INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
CÂMPUS PELOTAS - VISCONDE DA GRAÇA
CURSO TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS

AVALIAÇÃO DA QUALIDADE DE CÓDIGO GERADO POR ALGORITMOS GENÉTICOS: UM ESTUDO COM O SISTEMA ACE

Vinícius Armendaris Leal

Pelotas, 2025

Vinícius Armendaris Leal

Avaliação da Qualidade de Código Gerado por Algoritmos Genéticos: Um Estudo com o Sistema ACE

Trabalho de Conclusão de Curso apresentado como requisito na disciplina Metodologia de Pesquisa II do curso Técnico em Desenvolvimento de Sistemas, do Instituto Federal Sul-rio-grandense Campus Pelotas - Visconde da Graça.

Orientadora: Profa. Dra. Maria Isabel Giusti Moreira

Pelotas, 2025

RESUMO

Este trabalho aborda o uso de um sistema capaz de avaliar a qualidade de código gerado automaticamente por algoritmos genéticos, utilizando uma ferramenta de estudo que auxilia os usuários a terem um contato mais direto com esses algoritmos por meio da interação com o sistema Aleatory Coding Engine (ACE). A finalidade é ampliar a compreensão dos usuários acerca do funcionamento dos algoritmos genéticos e do seu potencial. Durante a pesquisa, foram identificadas duas referências conceituais fundamentais: a primeira é “Adaptation in Natural and Artificial Systems”, de John Holland, obra que estabeleceu, na década de 1970, as bases teóricas dos Algoritmos Genéticos (AGs), e a segunda é “Genetic programming”, de John Koza, que introduz a Programação Genética, paradigma no qual programas de computador evoluem por meio de autoavaliação, do cruzamento e da mutação, em um processo análogo à seleção natural. Além disso, apresenta o Teorema do Macaco Infinito, que serviu como inspiração adicional ao ilustrar, de forma probabilística, que a geração autônoma de textos ou programas, embora improvável, não é impossível ao longo de tempo e tentativas suficientes. A partir dessas bases teóricas, o objetivo central do trabalho consiste em auxiliar na compreensão do tema, oferecendo uma ferramenta capaz de identificar e validar métricas eficazes para avaliação da qualidade de código. Entre os indicadores analisados, destacam-se a eficiência de execução e o índice de manutenibilidade, buscando determinar quais métricas realmente contribuem para a evolução de soluções robustas e funcionais. A metodologia adotada envolveu a especificação de requisitos baseada em uma pesquisa com estudantes sobre o tema e prototipação contínua. Em seguida, desenvolveu-se o sistema ACE em linguagem Python, utilizando o framework Flask e o banco de dados SQLite. O ACE implementa operadores genéticos clássicos, como cruzamento e mutação, aplicados a blocos de código que são avaliados iterativamente, recebem pontuação conforme critérios pré-definidos e são armazenados em histórico evolutivo, permitindo comparações entre gerações. Os resultados obtidos evidenciam que houve expansão da compreensão dos usuários, que passaram a ter um repertório maior sobre algoritmos genéticos ao interagir com uma ferramenta capaz de gerar, validar e classificar código de forma autônoma. Considera-se, portanto, que este trabalho contribui tanto para o campo dos algoritmos genéticos quanto para o da avaliação automatizada de algoritmos, além de reforçar o potencial aplicado à geração de código. Assim, consolida-se um elo entre inspirações teóricas clássicas e práticas contemporâneas, oferecendo subsídios para futuros estudos sobre sistemas capazes de produzir e aprimorar código de maneira autônoma.

SUMÁRIO

1. INTRODUÇÃO..... 5
2. TEMA..... 7
3. MOTIVAÇÕES..... 8
4. OBJETIVOS..... 10
4.1. Objetivo Geral..... 10
4.2. Objetivos Específicos..... 10
5. ESPECIFICAÇÃO DE REQUISITOS..... 11
5.1. Requisitos Funcionais..... 23
5.2. Requisitos Não Funcionais..... 24
6. MODELAGEM..... 26
6.1. Modelo De Casos De Uso..... 26
6.2. Modelagem Conceitual Do Banco De Dados..... 26
7. TECNOLOGIAS UTILIZADAS..... 28
8. DESCRIÇÃO DO SISTEMA..... 32
8.1. Tela Inicial..... 32
8.2 Tela Como esse sistema funciona..... 33
8.3 Tela Veja a mágica acontecer..... 35
8.4 Tela Os melhores Resultados..... 42
9. CONSIDERAÇÕES FINAIS..... 45
10. REFERÊNCIAS..... 46
11. APÊNDICE..... 47

1. INTRODUÇÃO

O avanço das técnicas de automação e da evolução computacional tem ampliado significativamente as possibilidades de geração automática de software, impulsionando pesquisas que buscam integrar processos evolutivos aos métodos tradicionais de desenvolvimento. Nesse contexto, algoritmos inspirados na seleção natural têm se destacado por sua capacidade de explorar soluções complexas e adaptá-las ao longo do tempo.

A compreensão sobre algoritmos genéticos e a avaliação da qualidade de código são temas que se complementam no desenvolvimento de software, pois ambos contribuem para o aprimoramento da performance dos modelos evolutivos e para a criação de soluções de otimização aplicáveis, inclusive, ao treinamento de sistemas de Inteligência Artificial. Diferentes abordagens tentam definir critérios objetivos para medir se um código é eficiente, legível e funcional, mas muitas dessas métricas são subjetivas ou limitadas a contextos específicos. O projeto ACE: Aleatory Coding Engine busca auxiliar na compreensão dessa convergência: como entender algoritmos genéticos e como determinar quais métricas são realmente eficazes para avaliar e aprimorar código gerado por esses algoritmos.

O uso de algoritmos genéticos na programação pode ser uma alternativa para testar diferentes soluções e adaptá-las progressivamente, mas a definição de bons critérios de seleção é essencial para que o código evolua de forma significativa. Assim, a questão central deste estudo consiste em demonstrar o funcionamento dos algoritmos genéticos e identificar quais métricas realmente influenciam a qualidade do código gerado, distinguindo-as daquelas superficiais ou pouco relevantes no contexto da evolução computacional. Busca-se, ainda, evidenciar aos usuários o impacto que essas métricas exercem no desempenho e na evolução das soluções produzidas.

Para isso, este trabalho explora diferentes formas de avaliação, analisando o impacto de métricas como índice de manutenibilidade, eficiência de execução e complexidade na melhoria real do código ao longo de múltiplas iterações. O objetivo não é apenas quantificar a qualidade do código gerado, mas sim entender quais fatores contribuem efetivamente para a evolução de soluções mais robustas.

A pesquisa foi conduzida por meio de experimentos controlados, nos quais

diferentes usuários interagiram com o sistema ACE. A partir dessas interações, as métricas de avaliação foram aplicadas com o objetivo de identificar eventuais barreiras que dificultam a compreensão dos usuários, permitindo orientar melhor sobre os conceitos envolvidos nos algoritmos genéticos. Os resultados obtidos oferecem contribuições relevantes para compreender a relação entre avaliação de código e otimização evolutiva, além de apoiar a definição de diretrizes mais eficazes para a análise automática da qualidade de software..

A relevância deste estudo reside na necessidade crescente de compreender como métricas de avaliação influenciam processos evolutivos automatizados, especialmente em um cenário onde a geração autônoma de código se torna cada vez mais presente em pesquisas de IA e engenharia de software. Ao analisar os efeitos reais dessas métricas, o trabalho busca preencher uma lacuna existente entre a teoria evolutiva e a prática da avaliação automática de software.

2. TEMA

Estudo sobre a compreensão da qualidade de código e dos Algoritmos Genéticos, com foco na análise e aplicação de métricas de eficiência e computação evolutiva antes e depois de interagir com o sistema ACE – *Aleatory Coding Engine*.

3. MOTIVAÇÕES

O Projeto ACE (*Aleatory Code Engine*) surge de uma curiosidade prática: testar os limites do uso de algoritmos genéticos na geração de estruturas coerentes e funcionais, mesmo dentro de contextos tão formais quanto a programação. Inspirado pelo Teorema do Macaco Infinito, que sugere que, dado tempo suficiente, um número infinito de macacos digitando aleatoriamente produziria a obra completa de Shakespeare, o projeto propõe um modelo onde, por meio de mutações e seleções iterativas, soluções programáticas válidas e eficientes são geradas automaticamente.

Esse processo de "evolução" do código segue a lógica dos algoritmos genéticos, que emulam o processo natural de seleção e adaptação. Inicialmente, um bloco de código simples e rudimentar é criado e, a partir daí, passa por diversas gerações, nas quais mudanças no código são feitas, por exemplo, através de mutações ou cruzamento de diferentes "códigos pais". Cada nova geração é avaliada segundo critérios de qualidade definidos, como eficiência, legibilidade e correção sintática. As soluções mais bem adaptadas, ou seja, aquelas que apresentam melhores métricas, são selecionadas para "reproduzir" a próxima geração de código.

No início do projeto, as soluções geradas podem ser rudimentares e não necessariamente funcionais, mas, ao longo das gerações, o código tende a se tornar mais otimizado e complexo. Essa abordagem permite que o sistema encontre soluções não apenas para problemas simples, mas também para desafios mais complicados e para a implementação de soluções inovadoras, às vezes utilizando abordagens fora do convencional.

Ao focar na otimização do código e no aprimoramento contínuo das soluções geradas, o ACE também busca investigar quais métricas são mais eficazes para avaliar o código produzido por algoritmos genéticos. Um dos grandes desafios desse tipo de abordagem é, justamente, definir os critérios corretos para essa avaliação, visto que a qualidade do código não se limita apenas a sua sintaxe, mas também envolve aspectos como eficiência de execução, clareza e adequação às necessidades do problema em questão.

A motivação mudou durante o desenvolvimento, a partir da compreensão de que debater sobre o assunto se fazia complicado pela baixa compreensão sobre a temática. Ao entender isso, manteve-se o interesse pessoal em juntar teoria evolutiva com práticas computacionais, porém agora em conjunto com a proposta de elucidar os temas que compõem todo o contexto inicial desse projeto. Em vez de usar algoritmos genéticos para resolver problemas clássicos, como otimização de rotas ou parametrização de modelos, o ACE irá auxiliar na compreensão desse processo em algo mais abstrato: a construção de código por tentativa e erro, orientada por seleção e mutação.

Esse projeto é um laboratório. Um campo de testes para avaliar como essa solução auxilia ao entendimento sobre soluções "evoluídas" e de como elas se comportam quando forçadas a cumprir regras sintáticas e objetivos definidos. Esse estudo é motivado pelo interesse em compreender como sistemas aparentemente aleatórios podem, sob a seleção, gerar complexidade organizada, sendo o código uma arena ideal para essa investigação por seu caráter formal e rigoroso.

Além de seu papel como ferramenta de avaliação, o ACE serve como experimento para observar e aprender com os padrões emergentes, os erros recorrentes e os limites da evolução cega em sistemas computacionais. É uma forma de aplicar teoria na prática, sem a pretensão de substituir nada, mas com a intenção de aprender ao máximo com o processo.

4. OBJETIVOS

4.1. Objetivo Geral

Desenvolver um sistema que auxilie os usuários na compreensão de algoritmos genéticos, por meio da interação com uma ferramenta capaz de identificar e validar métricas eficazes para avaliar e aprimorar um código gerado por esses algoritmos, determinando quais indicadores contribuem para melhorias consistentes na qualidade do código e esclarecendo os principais conceitos envolvidos no tema.

4.2. Objetivos Específicos

a) Analisar as principais métricas utilizadas na avaliação de códigos, como complexidade ciclomática, eficiência de execução, legibilidade e modularidade;
b) projetar e implementar o sistema ACE – *Aleatory Coding Engine*, empregando algoritmos genéticos para a geração automática de código;
c) aplicar diferentes métricas de avaliação ao código gerado pelo sistema ACE, ao longo de múltiplas iterações evolutivas;
d) comparar o desempenho dessas métricas, identificando quais apresentam maior correlação com melhorias concretas na qualidade do código;
e) validar os resultados obtidos por meio de testes controlados, evidenciando os impactos das métricas mais eficazes no processo de evolução do código.

