UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURÍ
Bacharelado em Sistemas de Informação
Rafael de Oliveira Bernardes

SISTEMA PARA CLASSIFICAÇÃO DE ESTILOS DE CERVEJAS

Diamantina, MG
2021

Rafael de Oliveira Bernardes

SISTEMA PARA CLASSIFICAÇÃO DE ESTILOS DE CERVEJAS

Trabalho de Conclusão de Curso apresentado à Faculdade de Ciências Exatas da Universidade Federal dos Vales do Jequitinhonha e Mucuri como requisito parcial para a obtenção do grau de bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Alessandro Vivas Andrade

Diamantina, MG
2021

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

FOLHA DE APROVAÇÃO

Rafael de Oliveira Bernardes

SISTEMA PARA CLASSIFICAÇÃO DE ESTILOS DE CERVEJAS

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri, como requisitos parcial para conclusão do curso.

Orientador: Prof. Dr. Alessandro Vivas Andrade

Data de aprovação: 17/09//2021

Profª. Dra. Claudia Beatriz Berti

Faculdade de Ciências Exatas - UFVJM

Prof. Dr. Rafael Santin

Faculdade de Ciências Exatas - UFVJM

Documento assinado eletronicamente por **Alessandro Vivas Andrade**, **Servidor**, em 17/09/2021, às 16:50, conforme horário oficial de Brasília, com fundamento no art. 6°, § 1°, do Decreto n° 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Claudia Beatriz Berti**, **Servidor**, em 17/09/2021, às 16:51, conforme horário oficial de Brasília, com fundamento no art. 6°, § 1°, do Decreto n° 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rafael Santin**, **Servidor**, em 20/09/2021, às 07:46,

conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufvjm.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0462417** e o código CRC **4270DE6C**.

Referência: Processo nº 23086.008168/2021-07 SEI nº 0462417

AGRADECIMENTOS

Os agradecimentos principais são direcionados primeiramente aos meus pais pelo grandioso suporte no início da minha jornada na universidade, permitindo que eu pudesse me manter no curso. Às grandes amizades que fiz no decorrer do curso por me aturarem e pelos bons momentos que guardarei para sempre. Por fim, a todos professores da UFVJM que tive o prazer de assistir às aulas obter o conhecimento necessário para o mercado de trabalho e para a sequência do meu curso.

Sem nenhum de vocês, não teria chegado até aqui!

RESUMO

O interesse da população por cervejas artesanais tem crescido fortemente nos últimos anos no Brasil, aumentando consideravelmente o seu mercado. O objetivo central desse trabalho é estudar a composição e manufatura dessas cervejas, além de estudar tecnologias de desenvolvimento Web. Propõe-se, assim, apresentar o desenvolvimento de um sistema capaz de classificar diferentes receitas com base em dados enviados pelos usuários para disponibilizar uma ferramenta acessível e ágil para o auxílio no processo de fabricação de cervejas artesanais. Através desse estudo, pode-se definir os principais fatores no processo de fabricação de cervejas artesanais, bem como apresentar tecnologias de desenvolvimento ágil e conciso.

Palavras-chave: Cervejas artesanais. Python. Sistemas Web.

ABSTRACT

The population's interest in craft beers has grown strongly in recent years in Brazil, considerably increasing its market. The main objective of this work is to study the composition and manufacture of these beers, in addition to studying Web development technologies. Thus, it is proposed to present the development of a system capable of classifying different recipes based on data sent by users to provide a tool accessible and agile to help in the craft beer manufacturing process. Through this study, it is possible to define the main factors in the craft beer manufacturing process, as well as present agile and concise development technologies.

Keywords: Homemade Brew. Python. Web systems.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O mercado de cervejas artesanais tem crescido surpreendentemente nos últimos 10 anos. Segundo dados de 2020, estima-se que o número de cervejarias artesanais no Brasil saltou de 70 para 900 (CERVEJA ARTESANAL TEM MERCADO QUENTE PARA EMPREENDER E CARREIRAS EM ALTA | VOCÊ S/A..., 2021). Visando esse aumento no interesse da população em produzir a sua própria cerveja, este trabalho fornecerá um classificador de receitas de cervejas artesanais.

Na fabricação das cervejas, diversos fatores da receita e da manufatura da mesma podem afetar a classificação da mesma perante ao estilo buscado com a receita. Assim o classificador serviria como auxílio tanto para aquele que faz uma cerveja seguindo uma determinada receita e deseja saber se a mesma se encontra no estilo desejado, quanto para o produtor que não sabe ao certo qual estilo está produzindo e deseja identificar o resultado do processo.

Dessa forma, o objetivo do trabalho é desenvolver um sistema Web para a classificação de cervejas artesanais, de modo a amenizar os problemas encontrados pelos produtores de cerveja nos casos citados.

Para isso, foi-se utilizado principalmente o framework de desenvolvimento Web Django na linguagem Python. Além de ser uma tecnologia em alta no mercado, o Django permitiu uma alta produtividade no desenvolvimento do sistema, tendo toda a lógica de classificação sido efetuada através do mesmo.

Quanto aos dados utilizados pelo sistema para a classificação, eles foram retirados dos sites oficiais das associações que regulamentam a produção de cerveja no mundo, de modo a garantir a exatidão dos resultados, e compilados em um arquivo JSON que alimentará o sistema.

No próximo capítulo, será apresentado uma pesquisa sobre a produção de cerveja e suas peculiaridades. Em seguida, é feita a apresentação da linguagem Python, bem como o estudo sobre linguagens de programação. No capítulo 4, tem-se a explicação dos conceitos e funcionamento de frameworks e do Django. Por fim, é apresentado o sistema desenvolvido na totalidade, seguido das conclusões do trabalho.

1.0.1 Objetivo Geral

Este trabalho tem como objetivo geral o desenvolvimento de um Sistema Web para Classificação de Cervejas Artesanais que auxilie os produtores na avaliação de suas respectivas cervejas.

1.0.2 Metodologia

Para este projeto de pesquisa, utilizaram-se os sites de organizações de cerveja pelo mundo de modo a coletar-se os dados referentes aos estilos de cervejas, podendo assim montar uma base robusta com parâmetros definidos para cada um deles. Para o desenvolvimento do

sistema Web, foi utilizado o framework Django por conta de sua simplicidade e agilidade no processo.

O ambiente de desenvolvimento integrado (IDE) utilizado foi o Visual Studio Code, onde foi feita toda a edição de código referente ao sistema. Para o armazenamento dos dados foi utilizado um arquivo no formato JSON, bem como Python, Javascript para implementação do código.

1.0.3 Metodologia

Este trabalho foi dividido em seis capítulos. No primeiro capítulo temos uma introdução do tema, a motivação, objetivos e a metodologia utilizada. No segundo capítulo temos um estudo mais aprofundado sobre a produção da cerveja e os fatores levados em questão no momento da classificação. No terceiro capítulo é feita a apresentação da linguagem Python, bem como um estudo sobre linguagens de programação. No quarto capítulo, tem-se a explicação dos conceitos e funcionamento de frameworks e do Django. No quinto capítulo é apresentado o sistema desenvolvido e, por fim, no sexto capítulo encontramos as considerações finais sobre o trabalho.

2 ESTILOS DE CERVEJA

Partindo da definição básica, a cerveja é uma bebida alcoólica de extratos obtidos por cozimento de um cereal maltado (cereal germinado), cevada normalmente maltada e com a adição de lúpulo (MÜLLER, 2002). Atualmente, é a bebida alcoólica mais consumida no mundo e a terceira bebida geralmente, atrás somente do chá e do café (AS 10 BEBIDAS MAIS CONSUMIDAS NO MUNDO | DIÁRIO DO ESTADO..., 2021).

