- **Montrachet Grand Cru:** Montrachet Grand Cru is one of the most prestigious white wines from the Burgundy region of France. It is made from Chardonnay grapes and is renowned for its complexity, richness, and age-worthiness. Montrachet Grand Cru offers aromas of ripe fruits, citrus, honey, and toasted nuts, with a full body, creamy texture, and a long, nuanced finish.
- **Coteau De Vernon:** Coteau De Vernon is a white wine produced in the Chablis region of Condrieu, France. It is made from Chardonnay grapes grown on the renowned Coteau de Vernon vineyard. This wine exhibits fresh citrus flavors, minerality, and a crisp acidity, with a clean and vibrant finish.

- **Malene Rosé:** Malene Rosé is a premium rosé wine produced in California, United States. It is crafted in the traditional Provençal style, showcasing bright fruit flavors, floral notes, and a refreshing acidity. Malene Rosé offers a pale salmon color, delicate aromatics, and a crisp, dry finish.
- **Whispering Angel Rosé:** Whispering Angel is a popular rosé wine produced in the Côtes de Provence region of France. It is known for its pale pink color, delicate aromas of red berries and flowers, and a dry, crisp, and refreshing palate. Whispering Angel has gained a reputation as a quintessential Provence rosé, loved for its easy-drinking style.

3.2.1 Wine List

In this subsection there are more details about every wine selected in terms of grape, country, region, taste and food pairing.

Masseto

Masseto is a **red wine** grape variety.

Grape: Merlot;

Country: Italy;

Region: Tuscany.

Taste:

Sweetness: Masseto is not typically considered a sweet wine. It is a dry red wine, meaning it contains minimal residual sugar.

Acidity: Masseto tends to have a good level of acidity, which provides a refreshing and vibrant quality to the wine.

Saltiness: Saltiness is not a prominent characteristic in Masseto or most red wines. It is not typically associated with this wine style.

Bitterness: Masseto generally has low to moderate levels of bitterness.

Spice: Masseto can exhibit subtle spice notes, often in the form of warm, aromatic spices such as cloves, cinnamon, or nutmeg.

Sweetness	Acidity	Saltiness	Bitterness	Spice
×	✓	×	✓	✓

Figure 3.1: Masseto taste

Food pairing:

Red Meat: Masseto pairs exceptionally well with red meat dishes.

Fish: While Masseto is typically associated with red meat, it can also be paired with certain types of fish. Meatier fish like tuna or swordfish, which can stand up to the wine's richness.

Chicken: Masseto can be paired with flavorful, well-seasoned chicken dishes. Roasted or grilled chicken with herbs and spices, such as rosemary or thyme, can complement the wine nicely.

Vegetables: Masseto's bold character allows it to pair well with hearty, flavorful vegetables. Grilled or roasted vegetables like eggplant, bell peppers, or portobello mushrooms can work beautifully.

Dessert: Masseto is typically enjoyed on its own or with savory dishes rather than dessert. Due to its full-bodied and tannic nature, it may not pair well with most sweet desserts.

Red Meat	Fish	Chicken	Vegetables	Dessert
✓	✓	✓	✓	✗

Figure 3.2: Masseto food pairing

The Pastoralist

The Pastoralist is a **red wine** grape variety.

Grape: Cabernet Sauvignon;

Country: Australia;

Region: South Australia.

Taste:

Sweetness: The sweetness level in The Pastoralist is generally low. It is a dry red wine that emphasizes fruit flavors and structure rather than sweetness.

Acidity: The Pastoralist typically exhibits a moderate to high level of acidity.

Saltiness: Saltiness is not a significant characteristic in The Pastoralist or most red wines.

Bitterness: The bitterness level in The Pastoralist is usually moderate.

Spice: The Pastoralist may showcase subtle spice notes that can vary depending on the specific blend and vintage.

Sweetness	Acidity	Saltiness	Bitterness	Spice
✗	✓	✗	✓	✓

Figure 3.3: The Pastoralist taste

Food pairing:

Red Meat: The robust and full-bodied nature of The Pastoralist makes it an excellent match for red meat dishes.

Fish: While The Pastoralist is primarily associated with red meat, it can also pair well with certain types of fish. Meatier fish like salmon or tuna, prepared with flavorful sauces or spices.

Chicken: The Pastoralist can pair well with well-seasoned and flavorful chicken dishes.

Vegetables: The Pastoralist's rich and robust profile allows it to match well with hearty vegetables. Grilled or roasted vegetables like eggplant, bell peppers, or Portobello mushrooms can work nicely.

Dessert: The Pastoralist's bold and tannic nature makes it less suitable for pairing with sweet desserts.

Red Meat	Fish	Chicken	Vegetables	Dessert
✓	✓	✓	✓	✗

Figure 3.4: The Pastoralist food pairing

Clos De Los Siete

Clos De Los Siete is a **red wine** grape variety.

Grape: Malbec, Merlot, Cabernet Sauvignon, Syrah, Petit Verdot, Cabernet Franc;

Country: Argentina;

Region: Mendoza.

Taste:

Sweetness: Clos De Los Siete is generally a dry red wine, meaning it contains minimal residual sugar. While it may have some perception of sweetness due to ripe fruit flavors, it is not typically considered a sweet wine.

Acidity: Clos De Los Siete often displays a moderate to high level of acidity.

Saltiness: Saltiness is not a prominent characteristic in Clos De Los Siete or most red wines. It is not typically associated with this wine style.

Bitterness: Clos De Los Siete may have a moderate level of bitterness, primarily stemming from the presence of tannins.

Spice: Clos De Los Siete can exhibit subtle spice notes.

Sweetness	Acidity	Saltiness	Bitterness	Spice
✗	✓	✗	✓	✓

Figure 3.5: Clos De Los Siete taste

Food pairing:

Red Meat: Clos De Los Siete pairs excellently with red meat dishes. Grilled or roasted cuts of beef, such as steak or prime rib, are particularly complementary to the wine's rich flavors.

Fish: While Clos De Los Siete is generally better suited for red meat, it can be paired with fish of hearty and flavorful varieties.

Chicken: Clos De Los Siete can work well with grilled or roasted chicken dishes, especially if they are prepared with spices or herbs that add complexity and depth.

Vegetables: With its bold flavor profile, Clos De Los Siete can handle strongly flavored vegetables. The wine's tannins and fruitiness can complement the earthy and savory qualities of the vegetables.

Dessert: Clos De Los Siete is not typically recommended for pairing with desserts due to its full-bodied nature.

Red Meat	Fish	Chicken	Vegetables	Dessert
✓	✓	✓	✓	✗

Figure 3.6: Clos De Los Siete food pairing

Bourgogne Pinot Noir

Bourgogne Pinot Noir is a **red wine** grape variety.

Grape: Pinot Noir;

Country: France;

Region: Burgundy.

Taste:

Sweetness: Bourgogne Pinot Noir is generally a dry red wine. It has minimal residual sugar, and any perceived sweetness typically comes from the ripe fruit flavors rather than actual sugar content.

Acidity: Bourgogne Pinot Noir is known for its vibrant acidity.

Saltiness: Saltiness is not a prominent characteristic in Bourgogne Pinot Noir.

Bitterness: Bourgogne Pinot Noir may exhibit a moderate level of bitterness, which often comes from the presence of fine-grained tannins.

Spice: Bourgogne Pinot Noir can display subtle spice notes, which can include hints of clove, cinnamon, or other delicate spices.

Sweetness	Acidity	Saltiness	Bitterness	Spice
×	✓	×	✓	✓

Figure 3.7: Bourgogne Pinot Noir taste

Food pairing:

Red Meat: Bourgogne Pinot Noir pairs well with red meat dishes such as roasted beef, grilled lamb, or seared duck breast.

Fish: Bourgogne Pinot Noir can be paired with richer fish like salmon or tuna. It's best to choose fish dishes that have some depth of flavor, such as grilled or roasted preparations, as the wine's fruitiness can stand up to them.

Chicken: Bourgogne Pinot Noir can be enjoyed with roasted or grilled chicken, especially when served with mushroom-based sauces or herbs like thyme and rosemary.

Vegetables: This wine can also pair well with vegetable-based dishes. Try serving it with roasted or grilled vegetables like mushrooms, eggplant, or beets. The earthy notes of the wine complement the flavors of the vegetables.

Dessert: Bourgogne Pinot Noir is not typically paired with desserts, as it is more commonly enjoyed on its own or with savory dishes.

Red Meat	Fish	Chicken	Vegetables	Dessert
✓	✓	✓	✓	×

Figure 3.8: Bourgogne Pinot Noir food pairing

Fontanafredda Moscato Asti

Fontanafredda Moscato Asti is a **white wine** grape variety.

Grape: Moscato;

Country: Italy;

Region: Piedmont.

Taste:

Sweetness: Fontanafredda Moscato Asti is a sweet wine. It is known for its pronounced sweetness, derived from the natural sugars present in the Moscato Bianco grapes used to produce it. The wine is often enjoyed for its dessert-like sweetness.

Acidity: Despite its sweetness, Fontanafredda Moscato Asti retains a balancing acidity. The wine usually possesses a refreshing and lively acidity, which helps to counterbalance the sweetness and provides a pleasant crispness on the palate.

Saltiness: Saltiness is not a prominent characteristic in Fontanafredda Moscato Asti or most sweet wines.

Bitterness: Fontanafredda Moscato Asti generally has low levels of bitterness. The wine's focus is on the fruitiness and sweetness, rather than bitter elements. Any bitterness present is usually subtle and well-integrated.

Spice: Spice notes are not a typical characteristic of Fontanafredda Moscato Asti. Instead, the wine is known for its pronounced floral and fruity aromas, often displaying intense notes of peach, apricot, and orange blossom.

Sweetness	Acidity	Saltiness	Bitterness	Spice
✓	✓	✗	✗	✗

Figure 3.9: Fontanafredda Moscato Asti taste

Food pairing:

Red Meat: Since Moscato Asti is sweet and fruity, it might not be the ideal match for red meat, which tends to be rich and savory.

Fish: Moscato Asti can work well with light and delicate fish dishes. Grilled or baked fish such as sole, trout, or snapper can be complemented by the wine's fruity flavors.

Chicken: Moscato Asti can pair nicely with chicken dishes, especially those with fruity or mildly spiced flavors. Lemon herb chicken, apricot-glazed chicken, or grilled chicken with mango salsa can create a harmonious balance with the wine.

Vegetables: Moscato Asti can be a refreshing companion to vegetable-focused dishes. It can work well with roasted or grilled vegetables like zucchini, bell peppers, or eggplant.

Dessert: Moscato Asti is often enjoyed as a dessert wine, and it pairs exceptionally well with sweet treats. It can be a fantastic companion to fruity desserts such as peach cobbler, mixed berry tart, or fruit salads.

Red Meat	Fish	Chicken	Vegetables	Dessert
✗	✓	✓	✓	✓

Figure 3.10: Fontanafredda Moscato Asti food pairing

Mionetto Prosecco Brut

Mionetto Prosecco Brut is a **white wine** grape variety.

Grape: Glera;

Country: Italy;

Region: Veneto.

Taste:

Sweetness: Mionetto Prosecco Brut is a dry wine. It is typically characterized by its low residual sugar content, resulting in a crisp and refreshing taste profile. It is not sweet in nature, but rather showcases the natural fruit flavors without added sweetness.

Acidity: Mionetto Prosecco Brut is known for its vibrant acidity.

Saltiness: Saltiness is not a prominent characteristic in Mionetto Prosecco Brut or most sparkling wines.

Bitterness: Mionetto Prosecco Brut generally has low levels of bitterness. The focus of this wine is on its vibrant fruit flavors and lively acidity, with minimal presence of bitter elements.