5. ESPECIFICAÇÃO DE REQUISITOS

A especificação de requisitos é uma etapa fundamental no desenvolvimento de software, responsável por identificar, documentar e organizar as necessidades, expectativas e limitações relacionadas ao sistema a ser construído. Seu objetivo é traduzir as demandas dos usuários e dos objetivos do projeto em requisitos funcionais (o que o sistema deve fazer) e não funcionais (como o sistema deve se comportar), servindo como base para o planejamento, modelagem, implementação e validação da solução. Uma boa especificação garante alinhamento entre desenvolvedores e usuários, reduz ambiguidades e orienta a criação de um sistema mais eficiente, consistente e adequado ao seu propósito.

Para a realização da especificação de requisitos, conduziu-se uma pesquisa voltada ao usuário final, complementada por um processo de prototipação contínua ao longo do desenvolvimento do projeto. A primeira investigação, intitulada *“Pesquisa sobre conhecimentos e aplicações de algoritmos genéticos”* que se encontra no Apêndice A – Questionário da Pesquisa, que contou com a colaboração contou com a colaboração de cinco estudantes do curso Técnico em Desenvolvimento de Sistemas do IFSul Campus Pelotas - Visconde da Graça , teve como objetivo compreender o nível de familiaridade dos potenciais usuários com Algoritmos Genéticos (AGs) e levantar informações essenciais para orientar o desenho do sistema. Para isso, foi aplicado um formulário online destinado a identificar o perfil dos participantes, mapear dificuldades conceituais e avaliar o interesse em uma ferramenta visual e interativa como o ACE – *Aleatory Coding Engine*. A pesquisa também buscou coletar percepções sobre métricas, métodos de seleção e possíveis aplicações da plataforma, fornecendo subsídios importantes para a definição de funcionalidades e prioridades do projeto

O instrumento de coleta foi estruturado em três seções, totalizando 13 perguntas (entre questões objetivas, de múltipla escolha e perguntas abertas). A seguir, descreve-se cada etapa:

- **Seção 1 – Perfil do Participante:** teve como finalidade caracterizar o público respondente, permitindo compreender melhor o contexto formativo e o grau de familiaridade dos usuários com o tema investigado. Essa parte do formulário incluiu perguntas sobre a área de formação ou atuação principal dos participantes, possibilitando identificar se eram estudantes ou

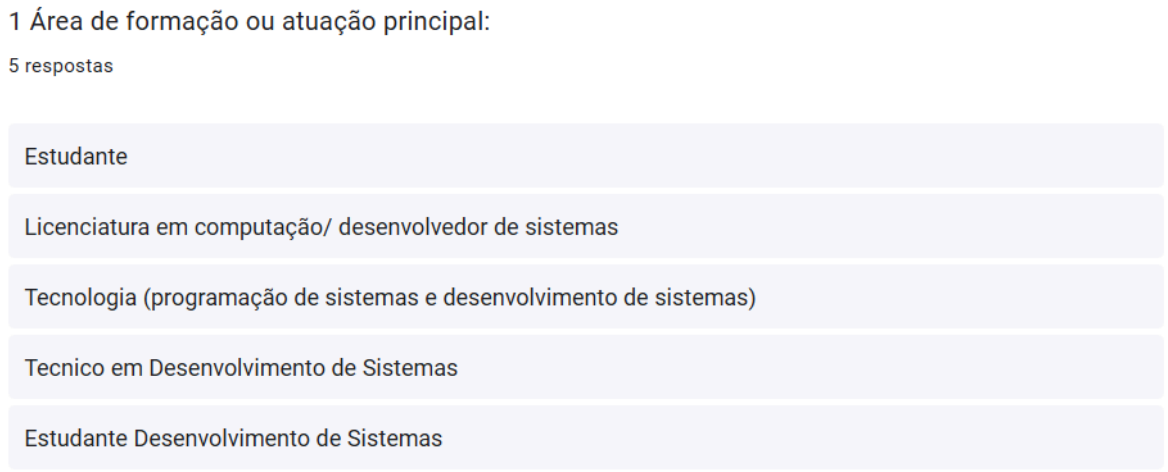
profissionais das áreas de computação e desenvolvimento de sistemas. Também buscou avaliar o nível de familiaridade com Algoritmos Genéticos, variando desde o desconhecimento total até a experiência prática avançada. Além disso, a seção investigou o uso prévio de ferramentas ou bibliotecas relacionadas a AGs, acompanhada de uma subquestão destinada à especificação dessas ferramentas, caso houvesse. Essa etapa forneceu informações essenciais para compreender o perfil do público-alvo e orientar o desenvolvimento do sistema ACE de acordo com suas necessidades e conhecimentos prévios.

- **Seção 2 – Entendimento e Aplicações:** teve como objetivo investigar o grau de compreensão dos usuários sobre Algoritmos Genéticos e sua capacidade de relacioná-los a problemas reais. Nessa etapa, o formulário buscou identificar o conhecimento dos participantes sobre áreas de aplicação dos AGs, solicitando também a citação de exemplos práticos de uso. Além disso, a seção incluiu uma pergunta destinada a compreender quais aspectos dos algoritmos são percebidos como mais difíceis, permitindo que os participantes selecionaram elementos como mutação, crossover, codificação, processos de seleção e interpretação dos resultados. Essa parte da pesquisa possibilitou mapear as principais dificuldades conceituais enfrentadas pelos usuários e identificar lacunas de entendimento relevantes para o desenvolvimento do sistema ACE.
- **Seção 3 – Interesse em uma Ferramenta Interativa:** teve como finalidade avaliar a receptividade dos participantes a uma plataforma como o ACE e identificar funcionalidades consideradas relevantes para o aprendizado. Essa etapa do formulário incluiu seis perguntas que investigaram a percepção dos usuários sobre a utilidade de uma plataforma visual ou interativa para auxiliar no estudo de Algoritmos Genéticos, bem como a avaliação de uma plataforma hipotética por meio de uma escala de 1 a 5. Também foram explorados os recursos que os participantes consideram desejáveis em uma ferramenta desse tipo, como a possibilidade de comparar métodos de seleção, exportar resultados e visualizar explicações passo a passo do algoritmo. Além disso, buscou-se compreender em quais contextos a plataforma seria utilizada — seja para fins acadêmicos, pesquisa ou uso pessoal — e registrar sugestões de aprimoramento, especialmente no que se refere à visualização,

apresentação e facilidade de uso.

Na Questão 1 da Seção 1, referente à área de formação ou atuação principal, obteve-se um total de cinco respostas (Figura 1). Todos os participantes declararam vínculo com a área de computação e desenvolvimento de sistemas, distribuindo-se entre estudantes de cursos de Desenvolvimento de Sistemas, Técnico em Desenvolvimento de Sistemas, curso tecnológico em programação de sistemas e desenvolvimento de sistemas, e licenciatura em computação com atuação como desenvolvedor de sistemas. Esse perfil indica que o público respondente possui formação ou interesse diretamente relacionado à programação e à engenharia de software, o que é coerente com a proposta do sistema ACE.

Figura 1. Área de formação ou atuação principal dos participantes da pesquisa sobre Algoritmos Genéticos.

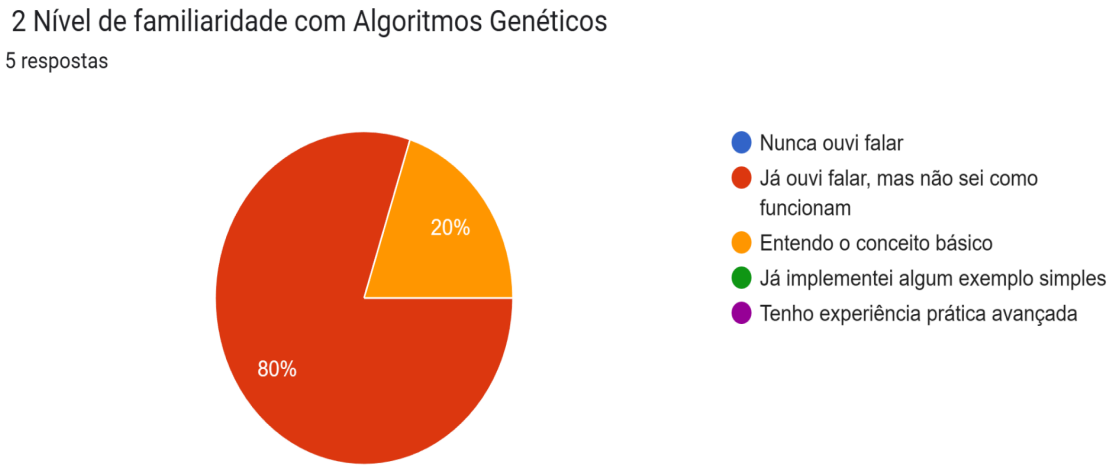


Fonte: Autor

Ainda na Seção 01, Questão 2, que investigou o nível de familiaridade com Algoritmos Genéticos, observou-se que a maior parte dos participantes ainda não tem conhecimento do tema (Figura 2). Dos cinco respondentes, 80% afirmaram que já ouviram falar sobre Algoritmos Genéticos, mas não sabem exatamente como funcionam, enquanto 20% indicaram compreender apenas o conceito básico. Nenhum participante declarou nunca ter ouvido falar sobre o assunto, tampouco relataram já ter implementado exemplos práticos ou possuir experiência avançada. Esses dados sugerem que o público possui algum contato inicial com a temática,

porém com lacunas significativas de compreensão conceitual e prática, o que reforça a pertinência de uma ferramenta como o ACE para apoio ao aprendizado.

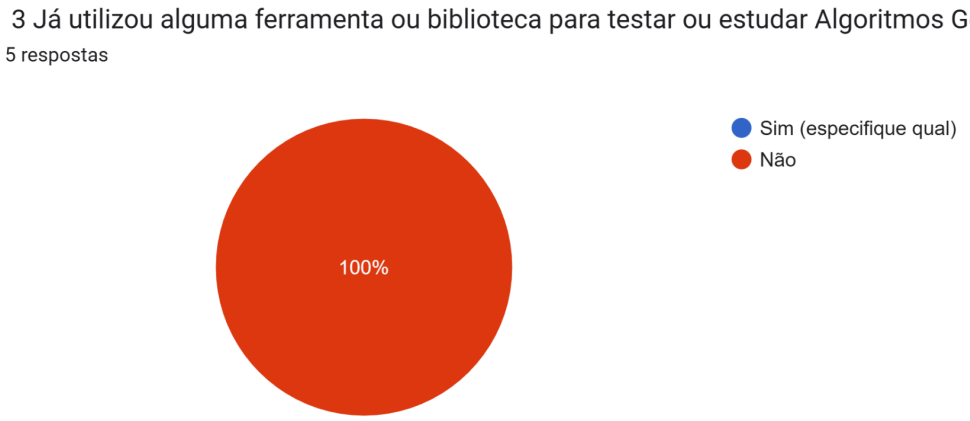
Figura 2. Nível de familiaridade dos participantes com Algoritmos Genéticos.



Fonte: Autor

Na Questão 3, que investigou se os participantes já haviam utilizado alguma ferramenta ou biblioteca para testar ou estudar Algoritmos Genéticos, verificou-se que 100% dos respondentes marcaram a opção “Não” (Figura 3). A subquestão 3.1, que solicitava a especificação da ferramenta em caso de resposta afirmativa, não recebeu nenhuma resposta. Esses dados indicam que, embora exista algum contato conceitual com o tema, o grupo não possui experiência prática com ferramentas específicas para AGs, reforçando a importância de uma plataforma como o ACE para apoiar experimentações e estudos aplicados.

