UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
Faculdade de Ciências Exatas Bacharelado em Sistemas de Informação
Ramon dos Santos Rodrigues

Desenvolvimento de sistema web para gestão de currículo acadêmico

Diamantina
2022

Ramon dos Santos Rodrigues

Desenvolvimento de sistema web para gestão de currículo acadêmico

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação, como parte dos requisitos exigidos para a obtenção título de Bacharel em Sistemas de Informação.

Orientador: Dra. Luciana de Assis Pereira

Diamantina
2022

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

FOLHA DE APROVAÇÃO

Ramon dos Santos Rodrigues

DESENVOLVIMENTO DE SISTEMA WEB PARA GESTÃO DE CURRÍCULO ACADÊMICO

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 (a): Luciana Pereira de Assis

Data de aprovação: 19/08/2022

Profa. Luciana Pereira de Assis

Faculdade de Ciências Exatas - UFVJM

Prof. Alessandro Vivas Andrade

Faculdade de Ciências Exatas - UFVJM

Prof. Rafael Santin

Faculdade de Ciências Exatas - UFVJM

Documento assinado eletronicamente por **Rafael Santin**, **Servidor (a)**, em 23/08/2022, às 16:45, 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 **Luciana Pereira de Assis**, **Servidor (a)**, em 24/08/2022, às 18:56, 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 **Alessandro Vivas Andrade**, **Servidor (a)**, em 29/08/2022, às 15:35, conforme horário oficial de Brasília, com fundamento no art. 6°, § 1°, do Decreto n° 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site

Dedico esse trabalho ao meu pai Paulo Afonso dos Santos Rodrigues (in memorian), sua lembrança me inspira e me faz persistir em ser uma pessoa cada vez melhor.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter me concedido força e perseverança em cada pensamento negativo e fraqueza durante minha trajetória acadêmica, Agradeço principalmente à minha mãe Maria Inês e minha irmã Paula, por serem meu ponto de apoio e meu porto seguro.

Agradeço à Indielly pela amizade e companheirismo e também a minha orientadora Luciana Pereira de Assis pela orientação durante este trabalho e principalmente pela passagem de conhecimento durante os levantamentos realizados.

Agradeço também aos meus amigos, especialmente aos do grupo ”Os doidão de Jamantina”que nos momentos de ansiedade e tensão me proporcionavam várias risadas me fazendo esquecer um pouco dos problemas cotidianos.

”Agradeço todas as dificuldades que enfrentei; não fosse por elas, eu não teria saído do lugar. As facilidades nos impedem de caminhar. Mesmo as críticas nos auxiliam muito. (Chico Xavier)”

RESUMO

A jornada acadêmica dos estudantes nas instituições de ensino torna-se, com frequência, bastante massiva e desgastante, isso acontece devido à má distribuição das disciplinas na grade curricular dos alunos decorrente da falta de sistemas para auxiliar as escolhas das disciplinas a serem matriculadas, de forma que elas sejam melhores distribuídas durante o curso, evitando sobrecarga de trabalho. A sobrecarga acontece geralmente, devido à escolha do aluno ao montar a grade acadêmica agrupando disciplinas com maior número de créditos ou com altos índices de retenção em um mesmo período.

Várias modelagens de Problema do Currículo Balanceado, ou BACP (*Balanced Academic Curriculum Problem*), foram propostas ao longo do tempo objetivando reduzir a carga de trabalho que os estudantes precisam cumprir nos períodos letivos.

Existem, porém, poucas ferramentas que exploram a utilização destes modelos a fim de gerar valor para gestores e alunos. Diante deste contexto, este trabalho propõe o desenvolvimento de uma aplicação Web que possibilite gestores e alunos a beneficiarem-se das modelagens BACP em situações reais, gerando grades balanceadas e personalizadas de acordo com a necessidade do aluno.

Palavras-chave: BACP. Sistemas Web. Python. CI/CD. Observabilidade.

ABSTRACT

The academic journey of students in educational institutions often becomes massive and exhausting, this happens due to the poor distribution of subjects in the students' curriculum. Several models of the BACP (Balanced Academic Curriculum Problem), have been proposed over time in order to reduce the workload that students need to fulfill during school periods. However, there are few tools that explore the use of these models in order to generate value for managers and students. Given this context, this work proposes the development of a web application that enables managers and students to benefit from BACP modeling in real situations, generating balanced and customized grids according to the student's needs.

Keywords: BACP. Web Systems. Python. CI/CD. Observability.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACID	Atomicidade, Consistência, Isolamento e Durabilidade
APM	Application Performance Monitoring
API	Application Programming Interface
BACP	Balanced Academic Curriculum Problem
BSD	Berkeley Software Distribution
CBV	Class Based Views
DDoS	Distributed Denial of Service
FBV	Functions Based Views
FURPS	Functionality, Usability, Reliability, Performance and Supportability
HTTP	Hyper Text Transfer Protocol
LTS	Long Time Support
LXC	Linux Containers
MVC	Model, Views and Controllers
MTV	Model, Templates and Views
ODM	Object Document Mapper
ORM	Object Relational Mapper
OSI	Open System Interconnection
RUP	Rational Unified Process
SGBD	Sistema Gerenciador de Banco de Dados
SQL	Structured Query Language
SSD	Solid State Disk
TCP	Transfer Control Protocol
TPS	Throughput Per Second
UFVJM	Universidade Federal dos Vales do Jequitinhonha e Mucuri
UML	Unified Modeling Language
WAL	Write Ahead Logging
WSGI	Web Server Gateway Interface

SUMÁRIO

1 INTRODUÇÃO

Existem vários problemas do mundo real que podem ser representados por conjuntos de restrições (HNICH; KIZILTAN; WALSH, 2002). Para um determinado problema, podem ser desenvolvidos diversos modelos de representação, tipos de soluções e modelagem de restrições, como o problema do balanceamento de currículo acadêmico.

O BACP (*Balanced Academic Curriculum Problem*), ou Problema de Balanceamento de Currículo, consiste na tentativa de se distribuir uniformemente as disciplinas de um determinado curso em um certo período, considerando o número de créditos e respeitando os pré-requisitos de forma que se balanceie a carga de crédito do curso (CASTRO; MANZANO, 2001), proporcionando um uso mais eficiente dos esforços empregados pelos alunos durante a trajetória acadêmica.

Além das variáveis de decisão, por exemplo, o número de créditos, período e pré-requisito, existem outros parâmetros a se considerar ao realizar o balanceamento de um currículo acadêmico. Elenque-se, por exemplo, o índice de retenção que registra o histórico de fracassos estudantis em determinada disciplina (LOPES *et al.*, 2021), o qual propõe um novo modelo descrito como BACPR (*Balanced Academic Curriculum Problem With Retention Data*), mais aprimorado do que seu antecessor.

Esta pesquisa, dessa forma, tem por objetivo o desenvolvimento de uma aplicação Web completa, seguindo as práticas mais eficientes para a construção, a manutenção e a observabilidade de software, possibilitando a aplicação de modelos BACP em utilização real e também proporcionando maior contexto de gestão para a equipe responsável pela gestão e suporte do sistema.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral deste trabalho concentra-se em apresentar a construção de uma aplicação Web, a qual irá prover uma interface para gestão de estruturas curriculares para instituições de ensino, seguindo as boas práticas de desenvolvimento e manutenibilidade do software, de modo que se viabilize uma ferramenta para o uso de alunos e professores, possibilitando a utilização de diferentes tipos de modelos BACP para geração de grade curricular.

1.1.2 Objetivos Específicos

- Conceituar o problema do balanceamento de currículo acadêmico.
- Apresentar as tecnologias para a construção da aplicação Web.
- Fornecer um sistema base para a aplicação de modelos BACP em estruturas de dados curriculares acadêmicos.

1.2 Motivação

Existem diversos modelos do BACP para balanceamento de currículos acadêmico, porém há uma carência de ferramentas que possibilitem a disponibilização e uso desses

modelos no contexto real, para uso de professores e alunos na obtenção de grades acadêmicas balanceadas. No contexto do usuário comum, ou seja, alunos e professores das outras áreas do conhecimento, é extremamente complexo o entendimento de como utilizar a modelagem BACP com o fim de compor alguma estratégia para a solução do problema elencado. Poucos trabalhos propõem uma interface de utilização de forma que se possibilite ao usuário final o benefício proveniente da aplicação desses modelos em sua jornada acadêmica. Por esse motivo, o presente trabalho apresenta uma ferramenta para a execução de algoritmos que empregam tipos de modelagem BACP, com o objetivo de abstrair as camadas mais complexas de implementação e execução das soluções em uma interface Web objetiva e de fácil acesso.

1.3 Organização

Este trabalho está estruturado em cinco capítulos. No segundo capítulo, tem-se a fundamentação teórica, contendo uma revisão bibliográfica sobre conceitos do problema do balanceamento de currículo acadêmico (BACP). O terceiro capítulo diz respeito ao detalhamento sobre o desenvolvimento do sistema, incluindo especificações, arquitetura, tecnologias utilizadas, interfaces de usuário e tecnologias para monitoramento e saúde da aplicação. O quarto capítulo apresenta as conclusões e resultados obtidos durante o desenvolvimento do trabalho.

2 REFERENCIAL TEÓRICO

Neste capítulo será conceituado o problema do currículo acadêmico balanceado (BACP) e também sobre sistemas Web. Na seção 2.1 é feito um levantamento de algumas obras da literatura com a finalidade de explorar o estado da arte ao que tange modelagem e solução de BACP, enquanto que, na seção 2.1 é feita uma revisão abordando os principais conceitos, vantagens e desvantagens da utilização de sistemas Web bem como a comunicação utilizada entre dispositivos cliente e servidor.

2.1 Problema do balanceamento de currículo acadêmico

Vários problemas relativos à situações do cotidiano podem ser representados através de modelos que são compostos por conjuntos de variáveis e restrições. Essa representação é conhecida como PSC (*Problem Solution Constraints*) ou Problema de Satisfação de Restrições. Segundo Ünal e Uysal (2014), um problema pode ser representado por diversas modelagens e também pode ser solucionado com diferentes tipos de metodologias, um exemplo é o BACP (*Balanced Curriculum Academic Problem*) que consiste na tentativa de se distribuir disciplinas em determinado período letivo, respeitando algumas restrições e levando em conta as suas variáveis de decisão (ALCANTARA; SÁ; HEINEN, 2012).

Percebe-se que no estudo de Hnich, Kiziltan e Walsh (2002), foram demonstradas possíveis maneiras de se modelar o problema de BACP, considerando que um currículo balanceado de forma otimizada, minimiza a carga máxima acadêmica para todos os períodos.

Alguns trabalhos exploram metodologias e técnicas computacionais diferentes para a implementação da solução para um modelo BACP. O trabalho de Rubio *et al.* (2013) é relevante nesse aspecto, visto que utiliza a metaheurística ACO (Ant Colony Optimization), tal qual o de Castro e Manzano (2001) que utilizam programação por restrição e programação inteira, ao que em (LOPES *et al.*, 2021), por fim, apresenta o modelo BACPR utilizando a programação inteira mista. Os modelos de representação do BACP, tem por objetivo a tentativa de otimizar a geração de uma grade curricular bem equilibrada. Tais modelos, portanto, se aplicados no contexto de execução de uma aplicação Web, podem ser de grande utilidade para professores e alunos.

Castro e Manzano (2001) definiram uma modelagem BACP com as regras subsequentes:

- Currículo acadêmico: um currículo acadêmico é definido por um conjunto de disciplinas e de relações de pré-requisitos entre estas;
- Número de períodos: disciplinas precisam estar associadas a um número máximo de períodos do curso;
- Número de créditos (peso): cada disciplina deve estar associada a uma unidade numérica, descrita como quantificador do esforço acadêmico requerido para segui-la com sucesso;
- Pré-requisitos: algumas disciplinas possuem outras disciplinas como pré-requisito;
- Número mínimo de créditos: trata-se da quantidade mínima de créditos por períodos, necessária para um aluno em tempo integral;

- Número máximo de créditos: trata-se da quantidade máxima de créditos por períodos com o objetivo de evitar sobrecarga;
- Quantidade mínima de disciplinas: trata-se da quantidade mínima de disciplinas requeridas para um aluno cursar um período;
- Quantidade máxima de disciplinas: trata-se da quantidade máxima de disciplinas que um aluno pode ter em sua grade acadêmica, de forma a evitar sobrecarga.

Segundo Monette *et al.* (2007), os trabalhos de Castro e Manzano (2001) e (HNICH; KIZILTAN; WALSH, 2002), haviam especulado como único critério de balanceamento a minimização da carga máxima de um curso. Onde C^{max} representa a carga máxima do curso; L_i, a carga do período; e m, o número de períodos. Note-se o exposto pela fórmula matemática 1:

$$C^{max} = max_{1 \leq i \leq m} L_i \tag{1}$$

Outros critérios podem ser utilizados para compor um modelo de BACP. Cite-se, por exemplo, o desvio de restrições introduzido no trabalho de Schaus, Deville e Dupont (2007), no qual essa restrição visa garantir uma atribuição a um conjunto de variáveis de acordo com uma média, esse desvio restringe um conjunto de variáveis a apresentar uma média e restringe também a soma dos desvios a essa média em questão.

A representação do balanceamento baseado no critério de desvio de restrições pode ser definido pelo caminho genérico demonstrado na equação 2, onde o objetivo é minimizar *C(p)* (MONETTE *et al.*, 2007 apud SCHAUS; DEVILLE; DUPONT, 2007).

$$C(p) = \sum_{i=1}^{m} |m * L_i - w|^p \tag{2}$$

A carga acadêmica pode ser representada pela equação: $w = \sum_{i=1}^{m} L_i = \sum_{i=1}^{n} w_i$, onde o objetivo é minimizar a carga máxima do curso (MONETTE *et al.*, 2007). Onde dado um parâmetro p instanciado com os seguintes valores: 1, 2 e ∞ obtém-se as seguintes interpretações:

- $C(1) = \sum_{i=1}^{m} |m * L_i - w|$ refere-se a soma dos desvios das médias;
- $C(2) = \sum_{i=1}^{m} (m * L_i - w)^2$ refere-se ao desvio das médias elevado ao quadrado;
- $C(\infty) = max_{1 \leq i \leq m} |m * L_i - w|$ refere-se ao desvio máximo da média;

O trabalho de Rubio *et al.* (2013), aplica a utilização de um modelo ACO (*Ant Colony Optimization*) utilizando o algoritmo BWAS (*Best Worst Ant System*) que segue três componentes principais descritos a seguir:

i *Best-Worst Pheromone Trail Update Rule*: trata-se de uma regra que reforça as arestas contidas na solução ótima e aplica penalizações em todas as arestas contidas no pior caso (Worst Case);

ii *Pheromone Trail Mutation*: consiste na alteração das trilhas de feromônios com certa probabilidade, buscando a introdução da diversidade no processo de busca. Os rastros de feromônio são modificados pela adição ou subtração da mesma quantidade de feromônio;

iii *Restart of the Search Process*: nessa etapa, refere-se ao momento em que os rastros de feromônios relacionados às arestas contidas nas soluções ótimas são muito altos e os demais estão próximos de zero (fase de estagnação). Ademais, a matriz de feromônios é ajustada para o valor inicial do feromônio.

Por meio do experimento utilizando as "formigas artificiais" do algoritmo BWAS, os autores obtiveram resultados relevantes analisando instâncias de modelagem BACP com 8, 10 e 12 períodos, respectivamente. Rubio *et al.* (2013) relata que o algoritmo BWAS resolveu o problema com uma alta eficiência, convergindo antes da centésima iteração nas instâncias.

Os apontamentos anteriores induzem a observar que houve diferentes propostas para a solução do balanceamento de currículo acadêmico, como em Lambert *et al.* (2006), utilizando algoritmos genéticos e em Rubio *et al.* (2019) — empregando otimização baseada em algoritmos dos vagalumes (*Fireflies*), que consiste em uma nova metaheurística a qual se assenta em comportamento natural para otimização de problemas, posteriormente se valendo dessa metaheurística através de programação funcional junto à linguagem de programação Haskell (RUBIO; VIDAL-SILVA; CABRERA, 2021).

O autor define que a utilização de programação funcional traz ganhos na definição e representação de variáveis e restrições, pois as soluções implementadas em Haskell apresentaram desempenho usualmente melhor quando se compara às soluções implementadas em outras linguagens como o C++ e Java (RUBIO; VIDAL-SILVA; CABRERA, 2021).

As abordagens estudadas na seção 2.1, não se preocupam com alunos que já estão no curso, ou que ingressaram na instituição de ensino através de processo de transferência, ou que sofreram alguma reprovação em sua jornada acadêmica, ou que por diferentes fatores, não conseguiram cursar determinadas disciplinas, como por exemplo, o choque de horários.

Nenhum dos trabalhos citados, propõe uma ferramenta que possibilite aos usuários a utilização dessas modelagens para que possam se orientar quanto à organização da grade de horários. Durante o processo de revisão da literatura, foi realizado a busca por ferramentas que disponibilizassem alguma interface para organização de grade curricular acadêmica e apenas a tese de Thompson-Arjona (2019) foi encontrada, no qual ele propõe uma ferramenta escrita na linguagem *PHP*, a qual os usuários carregam um arquivo com extensão *.csv* (*Comma Separated Value*), contendo os dados da grade curricular a ser balanceada.

A ideia proposta é construir e disponibilizar uma ferramenta Web com interface simplificada para que gestores, professores e alunos consigam utilizar-se de diferentes tipos de modelagem BACP para obtenção de grades balanceadas. A aplicação foi construída baseando-se em padrão de código que possibilite a implementação de diversos tipos de *solvers*[1], proporcionando maior flexibilidade ao gestor se houver necessidade de utilizar tecnologias que porventura possam surgir e melhorar o processo de balanceamento de currículo.

Visando uma maior disponibilidade de *insights* para manutenção da aplicação, bem como fornecimento de métricas sobre uso de recursos do servidor de hospedagem e também

[1] São métodos que resolvem matematicamente problemas de otimização

dados coletados relativos à experiência do usuário, foi utilizado uma *stack*[2] de código aberto descrita no capítulo 3 nas seções 3.7.1.1 a 3.7.1.4.

2.2 Sistemas Web

No cenário atual o desenvolvimento de aplicações Web, tem crescido consideravelmente, devido aos avanços tecnológicos que já decorriam de anos atrás, e vem adquirindo mais destaque diante da expressiva dependência da Internet hoje, situação de fato perceptível em razão do período de confinamento e isolamento social desencadeado pela pandemia de Covid-19 (SANTIAGO *et al.*, 2020).

Com a expansão crescente da Internet e sua principal vantagem de disseminar dados, bem como a quebra do limitador geográfico para a interconexão entre pessoas de diversos polos do globo. Isto proporciona o acesso à informação garantindo maior flexibilidade em diversas tarefas do cotidiano das pessoas. À medida que o mercado se torna cada vez mais digital, as organizações buscam adequar-se aos novos padrões, recorrendo a soluções em forma de software como maneira de otimizar seus processos (CUNHA, 2022).

Um sistema Web nada é senão um software hospedado em um servidor e que pode ser acessado através da Internet. O principal benefício está na possibilidade de acesso em qualquer lugar, sendo necessário apenas um dispositivo com navegador e acesso à Internet. As linguagens foram projetadas inicialmente para construir aplicações que funcionavam em sua grande maioria no lado cliente, ou seja, as aplicações eram instaladas nos dispositivos dos usuários para que pudessem ser utilizadas, precisaram evoluir e integrar novos protocolos de comunicação para funcionarem na camada de aplicação da Internet.

Para que um sistema Web exista, é necessário que um software em execução em um servidor com endereço IP (*Internet Protocol*) público. Este servidor, por sua vez, será acessado através de uma URL que corresponde ao seu IP público. Para que isso seja possível, é necessário a configuração de um servidor de resolução de nomes DNS (*Domain Name System*) e é ele quem faz a tradução da URL para o respectivo endereço IP.

O servidor onde a aplicação Web está em execução executa o processamento necessário de acordo com cada requisição emitida pelo usuário autenticado com acesso à aplicação, normalmente o acesso acontece por meio do navegador e a comunicação entre cliente e servidor ocorre pelo protocolo HTTP (Hypertext Transfer Protocol). A Figura 1 demonstra a comunicação entre um dispositivo com um navegador Web instalado e um servidor que contém o endereço referente ao recurso solicitado pelo usuário.

[2] Conjunto de soluções tecnológicas para fins específicos

Figura 1 – Requisição e resposta entre cliente e servidor utilizando protocolo http

Fonte: (FERNANDO, 2015)

Entretanto apesar da vantagem da remoção da limitação geográfica no acesso aos recursos disponibilizados por sistemas Web, existem alguns problemas em se manter um servidor privado para tornar o sistema acessível na Internet, como por exemplo, o alto custo de se manter a estrutura arquitetural mínima para garantir alta disponibilidade do servidor, bem como manutenção de equipamento e atualizações periódicas de segurança dos softwares, além de custos com energia elétrica e *upgrade* de recursos como memória estática e memória volátil visando a garantia de performance.

