

---

# Análise e Arquitetura de Sistemas Web

MODELAGEM E IMPLEMENTAÇÃO

Com base no dataset X-Wines.csv

---

Ana Caroline Soares Silva

6 de fevereiro de 2026

[github.com/acaroline-ss](https://github.com/acaroline-ss)

[linkedin.com/in/anacarolinessilva](https://linkedin.com/in/anacarolinessilva)

# Resumo

Este projeto apresenta a análise e arquitetura de um sistema digital para catalogação de vinhos, baseado no dataset X-Wines. O trabalho abrange a modelação conceptual do domínio através de um modelo Entidade-Relacionamento, a sua transformação para um modelo relacional normalizado em 3<sup>a</sup> Forma Normal e a implementação de uma base de dados em SQLite.

Foram definidos mecanismos para representar vinhos, produtores, regiões, países, castas, harmonizações gastronómicas e anos de colheita, incluindo relações do tipo um-para-muitos e muitos-para-muitos. O povoamento da base de dados foi realizado a partir de um ficheiro CSV, recorrendo a uma tabela intermédia e a um script em Python para tratamento de atributos multivalorados.

Adicionalmente, foi desenvolvida uma aplicação web em Flask que permite explorar interativamente os dados, consultar tabelas e executar interrogações SQL, disponibilizando uma interface dinâmica e visualmente consistente. O sistema resultante constitui uma base sólida para análise enológica, exploração de padrões de produção e estudo de combinações gastronómicas, demonstrando a aplicação prática de conceitos fundamentais de modelação e arquitetura de bases de dados.

## 1 Modelação

### 1.1 Descrição do Universo

O universo em consideração é um sistema de catálogo de vinhos comerciais. O objetivo é modelar esta informação numa Base de Dados relacional que permita registar, consultar e analisar vinhos de diferentes tipos, origens geográficas e produtores, bem como as suas características enológicas, sugestões de harmonização gastronómica e anos de colheita.

#### Requisitos Funcionais Identificados:

##### 1. Gestão de Vinhos

- Cada vinho é caracterizado por um identificador numérico único (*WineID*) e por uma designação comercial (*WineName*) apresentada no rótulo.
- Para cada vinho é registado o seu tipo categórico (*Type*), distinguindo, por exemplo, vinhos tranquilos (tinto, branco, rosé), espumantes gasificados e vinhos de sobremesa/fortificados.
- O modo de elaboração de cada vinho é classificado em categorias como *varietal* ou *assemblage/blend* (*Elaborate*), podendo ainda identificar blends específicos (por exemplo, Bordeaux blend ou Portuguese blend).
- Para cada vinho é armazenado o teor alcoólico em percentagem de volume (*ABV*), tendo em conta que o valor no rótulo pode ter uma tolerância regulamentar.
- Cada vinho possui ainda uma classificação categórica de corpo (*Body*), como *very light-bodied*, *light-bodied*, *medium-bodied*, *full-bodied* ou *very full-bodied*.
- É igualmente registada a classificação categórica de acidez (*Acidity*), por exemplo *low*, *medium* ou *high*.

##### 2. Gestão de Castas e Harmonização

- Cada vinho está associado a um conjunto de castas de uva utilizadas na sua elaboração (*Grapes*); Como uma mesma casta pode ser utilizada em diversos vinhos, este relacionamento é modelado como uma relação N:M entre as entidades WINE e GRAPE.
- Para cada vinho é registado um conjunto de pratos ou tipos de pratos recomendados para harmonização. Dado que um prato pode harmonizar com vários vinhos, este relacionamento é também representado como N:M entre WINE e DISH.

##### 3. Gestão de Informação Geográfica

- O país de produção de um vinho é determinado pela região à qual a vinícola pertence, através da relação: WINE  $\rightarrow$  WINERY  $\rightarrow$  REGION  $\rightarrow$  COUNTRY
- Cada vinho é produzido por uma única vinícola, identificada por um código único (*WineryID*). Esta relação “um-para-muitos” garante que vários vinhos podem ser produzidos pela mesma vinícola, mas cada vinho tem exatamente um produtor.
- Cada vinícola está localizada numa única região geográfica. Assim, várias vinícolas podem estar inseridas na mesma região, mas cada vinícola só pertence a uma região específica.
- Cada região pertence a um único país, identificado pelo código internacional CountryCode (como PT, US, BR). Este relacionamento organiza a informação geográfica de forma hierárquica: um país contém várias regiões, mas cada região pertence apenas a um país.

