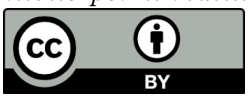
REPORT DI PROGETTO
Progetto "RASTA"

Nicol Alesi - Roberto Zanoni

3 giugno 2025

Indice



1 Introduzione

La presente sezione è dedicata ad approfondire la struttura del progetto proposto dal Gruppo di Lavoro (d'ora in poi GdL) analizzandone la struttura metodologica e il processo di creazione in toto. Per mettere in luce l'avanzamento del progetto e le fasi che lo hanno composto, si ritiene utile suddividere la trattazione sull'aggregazione delle task che sono state svolte.
Come ogni progetto che si rispetti, come fase iniziale il GdL ha creato il backlog di prodotto: l'intenzione è quella di creare una mappa italiana interattiva, suddivisa nelle corrispettive regioni, per la quale selezionando una specifica regione si possa accedere ai principali vini prodotti all'interno della stessa. Nelle specifiche di ogni vino, vi saranno dunque diverse voci che sono facilmente riassumibili, con la logica del dizionario python, nella tabella sottostante.
Per tutta la durata del lavoro il GdL ha adottato metodologie Scrum per l'organizzazione e lo svolgimento del progetto individuando dunque delle specifiche task da portare a termine. Per far ciò si è usufruito di GitHub che, oltre al pannello dedicato, è stato utilizzato per l'aggiornamento e la condivisione dei file tra i membri. Di seguito è possibile accedere alla Repository GitHub creata per il progetto:

Repository GitHub
Cliccando sul logo è possibile accedere alla repository utilizzata per il progetto RASTA

Chiarita dunque la finalità dell'elaborato e le principali caratteristiche, ci si può addentrare nel vivo dello sviluppo partendo dal reperimento dei dati.

LABEL RIFERIMENTO	**SIGNIFICATO**
Vino	Descrive l'insieme di tutti i vini, che possono essere associati ad una certa denominazione (ad esempio, "Abruzzo") e tipologia (ad esempio, "rosso").
Tipologia	Descrive le varie tipologie di vino (ad esempio, "rosso", "bianco", "spumante").
Denominazione	La denominazione di origine protetta (DOP) è un marchio di tutela giuridica della denominazione che viene attribuito dall'Unione europea agli alimenti le cui peculiari caratteristiche qualitative dipendono essenzialmente o esclusivamente dal territorio in cui sono stati prodotti.
Descrizione	Permette di associare una materia prima, relativamente ad una certa quantità (che può essere minima o massima) al prodotto che compone.
ValoreMinimo	Permette di associare un valore minimo per una certa caratteristica o materia prima (ad esempio, "umidità" inferiore al "30%").
ValoreMassimo	Permette di associare un valore massimo per una certa caratteristica o materia prima (ad esempio, "umidità" non superiore al "75%").
MateriaPrima	Descrive l'insieme delle materie prime che possono comporre un certo prodotto (ad esempio, un tipo di vitigno per un vino).
Regione	Le regioni sono, assieme ai comuni, alle città metropolitane, alle province e allo Stato, uno dei cinque elementi costitutivi della Repubblica Italiana. Ogni regione è un ente territoriale con propri statuti, poteri e funzioni secondo i principi fissati dalla Costituzione, come stabilito dall'art. 114, secondo comma del testo.
Provincia	Le province italiane hanno una duplice accezione: * sono circoscrizioni amministrative corrispondenti, salvo alcune eccezioni, all'ambito di competenza delle prefetture, delle questure e degli altri principali uffici periferici del governo italiano (attualmente sono 107)
Città	Un insediamento relativamente grande e permanente, in particolare un grande insediamento urbano

2 Parte Prima - Reperimento dei Dati

Il set di dati utilizzato deriva dalla pagina `etna.istc.cnr.it/food-page` nella quale vengono raccolte diverse ontologie sul tema food e, nel caso analizzato, sul vino. Data la conoscenze ed esperienze pregresse di parte del GdL in merito ad ontologie, uri e semantic web si è optato di procedere all'estrazione dei dati tramite una **Query Sparkle**. Per prima cosa sono stati definiti i prefissi per la ricerca:

```
sparql_query = """
PREFIX wine: <http://w3id.org/food/ontology/disciplinare-vino/>
PREFIX upper: <http://w3id.org/food/ontology/disciplinare-upper/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
```

Definiti i limiti della ricerca, e conseguente estrazione, sono state selezionate le label di interesse:

```
"""
SELECT DISTINCT
?vinoLabel
?tipologiaLabel
?denominazioneLabel
?luogo
?descrizioneLabel
?valoreMinimo
?valoreMassimo
?materiaPrimaLabel
"""
```

