UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS SOBRAL

CURSO DE GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

FRANCISCA NEUVÂNIA DE PAULA BARROS

UMA TAXONOMIA PARA AMBIENTE DE DESENVOLVIMENTO INTEGRADO BASEADA EM ENTIDADES

SOBRAL

2025

FRANCISCA NEUVÂNIA DE PAULA BARROS

UMA TAXONOMIA PARA AMBIENTE DE DESENVOLVIMENTO INTEGRADO BASEADA EM ENTIDADES

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da computação do Campus Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia da computação.

Orientador: Prof. Dr. Fischer Jônatas Ferreira

SOBRAL

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

B277t Barros, Francisca Neuvânia de Paula.
UMA TAXONOMIA PARA AMBIENTE DE DESENVOLVIMENTO INTEGRADO BASEADA EM ENTIDADES / Francisca Neuvânia de Paula Barros. – 2025.
49 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia da Computação, Sobral, 2025.
Orientação: Prof. Dr. Fischer Jônatas Ferreira.

1. Taxonomia. 2. Ambiente de Desenvolvimento Integrado. 3. IDE. 4. Desenvolvimento de Software. 5. Ferramentas de desenvolvimento. I. Título.

CDD 621.39

FRANCISCA NEUVÂNIA DE PAULA BARROS

UMA TAXONOMIA PARA AMBIENTE DE DESENVOLVIMENTO INTEGRADO BASEADA EM ENTIDADES

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da computação do Campus Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia da computação.

Aprovada em: 04 de Agosto de 2025

BANCA EXAMINADORA

Prof. Dr. Fischer Jônatas Ferreira (Orientador)
Universidade Federal de Itajubá (UNIFEI)

Prof. Dr. Evilásio Costa Júnior
Universidade Federal do Ceará (UFC)

Prof. Me. Erick Aguiar Donato
Universidade Federal do Ceará (UFC)

Aos meus pais, por me deixarem sonhar e ajudarem a realizar. À minha vó Maria de Barros, *in memoriam*, que ajudou a me criar e tantas vezes me chamou ao pé da cerca dizendo que era hora de ir para a escola e, com isso, me ensinou, com simplicidade e amor, o valor do estudo.

AGRADECIMENTOS

Agradeço, primeiramente, a Deus, por iluminar meu caminho, fortalecer minha fé e me sustentar nos dias difíceis.

Sou grata aos meus pais, Maria Vanúsia de Paula e Francisco de Assis Barros, e aos meus irmãos, João de Paula, Inary e Maria Clara, pelo esforço, dedicação, cuidado e por nunca medirem esforços para que eu tivesse oportunidades e pudesse seguir meus sonhos, mesmo nos momentos de dificuldade.

Agradeço ao meu parceiro, Daniel Alécio do Nascimento, por todo apoio, dedicação, paciência e incentivo ao longo dessa caminhada.

Ao meu orientador, professor Dr. Fischer Jônatas Ferreira, agradeço por me ajudar a chegar neste momento tão importante da minha vida. Sou grata pela paciência, apoio, ensinamentos e incentivos durante toda a orientação e trajetória acadêmica.

Minha gratidão aos membros da banca avaliadora, professor Dr. Evilásio Costa Júnior e professor Me. Erick Donato Aguiar, pelo tempo dedicado à leitura e avaliação deste trabalho e por participarem de um momento tão significativo da minha vida.

Agradeço também aos professores do curso de Engenharia de Computação da UFC – Campus Sobral, pelo conhecimento compartilhado e por todo o suporte ao longo da graduação.

Sou grata à Michelle Fontenele, secretária do curso de Engenharia de Computação da UFC – Campus Sobral, por toda a ajuda prestada em diversos momentos.

Aos meus amigos da graduação e do trabalho, agradeço pelas contribuições, companheirismo, amizade e incentivo durante essa jornada.

Minha gratidão a todos que, de alguma forma, contribuíram para o desenvolvimento deste trabalho.

RESUMO

Os Ambientes de Desenvolvimento Integrado (IDEs) desempenham um papel crucial no processo de desenvolvimento de software, pois proporcionam um conjunto unificado de ferramentas que tem como intuito facilitar as diversas etapas da construção do software, desde a edição de código até os testes e depuração. No entanto, existe um grande número de IDEs disponíveis no mercado, tornando assim um desafio para os desenvolvedores saber escolher a ferramenta mais adequada para as suas necessidades. Este estudo tem como objetivo propor uma taxonomia abrangente para categorizar os IDEs com base em suas principais características, tais como: linguagens de programação suportadas, ferramentas integradas, depuradores, gestão de repositórios de código, integração com ferramentas externas, entre outros aspectos. Oferecendo um sistema de classificação que leva em consideração as diferentes necessidades e contextos do desenvolvimento de software.