#### 4. Gestão de Produtores (Wineries)

- Cada vinho é produzido por uma única vinícola ou produtor, identificada por um identificador numérico (*WineryID*), utilizado como chave estrangeira.
- Para cada vinícola é guardado o seu nome (*WineryName*), que a identifica de forma única no contexto da base de dados.
- Quando disponível, é também registado o endereço eletrónico do website oficial da vinícola (*Website*), permitindo acesso a informação adicional sobre o produtor.
- Uma mesma vinícola pode estar associada a vários vinhos diferentes.

#### 5. Gestão de Anos de Colheita (Vintages)

- Para cada vinho é registada a informação relativa aos anos em que esse vinho foi produzido.
- O atributo multivalorado *Vintages* consiste numa lista de valores, em que cada elemento corresponde a um ano de colheita (representado como um número inteiro) ou, em alternativa, à abreviatura “N.V.” que significa *non-vintage*.
- Um mesmo vinho pode, assim, ter múltiplos anos de colheita associados (por exemplo, 2016, 2018 e 2019), ou ser classificado como non-vintage quando não existe um ano específico declarado no rótulo.

##### 1.1.1 Modelo ER

Após a identificação dos requisitos do universo da Base de Dados, foi desenvolvido o seguinte Modelo Entidade-Relacionamento, que descreve as principais entidades envolvidas no sistema e os relacionamentos entre elas.

##### Entidades-tipo

- **WINE** – Representa os vinhos comerciais do catálogo.
- **WINERY** – Representa as vinícolas responsáveis pela produção dos vinhos.
- **REGION** – Representa as regiões vitivinícolas de produção dos vinhos.
- **COUNTRY** – Representa os países de origem associados à produção dos vinhos.
- **GRAPE** – Representa as variedades de uva utilizadas na elaboração dos vinhos.
- **DISH** – Representa os pratos ou tipos de pratos recomendados para harmonização gastronómica.

### Atributos das Entidades-Tipo

- **WINE**: (WineID, WineName, Type, Elaborate, ABV, Body, Acidity, {Vintages})
- **GRAPE**: (GrapeID, GrapeName)
- **DISH**: (DishID, DishName)
- **WINERY**: (WineryID, WineryName, Website)
- **REGION**: (RegionID, RegionName)
- **COUNTRY**: (Code, Country)