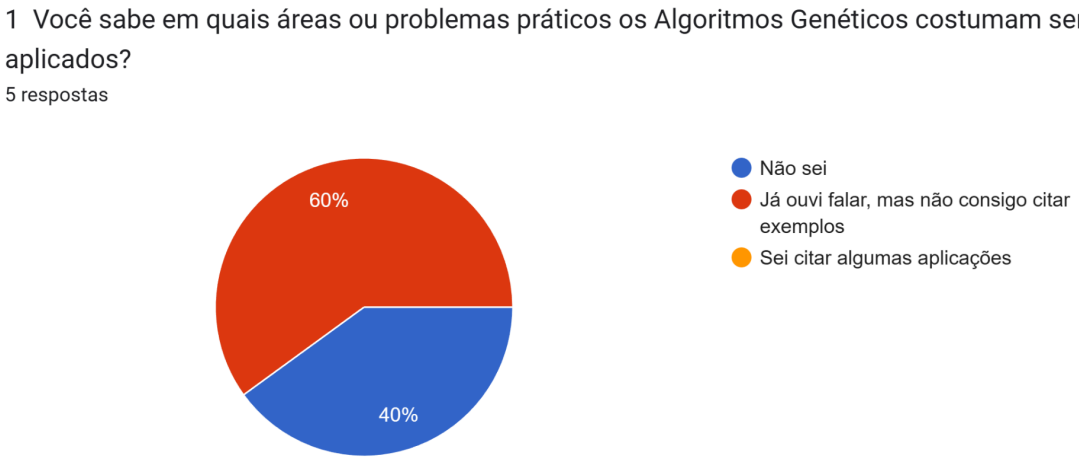
Figura 3. Uso prévio de ferramentas ou bibliotecas para estudo de Algoritmos Genéticos.



Fonte: Autor

Na questão referente ao conhecimento sobre áreas ou problemas práticos em que Algoritmos Genéticos (Figura 4) costumam ser aplicados, observou-se que nenhum participante demonstrou domínio concreto desse aspecto. Do total de respondentes, 40% declararam não saber em quais contextos os AGs são utilizados, enquanto 60% afirmaram já ter ouvido falar no tema, mas sem conseguir citar exemplos de aplicação. Não houve respostas indicando capacidade de mencionar casos específicos de uso. Esses resultados evidenciam que, embora exista algum contato inicial com o conceito, há uma lacuna importante no entendimento das aplicações práticas dos Algoritmos Genéticos, o que reforça o potencial do ACE como ferramenta de apoio à aprendizagem contextualizada.

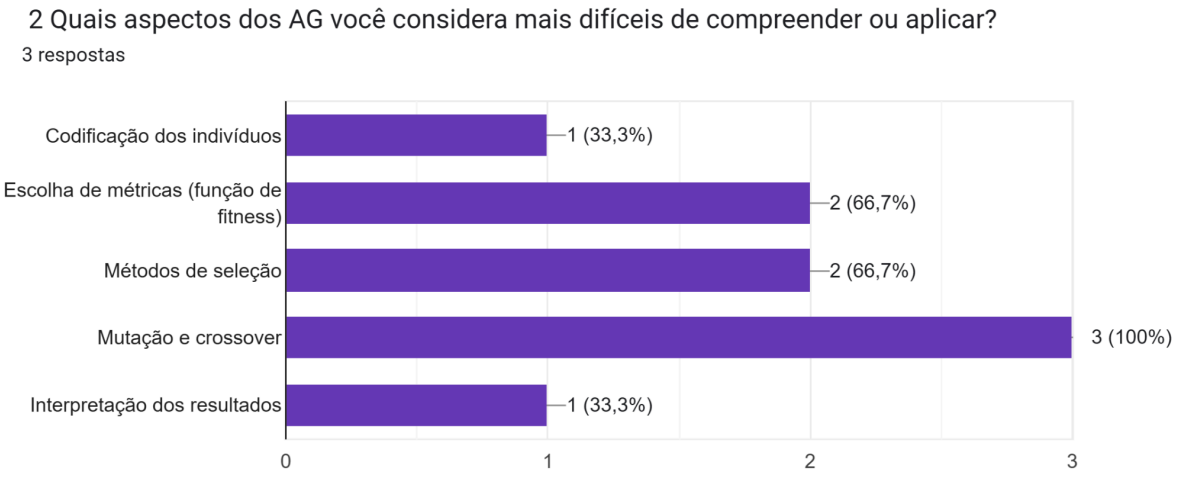
Figura 4. Conhecimento dos participantes sobre áreas de aplicação de Algoritmos Genéticos.



Fonte: Autor

Na questão que investigou quais aspectos dos Algoritmos Genéticos os participantes consideram mais difíceis de compreender ou aplicar, observou-se que “mutação e crossover” foram apontados por todos os respondentes, aparecendo como o ponto de maior dificuldade (Figura 5). Em seguida, destacaram-se a “escolha de métricas (função de fitness)” e os “métodos de seleção”, ambos mencionados pela maioria dos participantes. Já a “codificação dos indivíduos” e a “interpretação dos resultados” foram indicadas por uma parcela menor. Esses dados evidenciam que, além da implementação dos operadores genéticos, a definição da função de fitness e dos critérios de seleção constitui um dos principais desafios conceituais e práticos, reforçando a importância de que o ACE ofereça recursos explicativos e visuais voltados especificamente a esses elementos do algoritmo.

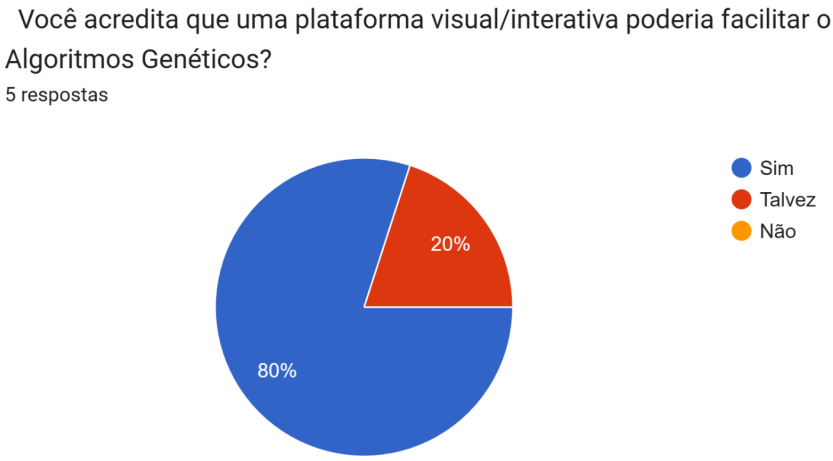
Figura 5 – Aspectos dos Algoritmos Genéticos considerados mais difíceis de compreender ou aplicar pelos participantes.



Fonte: Autor

Na questão que investigou se uma plataforma visual ou interativa poderia facilitar o aprendizado sobre Algoritmos Genéticos, observou-se uma aceitação bastante positiva da proposta (Figura 6). Dos cinco respondentes, 80% indicaram que acreditam que uma ferramenta desse tipo ajudaria no processo de aprendizagem (“Sim”), enquanto 20% responderam “Talvez”, sinalizando abertura à ideia, embora com alguma reserva. Não houve nenhuma resposta negativa. Esses resultados reforçam a relevância pedagógica de uma solução como o ACE, apontando que a visualização e a interação com o comportamento dos algoritmos são percebidas pelos usuários como estratégias potencialmente úteis para a compreensão do tema.

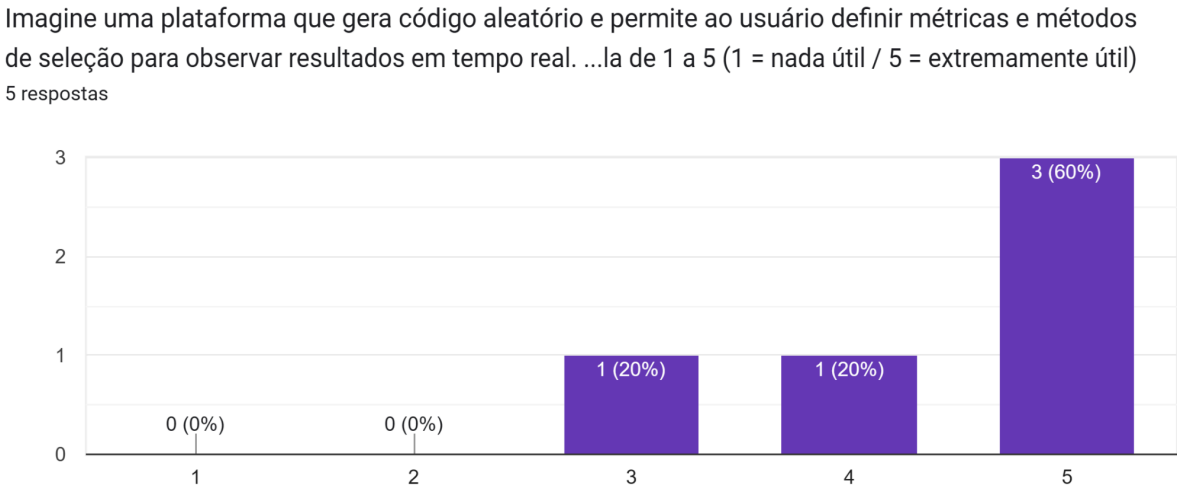
Figura 6. Percepção dos participantes sobre o potencial de uma plataforma visual/interativa para facilitar o aprendizado de Algoritmos Genéticos.



Fonte: Autor

Ao serem convidados a imaginar uma plataforma capaz de gerar código aleatório e permitir ao usuário definir métricas e métodos de seleção para observar os resultados em tempo real, os participantes avaliaram o quão útil essa ferramenta seria em uma escala de 1 a 5 (1 = nada útil; 5 = extremamente útil). Os dados indicam uma percepção predominantemente positiva: 60% atribuíram nota máxima (5) à utilidade da plataforma, enquanto 20% atribuíram nota 4 e 20% nota 3. Não houve registros nas notas 1 e 2. Esses resultados sugerem que a proposta de uma ferramenta como o ACE é vista como altamente promissora pelos respondentes, especialmente no que se refere ao apoio à experimentação e à visualização do comportamento dos Algoritmos Genéticos (Figura 7).

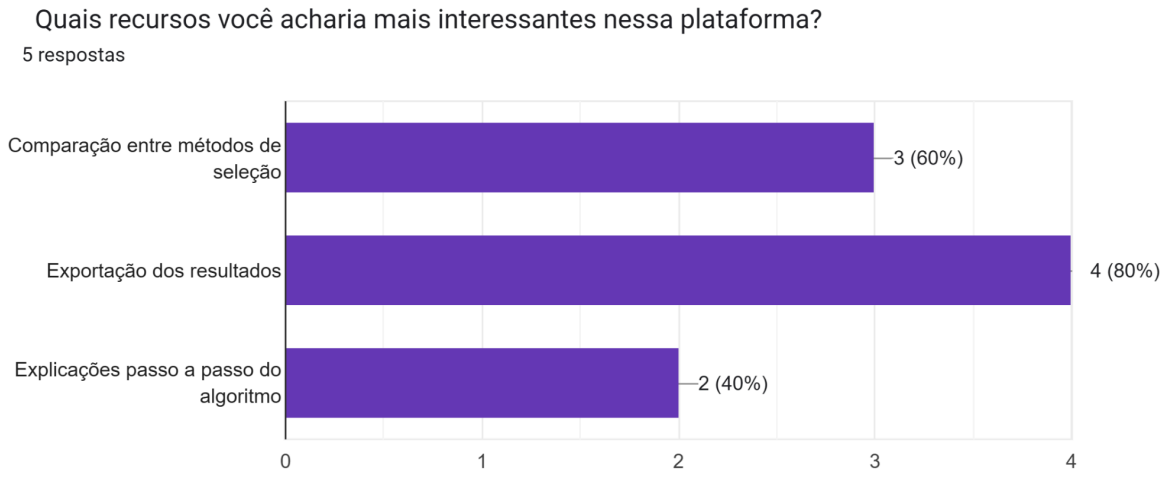
Figura 7. Avaliação da utilidade potencial de uma plataforma interativa para geração de código com definição de métricas e métodos de seleção (escala de 1 a 5).



Fonte: Autor

Quando questionados sobre quais recursos considerariam mais interessantes em uma plataforma voltada ao estudo de Algoritmos Genéticos, os participantes apontaram principalmente funcionalidades relacionadas à análise e ao registro dos resultados (Figura 8). A exportação dos resultados foi o recurso mais indicado, aparecendo em 80% das respostas, o que evidencia a importância de registrar, compartilhar ou reaproveitar os experimentos realizados. Em seguida, destacou-se a comparação entre métodos de seleção, mencionada por 60% dos respondentes, indicando o interesse em observar, de forma prática, como diferentes estratégias de seleção impactam o comportamento do algoritmo. As explicações passo a passo do algoritmo também foram valorizadas, aparecendo em 40% das respostas, reforçando a necessidade de recursos didáticos que tornem o funcionamento interno dos AGs mais transparente. Esses dados orientaram o projeto do ACE no sentido de priorizar mecanismos de visualização, comparação e documentação das execuções.