Palavras-chave: IDEs. Ambientes de Desenvolvimento Integrado. Ferramentas de Desenvolvimento. Desenvolvimento de Software. Taxonomia.

ABSTRACT

Integrated Development Environments (IDEs) play a crucial role in the software development process, as they provide a unified set of tools intended to facilitate the various stages of software construction, from code editing to testing and debugging. However, there is a large number of IDEs available on the market, making it a challenge for developers to know how to choose the most suitable tool for their needs. This study aims to propose a comprehensive taxonomy to categorize IDEs based on their main characteristics, such as: supported programming languages, integrated tools, debuggers, code repository management, integration with external tools, among other aspects. Offering a classification system that takes into account the different needs and contexts of software development.

Keywords: IDEs. Integrated Development Environments. Development Tools. Software Development. Taxonomy.

LISTA DE FIGURAS

Figura 1 – Processo iterativo de desenvolvimento de taxonomias 17
Figura 2 – Ambientes de desenvolvimento integrado mais utilizados em 2021 e 2024 . 20
Figura 3 – Ambientes de desenvolvimento integrado mais pesquisados no Google . . . 20
Figura 4 – Versão proposta da taxonomia para IDE 24
Figura 5 – Fluxo de condução do estudo 26
Figura 6 – Tempo de experiência dos entrevistados 32
Figura 7 – Ambientes utilizados pelos entrevistados 32
Figura 8 – Linguagens de programação utilizadas pelos entrevistados 33
Figura 9 – Nuvem de palavras da análise da pergunta P1 34
Figura 10 – Nuvem de palavras da análise da pergunta P2 35
Figura 11 – Nuvem de palavras da análise da pergunta P3 36
Figura 12 – Análise da pergunta P6 - sugestão de melhoria para a taxonomia 37
Figura 13 – Versão final da taxonomia para IDE 40
Figura 14 – Aplicação da taxonomia ao *Visual Studio Code* 41
Figura 15 – Aplicação da taxonomia ao *Pycharm (community)* 42

LISTA DE TABELAS

Tabela 1 – Funcionalidades comuns das ferramentas presentes nas pesquisas de 2021 e 2024 - Parte 1 21
Tabela 2 – Funcionalidades comuns das ferramentas presentes nas pesquisas de 2021 e 2024 - Parte 2 22
Tabela 3 – Formulário de perfil do entrevistado 28
Tabela 4 – Formulário de perfil do entrevistado 31
Tabela 5 – Resumo dos trabalhos relacionados 44

SUMÁRIO

1 INTRODUÇÃO 11
2 FUNDAMENTAÇÃO TEÓRICA 13
2.1 Visão geral sobre o desenvolvimento de software 13
2.2 Taxonomia 15
2.2.1 Importância e Geração de Taxonomias 16
2.3 Ambiente de desenvolvimento Integrado - IDE 18
3 METODOLOGIA 23
3.1 Tipo de pesquisa 23
3.2 Construção da taxonomia 23
3.3 Método de coleta 26
3.3.1 Etapas de condução do estudo 26
3.3.2 Definições das perguntas de entrevista 27
3.3.3 Criação de formulário de perfil 28
3.3.4 Entrevista piloto 28
3.3.5 Realização das entrevistas 29
3.3.6 Processamento dos dados 29
3.3.7 Análise dos dados 29
4 RESULTADOS 31
4.1 Resultado da pesquisa 31
4.1.1 Perfil dos participantes 31
4.1.1.1 P1: Na sua experiência, quais são os critérios que você considera ao escolher um IDE? *33*
4.1.1.2 P2: Como você percebe a evolução dos IDEs ao longo do tempo? Houve mudanças importantes nas que você usa ou usava? *34*
4.1.1.3 P3: Você acredita que existe um conjunto comum de características que toda IDE deveria ter? *35*
4.1.1.4 P4: Ao analisar esta taxonomia que estou propondo, ela representa bem os principais aspectos que você observa em um IDE? *36*
4.1.1.5 P5: Alguma parte da taxonomia parece desnecessária, incompleta ou mal organizada? *36*