Spice: Spice notes are not a typical characteristic of Mionetto Prosecco Brut. Instead, it is known for its bright and fresh fruit flavors, often showcasing citrus, green apple, and floral notes.

Sweetness	Acidity	Saltiness	Bitterness	Spice
×	✓	×	×	✓

Figure 3.11: Mionetto Prosecco Brut taste

Food pairing:

Red Meat: Mionetto Prosecco Brut is not typically the first choice for red meat pairings, as it is better suited for lighter dishes.

Fish: Mionetto Prosecco Brut pairs well with a range of seafood dishes. The wine's acidity and crispness complement the delicate flavors of the fish. Mionetto Prosecco Brut is a great match for chicken dishes, especially when they are prepared with lighter sauces or herbs.

Chicken: Mionetto Prosecco Brut is a great match for chicken dishes, especially when they are prepared with lighter sauces or herbs. It pairs well with grilled or roasted chicken breasts, chicken piccata, or chicken salads.

Vegetables: Mionetto Prosecco Brut is versatile enough to pair with a variety of vegetable-based dishes. It complements fresh salads, vegetable stir-fries, or roasted vegetables.

Dessert: Mionetto Prosecco Brut is often enjoyed as a standalone dessert wine due to its sweetness level and effervescence.

Red Meat	Fish	Chicken	Vegetables	Dessert
✗	✓	✓	✓	✓

Figure 3.12: Mionetto Prosecco Brut food pairing

Montrachet Grand Cru

Montrachet Grand Cru is a **white wine** grape variety.

Grape: Chardonnay;

Country: France;

Region: Burgundy.

Taste:

Sweetness: Montrachet Grand Cru is generally a dry white wine. It is known for its elegant and complex flavor profile, which emphasizes the expression of the terroir and the purity of the Chardonnay grape. It is not typically considered a sweet wine, although it may have subtle fruit sweetness.

Acidity: Montrachet Grand Cru typically possesses a vibrant and well-balanced acidity.

Saltiness: Saltiness is not a prominent characteristic in Montrachet Grand Cru or most white wines.

Bitterness: Montrachet Grand Cru may have a moderate level of bitterness. This bitterness can come from the presence of grape skins during fermentation or from the aging process in oak barrels.

Spice: Montrachet Grand Cru can exhibit subtle spice notes, particularly when oak aging is involved. The wine may showcase hints of vanilla, baking spices, or toastiness derived from the oak barrels. These spice elements add complexity and depth to the wine's flavor profile.

Sweetness	Acidity	Saltiness	Bitterness	Spice
✗	✓	✗	✓	✓

Figure 3.13: Montrachet Grand Cru taste

Food pairing:

Red Meat: Montrachet Grand Cru is not commonly paired with red meat.

Fish: Montrachet Grand Cru pairs well with rich and fatty fish such as salmon or tuna.

Chicken: Roast chicken with a creamy sauce or a buttery herb rub can be a good match for Montrachet Grand Cru. The wine's richness can complement the succulent flavors of the chicken.

Vegetables: This wine can be paired with various vegetable dishes, particularly those with creamy or buttery components.

Dessert: Montrachet Grand Cru can be enjoyed with certain desserts that are not overly sweet.

Red Meat	Fish	Chicken	Vegetables	Dessert
✗	✓	✓	✓	✓

Figure 3.14: Montrachet Grand Cru food pairing

Coteau De Vernon

Coteau De Vernon is a **white wine** grape variety.

Grape: Viognier;

Country: France;

Region: Burgundy.

Taste:

Sweetness: Coteau De Vernon is generally a dry white wine. It is known for its crisp and mineral-driven style, with minimal residual sugar.

Acidity: Coteau De Vernon is known for its vibrant and refreshing acidity.

Saltiness: Saltiness is not a prominent characteristic in Coteau De Vernon or most white wines.

Bitterness: Coteau De Vernon may exhibit a moderate level of bitterness, which can come from elements such as grape skins or the aging process.

Spice: Spice notes are not a prominent characteristic of Coteau De Vernon. Instead, the wine is known for its mineral-driven flavors, often showcasing nuances of flint, chalk, or wet stones. These mineral elements contribute to the wine's overall complexity and sense of terroir expression.

Sweetness	Acidity	Saltiness	Bitterness	Spice
✗	✓	✗	✓	✗

Figure 3.15: Coteau De Vernon taste

Food pairing:

Red Meat: Coteau de Vernon is typically not the first choice for pairing with red meat due to its light and delicate nature.

Fish: Coteau de Vernon is an excellent choice for seafood dishes. It pairs well with delicate and mild-flavored fish like sole, sea bass, or cod.

Chicken: Roast chicken or chicken breasts prepared with herbs and spices can be paired nicely with Coteau de Vernon. The wine's crisp acidity can cut through the richness of the chicken and enhance its flavors.

Vegetables: This wine complements vegetable dishes that have a subtle and delicate flavor profile. Think of lightly steamed or roasted vegetables like asparagus, green beans, or zucchini.

Dessert: Coteau de Vernon is not typically recommended for pairing with sweet desserts due to its dry and crisp nature.

Red Meat	Fish	Chicken	Vegetables	Dessert
×	✓	✓	✓	×

Figure 3.16: Coteau De Vernon food pairing

Malene Rosé

Malene Rosé is a **rose wine** grape variety.

Grape: Grenache;

Country: United States;

Region: California.

Taste:

Sweetness: Malene Rosé is generally a dry wine with low to medium levels of residual sugar. It is crafted to be refreshing and crisp, focusing more on vibrant fruit flavors rather than sweetness.

Acidity: Malene Rosé is known for its bright and lively acidity.

Saltiness: Saltiness is not a prominent characteristic in Malene Rosé or most rosé wines.

Bitterness: Malene Rosé typically has low levels of bitterness. The wine is crafted to emphasize its fruit-forward nature and smooth texture, with minimal presence of bitter elements.

Spice: Spice notes are not a dominant characteristic of Malene Rosé. Instead, the wine showcases vibrant fruit flavors, often highlighting notes of strawberries, raspberries, and citrus fruits. The focus is on its refreshing and fruit-forward profile rather than spice-driven complexity.

Sweetness	Acidity	Saltiness	Bitterness	Spice
×	✓	×	×	×

Figure 3.17: Malene Rosé taste

Food pairing:

Red Meat: Malene Rosé pairs well with lighter red meats, such as grilled or roasted lamb chops or tenderloin. The wine's acidity and fruity flavors complement the richness of the meat.

Fish: This rosé is a great match for a variety of fish and seafood dishes. It goes well with grilled or baked salmon, shrimp, or scallops.

Chicken: Malene Rosé can be paired with a range of chicken dishes. It complements grilled or roasted chicken breasts, chicken kebabs, or herb-marinated chicken thighs.

Vegetables: The vibrant and fresh nature of Malene Rosé makes it a great choice for vegetable-focused dishes. It pairs well with grilled or roasted vegetables like bell peppers, zucchini, or eggplant.

Dessert: While Malene Rosé is typically enjoyed as a refreshing aperitif or with savory dishes, it can also be paired with certain desserts, such as a mixed berry tart, strawberry shortcake, or a fruit salad with a touch of citrus zest.

Red Meat	Fish	Chicken	Vegetables	Dessert
✓	✓	✓	✓	✓

Figure 3.18: Malene Rosé food pairing

Whispering Angel Rosé

Whispering Angel Rosé is a **rose wine** grape variety.

Grape: Grenache, Rolle, Syrah, Cinsault, Tibouren;

Country: France;

Region: Provence.

Taste:

Sweetness: Whispering Angel Rosé is generally a dry wine. It is crafted to be elegant and crisp, with minimal residual sugar. While it may have a perception of fruit sweetness, it is not typically considered a sweet wine.

Acidity: Whispering Angel Rosé is known for its bright and refreshing acidity.

Saltiness: Saltiness is not a prominent characteristic in Whispering Angel Rosé or most rosé wines.

Bitterness: Whispering Angel Rosé typically has low levels of bitterness. The focus is on its delicate fruit flavors and smooth texture, with minimal presence of bitter elements.

Spice: Spice notes are not a dominant characteristic of Whispering Angel Rosé. Instead, the wine showcases bright and crisp fruit flavors, often including notes of strawberries, raspberries, and citrus fruits. The emphasis is on the wine's refreshing and fruit-forward profile rather than spice-driven complexity.

Sweetness	Acidity	Saltiness	Bitterness	Spice
×	✓	×	×	×

Figure 3.19: Whispering Angel Rosé taste

Food pairing:

Red Meat: While rosé wines are generally more suitable for light meats, Whispering Angel Rosé can be paired with leaner cuts of red meat such as grilled or roasted pork tenderloin.

Fish: Whispering Angel Rosé pairs well with a variety of fish and seafood dishes. It goes particularly well with grilled or roasted salmon, trout, or sea bass.

Chicken: This rosé wine is an excellent companion for chicken dishes, especially when they are prepared with lighter flavors.

Vegetables: Whispering Angel Rosé is versatile enough to pair with a wide range of vegetable dishes. It goes well with grilled vegetables, summer salads, and vegetable-based pasta dishes.

Dessert: While Whispering Angel is typically enjoyed as an aperitif or with savory dishes, it can also be paired with light and fruity desserts.

Red Meat	Fish	Chicken	Vegetables	Dessert
✓	✓	✓	✓	✓

Figure 3.20: Whispering Angel Rosé food pairing

4. Decision Tables

This chapter describes the development of a knowledge-based solution based on DMN and decision tables created using the Camunda Modeler tool (Chapter 2.1).

4.1 Decision Model and Notation

Decision Model and Notation (DMN) is a standard administered by the Object Management Group (OMG), a global consortium dedicated to standardizing the languages behind different process management systems and it has been widely adopted across various industries. Businesses leverage DMN to design decision models that are used for automation of the decision-making processes. DMN serves as a common language to align business and IT on repeatable business rules and decision management. The notation enhances business efficiency, reduces the risk of human error, and ensures that decision models are interchangeable across the organization.



Figure 4.1: Decision Model and Notation Logo

DMN is one of three important languages for business process modeling. Working alongside **Business Process Modeling Notation (BPMN)** and **Case Management Model and Notation (CMMN)**.

While BPMN, CMMN and DMN can be used independently, they were carefully designed to be complementary. Indeed, many organizations require a combination of process models for their prescriptive workflows, case models for their reactive activities, and decision models for their more complex, multi-criteria business rules. Those organizations will benefit from using the three standards in combination, selecting which one is most appropriate to each type of activity modeling. This is why BPMN, CMMN and DMN really constitute the “triple crown” of process improvement standards.

4.2 The DMN Diagram

In this project we focus on the DMN Diagram and the decision tables. For the knowledge base we tried to create a DMN that could be implemented in a realistic or hypothetical scenario of an application through which the customer of a restaurant can view a list of suitable wines based on the choice of the menu and his preferences. The interface of this application could consist of a series of fields to be entered, which represent preferences and dish of the chosen menu and one last button to generate the filtered menu based on the inserted fields. The starting point was to identify the list of all the customer's preferences and dishes, which the customer can choose and enter, subsequently calculating the proper wine list.

The DMN Diagram of our project consists of six decision tables: "Country preferences", "Region preferences", "Taste preferences", "Grape preferences", "Menu meal" and "Wine Selection". There are also a list of wines as Knowledge Source and five inputs data: "Country", "Region", "Taste", "Grape" and "Menu".