Como citado por Müller (2002), a cerveja possui 91% de água na sua composição básica. Existem ainda outros três ingredientes que servem de fórmula básica para sua produção. São eles a água, o malte, o lúpulo e as leveduras. Estes ingredientes devem ser selecionados conforme a receita que se deseja obter, tendo especificações para cada um dos componentes.

Com o objetivo de diversificar o resultado das receitas, são usados diversos outros ingredientes como frutas, trigo e milho, alterando desde a cor até o sabor da cerveja.

Na produção, existem 7 etapas principais na realização do processo:

- **Moagem** — O malte é moído para liberação do amido em seu interior
- **Mosturação** — O malte será misturado com água quente para que enzimas transformem o amido em açúcares fermentáveis.
- **Filtragem** — O amido transformado será filtrado para separação de cascas e bagaços do mosto da cerveja.
- **Fervura** — Nesta fase, o mosto recebe a adição do lúpulo, que dará amargor e aroma à cerveja, e será fervido para eliminar impurezas.
- **Resfriamento** — Resfriar o mosto para inoculação da levedura
- **Fermentação** — Os açucares serão consumidos e transformados em álcool e CO_2 .
- **Envase** — A cerveja será armazenada em barris ou garrafas.

2.0.1 A Cor da Cerveja

A cor é o principal elemento visual da cerveja, é aquele que irá despertar o interesse nas pessoas e o primeiro critério de julgamento de uma cerveja, afinal, a primeira avaliação se realiza com os olhos.

Ela também é diretamente ligada ao estilo da cerveja, variando em uma escala de amarelo palha até preto opaco. Da cor também podemos inferir a combinação de maltes no processo de fabricação e o seu grau de torrefação.

Hoje, existem duas escalas principais de definição de cores de cerveja, são elas a europeia EBC(European Brewing Convention) (HOME - EUROPEAN BREWERY CONVENTION..., 2021) e a americana SRM(Standard Reference Method) (WHAT IS SRM (STANDARD

REFERENCE METHOD)? | AMERICAN HOMEBREWERS ASSOCIATION..., 2021). Essas duas especificações de cores são derivadas do mesmo dado de espectro único: absorção de 1 cm à 430 nm (BAMFORTH, 2016).

Por serem relacionados, estes dois sistemas podem ser convertidos de um para o outro utilizando equações. São elas:

- **Escala EBC** — SRM × 1.97.
- **Escala SRM** — EBC × 0.508.

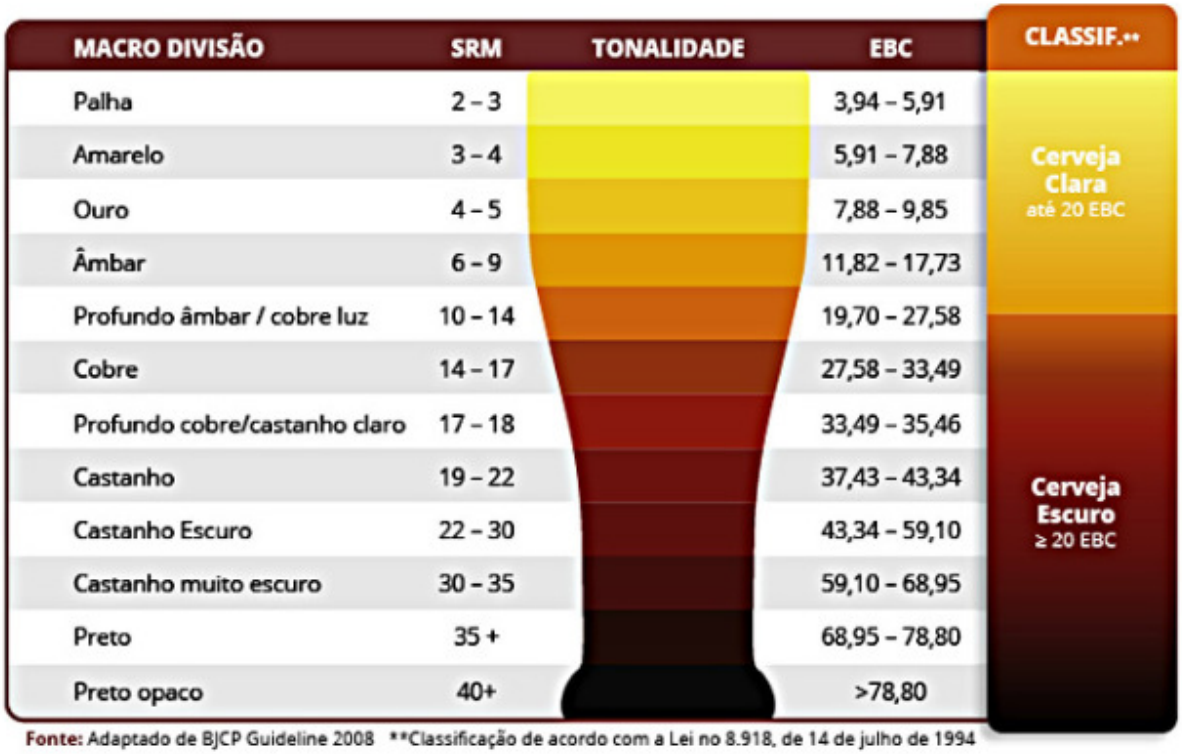

Figura 1 – Escalas de cores EBC e SRM. Fonte: http://www.cerveja.blog.br/

As duas escalas possuem uma diferença nos valores, sendo os valores da SRM equivalem a 40% dos encontrados na escala EBC.

O malte e o seu grau de torrefação são os principais determinantes na coloração da cerveja, porém, adjuntos podem ser adicionados no processo.

2.0.2 O Amargor da Cerveja

International Biterness Unit, essa é a unidade de medida oficial para o amargor das cervejas. A medição é feita baseada na concentração de iso-alfa-ácidos na cerveja, seguindo a regra de 1 IBU para cada 1 mg de iso-alfa-ácido por litro de cerveja.

Outros componentes diferentes dos iso-alfa-ácidos também podem ser detectados na cerveja pela medida IBU, como componentes do malte escuro e produtos da oxidação de resina de lúpulo (OLIVER; COLICCHIO, 2012).

O lúpulo é o ingrediente responsável por esse amargor, além de prover propriedades de aroma e sabor. Ele libera os iso-alfa-ácidos na cerveja após a fervura do mosto e, dependendo de sua propriedade, o lúpulo pode ser adicionado no início ou no final do processo. Se adicionado no início, o lúpulo deixa a cerveja mais amarga e se for adicionado no final, obtêm-se características de aroma.

Figura 2 – Lúpulo é o responsável pelo amargor da cerveja. Fonte: http://cervejaemalte.com.br/

Além da quantidade de lúpulo, deve-se considerar na medição do amargor o índice de alfa ácido do lúpulo, a densidade do mosto, o tempo de fervura e o volume final de cerveja.

Existe ainda uma técnica de utilização do lúpulo na cerveja que merece ser destacada, ela se chama **Dry Hopping**. Segundo (CALAGIONE, 2012), a técnica consiste em adicionar o lúpulo pós-fervura, antes das etapas de fermentação e envase, ou mesmo durante essas.

Essa técnica é importante para extrair óleos essenciais do lúpulo, que, porventura, são eliminados na fase de fervura quando adicionado em abundância.

O Dry Hopping é usado principalmente na fabricação de cervejas especiais, trazendo maiores propriedades de aroma para a cerveja.

2.0.3 Razão BU/GU

Esse cálculo refere-se a relação entre as unidades de amargor (BU: Bitterness Unit) e a gravidade original da cerveja em unidades de gravidade (GU: Gravity Unit) (GERBER HORNINK; GALEMBECK, 2019). Pode-se dizer que essa relação expressa o doce do malte contra o amargor do lúpulo.