4.1.1.6 P6: Há alguma categoria ou conceito que você incluiria na taxonomia? . . 37
4.1.2 Refinamento da taxonomia 38
4.2 Exemplos de uso da taxonomia 40
5 TRABALHOS RELACIONADOS 43
6 CONCLUSÕES E TRABALHOS FUTUROS 45
REFERÊNCIAS 47

1 INTRODUÇÃO

O desenvolvimento de software ao longo das últimas décadas vem exigindo soluções cada vez mais complexas, robustas e eficientes, impulsionadas pela transformação digital e pela crescente demanda por sistemas que atendam a diferentes contextos e plataformas. Essa evolução tem exigido não apenas o aprimoramento das linguagens de programação e metodologias de desenvolvimento, mas também a utilização de ferramentas que agilizem o processo e aumentem a produtividade das equipes. Nesse cenário, destacam-se os Ambientes de Desenvolvimento Integrado do inglês Integrated Development Environments (IDE), caracterizados como aplicações de software que centralizam, em uma única interface, recursos como edição, compilação, depuração, testes e empacotamento, facilitando o trabalho do desenvolvedor e reduzindo o tempo necessário para entrega de soluções. (PRESSMAN; MAXIM, 2021)

A ampla variedade de IDEs disponíveis no mercado, cada um com suas próprias características e orientações a diferentes contextos de uso, torna o processo de escolha da ferramenta mais adequada um desafio recorrente para profissionais e equipes. A ausência de um referencial estruturado para avaliação e comparação pode levar a decisões baseadas apenas em preferências pessoais ou experiências prévias, nem sempre resultando na escolha mais eficiente para o projeto. Essa lacuna evidencia a necessidade de metodologias que auxiliem na sistematização da análise dessas ferramentas, considerando múltiplos critérios de classificação.

A motivação para este trabalho surge da necessidade de sistematizar e organizar as informações sobre ambientes de desenvolvimento integrado, de modo a oferecer um referencial claro para sua análise e comparação. Observou-se, tanto na literatura quanto na prática profissional, que há uma grande diversidade de IDEs com diferentes conjuntos de funcionalidades, mas poucas propostas que reúnam essas características de forma estruturada. Embora existam estudos que tratem de aspectos específicos dessas ferramentas, como o suporte a múltiplas linguagens ou a integração com sistemas de controle de versão, são poucas as taxonomias que considerem de maneira integrada critérios técnicos, de usabilidade e de extensibilidade, permitindo comparações entre diferentes soluções. (NICKERSON *et al.*, 2013; ARAÚJO; PINTO, 2019)

O objetivo geral deste estudo é elaborar uma taxonomia para ambientes de desenvolvimento integrado, de modo a classificar e organizar essas ferramentas e orientar sua seleção com base em critérios diversificados. Para atingir esse objetivo, o trabalho foi estruturado em duas etapas principais: a primeira consistiu na construção da taxonomia, fundamentada em revisão de literatura, análise de documentação técnica e levantamento de funcionalidades a partir de

fontes especializadas; a segunda envolveu entrevistas com profissionais da área, com o objetivo de validar e refinar a proposta, garantindo seu alinhamento com a prática de mercado.

Os resultados obtidos indicam que funcionalidades como suporte a múltiplas linguagens, extensões, controle de versão, terminal integrado e depuração estão entre as mais valorizadas pelos profissionais, reforçando a relevância das dimensões propostas na taxonomia. Além de propor um modelo aplicável, este estudo busca contribuir para o avanço do conhecimento na área, servindo como base para profissionais, pesquisadores e estudantes na escolha fundamentada de IDEs, bem como para pesquisas futuras, estudos comparativos e revisões periódicas que acompanhem a evolução dessas ferramentas e as tendências tecnológicas.

A organização deste trabalho está estruturada da seguinte forma: o Capítulo 2 apresenta a fundamentação teórica, abordando o desenvolvimento de software, o conceito de taxonomia e a evolução dos IDEs; o Capítulo 3 descreve a metodologia empregada, desde a construção da taxonomia até a coleta e análise dos dados; o Capítulo 4 apresenta e discute os resultados obtidos, incluindo a validação da proposta junto a profissionais da área; o Capítulo 5 apresenta os trabalhos relacionados; e o Capítulo 6 mostra as conclusões e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os fundamentos teóricos essenciais para a compreensão do tema central deste trabalho, que busca propor uma taxonomia de IDEs. Para isso, é abordado o contexto do desenvolvimento de software, destacando sua importância crescente na sociedade e a evolução contínua ao longo do tempo. Em seguida, é explorado o conceito de taxonomia, que desempenha um papel crucial na organização e classificação de informações.