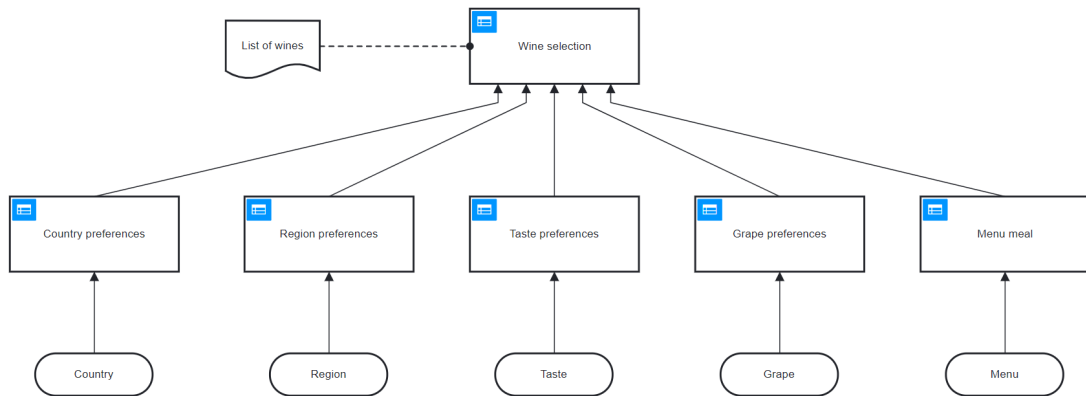


Figure 4.2: DMN Diagram

4.2.1 The "Country preferences" table

The "Country preferences" table (Figure 4.3) takes as input one string parameters, that is "Country" and returns as output the name of the single wine that matches the country preference, which therefore represents the country of origin of the wine, in string form.

Applying the hit policy "Collect" the wine list is obtained based on the guest's country preferences. With hit policy collect, it doesn't matter the order or any interdependencies between the rules. The output of the "Country preferences" table represents one of the inputs that is passed to the "Wine Selection" table.

Furthermore, cases in which the customer does not enter the country preference or enter another word than those considered in the program are managed and no output wines will be obtained.

Country preferences		Hit Policy: Collect ▼	
	When	Then	
	Country	Recommended Wines on Country	Annotations
	"Italy","Australia","Argentina","Fr...	"Mionetto Prosecco Brut","Font...	
1	"Italy"	"Mionetto Prosecco Brut"	
2	"Italy"	"Fontanafredda Moscato Asti"	
3	"Italy"	"Masseto"	
4	"France"	"Montrachet Grand Cru"	
5	"France"	"Coteau De Vernon"	
6	"France"	"Whispering Angel Rosé"	
7	"France"	"Bourgogne Pinot Noir"	
8	"Australia"	"The Pastoralist"	
9	"Argentina"	"Clos De Los Siete"	
10	"United States"	"Malene Rosé"	
11	""	"Masseto"	
12	""	"The Pastoralist"	
13	""	"Clos De Los Siete"	
14	""	"Bourgogne Pinot Noir"	
15	""	"Fontanafredda Moscato Asti"	

Figure 4.3: Country preferences table

4.2.2 The "Region preferences" table

The "Region preferences" table (Figure 4.4) takes as input one string parameters, that is "Region" and returns as output the name of the single wine that matches the region preference, which therefore represents the region of origin of the wine, in string form.

Applying the hit policy "Collect" the wine list is obtained based on the guest's region preferences. The output of the "Region preferences" table represents one of the inputs that is passed to the "Wine Selection" table.

Furthermore, cases in which the customer does not enter the region preference or enter another word than those considered in the program are managed and no output wines will be obtained.

Region preferences			
		Hit Policy: Collect ▼	
	When	Then	
	Region	Recommended wines on region	Annotations
	"Tuscany","Veneto","Piedmont", ...	"Masseto","The Pastoralist","Clo...	
1	"Tuscany"	"Masseto"	
2	"Veneto"	"Mionetto Prosecco Brut"	
3	"Piedmont"	"Fontanafredda Moscato Asti"	
4	"Burgundy"	"Montrachet Grand Cru"	
5	"Burgundy"	"Coteau De Vernon"	
6	"Burgundy"	"Bourgogne Pinot Noir"	
7	"Provence"	"Whispering Angel Rosé"	
8	"South Australia"	"The Pastoralist"	
9	"California"	"Malene Rosé"	
10	"Mendoza"	"Clos De Los Siete"	
11	""	"Masseto"	
12	""	"The Pastoralist"	
13	""	"Clos De Los Siete"	
14	""	"Bourgogne Pinot Noir"	
15	""	"Fontanafredda Moscato Asti"	

Figure 4.4: Region preferences table

4.2.3 The "Taste preferences" table

The "Taste preferences" table (Figure 4.5) takes as input one string parameters, that is "Taste" and returns as output the name of the single wine that matches the taste preference, which therefore represents the taste of the wine, in string form.

Applying the hit policy "Collect" the wine list is obtained based on the guest's taste preferences. The output of the "Taste preferences" table represents one of the inputs that is passed to the "Wine Selection" table.

Furthermore, cases in which the customer does not enter the taste preference or enter another word than those considered in the program are managed and no output wines will be obtained.

Taste preferences			
		Hit Policy:	Collect ▼
	When	Then	
	Taste	Recommended wines on taste	Annotations
	"acid","bitter","sweet","salt","spi...	"Masseto","The Pastoralist","Clo...	
1	"acid"	"Masseto"	
2	"acid"	"The Pastoralist"	
3	"acid"	"Clos De Los Siete"	
4	"acid"	"Bourgogne Pinot Noir"	
5	"acid"	"Fontanafredda Moscato Asti"	
6	"acid"	"Mionetto Prosecco Brut"	
7	"acid"	"Montrachet Grand Cru"	
8	"acid"	"Coteau De Vernon"	
9	"acid"	"Malene Rosé"	
10	"acid"	"Whispering Angel Rosé"	
11	"bitter"	"Masseto"	
12	"bitter"	"The Pastoralist"	
13	"bitter"	"Clos De Los Siete"	
14	"bitter"	"Bourgogne Pinot Noir"	
15	"bitter"	"Montrachet Grand Cru"	

Figure 4.5: Taste preferences table

4.2.4 The "Grape preferences" table

The "Grape preferences" table (Figure 4.6) takes as input one string parameters, that is "Grape" and returns as output the name of the single wine that matches the grape preference, which therefore represents the grape with which it is made, in string form.

Applying the hit policy "Collect" the wine list is obtained based on the guest's grape preferences. The output of the "Grape preferences" table represents one of the inputs that is passed to the "Wine Selection" table.

Furthermore, cases in which the customer does not enter the grape preference or enter another word than those considered in the program are managed and no output wines will be obtained.

Grape preferences			
		Hit Policy: Collect	
	When	Then	
	Grape	Recommend wines on grapes	Annotations
	"Merlot","Cabernet Sauvignon", ...	"Masseto","The Pastoralist","Clo...	
1	"Merlot"	"Masseto"	
2	"Cabernet Sauvignon"	"The Pastoralist"	
3	"Merlot", "Cabernet Sauvignon", "Malbec", "Syrah", "Petit Verdot", "Cabernet Franc"	"Clos De Los Siete"	
4	"Pinot Noir"	"Bourgogne Pinot Noir"	
5	"Moscato"	"Fontanafredda Moscato Asti"	
6	"Glera"	"Mionetto Prosecco Brut"	
7	"Chardonnay"	"Montrachet Grand Cru"	
8	"Viognier"	"Coteau De Vernon"	
9	"Grenache"	"Malene Rosé"	
10	"Grenache","Rolle","Syrah","Cinsault","Tibouren"	"Whispering Angel Rosé"	
11	""	"Masseto"	
12	""	"The Pastoralist"	
13	""	"Clos De Los Siete"	

Figure 4.6: Grape preferences table

4.2.5 The "Menu meal" table

The "Menu meal" table (Figure 4.7) takes as input one string parameters, that is "Menu" and returns as output the name of the single wine that matches the menu meal choose by the guest in string form.

Applying the hit policy "Collect" the wine list is obtained based on the guest's menu meal choice preferences. The output of the "Menu meal" table represents one of the inputs that is passed to the "Wine Selection" table.

Furthermore, cases in which the customer does not enter the menu meal or enter another word than those considered in the program are managed, in fact entering the dish is mandatory and if it is not entered or a dish or a word not managed by the program is entered, no output wines will be obtained.

Menu meal Hit Policy: Collect ▾			
	When	Then	
	Menu	Recommended wines on menu	Annotations
	"beef bourguignon","chicken cae...	"Masseto","The Pastoralist","Clo...	
1	"beef bourguignon"	"Masseto"	
2	"beef bourguignon"	"The Pastoralist"	
3	"beef bourguignon"	"Clos De Los Siete"	
4	"beef bourguignon"	"Bourgogne Pinot Noir"	
5	"beef bourguignon"	"Malene Rosé"	
6	"beef bourguignon"	"Whispering Angel Rosé"	
7	"chicken caesar salad"	"Masseto"	
8	"chicken caesar salad"	"The Pastoralist"	
9	"chicken caesar salad"	"Clos De Los Siete"	
10	"chicken caesar salad"	"Bourgogne Pinot Noir"	
11	"chicken caesar salad"	"Fontanafredda Moscato Asti"	
12	"chicken caesar salad"	"Mionetto Prosecco Brut"	
13	"chicken caesar salad"	"Montrachet Grand Cru"	
14	"chicken caesar salad"	"Coteau De Vernon"	
15	"chicken caesar salad"	"Malene Rosé"	

Figure 4.7: Menu meal table

4.2.6 The "Wine Selection" table

The "Wine Selection" table (Figure 4.8) takes as input five string parameters, that are "Country", "Region", "Taste", "Grape" and "Menu" and returns as a string output the name of the individual wine based on the guest preference and on the menu meal only those contained in the list passed as input. Again, the hit policy "collect" was used to group all the wines together.

Wine selection Hit Policy: Collect							
	When	And	And	And	And	Then	
	Country	Region	Taste	Grape	Menu	Recommended Wines	Annotations
	string	string	string	string	string	string	
1	list contains(winesC, "Masseto")	list contains(winesR, "Masseto")	list contains(winesT, "Masseto")	list contains(winesG, "Masseto")	list contains(winesM, "Masseto")	"Masseto"	
2	list contains(winesC, "The Pastoralist")	list contains(winesR, "The Pastoralist")	list contains(winesT, "The Pastoralist")	list contains(winesG, "The Pastoralist")	list contains(winesM, "The Pastoralist")	"The Pastoralist"	
3	list contains(winesC, "Clos De Los Siete")	list contains(winesR, "Clos De Los Siete")	list contains(winesT, "Clos De Los Siete")	list contains(winesG, "Clos De Los Siete")	list contains(winesM, "Clos De Los Siete")	"Clos De Los Siete"	
4	list contains(winesC, "Bourgogne Pinot Noir")	list contains(winesR, "Bourgogne Pinot Noir")	list contains(winesT, "Bourgogne Pinot Noir")	list contains(winesG, "Bourgogne Pinot Noir")	list contains(winesM, "Bourgogne Pinot Noir")	"Bourgogne Pinot Noir"	
5	list contains(winesC, "Fontanafredda Moscato Asti")	list contains(winesR, "Fontanafredda Moscato Asti")	list contains(winesT, "Fontanafredda Moscato Asti")	list contains(winesG, "Fontanafredda Moscato Asti")	list contains(winesM, "Fontanafredda Moscato Asti")	"Fontanafredda Moscato Asti"	
6	list contains(winesC, "Mionetto Prosecco Brut")	list contains(winesR, "Mionetto Prosecco Brut")	list contains(winesT, "Mionetto Prosecco Brut")	list contains(winesG, "Mionetto Prosecco Brut")	list contains(winesM, "Mionetto Prosecco Brut")	"Mionetto Prosecco Brut"	
7	list contains(winesC, "Montrachet Grand Cru")	list contains(winesR, "Montrachet Grand Cru")	list contains(winesT, "Montrachet Grand Cru")	list contains(winesG, "Montrachet Grand Cru")	list contains(winesM, "Montrachet Grand Cru")	"Montrachet Grand Cru"	
8	list contains(winesC, "Coteau De Vernon")	list contains(winesR, "Coteau De Vernon")	list contains(winesT, "Coteau De Vernon")	list contains(winesG, "Coteau De Vernon")	list contains(winesM, "Coteau De Vernon")	"Coteau De Vernon"	
9	list contains(winesC, "Malene Rosé")	list contains(winesR, "Malene Rosé")	list contains(winesT, "Malene Rosé")	list contains(winesG, "Malene Rosé")	list contains(winesM, "Malene Rosé")	"Malene Rosé"	

Figure 4.8: Wine Selection table

The "list contains" function belonging to the FEEL language was used to check that the lists of input contain particular wines.