A razão varia entre cerca de 0,2 e 1, sendo as cervejas próximas de 0,2 mais maltadas (mais doces), as próximas de 1 mais lupuladas (mais amargas) e as próximas de 0,5 mais equilibradas.

Esse parâmetro é muito importante na fabricação da cerveja, visto que cada estilo tem sua razão BU:GU específico, de modo que o produto final fique equilibrado no estilo desejado.

2.0.4 Álcool por Volume da Cerveja

Esta é a classificação adotada mundialmente para definir o percentual de álcool presente no volume total de uma bebida alcoólica. Segundo (BAMFORTH, 2016), essa medida é importante tanto para programas de garantia de qualidade da bebida quanto para fins de relatórios legais.

O álcool por volume é determinado pela diferença entre a Gravidade Inicial e a Gravidade Final, pois o resultado dessa diferença é exatamente a densidade dos açúcares consumidos na fermentação do mosto.

A partir dessa densidade dos açúcares transformados em álcool, é possível estimar a quantidade de álcool no mosto resultante da degradação desses açúcares.

O álcool na bebida tem a principal função de manter a cerveja estável, evitando a propagação de microorganismos na mesma. Dessa forma, as cervejas que possuem um teor de ABV mais alto, podem ser conservadas por mais tempo.

No Brasil, cervejas que possuem teor alcoólico entre 0.5% e 2.0% são consideradas de teor baixo, aquelas que estiverem entre 2.0% e 6.0% de teor médio, e as acima de 6.0% são consideradas de teor alto.

2.0.5 Estilos

Como é possível perceber, podemos obter diferentes resultados em cada um dos parâmetros de uma receita dependendo dos ingredientes selecionados e do momento em que são utilizados no processo de fabricação da cerveja.

Dessa forma, as cervejas são categorizadas por estilos, de modo a agrupar as diferenças baseadas nos métodos de produção, ingredientes utilizados, história, aparência, sabor e até mesmo origem.

Esse conceito de categorização de cerveja e diferenciação por estilos é baseado na obra *The World Guide to Beer*(JACKSON, 1977), do escritor inglês Michael Jackson. Nela, Michael descreve detalhadamente uma variedade enorme de marcas de cervejas clássicas espalhadas pelo mundo, além de organizá-las em estilos e categorizá-las segundo a cultura e região onde são produzidas.

Hoje, além da obra do Michael Jackson, existem diversas associações internacionais que tratam de categorizar os estilos de cerveja produzidos no mundo. Essas associações liberam um Guia de Estilos atualizado todos os anos com os estilos de cerveja e os atributos que cada receita deve ter para se enquadrar.

As principais associações são a BCJP (Beer Judge Certification Program) (BEER JUDGE CERTIFICATION PROGRAM – PROMOTING BEER LITERACY..., 2021), a CAMRA (Campaing for Real Ale) (HOME - CAMRA - CAMPAIGN FOR REAL ALE..., 2021) e a Brewers Association (BREWERS ASSOCIATION | PROMOTING INDEPENDENT CRAFT BREWERS..., 2021).

3 LINGUAGENS DE PROGRAMAÇÃO E O PYTHON

Apesar de não existir um conceito definido do que é uma linguagem de programação, (ELOQUENT JAVASCRIPT. . ., 2021) a apresenta como sendo:

> uma linguagem construída artificialmente usada para instruir computadores. É interessante que a maneira mais eficaz que encontramos para nos comunicarmos com um computador se baseia na forma como nos comunicamos na sociedade. Como as linguagens humanas, as linguagens de computador permitem que palavras e frases sejam combinadas de novas maneiras, tornando possível expressar conceitos sempre novos.

Apesar do primeiro computador eletrônico do mundo, o gigantesco ENIAC, ter entrado em funcionamento somente em fevereiro de 1946, o primeiro trabalho relacionado às linguagens de programação remete há mais de cem anos antes. Isso porque em 1843 Ada Lovelace inventou um algoritmo para um projeto de calculadora mecânica de seu amigo Charles Babbage (IEEE ANNALS OF THE HISTORY OF COMPUTING. . ., 2021). Contudo, o projeto não foi para frente e Babbage acabou implementando somente uma parte do algoritmo de Ada, chamado "The Analytical Engine".

Contudo, a primeira linguagem de alto nível seria lançada somente em 1954. Até então o grande trabalho dos programadores era derrubar as dificuldades impostas pelos próprios computadores da época como falta de registros de índices e arranjos de input e output antiquados. Dessa forma John Backus propôs ao seu chefe na IBM, a criação do FORTRAN, sendo essa a primeira linguagem de programação de ato nível amplamente utilizada pela comunidade científica.

A partir daí, a medida que a tecnologia dos computadores ia evoluindo, diversas outras linguagens foram surgindo para atender os mais diversos paradigmas e desafios da computação.

O principal sistema de classificação, desenvolvido pela ACM, é classificado em:

Paradigmas Imperativos:

- Paradigma Procedural — FORTRAN e BASIC.
- Paradigma de Estrutura de Blocos — Algol 60, Pascal e C.
- Paradigma Orientado a Objetos — C++, D, Java, Python e Ruby.
- Paradigma da Computação Distribuída — Ada.

Paradigmas Declarativos:

- Paradigma Funcional — Lisp, Scheme e Haskell.

- Paradigma da Programação Lógica — Prolog e Gödel.

Outra categoria de classificação utilizado é com relação ao modelo de estrutura das linguagens, sendo:

Forte ou Fracamente Tipada:

- Fracamente Tipada — PHP e Smalltalk.
- Fortemente Tipada — Java, Ruby e Python.

Dinâmica ou Estaticamente Tipada:

- Dinamicamente Tipada — SNOBOL, APL, Awk, Perl, Python e Ruby.
- Fortemente Tipada — Java e C.

3.1 Linguagem Python

Como apresentado em sua própria documentação oficial, o Python é uma linguagem de programação interpretada, interativa e orientada a objetos que incorpora módulos, exceptions, tipagem dinâmica, categorias de dados de alto nível e classes (GENERAL PYTHON FAQ — PYTHON 3. . ., 2021). A linguagem ainda suporta múltiplos paradigmas, tem uma sintaxe clara e é portátil, funcionando em diversos Sistemas Operacionais. Dessa forma, ela acaba sendo uma das linguagens mais dinâmicas existentes, podendo funcionar nas mais diversas aplicações e soluções de problemas.

A linguagem começou a ser desenvolvida no final de 1989 por Guido Van Rossum no Instituto de Pesquisa Nacional para Matemática e Ciência da Computação (CWI), tendo sua primeira versão lançada em 1991. Segundo o próprio Guido, ele necessitava de uma linguagem que fosse mais extensível que o ABC usado no instituto e realizasse uma melhor administração do sistema Amoeba em que trabalhava na época (GENERAL PYTHON FAQ — PYTHON 3. . ., 2021). Segundo o próprio autor, a linguagem Modula-3 é a origem da sintaxe e semântica usada para exceções no Python, entre outras features. Além do ABC e do Modula-3, Python também possui influências na sintaxe derivadas do C.

Com relação ao nome, Guido não queria uma sigla (como C e ABC, por exemplo) nem tampouco um nome fraco. Assim Guido resolveu homenagear um de seus grupos humorísticos preferidos, o Monty Python's Flying Circus. Ao mesmo tempo, em que era simples e cativante, o nome se encaixava na tradição de nomear linguagens de programação baseadas em pessoas famosas e na tradição do próprio CWI Amoeba que nomeava seus projetos baseados em programas de televisão famosos da época (THE HISTORY OF PYTHON: PERSONAL HISTORY - PART 1. . ., 2021).

Atualmente, a linguagem é desenvolvida abertamente e gerenciada pela Python Software Foundation, uma organização sem fins lucrativos. A fundação deteve os direitos de propriedade intelectual da linguagem a partir da versão 2.1.