Nos últimos anos, uma nova tecnologia surgiu na cena: a tecnologia de nuvem. Em síntese, a nuvem é um modelo para criar recursos compartilhados que podem ser dinamicamente alocados e compartilhados sob demanda (MACEDO, 2017). Os benefícios alcançados são, com efeito, inúmeros. Sob título de exemplo, tem-se a hospedagem do software na internet, que evita a necessidade de aquisição de computadores com elevada capacidade de processamento e estruturas de hardware específicas para manter a eficiência e disponibilidade do sistema.

Ademais, responsabilidades como custo de manutenção, disponibilidade, atualização e segurança da infraestrutura dos servidores são geridas pelas empresas que alugam esses servidores na Web. Essa terceirização relativa à infraestrutura, garante que os desenvolvedores direcionem seu foco apenas no produto de software, minimizando drasticamente a necessidade de gestão e monitoramento dos servidores.

3 METODOLOGIA

Neste capítulo contém a descrição das metodologias utilizadas durante a fase de planejamento e desenvolvimento do sistema Web e também é feito o levantamento de tecnologias e ferramentas que serão utilizadas para a construção da aplicação.

3.1 Levantamento de Requisitos

Os requisitos para a construção da aplicação Web foram levantadas por meio de entrevista com um professor do curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri, ao qual foram propostas algumas funcionalidades de base para a disponibilidade e execução da solução de modelo de balanceamento de currículo acadêmico.

O levantamento de requisitos é peça fundamental do processo de desenvolvimento de software. Pois é nessa fase que são documentados todas as atividades que um sistema deve cumprir para se atingir o objetivo esperado pelo cliente e *stakeholders* [1]. Nesta fase, quatro fatores são considerados: entendimento do domínio da aplicação, entendimento do problema a ser resolvido, entendimento do negócio e entendimento das necessidades e restrições dos *stakeholders* (FIGUEIRA, 2012). Documentar requisitos de maneira objetiva e concisa, minimiza impactos relativos à má interpretação da equipe de desenvolvimento além de amenizar possíveis atrasos na entrega do produto.

A elicitação de requisitos ou, simplesmente, fase de levantamento de requisitos, compreende um conjunto de atividades em que os desenvolvedores utilizam para descobrir as necessidades dos *stakeholders*, buscando aplicar da melhor maneira os requisitos essenciais do sistema a ser desenvolvido. As necessidades, expectativas e recursos esperados do sistema serão levados em consideração de cada *stakeholder*. O intuito é prover de forma correta e completa o entendimento do sistema.

3.1.1 Requisitos Funcionais

De acordo com Machado (2018), os requisitos definem as características e restrições de um software, no ponto de vista do usuário. São um conjunto de restrições e objetivos estabelecidos pelo cliente com a finalidade de resolver um problema relativo ao negócio em questão. Eles dizem respeito ao comportamento do sistema, ou seja, o conjunto de funcionalidades expressado pelas necessidades dos clientes e usuários, em outras palavras, eles representam o comportamento do resultado da execução de alguma função do sistema (POHL; RUPP, 2012).

Na tabela 1 estão concentrados os principais requisitos que descrevem as funcionalidades que irão compor a finalidade da aplicação desenvolvida nesse trabalho.

[1] Usuários do sistema

Código	Identificação	Prioridade	Ator	Objetivo
RF001	Manter instituições	Alta	Admin	Esse caso de uso serve para que um usuário possa cadastrar, editar ou excluir uma instituição de ensino.
RF002	Manter conta	Média	Usuário	Esse caso de uso serve para o usuário criar, editar ou excluir sua conta de acesso.
RF003	Manter curso	Alta	Gestor	Esse caso de uso serve para que o usuário possa cadastrar, editar ou excluir um curso.
RF004	Manter disciplinas	Alta	Gestor	Esse caso de uso serve para que o usuário possa cadastrar, editar ou remover uma disciplina de determinado curso.
RF005	Manter relação entre disciplinas	Alta	Gestor	Esse caso de uso serve para que o usuário possa definir as relações de peso (valor) entre as disciplinas do curso.
RF006	Vincular a um curso	Alta	Aluno	Esse caso de uso serve para que um usuário consiga selecionar o curso em que está matriculado.
RF007	Selecionar disciplinas já cursadas	Média	Aluno	Esse caso de uso serve para que um usuário consiga selecionar as disciplinas que já cursou no momento da geração da grade curricular.
RF008	Gerar grade curricular	Média	Aluno	Esse caso de uso serve para que o usuário consiga gerar uma grade curricular balanceada de acordo com um modelo BACP.
RF009	Exportar grade curricular gerada	Média	Aluno	Esse caso de uso serve para prover ao aluno a visualização da grade gerada através de um arquivo pdf.

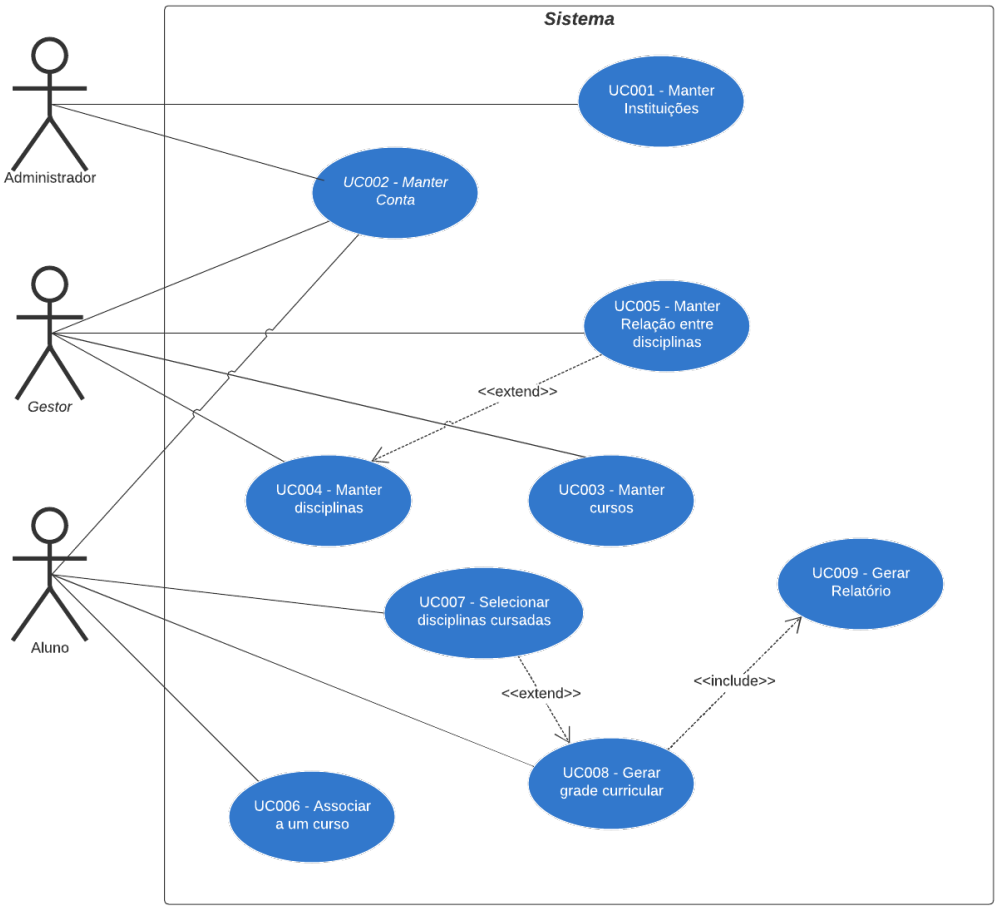
Tabela 1 – Requisitos funcionais

Na engenharia de requisitos utiliza-se algumas ferramentas para que seja possível documentar com o máximo de detalhes, as especificações, domínio da aplicação e os principais utilizadores e suas restrições relativas ao acesso das funcionalidades levantadas no processo de elicitação. Uma dessas ferramentas são os diagramas utilizando o UML (*Unified Modeling Language*), que consiste em uma linguagem de notação para uso em projetos de sistemas (VENTURA, 2021).

Sendo uma linguagem expressa com diagramas que tentam dar o máximo de detalhes em relação aos fluxos que podem conter em um software, como por exemplo, as funcionalidades e quem irá acessá-las. Existem vários tipos de diagramas providos pela linguagem UML, como diagrama de classes, diagrama de sequência, diagrama de estados, entre vários outros. Neste trabalho foi utilizado apenas o diagrama de casos de uso para ilustração das funcionalidades e seus respectivos atores; o diagrama de classes não foi utilizado devido ao padrão de desenvolvimento empregado para construção da aplicação proposta.

A Figura 2, ilustra o diagrama de casos de uso modelados para o sistema Web. No diagrama estão representados o três atores principais, sendo eles: administrador responsável pelo cadastro de instituições de ensino e demais gerenciamentos do sistema, gestor responsável pelo cadastro de cursos, disciplinas e a relação de vínculo entre as mesmas, por fim o aluno que utilizará o sistema como meio de beneficiar-se dos modelos BACP para a geração de sua grade curricular de forma equilibrada. Já o detalhamento dos requisitos descritos na tabela 1 e representados no diagrama, estão dispostos no apêndice A.

Figura 2 – Diagrama de *Caso de Uso*



Fonte: Próprio autor.

3.1.2 Requisitos de qualidade

Segundo Pohl e Rupp (2012), requisitos de qualidade ou requisitos não funcionais, são requisitos não cobertos por um requisito funcional. Cavano e McCall (1978), foram os precursores ao definirem os fatores que influenciaram na criação de métricas para mensurar a qualidade de software, organizados em três pontos de vista. Esse modelo foi um dos primeiros amplamente difundidos no campo da qualidade de software.

Figura 3 – Fatores de qualidade de McCall

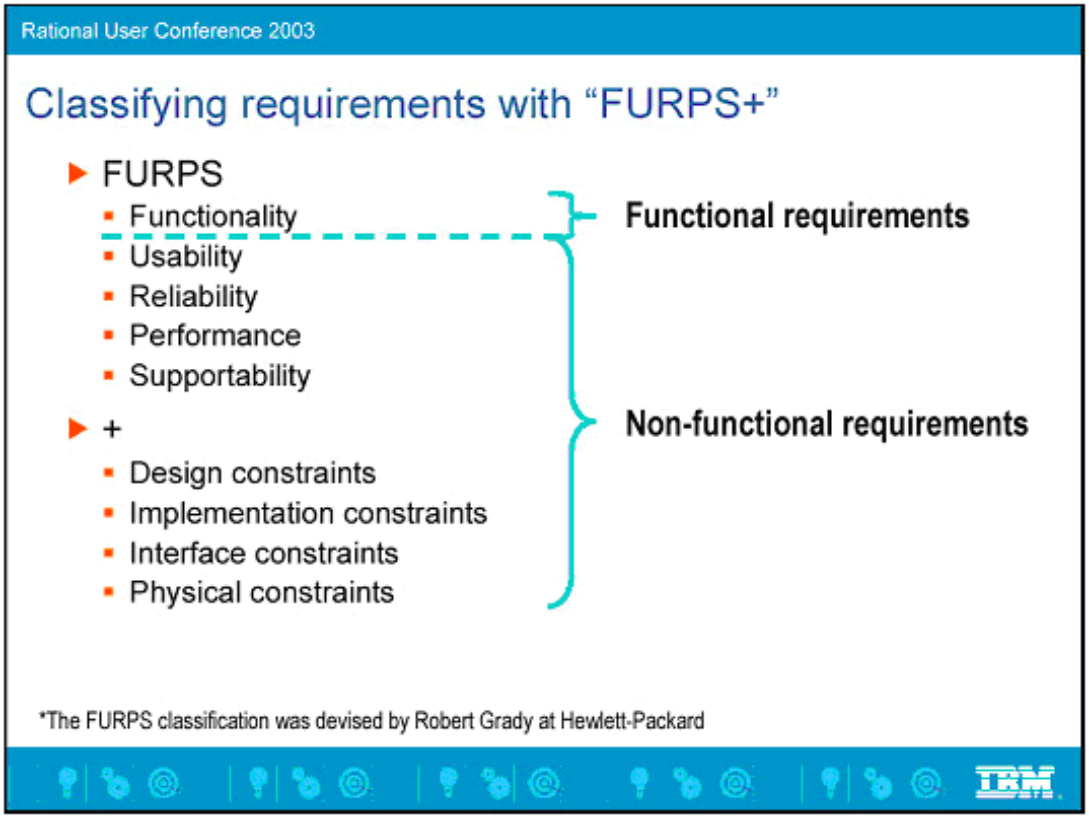
Fonte: Adaptado de (CAVANO; MCCALL, 1978)

Do ponto de vista da Operação do Produto, estão as cinco características que definem os principais aspectos a serem observados em um sistema, ilustrados na Figura 3 e descritos a seguir:

1. Correção: O quanto um programa satisfaz a sua especificação e atende aos objetivos da missão do cliente;
2. Confiabilidade: O quanto se pode esperar que um programa realize a função pretendida com a precisão exigida;
3. Usabilidade: Esforço necessário para aprender, operar, preparar a entrada de dados e interpretar a saída de um programa;
4. Integridade: O quanto o acesso ao software ou dados por pessoas não autorizadas pode ser controlado;
5. Eficiência: A quantidade de recursos computacionais e código exigidos por um programa para desempenhar sua função;

Atualmente existem outros modelos de avaliação de qualidade como por exemplo, o modelo FURPS+ (*Functionality, Usability, Reliability, Performance and Supportability*), proposto por Grady e Caswell (1987). Ele é um sistema para a classificação de requisitos, o acrônimo representa categorias que podem ser usadas na definição de requisitos, assim como representa atributos de qualidade de *software*, sendo ele parte do RUP (*Rational Unified Process*) (QUALIDADEBR, 2008). O ”+”do acrônimo, refere-se a outros requisitos não funcionais e restrições, que são demonstrados na Figura 4.

Figura 4 – Classificação de requisitos segundo o modelo FURPS+



Fonte: (QUALIDADEBR, 2008)

A norma ISO 9126 (ISO, 2003), também sugere um esquema de classificação para requisitos de qualidade, representados por uma lista com algumas categorias típicas demonstradas a seguir:

- Requisitos que definem a qualidade das funções do sistema, como: segurança de uso, segurança de dados e recursos e acurácia dos cálculos.
- Requisitos relativos a confiabilidade, robustez, tolerância a falhas e recuperabilidade do sistema.
- Requisitos que definem a usabilidade do sistema, especialmente em relação à sua compreensibilidade, facilidade de aprendizagem e facilidade de uso.
- Requisitos que definem a eficiência do sistema, especialmente em relação ao seu comportamento em termos de tempo de resposta ou de consumo de recursos.

- Requisitos que definem a modificabilidade de um sistema, especialmente em relação à sua observabilidade, estabilidade e testabilidade.

Os três modelos levantados neste estudo, são similares em seus fatores para classificar requisitos de qualidade de software, tendo o atributo ”Confiabilidade”como ponto de intersecção. Seguindo as recomendações sugeridas na norma ISO 9126 e no modelo FURPS+. Será utilizado uma arquitetura similar à arquitetura de microsserviços, a fim de se manter a proximidade com os fatores de qualidade elencado nestes dois modelos.

Serão utilizadas ferramentas como Nginx descrito na seção 3.5.1 como componente mediador de requisições HTTP e que também terá a responsabilidade de prover a disponibilidade do sistema. O Docker e o Docker Compose descritos nas seções 3.6 e subseção 3.6.1 respectivamente, são os componentes responsáveis por garantir a redundância e resiliência do sistema e seus serviços agregados, cobrindo os requisitos de tolerância a falhas e recuperabilidade. Enquanto que os requisitos de observabilidade e coleta de métricas relacionadas ao requisito de usabilidade, serão descritos nas seções 3.7 e 3.8.

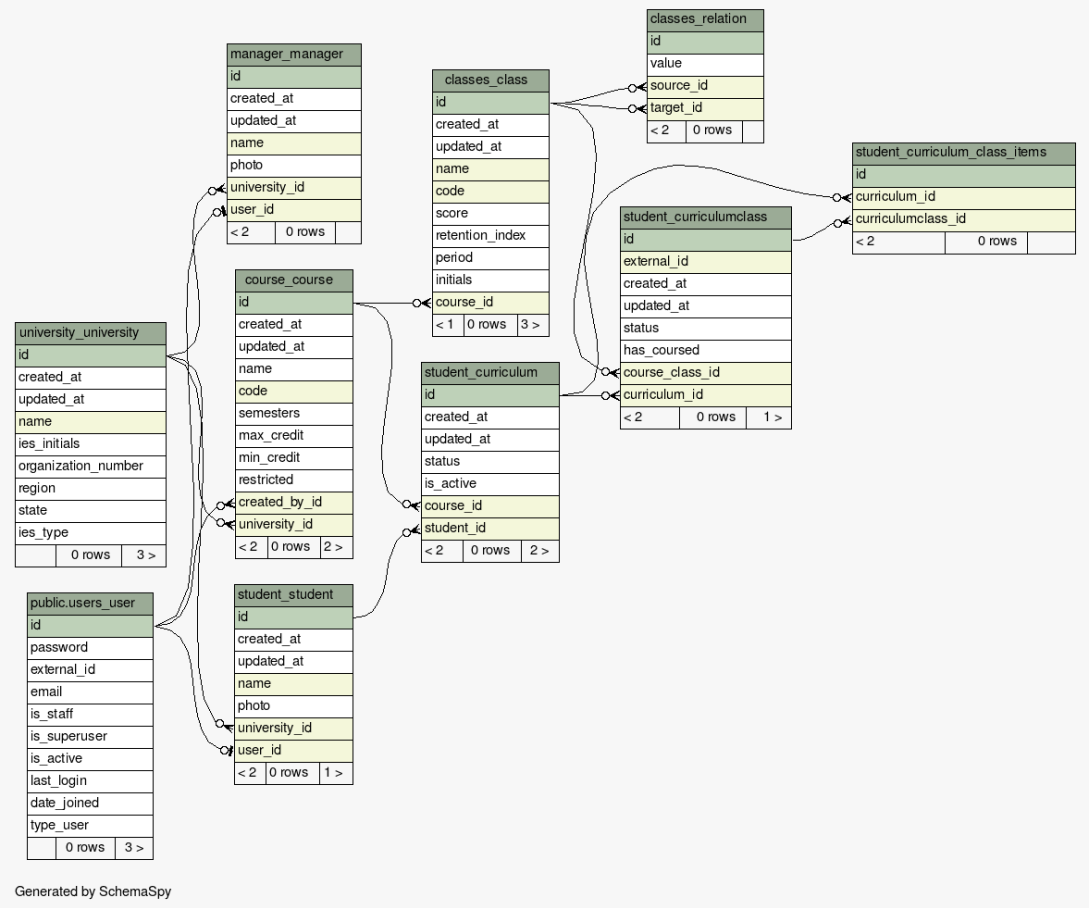
3.2 Banco de Dados

A escolha do banco de dados para manter os dados da aplicação Web proposta neste trabalho se baseou nos requisitos de gratuidade da licença de utilização e também por questões de garantia de consistência de dados a opção foi adotar uma base de dados relacional a fim de melhor atender ao escopo do projeto. O PostgreSQL foi a escolha definida pela sua ampla utilização e pela sua grande comunidade e suporte.

O Postgres é um SGBD (*Sistema Gerenciador de Banco de Dados*) Objeto-Relacional de código fonte aberto lançado em 1989. Seu nome é uma derivação da palavra ”Post-Ingres”, sendo uma variante de seu nome oficial PostgreSQL (CONRAD, 2021). O PostgreSQL usa e estende a linguagem SQL (*Structured Query Language*) e também cumpre os princípios do *ACID* (*Atomicidade, Consistência, Isolamento e Durabilidade*), possui linguagem procedural e suporte a chaves estrangeiras, controle de concorrência multi-versionado, *tablespaces* e registrador de transações sequencial WAL (do inglês *Write Ahead Logging*) para tolerância a falhas. Utiliza-se da licença *Berkeley Software Distribution* (BSD), que permite o uso para fins comerciais sem a necessidade de pagamentos de taxas (SOUZA; PEREIRA, 2020).

Na Figura 5 está a representação da modelagem de entidades relacionais do sistema, mapeado através do ORM do Django descritos na seção 3.3.1.1, seguindo os requisitos funcionais descritos na seção 3.1.1 e de qualidade elicitados na seção 3.1.2.

Figura 5 – Diagrama de Entidade e Relacionamento



Fonte: Próprio autor.