In the context of business process modeling and decision management, FEEL (Friendly Enough Expression Language) is a language specifically designed for expressing decision rules and logic in a human-readable and understandable format. It is used in conjunction with the Decision Model and Notation (DMN) standard.

FEEL is a highly expressive language that aims to strike a balance between being business-friendly and having the necessary computational power for decision automation. It allows for the representation of complex expressions and decision logic in a straightforward and intuitive manner.

4.3 Camunda DMN-Simulator

In this project Camunda DMN-Simulator (Chapter 2.2) was used to test the correct functioning of the DMN and the decision tables created.

In our simulation (Figure 4.9) it is possible to see the input fields that the customer has to enter regarding his preferences on country, region, taste, grape and the chosen menu meal that is obligatory to insert. (*Please note that taste and meal are written in lowercase, while the other criteria have the first letter in upper case.*)

After entering the input fields, the user can click the "Simulate now" button.

There are five output fields, one for each decision table created. The first five output fields represent the wines recommended by the program based on each of your preferences and dish individually. The last output field represents the recommended wines based on the customer's preferences and the chosen dish and is therefore the intersection of the wines from the previous outputs.

The screenshot shows the Camunda DMN Simulator interface. At the top, there's a header with the Camunda logo and the title "DMN Simulator". Below this, the interface is divided into "Inputs:" and "Outputs:" sections.

Inputs:

- Decision table:** A dropdown menu showing "Wine selection".
- Country preferences:** A text input field with "France" and a "string" type indicator.
- Grape preferences:** A text input field with "Chardonnay" and a "string" type indicator.
- Taste preferences:** A text input field with "bitter" and a "string" type indicator.
- Region preferences:** A text input field with "Burgundy" and a "string" type indicator.
- Menu meal:** A text input field with "fresh fruit salad" and a "string" type indicator.

A green "Simulate now" button is located below the input fields.

Outputs:

- Recommended Wines on Country:** A text box containing "Montrachet Grand Cru, Coteau De Vernon, Whispering Angel Rosé".
- Recommend wines on grapes:** A text box containing "Montrachet Grand Cru".
- Recommended wines on taste:** A text box containing "Masseto, The Pastoralist, Clos De Los Siete, Bourgogne Pinot Noir".
- Recommended wines on region:** A text box containing "Montrachet Grand Cru, Coteau De Vernon, Bourgogne Pinot Noir".
- Recommended wines on menu:** A text box containing "Mionetto Prosecco Brut, Montrachet Grand Cru, Malene Rosé, Whis".
- Recommended Wines:** A text box containing "Montrachet Grand Cru".

At the bottom of the interface, there's a section titled "Wine Selection" with a small diagram showing a decision tree structure.

Figure 4.9: Simulation of DMN Diagram

5. Rule-Based Systems: Logic Programming and Prolog

This chapter will illustrate the development of a knowledge-based solution based on Prolog.

5.1 Introduction

5.1.1 Rule-Based Systems

A rule-based system is a type of knowledge-based system that uses a set of rules to make decisions or perform reasoning tasks. It is designed to simulate human expertise and problem-solving abilities within a specific domain. The rules in a rule-based system consist of conditions and corresponding actions. When presented with input data, the system matches the input against the conditions specified in the rules and executes the corresponding actions for the rules whose conditions are satisfied.

5.1.2 Logic programming

On the other hand, logic programming is a programming paradigm based on formal logic. It aims to solve problems by using logical statements and rules to infer new knowledge or derive conclusions. In logic programming, programs are composed of logical statements written in a formal logic language, such as Prolog.

Logic programming languages, like Prolog, provide a mechanism to define facts and rules that represent knowledge about a particular domain. Facts are statements that represent specific information or relationships, while rules define logical relationships and can be used to deduce new facts based on existing ones. Logic programming languages use a form of automated reasoning called resolution to search for solutions by applying logical inference rules.

One of the key concepts in logic programming is pattern matching, where queries or input statements are matched against the facts and rules in the program. The logic programming engine attempts to find solutions by unifying the query with the facts and rules, and if a match is found, it can derive new facts or provide the desired output.

5.1.3 Usages

Both rule-based systems and logic programming share similarities in their use of rules and pattern matching. However, rule-based systems often focus on applying rules to make decisions or perform actions based on the input, while logic programming aims to solve problems through logical inference and deduction based on the available knowledge.

5.2 Prolog

5.2.1 Facts and Rules

To model a knowledge-based solution with Prolog (Chapter 2.3), we followed the idea to describe the wines in terms of characteristics (i. e. "Country/Region provenience", "Grapes used", "Taste"). To do so, we defined a set of facts to describe the characteristics of the wines and the pairing with meals.

Facts are statements that represent information about the world. They are the basic building blocks of knowledge in Prolog. Facts are written in the form of predicates, which are relationships between objects or properties of objects. A fact simply states that a particular predicate is true.

The facts written include the following examples shown below:

- `tuscany(masseto).`
- `merlot(masseto).`
- `acid(masseto).`
- `bitter(masseto).`
- `pairs(beef_bourguignon, masseto).`

Rules define relationships and logical implications based on facts. They allow you to infer new information from existing knowledge. Rules consist of a head and a body. The head specifies a predicate that can be inferred, while the body contains a logical condition or a conjunction of conditions. If the conditions in the body are satisfied, the predicate in the head is considered true.

Some example of rule in this project:

- `italy(X) :- tuscany(X).`
- `france(X) :- provence(X).`

For this implementation a simple to understand set of rules and facts and a choice of the user through numerical values present in the preference lists have been chosen, for a complete simplicity in the insertion of the input and an easy interpretation of the code.

5.2.2 Online Editor: SWISH¹

To write and test the knowledge-based solution with Prolog, the online SWI-Prolog tool was used. To continue with the testing you have to paste your facts and rules into SWI-Prolog and it enables you to execute your rules and give you an output to test your solution.

The user can access to the menu, entering his preferences, through a choice menu (Figure 5.1) that asks the user to select various preferences by entering a numeric value corresponding to the choice. Once the user's preferences are obtained, the list of wines is filtered for each constraint chosen by the user.

¹<https://swish.swi-prolog.org/>

WELCOME TO THE KEBI INTERNATIONAL RESTAURANT!

Please help us show which wines are most suitable for you by providing information about your preferences

Would you like the wine to come from a specific Country?

0. No
1. Italy
2. France
3. Argentina
4. Australia
5. United States

Enter your choice number below:

2

Would you like the wine to come from a specific Region?

0. No
1. Tuscany, Italy
2. Piedmont, Italy
3. Veneto, Italy
4. Burgundy, France
5. Provence, France
6. Mendoza, Argentina
7. South Australia, Australia
8. California, United States

Enter your choice number below:

Please enter a Prolog term

Send Abort

? wineMenu(X).

Examples History Solutions

☒ table results Run

Figure 5.1: Prolog Choice Menu

5.2.3 Examples

Example 1

To run the choice menu, execute the rule: wineMenu(X).

The variable X corresponds to a wine. The end result (Figure 5.2) is the list of all possible wines that correspond to the constraints put by the user and the pairing with the meal chosen (if there is no wine that matches the specified rules it returns false).

A first example of this choice, with the following constraint:

- Country: France,
- Region: Burgundy,
- Grape preferences: none,
- Grape to exclude: none,
- Taste: Spice,
- Meal chosen: Fresh Fruit Salad.
- **Result: Montrachet Grand Cru.**

Based on the preferences you provided, and the meal chosen, this is the list of wines that meet your needs:

	X	
montrachet_grand_cru		1

Figure 5.2: First example in Prolog

Example 2

An other example, with no preferences of the user, but only the meal pairing is the following:

- Meal chosen: Beef Bourguignon.
- **Result:**
 - Masseto
 - Clos De Los Siete
 - The Pastoralist
 - Bourgogne Pinot Noir
 - Malene Rosé
 - Whispering Angel Rosé

Based on the preferences you provided, and the meal chosen, this is the list of wines that meet your needs:

X	
masseto	1
clos_de_los_siete	2
the_pastoralist	3
bourgogne_pinot_noir	4
malene_rose	5
whispering_angel_rose	6

Figure 5.3: Second example in Prolog

Example 3

An example regarding the grape preferences is the following:

- Country: none,
- Region: none,
- Grape preferences: Cabernet Sauvignon,
- Grape to exclude: Merlot,
- Taste: none,
- Meal chosen: Chicken Caesar Salad.
- **Result: The Pastoralist.**

Based on the preferences you provided, and the meal chosen, this is the list of wines that meet your needs:

X	
the_pastoralist	1

Figure 5.4: Third example in Prolog

6. Knowledge Graph

This chapter will explain the development of an Ontology Structure using the editor Protégé and the creation of a Knowledge Graph.

6.1 Ontologies in Protégé

Ontologies in Protégé are representations of knowledge that describe concepts, relationships, and properties within a specific domain. Protégé is a popular ontology development platform widely used (Chapter 2.4).

When creating ontologies in Protégé, it is typical to start by defining classes, which represent the concepts or types of entities in the domain. Classes can be hierarchical, forming a taxonomy or classification hierarchy.

Properties play a crucial role in ontologies to describe relationships between entities. In Protégé, it is possible to define object properties and data properties. Object properties connect entities to other entities, indicating relationships. Data properties, on the other hand, assign attribute values to entities. They describe characteristics or attributes of the entities.

Protégé provides a graphical user interface that allows to create and modify ontology elements using a visual representation.

6.2 Class Hierarchy

A class hierarchy in Protégé is a hierarchical structure that organizes classes in an ontology based on their inheritance relationships. It represents a taxonomy or classification of concepts in a specific domain.

In a class hierarchy, classes are arranged in a parent-child relationship, where a child class inherits the properties and characteristics of its parent class. The parent class is referred to as the superclass, and the child class is called the subclass. This hierarchy allows for the organization and categorization of concepts in a logical and structured manner.

The class hierarchy in Protégé allows to define the inheritance relationships between classes, establish the taxonomy of concepts, and capture the generalization-specialization relationships in the domain. It facilitates organizing and reasoning with the concepts and enables to perform queries and inference operations based on the defined hierarchy.

The class hierarchy of this project, Figure 6.1, shows that "Thing" is the top-level superclass, and it has direct subclasses, "country", "region", "grape", "taste", "meal" and "wine". All of them contain a set of individuals created to represent the real scenario cases. For example, the class "meal" contains the menu of an international restaurant, with a list of dishes. In fact, Individuals in Protégé are typed as members of one or more classes organized and displayed as a hierarchy.



Figure 6.1: Class Hierarchy in Protégé

6.3 Object Property Hierarchy

The object property hierarchy refers to the hierarchical relationships established between object properties.

Object properties represent relationships or connections between individuals in an ontology. They describe how instances of one class are related to instances of another class.

In Protégé, a hierarchy of object properties is defining by subproperties and superproperties. A subproperty is a more specific version of a superproperty. It inherits the characteristics of the superproperty while adding more specific details.