Figura 8. Recursos considerados mais interessantes pelos participantes em uma plataforma para estudo de Algoritmos Genéticos.



Fonte: Autor

Ao serem questionados sobre em quais contextos utilizariam uma ferramenta como o ACE, os participantes indicaram principalmente situações relacionadas ao ensino e à aprendizagem (Figura 9). A opção “Aprendizado (ensino acadêmico)” foi assinalada por 80% dos respondentes, revelando forte interesse em empregar a plataforma como recurso de apoio em disciplinas ou estudos formais. Além disso, 60% afirmaram que usariam a ferramenta por curiosidade pessoal, o que sugere um potencial também para auto estudo e exploração independente do tema. A opção “Pesquisa científica” foi mencionada por 20% dos participantes, apontando que, embora em menor proporção, há também interesse em usos mais avançados e investigativos. Esses dados reforçam o caráter predominantemente educacional da proposta do ACE, ao mesmo tempo em que indicam possibilidades de uso em contextos de pesquisa e aprendizado informal.

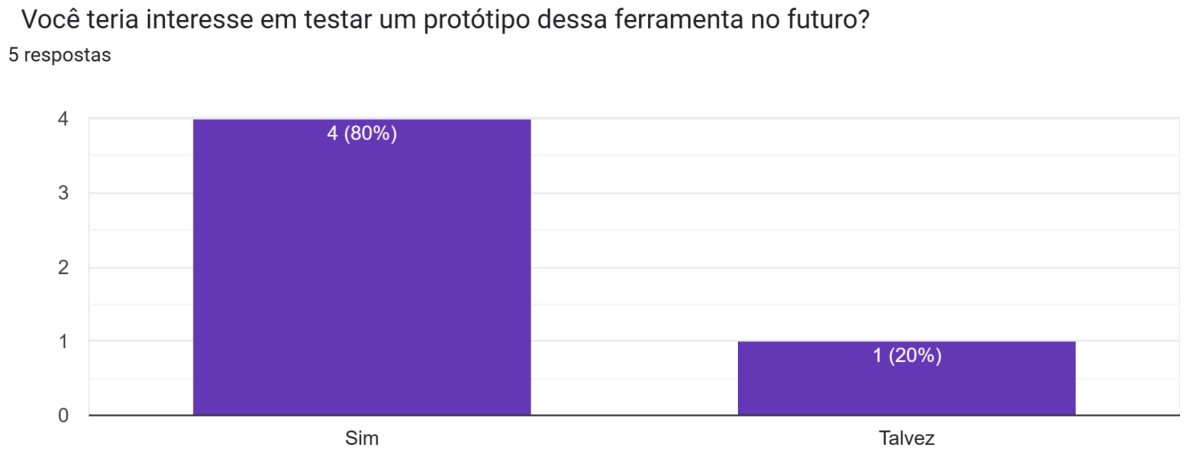
Figura 9. Contextos em que os participantes utilizariam uma ferramenta para estudo de Algoritmos Genéticos.

Fonte: Autor

Na questão aberta que perguntava *“O que tornaria essa plataforma mais atrativa ou útil para você?”*, foram registradas duas sugestões principais. A primeira destacou a importância de dispor de gráficos para melhor visualização, reforçando a necessidade de recursos visuais que permitam acompanhar a evolução dos resultados de forma clara. A segunda recomendação enfatizou que a plataforma deve ser simples e visualmente atrativa, possibilitando o uso mesmo por pessoas sem conhecimento técnico aprofundado. Essas respostas qualitativas indicam que, além das funcionalidades algorítmicas, aspectos de usabilidade e visualização de dados são considerados centrais para a aceitação e a efetividade de uma ferramenta como o ACE.

Na última questão da pesquisa, que indagou se os participantes teriam interesse em testar um protótipo da ferramenta em um momento futuro, observou-se uma predisposição bastante favorável à experimentação. Dos cinco respondentes, quatro (80%) indicaram “Sim”, enquanto um participante (20%) respondeu “Talvez” (Figura 10). Não houve respostas negativas. Esses dados reforçam não apenas a aceitação da proposta do ACE, mas também a disponibilidade do público-alvo em participar de etapas posteriores de validação, o que é relevante para ciclos iterativos de melhoria do sistema e para futuras pesquisas empíricas envolvendo o uso da ferramenta.

Figura 10. Interesse dos participantes em testar um protótipo da ferramenta no futuro.



Fonte: Autor

Com os dados dessa pesquisa, foi possível delinear o perfil do usuário-alvo do sistema ACE. Observou-se que se trata, predominantemente, de estudantes e profissionais da área de Tecnologia da Informação, com formação ou atuação em cursos de computação e desenvolvimento de sistemas. Os participantes demonstram possuir algum nível de compreensão sobre Algoritmos Genéticos, porém não aprofundada, o que evidencia lacunas conceituais importantes. Verificou-se também que não há experiência prévia com ferramentas ou bibliotecas específicas para aprender e testar conceitos relacionados a AGs, bem como não há conhecimento consolidado sobre aplicações práticas que fazem uso desses algoritmos.

Os resultados indicam ainda que os usuários percebem uma ampla variedade de conceitos difíceis de apreender e aplicar, especialmente no que diz respeito a operadores genéticos, escolha de métricas e métodos de seleção. Ao mesmo tempo, ficou clara a crença de que uma ferramenta visual e interativa pode facilitar a compreensão sobre Algoritmos Genéticos, sobretudo se permitir ao usuário experimentar diferentes métricas e métodos de seleção, o que foi avaliado como um aspecto de alto valor. Outro ponto relevante é o interesse em exportar os resultados gerados pela ferramenta, seja para registro, estudo ou compartilhamento. Constatou-se, por fim, que o principal contexto de uso desejado é o acadêmico e

que há demanda por uma interface simples, intuitiva e visualmente atrativa, capaz de ser utilizada inclusive por pessoas com pouco conhecimento técnico.

Diante desse conjunto de necessidades e expectativas, foram levantados os seguintes requisitos para o sistema ACE, de forma a alinhar o desenvolvimento da ferramenta ao perfil e às demandas dos usuários identificados na pesquisa.

5.1. Requisitos Funcionais

Os requisitos funcionais descrevem o conjunto de funções que o sistema deve realizar para atender às necessidades dos usuários e aos objetivos do projeto. Em outras palavras, tratam do comportamento esperado do software, especificando o que ele deve fazer diante de determinadas ações, entradas ou situações. Cada requisito funcional corresponde a uma funcionalidade observável. Os requisitos funcionais do Sistema ACE podem ser observados no Quadro 1.

Quadro 1. Requisitos Funcionais do Sistema ACE

REF	Caso de Uso	Descrição
REF01	Definir Parâmetros	O sistema deve permitir que o usuário defina e ajuste parâmetros para criação, modificação e comparação de código, incluindo, por exemplo, número de gerações, tamanho da população, métricas de avaliação e métodos de seleção.
REF02	Gerar Código Automático	O sistema deve gerar automaticamente blocos de código em linguagem Python, com base nos parâmetros definidos pelo usuário e/ou em algoritmos de otimização utilizados pelo ACE.
REF03	Selecionar Código Válido	O sistema deve validar o código gerado, identificando quais blocos são sintaticamente e semanticamente válidos, descartando aqueles que apresentarem erros.
REF04	Pontuar Código	O sistema deve calcular a pontuação dos blocos de código de acordo com métricas definidas (por exemplo, eficiência, complexidade, manutenção) e classificá-los com base nesses resultados.
REF05	Armazenar o código	O sistema deve armazenar os blocos de

		código gerados em seus respectivos grupos, gerações ou experimentos, permitindo sua recuperação posterior.
REF06	Comparar Código	O sistema deve comparar blocos de código entre si, possibilitando ao usuário analisar diferenças de pontuação, desempenho e outras métricas entre gerações ou versões.
REF07	Realizar Cruzamento de Códigos	O sistema deve realizar operações de cruzamento (crossover) entre dois blocos de código “pais”, gerando novos blocos “descendentes” que combinem características de ambos.
REF08	Aplicar Mutações no Código	O sistema deve aplicar mutações em blocos de código, alterando parâmetros ou estruturas específicas com o objetivo de explorar novas soluções, melhorar o desempenho ou corrigir falhas.
REF09	Exibir Resultados de Geração	O sistema deve apresentar os resultados das gerações, exibindo pontuações, classificações, grupos e demais informações relevantes de forma organizada.
REF010	Armazenar Histórico de Evolução	O sistema deve registrar e armazenar o histórico de evolução, mantendo versões anteriores dos blocos de código ao longo das gerações, de modo que o usuário possa acompanhar o processo evolutivo.

Fonte: Autor

5.2. Requisitos Não Funcionais

Os requisitos não funcionais correspondem às características de qualidade e às restrições que um sistema de software deve atender, não em relação ao que ele faz, mas a como ele deve se comportar ao executar suas funções. Diferentemente dos requisitos funcionais, que descrevem operações específicas (como gerar, armazenar ou comparar códigos), os requisitos não funcionais tratam de aspectos como desempenho, usabilidade, segurança, confiabilidade, manutenibilidade e portabilidade, entre outros. Os requisitos não funcionais do Sistema ACE podem ser observados no Quadro 2.

Quadro 2. Requisitos não Funcionais do Sistema ACE

RNF	Categoria	Descrição
RNF01	Usabilidade	O sistema deve possuir uma interface simples e intuitiva, permitindo que usuários com conhecimento básico em programação consigam utilizá-lo sem necessidade de treinamento formal.
RNF02	Usabilidade	Os logs de pontuação e de execução devem ser apresentados de forma organizada e devem poder ser exportados em formato CSV, para fins de estudo, documentação e análise externa.
RNF03	Desenvolvimento	O sistema deve ser desenvolvido utilizando a linguagem de programação Python, com o framework Flask para a camada web e banco de dados relacional (por exemplo, SQLite) para armazenamento das informações.
RNF04	Desempenho	O sistema deve responder às ações principais do usuário (definir parâmetros, gerar código e exibir resultados) em tempo hábil, mantendo um tempo de resposta aceitável para turmas em ambiente acadêmico (por exemplo, até alguns segundos por execução em populações padrão).
RNF05	Confiabilidade	O sistema deve tratar erros de geração ou execução de código sem interromper sua operação, exibindo mensagens compreensíveis ao usuário sempre que ocorrerem falhas de compilação, sintaxe ou execução.

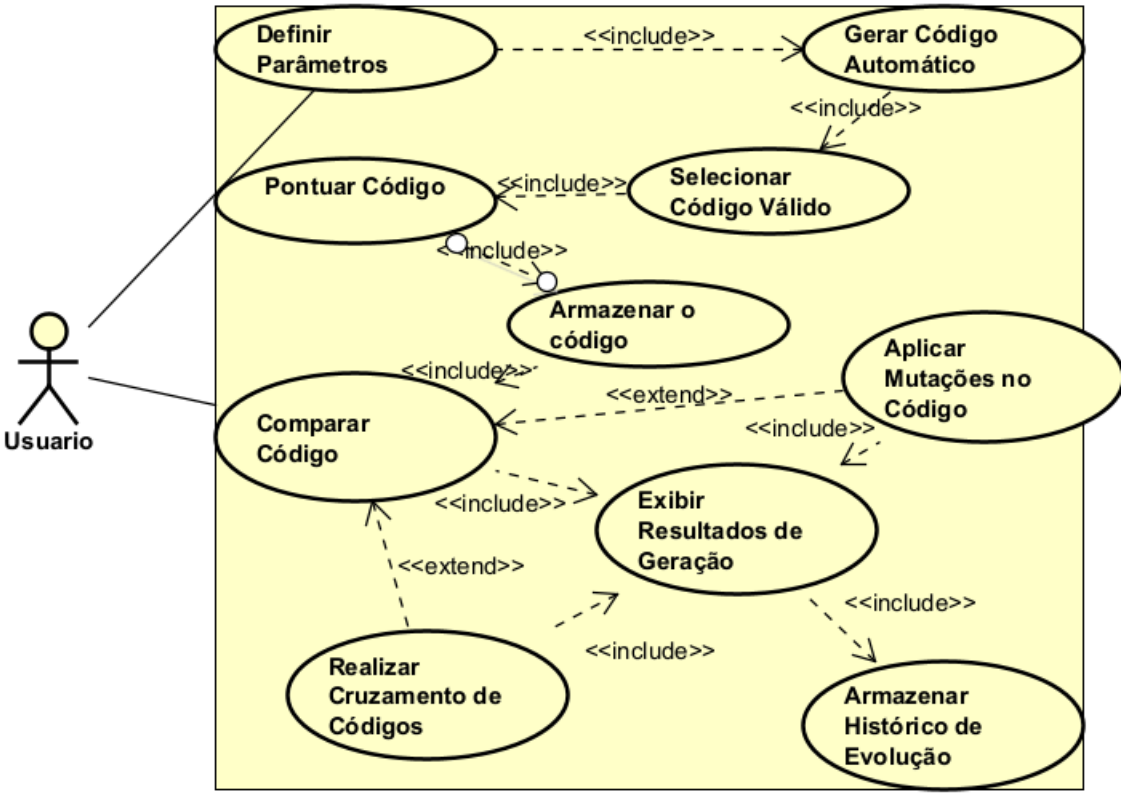
Fonte: Autor

6. MODELAGEM

6.1. Modelo De Casos De Uso

Um diagrama de caso de uso é um tipo de diagrama da UML (Unified Modeling Language) usado para mostrar, de forma simples, o que o sistema deve fazer do ponto de vista do usuário. Na Figura 11, podemos analisar o Diagrama de Caso de Uso do Sistema ACE.