A modelagem de ER (Entidade e Relacionamento) demonstra as dependências entre as tabelas que representam as entidades contidas no sistema. A entidade Curriculum (Grade) representa uma grade acadêmica e tem ligações com as entidades Student (Aluno), Course (Curso) e CurriculumClass (Disciplina da grade) que por sua vez representa as disciplinas selecionadas pelo aluno no processo de criação da grade.

O diagrama foi gerado com o auxílio da ferramenta *SchemaSpy*. As tabelas são representadas no sistema como classes de objetos, onde o Django utiliza um mecanismo de mapeamento para a criação das tabelas no banco de dados. O *SchemaSpy*, de acordo com os parâmetros de acesso ao banco, como: *host* [2], usuário, senha, tipo do banco de dados e diretório de destino, faz a leitura das tabelas contidas no SGBD e gera os diagramas com as tabelas e seus respectivos relacionamentos.

3.3 Linguagem Python

Criada em 1990 por Guido Vonn Rossum no Instituto Nacional de Pesquisa para Matemática e Ciência da computação da Holanda, construída a partir da antiga linguagem ABC,

[2] Qualquer dispositivo tecnológico conectado a uma rede

o Python teve um foco inicial para engenheiros e físicos (BORGES, 2014). O Python é uma linguagem de código aberto e tem como característica a indentação obrigatória do código, possui estruturas de alto nível e uma ampla coletânea de módulos, alinhando à sua sintaxe clara e concisa de forma a favorecer a legibilidade de código, tornando-a uma linguagem altamente produtiva.

Por ser uma linguagem interpretada, multi-paradigma e de tipagem dinâmica, o Python possui uma versatilidade ampla para diferentes tipos de propósitos, servindo às áreas de jogos, *machine learning, data science, big data* e desenvolvimento Web. A linguagem também possui uma vasta comunidade que mantém o suporte de bibliotecas que funcionam como *plugins* provendo uma gama de funcionalidades extra aos desenvolvedores e facilitando o reuso de código e produtividade no processo de desenvolvimento de software.

Seu principal interpretador/compilador de código é o CPython implementado puramente em C, esse interpretador abstrai as complexidades do C e do sistema operacional retirando a dolorosa tarefa do gerenciamento de memória das mãos do desenvolvedor (COOKSON; STIRK, 2019). Ele tem a responsabilidade de compilar código Python em *bytecode* para depois interpretá-lo, existem outras alternativas de interpretadores como o projeto PyPy que resultou na construção de um implementador mais flexível e rápido que o CPython (LAMPKOWSKI; OLIVEIRA, 2018).

O Python também possui alguns *frameworks* poderosos que fornecem uma gama de ferramentas para construção de sistemas Web, desde *frameworks FullStack* como Django à micro *frameworks* como Flask, Starlette e FastAPI, essas ferramentas integradas tem por finalidade oferecer os recursos necessários aos desenvolvedores para que tenha disponível sem maiores esforços todos os requisitos necessários para a construção de uma aplicação Web, como por exemplo, um servidor de aplicação para executar a aplicação localmente com a finalidade de testes e também sistemas ORM (*Object Relational Mapper*) e ODM (*Object Document Mapper*) integrados, facilitando a manipulação de bancos de dados como os relacionais PostgresSQL, MySQL e os não relacionais como MongoDB e Cassandra.

3.4 Framework Django

Frente a ascensão dos sistemas Web que executam em servidores remotos e provêm acessibilidade, segurança e disponibilidade para o acesso a seus recursos, as equipes de produção do software não constroem um novo produto a partir de um projeto “em branco”, elas o desenvolvem fazendo reuso de componentes e de outros programas (SANTIAGO *et al.*, 2020). Esses programas utilizados para dar vantagem aos desenvolvedores, são conhecidos como *frameworks*, que basicamente são um conjunto de bibliotecas e ferramentas padronizadas para uma linguagem de programação específica (Ghimire, 2020 apud SANTIAGO, 2020).

Em linhas gerais, os *frameworks* são um compilado de funcionalidades e utilidades criadas para eximir a necessidade de implementar do início algoritmos que seriam comuns em determinadas situações. Cada *framework* possui uma linguagem de programação como sua base,

nesse caso o Python possui alguns *frameworks*, como por exemplo o Flask, Starlette, FastAPI e Django.

Diferente do Flask, Starlette e FastAPI, que são micro *frameworks*, o Django é uma ferramenta *fullstack*[3] completa e robusta que visa fornecer todas as utilidades necessárias para se utilizar o Python para desenvolver sistemas Web criando uma abstração dos problemas relativos ao desenvolvimento Web para que o desenvolvedor mantenha o foco nos requisitos e regras de negócio do seu projeto (SOLIQUE, 2018). Esse foi o principal motivo para a escolha deste *framework* como a ferramenta base para a construção do sistema deste trabalho, pois um dos principais focos do Django é tentar atender as necessidades da grande maioria dos desenvolvedores da maneira mais prática possível fornecendo recursos robustos sem a necessidade de instalar uma grande variedades de bibliotecas extras para colocar o sistema em funcionamento.

Dentro de seu ecossistema, é incentivado o desenvolvimento em pequenos blocos de funcionalidades chamados de *apps*, que funcionam como se fossem miniaplicativos dentro do projeto Django e de forma isolada uns dos outros, cada um com suas próprias configurações de arquivos, seguindo a premissa do modelo MTV (*Models, Templates and Views*) que será explicado na seção 3.4.1.

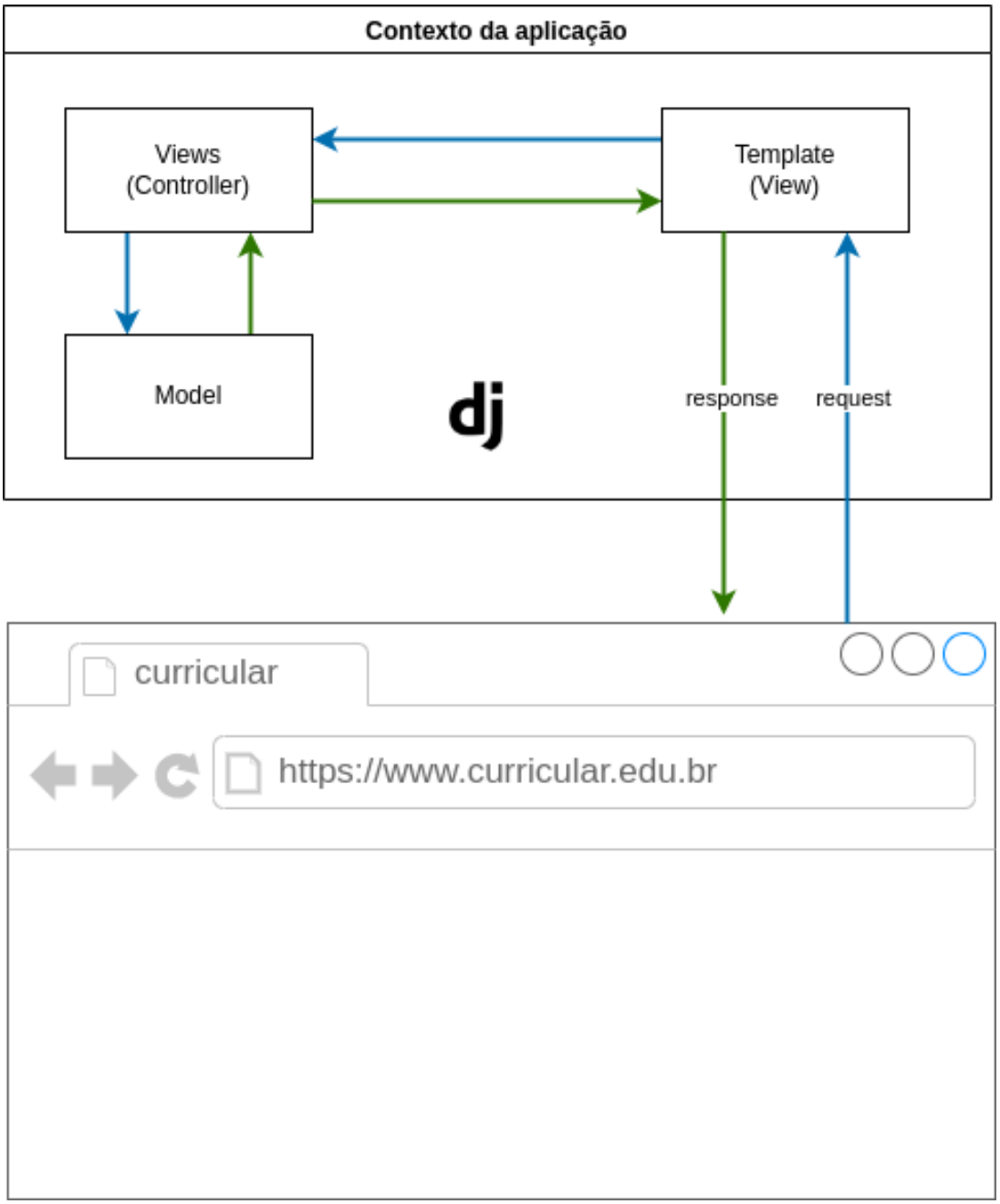
3.4.1 Arquitetura de Projeto

O *framework* Django funciona sob um padrão de projeto MTV (*Models, Templates and Views*), que é similar ao MVC (*Models, Views and Controllers*) (ANDRADE, 2018), amplamente utilizado por diversos outros *frameworks* de outras linguagens como, por exemplo, o Laravel da linguagem PHP. Na arquitetura MVC, os Models ficam responsáveis pelo gerenciamento da camada de comunicação com o banco de dados, já as Views possuem o papel de filtrar as informações e decidir o que mostrar aos usuários e prover meios de coleta de dados para os mesmos, enquanto que a regra de negócio fica a cargo dos Controllers que por sua vez, realizam o processamento dos dados de acordo com o objetivo de sua natureza, fazendo a ponte de ligação entre as camadas de Views e Models (ANDRADE, 2018).

Já no modelo modelo MTV adotado pelo Django, a dinâmica se torna um pouco diferente e podemos conferir as pequenas diferenças entre esses modelos nas subseções 3.4.1.1, 3.4.1.2 e 3.4.1.3. A Figura 6 ilustra um exemplo típico de arquitetura de um projeto Django.

[3] Característica de ferramentas que contemplam uma biblioteca completa de recursos para que seja possível agilizar o desenvolvimento de sistemas.

Figura 6 – Analogia do padrão MTV para MVC



Fonte: Próprio autor.

3.4.1.1 Model

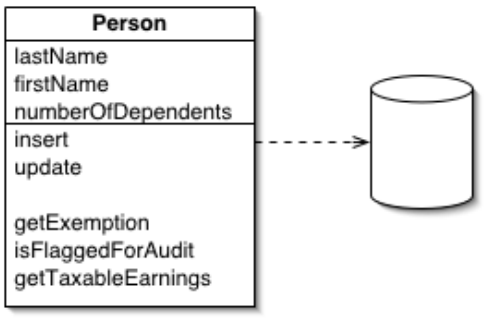
A primeira letra define o Model que é onde se concentra toda a estrutura de tabelas ou documentos dos bancos de dados, nos arquivos com extensão model.py ficam a estrutura das tabelas por meio de classes Python que são mapeadas para um banco de dados através do mecanismo de ORM (*Object Relational Mapping*) *built-in*[4] do Django, onde o ORM traduz a classe e seus atributos para criação das tabelas e campos do banco de dados relacional ou não relacional, definidos no arquivo de configuração do projeto. Essa abstração fornece uma modelagem

[4] Módulo ou funcionalidade que é nativa da ferramenta

de alto nível das entidades do banco de dados, eliminando a necessidade de criação da estrutura manualmente através do SGBD. O Django ORM, em linhas gerais é uma API (*Application Programming Interface*) que fornece uma abstração para a manipulação de dados integrando com diversos SGBD's, possibilitando flexibilidade para o desenvolvedor lidar com operações simples e ou complexas de acordo com sua necessidade.

A Figura 7 representa o esquema básico de funcionamento do ORM, enquanto que, a Figura 8 é uma representação de uma entidade que será mapeada pelo ORM do Django para o banco de dados como a tabela de disciplinas.

Figura 7 – Representação de um modelo Active Record

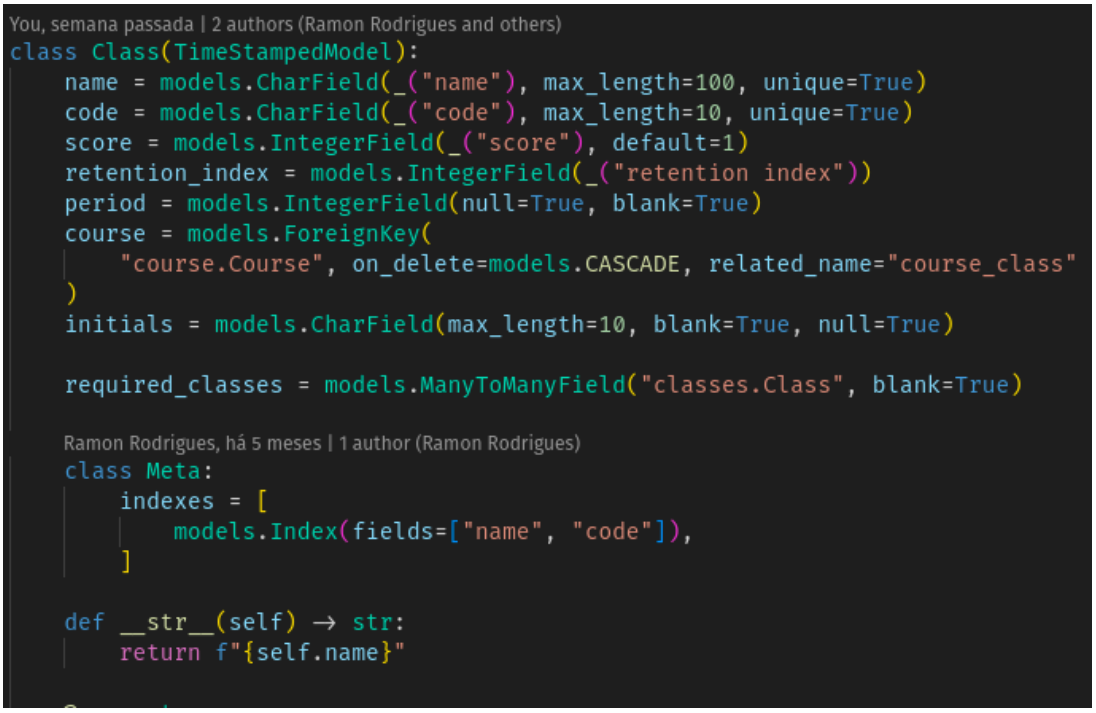


Fonte: FOWLER, et.al, 2002

O Model do Django aplica uma arquitetura baseada no *Active Record* concebida por Fowler *et al.* (2002), onde um objeto *Active Record* é um modelo de domínio, em que as classes equivalem a uma estrutura semelhante ao registro em um banco de dados, cada objeto desse tipo é responsável pela leitura e gravação de dados no banco relativos ao seu próprio domínio, incluindo também rotinas de outros tipos de relação envolvendo o domínio do dado em questão.

A Figura 8 demonstra uma classe Model, nesta classe, é definido todos os atributos de uma disciplina acadêmica, como por exemplo: nome, código, quantidade de créditos, índice de retenção, etc. Esta classe possui uma super classe definida como parâmetro de herança, denominada *TimeStampedModel* e que define dois atributos padrão: *created_at* e *updated_at*, responsáveis por registrar a data de criação de um registro e sua última modificação. Além disso, é utilizado indexação em dois atributos, sendo eles: *name* e *code*. Isso permite otimização nas consultas do banco de dados, permitindo que o SGBD localize esses registros de maneira mais performática.

Figura 8 – Classe Python representando a tabela de disciplinas



```
You, semana passada | 2 authors (Ramon Rodrigues and others)
class Class(TimeStampedModel):
    name = models.CharField(_("name"), max_length=100, unique=True)
    code = models.CharField(_("code"), max_length=10, unique=True)
    score = models.IntegerField(_("score"), default=1)
    retention_index = models.IntegerField(_("retention index"))
    period = models.IntegerField(null=True, blank=True)
    course = models.ForeignKey(
        "course.Course", on_delete=models.CASCADE, related_name="course_class"
    )
    initials = models.CharField(max_length=10, blank=True, null=True)

    required_classes = models.ManyToManyField("classes.Class", blank=True)

    Ramon Rodrigues, há 5 meses | 1 author (Ramon Rodrigues)
    class Meta:
        indexes = [
            models.Index(fields=["name", "code"]),
        ]

    def __str__(self) -> str:
        return f"{self.name}"
```

Fonte: Próprio autor.

3.4.1.2 Template

Os templates definem a camada de visualização da aplicação, é a parte alto nível apresentada ao usuário final, são páginas HTML estilizadas com CSS e funcionalidades Javascript, que apresentam a interface do sistema no navegador dos usuários. Através dos templates são realizadas as ligações entre as URL's e as funções que contém a lógica da regra de negócio. Quando o usuário acessa determinada opção da aplicação Web, como por exemplo o clique no menu de listagem de cursos disponíveis, uma requisição HTTP é enviada ao servidor que por sua vez retorna uma resposta ao navegador do usuário.

Visando aproveitar os benefícios da programação orientada a objetos, o Django provê uma linguagem de templates própria baseada na linguagem Jinja, o sistema de templates do Django foi projetado para ser confortável para quem está acostumado a trabalhar com HTML (Django Software Foundation, 2022), possibilitando o uso do conceito de herança de templates, reduzindo drasticamente a replicação de código e provendo uma melhor legibilidade e manutenibilidade das páginas HTML. As vantagens de se utilizar uma linguagem de templates de alto nível como o Jinja, se dá pela reutilização de código e na demarcação de blocos que precisam de uma renderização dinâmica da página de acordo com a funcionalidade em que ela está conectada, além de tornar possível a utilização de código Python dentro das páginas HTML através de blocos de marcação.

O mecanismo de herança de templates, utiliza delimitadores de blocos de código. Os dados são enviados pelas *views* vide seção 3.4.1.3, na forma de contexto que por sua vez são representados por estruturas de dicionário Python, essa estrutura utiliza o padrão chave e valor

para armazenamento de dados. No template HTML, é possível acessar o valor dos dados através de suas respectivas chaves que são interpoladas em blocos com chaves duplas. Abaixo, um exemplo de contexto montado em uma *view*, que em seguida será exibido em template HTML através de interpolação das variáveis enviadas, como demonstrado na Figura 9 .

```
context = {
  "course_name": "Sistemas de Informação",
  "classes": classes # lista de disciplinas recuperadas do banco de dados
}
```

Figura 9 – Página HTML utilizando a linguagem de templates do Django

```
{% extends 'base.html' %}
{% block main %}
<div class="row">
    <div class="col">
        <h5>{{ course_name }}</h5>
    </div>
</div>
<br>
<div class="row">
    <div class="col">
        <table class="table table-hover">
            <thead>
                <tr>
                    <th>Nome</th>
                    <th>Créditos</th>      Ramon Rodrigues, há 5 meses • Add project
                    <th>Disciplinas Cursadas</th>
                </tr>
            </thead>
            <tbody>
                {% for class in classes %}
                <tr>
                    <td>{{ class.name }}</td>
                    <td>{{ class.score }}</td>
                    <td>
                        <input type="checkbox" name="finished" id="">
                    </td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
    </div>
</div>
{% endblock %}
```

Fonte: Próprio autor.

3.4.1.3 View

A camada de *views* do padrão MTV do Django, é responsável por conter toda a regra de negócio da aplicação, é nesse contexto que ficam as funções contendo toda a lógica implementada da aplicação Web. Comumente definido num arquivo de nome *views* com extensão .py, as *views* são os controladores que executam os algoritmos e se comunicam com os *models* quando é necessário realizar operações de banco de dados e operações de IO. Para haver a conexão entre as funções contidas dentro do arquivo views.py com os botões, links e menus disponibilizados pelos templates, o Django fornece um esquema de rotas que são definidas em um arquivo de-

nominado *urls*, esse arquivo é responsável por conter as pontes entre funcionalidades descritas nos templates e suas respectivas funções dentro do arquivo de *views*.

No bloco 3.1, podemos visualizar um exemplo de função de *views* escrita no modelo FBV (*Function Based Views*) ou Views Baseadas em Funções.

```
1 def classes(request):
2   queryset = Class.objects.all() # recupera as disciplinas no banco de
      dados
3
4   context = {
5     "classes": queryset
6   }
7
8   return render(request, "classes/classes.html", context)
```

Listing 3.1 – Função view responsável por listar disciplinas

O Django disponibiliza duas maneiras para programação das *views*, o modelo FBV e também o modelo CBV (*Class Based Views*) ou Views Baseadas em Classe. No FBV à lógica da aplicação é implementada no formato de funções Python, como demonstrado no bloco 3.1, enquanto que no modelo CBV, a lógica é escrita no formato de classes Python, onde a classe possui uma ou mais funções que irão compor o contexto da regra de negócio a qual ela se objetiva.