3.2 Paradigmas de Programação Suportados pelo Python

3.2.0.1 Programação Funcional

Programação funcional é um estilo de programação do tipo declarativo, onde é realizada com expressões ou declarações. Essa categoria de programação trata a computação como funções matemáticas, enfatizando a aplicação de funções, as quais são tratadas como valores de primeira importância, ou seja, funções podem ser parâmetros ou valores de entrada para outras funções e podem ser os valores de retorno ou saída de uma função (JUNGTHON; MACHADO GOULART, s.d.).

Muitos tratam a programação funcional como sendo oposta à programação imperativa, pois essa enfatiza mudanças no estado do programa. Contudo, isso não é verdade, Hudak (s.d.) explica que, na verdade, existe muito em comum entre os dois estilos e ambos podem, inclusive, serem usados juntos.

A grande influência da programação funcional foi o framework desenvolvido por Alonzo Church, chamado Cálculo Lambda. Na verdade, o Cálculo Lambda é por vezes considerado a primeira linguagem funcional, mesmo que na época não existiam computadores para executarem os programas. Em todo caso, as linguagens funcionais modernas podem ser vistas como um Cálculo Lambda melhorado. (HUDAK, s.d.)

3.2.0.2 Programação Orientada a Objetos

A Programação Orientada a Objeto (também conhecida como POO), é um tipo de programação baseado especificamente no conceito de objetos. Esses objetos, sendo geralmente instâncias de classes, são usados em interações entre si para a criação de aplicações e programas computacionais (OOAD - OBJECT ORIENTED PARADIGM - TUTORIALSPOINT..., 2019).

Geralmente, podemos citar as principais ideias do POO como sendo:

- Tudo que será computado é um objeto, que nada mais é do que uma instância de uma determinada classe.
- A classe é a principal ferramenta neste conceito, ou melhor dizendo, sua definição. A partir dessa definição, os objetos da classe podem compartilhar a estrutura definida pelas propriedades comuns a todos, sendo que essas propriedades são objetos de alguma classe especificada.
- Pode existir um modelo de herança entre as classes, de forma que um objeto de uma determinada classe também será objeto da sua superclasse.
- Pode-se utilizar o conceito de definição de métodos para trabalhar com os objetos. Assim, esses métodos serão somente utilizados quando o objeto for de determinada classe.

Este paradigma foi muito importante como parte da solução para a chamada *Crise de Software*, isso por abrir a possibilidade para que os componentes de software pudessem ser construídos e reaproveitados com muito mais credibilidade (CRAIG, 2007).

3.2.0.3 Programação Imperativa

A Programação Imperativa foi projetado com base na arquitetura criada por Von Neumann, concebido a partir de 1946, assim o primeiro e mais abrangente paradigma de programação desenvolvido até hoje.

Esse modelo de programação executa o código de forma passo-a-passo, realizando mudanças direto no estado do programa, assim chamado também de modelo de programação com estados (PERCEIVING PYTHON PROGRAMMING PARADIGMS | OPENSOURCE..., 2021). O nome do paradigma está ligado ao tempo verbal imperativo, onde o programador delega funções ao computador sobre como alterar o seu estado.

O paradigma imperativo é também o mais antigo e abrangente paradigma de linguagens de programação.

3.2.0.4 Programação Procedural

A programação procedural pode ser considerada um sub-tipo da programação imperativa. Ambos os paradigmas seguem o padrão de mudanças nos estados do programa, porém a diferença é que na programação procedural essa mudança é realizada através de procedimentos (rotinas, sub-rotinas, métodos, ou funções).

Um grande benefício da programação procedural é que ela facilita a prática de design de um bom programa e permite a reutilização de módulos através de bibliotecas de código (PERCEIVING PYTHON PROGRAMMING PARADIGMS | OPENSOURCE..., 2021).

3.3 Bibliotecas Utilizadas

3.3.1 Pandas

Esta é uma biblioteca com foco na análise e manipulação de dados desenvolvida em Python. Ela é reconhecida por ser rápida, poderosa, flexível e fácil de usar (PANDAS - PYTHON DATA ANALYSIS LIBRARY..., 2021).

Pandas é rica em estruturas de dados e ferramentas para se trabalhar com conjuntos de dados estruturados comuns a estatística, finanças, ciências sociais e muitos outros campos de estudo (MCKINNEY, 2021). Ela está em desenvolvimento desde 2008 e, desde então, tem sido recebida muito pela comunidade.

Neste projeto ela foi utilizada para tratar um arquivo JSON contendo os valores dos estilos de cerveja analisados e, a partir daí, transformá-lo em um data frame, possibilitando realizar de forma simples e rápida a classificação dos dados enviados pelo usuário.

3.3.2 Apache ECharts

O Apache ECharts é uma biblioteca voltada para a visualização de dados, permitindo a criação de gráficos intuitivamente, interativa, simples e altamente customizável.

Além a grande variedade de gráficos disponíveis, possui também outras features importantes como otimização para aparelhos mobile, dados dinâmicos, acessibilidade e cross-platform.

Neste projeto ele foi utilizado para apresentar (Figura 3) o grau de compatibilidade dos dados inseridos pelo usuário com os estilos de cerveja.

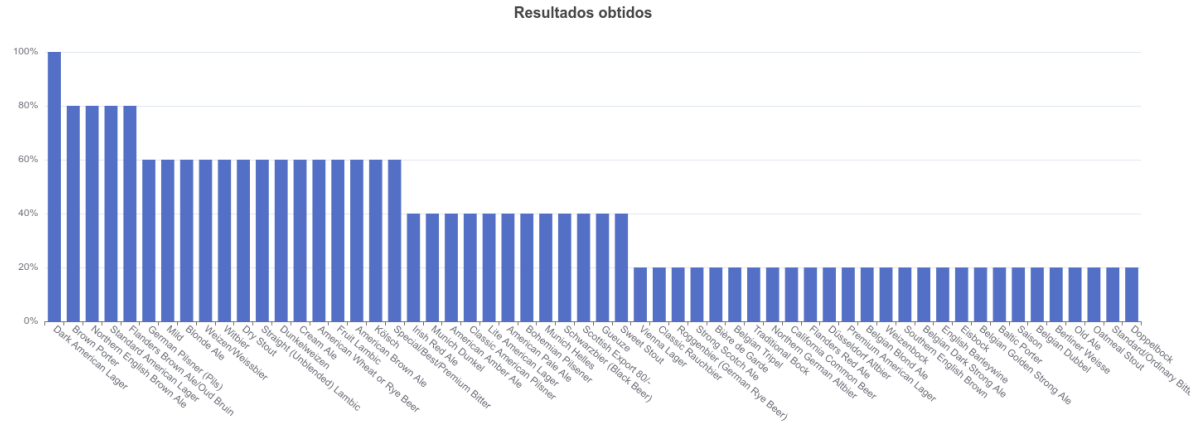

Figura 3 – Gráfico de barras gerado com o Apache ECharts. Fonte: Autor

4 FRAMEWORKS E DJANGO

Desde a criação do paradigma orientado a objetos, o conceito de reaproveitamento de códigos foi amplamente adotado na comunidade de desenvolvimento, afinal, não é necessário reinventarmos a roda. A partir daí então se popularizaram as bibliotecas e os frameworks nos mais diversos projetos.

Assim, (CARLOS GROTT; HUGO, s.d.) define um framework da seguinte forma:

> (...) considerar um framework como sendo uma coleção de classes que fornece um conjunto de serviços para um domínio particular podendo conter um conjunto de classes em que foram aplicados um ou vários padrões na sua modelagem.

O grande diferencial de um framework são as suas implementações reutilizáveis na forma de implementações classes abstratas e concretas. Essas implementações abstratas, por sua vez, são classes abstratas que implementam partes de uma abstração do framework, porém deixam as decisões importantes de implementação nas subclasses (RIEHLE DIPL, 2000).