By organizing object properties in a hierarchy, it is possible to establish a more structured and organized representation of relationships in the ontology. This hierarchy enables reasoning systems to infer additional knowledge and make more accurate deductions based on the relationships defined in the ontology.

In our case, (Figure 6.2), we have the Object Properties listed in the Figure 6.2a: "comes_from", "has", "is_made_of", "is_part_of" and "pairs_with".

The first one, "**comes_from**", refers to the connection of a wine that comes from a region. The Domain, therefore, is the wine and the Ranges is the region (Figure 6.2b).

The "**has**" property, instead, relate the wine to some taste it has.

The "**is_made_of**", represent the wine and the grape it is, in fact, made of.

"**is_part_of**", connect a region and the country it is situated in.

And the last one, "**pairs_with**", connect each wine to all the meals it pairs well with.

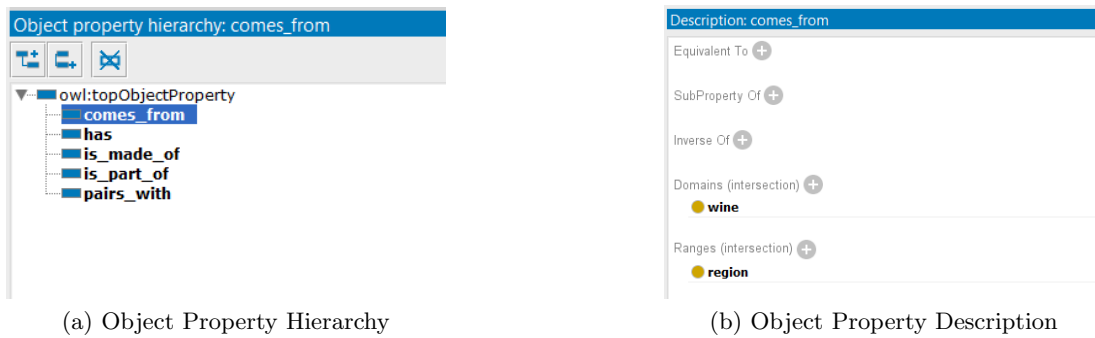


Figure 6.2: Object Property

6.4 Data Property Hierarchy

The data property hierarchy refers to the hierarchical relationships established between data properties.

Data properties represent attributes or characteristics of individuals in an ontology. They describe the data values associated with instances of a class.

Also here, there are subproperties and superproperties.

In the data properties the predicate connects a single subject with some form of attribute data. Data properties have defined datatypes including string, integer, date, datetime, or boolean.

In our case, (Figure 6.3), we have the Data Properties of the Figure 6.3a: *"has_id"*, *"country_name"*, *"region_name"*, *"grape_name"*, *"taste_name"*, *"meal_name"* and *"wine_name"*.

The *"has_id"* property describe the id of a wine. The *"*_name"* properties, instead, are strings with the name of each instance.

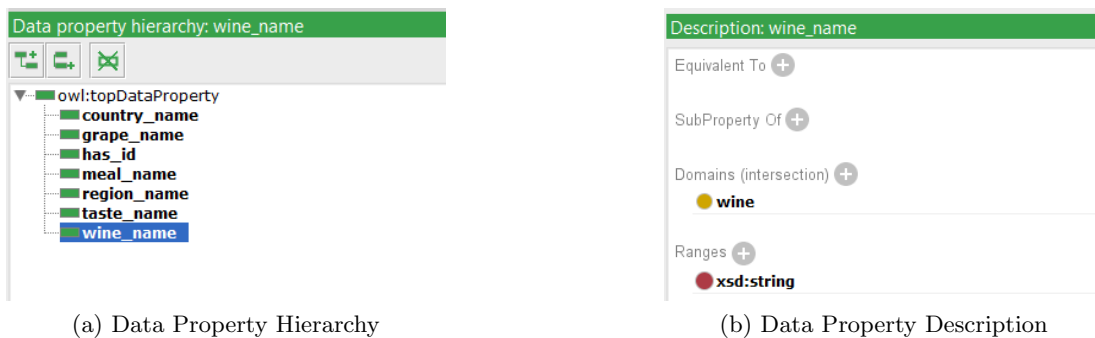


Figure 6.3: Data Property

6.5 Resulting Knowledge Graph

A knowledge graph refers to a representation of knowledge and information in the form of a graph structure. It is created using the ontology modeling capabilities of Protégé.

A knowledge graph in Protégé consists of nodes and edges, where nodes represent entities or concepts, and edges represent relationships between these entities. Nodes are typically associated with classes or individuals in an ontology, while edges represent object properties or data properties that connect the nodes.

Knowledge graphs in Protégé provide a visual and intuitive way to represent complex relationships and dependencies between entities in an ontology. They allow users to explore and analyze the connections and associations between different entities, enabling better understanding and reasoning about the domain being modeled.

The resulting Knowledge Graph of this project's ontology (Figure 6.4) is divided into the classes: wine, meal, taste, grape, region and country. As visible in the figure the *wine* class is connected with the *taste* class through the Object Property "has", with the *grape* class through "is-made-of" and with the *meal* class thanks to "pairs-with". Finally, the *wine* class is also connected with the *region* class by the Object Property "comes-from" and the *region* class is in turn connected with the *country* class by the Object Property "is-part-of".

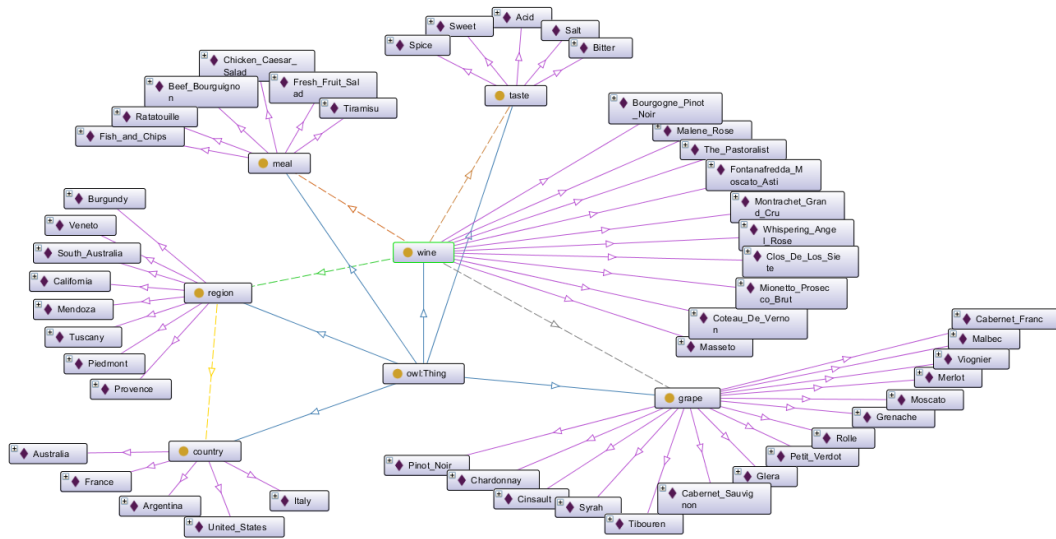


Figure 6.4: Knowledge Graph

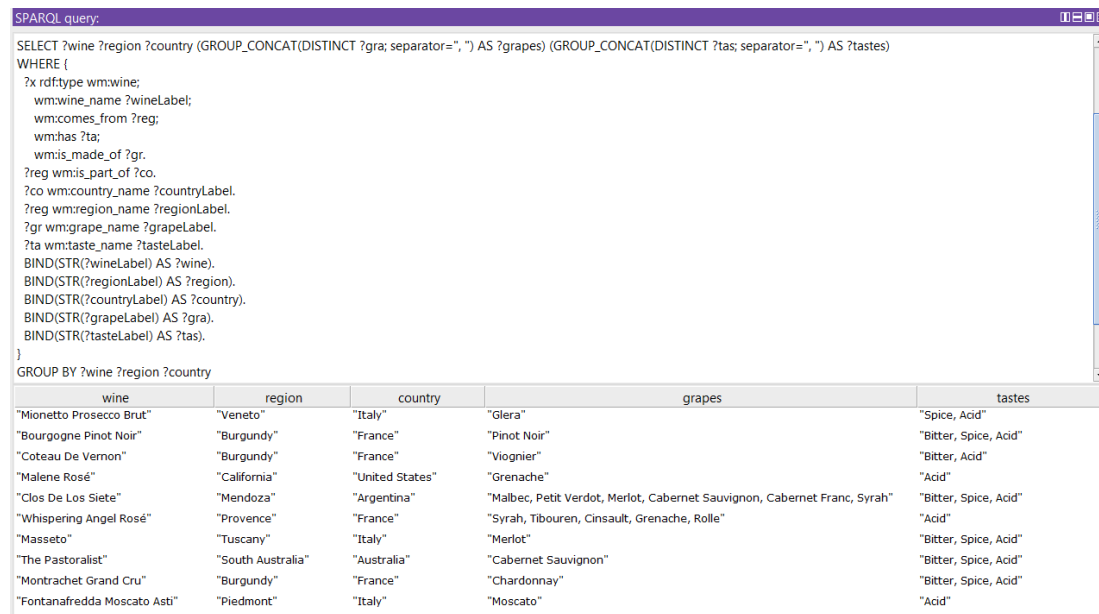
6.6 SPARQL Query Plugin and queries

The SPARQL Plugin for Protégé is an extension that adds SPARQL query capabilities to the Protégé ontology editing tool. SPARQL (SPARQL Protocol and RDF Query Language) is a standardized query language for querying and manipulating RDF data.

The SPARQL Plugin allows users to execute SPARQL queries directly within Protégé, enabling them to retrieve specific information from ontologies or knowledge graphs stored in RDF format. It provides an interface where users can write SPARQL queries and execute them against the loaded ontology.

In our case, we defined a set of query to get the different filtered wines menu.

For example, in the (Figure 6.5), we defined a query for the list of all the wines present and the characteristics of each.



The screenshot shows a SPARQL query window with the following query:

```
SELECT ?wine ?region ?country (GROUP_CONCAT(DISTINCT ?gr; separator=", ") AS ?grapes) (GROUP_CONCAT(DISTINCT ?tas; separator=", ") AS ?tastes)
WHERE {
  ?x rdf:type wmwine;
  wmwine_name ?wineLabel;
  wmwcomes_from ?reg;
  wmwhas ?ta;
  wmwis_made_of ?gr.
  ?reg wmwis_part_of ?co.
  ?co wmwcountry_name ?countryLabel.
  ?reg wmwregion_name ?regionLabel.
  ?gr wmwgrape_name ?grapeLabel.
  ?ta wmw taste_name ?tasteLabel.
  BIND(STR(?wineLabel) AS ?wine).
  BIND(STR(?regionLabel) AS ?region).
  BIND(STR(?countryLabel) AS ?country).
  BIND(STR(?grapeLabel) AS ?gr).
  BIND(STR(?tasteLabel) AS ?tas).
}
GROUP BY ?wine ?region ?country
```

The results are displayed in a table with the following columns: wine, region, country, grapes, and tastes.

wine	region	country	grapes	tastes
"Mionetto Prosecco Brut"	"Veneto"	"Italy"	"Glera"	"Spice, Acid"
"Bourgogne Pinot Noir"	"Burgundy"	"France"	"Pinot Noir"	"Bitter, Spice, Acid"
"Coteau De Vernon"	"Burgundy"	"France"	"Viognier"	"Bitter, Acid"
"Malene Rosé"	"California"	"United States"	"Grenache"	"Acid"
"Clos De Los Siete"	"Mendoza"	"Argentina"	"Malbec, Petit Verdot, Merlot, Cabernet Sauvignon, Cabernet Franc, Syrah"	"Bitter, Spice, Acid"
"Whispering Angel Rosé"	"Provence"	"France"	"Syrah, Tibouren, Cinsault, Grenache, Rolle"	"Acid"
"Masseto"	"Tuscany"	"Italy"	"Merlot"	"Bitter, Spice, Acid"
"The Pastoralist"	"South Australia"	"Australia"	"Cabernet Sauvignon"	"Bitter, Spice, Acid"
"Montrachet Grand Cru"	"Burgundy"	"France"	"Chardonnay"	"Bitter, Spice, Acid"
"Fontanafredda Moscato Asti"	"Piedmont"	"Italy"	"Moscato"	"Acid"

Figure 6.5: SPARQL Query for listing all wines

Note that each query requires some prefixes, the following are necessary to run our queries.