No modelo CBV temos a vantagem da utilização do princípio de herança, onde conseguimos reduzir drasticamente a quantidades de linhas de código em um arquivo Python. Porém sua desvantagem se dá ao fato de correr risco de o código ficar muito acoplado e dificultar a compreensão do que está implementado, devido a abstração excessiva fazendo com que boa parte dos códigos fiquem ocultos à primeira vista.

Em resumo, o arquivo *urls* corresponde às rotas do recursos da aplicação e contém uma variável do tipo *list* nomeada por padrão como *urlpatterns*, este por sua vez, contém objetos *path* responsáveis por atribuir um *namespace* às funções. São esses *namespaces* [5] que são utilizados nas páginas de template HTML para invocar as respectivas funções. No bloco 3.2 é demonstrado um exemplo de arquivo de rotas utilizado por aplicações Django.

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5   path("disciplinas/", views.classes, name="classes"),
6 ]
```

Listing 3.2 – Exemplo de rota para listagem de disciplinas

[5] Um namespace é usado para declarar um escopo que contém um conjunto de objetos relacionados.

A variável *urlspatterns* é utilizada implicitamente pelo Django para fazer a conexão entre funções ou classes *View* a um *namespace*. Na linha 5 do bloco 3.2 dentro do módulo *path*, temos o primeiro parâmetro que corresponde ao valor a ser exibido na *url* do navegador Web, já o segundo parâmetro corresponde à função destino que será invocada, enquanto que o terceiro parâmetro é o *name*, que corresponde ao *namespace* que será usado nos templates HTML para invocar a funcionalidade.

3.5 Infraestrutura da aplicação

Nessa seção são descritas as tecnologias utilizadas na composição da infraestrutura da aplicação concernentes aos requisitos de qualidade descritos na seção 3.1.2, como disponibilidade, resiliência e escalabilidade.

3.5.1 Servidor Web

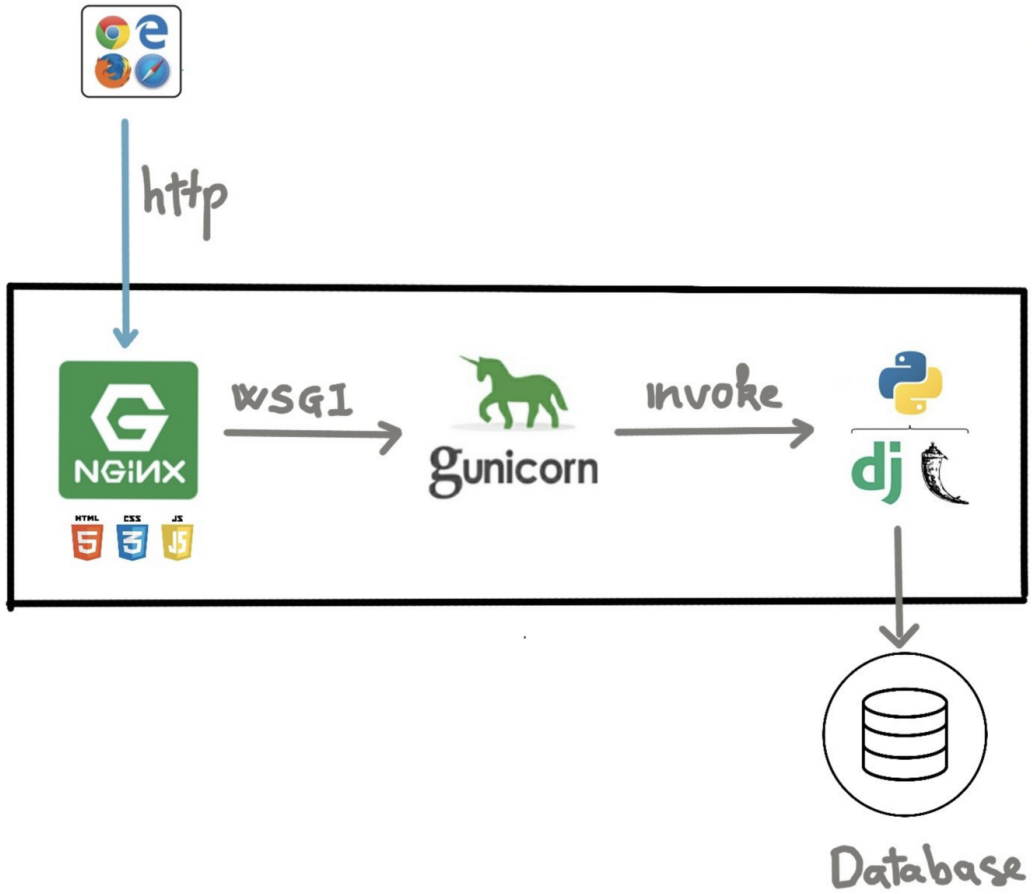
Servidores Web, em linhas gerais, são programas de computador responsáveis por entregar páginas Web sob demanda ao solicitante (VASCONCELOS, 2021), ou seja, são aplicações rodando em um servidor remoto ou local, que tenha como responsabilidade atender requisições de outras aplicações.

O Nginx (*Engine X*) é um servidor Web, proxy reverso e proxy de e-mail escrito por Igor Sysoev em 2002 (www.nginx.org), em resposta ao desafio C10K, que consistia em gerenciar dez mil conexões simultâneas. Segundo Kholodkov (2015), ele é um servidor Web de código aberto, robusto e escalável, sendo a principal escolha dos *webmasters* e gestores de *startups* devido à sua arquitetura simples e expansível. Além de possuir muitos recursos úteis, como a compactação em tempo real, armazenamento em cache, ele também fornece suporte ao balanceamento de carga para requisições HTTP, com a vantagem de ser uma aplicação com baixo consumo de recursos computacionais.

A motivação para a utilização do Nginx como servidor Web, se dá pelas vantagens elencadas no parágrafo anterior e também por sua configuração simplificada. O uso de um servidor Web se faz necessário, pois para rodar a aplicação Django, será utilizado o tradicional Gunicorn[6] (*Green Unicorn*) que consiste em uma aplicação WSGI (*Web Server Gateway Interface*) que por sua vez, não é capaz de fornecer arquivos estáticos sobre HTTP, visto que o servidor padrão disponibilizado pelo Django é contraindicado em ambientes produtivos. Portanto essa responsabilidade é delegada ao Nginx atuando como proxy reverso, repassa as requisições ao Gunicorn para que seja invocado na aplicação a funcionalidade solicitada. A Figura 10 ilustra as etapas de uma requisição HTTP à aplicação Web passando pelo Nginx e Gunicorn.

[6] O Gunicorn ”Green Unicorn”é um servidor HTTP Python WSGI. É um modelo de *worker* pré-fork, portado do projeto Unicorn da linguagem Ruby.

Figura 10 – Esquema de fluxo de requisições à aplicação Web



Fonte: (CHOU, 2020)

3.5.1.1 Load Balancer

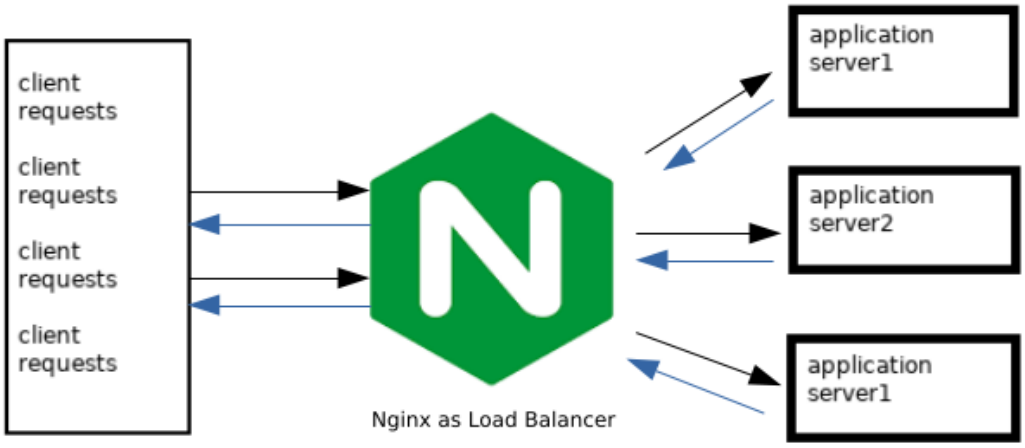
Uma dos pontos fortes do Nginx é a sua capacidade de configuração de balanceamento de carga (*Load Balancer*) para servidores HTTP e TCP[7] com escalonamento rápido (EVEO, 2022). A experiência do usuário da internet demanda velocidade e disponibilidade dos sistemas Web, para atingir esse objetivo, algumas cópias dos servidores com a aplicação Web são executadas no intuito de prover redundância e garantir a disponibilidade do sistema caso ocorra alguma falha em um dos nós. Essa técnica de arquitetura é conhecida como escalabilidade horizontal (DEJONGHE, 2019). Um dos papéis do Nginx na arquitetura da aplicação proposta nesse trabalho, é distribuir a carga de requisições entre as instâncias replicadas dos servidores HTTP rodando o sistema, dessa forma tentando evitar a queda do sistema por DDoS (*Distributed Denial of Service*) decorrente do excesso de requisições a um endereço de site.

Na Figura 11 é demonstrado o Nginx realizando o balanceamento de carga entre instâncias de aplicações web de acordo com as requisições de usuários.

---

[7] Transfer Control Protocol é um dos protocolos de comunicação, da camada de transporte da rede de computadores do Modelo OSI.

Figura 11 – Nginx atuando como Load Balancer



Fonte: Adaptado de (FARSAOUI, 2021)

De acordo com Dejonghe (2019), o Nginx fornece diversos algoritmos para balanceamento de carga, para que se possibilite uma melhor adequação de acordo com as configurações dos servidores e das necessidades de um sistema Web. O algoritmo padrão adotado por ele é o Round Robin, dependendo das configuração e dos recursos do servidores da aplicação, ele não atenderá a necessidade de todas as aplicações ou fluxos de tráfego. Diante disso, o Nginx possui algumas opções de algoritmos de balanceamento que serão descritas a seguir:

- *Round Robin*: Esse é algoritmo de balanceamento de carga padrão, que distribui solicitações na ordem da lista de servidores no *pool upstream*. A cada requisição recebida, o Nginx irá alternar entre as instâncias da aplicação disponíveis, encaminhando do primeiro servidor ao último, retornando ao primeiro servidor quando o último encaminhamento for realizado no nó final. Existe a opção de usá-lo com definição de pesos sinalizando o nível de prioridade de encaminhamento entre as instâncias (*Round Robin Ponderado*).

- *Least Connection*: Esse método equilibra a carga fazendo proxy da solicitação atual para o servidor *upstream* com o menor número de conexões abertas. Este algoritmo assim como o Round Robin Ponderado, também leva em consideração os pesos ao decidir para qual servidor encaminhar a conexão.

- *Generic Hash*: O administrador especifica um valor que será usado para gerar um *hash*. O Nginx distribui a carga entre os servidores produzindo um *hash* para a solicitação atual e colocando-o nos servidores *upstream*. Esse método é muito útil quando você precisa de mais controle sobre onde as solicitações são enviadas ou para determinar qual servidor

upstream provavelmente terá os dados armazenados em cache.

- *Random*: O Nginx basicamente seleciona um servidor do grupo *upstream* de forma aleatória, levando em consideração os pesos atribuídos aos servidores.

- *IP Hash*: Este método funciona apenas para servidores HTTP. Esse algoritmo usa o endereço IP do cliente como o *hash*, ele usa os três primeiros octetos de um endereço IPv4 ou todo o endereço IPv6. Esse método garante que os clientes sejam encaminhados para o mesmo servidor *upstream* enquanto esse servidor estiver disponível, o que é extremamente útil quando o estado da sessão é preocupante e não é tratado pela memória compartilhada do aplicativo.

Para este sistema, foi escolhido o algoritmo *Least Connection*, visto que as cargas serão distribuídas intercalando entre as instâncias que tiveram menor número de conexões abertas durante um período de tempo.

3.6 Docker

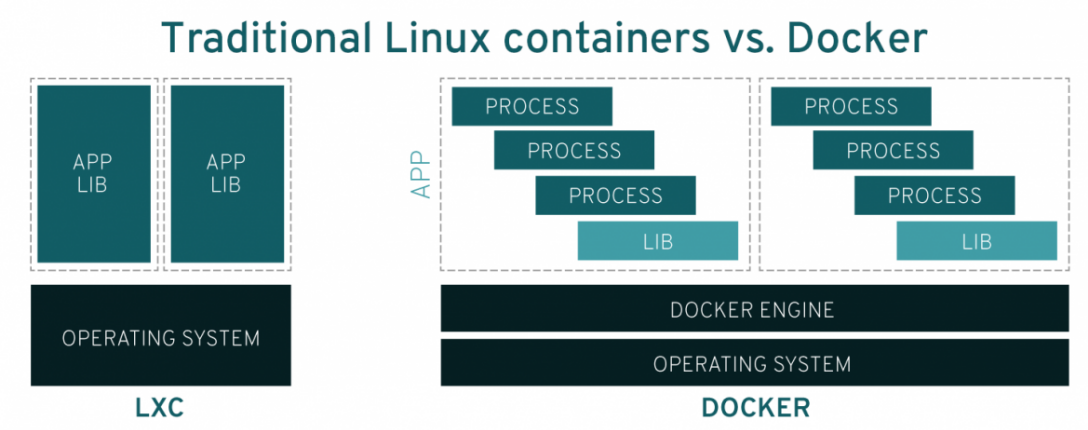
Solomon Hykes criou a dotCloud em 2008 para ser um PaaS (*Platform As A Service*), com um diferencial de não estar restrito a uma linguagem específica como era o caso do Heroku que suportava apenas a linguagem Ruby (VITALINO, 2016). Em 2013 a dotCloud decidiu abrir o código de sua plataforma, devido ao amplo suporte que recebeu da comunidade *open-source* e o sucesso iminente da ferramenta, a dotCloud passou a se chamar Docker desde então.

Docker é um projeto de código aberto para construção, *deploy* e execução de programas. É um programa de linha de comando, um processo em segundo plano e um conjunto de serviços remotos que adotam uma abordagem logística para resolver problemas comuns de software e simplificar a experiência de instalação, execução, publicação e remoção de software. Ele faz isso usando uma tecnologia de sistema operacional chamada contêineres (NICKOLOFF *et al.*, 2019). Segundo Anderson (2015), o Docker é análogo a uma máquina virtual, porém muito mais leve em comparação com as outras disponíveis, como por exemplo o VMWare e VirtualBox.

A tecnologia Docker usa o *kernel* do Linux e recursos do *kernel* como *Cgroups* e *namespaces* para segregar processos. Dessa forma, eles podem ser executados de maneira independente. O objetivo dos contêineres é criar essa independência, proporcionando a habilidade de executar diversos processos e aplicações separadamente para utilizar melhor a infraestrutura e ao mesmo tempo, manter a segurança que você teria em sistemas separados (Red Hat, 2018). Contêineres e tecnologia de isolamento já estão presentes no Linux desde 2008 conhecidos como *Contêineres Linux* (*LXC*), o Docker não fornece a tecnologia de contêineres, porém ele provê a simplificação para sua utilização (NICKOLOFF *et al.*, 2019).

O LXC é uma interface de baixo nível que fornece serviços de contenção de recursos do *kernel* Linux, permitindo a gestão de contêiner e ambientes. O Docker é basicamente um facilitador para a manipulação de contêineres, baseando sua configuração no LXC, porém com ferramentas mais aprimoradas (MENDES; DUARTE, 2019), essa comparação é demonstrada na Figura 12.

Figura 12 – Comparativo Docker vs LXC



Fonte: (Red Hat, 2018)

Contêineres são uma unidade padronizada de software que permite que os desenvolvedores isolem o aplicativo de seu ambiente, resolvendo um problema comum que existia nas equipes de desenvolvimento de software: "funciona na minha máquina" (DOCKER, 2022). A tecnologia para manipulação de contêineres do Docker, trouxe diversos benefícios no fluxo de desenvolvimento de software, pois a tarefa de desenvolver aplicações independia do sistema operacional dos desenvolvedores, sendo necessário apenas ter a ferramenta instalada na máquina local.

O Docker fornece um modelo de implantação com base em imagem, isso facilita o compartilhamento de uma aplicação ou conjunto de serviços, incluindo todas as dependências deles em vários ambientes. O Docker também automatiza a implantação da aplicação dentro desse ambiente de contêiner (Red Hat, 2018). Visando a homogeneidade do ambiente de desenvolvimento e o controle de dependências do projeto conforme descrito nos parágrafos acima, toda a estrutura arquitetural que compõe a aplicação proposta nesse trabalho, será isolada em contêineres a fim de se garantir uma melhor gestão dos componentes do ecossistema da aplicação.

3.6.1 Docker Compose

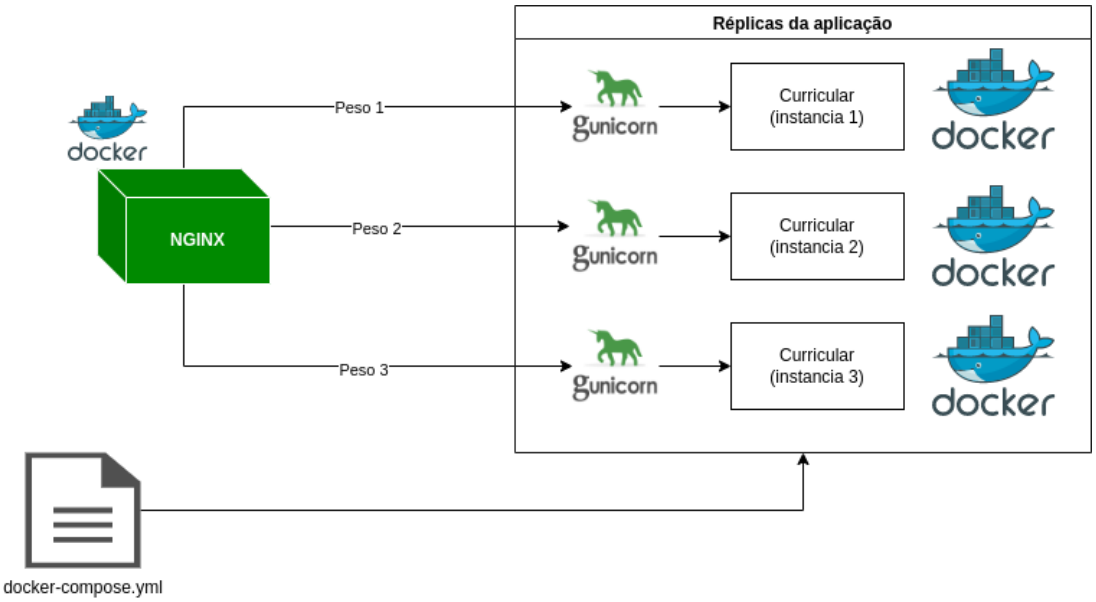
Ao definir a utilização de sistemas Web encapsulados em contêineres, obtém-se a vantagem do isolamento do ambiente da aplicação em relação ao sistema operacional da máquina local e também o controle sobre as versões das bibliotecas utilizadas no projeto de software. Ao passo que tem-se um ganho com o controle no contexto da execução das aplicações, por outro

lado a orquestração manual da execução dos contêineres e suas configurações, pode se tornar uma tarefa onerosa dependendo do tamanho do ecossistema do projeto.

Diante da necessidade de se gerenciar os contêineres de forma eficaz, é possível utilizar a ferramenta *Docker-Compose* para definir e executar aplicativos Docker em vários contêineres. Com o Compose, você usa um arquivo YAML para configurar os serviços do seu aplicativo. Então, com um único comando, você cria e inicia todos os serviços da sua configuração (Docker Inc., 2022). Através do Compose pode-se inserir os parâmetros para a criação de réplicas com a finalidade de prover a redundância dos servidores HTTP que serão usados para balanceamento de carga utilizando o Nginx como descrito na subseção 3.5.1.1.

Na Figura 13 é demonstrado o Nginx como *proxy* reverso servindo como ponte de conexão entre as solicitações de dispositivos externos e os servidores do escopo da aplicação. O Docker Compose replica as instâncias da aplicação Web através do parâmetro definido no arquivo de configuração, o Nginx então, baseado no algoritmo de balanceamento, distribui a carga entre as instâncias de acordo com o critério de distribuição.

Figura 13 – Redundância configurada no docker compose



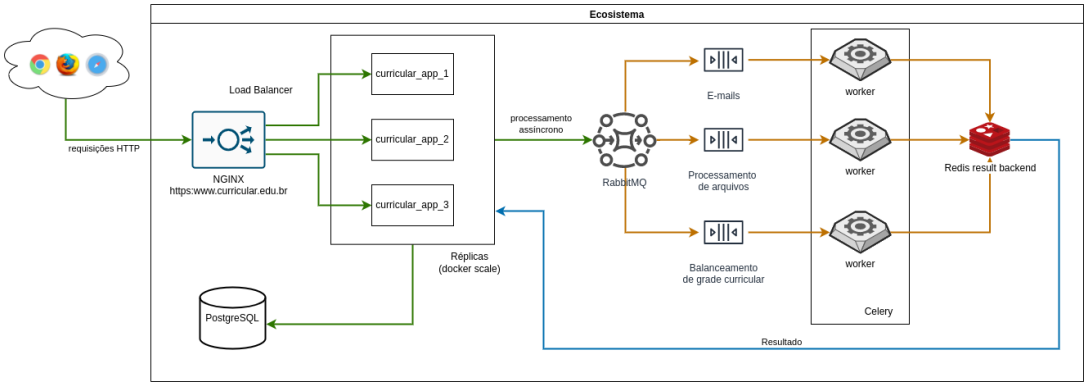
Fonte: Próprio autor