Além do aumento de produtividade que proporciona no desenvolvimento, também reduz custos para as empresas otimizando processos e encurtando o tempo de entrega de um produto (O QUE É DJANGO. . ., 2021). Em termos mais simples, podemos dizer que os frameworks funcionam como um esqueleto, que irá determinar como os objetos se relacionarão entre si.

Contudo, é importante destacar que, apesar de possuírem definições parecidas, frameworks e bibliotecas são diferentes. Em uma biblioteca de classes, cada classe é única e independente das outras (O QUE É UM FRAMEWORK?. . ., 2021), enquanto em um framework existe um modelo de colaboração entre suas implementações funcionando por baixo dos panos. É importante destacar também que em um framework existem várias bibliotecas.

4.0.1 Padrão Arquitetural de Software

4.0.2 Servidores Web

Os servidores Web são os responsáveis por manter no ar tudo que vemos na Internet e está totalmente relacionado com a história da mesma. Em março de 1989, Sir Tim Berners-Lee apresentou a proposta de um sistema de gerenciamento de informações (CERN. . ., 2021) com o intuito de facilitar a troca de informações no instituto onde trabalhava, o CERN, utilizando um sistema de hipertexto.

Em 1990, Berners-Lee programou o primeiro navegador Web, chamado WorldWideWeb, e o primeiro servidor Web, CERN HTTPd, tendo ambos rodando em uma estação de trabalho NeXTcube (THE WORLD WIDE WEB OF TIM BERNERS-LEE - HISTORY COMPUTER. . ., 2021).

Um servidor Web é uma combinação de software e hardware. A parte do software é necessária para responder às requisições, produzir formulários, rodar scripts, etc., sempre

seguindo as regras do protocolo HTTP (YEAGER; MCGRATH, 1996). Qualquer programa que cumpra esses requisitos pode ser usado como Web server.

Quanto ao hardware, deve-se existir um computador onde serão hospedados os arquivos referentes aos Web sites. Por questões de disponibilidade dos sites, esse computador deve possuir acesso à Internet 24 horas.

4.0.2.1 Protocolo HTTP

O Hypertext Transfer Protocol (HTTP) é um protocolo de nível de aplicação voltado para sistemas de informação hipermídia distribuídos e colaborativos (BERNERS-LEE, 1996).

o funcionamento do protocolo HTTP é apresentado da seguinte forma por (RFC2616. . ., 2021):

> O protocolo funciona em um esquema de requisição/resposta. Um cliente envia uma requisição ao servidor na forma de um método de requisição, URI e versão do protocolo, seguido por uma mensagem semelhante ao tipo MIME contendo modificadores de requisição, informações do cliente e possível conteúdo do corpo da mensagem através de uma conexão com um servidor. O servidor responde com uma linha de status, incluindo a versão do protocolo da mensagem e um código de sucesso ou erro, seguido por uma mensagem semelhante ao tipo MIME contendo informações do servidor, metainformação da entidade e possível conteúdo do corpo da entidade.

O HTTP é a fundação para a comunicação em toda a World Wide Web. Nessa comunicação, existem dois tipos de protocolos de transporte comumente utilizados: TCP, o mais seguro e confiável, e UDP, que não possui o mesmo nível de confiança na entrega das mensagens, porém é necessário em aplicações de tempo real.

4.0.3 Django

Django é um framework de código aberto Python de alto-nível voltado para o desenvolvimento rápido e limpo, e design pragmático, seguindo o padrão de arquitetura model-template-views (MTV) (THE WEB FRAMEWORK FOR PERFECTIONISTS WITH DEADLINES | DJANGO. . ., 2021). Esse framework também tem como diferencial a segurança aplicada e a alta escalabilidade, permitindo uma alta flexibilidade ao desenvolvedor.

Um projeto desenvolvido em Django é geralmente dividido em pequenas aplicações baseadas no padrão MTV. Essas aplicações são pacotes Python que podem utilizar as convenções básicas do Django como models, tests, urls e views. Dessa forma, o desenvolvedor consegue resolver problemas específicos do projeto dentro dessas aplicações separadamente.

Um bom exemplo seria um sistema Web de loja online, onde o desenvolvedor poderia criar as regras do módulo financeiro e da vitrine de produtos em aplicações separadas, porém

funcionando no mesmo escopo do projeto. Nesse exemplo, o sistema Web seria o projeto Django, enquanto os módulos financeiros e da vitrine são as aplicações Django.

A ideia do framework surgiu no outono de 2003, quando Adrian Holovaty e Simon Willison no jornal Lawrence Journal-World em Kansas. Eles precisavam criar diversos sites com aplicações como classificados, calendários e divulgação do comércio local para a comunidade de Lawrence. Todo o trabalho inicial era muito custoso já que o desenvolvimento Web em Python ainda estava no seu início e eles tinham que construir tudo do zero (THE STORY OF DJANGO | DJANGO DESIGN PATTERNS AND BEST PRACTICES - SECOND EDITION..., 2021).

Assim, eles passaram a refatorar os módulos e ferramentas comuns em um projeto chamado The CMS. Em 2005, o The CMS, agora com o nome de Django em homenagem ao guitarrista de Jazz Django Reinhardt, (THE STORY OF DJANGO | DJANGO DESIGN PATTERNS AND BEST PRACTICES - SECOND EDITION..., 2021) foi publicado sob a licença de código aberto Berkeley Software Distribution (BSD) (THE 4..., 2021).

Atualmente, o desenvolvimento e suporte do framework é realizado pela Django Software Foundation, criada exclusivamente para este fim (THE WEB FRAMEWORK FOR PERFECTIONISTS WITH DEADLINES | DJANGO..., 2021).

4.0.4 Padrão MTV

Como citado, o Django possui um padrão de projeto próprio, derivado do já conhecido padrão MVC (Model-View-Controller), chamado MTV (Model-Template-Views). O padrão MVC pode ser definido assim:

- Model — Representam os dados que o usuário trabalha. Essa parte faz a ligação dos dados entre o Controller e as Views (FREEMAN, 2014).
- Views — Apresentam parte da Model em forma de interface para o usuário (FREEMAN, 2014).
- Controller — Processa as requisições, realiza operações na Model e seleciona as views a serem apresentadas para o usuário (FREEMAN, 2014).

Essa diferença no padrão utilizado pelo Django se dá pelo fato das funções referentes ao Controller serem realizadas pelo próprio framework, deixando ao desenvolvedor o trabalho de configurar somente as Models, Views e Templates (FAQ: GERAL | DOCUMENTAÇÃO DO DJANGO | DJANGO..., 2021).

Dessa forma, podemos apresentar o padrão MTV como sendo:

- Model — É a camada de acesso dos dados. Contém todas as informações sobre os dados: como acessá-los, como validá-los, qual comportamento possui e as suas respectivas relações (THE MTV DEVELOPMENT PATTERN - TUTORIAL..., 2021).

- Templates — É a camada de apresentação. Contém as decisões relacionadas à apresentação dos dados: como o dado deve ser apresentado na tela ou outro tipo de documento (THE MTV DEVELOPMENT PATTERN - TUTORIAL. . ., 2021).
- View — camada de lógica do negócio. Contém a lógica de acesso a Model e redireciona ao template apropriado (THE MTV DEVELOPMENT PATTERN - TUTORIAL. . ., 2021).

No que diz respeito ao funcionamento, ambos os padrões são muito parecidos, tendo como diferença como cada um interpreta os componentes de código. Contudo, o modelo apresentado no Django permite que o desenvolvedor se concentre somente nas tarefas que dizem respeito ao necessário para a aplicação Web, enquanto cuida do tratamento das requisições das views segundo a própria configuração do framework, aumentando assim a produtividade.