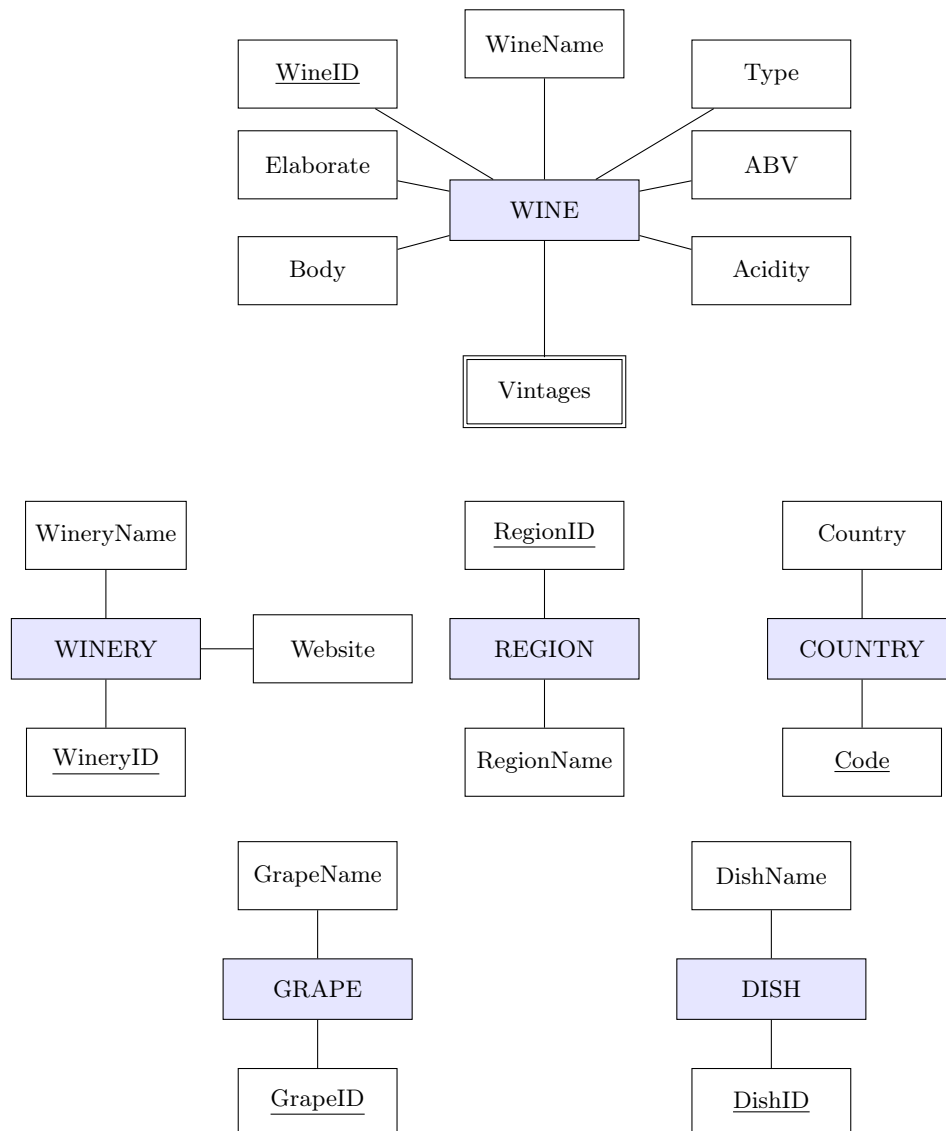


Figura 1: Diagrama ER – Entidades-Tipo e seus atributos

### Relacionamentos

- **PRODUCED\_BY** – Relacionamento 1:N entre WINERY e WINE.
  - Uma vinícola pode produzir vários vinhos, mas cada vinho é produzido por uma única vinícola.
  - Participação total do lado de WINE e parcial do lado de WINERY.

- **LOCATED\_IN** – Relacionamento 1:N entre *REGION* e *WINERY*.
  - Uma região pode incluir várias vinícolas, mas cada vinícola pertence a exatamente uma região.
  - A participação é total do lado de *WINERY*, e parcial do lado de *REGION*.
- **BELONGS\_TO** – Relacionamento 1:N entre *COUNTRY* e *REGION*.
  - Um país pode conter várias regiões vitivinícolas, mas cada região pertence a um único país.
  - A participação é total do lado de *REGION* e parcial do lado de *COUNTRY*.
- **MADE\_WITH** – Relacionamento N:M entre *WINE* e *GRAPE*.
  - Um vinho pode ser elaborado com várias uvas, e uma mesma uva pode ser utilizada em vários vinhos.
  - A participação é total do lado de *WINE* e total do lado de *GRAPE*.
- **PAIRS\_WITH** – Relacionamento N:M entre *WINE* e *DISH*.
  - Um vinho pode harmonizar com vários pratos, e um mesmo prato pode harmonizar com diferentes vinhos.
  - Participação total do lado de *WINE* e total do lado de *DISH*.

**Nota.** Embora seja possível conceber relacionamentos diretos entre *WINE-REGION* e *WINE-COUNTRY*, tais ligações são consideradas redundantes no modelo ER conceitual. A associação entre um vinho e sua respectiva região ou país pode ser inferida através dos relacionamentos já definidos ( $COUNTRY \rightarrow REGION \rightarrow WINERY \rightarrow WINE$ ), evitando duplicação estrutural e preservando a clareza do modelo.