Dagli URI delle città, estratte "a ritorso" dai singoli vini, il codice permette di ricavarne la singola provincia, grazie al collegamento intrinseco con il sito DBPedia (non è un DataBase ufficiale ma uno standard semantico collegato a wikipedia).
Viene dunque estratta la provincia come segue:

```
provincia_query = f"""
    PREFIX dbo: <http://dbpedia.org/ontology/>
    SELECT ?provincia WHERE {{
        <{luogo_uri}> dbo:province ?provincia .
    }}
    LIMIT 1
    """
```

L'ultima parte di maggiore rilevanza della query è l'estrazione delle regioni (come in precedenza ma partendo dalla provincia puittosto che dalle regioni) che viene eseguita come segue:

```
query = f"""
            PREFIX dbo: <http://dbpedia.org/ontology/>
            PREFIX dbp: <http://dbpedia.org/property/>
            SELECT ?regione WHERE {{
                OPTIONAL {{ <{luogo}> dbp:seat ?comune .
                ?comune dbo:region ?regione . }}
                OPTIONAL {{ <{luogo}> dbo:region ?regione . }}
                FILTER(BOUND(?regione))
            }}
            LIMIT 1
            """
```

Dalla query riportata sopra vengono estratti **537 Vini** che, tuttavia, non comprendono cinque regioni:

1. Calabria
2. Basilicata
3. Abruzzo
4. Molise
5. Liguria

Tale mancanza di per sè non causa problemi ai fini progettuali ma, facendo riferimento al backlog, farebbe sì che premendo su tali regioni risulterebbe una pagina vuota. Per evitare tale disagio il GdL ha optato, tramite l'utilizzo di *ChatGPT Pro*, per creare 5 schede fittizie basandosi sulla struttura già presente. Aggiungendo questi venticinque nuovi contenuti al conteggio risulta che il DataFrame è composto da **562** schede di vini. Da questa prima fase risulta dunque un file XML completo con un'ampia quantità di dati su cui poter lavorare; prima di iniziare con l'elaborazione è fondamentale creare la struttura DTD per poi procedere alla validazione del file, operazioni che affrontiamo nella parte seguente.

3 Parte Seconda - Struttura Gerarchica e Validazione

Come anticipato in precedenza il file XML necessita di una struttura gerarchica stringente e che si adatti alla totalità dei casi, verificandone la correttezza tramite lo script di validazione.

3.1 Struttura: DTD

Per creare la struttura gerarchica si è proceduto alla creazione del modello DTD con l'obiettivo di forzare la presenza (parziale, generale etc.) degli elementi presenti ma soprattutto l'ordine gerarchico che essi devono assumere. A livello di codice la struttura del file XML può essere rappresentata dal file .dtd come segue:

```
<!ELEMENT Vini (Vino+)>

<!ELEMENT Vino (Tipologia, Denominazione, Descrizione, ValoreMinimo?,
                ValoreMassimo?, MateriaPrima?, Luogo)>
<!ATTLIST Vino nome CDATA #REQUIRED>

<!ELEMENT Tipologia (#PCDATA)>
<!ELEMENT Denominazione (#PCDATA)>
<!ELEMENT Descrizione (#PCDATA)>
<!ELEMENT ValoreMassimo (#PCDATA)>
<!ELEMENT ValoreMinimo (#PCDATA)>
<!ELEMENT MateriaPrima (#PCDATA)>
<!ELEMENT Luogo (Regione+)>
<!ELEMENT Regione (Provincia+)>
<!ATTLIST Regione nome CDATA #REQUIRED>

<!ELEMENT Provincia (Città*)>
<!ATTLIST Provincia nome CDATA #REQUIRED>

<!ELEMENT Città EMPTY>
<!ATTLIST Città nome CDATA #REQUIRED>
```

Dato che il linguaggio usato dal file .dtd non risulta essere particolarmente intuitivo, il GdL ha optato per evidenziare la struttura gerarchica dell'XML tramite una rappresentazione grafica maggiormente intuitiva:

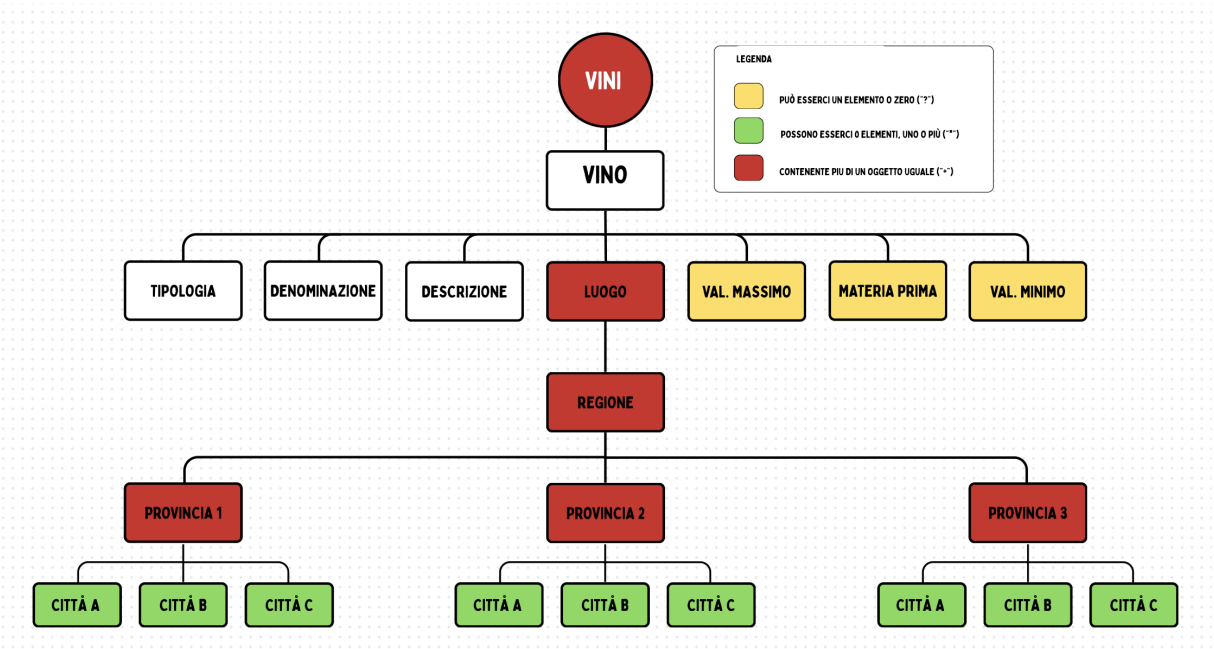


Figura 1: *Elaborazione Propria Gerarchia XML*

3.2 Validazione

Definita la struttura che l'XML deve seguire non manca che valutarne l'attinenza tramite la validazione; per far ciò si è usufruito di un breve script python visto durante il corso che, grazie alla libreria **entree**, valuta l'attinenza dell'XML alla struttura definita del DTD:

```
from lxml import etree
dtd_filename = "../dati/DTD_ViniXML.dtd"
xml_filename = "../dati/vini.xml"

dtd_file = open(dtd_filename, 'r',encoding='utf-8')
xml_file = open(xml_filename, 'r', encoding='utf-8')
xml_root = etree.parse(xml_filename)

try:
    dtd = etree.DTD(dtd_file)
    if dtd.validate(xml_root):
        print("Valid XML!")
    else:
        validation_error = dtd.error_log.filter_from_errors()[0]
        print("XML not valid!\n", validation_error)
except:
  print("DTD non corretto")
```

4 Parte Quarta - Index.html

Nonostante le diverse pagine create siano tutte collegate e navigabili tra di esse, la pagina index.html ha la finalità (esclusivamente per il progetto in questione) di riassumere brevemente i diversi script creati e utilizzati. Tale pagina è dunque iniziale per l'ottica del progetto ma, in un'ipotetica pubblicazione del progetto, l' *home page* sarebbe sicuramente quella che attualmente viene chiamata **home.html**.
Dato il breve contenuto necessario della pagina si è optato per scrivere manualmente in file html (sempre disponibile nella repo) e ne risulta quanto segue:

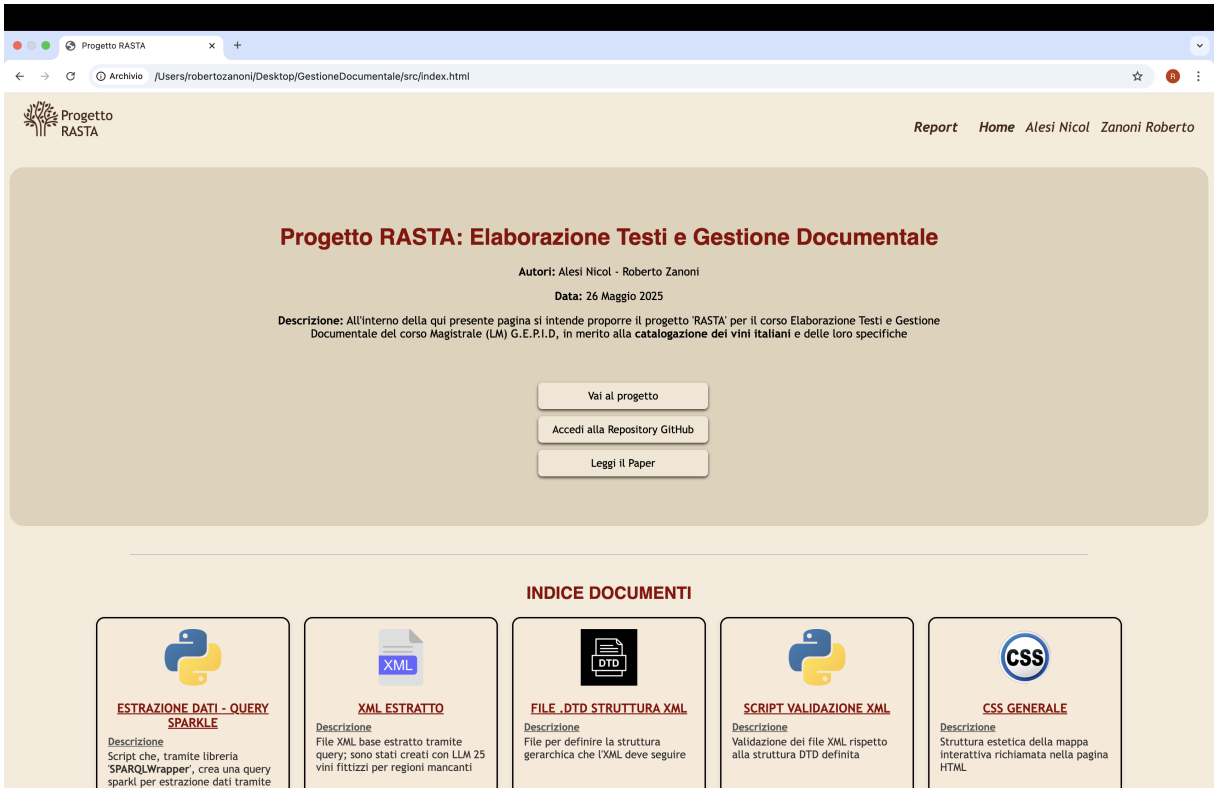


Figura 2: *Screenshot index.html*

Qui, oltre ad avere una breve introduzione al progetto, è possibile entrare nella Repository Git, accedere al qui presente Report di Progetto e visualizzare quella che abbiamo definito essere la homepage. In una seconda sezione, chiamata "Indice Documenti" è possibile accedere e visualizzare da web i diversi codici/file utilizzati per tutto lo svolgimento del progetto (non è presente la totalità dei file in quanto è sempre possibile reperirli dalla repo).
Per **tutte la pagine html** create è stato creato un **singolo file CSS** che, al suo interno, racchiude tutte le impostazioni di layout utilizzate durante lo svolgimento dell'elaborato; per far ciò sono state create **diverse classi div** in base a finalità e struttura estetica.

5 Parte Terza - Home.html

5.1 Initial Page

Validata la struttura e la correttezza del file XML come primo passo, per garantire un layout funzionale ed esteticamente accattivante, il GdL ha reperito i codici preesistenti per inserire la mappa all'interno della pagina grazie al qui presente sito: `https://mapsvg.com/maps`. Avendo inserito la mappa sorge tuttavia una complicazione che necessita l'utilizzo di **JS** (Java Script) per permettere al codice di cogliere su quale regione l'utente sta premendo, "passando" il valore corrispondente alla zona premuta al codice python (ogni regione ha uno specifico path).

```
<script>
  // Funzione per gestire il click su tutti i <path>
  document.addEventListener("DOMContentLoaded", () => {
    // Seleziona tutti i tag <path> con attributo title
    const paths = document.querySelectorAll("path[title]");

    paths.forEach(path => {
      path.addEventListener("click", () => {
        // Legge l'attributo title
        const regione = path.getAttribute("title");

        // Codifica il parametro (es: spazi, caratteri speciali)
        const regioneEncoded = encodeURIComponent(regione);

        // Apre una nuova pagina con il parametro
        window.location.href = `regioni/${regioneEncoded}.html`;
         });
      });
   });
   </script>
```

Inserita dunque la mappa all'interno della pagina il GdL ha proseguito semplicemente popolando la homepage di tutti gli aspetti di layout (tutti inseriti in un **unico CSS per tutte le pagine html**). All'interno della stessa pagina è inoltre possibile scaricare i dati utilizzati in tre diversi formati (i quali necessitano script autonomi di conversione)