Listing 6.1: Prefixes required in SPARQL Queries

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX wm: <http://www.semanticweb.org/greta/ontologies/2023/5/wineMenu#>
```

6.6.1 Queries implemented

To help the guest choosing a specific wine, according to his preferences and the meal picked, some queries were implemented in SPARQL.

Following the most useful:

Filtering by grapes

With these queries the user can filter the wine list, specifying one or two criteria, like the grape preferred (Listing 6.2), the grape to exclude (Listing 6.3), or both (Listing 6.4).

Listing 6.2: SPARQL Query filtering grape preferred

```
#Select wines with a particular Grape (ex. Merlot)
SELECT (str(?wineLabel) as ?wine)
WHERE {
    ?x rdf:type wm:wine;
    wm:wine_name ?wineLabel;
    wm:is_made_of wm:Merlot.
}
```

Listing 6.3: SPARQL Query filtering grape to exclude

```
#Select wines without a particular Grape (ex. Tibouren)
SELECT (str(?wineLabel) as ?wine)
WHERE {
    ?x rdf:type wm:wine;
    wm:wine_name ?wineLabel;
    MINUS {
        ?x wm:is_made_of wm:Tibouren.
    }
}
```

Listing 6.4: SPARQL Query filtering grape preferred and grape to exclude

```
#Select wines with a particular Grape (ex. Merlot) but
#excluding an other Grape (ex. Cabernet Sauvignon)
SELECT (str(?wineLabel) as ?wine)
WHERE {
    ?x rdf:type wm:wine;
    wm:wine_name ?wineLabel;
    wm:is_made_of wm:Merlot.
    MINUS {
        ?x wm:is_made_of wm:Cabernet_Sauvignon.
    }
}
```


Filtering by country or region

With these queries the user can filter the wine list, specifying the country or the region, in fact if he can select the country of origin (Listing 6.5) and all the wines that come from the regions inside the country specified are returned, or he can select directly the region for a more specific research (Listing 6.6).

Listing 6.5: SPARQL Query filtering country preferred

```
#Select wines from a specific Country (ex. United States)
SELECT (str(?wineLabel) as ?wine)
WHERE {
    ?x rdf:type wm:wine;
    wm:wine_name ?wineLabel;
    wm:comes_from ?region.
    ?region wm:is_part_of wm:United_States.
}
```

Listing 6.6: SPARQL Query filtering region preferred

```
#Select wines from a specific Region (ex. Burgundy)
SELECT (str(?wineLabel) as ?wine)
WHERE {
    ?x rdf:type wm:wine;
    wm:wine_name ?wineLabel;
    wm:comes_from wm:Burgundy.
}
```

Filtering by meal chosen

With this query the user can filter the wine list, specifying the meal chosen, and the result is the list of wines that can actually pair well with that meal (Listing 6.7).

Listing 6.7: SPARQL Query filtering pairing with the meal chosen

```
#Select wines that pair with a specific Meal
 #(ex. Fresh Fruit Salad)
SELECT ?wine
WHERE {
    ?x rdf:type wm:wine;
    wm:wine_name ?wineLabel;
    wm:pairs_with wm:Fresh_Fruit_Salad.
    BIND(STR(?wineLabel) AS ?wine).
}
```

Filtering with more criteria at the same time

With these queries the user can filter the wine list, specifying more than one characteristic.

For example, it is possible filtering on region, taste and grape preferences (Listing 6.8).

Listing 6.8: SPARQL Query filtering region, taste, grape preferred and grape to exclude

```
#Select wines with a specific Region (Veneto), Taste (Acid),  
#Grape (Glera) and without a Grape (Moscato).  
SELECT ?wine  
WHERE {  
    ?x rdf:type wm:wine ;  
    wm:wine_name ?wineLabel;  
    wm:comes_from wm:Veneto;  
    wm:has wm:Acid;  
    wm:is_made_of wm:Glera .  
    filter not exists {  
        ?x wm:is_made_of wm:Moscato;  
    }  
    BIND(STR(?wineLabel) AS ?wine).  
}
```

An other example, the more specific one, is to filter with all the possible characteristics and also the pairing with meal chosen. A version of it is specifying the region (Listing 6.9), and the other the country (Listing 6.10).

Listing 6.9: SPARQL Query filtering region, taste, grape preferred, grape to exclude and pairing with meal chosen

```
#Select wines with a specific Region (Burgundy), Taste (Spice),  
#pair with a Meal (Fish and Chips), with a Grape (Chardonnay)  
#and without a Grape (Pinot Noir).  
SELECT ?wine  
WHERE {  
    ?x rdf:type wm:wine ;  
    wm:wine_name ?wineLabel;  
    wm:comes_from wm:Burgundy;  
    wm:has wm:Spice;  
    wm:pairs_with wm:Fish_and_Chips;  
    wm:is_made_of wm:Chardonnay .  
    filter not exists {  
        ?x wm:is_made_of wm:Pinot_Noir;  
    }  
    BIND(STR(?wineLabel) AS ?wine).  
}
```

Listing 6.10: SPARQL Query filtering country, taste, grape preferred, grape to exclude and pairing with meal chosen

```
#Select wines from France, that are Acid, contain Chardonnay
#but not Viognier, and pair with Ratatouille.
SELECT ?wine
WHERE {
    ?x rdf:type wm:wine;
    wm:wine_name ?wineLabel;
    wm:comes_from ?region;
    wm:has wm:Acid;
    wm:is_made_of wm:Chardonnay;
    wm:pairs_with wm:Ratatouille.
    ?region wm:is_part_of wm:France.
    filter not exists {
        ?x wm:is_made_of wm:Viognier;
    }
    BIND(STR(?wineLabel) AS ?wine).
}
```

Query to select and visualize all wines

Finally, an other query was implemented (Listing 6.11), not to filter but to show all the wines present in the menu, with all their characteristics, as shown in the Figure 6.5.

Listing 6.11: SPARQL Query showing all wines and their characteristics

```
SELECT ?wine ?region ?country
(GROUP_CONCAT(DISTINCT ?gra; separator="," ) AS ?grapes)
(GROUP_CONCAT(DISTINCT ?tas; separator="," ) AS ?tastes)
WHERE {
    ?x rdf:type wm:wine;
    wm:wine_name ?wineLabel;
    wm:comes_from ?reg;
    wm:has ?ta;
    wm:is_made_of ?gr.

    ?reg wm:is_part_of ?co.
    ?co wm:country_name ?countryLabel.
    ?reg wm:region_name ?regionLabel.
    ?gr wm:grape_name ?grapeLabel.
    ?ta wm:taste_name ?tasteLabel.