Através do arquivo de configuração YML, pode-se orquestrar o comportamento de cada contêiner, bem como a recuperação, se algum contêiner falhar o Docker imediatamente irá provisionar outro para substituir o nó que falhou. Também é possível isolar os contêineres em redes separadas conforme a necessidade (TRUCCO, 2018).

A arquitetura da aplicação demonstrada na Figura 14, apresenta os contêineres Docker de cada serviço que compõe o escopo da aplicação. Na imagem é mostrado o Nginx servindo como proxy reverso e Load Balancer em três instâncias da aplicação Web referente ao sistema proposto neste trabalho. As três instâncias são réplicas do contêiner da aplicação providas via

configuração no Docker Compose, estas instâncias são utilizadas pelo Nginx no processo de balanceamento de carga.

Figura 14 – Arquitetura da aplicação



Fonte: Próprio autor

Sistemas podem possuir funcionalidades que demandam longo tempo de processamento, não retornando o resultado imediatamente após a requisição. Funcionalidades como envio de *e-mails*, leitura de arquivos com grande quantidade de dados podem levar um tempo de resposta excessivo ocasionando erro de *timeout*[8] no servidor.

Ainda neste contexto, pode-se observar na Figura 14, o *message broker* representado pela ferramenta RabbitMQ (Cedro Technologies, 2018). Um *message broker* é um software que tem por finalidade mediar a comunicação entre sistemas distribuídos utilizando um protocolo de mensagens e as distribuindo para as partes que fazem sentido (ARRUDA; MARTINS, 2020). As mensagens neste tipo de comunicação podem representar eventos, requisições e ou *logs* de aplicação.

Este tipo de comunicação é executado de forma assíncrona e garante o desacoplamento entre os sistemas através da utilização de tópicos e filas de processamento, onde a aplicação publica uma mensagem em um tópico disponibilizado pelo *broker* e este encaminha a requisição para a respectiva fila de processamento.

As tarefas são enfileiradas até que um consumidor (*worker*) as consuma e execute o processamento. Os *workers* são providos através de uma biblioteca Python denominada Celery, que é uma biblioteca flexível e confiável para processar grandes quantidades de mensagens (CELERY, 2021). Por sua vez os resultados dos processamentos executados pelos *workers*, são armazenados no Redis que é um banco de dados *In-memmory* (REDIS, 2022) e então os resultados ficam disponíveis para que possam ser consultados pela aplicação que solicitou o processamento.

[8] Erro que ocorre no servidor quando o tempo limite de alguma transação de informações ou processamento de dados se esgota e não responde dentro do tempo mínimo exigido

3.6.2 DevOps e versionamento de código

Um fator importante do processo de desenvolvimento de software, é a rastreabilidade de alterações realizadas ao longo do processo de desenvolvimento. Isto garante que as equipes de desenvolvedores possam acessar pontos específicos de modificações de código ou inclusão de novas funcionalidades, isso é muito útil por exemplo, quando se trata de alguma alteração que subiu para o servidor de produção e acabou ocasionando algum *bug* que consequentemente trouxe instabilidade à aplicação Web ou deixou o servidor inoperante em determinado período de tempo.

O versionamento de código tem por objetivo prover esse histórico de alterações de código, dando uma maior segurança relativa ao armazenamento de código e proporcionando pontos de restauração para versões estáveis do software. Em conjunto com as ferramentas de versionamento de código existem os repositórios onde o código é de fato armazenado, esses repositórios são plataformas para armazenamento de código, que permitem aos desenvolvedores contribuam em projetos privados ou projetos *open-source* (BERTOLA, 2019).

Assim como existem os repositórios de código, como por exemplo, GitLab, GitHub e BitBucket que são as plataformas de armazenamento, temos por outro lado as ferramentas que funcionam como *clients*, como por exemplo o Git. O Git é a ferramenta responsável por marcar as alterações efetuadas no projeto de software e organizar essas mudanças em blocos com mensagens de identificação, essas alterações são chamadas de *commits* e são esses blocos que são enviados para os repositórios a fim de se manter a organização e histórico de alterações.

A plataforma de armazenamento de código selecionada neste trabalho, foi o GitLab por ter um nível gratuito de uso e principalmente por possuir uma forte integração com ferramentas de DevOps, que segundo Baldissera (2021), é um conjunto de práticas de integração entre equipes de desenvolvimento de software, operações e segurança, que tem como objetivo gerar valor ao cliente, onde a principal prática é a adoção de processos automatizados para produção rápida e segura de aplicativos e serviços.

Aproveitando-se das ferramentas nativas do GitLab para integração ou entrega contínua ou CI/CD, foi implementado conjuntos de instruções em um arquivo .yml contendo alguns passos importantes para as etapas do versionamento e *deploy*[9] das funcionalidades no servidor de produção. Os passos incluídos no pipeline de entrega contínua são: *build* da imagem Docker contendo o código do projeto, a etapa de execução de testes unitários e por fim a etapa de *deploy* para os servidores de *staging*[10] e de produção.

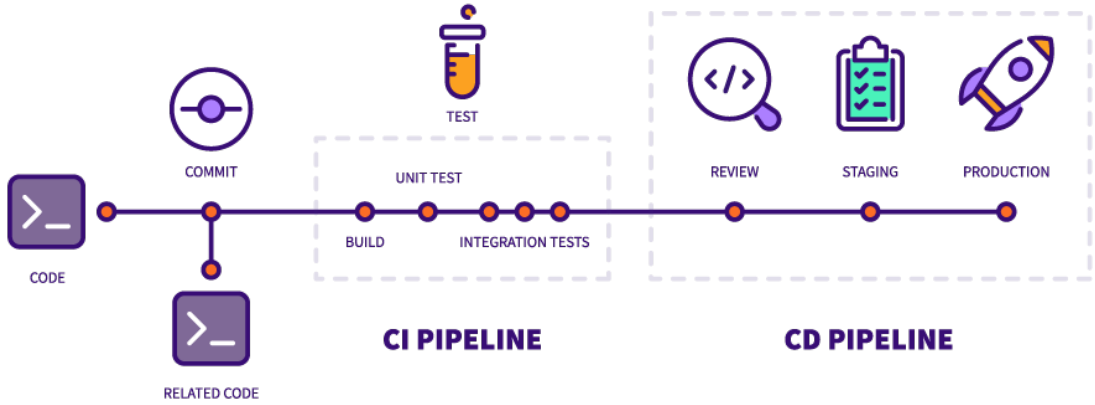
Na Figura 15 é ilustrado um fluxo de integração e entrega contínua, onde pode-se visualizar a etapa de codificação, o *commit* contendo a alteração ou inclusão de nova funcionalidade, seguido pela etapa de *build* e execução dos testes unitários. Por fim a etapa de revisão de código, *deploy* no ambiente de *staging* e ambiente de produção após as validações dos *stakeholders*.

---

[9] verbo inglês que significa implantação

[10] Ambiente para validação e teste de funcionalidades, similar ao ambiente de produção

Figura 15 – Pipeline de CI/CD no GitLab



Fonte: (STEER, 2022)

De acordo com Red Hat (2019), pipeline de entrega contínua são basicamente uma série de etapas realizadas para a disponibilização de versões de um software. Estas etapas abstraem a complexidade da implantação de software nos servidores fazendo com que os desenvolvedores se preocupem apenas na tarefa de escrever código.

3.7 Observabilidade

Imagine um cenário onde temos uma aplicação em arquitetura de monolito [11], o monitoramento dos *logs* de eventos do sistema e o gerenciamento relativo à disponibilidade da aplicação Web não seria inicialmente uma tarefa tão complexa.

Porém se o projeto evoluir para uma arquitetura distribuída como a arquitetura de microsserviços[12], teremos um maior custo para manter o monitoramento dos componentes da aplicação como um todo. Visto que monitorar os *logs* de todas as instâncias envolvidas demandaria esforço e tempo. Nesta seção, será abordado o uso do *Elastic Stack*, como plataforma de gestão e monitoramento da aplicação Web e seus serviços agregados.

3.7.1 Elastic Stack

A *Elastic Stack* é um conjunto de ferramentas construídas pela *Elastic Co.* sob uma base gratuita e aberta (ELASTIC, 2022). A *Elastic Stack* inclui diversas funcionalidades descritas no próprio *website* da plataforma, como por exemplo:

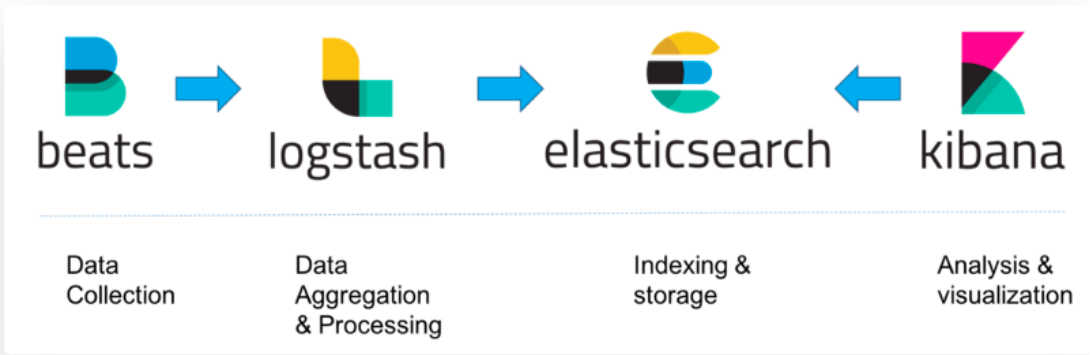
- Observabilidade e alta disponibilidade;
- Gestão e monitoramento integrado de logs;
- Acompanhamento do ciclo de vida da aplicação;
- Automatização de alertas de eventos;

[11] Arquitetura de projeto onde a aplicação é centralizada em um único projeto.
[12] Arquitetura de projeto em que a aplicação é distribuída em vários projetos, onde cada projeto implementa isoladamente alguns módulos da aplicação.

Existe um leque amplo de funcionalidades que se enquadram de acordo com a necessidade e demanda das aplicações. Nas seções 3.7.1.1 a 3.7.1.4, serão abordadas as principais soluções gratuitas e de código aberto que fizeram parte do escopo deste trabalho, sendo elas Elasticsearch, Kibana, Logstash e Beats.

A Figura 16 ilustra o fluxo geral dos componentes da ELK. O agente *Beat* descrito na seção 3.7.1.4, coleta os dados do servidor onde está instalado e envia ao *Logstash* que é responsável por realizar a normalização dos dados e então enviar esses dados indexados para o servidor do ElasticSearch. O Kibana, por sua vez, é a interface onde o usuário poderá ver esses dados organizados em *dashboards* compostos por gráficos ou mapas de acordo com a necessidade.

Figura 16 – Composição da ELK(Elastic Stack)



Fonte: (ELASTIC, 2022)

3.7.1.1 ElasticSearch

O ElasticSearch é uma ferramenta criada por Shay Banon desenvolvida em Java e de código aberto, é baseada no Apache Lucene, sob uma licença Apache (SHARMA, 2016). O ElasticSearch foi desenvolvido para ser um mecanismo de busca e análise de textos, completo e escalável. O *core* do ElasticSearch sempre está em evolução, não apenas com a inclusão de novos recursos mas também sua funcionalidade principal está constantemente sofrendo mudanças (KUC; ROGOZIŃSKI, 2016). Atualmente sua estrutura se popularizou entre os desenvolvedores, que devido à sua robustez em lidar com quantidades massivas de dados, passou também a ser utilizado como banco de dados *NoSQL* devido ao seu poder de indexação de documentos alinhado com seu poderoso motor de busca.

3.7.1.2 Logstash

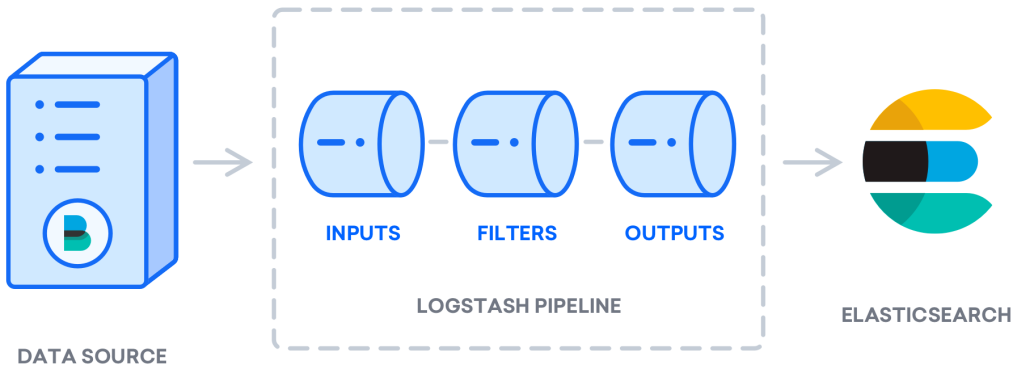
Logs contém informações de grande relevância sobre comportamentos de um sistema. Eles são gerados pelos servidores e aplicações conforme eventos significativos aconteçam (CLEMENTE, 2008). O Logstash é um componente do ELK mantido pela *Elastic Co*, que segundo a Elastic (2022), o Logstash é um pipeline *Lightweigth* [13], gratuito e aberto para pro-

[13] Aplicação ou plugin com tamanho pequeno em disco e que possuem consumo ínfimo de recursos computacionais.

cessamento de dados do lado do servidor que faz a ingestão de dados provenientes de inúmeras fontes. Ele realiza a filtragem e transformação de entrada de dados segundo um padrão configurável pelo desenvolvedor; além de tratar dados não estruturados utilizando um *plugin* chamado *grok* que permite extrair padrões e informações valiosas para tomada de decisão.

A Figura 17 demonstra um agente enviando dados para o Logstash. Esses dados passam pelo fluxo do pipeline para normalização, seguindo as etapas de entrada, filtragem e saída. Na etapa de saída é configurado a comunicação com o ElasticSearch, responsável por armazenar os dados indexados.

Figura 17 – Pipeline com Logstash



Fonte: (KALSIN; ELLINGWOOD, 2019)

Assim como o ElasticSearch, o Logstash é extensível para utilização de *plugins*; os usuários podem contribuir desenvolvendo novos *plugins*, dessa forma aumentando a flexibilidade da ferramenta (SRIVASTAVA; MILLER, 2019). Ele filtra e converte os dados em documentos *JSON*, que são enviados ao ElasticSearch.

3.7.1.3 Kibana

Kibana é um *plugin* de visualização e análise interativo de código aberto usado pela Elastic (SRIVASTAVA; MILLER, 2019). Ele fornece ao usuário diversas ferramentas para leitura dos dados processados e indexados no Elasticsearch, como por exemplo: tabelas, gráficos, *dashboards*, mapas e outros.

A interface do Kibana é baseada em *browser*, ou seja, é uma ferramenta acessada através de um navegador Web. Dentre os seus mecanismos de visualização dos dados, ele também permite o acompanhamento dos fluxos em tempo real, atualizando as informações conforme o processamento é executado.

3.7.1.4 Beats

Enquanto que o Logstash é um transformador de dados, os componentes Beats são agentes coletores de dados *Lightweight* que coletam dados a partir de várias fontes geradoras, como por exemplo arquivos, fluxo de dados ou *logs* (SELVARAJ; GUPTA, 2017). De acordo com (SRIVASTAVA; MILLER, 2019) eles são designados a propósitos específicos, como por exemplo coletar *logs* de aplicação, coletar métricas de sistema, checar se a aplicação permanece disponível, coletar dados de tráfego de rede; entre outros. Um Beat pode ser instalado diretamente no servidor onde se deseja fazer a coleta de informações, devido ao fato de serem componentes leves e não demandarem utilização expressiva de recursos. Além dos agentes desenvolvidos pela própria Elastic, existe também a possibilidade da criação de agentes customizados para propósitos ainda mais específicos, isso se torna possível devido à biblioteca *libbeat* que é o pilar para que esses componentes consigam fazer o envio de dados (ELASTIC, 2022).

A tabela 2 demonstra os *beats* oficiais criados pela Elastic e alguns do agentes criados pela comunidade *open-source*:

Desenvolvidos pela Elastic	**Finalidade**	**Desenvolvidos pela Comunidade**	**Finalidade**
Filebeat	Coletar logs	Amazonbeat	Usado para coletar dados dos serviços AWS
Metricbeat	Coletar métricas	Cloudflarebeat	Usado para coletar dados do CloudFlare
Heartbeat	Checar disponibilidade de aplicações	Dockbeat	Usado para coletar dados de containers Docker
Packetbeat	Coletar dados de tráfego de rede	Gabeat	Usado para coletar dados Google Analytic RealTime API