1. Download Dati Formato **XML** (dati di partenza)

2. Download Dati Formato **JSON**

```python
import xmltodict
import json

xml_input_path = "../dati/vini.xml"
json_output_path = "../dati/json/vini_completo.json"

def convert_xml_to_json(xml_path, output_path, regione=None):
    with open(xml_path, 'r', encoding='utf-8') as f:
        doc = xmltodict.parse(f.read())
    with open(output_path, 'w', encoding='utf-8') as f:
        json.dump(doc, f, indent=2, ensure_ascii=False)
    print(f"File JSON salvato in: {output_path}")

convert_xml_to_json(xml_input_path, json_output_path)
```

3. Download Dati Formato **CSV**: sempre grazie all'utilizzo della libreria **xmltodict** si mettono a disposizione i dati XML anche in formato CSV (non viene inserito di seguito data la grandezza del codice; reperibile nella repository)

Premendo dunque sulla singola scelta fatta dall'utente, viene **automaticamente attivato il download**; i file tuttavia vengono **scaricati come zip** dunque andranno, una volta averli sulla propria macchina, decompressi per permetterne la visualizzazione. Tale aspetto non risulta essere una vera e propria scelta bensì un'imposizione della **libreria download** (="GFG") che, nel caso in cui i file non siano zippati, visualizza il codice sorgente in una pagina web.
All'interno della stessa pagina, oltre all'interattività della mappa e la possibilità di scaricare i dati, è stato aggiunto un breve indice di navigazione che permette dunque di spostarsi tra le diverse pagine html(home-index-report). a tutto ciò ne deriva una pagina come è possibile vedere nella seguente immagine:

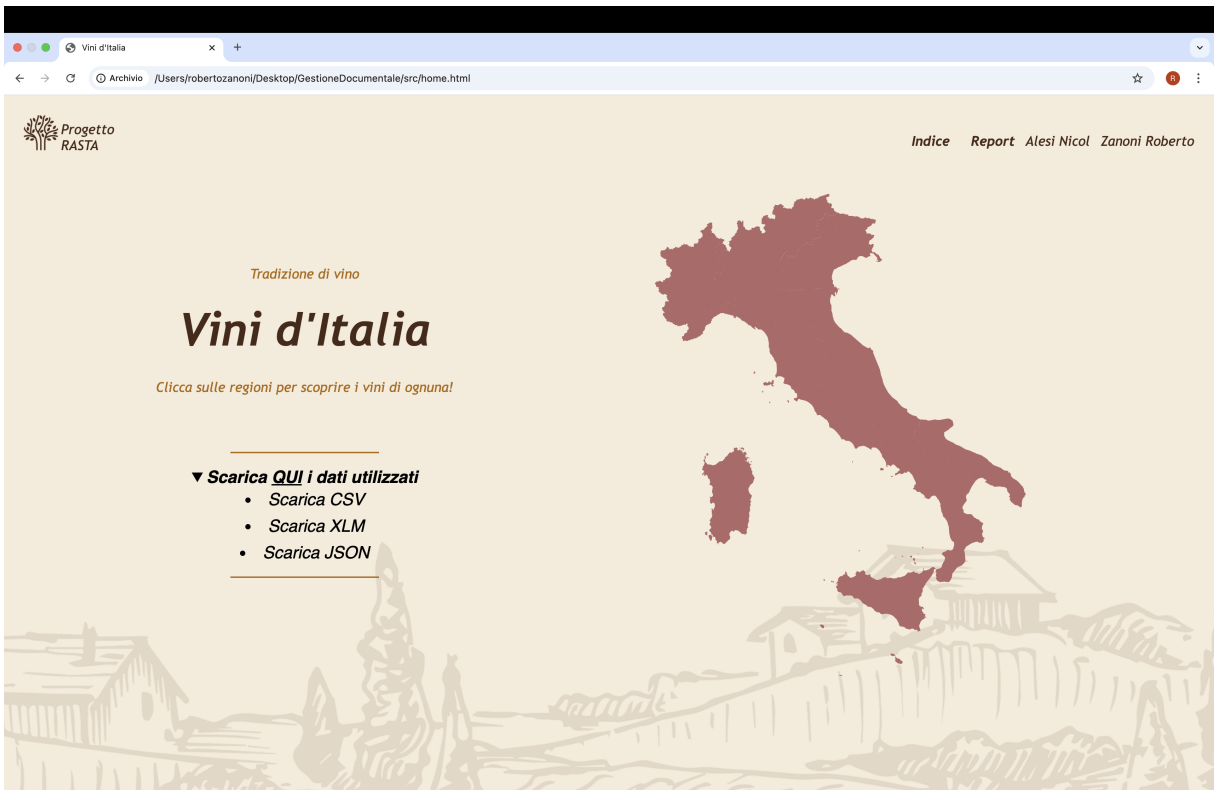


Figura 3: Screenshot schermata "home.html"