Figura 11. Diagrama de Caso de Uso do Sistema

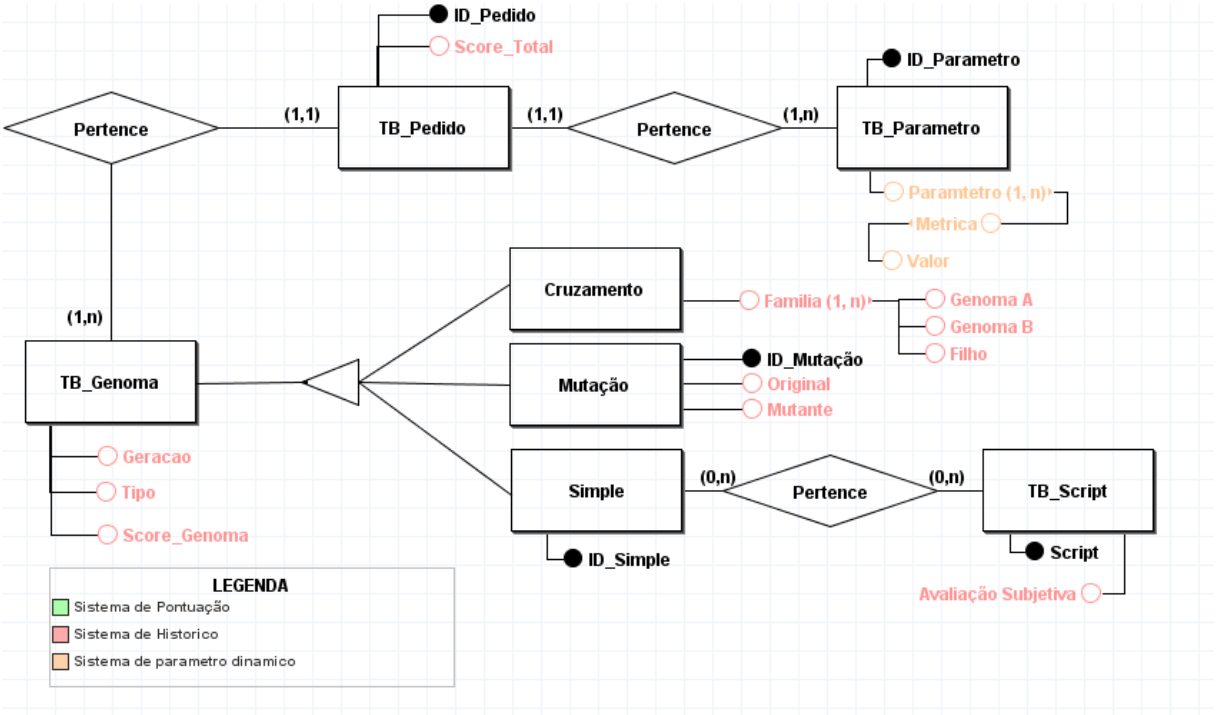


Fonte: Autor

6.2. Modelagem Conceitual Do Banco De Dados

A modelagem conceitual do banco de dados corresponde à etapa em que se realiza a representação abstrata dos dados de um sistema, independentemente de qualquer tecnologia ou SGBD específico. Seu objetivo é descrever, em nível conceitual, quais informações precisam ser armazenadas e como elas se relacionam entre si, tomando como referência os requisitos do domínio do problema. Na Figura 12, podemos analisar a Modelagem Conceitual do Sistema ACE.

Figura 12. Modelagem Conceitual do Sistema



Fonte: Autor

6.2.1. Descrição Técnica do Fluxo de Dados

O fluxo de informações no sistema ACE inicia-se com a configuração do experimento na entidade TB_Parametro. Nesta tabela, o atributo parâmetro funciona como uma estrutura composta por métrica e valor, definindo as diretrizes de execução do algoritmo genético. Cada parâmetro está vinculado a um registro na TB_Pedido por meio de uma relação de cardinalidade, onde o ID_Pedido centraliza os dados de uma sessão específica e o Score_Total armazena a média de desempenho da última geração processada.

A evolução biológica simulada é registrada na TB_Genoma, que pertence à TB_Pedido. Esta tabela armazena o histórico evolutivo por meio de campos fundamentais:

- Geracao: armazena o grau evolutivo da geração dos pedidos.
- Score_Genoma: registra a pontuação obtida pelo indivíduo na escala de avaliação.

- Tipo: armazena a especificidade da operação, sendo classificada como Cruzamento, Mutação ou Simples.

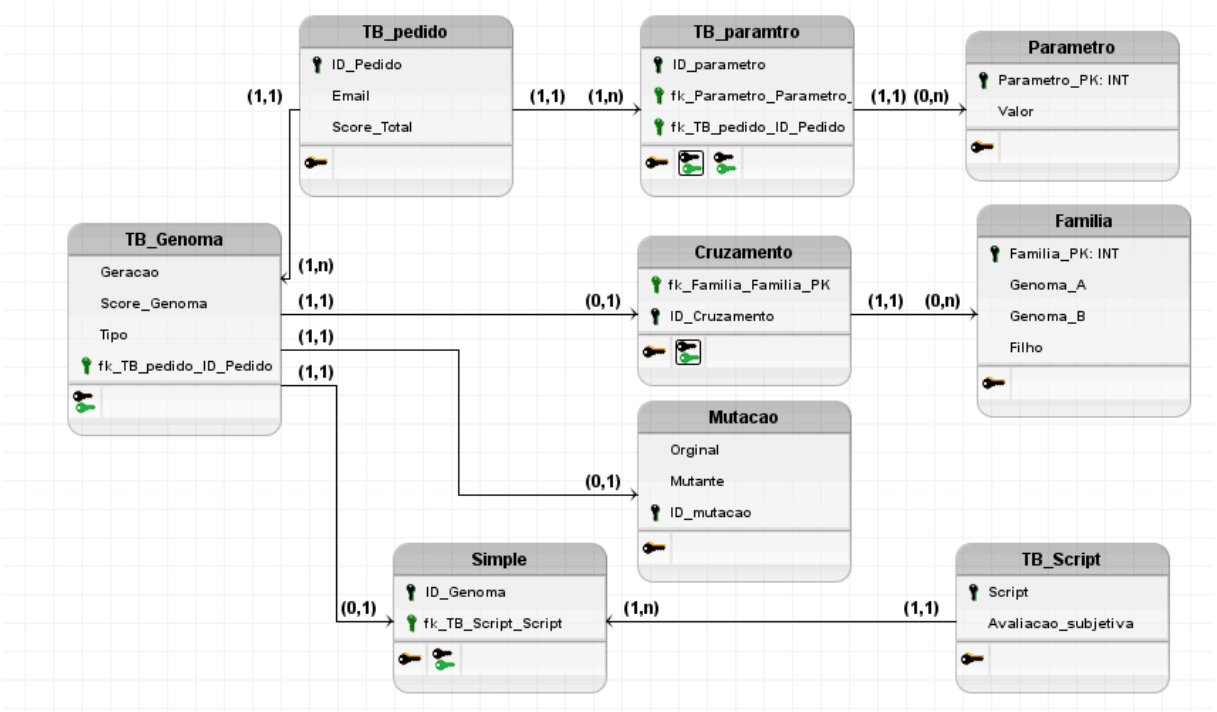
O detalhamento dessas especializações ocorre da seguinte forma:

- Cruzamento: documenta a relação de família, associando os genomas pais ao filho resultante.
- Mutação: mantém a rastreabilidade entre o código original e o código mutante após a alteração aleatória.
- Simple: refere-se ao primeiro código gerado, servindo como base para a população inicial.

Por fim, todos os indivíduos são persistidos na TB_Script, que armazena o conteúdo textual do código Python e a Avaliação Subjetiva, uma análise qualitativa gerada por um modelo de linguagem natural. Esta estrutura garante a rastreabilidade completa do processo, desde os parâmetros iniciais até o código final otimizado.

Já a modelagem lógica do banco de dados é a etapa em que o modelo conceitual, ainda abstrato, é traduzido para uma estrutura mais próxima da implementação real, porém ainda independente de um SGBD específico. Na Figura 13, podemos analisar a Modelagem lógica do Sistema ACE.

Figura 13. Modelagem lógica do Sistema



Fonte: Autor

A modelagem lógica traduz a estrutura conceitual para uma arquitetura técnica mais próxima da implementação em um SGBD, detalhando as chaves primárias (PK), chaves estrangeiras (FK) e os relacionamentos de integridade. O fluxo de dados estrutura-se da seguinte forma:

Gestão de Pedidos e Parâmetros: A tabela TB_pedido centraliza a execução, armazenando o identificador único e o Score_Total, que consolida o desempenho médio alcançado. Ela se relaciona com a TB_paramtro, que por sua vez utiliza a tabela Parametro para definir as métricas (como LOC ou tempo de execução) e os valores de calibração do algoritmo.

Hierarquia Genômica: A TB_Genoma é o núcleo do histórico evolutivo, registrando a Geracao, o Score_Genoma e o Tipo de operação realizada. Ela possui chaves estrangeiras que a conectam à TB_pedido e às tabelas de especialização.

Especializações de Evolução:

Cruzamento: Associa-se à tabela Familia para documentar a ancestralidade, registrando quais genomas "pais" deram origem ao "filho".

Mutacao: Armazena a rastreabilidade entre o código Original e o código Mutante gerado após a alteração aleatória.

Simple: Atua como o ponto de entrada para a população inicial, conectando o genoma ao código-fonte original.
Persistência de Código: A tabela TB_Script armazena o conteúdo textual do código Python gerado e a Avaliacao_subjetiva, que é o resultado da análise qualitativa processada pela inteligência artificial.

7. TECNOLOGIAS UTILIZADAS

A implementação do sistema ACE – *Aleatory Coding Engine* foi realizada utilizando um conjunto de tecnologias que buscou equilibrar simplicidade, flexibilidade e aderência ao contexto acadêmico do projeto. A linguagem de programação adotada foi o Python, em função de sua sintaxe acessível, ampla comunidade e grande disponibilidade de bibliotecas voltadas tanto ao desenvolvimento web quanto à experimentação com algoritmos. Essa escolha favoreceu a prototipação rápida, a escrita de código legível e a integração entre os módulos responsáveis pela geração, avaliação e armazenamento dos blocos de código produzidos pelos algoritmos genéticos.

Para o desenvolvimento da camada de interface do sistema, foram empregadas as tecnologias HTML, CSS e JavaScript, amplamente consolidadas no contexto da web. O HTML (HyperText Markup Language) foi utilizado para estruturar o conteúdo das páginas, definindo elementos como formulários, tabelas, botões e seções de visualização dos resultados. O CSS (Cascading Style Sheets) foi responsável pela definição do layout e da identidade visual da aplicação, permitindo organizar a disposição dos componentes na tela, ajustar tipografia, cores e espaçamentos, de modo a tornar a interface mais clara, agradável e coerente com o propósito didático do sistema ACE.