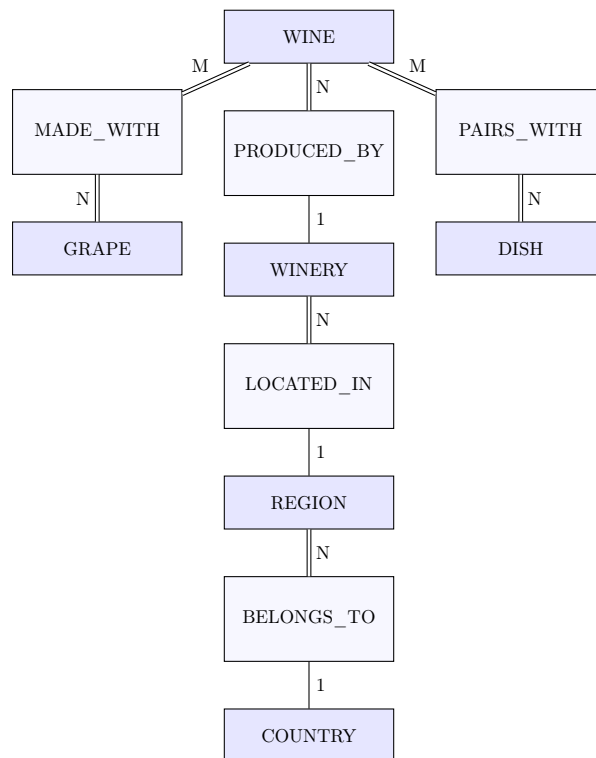


Figura 2: Diagrama ER — relacionamentos entre entidades-tipo

## 1.2 Modelo Relacional

### 1.2.1 Mapeamento do Modelo ER para o Modelo Relacional

O modelo relacional foi obtido a partir do modelo Entidade-Relacionamento aplicando as regras formais de mapeamento. As entidades-tipo originaram tabelas independentes, enquanto os relacionamentos 1:N foram convertidos em chaves externas, e os relacionamentos N:M (resultantes dos atributos multivalorados) deram origem a tabelas associativas.

**Entidades fortes → tabelas.**

- **COUNTRY**

A entidade *Country* originou a tabela:

COUNTRY(Code, Country)

onde Code é a chave primária.

- **REGION**

A entidade *Region* deu origem à tabela:

REGION(RegionID, RegionName, CountryCode)

em que RegionID é a chave primária e CountryCode funciona como chave externa para COUNTRY(Code), representando o relacionamento 1:N entre país e região.

- **WINERY**

A entidade *Winery* foi mapeada para:

WINERY(WineryID, WineryName, Website, RegionID)

onde WineryID é a chave primária e RegionID referencia REGION(RegionID), refletindo a ligação 1:N entre região e vinícola.

- **WINE**

A entidade *Wine* originou a tabela:

WINE(WineID, WineName, Type, Elaborate, ABV, Body, Acidity, RegionID, WineryID)

em que WineID é a chave primária. O atributo WineryID atua como chave externa para WINERY(WineryID), representando o relacionamento 1:N entre vinícola e vinho.

- **GRAPE**

A entidade *Grape* foi convertida na tabela:

GRAPE(GrapeID, GrapeName)

onde GrapeID é a chave primária.

- **DISH**

A entidade *Dish* originou a tabela:

DISH(DishID, DishName)

onde DishID é a chave primária.

**Atributos multivalorados / relacionamentos N:M → tabelas associativas.** Os relacionamentos N:M presentes no modelo ER (entre *Wine-Grape* e *Wine-Dish*) originam, no modelo relacional, tabelas associativas com chaves primárias compostas. O atributo multivalorado *Vintages* é igualmente convertido numa tabela própria para garantir atomicidade:

- **WINE\_GRAPE(WineID, GrapeID)**  
Chave primária: (WineID, GrapeID)  
WineID é chave externa para WINE(WineID)  
GrapeID é chave externa para GRAPE(GrapeID)  
Representa o relacionamento N:M entre vinhos e castas de uva.
- **WINE\_DISH(WineID, DishID)**  
Chave primária: (WineID, DishID)  
WineID referencia WINE(WineID)  
DishID referencia DISH(DishID)  
Representa o relacionamento N:M entre vinhos e pratos harmonizados.
- **WINE\_VINTAGE(WineID, VintageID)**  
Chave primária composta: (WineID, VintageID).  
WineID é chave externa para WINE(WineID) e VintageID é chave externa para VINTAGE(VintageID).  
Esta tabela associativa resulta da decomposição do atributo multivalorado *Vintages* do modelo ER, permitindo, no modelo relacional, ligar cada vinho a um ou mais registos da entidade VINTAGE, a qual representa os diferentes anos de colheita associados ao vinho (incluindo o caso especial “N.V.”).