5.2 Region Page

Premendo sulla singola regione selezionata, grazie ad uno script python che genera singole pagine regionali html (presente nella repo), si viene reindirizzati sulla pagina **specifica della regione**. Qui sarà possibile scorrere (verticalmente) la lista dei vini della regione selezionata, e dei dati presenti nell'XML, secondo i significati definiti in precedenza; a tale sezione sono state inoltre associate le immagini dei singoli vini che sono stati estratti grazie ad uno script python che usufruisce della libreria **BingImageCrawler** che tramite parametri di ricerca settabili (nel nostro caso nome) cerca nel browser ed esegue azioni pre-settate (per noi download e accoppiamento).
Avendo un ampio numero di dati si è cercato di standardizzare il più possibile il processo ma, nonostante il corretto funzionamento, vi è la consapevolezza che in alcuni casi *le immagini estratte non sempre rappresentano correttamente il vino descritto*, fattore definito non centrale nell'obiettivo del progetto. Da tale elaborazione risulta una visualizzazione simile a quanto segue (esempio Regione Lombardia)

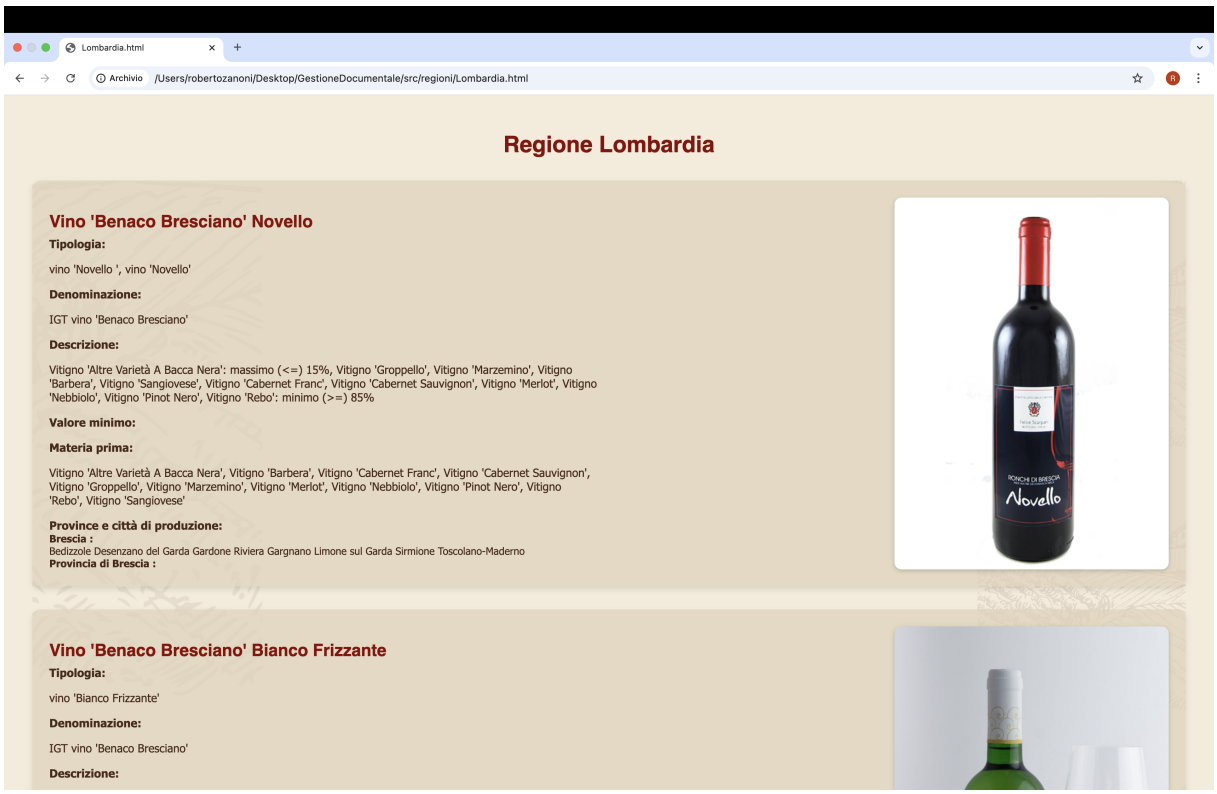


Figura 4: *Screenshot Esempio Regionale: Lombardia*

6 Parte Quinta - Report.html

La terza e ultima pagina alla quale è possibile accedere è quella che è stata chiamata "Report" dove, sempre tramite l'utilizzo del file XML iniziale e i rapporti gerarchici definiti, sono state create delle rappresentazioni grafiche per evidenziare la distribuzione di alcune caratteristiche scelte del DataFrame.
Si è dunque scelto di mettere in luce tre principali caratteristiche del set di dati (scelta condizionata anche dalla capillare presenza di alcuni dati piuttosto che altri):