5 SISTEMA PARA CLASSIFICAÇÃO DE ESTILOS DE CERVEJA

O sistema foi desenvolvido principalmente utilizando a linguagem Python com o framework de desenvolvimento web Django. A biblioteca responsável pelo gráfico foi desenvolvida em Javascript, enquanto o arquivo de armazenamento dos dados relativos aos estilos das cervejas está no formato JSON. Quanto ao estilo geral do sistema, foi utilizado o framework Bootstrap 5.

A ideia principal é que o usuário insira os dados obtidos com a sua cerveja e tenha os resultados obtidos com base nos valores homologados pelas principais associações de cerveja no mundo.

Figura 4 – Página inicial do sistema. Fonte: Autor

Logo na tela principal (Figura 4), temos o formulário com os campos necessários em que o usuário informará os seus dados para a comparação dos estilos.

Código 1 – Código da Dashboard do sistema

```
{% extends '../main.html' %}

{% block content %}
<div class="container text-center">
    {% load static %}
    <link href="{% static "css/dashboard.css" %}" rel="stylesheet
       " >

    <h2 class="title" >Classificador de Cervejas</h2>

    <form action="/results" method="POST">
    {% csrf_token %}
        <div class="container">
            <div class="row row-cols-sm-6 ">
                <div class=".col-auto">
                    <label class="form-label ">IBU</label>
```

```
17                      <input type="number" class="form-control p-3
                            border bg-light" placeholder="Ex: 8" aria-
                            label="IBU" name="IBU" >
18                  </div>
19                  <div class=".col-auto">
20                      <label class="form-label">SRM</label>
21                      <input type="number" class="form-control p-3
                            border bg-light" placeholder="Ex: 20" aria-
                            label="SRM" name="SRM" >
22                  </div>
23                  <div class=".col-auto">
24                      <label class="form-label">OG</label>
25                      <input type="number" class="form-control p-3
                            border bg-light" step="0.001" placeholder="
                            Ex: 1,045" aria-label="OG" name="OG" >
26                  </div>
27                  <div class=".col-auto">
28                      <label class="form-label">FG</label>
29                      <input type="number" class="form-control p-3
                            border bg-light" step="0.001" placeholder="
                            Ex: 1,008" aria-label="FG" name="FG" >
30                  </div>
31                  <div class=".col-auto">
32                      <label class="form-label">ABV</label>
33                      <input type="number" class="form-control p-3
                            border bg-light" step="0.1" placeholder="Ex
                            : 4,5" aria-label="ABV" name="ABV" >
34                  </div>
35                  <div class=".col-auto">
36                      <input class="btn btn-info p-3 btn-space"
                            type="submit" value="Submit">
37                  </div>
38              </div>
39          </div>
40      </form>

42  </div>



45  {% endblock %}
```

É interessante citar também a utilização do forms. Essa ferramenta permite fazer a validação dos campos em que o usuário irá inserir os dados de forma simples e fácil.

Código 2 – Utilização dos forms no Django

```
from django import forms

class BeerForm(forms.Form):
    IBU = forms.DecimalField(max_digits=5, decimal_places=3,
       label='IBU')
    SRM = forms.DecimalField(max_digits=5, decimal_places=3,
       label='SRM')
    OG = forms.DecimalField(max_digits=5, decimal_places=3, label
       ='OG')
    FG = forms.DecimalField(max_digits=5, decimal_places=3, label
       ='FG')
    ABV = forms.DecimalField(max_digits=5, decimal_places=3,
       label='ABV')
```

Depois que o usuário informa os dados que deseja realizar a comparação, é invocada a view da página de resultados. Nela, os dados são enviados à classe que realiza as comparações com os estilos de cervejas.

Código 3 – View da página de resultados

```
def results(request):

    form = BeerForm(request.POST)
    ibu = form.data['IBU']
    srm = form.data['SRM']
    og = form.data['OG']
    fg = form.data['FG']
    abv = form.data['ABV']

    # Busca os dados do banco
    df = pd.read_json("styles.json")

    # Exemple
    ## Create Object
    myBeer = Styles.Selector(df)
```

```
18      # Define variable values
19      myBeer.setVariable('ibu', int(ibu))
20      myBeer.setVariable('srm', int(srm))
21      myBeer.setVariable('og', int(og))
22      myBeer.setVariable('fg', int(fg))
23      myBeer.setVariable('abv', float(abv))

25      # Get Possible Classes and its Proabilities and Frequencies
26      # Run -> myBeer.getPossibilities(True) to get JSON Response
27      context = {'results' : myBeer.getPossibilities(True)}

29      return render(request, 'beer/results.html', context)
```

Nas linhas 19 à 23, setamos para a classe de comparação os valores enviados pelo usuário e em seguida, na linha 27, o método que realiza as comparações é chamado.

É importante destacar também a conversão do arquivo JSON com os estilos para o formato de data frame utilizando a biblioteca Pandas na linha 12. Essa etapa é necessária para a realização na comparação dos dados.

Código 4 – Classe para comparação dos dados

```
2   from django.db import models
3   import pandas as pd

5   # Create your models here.
6   class Styles(models.Model):
7       def __str__(self):
8           return self.name

10      objects = models.Manager()

12      class Selector:
13          def __init__(self, data):

15              self.variables = {
16                  "ibu": None,
17                  "srm": None,
18                  "og": None,
19                  "fg": None,
20                  "abv": None
```

```
21              }
23              self.dfs = {
24                  "classes": data[['subCategory']].set_index(['
                        subCategory']),
25                  "ibu": data[['subCategory','IBUMin','IBUMax']].
                        rename(columns={'IBUMin':'min','IBUMax':'max'})
                        ,
26                  "srm": data[['subCategory','SRMMin','SRMMax']].
                        rename(columns={'SRMMin':'min','SRMMax':'max'})
                        ,
27                  "og": data[['subCategory','OGMin','OGMax']].
                        rename(columns={'OGMin':'min','OGMax':'max'}),
28                  "fg": data[['subCategory','FGMin','FGMax']].
                        rename(columns={'FGMin':'min','FGMax':'max'}),
29                  "abv": data[['subCategory','ABVMin','ABVMax']].
                        rename(columns={'ABVMin':'min','ABVMax':'max'})
30              }
32          def getVariable(self, variable):
33              return self.variables[variable]
35          def setVariable(self, variable, value):
36              self.variables[variable] = value
38          def getVariablePossibilities(self, variable):
39              return self.dfs[variable][
40                      (self.dfs[variable]['min'] <= self.
                            getVariable(variable))
41                  & (self.dfs[variable]['max'] >= self.getVariable(
                        variable))
42              ]['subCategory']
44          def getPossibilities(self, json=False):
45              allPossibilities = []
46              for variable in self.variables.keys():
47                  if self.variables[variable] is None:
48                      raise TypeError("O valor da variövel {
                            variavel} nöo poder ser 'Vazio'.".format(
                            variavel=variable))
50                  allPossibilities.extend(
```

```
51                      self.getVariablePossibilities(variable).to_
                            list()
52                  )

54              allPossibilities = pd.DataFrame(allPossibilities,
                    columns=['classes'])['classes'] \
55                  .value_counts() \
56                  .to_frame() \
57                  .join(
58                      self.dfs['classes']
59                  ) \
60                  .reset_index() \
61                  .rename(columns={'classes': 'frequency', 'index':
                        'subCategory'})[[
62                  'subCategory',
63                  'frequency'
64              ]]

66              allPossibilities['probability'] = allPossibilities['
                    frequency'] / len(self.variables.keys())

68              if json:
69                  return allPossibilities.to_json(orient="records")
70              else:
71                  return allPossibilities
```

Na linha 44 da classe temos a função principal, onde os dados serão feitas as comparações principais. O Pandas é muito utilizado na formatação desses dados e na geração do JSON com os resultados das probabilidades.