Auditbeat	Coletar dados de auditoria e segurança de aplicações	Gcsbeat	Usado para coleta de dados do Google Cloud Storage
Functionbeat	Coletar dados de aplicações Serveless	Redisbeat	Usado para monitoramento do Redis
Winlogbeat	Coletar dados de servidores Microsoft Windows	Pubsubbeat	Usado para coleta de dados do Google PubSub

Tabela 2 – Agentes coletores Elastic x Comunidade Open-Source

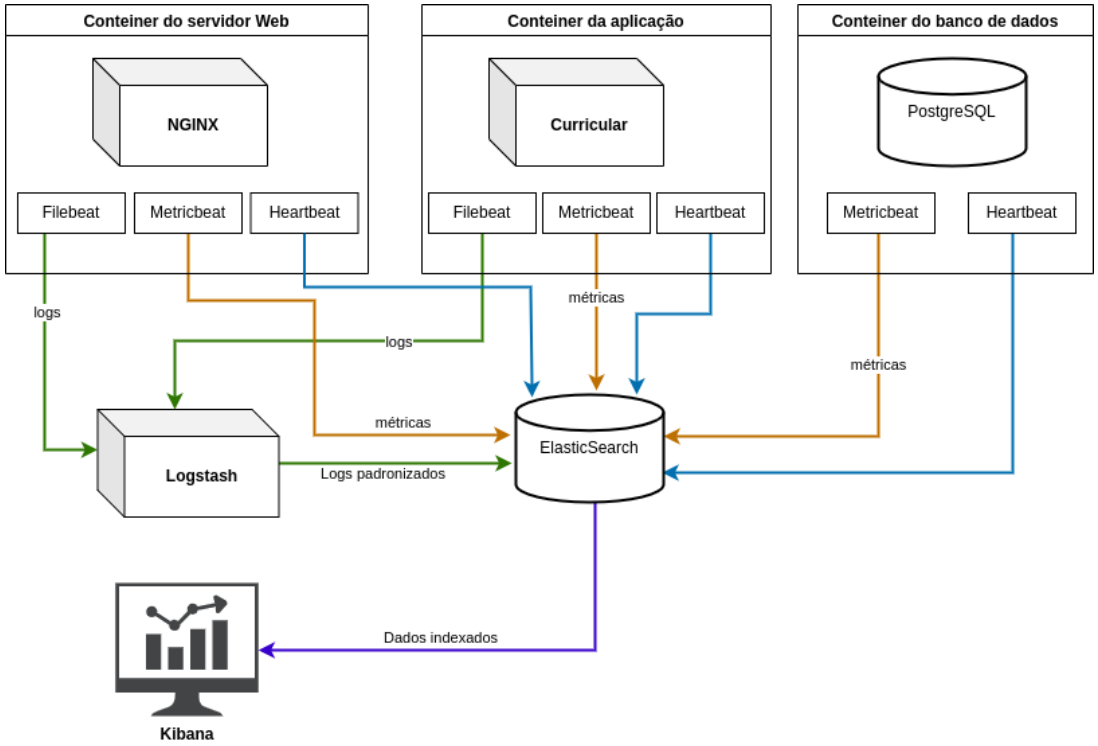
Fonte: Próprio autor

Os agentes *beats* utilizados nos componentes do ecossistema da aplicação foram: Filebeat responsável por coletar *logs* de aplicação, Metricbeat responsável pela coleta de métricas do sistemas e do contêiner Docker e o Heartbeat que é responsável por verificar se o *host* está operante através de requisições HTTP ou ICMP.

Estes coletores são instalados diretamente nos contêineres dos sistemas e possuem comunicação direta com o ElasticSearch ou podem utilizar o Logstash para tratamento dos dados antes que sejam armazenados e indexados pelo ElasticSearch. Os componentes de observabili-

dade bem como a integração entre os contêineres da aplicação, servidor web e banco de dados são demonstrados na Figura 18.

Figura 18 – Fluxo de coleta de dados com os recursos de observabilidade



Fonte: Próprio autor

3.8 APM

Ao passo que se constrói sistemas mais elaborados e com infraestrutura escalável, o aumento da complexidade em se analisar *logs* e monitorar comportamento do ecossistema da aplicação aumenta conforme as funcionalidades escalam, faz-se necessário a utilização de ferramentas centralizadoras e transformadoras de dados, tal como as ferramentas de observabilidade descritas nas seções 3.7.1.1 e 3.7.1.2.

À medida que a complexidade de um sistema aumenta, a necessidade de monitorar e analisar esses sistemas também cresce (RABL *et al.*, 2012). Várias empresas construíram ferramentas de monitoramento sofisticadas que vão muito além dos simples relatórios de utilização de recursos. Por exemplo, com base em instrumentação e *APIs* especializadas, agora é possível monitorar invocações de método único e rastrear transações individuais em sistemas geograficamente distribuídos.

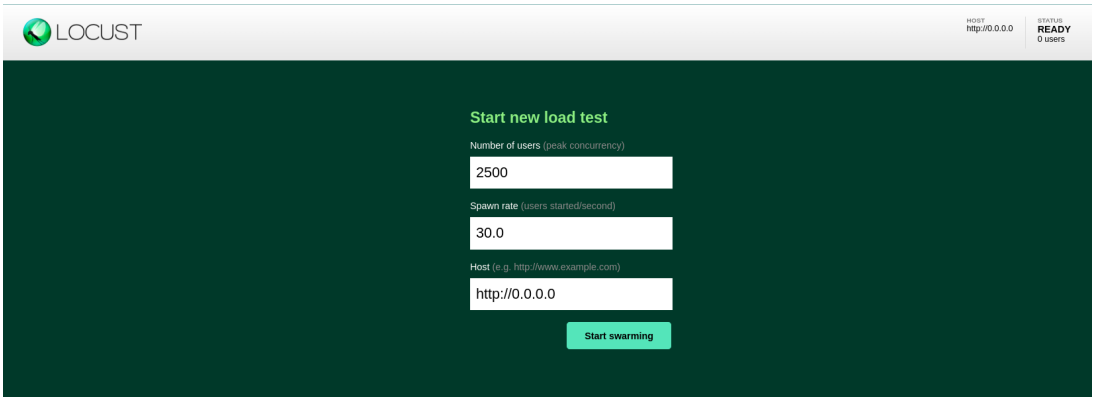
Segundo Anderson (2021), APM (*Application Performance Monitoring*) é a prática de rastrear as principais métricas de desempenho de aplicativos de software usando software de monitoramento e dados de telemetria. Os profissionais usam o APM para garantir a disponibilidade do sistema, otimizar o desempenho do serviço e os tempos de resposta, a fim de se melhorar as experiências do usuário. Uma das principais vantagens na utilização de APM's é identificar

e resolver rapidamente problemas de causa raiz com *traces*, *logs* e métricas correlacionadas (ELASTIC, 2022).

Dentre os recursos mais importantes das APM, se destaca a medição do desempenho de cada pedido/transação na Web, pois com esse recurso fica mais fácil de entender quais as requisições mais solicitadas e o porquê de uma possível lentidão (ROCHA, 2020). Para finalidade de testes de carga e comportamento do sistema, foi utilizado a ferramenta Locust. O Locust é um *framework* de testes de performance escrito em Python e de código aberto e que segundo a própria documentação, é programável via *scripts* e também escalonável.

Para o monitoramento de performance da aplicação, utilizou-se amostragem de 2500 usuários simultâneos, com uma taxa de criação de novos usuários com valor 30. Estes valores são parâmetros passados para a tela de configuração de testes do Locust como mostrado na Figura 19, isso significa que o Locust irá gerar 30 usuários a cada segundo até que se atinja a quantidade máxima de 2500 usuários simulados.

Figura 19 – Tela de configuração do teste de carga no Locust



Fonte: Próprio autor

Os testes foram executados em máquina local com os seguintes recursos: processador Intel core i5 2.40 GHz, 32 GB de memória RAM e 1.5 TB de SSD sob o sistema operacional Pop!_OS na versão 22.04 LTS 64 bits.

Conforme mostrado na Figura 20, observa-se a quantidade de requisições nas páginas: página com disciplinas para geração de grade, página de login, página de registro, página de cursos disponíveis e página com as disciplinas selecionadas, este comportamento foi configurado através de pesos para esses acessos levando-se em conta que serão uma das páginas mais acessadas do sistema.

Figura 20 – Quantidade de requisições

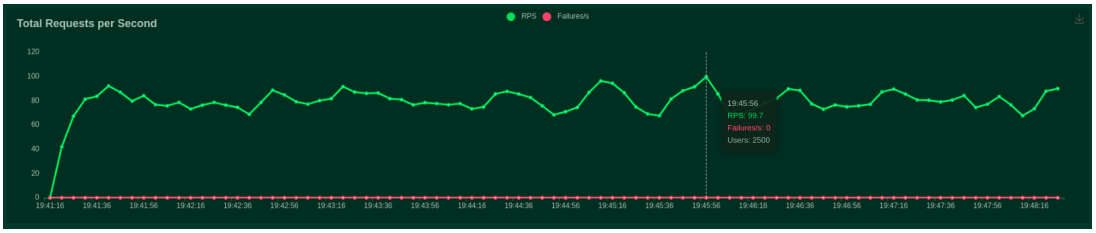
Request Statistics

Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/	2215	0	16280	25	24036	10786	5.2	0.0
GET	/alunos/cursos_disponiveis	3794	0	32223	16	45113	3618	8.8	0.0
GET	/alunos/disciplinas	4622	0	32245	19	44989	3618	10.8	0.0
GET	/gestor/cursos/lista	4498	0	32090	17	44911	3618	10.5	0.0
GET	/usuarios/login	3228	0	16110	32	24258	3618	7.5	0.0
POST	/usuarios/login	3089	0	17434	396	25317	3894	7.2	0.0
GET	/usuarios/meu_perfil	6561	0	32337	17	45038	3618	15.3	0.0
GET	/usuarios/registrar	3319	0	16418	35	24252	4484	7.7	0.0
POST	/usuarios/registrar	3174	0	16647	31	24164	4484	7.4	0.0
	Aggregated	**34500**	**0**	**25421**	**16**	**45113**	**4265**	**80.3**	**0.0**

Fonte: Próprio autor

Na Figura 21, observa-se o início do teste de carga as 19:41:16 com uma baixa quantidade de requisições por segundo e sem falhas, conforme o número de usuários aumenta, a quantidade de requisições por segundo também se eleva. Para a quantidade de 2500 usuários simultâneos gerando um total de 34.500 requisições, o sistema se manteve em pleno funcionamento e sem ocorrência de falhas.

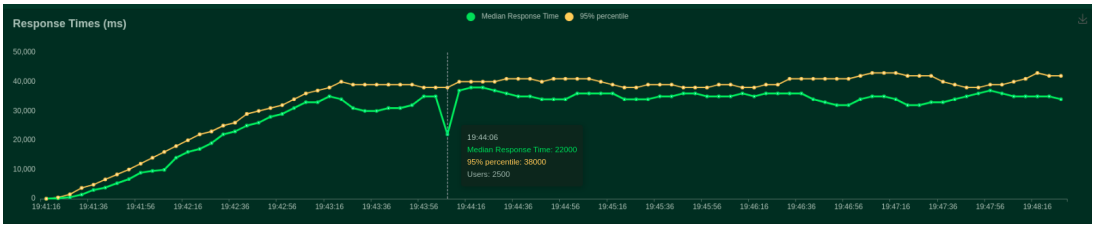
Figura 21 – Requisições por segundo x Falhas



Fonte: Próprio autor

Os resultados demonstrados na Figura 22, dizem respeito ao tempo de resposta e o uso regular e sustentado link de comunicação de rede. De acordo com Pinheiro (2019), o percentil 95 é um calculo matemático utilizado para medir o uso da largura de banda, que no gráfico é representado pela linha amarela, este valor indica se a largura de banda está acima ou abaixo 95% do tempo. Enquanto que a linha verde, indica o tempo médio de resposta para as transações efetuadas. Este cálculo já vem nativamente com o Locust e é utilizado após a execução dos testes e no momento de geração dos relatórios.

Figura 22 – Tempo de resposta - cálculo percentil 95

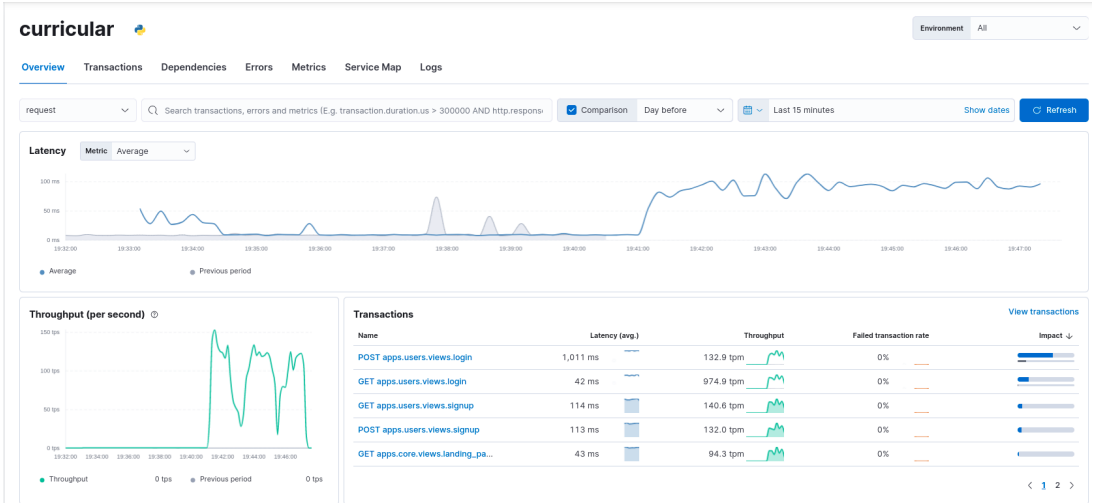


Fonte: Próprio autor

A principal vantagem da observabilidade disponibilizada via ELK, principalmente com o uso de APM para monitoramento de serviços Web se dá pela organização e concentração dos dados em um *dashboard*. Isso facilita a visualização das principais métricas coletadas durante os picos de acesso aos recursos da aplicação e também provê insumos para tomada de decisão quanto à melhorias relativas à arquitetura e tecnologias que compõe o escopo da aplicação.

Os dados de monitoramento demonstrados na Figura 23, representam as atividades ocorridas na aplicação durante um período de pico de acesso, que nesse caso foi simulado via teste de carga com a ferramenta Locust. O *dashboard* do APM da Elastic ilustra a latência no primeiro quadro onde pode ser visualizado o início do pico de acessos, seguido pelo gráfico de taxa de transferência de dados por segundo à esquerda, mostrando um pico máximo de 150 TPS (*Throughput* por segundo). No gráfico à direita contém dados de acesso por recurso, como por exemplo: latência média, taxa de transferência de dados, taxa de ocorrência de falhas e impacto.

Figura 23 – Monitoramento via APM no Kibana



Fonte: Próprio autor

As ferramentas de observabilidade provêm *insights* relevantes sobre a disponibilidade e resiliência dos recursos do sistema. Essas informações ajudam os gestores das aplicações Web a identificar gargalos de maneira mais rápida e eficiente, isso faz com que as equipes de de-

senvolvimento consigam menor tempo para identificar o problema e trabalhar na normalização do recurso com falhas.

4 RESULTADOS

Este capítulo contém as interfaces de usuário obtidas ao longo do processo de desenvolvimento da aplicação e também as conclusões inerentes à utilidade da ferramenta para prover uso em condições reais para balanceamento de currículo acadêmico.

4.1 Telas Desenvolvidas

A primeira tela apresentada ao usuário é a tela de *Landing Page*, mostrada na Figura 24, o qual apresenta ao usuário o objetivo da ferramenta. Nesta página também constam os menus para efetuar *login* e o de criar uma conta para utilização da plataforma.

Figura 24 – *LandingPage* do sistema

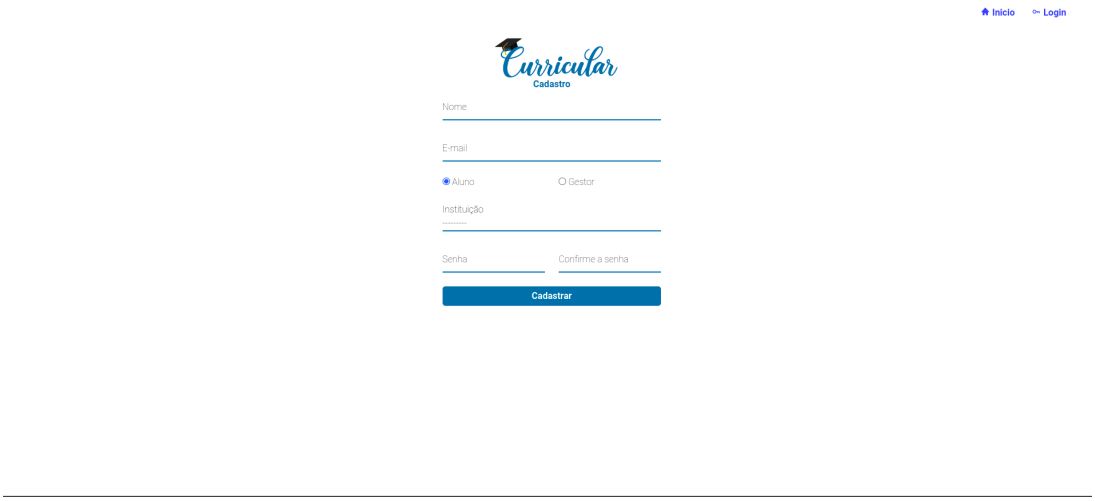
Fonte: Próprio autor

A tela principal, ilustrada na Figura 24, possui dois menus com acesso às telas de *login* e registro de nova conta demonstradas nas Figuras 25 e 26, respectivamente:

Figura 25 – Tela de login

Fonte: Próprio autor

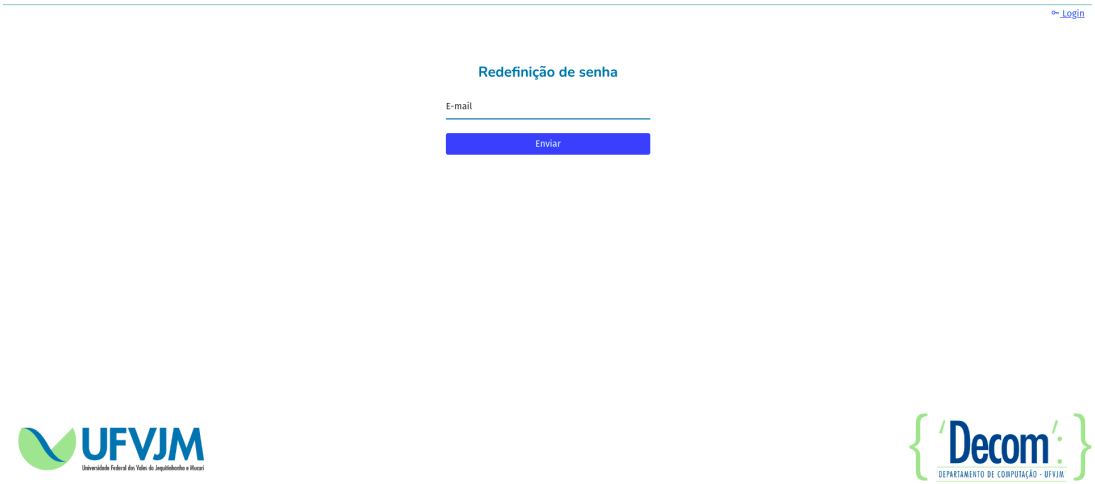
Figura 26 – Tela para criação de conta de usuário



Fonte: Próprio autor

Caso o usuário esqueça sua senha de acesso, o menu *Esqueci minha senha* apresenta a tela 27. Nesta etapa o aluno ou gestor informa seu e-mail de cadastro para que seja enviada a mensagem com instruções para redefinição da senha.

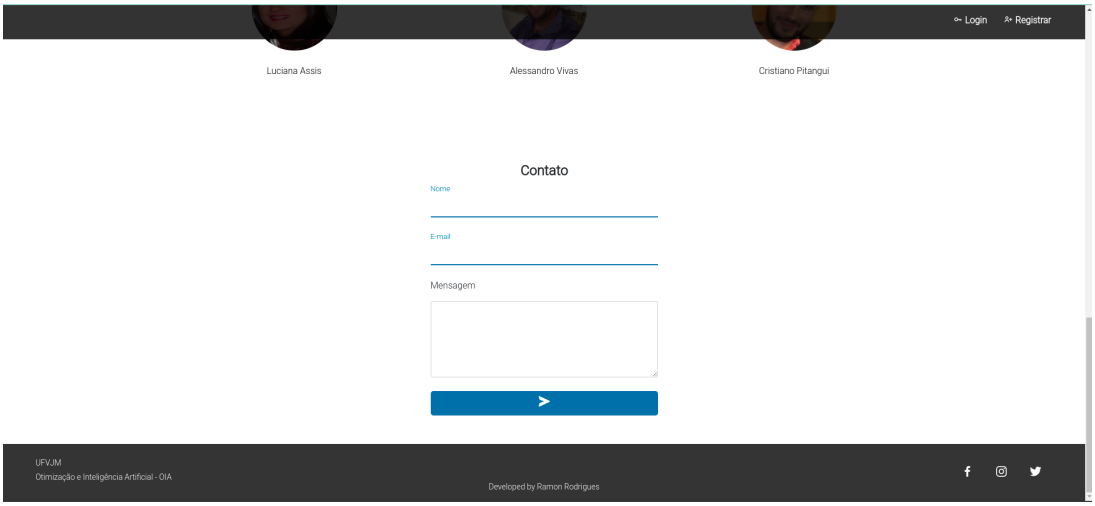
Figura 27 – Tela para redefinir senha de acesso à conta



Fonte: Próprio autor

Ao final da página de *landing page*, pode-se encontrar um formulário para envio de e-mail demonstrado na Figura 28, relativo a dúvidas ou sugestões acerca do sistema. Neste formulário o usuário informa seu nome, e-mail de contato e a mensagem que deseja enviar aos administradores do sistema.

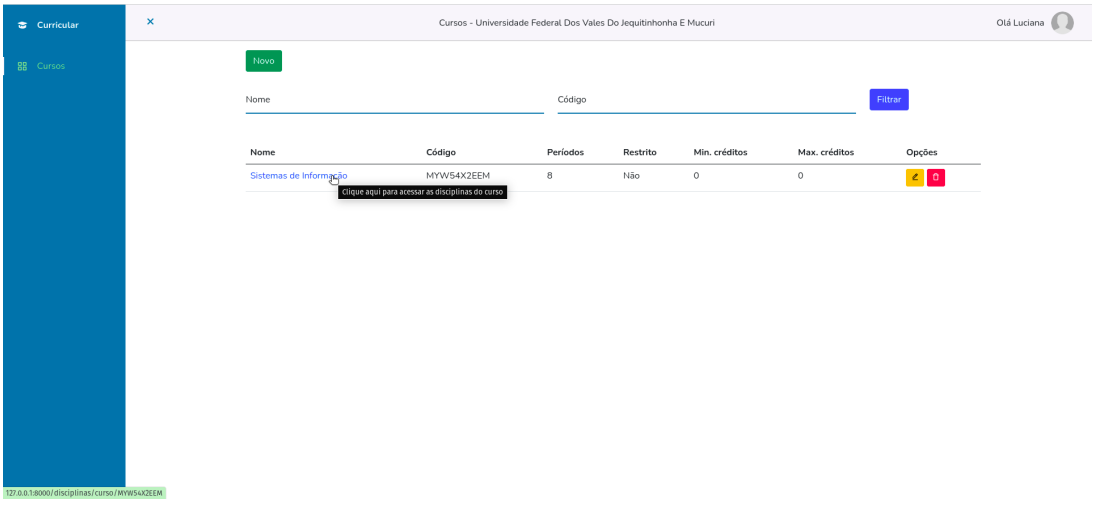
Figura 28 – Formulário para contato via e-mail



Fonte: Próprio autor

No contexto de acesso de um gestor, a tela de listagem de cursos cadastrados é apresentada no momento em que se efetua o login na conta. As opções disponíveis ao gestor são: filtrar; cadastrar; editar ou remover um curso para a universidade em que informou quando houve o cadastro de sua conta. Se o gestor já possui cursos cadastrados, é possível acessar as telas para gestão de disciplinas clicando no nome do curso em que se deseja gerenciar as disciplinas, como mostrado na Figura 29.

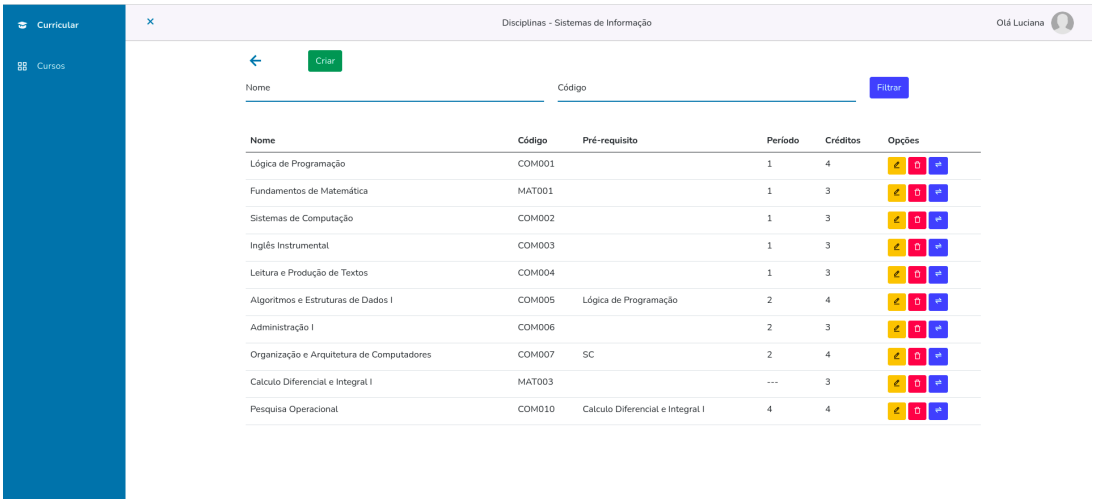
Figura 29 – Tela de listagem de cursos



Fonte: Próprio autor

Ao clicar no nome de um curso, o usuário é encaminhado para a tela demonstrada pela Figura 30. Nesta tela, por seu turno, o gestor pode exercer as funções já/há pouco mencionadas, ou seja, o gestor poderá executar todas as operações inerentes à manutenção de cursos. As relações entre disciplinas são fatores importantes para o processo de balanceamento, pois referem-se à distancia entre as disciplinas e seu grau de dependência.

Figura 30 – Tela de listagem de disciplinas



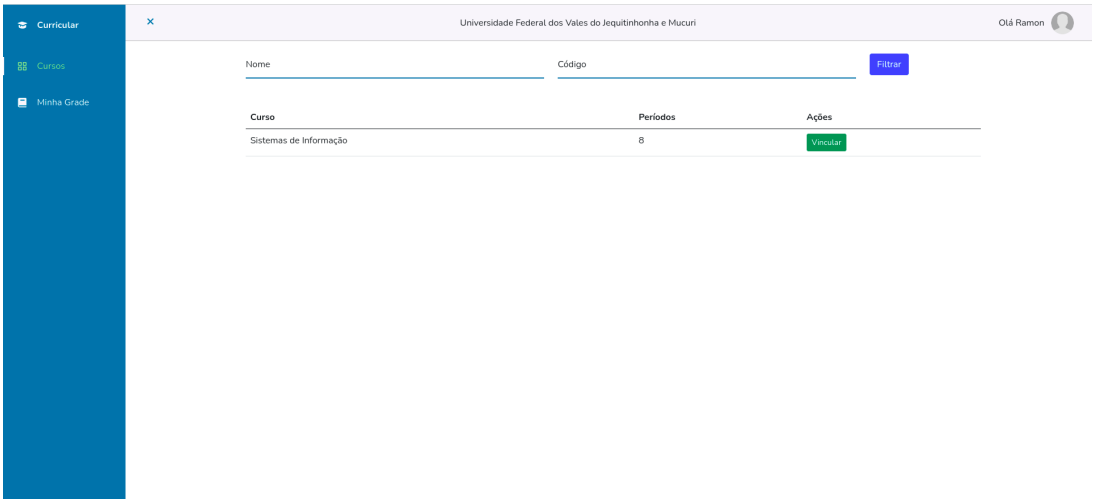
Fonte: Próprio autor

Ao cadastrar uma certa disciplina que possua outra como pré-requisito, o sistema atribui automaticamente o valor 9 para a relação entre elas. Este valor pode ser editado pelo gestor clicando no botão de *relações*, localizado logo após o botão de exclusão, mostrado na Figura 30.

No contexto do login do estudante, apresenta-se inicialmente a tela com os cursos disponíveis relacionados à universidade informada no momento do cadastro da conta. Caso o estudante tenha feito a vinculação a algum curso, apresentam-se informações referentes ao curso, p. e., o código do curso, a quantidade total de semestres, a quantidade mínima, e a máxima, de créditos e a quantidade total de disciplinas cadastradas.

A Figura 31 representa a lista de cursos disponíveis para que o aluno possa fazer a vinculação e utilizar modelagens BACP para gerar a grade acadêmica.

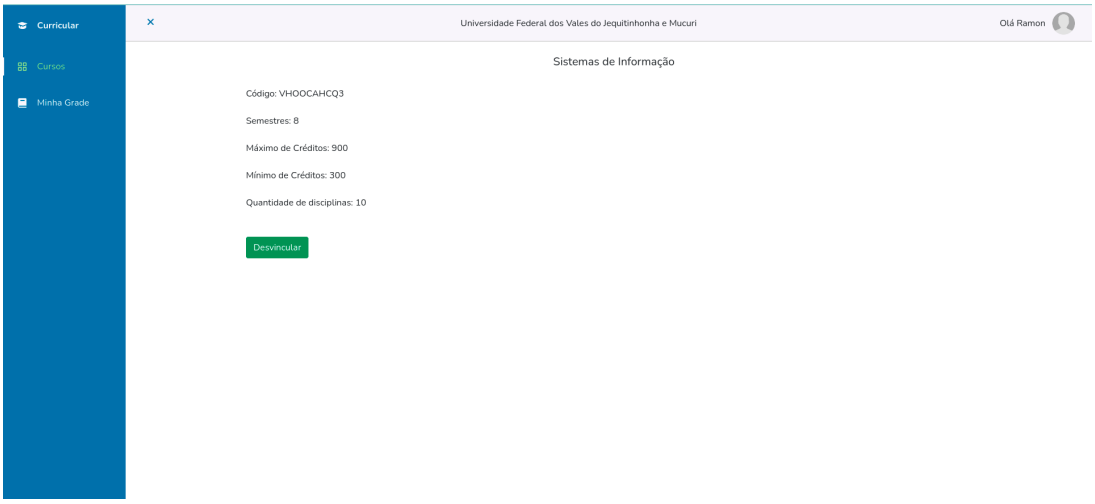
Figura 31 – Tela com os cursos disponíveis



Fonte: Próprio autor

Uma vez que um dado aluno tenha executado o passo de vinculação ao curso, clicando no botão *vincular* demonstrado na Figura 31, expõe-se a tela com informações do curso ilustradas na Figura 32.

Figura 32 – Tela com informações do curso do aluno

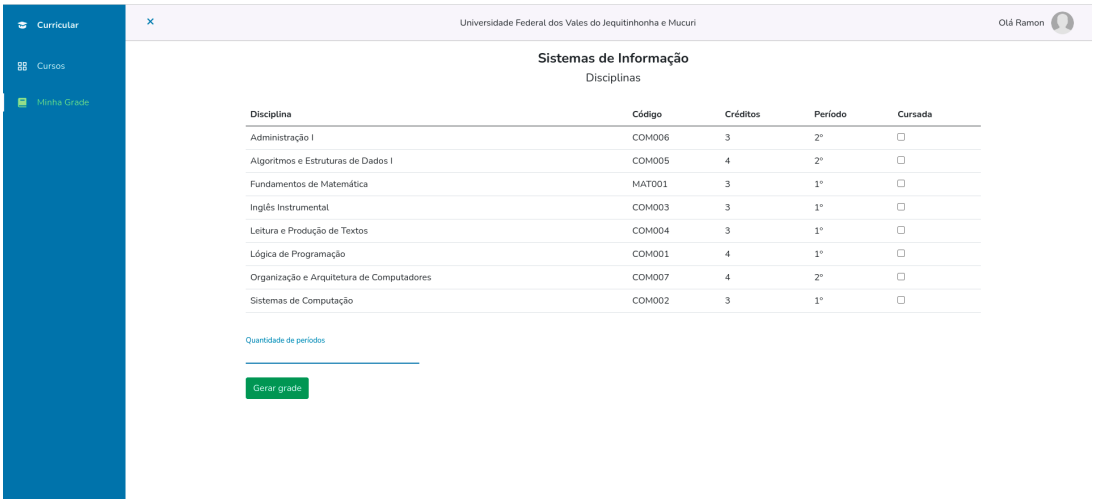


Fonte: Próprio autor

No menu *Minha Grade*, porém, encontram-se as disciplinas disponíveis relativas ao curso selecionado pelo aluno. Nesta etapa o aluno pode selecionar as disciplinas que já cursou e as que deseja excluir do processo de balanceamento. Tal opção é relevante para alunos que ingressaram na instituição de ensino através de processo de transferência.

A Figura 33 refere-se à etapa em que o aluno pode executar o processamento a fim de se obter grade curricular balanceada, clicando no botão *Gerar Grade*, o que pode também incluir a quantidade de períodos referentes ao qual se deseja realizar o balanceamento.

Figura 33 – Tela com as disciplinas do curso

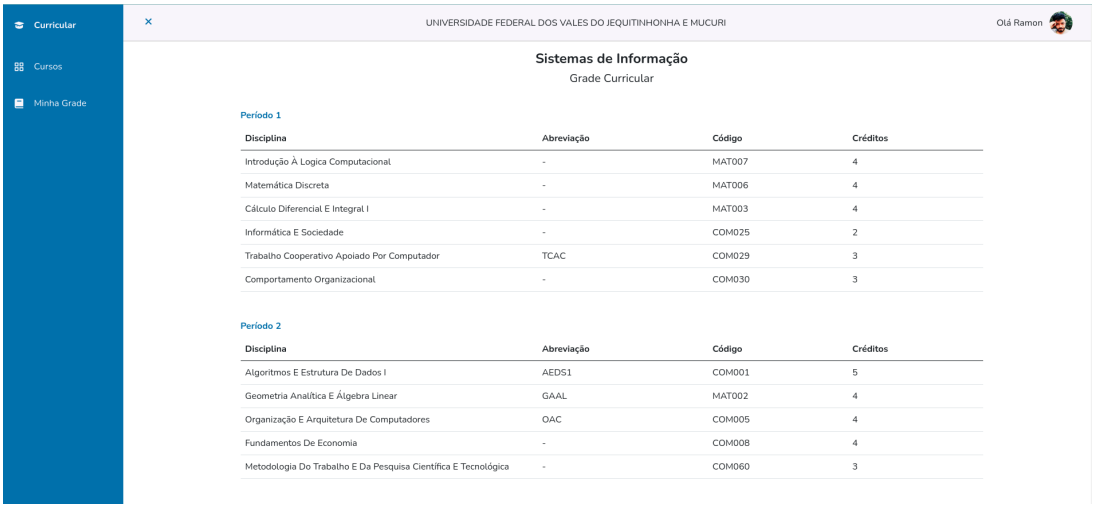


Fonte: Próprio autor

Ao clicar no botão *Gerar grade* ilustrado na Figura 33, a grade é gerada de acordo com a estratégia adotada para resolução do modelo BACP. Na Figura 34, foi utilizado o Pulp