Já o JavaScript foi adotado para conferir interatividade às páginas, possibilitando, por exemplo, a atualização dinâmica de elementos da interface, a validação básica de entradas dos usuários e a melhoria da experiência de visualização dos dados gerados pelos algoritmos genéticos. A combinação dessas três tecnologias atendeu adequadamente às demandas do projeto, uma vez que oferece simplicidade na implementação, ampla compatibilidade com navegadores modernos e redução do tempo de desenvolvimento, sem a necessidade de recursos complexos adicionais para a construção da camada de apresentação.

Flask foi adotado como framework principal para o desenvolvimento da camada web do sistema ACE, em razão de suas características de simplicidade, flexibilidade e baixo acoplamento. Por se tratar de um microframework, o Flask não impõe uma estrutura rígida ao projeto nem exige a utilização de um banco de dados específico, o que oferece maior liberdade arquitetural e facilita a integração com diferentes soluções de persistência. Essa característica foi considerada vantajosa no contexto do ACE, pois permite que o sistema seja adaptado ou estendido futuramente para outros bancos de dados ou serviços, caso necessário.

Além disso, o Flask apresenta fácil curva de aprendizagem e implementação, disponibilizando um conjunto de ferramentas robustas para definição de rotas, tratamento de requisições HTTP, renderização de templates e gerenciamento básico de sessões. Tais funcionalidades foram suficientes para atender às necessidades do sistema, viabilizando a criação das interfaces responsáveis por receber parâmetros dos usuários, acionar os algoritmos genéticos, exibir resultados e organizar o fluxo de interação com a aplicação. Dessa forma, o uso do Flask contribuiu para um desenvolvimento mais ágil, mantendo o código enxuto e organizado, sem abrir mão da flexibilidade necessária em um projeto de caráter experimental e acadêmico.

Para apoiar o processo de concepção e implementação do banco de dados do sistema ACE, foram utilizadas ferramentas que atuam em momentos distintos do desenvolvimento: uma voltada à modelagem e outra à integração com o código.

O brModelo foi empregado na etapa de modelagem conceitual e lógica do banco de dados. A ferramenta se destacou por oferecer uma interface gráfica prática e intuitiva, permitindo a construção de diagramas entidade-relacionamento de forma visual. Isso contribuiu para reduzir o nível de abstração ao representar como os dados seriam armazenados, facilitando a compreensão das entidades, relacionamentos e cardinalidades envolvidas. Dessa maneira, o brModelo auxiliou na organização do esquema de dados e na documentação da estrutura do banco, servindo como ponte entre a fase de análise e a posterior implementação física.

Para a camada de acesso aos dados na aplicação, foi utilizado o Peewee, um Object-Relational Mapper (ORM) em Python. Esse ORM foi escolhido por oferecer simplicidade na definição e manipulação de modelos de banco de dados, permitindo mapear tabelas para classes Python de forma direta e clara. Sua adoção facilitou a escrita de consultas, inserções e atualizações, reduzindo a necessidade de comandos SQL explícitos no código do lado do servidor. Além disso, o Peewee foi

considerado uma solução mais leve e prática em comparação a outros ORMs, favorecendo uma integração mais segura e organizada entre a aplicação Flask e o banco de dados, contribuindo para a manutenibilidade e legibilidade do código-fonte.

Além disso, no desenvolvimento do sistema ACE foram utilizadas diversas bibliotecas e ferramentas de apoio, com o objetivo de organizar o ambiente de desenvolvimento, garantir segurança mínima de credenciais, ampliar o conjunto de métricas disponíveis e viabilizar testes e versionamento do código de forma estruturada.

A ferramenta venv, nativa do Python, foi empregada para a criação de ambientes virtuais. Seu uso permitiu isolar o conjunto de bibliotecas utilizadas pelo projeto, evitando conflitos de dependências com outras aplicações instaladas na mesma máquina. Dessa forma, foi possível versionar e gerenciar pacotes de forma independente para cada projeto, o que se mostrou essencial para a realização de testes com diferentes combinações de bibliotecas sem comprometer o ambiente global da linguagem.

Para o tratamento de informações sensíveis, como chaves de API, foi utilizada a biblioteca python-dotenv. Por meio dela, variáveis de ambiente puderam ser armazenadas em arquivos específicos (.env), que não são versionados no repositório público. Essa abordagem contribuiu para aumentar a segurança ao publicar o código no GitHub, evitando a exposição de credenciais utilizadas na comunicação com serviços externos.

No contexto da avaliação qualitativa do código gerado pelos algoritmos genéticos, foi utilizada a biblioteca ganai, responsável por intermediar o consumo da API do modelo de linguagem Gemini (LLM). Essa integração possibilitou a obtenção de análises qualitativas automatizadas sobre trechos de código, complementando as métricas quantitativas tradicionais com uma camada adicional de interpretação baseada em IA generativa. O próprio Gemini foi escolhido como serviço de LLM por oferecer, em sua modalidade utilizada, um número mínimo de consultas sem custo, o que viabilizou sua incorporação ao projeto para fins de pesquisa, permitindo a análise qualitativa dos códigos gerados sem impactar o orçamento do desenvolvimento.

Para a definição e cálculo de métricas estruturais sobre o código, foram empregadas bibliotecas específicas da linguagem Python. A biblioteca ast (Abstract Syntax Tree) foi utilizada para inspecionar a árvore sintática abstrata dos programas

gerados, possibilitando, entre outras coisas, a contagem do número de linhas ou nós por indivíduo, o que contribuiu para ampliar o conjunto de métricas analisadas. Já a biblioteca radon foi empregada para cálculo do índice de manutenibilidade e de outras métricas de complexidade de código, agregando uma dimensão importante à avaliação de qualidade, sobretudo no que diz respeito à legibilidade e à facilidade de manutenção. A biblioteca time, por sua vez, foi utilizada para mensurar o tempo de execução de determinadas operações, permitindo que o desempenho também fosse considerado como métrica relevante no conjunto de análises do sistema.

Por fim, o GitHub foi adotado como plataforma de versionamento e colaboração. O repositório remoto serviu tanto para armazenar historicamente as versões do código quanto para possibilitar testes em diferentes ambientes, uma vez que o projeto pode ser clonado e executado em qualquer máquina com os requisitos mínimos instalados. Além disso, o GitHub forneceu uma infraestrutura organizada para acompanhamento das evoluções do sistema, registro de alterações e eventual compartilhamento do projeto em contextos acadêmicos e de pesquisa.

Em conjunto, essas bibliotecas e ferramentas desempenharam papéis complementares, apoiando desde a organização do ambiente de desenvolvimento e segurança de credenciais até a ampliação e diversificação das métricas de avaliação e a manutenção estruturada do código-fonte do ACE.

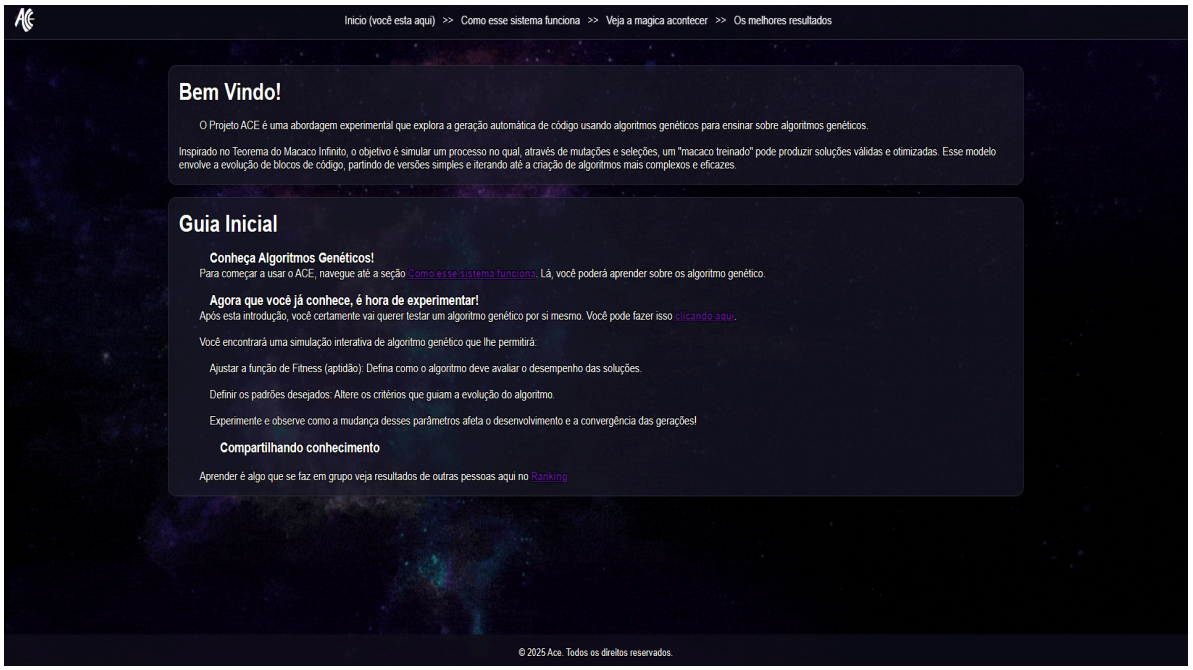
8. DESCRIÇÃO DO SISTEMA

8.1. Tela Inicial

A Tela Inicial (Figura 14) corresponde à rota raiz do sistema (/) e funciona como porta de entrada da aplicação web ACE. Visualmente, apresenta um tema escuro com imagem de fundo que remete ao espaço/galáxia, criando um ambiente estético associado à ideia de experimento e exploração. Na parte superior, há uma barra de navegação horizontal contendo o logotipo do sistema à esquerda e, ao centro, uma espécie de trilha de navegação com os seguintes itens: Início, Como esse sistema funciona, Veja a mágica acontecer e, Os melhores resultados. Essa

estrutura indica ao usuário em qual página se encontra e antecipa as demais seções da aplicação, contribuindo para a orientação e a organização do fluxo de navegação.

Figura 14. Tela inicial do sistema ACE



Fonte: Autor

No corpo da página, a interface é organizada em dois blocos principais em formato de cartões (cards), posicionados centralmente sobre o fundo escuro. O primeiro card, intitulado “Bem Vindo!”, apresenta um texto de introdução ao Projeto ACE, explicando que se trata de uma abordagem experimental voltada à geração automática de código por meio de algoritmos genéticos, com foco didático na compreensão desses algoritmos. Nessa seção, também é mencionada a inspiração no Teorema do Macaco Infinito e a ideia de evolução de blocos de código ao longo de mutações e seleções sucessivas, situando o usuário no contexto conceitual da ferramenta.