    BIND(STR(?wineLabel) AS ?wine).
    BIND(STR(?regionLabel) AS ?region).
    BIND(STR(?countryLabel) AS ?country).
    BIND(STR(?grapeLabel) AS ?gra).
    BIND(STR(?tasteLabel) AS ?tas).
}
GROUP BY ?wine ?region ?country
```

7. Graph Modeling Language

This chapter describes how we tried with ADOxx (Chapter 2.5) to design a graphical modeling language, which allows a chef to represent meals and wines in a graphical way, such that it contains all information relevant for the guest to select wines according to their preferences.

7.1 Introduction to a graphical modeling language

ADOxx is a modeling tool that allows the creation of graphical modeling languages. A graphical modeling language, in the context of ADOxx, refers to a customized language specifically designed for a particular domain or system. It uses graphical elements, such as symbols, icons, and diagrams, to represent concepts, relationships, and processes within that domain or system. The language is defined using ADOxx's modeling capabilities, enabling users to visually represent and communicate complex information in a clear and intuitive manner. With ADOxx, users can create graphical models that capture the essential aspects of a domain or system, facilitating understanding, analysis, and decision-making.

7.2 User and Library Management

With the Development Toolkit it has been registered a new User with a Password (Figure 7.1) to which a specific created Library has been associated.

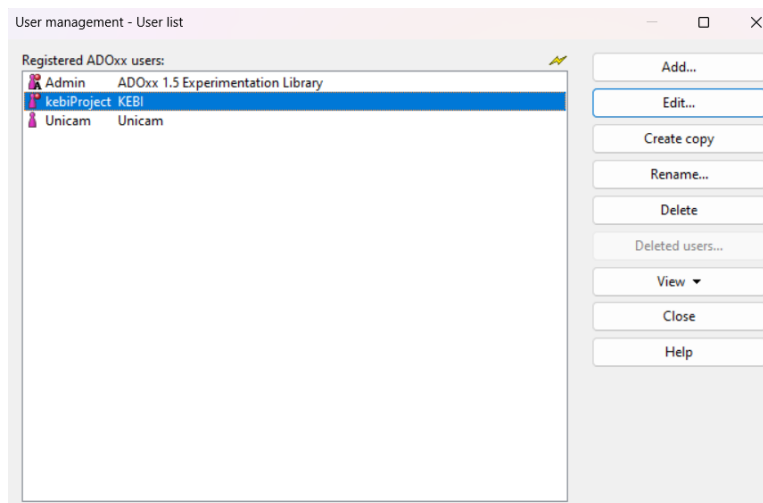


Figure 7.1: User management

Username: kebiProject
Password: kebi
Application library: KEBI

The KEBI Library is composed by the Dynamic Experimentation Library called KEBI Dynamic and the Static Experimentation Library called KEBI Static (Figure 7.2).

Initially, the Model Type, that is a well-defined sub-collection of classes and relation classes of a meta model, must be defined in the Library attributes section of the dynamic library, which in this case is *International Restaurant*.

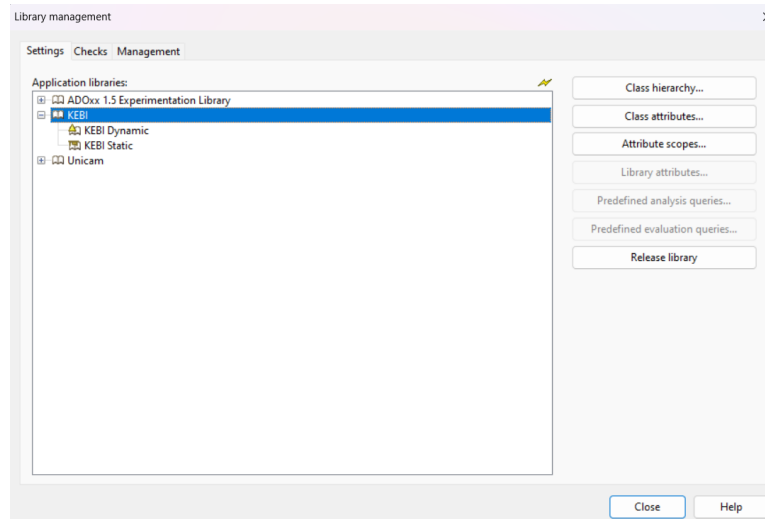


Figure 7.2: Library management

7.3 Class hierarchy

The meta model of a model language is defined by classes of elements and relations, class hierarchy and attributes of the elements. The notation is defined by special attribute GraphRep.

In the class hierarchy, ADOxx distinguishes Classes and Relation classes.

7.3.1 Classes

The classes that have been created within the dynamic library are:

- **Wine:** is the class that represents each wine of the restaurant. The attributes of this class, other than the name, are: Wine taste and Wine type. These are included in the AttrRep of the Wine class and they are defined as ENUMERATION for Wine type and as ENUMERATIONLIST for Wine taste.



Figure 7.3: Wine Attributes

"*Wine type*" represents the type of wine, i.e. whether it is a red, white or rosé wine. Also in this case, depending on the type, the corresponding icon will appear inside the class rectangle, i.e. a glass with red, white or rosé wine inside.

(a) Wine type attribute

(b) Wine taste attribute

Figure 7.4: Values of wine attributes

"*Wine taste*" defines the taste or tastes of wine among the five defined for the knowledge base: acid, bitter, sweet, spice and salt. For each flavour, the corresponding icon will become coloured, i.e. a lemon for the acid taste, a red icon for the bitter taste, a candy for the sweet taste, chillies for the spice taste and a "salt shaker" for the salty taste.

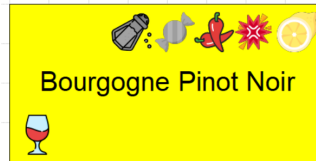


Figure 7.5: Wine Attributes icons example

- **Meal:** is the class that represents each meal of the menu. The attribute of this class, other than the name, is Meal type that is included in the AttrRep of the Meal class and it's defined as ENUMERATION.

Figure 7.6: Meal type attribute

"*Meal type*" defines the category of dish of the menu, whether it is red meat, fish, chicken, vegetables or desserts. The corresponding icon will appear for each type.

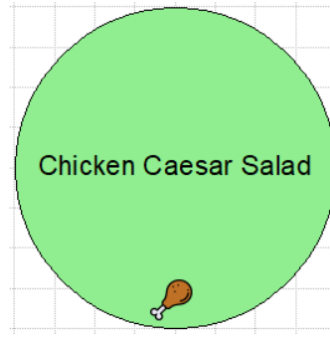


Figure 7.7: Meal attribute icons example

- **Country:** is the class that represents each country from which the restaurant's wines come from.



Figure 7.8: Country class example

- **Region:** is the class that represents each region of the countries from which the restaurant's wines come from.

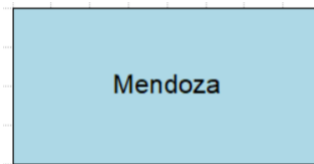


Figure 7.9: Region class example

- **Grape:** is the class that represents each grape of which the restaurant's wines are made.



Figure 7.10: Grape class example

7.3.2 Relation Classes

The Relation Classes that have been created within the dynamic library are:

- **comes_from**: is the class that defines the origin relationship of the wine from a specific region; in fact, the direction of the relationship goes from wine to region.

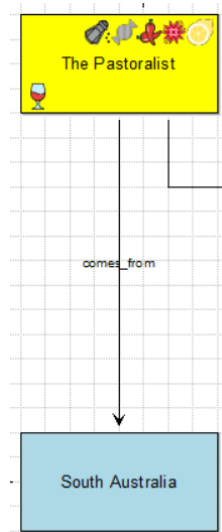


Figure 7.11: "Comes from" relation class example

- **is_a_part_of**: is the class that defines the relationship between a region and a country, indicating that the region is located within the specified country.

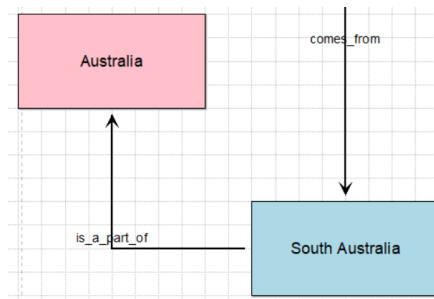


Figure 7.12: "Is a part of" relation class example

- **is_made_of:** is the class that defines the relationship between wine and grape, specifying which grapes the wine is made of.

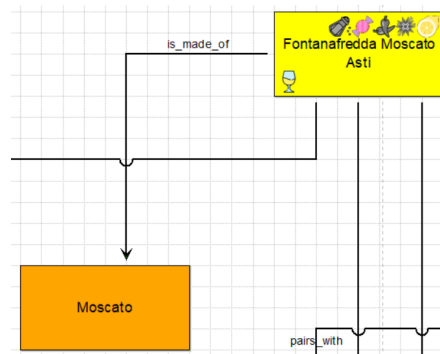


Figure 7.13: "Is made of" relation class example

- **pairs_with:** is the class that defines the relationship between wine and menu meal, representing the pairing of all wines with all dishes. It indicates whether a certain wine pairs well with a meal and is therefore recommended.

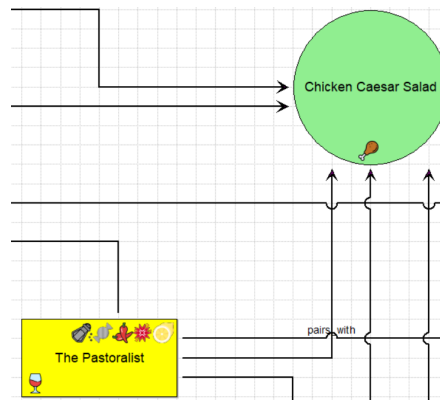


Figure 7.14: "Pairs with" relation class example

7.4 The special attribute **GraphRep**

GraphRep is a script language for the graphical representation used in ADOxx Development Toolkit. GraphRep allows to draw elements and display texts.

In every class and relation class, the GraphRep code is defined to visually represent both the class itself and its attributes and relations. The graphical representation will be visible in the Modelling Toolkit.

7.4.1 GraphRep of Wine class

In the following code, you can see that the two variables "winetaste" and "winetype" are defined for the attributes of the wine.

Additionally, icons representing the wine taste are displayed, and they are colored based on the presence or absence of specific tastes.

In the last part, the wine type is handled, and an icon with a red, white, or rosé wine glass is displayed based on the type.

Listing 7.1: Wine's GraphRep

```
GRAPHREP
AVAL winetaste: "Wine taste"
AVAL winetype: "Wine type"
FILL color:yellow
RECTANGLE x:-2cm y:-1cm w:4cm h:2cm
ATTR "Name" w:c:3.8cm h:c line-break:rigorous

BITMAP "db:\\lemonBW.png" x:1.4cm y:-0.9cm w:0.5cm h:0.5cm
BITMAP "db:\\angerBW.png" x:0.85cm y:-0.9cm w:0.5cm h:0.5cm
BITMAP "db:\\spiceBW.png" x:0.35cm y:-0.9cm w:0.5cm h:0.5cm
BITMAP "db:\\candyBW.png" x:-0.15cm y:-0.9cm w:0.5cm h:0.5cm
BITMAP "db:\\saltBW.png" x:-0.7cm y:-0.9cm w:0.5cm h:0.5cm

IF (search(winetaste, "acid", 0) != -1 )
    BITMAP "db:\\lemon.png" x:1.4cm y:-0.9cm w:0.5cm h:0.5cm
ENDIF

IF (search(winetaste, "bitter", 0) != -1 )
    BITMAP "db:\\anger.png" x:0.85cm y:-0.9cm w:0.5cm h:0.5cm
ENDIF

IF (search(winetaste, "spice", 0) != -1 )
    BITMAP "db:\\spice.png" x:0.35cm y:-0.9cm w:0.5cm h:0.5cm
ENDIF

IF (search(winetaste, "sweet", 0) != -1 )
    BITMAP "db:\\candy.png" x:-0.15cm y:-0.9cm w:0.5cm h:0.5cm
ENDIF

IF (search(winetaste, "salt", 0) != -1 )
    BITMAP "db:\\salt.png" x:-0.7cm y:-0.9cm w:0.5cm h:0.5cm
ENDIF

IF (winetype = "Red")
    BITMAP "db:\\red.png" x:-1.9cm y:0.4cm w:0.5cm h:0.5cm
ELSIF (winetype = "White")
    BITMAP "db:\\white.png" x:-1.9cm y:0.4cm w:0.5cm h:0.5cm
ELSIF (winetype = "Rose")
    BITMAP "db:\\rose.png" x:-1.9cm y:0.4cm w:0.5cm h:0.5cm
ENDIF
```

7.4.2 GraphRep of Meal class

In the code, there is the definition of the variable "mealtype" for the menu meal category. Additionally, based on the type, an icon corresponding to the meal appears within the rectangle of the class.

Listing 7.2: Meal's GraphRep

```
GRAPHREP
SHADOW off
FILL color:lightgreen
AVAL mealtype: "Meal type"

ELLIPSE x:0.00cm y:0cm rx:2cm ry:2cm
ATTR "Name" w:c:3.8cm h:c line-break:rigorous

IF (mealtype = "Red meat")
    BITMAP "db:\\redmeat.png" x:-0.3cm y:1.4cm w:0.5cm h:0.5cm
ELSIF (mealtype = "Fish")
    BITMAP "db:\\fish.png" x:-0.3cm y:1.4cm w:0.5cm h:0.5cm
ELSIF (mealtype = "Vegetables")
    BITMAP "db:\\vegetable.png" x:-0.3cm y:1.4cm w:0.5cm h:0.5cm
ELSIF (mealtype = "Chicken")
    BITMAP "db:\\chicken.png" x:-0.3cm y:1.4cm w:0.5cm h:0.5cm
ELSIF (mealtype = "Desserts")
    BITMAP "db:\\dessert.png" x:-0.3cm y:1.4cm w:0.5cm h:0.5cm
ENDIF
```

7.4.3 GraphRep of Country, Region and Grape classes

The following code represents the basic graphical visualization of the "Country," "Region," and "Grape" classes.

Listing 7.3: Country's GraphRep

```
GRAPHREP
FILL color:pink
RECTANGLE x:-2cm y:-1cm w:4cm h:2cm
ATTR "Name" w:c:3.8cm h:c line-break:rigorous
```

Listing 7.4: Region's GraphRep

```
GRAPHREP
FILL color:lightblue
RECTANGLE x:-2cm y:-1cm w:4cm h:2cm
ATTR "Name" w:c:3.8cm h:c line-break:rigorous
```

Listing 7.5: Grape's GraphRep