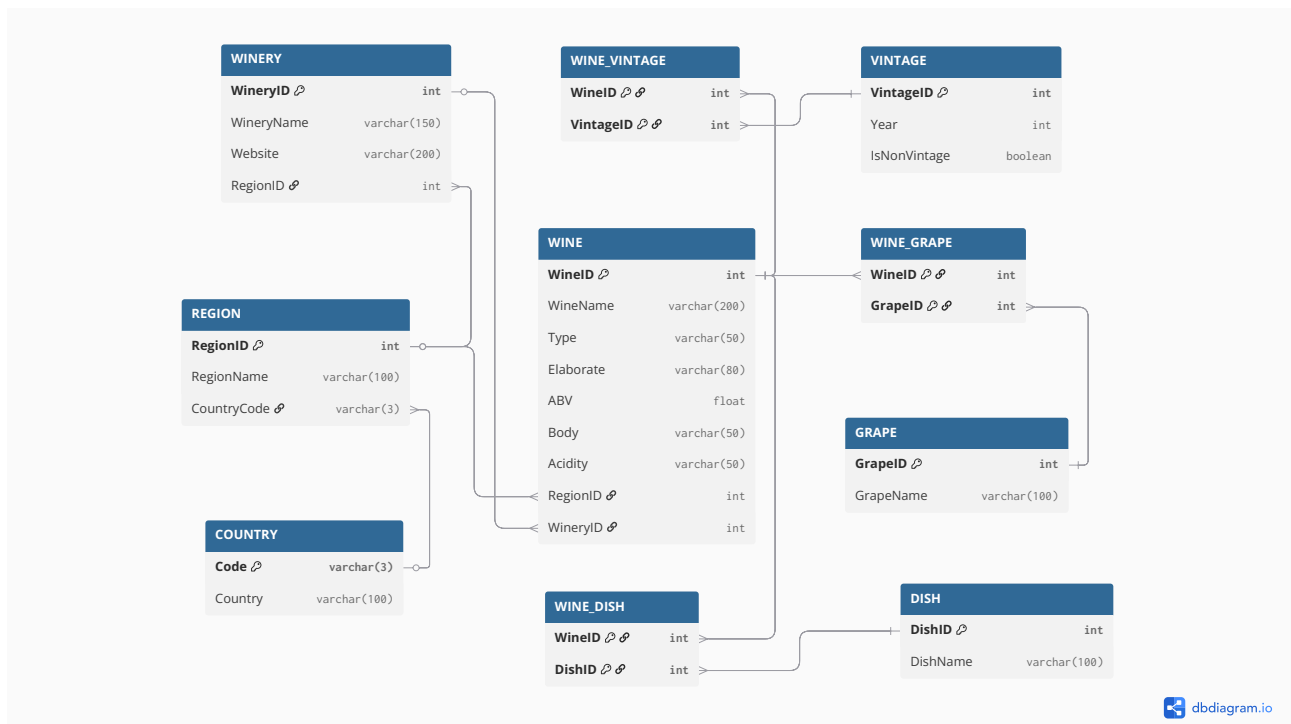


Figura 3: Modelo Relacional

### 1.2.2 Verificação da 3ª Forma Normal

O modelo relacional encontra-se na 3ª Forma Normal, uma vez que cumpre os seguintes requisitos:

- **Todos os atributos são atômicos.** Não existem atributos multivalorados nas tabelas. O atributo multivalorado *Vintages* foi decomposto nas tabelas VINTAGE e WINE\_VINTAGE, garantindo atomicidade.

- **Não há dependências parciais.** Nas tabelas com chave primária simples, todos os atributos não-chave dependem totalmente da chave. Nas tabelas associativas com chave primária composta (WINE\_GRAPE, WINE\_DISH, WINE\_VINTAGE), não existem atributos não-chave: cada tuplo é definido exclusivamente pelo par que compõe a chave primária.
- **Não há dependências transitivas dentro da mesma relação.** Atributos como país e região não são armazenados redundantemente em WINE; tais informações são obtidas através das chaves externas WINE → WINERY → REGION → COUNTRY. Assim, nenhum atributo não-chave depende de outro atributo não-chave dentro da mesma tabela.

Deste modo, o modelo encontra-se devidamente normalizado, preservando as dependências funcionais e evitando redundância desnecessária.

## 2 Povoamento de Tabelas

Nesta secção descreve-se o processo de povoamento da base de dados a partir do ficheiro *X-Wines.csv*, seguindo o modelo relacional definido previamente. O objetivo foi garantir que todos os dados fossem corretamente importados, normalizados e distribuídos pelas tabelas correspondentes, respeitando-se as chaves primárias, chaves estrangeiras e os tipos de dados apropriados.