1. **Distribuzione** dei vini presenti nel DataFrame **in base a divisione Istat** di Nord, Centro e Sud (o "Sconosciuta")
2. **Distribuzione** dei vini analizzati in base alla **Regione di produzione**
3. **Distribuzione** dei vini di ogni regione (20) in base alla **materia prima utilizzata**

Tale rappresentazione grafica è stata creata tramite script python (iterante per la produzione dei grafici per le singole regioni) grazie alle librerie per la creazione di grafici (pandas, seaborn etc.). In breve, ne risulta una pagina (a scorrimento verticale) come di seguito:

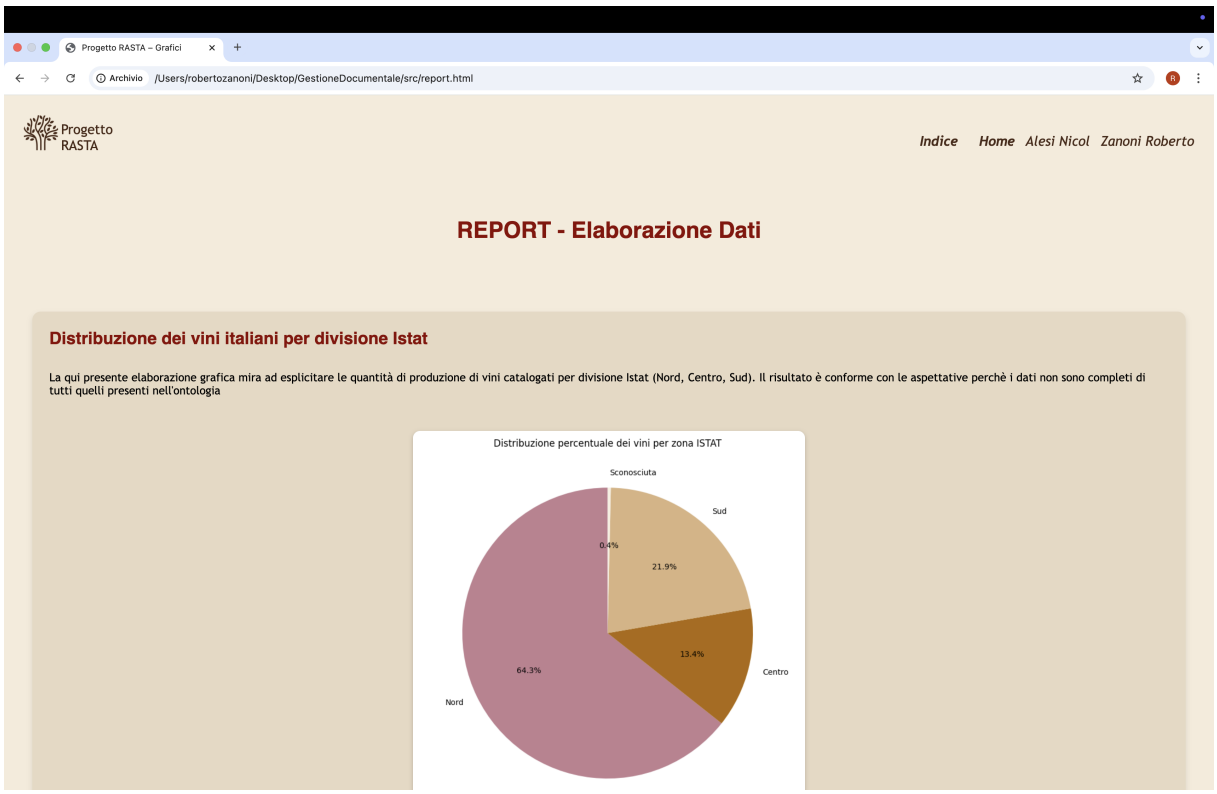


Figura 5: *Screenshot report.html*

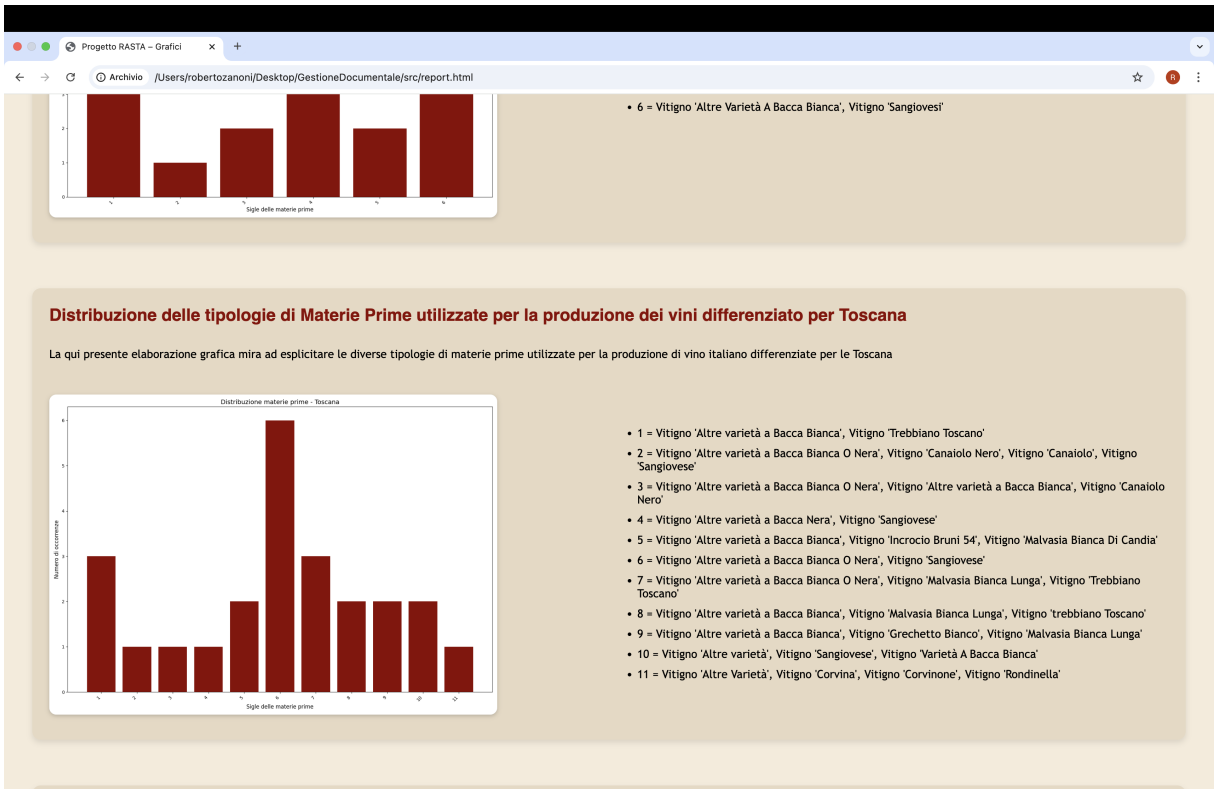


Figura 6: *Screenshot analisi regionale*

7 Conclusioni

Il progetto RASTA ha permesso di affrontare un percorso completo di progettazione e realizzazione di un sistema interattivo basato su dati strutturati in XML, con l'obiettivo di valorizzare le informazioni legate alla produzione vinicola italiana. Attraverso un approccio metodologico di tipo Scrum, il Gruppo di Lavoro ha definito un piano di sviluppo chiaro, articolato in fasi distinte che hanno toccato aspetti fondamentali della gestione dati: dal reperimento e validazione, fino alla rappresentazione visiva.
Dal punto di vista tecnico, il progetto ha integrato in modo coerente conoscenze eterogenee: interrogazioni SPARQL per l'estrazione di dati semantici, costruzione e verifica della struttura DTD, creazione di pagine web dinamiche in HTML arricchite da CSS e JavaScript, e infine rappresentazioni grafiche con Python per l'analisi quantitativa. L'utilizzo di strumenti automatizzati e l'integrazione di librerie esterne (es. `BingImageCrawler`) ha inoltre mostrato come sia possibile potenziare l'esperienza utente pur mantenendo coerenza e rigore nella gestione informativa.

In sintesi, il progetto non solo ha raggiunto l'obiettivo iniziale di creare una mappa interattiva e un sistema consultabile dei vini italiani, ma ha anche dimostrato la fattibilità e l'efficacia di un approccio interdisciplinare fondato su buone pratiche di sviluppo software, gestione dati e comunicazione visiva.