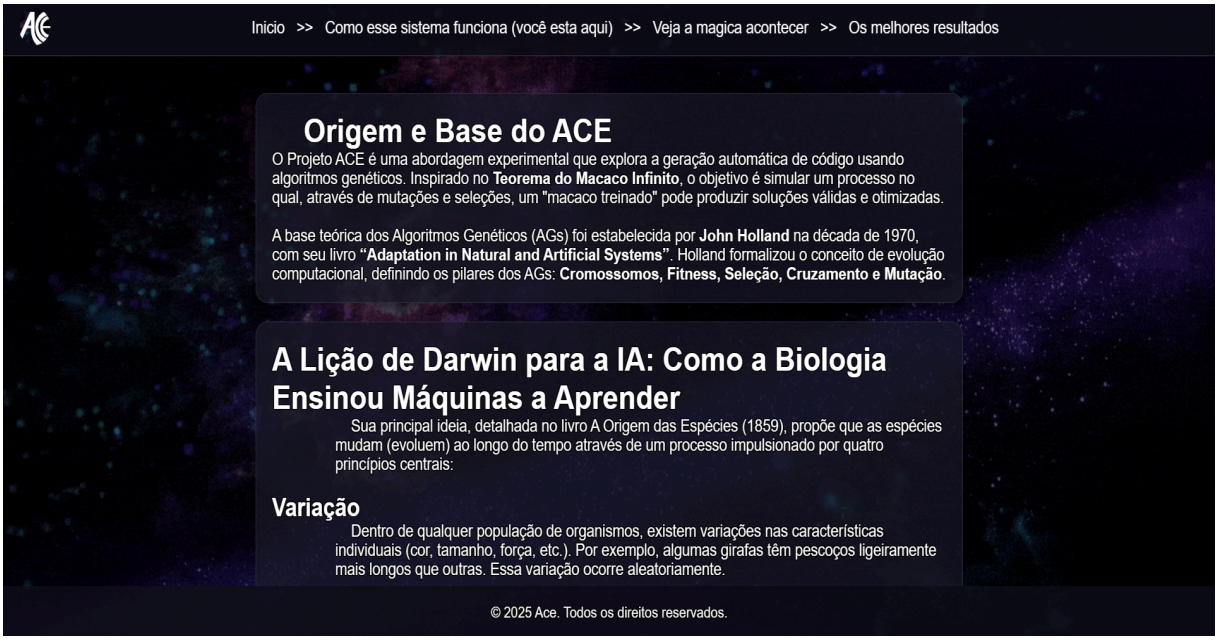
Logo abaixo, encontra-se o segundo card, den ominado “Guia Inicial”, que funciona como um roteiro de uso da aplicação. Ele apresenta subtítulos e trechos destacados que orientam o usuário em etapas: primeiro, convida-o a “Conhecer Algoritmos Genéticos”, sugerindo que acesse a seção Como esse sistema funciona para compreender o funcionamento do algoritmo; em seguida, incentiva a partir para a experimentação, indicando um link para a área interativa do sistema (por exemplo,

“clicando aqui”). Nessa mesma área são listadas, em forma de texto corrido, as principais ações que o usuário poderá realizar na simulação: ajustar a função de fitness, definir padrões desejados, observar o impacto da mudança de parâmetros na evolução das gerações e, por fim, acessar um ranking com resultados de outros experimentos, reforçando o caráter de compartilhamento de conhecimento.

8.2 Tela Como esse sistema funciona

A tela correspondente à rota “Como esse sistema funciona” (Figura 15) aprofunda a fundamentação conceitual do Projeto ACE. Mantendo o mesmo padrão visual da aplicação (fundo escuro com temática espacial, barra de navegação superior e rodapé com os direitos autorais), a página organiza o conteúdo em dois grandes blocos centrais.

Figura 15. Tela “Como esse sistema funciona” do sistema ACE, apresentando a origem teórica do projeto e a relação entre Algoritmos Genéticos e conceitos da Biologia Evolutiva.



Fonte: Autor

O primeiro card, intitulado “Origem e Base do ACE” apresenta o projeto de forma sintética, explicando seus objetivos e a proposta de utilizar algoritmos genéticos para geração e avaliação automática de código. Em seguida, a seção “A Lição de Darwin para a IA: Como a Biologia Ensinou Máquinas a Aprender” estabelece a relação entre as ideias centrais do livro A Origem das Espécies e o funcionamento dos Algoritmos Genéticos, destacando como conceitos de variação, seleção e adaptação inspiraram a computação evolutiva. Na sequência, o tópico “Algoritmos Genéticos Hoje” ilustra aplicações contemporâneas desses algoritmos em diferentes áreas, enquanto “Tecnologias Descendentes da Computação Evolutiva” apresenta outras tecnologias e abordagens que compartilham a mesma filosofia evolutiva, como programação genética e algoritmos evolutivos em geral.

A parte final da página é dedicada à explicação da estrutura de um Algoritmo Genético (AG). Nela, são descritas as principais etapas do processo: “Inicialização”, que traz um breve texto sobre a forma como são geradas as amostras iniciais que compõem a população; “Avaliação”, que explica como a qualidade de cada indivíduo é medida por meio de funções de fitness; “Seleção”, que discute diferentes métodos de escolha dos candidatos mais aptos; e “Operadores Genéticos”, seção em que são apresentadas as metodologias responsáveis por produzir a evolução propriamente dita, como cruzamento e mutação. Dessa forma, essa tela funciona como um material de estudo integrado ao sistema, permitindo que o usuário

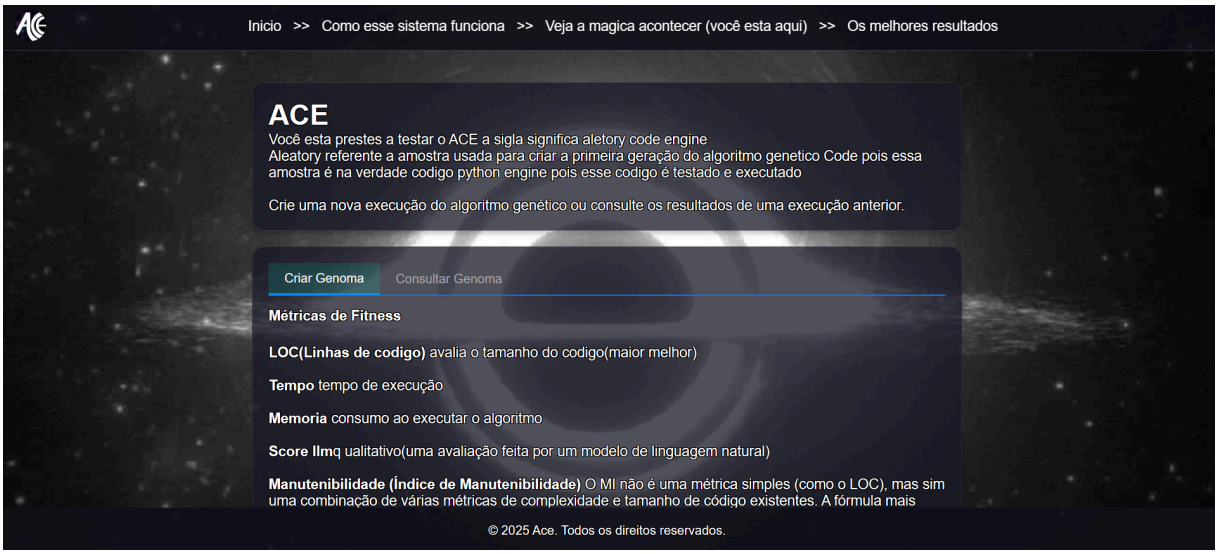
compreenda os fundamentos teóricos antes ou durante a utilização das funcionalidades interativas do ACE.

8.3 Tela Veja a mágica acontecer

A tela “Veja a mágica acontecer” (Figura 16) corresponde à tela em que o usuário explicita sua intenção de uso do sistema, funcionando como interface de calibração do algoritmo genético.

Na parte superior do conteúdo principal, um card apresenta o título “ACE” e um texto introdutório que contextualiza o usuário: explica o significado da sigla (*Aleatory Code Engine*), esclarece que a “amostra” inicial é, na verdade, código Python utilizado para compor a primeira geração do algoritmo genético e informa que essa amostra é executada e avaliada pelo sistema. Esse bloco também orienta o usuário a criar uma nova execução ou consultar resultados de execuções anteriores, estruturando o fluxo de interação.

Figura 16. Tela de calibração do algoritmo genético, permitindo a criação e consulta de genomas no sistema ACE.



Fonte: Autor

Logo abaixo, há um segundo card que organiza as funcionalidades em duas abas: “Criar Genoma” e “Consultar Genoma”.

- Na aba “Criar Genoma”, é apresentado um formulário de calibração que permite ao usuário definir parâmetros que determinam o funcionamento do algoritmo genético, como tamanho da população, número de gerações, taxas de cruzamento e mutação, além da escolha das métricas de fitness que serão utilizadas. Essa aba representa o momento em que o usuário “configura o experimento”, especificando como o algoritmo irá evoluir o código.

  Ao submeter o formulário, o sistema inicia a simulação do Algoritmo Genético (AG), que opera através da conversação entre diversos módulos. A simulação replica o processo evolutivo biológico para otimizar soluções.

  1. Criação da População Inicial

  O processo começa com o sistema AG Aleatory Gene, responsável pela criação aleatória da amostra inicial de indivíduos, que estabelece um paralelo direto com a biologia:

  - Indivíduo (Gene): Componente menor, análogo a um gene, representando uma solução potencial (um fragmento de código).
  - População (Genoma): O agrupamento desses indivíduos, análogo ao genoma, representa o conjunto de todas as soluções candidatas.

  Estruturas de Código (Genes)

  As estruturas de código que compõem cada indivíduo são selecionadas para garantir dois requisitos essenciais:

  1. Continuidade: As estruturas devem evitar problemas de execução, como loops infinitos ou pausas que levem a tempo de execução fatorial (complexidade indesejada). Estruturas como break e loops são excluídas. Exceções como try são permitidas por não causarem interrupção indesejada.

  2. Independência: As estruturas devem ser autossuficientes, não exigindo chamadas externas (funções) para funcionar. Isso evita desperdício de recursos e longos tempos de execução.

As estruturas incluídas são: estruturas condicionais (If, If else, If elif else, try), operadores lógicos (is, in, not) e variáveis booleanas (True e False). A escolha por uma lógica puramente booleana se baseia no entendimento de que a comparação entre dois valores resulta unicamente em Verdadeiro ou Falso, alinhando-se com a abstração fundamental do sistema.

Construção do Código

A construção do código de um indivíduo é feita por seleção aleatória de um grupo de estruturas iniciais. Dentro de estruturas como o If, as comparações possíveis são pré-escritas, deixando a variável a ser comparada como um espaço a ser preenchido aleatoriamente, o que permite a criação de uma tabela verdade durante a execução. Apenas códigos que cumprem os requisitos de tamanho mínimo são executados e testados; os não executáveis são descartados para acelerar o avanço da população.

2. Avaliação (Sistema Fitness)

Após a criação, o sistema Gerações encaminha a população ao sistema Fitness para avaliação de cada código (indivíduo). A avaliação é baseada em múltiplas métricas:

- Qualitativo: Uma LLM (Large Language Model) atribui uma pontuação subjetiva de 0 a 10 para a qualidade do código.
- Big O: Uma LLM avalia a estrutura e estabelece a pontuação de desempenho, sendo 0 para complexidade próxima à fatorial e 10 para tempo constante.
- LOC (Lines of Code): Avalia o tamanho do código; quanto menor, mais bem avaliado.
- Tempo: O tempo de execução real do código (métrica de desempenho).
- Memória: O consumo de RAM utilizada durante a execução (métrica de consumo de recurso).
- Índice de Manutenibilidade: Uma métrica para avaliar a "qualidade" da manutenção do código, combinando métricas de análise estática como Volume de Halstead, Complexidade Ciclomática e linhas de código.

3. Critério de Parada e Evolução

A População é analisada em relação aos critérios de parada definidos no formulário: se a média de score da população atingir o nível estipulado (Escore Médio) ou se o Número de Gerações máximo for alcançado.

Seleção, Cruzamento e Mutação

Se o critério de parada não for atendido, o sistema Forwarder coordena a logística da evolução, reduzindo a amostra da população através da Seleção (Torneio ou Elitismo), que elimina os indivíduos menos aptos.

- Seleção: É definida no formulário, sendo Torneio (onde o pior é eliminado) ou Elitismo (onde o melhor indivíduo é mantido).
- Cruzamento (Crossover) e Mutação: Preenchem o espaço vago. O cruzamento permite a combinação de características, e a mutação introduz diversidade (alterações aleatórias), impedindo que o AG encontre apenas uma solução localmente ótima.

Após a reprodução, a nova geração retorna ao sistema de avaliação e construção de gerações, reiniciando o ciclo. Ao final da rotina de repetições, os resultados são exibidos na aba “Consultar Genoma”[1].