### 1) Processo de povoamento

O povoamento foi realizado em três fases principais.

#### Fase A — Importação do ficheiro CSV para a tabela RawWine

Inicialmente foi criada uma tabela intermédia, **RawWine**, contendo todas as colunas presentes no ficheiro *X-Wines.csv*. Esta tabela funcionou como área de *staging*, permitindo preparar os dados antes da sua distribuição pelas restantes entidades do modelo.

A importação foi realizada no *SQLiteStudio*, recorrendo à funcionalidade de importação de CSV, configurada com:

- Separador de campos: vírgula (,)
- Codificação: UTF-8
- Primeira linha considerada cabeçalho

Com a tabela **RawWine** preenchida, tornou-se possível extrair valores distintos e organizar a informação segundo o modelo normalizado.

#### Fase B — Povoamento das tabelas principais

A partir da tabela **RawWine**, foram executadas instruções SQL do tipo:

```
INSERT INTO TabelaDestino (...)
SELECT DISTINCT ...
FROM RawWine;
```

Estas instruções permitiram:

- Extrair os países únicos para a tabela **Country**;
- Extrair regiões e relacioná-las ao respetivo país (**Region**);
- Extrair vinícolas (**Winery**);
- Preencher a tabela **Wine**, convertendo tipos de dados sempre que necessário (por exemplo, ABV para REAL).



## Fase C — Tratamento de listas e povoamento das tabelas associativas

As colunas **Grapes**, **Harmonize** e **Vintages** contêm listas de valores separadas por vírgulas. Para compatibilizar estes atributos multivalorados com o modelo relacional, tornou-se necessário decompor essas listas e distribuí-las pelas respectivas tabelas:

- **Grape** e **WineGrape**, para o conjunto de castas;
- **Dish** e **WineDish**, para combinações gastronómicas;
- **Vintage** e **WineVintage**, para os anos de colheita (incluindo o caso especial N.V.).

Como o SQLite não possui mecanismos nativos para dividir listas, foi desenvolvido um script Python (`populate_associative.py`) utilizando a biblioteca `sqlite3`. Este script:

- Lê cada registo da tabela **RawWine**;
- Separa as listas de uvas, pratos e anos de colheita;
- Insere entradas únicas nas tabelas **Grape**, **Dish** e **Vintage** usando `INSERT OR IGNORE`;
- Cria as correspondentes relações N:M nas tabelas **WineGrape**, **WineDish** e **WineVintage**;
- Tratou o valor especial “N.V.” (non-vintage) através do atributo **IsNonVintage**.

### 3) Número de linhas por tabela

A Tabela 1 apresenta o número total de entradas obtidas após o processo de povoamento.

Tabela	Nº de Linhas
Country	31
Region	324
Winery	792
Wine	1007
Grape	341
WineGrape	1584
Dish	100
WineDish	4199
Vintage	73
WineVintage	28241

Tabela 1: Número de registos em cada tabela após o povoamento.

## 3 Interrogações SQL

Nesta secção apresentam-se dez interrogações realizadas à base de dados. Para cada consulta descreve-se o objetivo, apresenta-se o código SQL correspondente e inclui-se um pequeno extrato do resultado obtido.

### I.1 — Número de vinhos por país

**Objetivo:** Determinar quantos vinhos existem por país representado na base de dados.

```
SELECT c.CountryName AS Country,
       COUNT(w.WineID) AS NumWines
FROM Wine w
JOIN Region r ON w.RegionID = r.RegionID
JOIN Country c ON r.CountryCode = c.Code
```

```
GROUP BY c.CountryName
ORDER BY NumWines DESC;
```

Extrato do resultado:

Country	NumWines
...	...

## I.2 — Número de vinhos por região

**Objetivo:** Identificar as regiões com maior diversidade de vinhos.

```
SELECT r.RegionName AS Region,
       c.CountryName AS Country,
       COUNT(w.WineID) AS NumWines
FROM Wine w
JOIN Region r ON w.RegionID = r.RegionID
JOIN Country c ON r.CountryCode = c.Code
GROUP BY r.RegionName, c.CountryName
ORDER BY NumWines DESC;
```

Extrato do resultado:

Region	Country	NumWines
...	...	...