Por fim, após todo o tratamento e comparação dos dados, o sistema redireciona o usuário para a página com os resultados obtidos, apresentando a porcentagem de assertividade com cada estilo de cerveja.

Código 5 – Página com os resultados

```
2  {% extends '../main.html' %}

4  {% block content %}
```

```
<div id="chart" style="width: 100vw;height:55vh;margin: auto"></div>



{% load static %}
<script src="{% static "js/echarts.js" %}"></script>

<script type="text/javascript">
    const results = {{ results|safe }}

    const myChart = echarts.init(document.getElementById('chart'));

    const option = {
        title: {
            text: 'Resultados obtidos',
            left: 'center'
        },
        tooltip: {
            trigger: 'axis',
            axisPointer: {
                type: 'shadow'
            },
            formatter: function (params) {
                const style = params[0];
                return style.axisValueLabel + '<br/>' + (style.data) + '%';
            }
        },
        xAxis: {
            type: 'category',
            data: results.map(item => item.subCategory),
            axisLabel: {
                interval: 0,
                rotate: -40
            }
        },
        yAxis: {
            type: 'value',
            axisLabel: {
                formatter: '{value}%'
            }
```

```
46          },
47          grid: {
48              left: '3%',
49              right: '4%',
50              bottom: '3%',
51              containLabel: true
52          },
53          series: [{
54              data: results.map(item => item.probability * 100),
55              type: 'bar'
56          }]
57      }
58      myChart.setOption(option);
59  </script>

61  {% endblock %}
```

5.1 Resultados Obtidos

Para apresentar um exemplo, será utilizado os valores encontrados em um perfil de uma cerveja Porter. Neste caso, teremos:

- ABV — 5.6%;
- FG — 1.014;
- OG — 1.056;
- IBU — 26;
- SRM — 25;

Como resultado dessa entrada no sistema (figura 5), ele nos gera 3 principais resultados, sendo eles Robust Porter, American Brown Ale, Munich Dunkel e Oatmeal Stout.

Efetuando agora a comparação com os valores presentes no arquivo de estilos de cerveja, podemos observar que a comparação foi realizada corretamente.

Código 6 – Estilos de cerveja retornados no resultado

```
2   {% extends '../main.html' %}

4   {% block content %}
```

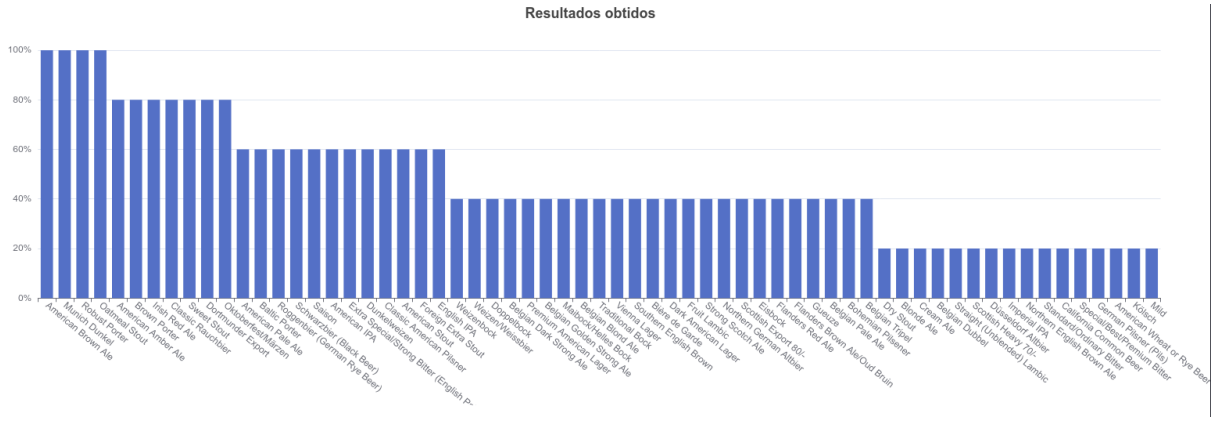

Figura 5 – Gráfico com os resultados do teste. Fonte: Autor

```
<div id="chart" style="width: 100vw;height:55vh;margin: auto"></
    div>

  {
    "cod": "12B",
    "category": "PORTER",
    "subCategory": "Robust Porter",
    "IBUMin": 25,
    "IBUMax": 40,
    "SRMMin": 30,
    "SRMMax": 99999,
    "OGMin": 1045,
    "OGMax": 1060,
    "FGMin": 1008,
    "FGMax": 1016,
    "ABVMin": 5.1,
    "ABVMax": 6.6
  },
  {
    "cod": "10C",
    "category": "AMERICAN ALE",
    "subCategory": "American Brown Ale",
    "IBUMin": 20,
    "IBUMax": 40,
    "SRMMin": 18,
    "SRMMax": 35,
    "OGMin": 1045,
    "OGMax": 1060,
    "FGMin": 1010,
```

```
    "FGMax": 1016,
    "ABVMin": 4.3,
    "ABVMax": 6.2
  },
    {
    "cod": "4B",
    "category": "DARK LAGER",
    "subCategory": "Munich Dunkel",
    "IBUMin": 18,
    "IBUMax": 28,
    "SRMMin": 14,
    "SRMMax": 28,
    "OGMin": 1048,
    "OGMax": 1056,
    "FGMin": 1010,
    "FGMax": 1016,
    "ABVMin": 4.5,
    "ABVMax": 5.6
  },
  {
    "cod": "13C",
    "category": "STOUT",
    "subCategory": "Oatmeal Stout",
    "IBUMin": 25,
    "IBUMax": 40,
    "SRMMin": 22,
    "SRMMax": 40,
    "OGMin": 1048,
    "OGMax": 1065,
    "FGMin": 1010,
    "FGMax": 1018,
    "ABVMin": 4.2,
    "ABVMax": 5.9
  }

{% endblock %}
```

6 CONSIDERAÇÕES FINAIS

O desenvolvimento do presente trabalho possibilitou uma análise mais detalhada da produção de cervejas artesanais de modo a explicar as diferenças e peculiaridades de cada receita e estilo. Ainda mais, permitiu também uma pesquisa sobre a linguagens de programação Python e o framework Django, bem como o desenvolvimento de um sistema web baseado nessas tecnologias.

Ao realizar o estudo aprofundadamente, pôde-se verificar os aspectos mais importantes que caracterizam a composição da cerveja como tipo do malte, tempo de fermentação, grau de torrefação do malte, quantidade de cada ingrediente, entre outros. Foi possível também evidenciar os principais conceitos que separam as diversas classes de cerveja ao redor do mundo.

Quanto ao sistema desenvolvido, pôde-se aprofundar mais no conhecimento da linguagem Python e do seu framework Django, apresentando seus principais padrões e melhores práticas de programação de modo a entregar um sistema funcional e com alto grau de produtividade.

Dada a quantidade de estilos e ingredientes diferentes que podem ser adicionados na receita de uma cerveja, um sistema torna-se necessário para auxiliar no processo de classificação das mesmas, de modo a facilitar ao cervejeiro classificar a cerveja que acabou de produzir, isso porque ele não precisará buscar essas informações direto nas associações já que as mesmas são utilizadas pelo classificador a fim de obter o melhor resultado.

Nesse sentido, o fato de ser um sistema web também permite que o seu acesso seja efetuado de forma simples e rápida, necessitando somente do acesso à internet, visto que o mesmo está disponível a todo o momento e a todo tipo de dispositivo.

BIBLIOGRAFIA

AS 10 BEBIDAS MAIS CONSUMIDAS NO MUNDO | DIÁRIO DO ESTADO. Disponível em: <https://diariodoestadogo.com.br/as-10-bebidas-mais-consumidas-no-mundo-110252/>. Acesso em: 12 ago. 2021.