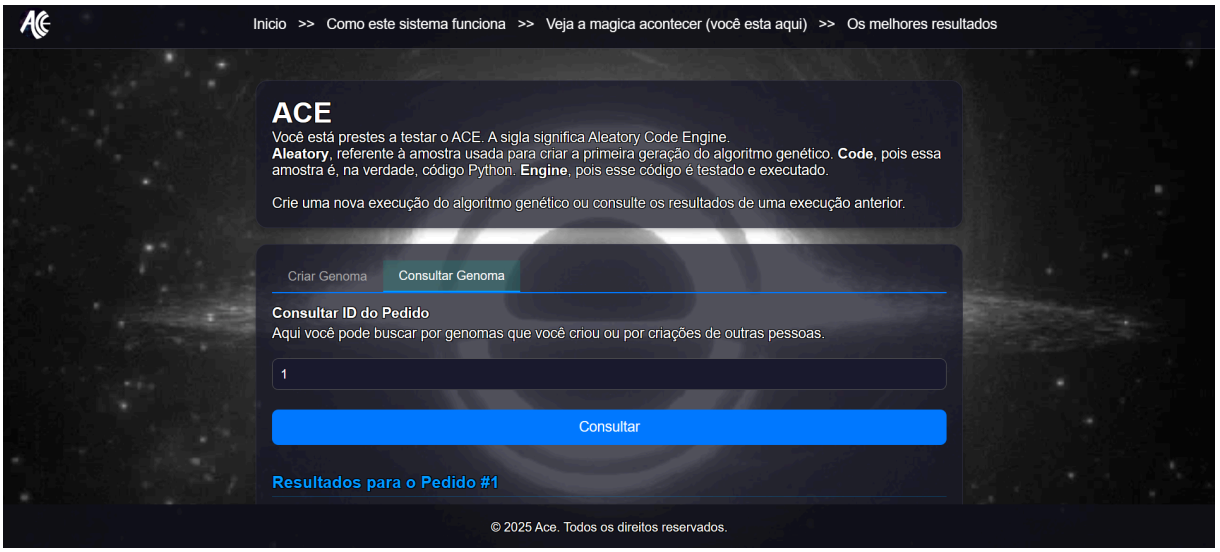
- A aba “Consultar Genoma” destina-se à recuperação de execuções já realizadas, permitindo ao usuário visualizar os resultados de algoritmos previamente calibrados por ele ou por outros usuários, favorecendo a comparação e o reuso de configurações bem-sucedidas.

- Ainda nesse card, a tela lista as métricas de fitness empregadas pelo sistema, como LOC (Linhas de Código), tempo de execução, consumo de memória, *score* qualitativo obtido via modelo de linguagem natural e índice de manutenibilidade. Cada métrica é acompanhada de uma breve descrição, esclarecendo seu papel na avaliação dos indivíduos gerados. Dessa forma, a rota /pedido materializa o núcleo interativo do ACE, pois é nela que o usuário

[1] O código-fonte do sistema está disponível no repositório GitHub: https://github.com/ViniciusALeal/ACE.

calibra o algoritmo genético, seleciona as métricas e, posteriormente, acessa os resultados de suas experiências evolutivas.

Figura 17. Tela de consulta permite a recuperação de execuções já realizadas inserido Id do pedido de execução



Fonte: Autor

- Na tela figura 17 é possível ver uma consulta ao inserir o ID de Execução (ou Identificador de Processamento) atua como a chave primária para a recuperação de todos os dados gerados durante o processamento. Ele viabiliza tanto a consulta individualizada de cada interação entre usuário e sistema quanto o acesso a registros de solicitações feitas por outros usuários (terceiros).

Figura 18. Tela de consulta parametros de metodos e metricas

Fonte: Autor

- Na tela 18 é apresentado os parâmetros e das métricas definidos pelo usuário, após a conclusão da execução, possibilita uma interpretação detalhada dos resultados. Este retorno viabiliza a análise dos dados representados no gráfico e na tabela de evolução por geração, permitindo a compreensão do efeito da interação entre a métrica, o método e o parâmetro adotados no processo.

Figura 19. Tela de consulta gráfico da evolução

Fonte: Autor

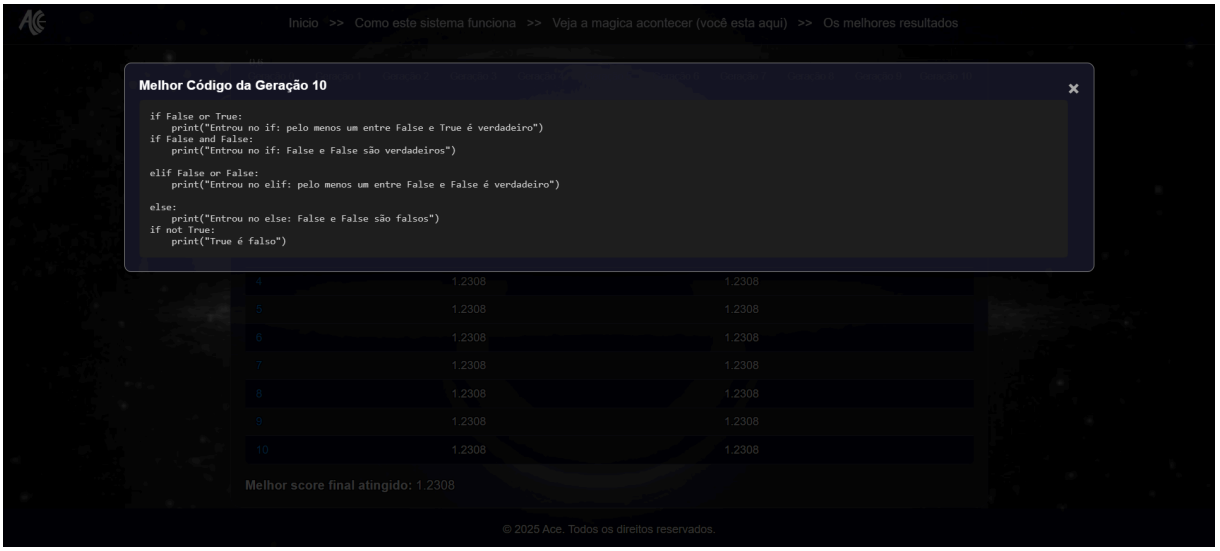
- "O gráfico de evolução por geração (Figura 19) complementa os dados da tabela correspondente, proporcionando uma escala visual imediata dos resultados. Essa representação gráfica facilita a percepção de tendências, anomalias e a progressão do processo ao longo das gerações, potencializando a interpretação analítica dos dados contidos na tabela.

Figura 20. Tela de consulta tabela evolução por geração

Fonte: Autor

- A tabela de resultados (Figura 20) exibe, de forma organizada, todos os valores dos parâmetros processados. Sua estrutura apresenta os dados dispostos em colunas, sendo a primeira dedicada ao número da geração, seguida pelos indicadores de desempenho, como o maior score (melhor indivíduo) e o score médio da população de cada geração

Figura 21. Tela de consutla interface do melhor código



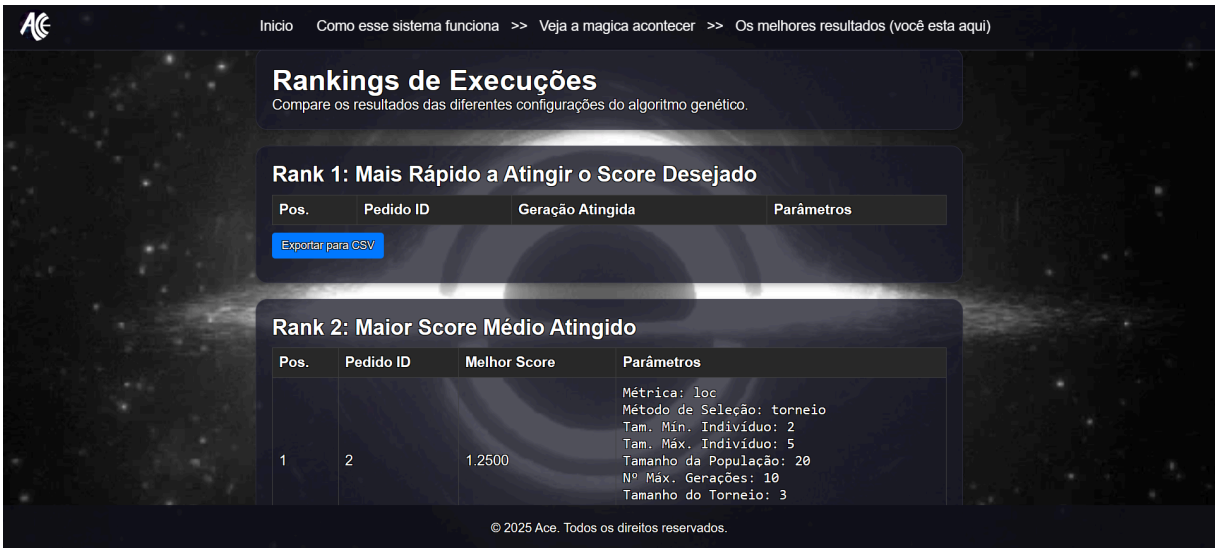
Fonte: Autor

- A seleção de uma geração específica na tabela resulta na recuperação do registro completo do indivíduo de melhor desempenho (Figura 21) designado internamente como 'melhor código'. Este recurso viabiliza a rastreabilidade da evolução do melhor indivíduo ao longo das gerações. Dessa forma, é possível realizar uma análise detalhada sobre o impacto da configuração dos parâmetros na performance e nas características desse indivíduo.

8.4 Tela Os melhores Resultados

A tela da Figura 22, disponibiliza ao usuário o sistema de ranqueamento das execuções, permitindo comparar o desempenho de diferentes genomas e configurações do algoritmo genético.

Figura 22. Tela de ranking de execuções, apresentando a comparação entre diferentes configurações do algoritmo genético no sistema ACE.



Fonte: Autor

No topo do conteúdo principal, um card apresenta o título “Rankings de Execuções” e um subtítulo que orienta o usuário a comparar os resultados das diferentes configurações do algoritmo. Em seguida, a tela é organizada em blocos de ranking, cada um com foco em um critério de desempenho.

O primeiro bloco, “Rank 1: Mais Rápido a Atingir o Score Desejado”, lista as execuções que alcançaram um determinado *score*-alvo no menor número de gerações. A tabela apresenta colunas como posição, identificador do pedido (genoma) e geração em que o objetivo foi atingido, além de um botão para exportação dos dados em formato CSV, possibilitando análises externas.

O segundo bloco, “Rank 2: Maior *Score* Médio Atingido”, contempla execuções avaliadas de acordo com o maior score médio obtido ao longo de um número pré-definido de gerações. Nele são exibidos, para cada pedido, a posição no ranking, o identificador, o melhor score e os parâmetros associados à execução (métrica utilizada, método de seleção, tamanho da população, limites de indivíduos, quantidade de gerações, entre outros).

Dessa forma, essa tela funciona como um painel comparativo de desempenho, permitindo que o usuário identifique rapidamente quais configurações de parâmetros produziram resultados mais eficientes ou mais estáveis, além de

favorecer o compartilhamento e a reprodutibilidade dos experimentos conduzidos no sistema ACE.

9. CONSIDERAÇÕES FINAIS

O processo de validação evidencia que o ACE ainda tem uma longa jornada de aperfeiçoamento pela frente, mas já cumpre, no momento, seu propósito central de fornecer conhecimento aos usuários. Em uma etapa futura, planeja-se potencializar esse recurso por meio de duas frentes principais de melhoria.

A primeira diz respeito à refatoração do núcleo aleatório, que será reescrito para substituir a operação baseada em dicionários por estruturas em árvores. Essa mudança permitirá uma compreensão mais precisa da estrutura real do código, facilitando a criação de layouts mais variados, que contemplem todas as sintaxes disponíveis, elevando, assim, o nível de complexidade e de fidelidade das respostas geradas.

A segunda frente envolve o aprimoramento da interface e da experiência do usuário (UI/UX). Nessa direção, busca-se aproximar ainda mais o ACE de seus usuários, investigando continuamente suas necessidades e expectativas, de modo a oferecer uma interação mais intuitiva, agradável e efetiva.

Conclui-se que o trabalho realizado até o momento ainda não finalizou o projeto ACE, existindo um amplo espaço para aprimoramento e desenvolvimento futuro. Transcende o campo da IA e ingressando na ciência de dados a fim de permitir obter conclusões mais sólidas sobre os resultados gerados pelo ACE.

10. REFERÊNCIAS

BEN COPPIN. *Artificial Intelligence Illuminated.* Jones & Bartlett Learning, 2004.

The Python Software Foundation. *Python Documentation.* Disponível em: https://docs.python.org/3/

Repositório de software para a linguagem de programação Python. Disponivel em: https://pypi.org/

11. APÊNDICE

APÊNDICE A – Questionário da Pesquisa