que é uma biblioteca Python para programação linear (COIN-OR, 2009), onde o balanceamento da grade foi realizado considerando o número de créditos das disciplinas e seus pré-requisitos.

Figura 34 – Tela com a grade curricular balanceada



Fonte: Próprio autor

4.2 Conclusão e trabalhos futuros

Devido a falta de ferramentas em que possibilite aos usuários a personalização no processo de geração de grade curricular acadêmica, este trabalho teve como proposta a implementação de uma ferramenta Web como interface de uso para a resolução de modelos BACP, provendo a alunos, professores e gestores a utilidade de se gerar grades curriculares balanceadas não se limitando a apenas uma modelagem BACP específica e também considerando alunos com situação irregular no curso, ou seja, que sofreram reprovação em alguma disciplina ou ingressaram no curso através de processo de transferência.

Ao passo que se têm os modelos BACP com finalidade de amenizar a carga de trabalho estudantil, têm-se por outro lado, a complexidade de gerenciar os dados relativos ao contexto acadêmico dos alunos a fim de se obter os insumos necessários ao balanceamento da grade curricular. Por outro lado há também a gestão das métricas de acesso e dados relevantes para se manter a usabilidade e disponibilidade do sistema Web. O sistema foi construído para suportar a extensibilidade de *solvers* de programação linear, programação inteira mista e programação quadrática.

Esta extensibilidade foi provida pela utilização do *Liskov Substitution Principle* ou princípio da substituição de Liskov que é um dos princípios de design de software pertencentes ao S.O.L.I.D (MARTIN, 2015), tornando possível a adição de novas modelagens e solucionadores para a execução de balanceamento de currículo de forma facilitada e conforme a necessidade. Além do escopo principal de fornecer uma interface amigável aos usuários, foi utilizado algumas soluções no que tange a coleta de métricas e monitoramento de utilização do sistema a fim de se prover dados relevantes aos gestores da aplicação.

Sugere-se como melhorias a adicionar ao sistema, a restrição de cadastro de gestores via código de identificação, a fim de se validar contas de usuário para gestores. Outros pontos de melhorias em relação a infraestrutura e gestão dos componentes da aplicação, seria a utilização do Kubernetes como ferramenta de orquestração dos contêineres, pois o Kubernetes possibilita várias funcionalidades que abrangem vários requisitos de qualidade descritos na seção 3.1.2, como por exemplo, a configuração de *autoscale* em que o Kubernetes levanta instâncias da aplicação conforme a quantidade de acessos, provendo uma escalabilidade horizontal mais otimizada e performática.

Os *pipelines* de integração contínua no GitLab, também podem ser otimizados para que se reduza o tempo de execução, possivelmente restringindo o *build* das imagens Docker a etapas específicas, como por exemplo, o *deploy* em ambiente de *staging* e produção.

REFERÊNCIAS

ALCANTARA, F. M.; SÁ, C. C.; HEINEN, R. M. Modelagem do problema de balanceamento de currículo acadêmico utilzando programação em lógica por restrições. **Anais do XXI Seminário de Computação**, p. 83, 2012.

ANDERSON, C. Docker. **IEEE Software**, IEEE, v. 32, n. 3, p. 102–105, 2015. ISSN 07407459.

ANDERSON, D. **What is APM? Application performance monitoring | Dynatrace news**. 2021. Disponível em: <https://www.dynatrace.com/news/blog/what-is-apm-2/>.

ANDRADE, H. R. B. Django-SSTenants - Uma ferramenta para construir aplicações Multi-Tenants Django-SSTenants - Uma ferramenta para construir aplicações Multi-Tenants. 2018.

ARRUDA, F. J. d. C.; MARTINS, D. M. S. Análise Comparativa entre sistemas de mensageria: Apache Kafka vs RabbitMQ. **Seminários de Trabalho de Conclusão de Curso do Bacharelado em Sistemas de Informação**, 2020.

BALDISSERA, O. **O que é DevOps**. 2021. Disponível em: <https://posdigital.pucpr.br/blog/devops>.

BERTOLA, F. **Git, Github e Gitlab: o que são e principais diferenças | Zup**. 2019. Disponível em: <https://www.zup.com.br/blog/git-github-e-gitlab>.

BORGES, L. E. **Python para Desenvolvedores**. 2. ed. Novatec, 2014. 360 p. ISBN 9788590945116. Disponível em: <http://ark4n.files.wordpress.com/2010/01/python_para_desenvolvedores_2ed.pdf>.

CASTRO, C.; MANZANO, S. **Variable and Value Ordering When Solving Balanced Academic Curriculum Problems** □. [S.l.], 2001.

CAVANO, J. P.; MCCALL, J. A. A framework for the measurement of software quality. **Proceedings of the Software Quality Assurance Workshop on Functional and Performance Issues**, p. 133–139, 1978.

Cedro Technologies. **RabbitMQ: o que é e como utilizar**. 2018. Disponível em: <https://blog.cedrotech.com/rabbitmq-o-que-e-e-como-utilizar>.

CELERY. **Celery - Distributed Task Queue — Celery 5.2.7 documentation**. 2021. Disponível em: <https://docs.celeryq.dev/en/stable/>.

CHOU, V. **Ubuntu□□□Gunicorn + Nginx + SSL□□Django+Vue□□ | by □□□ Victor.Chou | Medium**. 2020. Disponível em: <https://vicchoutw.medium.com/ubuntu□□□gunicorn-nginx-ssl□□django-vue□□-9fd12ebc7c2c>.

CLEMENTE, R. Gerenciamento de Log. **PUC-Rio**, 2008.

COIN-OR. **Optimization with PuLP — PuLP 2.6.0 documentation**. 2009. Disponível em: <https://coin-or.github.io/pulp/>.

CONRAD, A. Database of the Year: Postgres. **IEEE Software**, v. 38, n. 5, p. 130–132, 2021. ISSN 19374194.

COOKSON, M. D.; STIRK, P. M. **CPython internals you guide to the Python 3 interpreter**. [S.l.: s.n.], 2019. ISBN 9781119130536.

CUNHA, F. **Sistema Web: o que é e como funciona? - Mestres da Web - Aplicativos**. 2022. Disponível em: <https://mestresdaweb.com.br/tecnologias/sistema-web-o-que-e-e-como-funciona/>.

DEJONGHE, D. **NGINX Cookbook: Advanced Recipes for High Performance Load Balancing**. [S.l.: s.n.], 2019. 175 p. ISBN 9781491968932.

Django Software Foundation. **The Django template language | Django documentation | Django**. 2022. Disponível em: <https://docs.djangoproject.com/en/4.0/ref/templates/language/>.

DOCKER. **Why Docker - Docker**. 2022. Disponível em: <https://www.docker.com/why-docker/>.

Docker Inc. **Overview of Docker Compose | Docker Documentation**. 2022. Disponível em: <https://docs.docker.com/compose/>.

ELASTIC. **Elastic Stack: Elasticsearch, Kibana, Beats e Logstash | Elastic**. 2022. Disponível em: <https://www.elastic.co/pt/elastic-stack/>.

EVEO. **NGINX: o que é e por que utilizar?** 2022. Disponível em: <http://blog.eveo.com.br/nginx>.

FARSAOUI, O. E. **How to configure Nginx as a load balancer (Docker + Flask + Nginx) | by Omar EL Farsaoui □□□ | FAUN Publication**. 2021. Disponível em: <https://faun.pub/how-to-configure-nginx-as-a-load-balancer-docker-flask-nginx-de95766b749b>.

FERNANDO, D. **Aplicação Web?! HTTP?! - High5Devs**. 2015. Disponível em: <http://high5devs.com/2015/05/aplicacao-web-http/>.

FIGUEIRA, A. M. d. S. Para Desenvolvimento De Software Nas Empresas De Vitória Da Conquista – Ba Para Desenvolvimento De Software Nas Empresas De. **UNIVERSIDADADE ESTADUAL DO SUDOESTE DA BAHIA**, 2012. Disponível em: <http://www2.uesb.br/computacao/wp-content/uploads/2014/09/AN{Á}LISE-DAS-T{É}CNICAS-DE-LEVANTAMENTO-DE-REQUISITOS-PARA-DESENVOLVIMENTO-DE-SOFTWARE-NAS-EMPRESAS-DE-VIT{Ó}RIA-DA-CO>.

FOWLER, M.; RICE, D.; FOEMMEL, M.; HIEET, E.; MEE, R.; STAFFORD, R. **Patterns of Enterprise Application Architecture**. [S.l.: s.n.], 2002. 560 p. ISBN 0-321-12742-0.

GRADY, R. B.; CASWELL, D. L. **Software metrics: establishing a company-wide program**. [S.l.]: Prentice-Hall, Inc., 1987.

HNICH, B.; KIZILTAN, Z.; WALSH, T. Modelling a Balanced Academic Curriculum Problem. v. 3, 2002.

ISO. ISO/IEC 9126: Qualidade de produto-Parte 1: Modelo de qualidade. **Rio de Janeiro: ABNT**, p. 1–21, 2003. ISSN 2163-1484. Disponível em: <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Engenharia+de+software+-+Qualidade+de+produto+Parte+1+:+Modelo+de+qualidade#5>.

KALSIN, V.; ELLINGWOOD, J. **Como Instalar Elasticsearch, Logstash e Kibana (Elastic Stack) no Ubuntu 18.04 | DigitalOcean**. 2019. Disponível em: <https://www.digitalocean.com/community/tutorials/como-instalar-elasticsearch-logstash-e-kibana-elastic-stack-no-ubuntu-18-04-pt>.

KHOLODKOV, V. **Nginx Essentials**. Birmingham: Packt Publishing LTD., 2015. 151 p. ISBN 9781785289538.

KUC, R.; ROGOZIŃSKI, M. Elasticsearch server. v. 3, p. 756, 2016.

LAMBERT, T.; CASTRO, C.; MONFROY, E.; SAUBION, F. Solving the balanced academic curriculum problem with an hybridization of genetic algorithm and constraint propagation. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 4029 LNAI, p. 410–419, 2006. ISSN 16113349.

LAMPKOWSKI, M.; OLIVEIRA, K. R. D. ENSAIO SOBRE APLICAÇÃO JUST IN TIME (JIT): UM ESTUDO COMPARATIVO ENTRE INTERPRETADORES PYTHON E PYPY JUST IN TIME (JIT) APPLICATION TEST : A COMPARATIVE STUDY BETWEEN PYTHON AND PYPY. p. 1–10, 2018.

LOPES, C.; ASSIS, L.; VIVAS, A.; PITANGUI, C. Análise do balanceamento do currículo do curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri. **Anais do Coned IV Congresso Nacional em Educação**, n. 38, p. 1–7, 2021.

MACEDO, D. **Entendendo as aplicações Web - Diego Macêdo**. 2017. Disponível em: <https://www.diegomacedo.com.br/entendendo-as-aplicacoes-web/>.

MACHADO, N. F. **Análise e Gestão de Requisitos de Software - Onde Nascem os Sistemas**. [s.n.], 2018. 288 p. ISBN 9788536509693. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=MYdiDwAAQBAJ&oi=fnd&pg=PT5&dq=requisitos+funcionais&ots=Dk_chEfXK2&sig=lj3jvC49pBiQLRvoqG5j1G3wM4U#v=onepage&q=requisitosfuncionais&f=false>.

MARTIN, R. C. **Arquitetura Limpa O guia do Artesão para Estrutura e Design de Software**. [s.n.], 2015. v. 7. 129–134 p. ISSN 17549469. ISBN 9772081415. Disponível em: <https://www.researchgate.net/publication/269107473_What_is_governance/link/548173090cf22525dcb61443/download%0Ahttp://www.econ.upf.edu/$\sim$reynal/Civilwars_12December2010.pdf%0Ahttps://think-asia.org/handle/11540/8282%0Ahttps://www.jstor.org/stable/41857625>.

MENDES, A. R. L.; DUARTE, A. A. Comparativo entre DOCKER e LXC/LXD para a virtualização. **Anais da Escola Regional de Computação Bahia, Alagoas e Sergipe (ERBASE)**, p. 295–303, 2019. Disponível em: <https://sol.sbc.org.br/index.php/erbase/article/view/8990>.

MONETTE, J.-N.; SCHAUS, P.; ZAMPELLI, S.; DEVILLE, Y.; DUPONT, P. A CP Approach to the Balanced Academic Curriculum Problem. **Proceedings of the Seventh International Workshop on Symmetry and Constraint Satisfaction Problems**, 2007.

NICKOLOFF, J.; KUENZLI, S.; BY, F. O.; FISHER, B. **Docker In action**. 2. ed. New York: Manning Publications Co, 2019. ISBN 9781617294761.

PINHEIRO, J. M. **Percentil 95**. 2019. Disponível em: <https://www.ispblog.com.br/2019/10/16/percentil-95/>.

POHL, K.; RUPP, C. **Fundamentos da Engenharia de Requisitos**. 1. ed. Santa Bárbara: [s.n.], 2012. 189 p. ISBN 8571930996.

QUALIDADEBR. **FURPS+ | QualidadeBR**. 2008. Disponível em: <https://qualidadebr.wordpress.com/2008/07/10/furps/>.

RABL, T.; SADOGHI, M.; JACOBSEN, H. A.; GÓMEZ-VILLAMOR, S.; MUNTÉS-MULERO, V.; MANKOVSKII, S. Solving big data challenges for enterprise application performance management. **Proceedings of the VLDB Endowment**, v. 5, n. 12, p. 1724–1735, 2012. ISSN 21508097.

Red Hat. **O que é Docker?** 2018. Disponível em: <https://www.redhat.com/pt-br/topics/containers/what-is-docker>.

Red Hat. **O que são pipelines de CI/CD?** 2019. Disponível em: <https://www.redhat.com/pt-br/topics/devops/what-cicd-pipeline>.

REDIS. **Redis**. 2022. Disponível em: <https://redis.io/>.

ROCHA, A. **APM | O que é Application Performance Management e como pode ser útil!** 2020. Disponível em: <https://www.opservices.com.br/apm/>.

RUBIO, J. M.; PALMA, W.; RODRIGUEZ, N.; SOTO, R.; CRAWFORD, B.; PAREDES, F.; CABRERA, G. Solving the balanced academic curriculum problem using the ACO metaheuristic. **Mathematical Problems in Engineering**, Hindawi Publishing Corporation, v. 2013, 2013. ISSN 1024123X. Disponível em: <http://dx.doi.org/10.1155/2013/793671>.

RUBIO, J. M.; VIDAL-SILVA, C. L.; CABRERA, G. Seeking an optimal solution for a balanced academic curriculum by metaheuristics and functional programming. **Informacion Tecnologica**, v. 31, n. 6, p. 87–94, 2021. ISSN 07180764.

RUBIO, J. M.; VIDAL-SILVA, C. L.; SOTO, R.; MADARIAGA, E.; JOHNSON, F.; CARTER, L. Applying FireFly Algorithm to solve the problem of balancing curricula. **International Journal of Advanced Computer Science and Applications**, v. 10, n. 1, p. 68–75, 2019. ISSN 21565570.

SANTIAGO, C.; VERAS, N.; ARAGÃO, A. de; CARVALHO, D.; AMARAL, L. Desenvolvimento de sistemas Web orientado a reuso com Python, Django e Bootstrap. **Minicursos da ERCEMAPI 2020**, p. 97–120, 2020.

SCHAUS, P.; DEVILLE, Y.; DUPONT, P. Bound-consistent deviation constraint. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 4741 LNCS, p. 620–634, 2007. ISSN 16113349.

SELVARAJ, K.; GUPTA, R. K. **Mastering Elastic Stack : get the most out of the Elastic Stack for various complex analytics using this comprehensive and practical guide**. 1. ed. Birmingham: Packt Publishing LTD., 2017. 494 p. ISBN 9781786460011.