## I.3 — Funções de agregação sobre o teor alcoólico

**Objetivo:** Obter o valor médio, o número de vinhos, o total, o máximo e o mínimo do teor alcoólico.

```
SELECT AVG(ABV) AS MediaABV,
       COUNT(*) AS NumWines,
       SUM(ABV) AS TotalABV,
       MAX(ABV) AS MaxABV,
       MIN(ABV) AS MinABV
FROM Wine
WHERE ABV IS NOT NULL;
```

Extrato do resultado:

MediaABV	NumWines	TotalABV	MaxABV	MinABV
...	...	...	...	...

## I.4 — Castas usadas em muitos vinhos

**Objetivo:** Identificar as castas que aparecem em mais de 40 vinhos.

```
SELECT g.GrapeName,
       COUNT(wg.WineID) AS NumWines
FROM Grape g
JOIN WineGrape wg ON g.GrapeID = wg.GrapeID
GROUP BY g.GrapeName
HAVING COUNT(wg.WineID) > 40
ORDER BY NumWines DESC, g.GrapeName;
```

Extrato do resultado:

GrapeName	NumWines
...	...

## I.5 — Vinhos que harmonizam com pratos contendo “Beef”

**Objetivo:** Listar os vinhos que harmonizam com qualquer prato que contenha a palavra *Beef*.

```
SELECT d.DishName,
       w.WineName,
       wy.WineryName,
       c.CountryName
FROM Wine w
JOIN WineDish wd ON w.WineID = wd.WineID
JOIN Dish d ON wd.DishID = d.DishID
JOIN Winery wy ON w.WineryID = wy.WineryID
JOIN Region r ON w.RegionID = r.RegionID
JOIN Country c ON r.CountryCode = c.Code
WHERE d.DishName like '%Beef%'
ORDER BY w.WineName
```

**Extrato do resultado:**

DishName	WineName	WineryName	CountryName
...	...	...	...

## I.6 — Vinhos com a segunda maior graduação alcoólica

**Objetivo:** Encontrar os vinhos cuja graduação alcoólica é a segunda mais alta na base de dados.

```
SELECT WineName,
       ABV
FROM Wine
WHERE ABV = (
    SELECT MAX(ABV)
    FROM Wine
    WHERE ABV < (SELECT MAX(ABV) FROM Wine)
);
```

**Extrato do resultado:**

WineName	ABV
...	...

## I.7 — Vinhos cujas vinícolas não possuem website

**Objetivo:** Identificar os vinhos associados a vinícolas que não possuem website registrado.

```
SELECT
    w.WineID,
    w.WineName,
    wy.WineryName
FROM Wine w
JOIN Winery wy ON wy.WineryID = w.WineryID
WHERE w.WineryID NOT IN (
    SELECT WineryID
    FROM Winery
    WHERE Website IS NOT NULL AND Website <> ''
)
)
```

**Extrato do resultado:**

WineID	WineName	WineryName
...	...	...

## I.8 — Castas comuns a vinhos tintos e brancos

**Objetivo:** Determinar quais castas de uva (**Grape**) são utilizadas simultaneamente em vinhos do tipo Red e do tipo White.

```
SELECT DISTINCT g.GrapeName
FROM Grape g
JOIN WineGrape wg ON g.GrapeID = wg.GrapeID
JOIN Wine w      ON w.WineID = wg.WineID
WHERE w.Type = 'Red'
INTERSECT
SELECT DISTINCT g.GrapeName
FROM Grape g
JOIN WineGrape wg ON g.GrapeID = wg.GrapeID
JOIN Wine w      ON w.WineID = wg.WineID
WHERE w.Type = 'White';
```

**Extrato do resultado:**

GrapeName
...

## I.9 — Castas exclusivas de um único vinho

**Objetivo:** Identificar as castas (**Grape**) que são utilizadas em exatamente um vinho, permitindo destacar variedades pouco comuns ou exclusivas.

```
SELECT g.GrapeName
FROM Grape g
JOIN WineGrape wg ON g.GrapeID = wg.GrapeID
GROUP BY g.GrapeName
HAVING COUNT(DISTINCT wg.WineID) = 1
ORDER BY g.GrapeName;
```

**Extrato do resultado:**

GrapeName
...