```
GRAPHREP
FILL color:orange
RECTANGLE x:-2cm y:-1cm w:4cm h:2cm
ATTR "Name" w:c:3.8cm h:c line-break:rigorous
```

7.5 Model result

The model, consisting of the "Wine," "Meal," "Country," "Region," and "Grape" classes, has been visually represented using the Modelling Toolkit . The graphical representation showcases the relations between the classes and the attributes of wines and menu dishes. This is the view of the graphic modeling language model created, which allows to represent meals and wines in a graphic way and contains all the relevant information for guests to select according to their preferences.

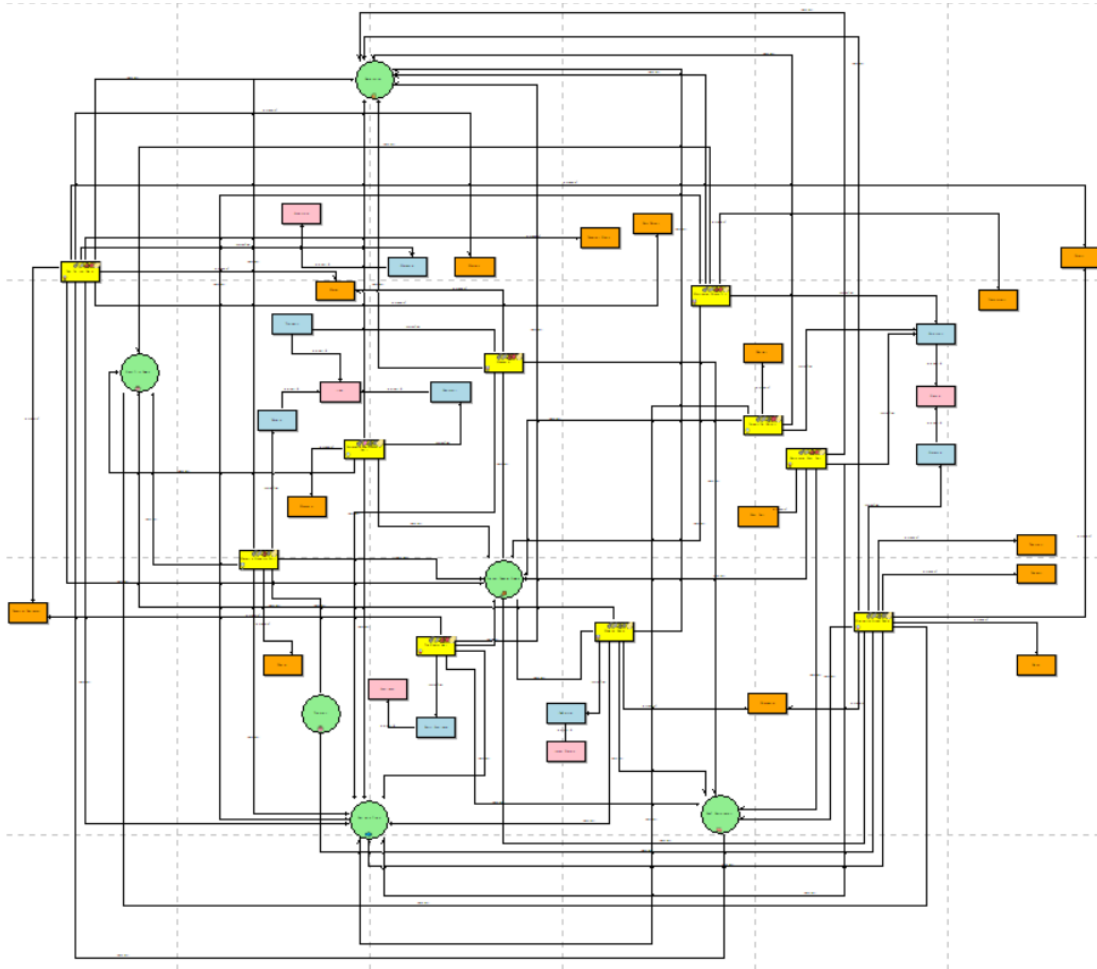


Figure 7.15: Model result

8. Albanese Clarissa's Conclusions

8.1 Introduction

In this chapter i will talk about advantages and disadvantages of the three knowledge-based solutions employed in our project: decision tables, Prolog, and knowledge graphs. I will discuss also about the design of a graphical modeling language realized with ADOxx. Each solution offers unique characteristics and capabilities, and understanding their strengths and limitations is crucial for making informed decisions regarding their implementation.

The project has demonstrated the necessity of continuous adaptation and updates to the knowledge base to accommodate evolving customer preferences and changes in menu offerings. Flexibility and maintainability of the knowledge-based systems are crucial for ensuring their long-term effectiveness.

8.2 The aim

In this project, we aimed to address the challenges faced by restaurants in the era of COVID-19 by creating a customized wine menu system. By digitizing menus and incorporating guest preferences, we aimed to assist customers in selecting wines that align with their preferences and complement the menu offerings. The aim was to represent knowledge of wines, considering factors such as country, region, grape, taste and their compatibility with different meals.

By incorporating personalized recommendations and facilitating a better understanding of wine options, the project aimed to enhance the overall dining experience for customers in the context of digitized menus.

8.3 Advantages and disadvantages

8.3.1 Decision Tables

Advantages

- **Structured Representation:** decision tables provide a structured format for representing knowledge, making it easier to organize and comprehend complex relationships between guest preferences, wine, and meal pairings.
- **Flexibility:** decision tables can be easily modified and updated, allowing for quick adjustments based on changing requirements or preferences.

- **Intuitive Interpretation:** The tabular format of decision tables lends itself to intuitive interpretation.

Disadvantages

- **Scalability:** managing decision tables can become cumbersome when dealing with a large number of preferences, options, and conditions. As the complexity increases, maintaining and updating the tables may become challenging.
- **Lack of Interconnectedness:** decision tables often lack the visual representation of interconnected relationships, making it difficult to capture nuanced connections between different elements of the knowledge base.

8.3.2 Prolog

Advantages

- **Powerful Inference Engine:** Prolog's logic programming language offers a powerful inference engine that can deduce logical conclusions based on defined rules. This ability is beneficial for making complex inferences and reasoning about wine selections based on guest preferences and meal characteristics.
- **Expressive Representation:** Prolog allows for a concise representation of knowledge using rules and facts, enabling the capture of intricate relationships and dependencies.

Disadvantages

- **Learning Curve:** Prolog can be challenging to learn and understand, particularly for individuals without prior experience in logic programming. The syntax and concepts may require a significant investment of time and effort.
- **Limited Visual Representation:** Prolog does not offer built-in visualizations, which can make it challenging to grasp the overall structure and connections within the knowledge base.

8.3.3 Knowledge Graph

Advantages

- **Visual Representation:** knowledge graphs provide a visual and interconnected representation of wine knowledge, facilitating a clear understanding of relationships and dependencies.
- **Query and Navigation:** knowledge graphs allow for easy querying and navigation through the data, enabling efficient exploration and retrieval of relevant information.

Disadvantages

- **Time-Consuming Construction:** building a comprehensive knowledge graph can be time-consuming, requiring careful definition of the schema and accurate capturing of the relationships between wines, regions, grapes, tastes, and meals.

- **Schema Design Challenges:** creating a well-defined schema that encompasses all necessary information while maintaining flexibility and extensibility can be a complex task.

8.3.4 Graphical Modeling Language

Advantages

- **Intuitive Representation:** the graphical modeling language offers an intuitive representation of meals and wines through visual elements such as icons, symbols, and diagrams. This enhances the understanding and interpretation of information for both chefs and customers.
- **User-Friendly Design:** the graphical modeling language is designed with a focus on user-friendliness, ensuring that chefs can easily navigate and utilize the language to represent meals and wines effectively.

Disadvantages

- **Development Effort:** designing and implementing a graphical modeling language requires expertise in user interface design and software development. The development process can be time-consuming and resource-intensive.
- **Maintenance and Updates:** as customer preferences and menu offerings evolve, the graphical modeling language may require updates and modifications to ensure its relevance and usability. Maintaining and updating the language can be a complex and ongoing process.

8.4 Final Conclusion

The project highlighted the advantages and disadvantages of each knowledge-based solution. Decision tables offered a structured representation and ease of understanding, while Prolog provided powerful logical reasoning capabilities. Knowledge graphs facilitated visual representations and relationship modeling. The graphical modeling language, on the other hand, allowed for intuitive representation of meals and wines, catering to customer preferences.

In conclusion, the project "Personalized Wine Menu" demonstrated the value of personalized recommendations, effective knowledge representation, context-aware decision-making, and user-friendly interfaces in the realm of wine selection.

The insights gained from this project can guide future endeavors in creating enhanced digital menu systems that provide a tailored and enjoyable dining experience for customers.

9. Sorritelli Greta's Conclusions

9.1 Introduction

In this concluding chapter, I will summarize the findings and outcomes of our project on representing knowledge about wines in a digital menu system. I will discuss the advantages and disadvantages of each task completed, including the creation of knowledge-based solutions using decision tables, Prolog, and knowledge graphs. Additionally, I will highlight the design of a graphical modeling language for meals and wines, catering to customer preferences.

Throughout this project, we focused on addressing the challenges faced by restaurants with digitized menus and assisting customers in selecting wines that align with their preferences and the menu. I recognized the disadvantage of small smartphone screens and the difficulty of obtaining a comprehensive overview, particularly for extensive wine menus. By considering customer preferences, including country, region, grape, and taste, we aimed to enhance the wine selection process.

9.2 Implications and Applications

The knowledge-based solutions we developed offer practical applications in real-world scenarios. Decision tables provide a structured approach to capturing and evaluating decision logic, enabling organizations to automate and optimize decision-making processes. Prolog, with its logical programming capabilities, offers flexible and powerful reasoning for solving complex problems. Knowledge graphs, on the other hand, provide a comprehensive representation of domain knowledge and facilitate effective knowledge management. Then, the graphical modeling language provides a clear and intuitive interface enhancing user experience by visually representing the classes and instances.

9.3 Advantages and Disadvantages of Each Task

9.3.1 Decision Tables

Advantages

I learned that the decision tables provide a **structured approach** to capturing decision logic, **simplifying the process** of defining customer preferences and matching them with suitable wines.

Disadvantages

But on the other hand they can become **complex and difficult to manage** with an extensive range of customer preferences and wine options, they also **do not represent**

the flow of logic of a solution. In our case, it was necessary to use functions of the FEEL language in order to handle output lists as inputs to new tables. In fact, without the help of languages such as FEEL and JavaScript, the possibilities made available by decision tables may be limited.

9.3.2 Prolog

Advantages

Prolog is a **powerful logical programming language** for representing and reasoning about wine selection criteria. It allows **flexible and customizable decision-making rules** based on customer preferences and meal compatibility.

Disadvantages

Prolog, however, **requires a deeper understanding of the logical programming**, potentially limiting its accessibility and ease of use for non-technical users. Coming from a study and use of mainly object-oriented languages, I had some difficulty in changing my way of seeing things.

9.3.3 Knowledge Graphs

Advantages

The knowledge graphs provide a **comprehensive representation** of wine knowledge, including relationships between wines, regions, grapes, and taste profiles. They also allows **efficient knowledge management** and exploration of different wine options, with the possibility of filter the instances and relations thanks to the **SPARQL query language**.

Disadvantages

Some difficulties may be in building and maintaining the knowledge graph because they **require significant effort and expertise**. Ensuring the accuracy and relevance of the knowledge graph's information is crucial for reliable wine recommendations. I also had some problems with the installed **program version and plugin compatibility**, in this case working with Protégé is a bit frustrating.

9.3.4 Graphical Modeling Language

Advantages

Thanks to ADOxx, we were able to design a graphical modeling language, that **enhances user experience by visually representing** meals and wines, making it **easier for customers to understand** and select according to their preferences.

Disadvantages

The aim was to provide a clear and intuitive interface for chefs to input and update information relevant to customer preferences but the developing **requires careful consideration of various factors**, such as **ease of use, scalability**, and accommodating diverse customer preferences.

9.4 Final Conclusion

In conclusion, our project aimed to overcome the challenges faced by restaurants in the digital menu era by providing a knowledge-based system for wine selection.

Through the utilization of Decision Tables we defined a structured and tabular representation of the menu, thanks to Prolog, instead, we used the reasoning capabilities of the language. With the Knowledge Graphs we provided a graphical representation of relations among entities and a way to filter the possibilities through the SPARQL query language. We offered different approaches to support the decision-making process. Each method has its own advantages and disadvantages, which should be considered based on the specific requirements and capabilities of the restaurant.

Last but not least, the design of a Graphical Modeling Language for meals and wines brings a visually appealing and user-friendly aspect to the selection process. By incorporating relevant information for customers to choose wines according to their preferences, we strive to enhance the overall dining experience.

9.4.1 Further Work

Moving forward, further research and development could focus on refining and expanding the knowledge-based system, exploring additional technologies such as machine learning to improve wine recommendations, and incorporating feedback from customers and restaurant staff to continuously enhance the system's performance and usability. Ultimately, our project contributes to facilitating informed wine selections and enriching the dining experiences of restaurant guests.