SHARMA, V. **Beginning Elastic Stack**. [S.l.: s.n.], 2016. ISBN 9781484216934.

SOLIQUE, I. d. S. ESTUDO COMPARATIVO DAS TECNOLOGIAS DE DESENVOLVIMENTO DE SISTEMAS WEB: POLYMER, DJANGO E METEOR. 2018.

SOUZA, H. V. de; PEREIRA, R. B. d. O. ANÁLISE DE DESEMPENHO DE BANCO DE DADOS: POSTGRESQL PADRÃO E UM CLUSTER UTILIZANDO O POSTGRES-BDR. **Profiscientia**, v. 0, n. 14, p. 88–108, dec 2020. ISSN 1806-0285. Disponível em: <http://profiscientia.ifmt.edu.br/profiscientia/index.php/profiscientia/article/view/268>.

SRIVASTAVA, A.; MILLER, D. **Elasticsearch 7 Quick Start Guide**. 1. ed. Birmingham: Packt Publishing LTD., 2019. v. 1999. 316 p. ISBN 9781789803327.

STEER, D. **CI/CD Tools Primer | GitLab**. 2022. Disponível em: <https://about.gitlab.com/handbook/marketing/strategic-marketing/competitive/cicd/>.

THOMPSON-ARJONA, W. G. Curricular optimization: Solving for the optimal student success pathway. **Theses and Dissertations–Electrical and Computer Engineering**, n. 139, p. 48, 2019.

TRUCCO, C. **Docker Compose: O que é? Para que serve? O que come?** 2018. Disponível em: <https://imasters.com.br/banco-de-dados/docker-compose-o-que-e-para-que-serve-o-que-come>.

ÜNAL, Y. Z.; UYSAL, Ö. A new mixed integer programming model for curriculum balancing: Application to a Turkish university. **European Journal of Operational Research**, North-Holland, v. 238, n. 1, p. 339–347, oct 2014. ISSN 03772217.

VASCONCELOS, L. A. **O que é um Servidor Web e como funciona?** 2021. Disponível em: <https://www.hostgator.com.br/blog/o-que-e-um-servidor-web-e-como-funciona/>.

VENTURA, P. **O que é UML - o que é, para que serve, quando usar, e muito mais!** 2021. Disponível em: <https://www.ateomomento.com.br/diagramas-uml/>.

VITALINO, J. F. N. **Descomplicando o Docker - Jeferson Fernando Noronha Vital**. Rio de Janeiro: Brasport, 2016. 422 p.

APÊNDICE A – DESCRIÇÃO DOS CASOS DE USO

1. **Manter instituições**
    - **Ator Primário:** Administrador;
    - **Interessados e Interesses:** O administrador deseja cadastrar, editar ou remover instituições de ensino no sistema.
    - **Pré-condições:** O administrador deve estar previamente cadastrado e logado no sistema para efetuar cadastro, edição ou remoção de instituições de ensino.
    - **Pós-condições:**
        a) Os dados da nova instituição de ensino armazenados no sistema (em caso de cadastro);
        b) Os dados de alguma instituição de ensino atualizados no sistema (em caso de edição);
        c) Os dados de alguma instituição de ensino removidos do sistema (em caso de exclusão);
    - **Cenário de Sucesso Principal:**
        a) Administrador abre o menu ”Universidades”;
        b) O sistema apresentará a listagem de instituições de ensino;
        c) Se o administrador selecionar ”Cadastrar”será apresentado um formulário de cadastro;
        d) Se o administrador seleciona ”Editar”será apresentado um formulário com os dados disponíveis para edição;
        e) Se o administrador seleciona ”Excluir”será apresentado um modal para confirmar a exclusão;
        f) O administrador salva os dados ou confirma a exclusão;
        g) O sistema exibe a mensagem ”Instituição cadastrada com sucesso”(em caso de cadastro) e o caso de uso se encerra;
        h) O sistema exibe a mensagem ”Dados atualizados com sucesso”(em caso de edição) e o caso de uso se encerra;
        i) O sistema exibe a mensagem ”Instituição (nome da instituição) excluída com sucesso”(em caso de exclusão) e o caso de uso se encerra;
    - **Fluxo de Exceção:** O caso de uso é interrompido se o administrador optar pelo cancelamento do cadastro, edição, exclusão ou fazer logout no sistema.

2. **Manter conta**
    - **Ator Primário:** Usuários;
    - **Interessados e Interesses:** O usuário deseja cadastrar, editar ou remover sua conta de acesso ao sistema.
    - **Pré-condições:** Deve haver instituições de ensino cadastradas no sistema.
    - **Pós-condições:**

a) Os dados de um usuário armazenados no sistema (em caso de cadastro);
b) Os dados de um usuário atualizados no sistema (em caso de edição);
c) Os dados de um usuário removidos do sistema (em caso de exclusão);

- **Cenário de Sucesso Principal:**
  a) O usuário acessa o menu ”Registrar”;
  b) O sistema apresentará um formulário de cadastro;
  c) O usuário preenche seus dados pessoais e escolhe o tipo de usuário (Aluno ou Gestor);
  d) O usuário seleciona uma instituição de ensino para vincular seu cadastro;
  e) O usuário preenche os campos de senha e confirmação de senha;
  f) O sistema redireciona o usuário para a tela de login e exibe a mensagem ”Usuário cadastrado com sucesso”(em caso de cadastro) e o caso de uso se encerra;
  g) O sistema exibe a mensagem ”Dados atualizados com sucesso”(em caso de edição) e o caso de uso se encerra;
  h) Em caso de exclusão de conta, o sistema redireciona o usuário para a tela de login;
- **Fluxo de Exceção:** O caso de uso é interrompido se o usuário optar pelo cancelamento do cadastro, edição, exclusão ou fazer logout no sistema (em caso de conta existente).

3. **Manter curso**
   - **Ator Primário:** Gestor;
   - **Interessados e Interesses:** O gestor deseja cadastrar, editar ou remover um curso no sistema.
   - **Pré-condições:** O gestor deve estar logado no sistema.
   - **Pós-condições:**
     a) Os dados de um curso armazenados no sistema (em caso de cadastro);
     b) Os dados de um curso atualizados no sistema (em caso de edição);
     c) Os dados de um curso removidos do sistema (em caso de exclusão);
   - **Cenário de Sucesso Principal:**
     a) O gestor acessa o menu ”Cursos”;
     b) O sistema apresentará a listagem de cursos;
     c) Se o gestor clicar no botão ”Novo”será apresentado um formulário de cadastro;
     d) Se o gestor seleciona ”Editar”será apresentado um formulário com os dados disponíveis para edição;
     e) Se o gestor seleciona ”Excluir”será apresentado um modal para confirmar a exclusão;
     f) Se o gestor clicar no nome do curso, o sistema irá redirecionar para a listagem de disciplinas do curso.

g) O gestor salva os dados ou confirma a exclusão;
h) O sistema exibe a mensagem ”Curso cadastrado com sucesso”(em caso de cadastro) e o caso de uso se encerra;
i) O sistema exibe a mensagem ”Curso atualizado com sucesso”(em caso de edição) e o caso de uso se encerra;
j) O sistema exibe a mensagem ”Curso (nome do curso) excluído com sucesso”(em caso de exclusão) e o caso de uso se encerra;

- **Fluxo de Exceção:** O caso de uso é interrompido se o usuário optar pelo cancelamento do cadastro, edição, exclusão ou fazer logout no sistema (em caso de conta existente).

4. **Manter disciplinas**
   - **Ator Primário:** Gestor;
   - **Interessados e Interesses:** O gestor deseja cadastrar, editar ou remover um curso no sistema.
   - **Pré-condições:** O gestor deve estar logado no sistema e o curso da disciplina deve ser previamente cadastrado (caso não esteja).
   - **Pós-condições:**
     a) Os dados de uma disciplina armazenados no sistema (em caso de cadastro);
     b) Os dados de uma disciplina atualizados no sistema (em caso de edição);
     c) Os dados de uma disciplina removidos do sistema (em caso de exclusão);
   - **Cenário de Sucesso Principal:**
     a) O gestor clica no nome do curso para qual deseja cadastrar a disciplina;
     b) O sistema apresentará a listagem de disciplinas do curso selecionado;
     c) Se o gestor clicar no botão ”Novo”será apresentado um formulário de cadastro;
     d) Se o gestor seleciona ”Editar”será apresentado um formulário com os dados disponíveis para edição;
     e) Se o gestor seleciona ”Excluir”será apresentado um modal para confirmar a exclusão;
     f) Se o gestor clicar no botão ”Vínculos”, o sistema irá redirecionar para a tela de vinculação com outras disciplinas.
     g) O gestor salva os dados ou confirma a exclusão;
     h) O sistema exibe a mensagem ”Curso cadastrado com sucesso”(em caso de cadastro) e o caso de uso se encerra;
     i) O sistema exibe a mensagem ”Curso atualizado com sucesso”(em caso de edição) e o caso de uso se encerra;
     j) O sistema exibe a mensagem ”Curso (nome do curso) excluído com sucesso”(em caso de exclusão) e o caso de uso se encerra;

- **Fluxo de Exceção:** O caso de uso é interrompido se o gestor optar pelo cancelamento do cadastro, edição, exclusão ou fazer logout no sistema.

5. **Manter vínculo entre disciplinas**
   - **Ator Primário:** Gestor;
   - **Interessados e Interesses:** O gestor deseja cadastrar, editar ou remover um vínculo entre disciplinas.
   - **Pré-condições:** O gestor deve estar logado no sistema e a disciplina deve ser previamente cadastrado (caso não esteja).
   - **Pós-condições:**
     a) Os dados de uma disciplina armazenados no sistema (em caso de cadastro);
     b) Os dados de uma disciplina atualizados no sistema (em caso de edição);
     c) Os dados de uma disciplina removidos do sistema (em caso de exclusão);
   - **Cenário de Sucesso Principal:**
     a) O gestor clica no nome do curso para qual deseja cadastrar a disciplina;
     b) O sistema apresentará a listagem de disciplinas do curso selecionado;
     c) Se o gestor clicar no botão ”Novo”será apresentado um formulário de cadastro;
     d) Se o gestor seleciona ”Editar”será apresentado um formulário com os dados disponíveis para edição;
     e) Se o gestor seleciona ”Excluir”será apresentado um modal para confirmar a exclusão;
     f) Se o gestor clicar no botão ”Vínculos”, o sistema irá redirecionar para a tela de vinculação com outras disciplinas.
     g) O gestor salva os dados ou confirma a exclusão;
   - **Fluxo de Exceção:** O caso de uso é interrompido se o gestor optar pelo cancelamento do cadastro, edição, exclusão ou fazer logout no sistema.

6. **Vincular a um curso**
   - **Ator Primário:** Aluno;
   - **Interessados e Interesses:** O aluno deseja se vincular a um curso.
   - **Pré-condições:** O aluno deve estar logado no sistema e deve haver cursos cadastrados.
   - **Pós-condições:**
     a) As disciplinas do curso deverão ser listadas para que seja possível a realização do balanceamento de currículo;
     b) O menu ”Cursos”deve exibir as informações do curso em que o aluno se vinculou;
   - **Cenário de Sucesso Principal:**
     a) O aluno clica no botão ”Vincular”em um curso disponível na lista de cursos;

b) O sistema irá redirecionar o aluno para o menu ”Minha Grade”;
c) Se o aluno acessar o menu ”Cursos”e clicar no botão ”Desvincular”, o menu ”Cursos”irá exibir novamente a listagem de cursos disponíveis;
d) Se o aluno se desvinculou de um curso, ao clicar no menu ”Minha Grade”, o sistema exibirá uma tabela vazia e a mensagem ”Não vinculado a um curso”;

- **Fluxo de Exceção:** sem fluxo de exceção.

7. **Selecionar disciplinas já cursadas**
   - **Ator Primário:** Aluno;
   - **Interessados e Interesses:** O aluno deseja selecionar as disciplinas que já cursou.
   - **Pré-condições:** O aluno deve estar logado no sistema e deve estar vinculado a um curso.
   - **Pós-condições:**
     a) As disciplinas marcadas como cursadas pelo aluno, deveram ser salvas no banco de dados com a finalidade de se manter o histórico de disciplinas concluídas; vinculou;
   - **Cenário de Sucesso Principal:**
     a) O aluno marca a opção ”Cursada”na tabela com a listagem de disciplinas no menu ”Minha Grade”;
   - **Fluxo de Exceção:** sem fluxo de exceção.

8. **Gerar grade curricular**
   - **Ator Primário:** Aluno;
   - **Interessados e Interesses:** O aluno deseja gerar uma grade curricular balanceada.
   - **Pré-condições:** O aluno deve estar logado no sistema e deve estar vinculado a um curso.
   - **Pós-condições:**
     a) O sistema exibirá um arquivo PDF com a grade balanceada;
   - **Cenário de Sucesso Principal:**
     a) O aluno acessa o menu ”Minha Grade”;
     b) O aluno tem a opção de marcar ou desmarcar disciplinas já concluídas;
     c) O aluno informa a quantidade de períodos em que se deseja realizar o balanceamento;
     d) O aluno clica no botão ”Gerar Grade”.
     e) O sistema abre uma nova aba no navegador com o arquivo PDF com a grade balanceada gerada.
   - **Fluxo de Exceção:** sem fluxo de exceção.

9. **Exportar grade curricular gerada**

- **Ator Primário:** Aluno;
- **Interessados e Interesses:** O aluno deseja salvar o arquivo PDF com a grade balanceada.
- **Pré-condições:** O aluno deve estar logado no sistema e deve realizar o processo de balanceamento previamente.
- **Pós-condições:**
  a) O sistema exibirá um arquivo PDF com a grade balanceada;
- **Cenário de Sucesso Principal:**
  a) O aluno acessa o menu ”Minha Grade”;
  b) O aluno tem a opção de marcar ou desmarcar disciplinas já concluídas;
  c) O aluno informa a quantidade de períodos em que se deseja realizar o balanceamento;
  d) O aluno clica no botão ”Gerar Grade”.
  e) O sistema abre uma nova aba no navegador com o arquivo PDF com a grade balanceada gerada.
- **Fluxo de Exceção:** sem fluxo de exceção.


UFVJM