## I.10 — Produtores com muitos vinhos

**Objetivo:** Calcular o número total de vinhos por produtor, mostrando apenas as vinícolas que produzem mais de 5 vinhos.

```
SELECT wy.WineryName,
       COUNT(w.WineID) AS NumWines
FROM Winery wy
JOIN Wine w ON w.WineryID = wy.WineryID
GROUP BY wy.WineryName
HAVING COUNT(w.WineID) > 5
ORDER BY NumWines DESC, wy.WineryName;
```

## Extrato do resultado:

WineryName	NumWines
...	...

## 4 Implementação da Aplicação Python

A aplicação web desenvolvida para este projeto foi construída em Flask, um microframework em Python que permite estruturar a lógica da aplicação através de rotas (endpoints) e gerar páginas dinâmicas integradas com a base de dados `wines.db`, implementada em SQLite.

### Estrutura da Aplicação

A aplicação define um endpoint principal, que funciona como página de entrada. Esta página disponibiliza acesso direto às tabelas e às dez interrogações propostas.

Para cada tabela do esquema relacional foi implementado um par de endpoints:

- `/T/` — apresenta todos os registos existentes na tabela `T`, exibindo a informação em formato tabular e fornecendo ligações diretas para aceder a cada registo individual;
- `/T/k/` — apresenta todos os detalhes associados ao registo cuja chave primária tem valor `k`.

Este mecanismo foi implementado para todas as entidades da base de dados: `Wine`, `Winery`, `Region`, `Country`, `Grape`, `Dish` e `Vintage`. As páginas individuais incluem ligações dinâmicas para registos de outras tabelas, explorando as relações existentes no modelo. Por exemplo, na página de detalhes de um vinho é possível navegar para a sua região, vinícola, castas utilizadas, pratos de harmonização e anos de colheita.

### Interrogações SQL

Foram implementados endpoints específicos para as dez interrogações pedidas. Cada interrogação possui uma página dedicada contendo:

- uma descrição textual do objetivo;
- a execução da consulta SQL correspondente;
- uma tabela dinâmica gerada a partir dos resultados obtidos.

### Sistema de Templates e Interface

A aplicação utiliza um sistema de templates HTML, onde o Flask insere dinamicamente os resultados das consultas SQL. Esta abordagem separa a lógica da aplicação da sua apresentação visual, mantendo consistência entre todas as páginas.

O estilo visual foi definido num ficheiro dedicado, `style.css`, criado especificamente para este projeto. Este ficheiro implementa uma identidade gráfica baseada em tons de vinho tinto, incluindo cartões interativos, botões estilizados, grelhas de navegação e uma disposição responsiva.

## 5 Conclusão

Este projeto permitiu aplicar, de forma integrada, conceitos de análise e arquitetura de sistemas de informação, com especial enfoque na utilização de SQL como ferramenta central para exploração e análise de dados. A partir do dataset X-Wines, foi concebido um modelo Entidade-Relacionamento que serviu de base à construção de um esquema relacional normalizado em 3<sup>a</sup> Forma Normal, assegurando integridade referencial, atomicidade e eliminação de redundâncias.

O processo de povoamento evidenciou a importância da preparação e transformação dos dados, recorrendo a uma tabela intermédia e a scripts em Python para decompor atributos multivalorados e estruturar a informação de forma consistente. Uma vez estabelecida a base de dados, as interrogações SQL desempenharam um papel fundamental na extração de conhecimento, permitindo analisar padrões de produção, relações entre castas e vinhos, preferências de harmonização gastronómica e distribuição geográfica.

A implementação da aplicação web em Flask possibilitou a integração direta entre a camada de dados e a interface de utilizador, tornando as consultas SQL acessíveis de forma dinâmica e interativa. Esta abordagem reforçou a compreensão do papel do SQL não apenas como linguagem de manipulação de dados, mas também como instrumento de apoio à análise e tomada de decisão.

De forma global, o trabalho demonstrou como uma arquitetura bem definida, aliada a consultas SQL estruturadas, permite transformar dados brutos em informação relevante. O sistema desenvolvido estabelece uma base sólida para futuras extensões, incluindo análises estatísticas mais avançadas, mecanismos de filtragem e pesquisa otimizados ou visualizações de dados, evidenciando o potencial dos sistemas relacionais como suporte à análise de dados em contextos reais.