BAMFORTH, C. **Brewing Materials and Processes: A Practical Approach to Beer Excellence**. Elsevier Science, 2016. ISBN 9780128004685. Disponível em: <https://books.google.com.br/books?id=52OdBgAAQBAJ>.

BEER JUDGE CERTIFICATION PROGRAM – PROMOTING BEER LITERACY, RECOGNIZING BEER TASTING AND EVALUATION SKILLS. Disponível em: <https://dev.bjcp.org/>. Acesso em: 12 ago. 2021.

BERNERS-LEE, T. WWW: past, present, and future. **Computer**, v. 29, n. 10, p. 69–77, 1996. ISSN 1558-0814. DOI: 10.1109/2.539724.

BREWERS ASSOCIATION | PROMOTING INDEPENDENT CRAFT BREWERS. Disponível em: <https://www.brewersassociation.org/>. Acesso em: 12 ago. 2021.

CALAGIONE, S. **Extreme Brewing, A Deluxe Edition with 14 New Homebrew Recipes: An Introduction to Brewing Craft Beer at Home**. Quarry Books, 2012. ISBN 9781610599481. Disponível em: <https://books.google.com.br/books?id=zsWEhGF2trsC>.

CARLOS GROTT, Márcio; HUGO, Marcel. Reutilização de soluções com patterns e frameworks na camada de negócio.

CERN.INFO.CH - TIM BERNERS-LEE'S PROPOSAL. Disponível em: <http://info.cern.ch/Proposal.html>. Acesso em: 19 ago. 2021.

CERVEJA ARTESANAL TEM MERCADO QUENTE PARA EMPREENDER E CARREIRAS EM ALTA | VOCÊ S/A. Disponível em: <https://vocesa.abril.com.br/empreendedorismo/mercado-de-cervejas-artesanais/>. Acesso em: 31 ago. 2021.

CRAIG, I D. **Object-Oriented Programming Languages: Interpretation**. Springer London, 2007. (Undergraduate Topics in Computer Science). ISBN 9781846287749. Disponível em: <https://books.google.com.br/books?id=-0FrB_fXlZwC>.

ELOQUENT JAVASCRIPT. Disponível em: <https://eloquentjavascript.net/>. Acesso em: 7 ago. 2021.

FAQ: GERAL | DOCUMENTAÇÃO DO DJANGO | DJANGO. Disponível em: <https://docs.djangoproject.com/pt-br/3.2/faq/general/>. Acesso em: 18 ago. 2021.

FREEMAN, A. **Pro ASP.NET MVC 5**. Apress, 2014. (Expert's voice in ASP.NET). ISBN 9781430265306. Disponível em: <https://books.google.com.br/books?id=AqoIAwAAQBAJ>.

GENERAL PYTHON FAQ — PYTHON 3.9.6 DOCUMENTATION. Disponível em: <https://docs.python.org/3/faq/general.html#what-is-python>. Acesso em: 3 ago. 2021.

GERBER HORNINK, Gabriel; GALEMBECK, Gabriel. **Glossário cervejeiro: da cultura à ciência**. 2019.

HOME - CAMRA - CAMPAIGN FOR REAL ALE. Disponível em: <https://camra.org.uk/>. Acesso em: 12 ago. 2021.

HOME - EUROPEAN BREWERY CONVENTION. Disponível em: <https://europeanbreweryconvention.eu/>. Acesso em: 12 ago. 2021.

HUDAK, Paul. Conception, Evolution, and Application of Functional Programming Languages.

IEEE ANNALS OF THE HISTORY OF COMPUTING. **Computer**, v. 54, n. 1, 2021. ISSN 0018-9162. DOI: 10.1109/mc.2020.3045244.

JACKSON, M. **The World Guide to Beer: The Brewing Styles, the Brands, the Countries**. A.P. Publishing, 1977. ("A Quarto book."). ISBN 9780868270005. Disponível em: <https://books.google.com.br/books?id=AoFnAAAACAAJ>.

JUNGTHON, Gustavo; MACHADO GOULART, Cristian. Paradigmas de Programação.

MCKINNEY, Wes. **pandas: a Foundational Python Library for Data Analysis and Statistics**. Disponível em: <http://pandas.sf.net>. Acesso em: 10 ago. 2021.

MÜLLER, A. **Cerveja!** Editora da ULBRA, 2002. ISBN 9788575280584. Disponível em: <https://books.google.com.br/books?id=ZziMjE_85EcC>.

O QUE É DJANGO, PARA QUE SERVE E COMO USAR ESTE FRAMEWORK. Disponível em: <https://kenzie.com.br/blog/django/>. Acesso em: 19 ago. 2021.

O QUE É UM FRAMEWORK? Disponível em: <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm>. Acesso em: 19 ago. 2021.

OLIVER, G; COLICCHIO, T. **The Oxford Companion to Beer**. Oxford University Press, USA, 2012. (Oxford Companion To...). ISBN 9780195367133. Disponível em: <https://books.google.com.br/books?id=Ga4MYyZq-RMC>.

OOAD - OBJECT ORIENTED PARADIGM - TUTORIALSPOINT. Disponível em: <https://www.tutorialspoint.com/object_oriented_analysis_design/ooad_object_oriented_paradigm.htm>. Acesso em: 19 out. 2019.

PANDAS - PYTHON DATA ANALYSIS LIBRARY. Disponível em: <https://pandas.pydata.org/about/>. Acesso em: 10 ago. 2021.

PERCEIVING PYTHON PROGRAMMING PARADIGMS | OPENSOURCE.COM. Disponível em: <https://opensource.com/article/19/10/python-programming-paradigms>. Acesso em: 8 ago. 2021.

RFC2616. Disponível em: <https://datatracker.ietf.org/doc/html/rfc2616>. Acesso em: 31 ago. 2021.

RIEHLE DIPL, Dirk. Framework Design A Role Modeling Approach, 2000.

THE 4.4BSD COPYRIGHT | THE FREEBSD PROJECT. Disponível em: <https://www.freebsd.org/copyright/license/>. Acesso em: 18 ago. 2021.

THE HISTORY OF PYTHON: PERSONAL HISTORY - PART 1, CWI. Disponível em: <http://python-history.blogspot.com/2009/01/personal-history-part-1-cwi.html>. Acesso em: 4 ago. 2021.

THE MTV DEVELOPMENT PATTERN - TUTORIAL. Disponível em: <https://www.vskills.in/certification/tutorial/the-mtv-development-pattern/>. Acesso em: 18 ago. 2021.

THE STORY OF DJANGO | DJANGO DESIGN PATTERNS AND BEST PRACTICES - SECOND EDITION. Disponível em: <https://subscription.packtpub.com/book/web-development/9781788831345/1/ch01lvl1sec11/the-story-of-django>. Acesso em: 18 ago. 2021.

THE WEB FRAMEWORK FOR PERFECTIONISTS WITH DEADLINES | DJANGO. Disponível em: <https://www.djangoproject.com/>. Acesso em: 16 ago. 2021.

THE WORLD WIDE WEB OF TIM BERNERS-LEE - HISTORY COMPUTER. Disponível em: <https://history-computer.com/people/the-world-wide-web-of-tim-berners-lee/>. Acesso em: 19 ago. 2021.

WHAT IS SRM (STANDARD REFERENCE METHOD)? | AMERICAN HOMEBREWERS ASSOCIATION. Disponível em: <https://www.homebrewersassociation.org/how-to-brew/what-is-srm-standard-reference-method/>. Acesso em: 12 ago. 2021.

YEAGER, N J; MCGRATH, R E. **Web Server Technology**. Elsevier Science, 1996. ISBN 9781558603769. Disponível em: <https://books.google.com.br/books?id=0jExRH3_-hQC>.