2.1 Visão geral sobre o desenvolvimento de software

O desenvolvimento de software é um componente fundamental na sociedade moderna, impulsionado pelos avanços da tecnologia e pela crescente dependência de sistemas computacionais. Desde aplicativos móveis até sistemas operacionais e plataformas web, o desenvolvimento de software tornou-se essencial para a criação de soluções tecnológicas inovadoras. Além disso, o papel evolutivo do software, conforme destacado por (PRESSMAN, 2010), desempenha um papel crucial nesse contexto. O software, como produto tecnológico, possui a capacidade de ser modificado e aprimorado ao longo do tempo, o que permite sua adaptação às mudanças no ambiente operacional, às necessidades dos usuários e aos avanços tecnológicos. As demandas em constante evolução do mercado e a rápida mudança nas preferências e expectativas dos usuários exigem que os softwares sejam flexíveis e capazes de acompanhar essas transformações (PRESSMAN; MAXIM, 2021).

No processo de desenvolvimento de software, os desenvolvedores enfrentam uma série de desafios e dificuldades. Essa capacidade de evolução do software se manifesta de diferentes formas. Por um lado, correções e atualizações são necessárias para solucionar erros ou falhas identificadas, garantindo assim a segurança e confiabilidade do software. Por outro lado, o acréscimo de novas funcionalidades permite que o software atenda às demandas emergentes e ofereça um maior valor aos usuários, mantendo-se competitivo no mercado tecnológico.

No desenvolvimento de software a modularidade é um dos desafios mais significativos, com ela é possível dividir o código em módulos independentes e bem definidos, o que resulta em uma série de benefícios. A abordagem modular facilita a compreensão do código, uma vez que cada módulo trata de uma funcionalidade específica e contribui para a reutilização de código, promovendo a eficiência e a escalabilidade no desenvolvimento. A modularidade também desempenha um papel crucial na evolução do software, pois possibilita que novas

funcionalidades sejam adicionadas de forma mais eficiente e que o software seja escalável para atender às necessidades futuras. Além disso, ela também facilita a manutenção do software, já que as alterações podem ser realizadas em módulos individuais sem afetar o restante do sistema (OLIVEIRA, 2017).

Outro desafio importante é a configuração do software, que envolve adaptá-lo a diferentes ambientes operacionais e requisitos dos usuários. A flexibilidade na configuração é essencial para garantir a adaptação adequada do software (JR, 2010). Adicionalmente, as refatorações são práticas constantes utilizadas no desenvolvimento de software, pois permitem a reestruturação do código para melhorar sua legibilidade, eficiência, manutenibilidade ou extensibilidade, sem alterar seu comportamento externo. Isso contribui para a redução da complexidade do código e o aprimoramento de sua qualidade (FOWLER, 1997).

As funcionalidades de rastrear e depurar e a integração contínua do código também são tarefas importantes no desenvolvimento de software. O rastreamento, bem como destacado em sua definição, trata-se do ato ou efeito de rastrear. Em um IDE, ela registra e monitora a execução do código em tempo real, ajudando a identificar falhas e entender o comportamento do programa (MATLOFF, 2012). A depuração é o procedimento pelo qual as causas de comportamentos incorretos em sistemas de software são identificadas e isoladas, e os desenvolvedores precisam ser capazes de identificar e corrigir esses erros ou falhas para garantir a segurança e confiabilidade do software (MEGA; KON, 2005). Já a integração contínua refere-se a prática de desenvolvimento de software onde os desenvolvedores juntam suas alterações de código em um repositório central, facilitando o gerenciamento de versões e alterações ao longo do tempo, bem como a colaboração entre os membros da equipe de desenvolvimento (SOUSA, 2022).

A utilização de um IDE (Integrated Development Environment) desempenha um papel fundamental no desenvolvimento de software. O IDE é uma ferramenta que oferece recursos integrados para edição, compilação, depuração e teste de código, proporcionando aos desenvolvedores uma plataforma completa para realizar suas tarefas de desenvolvimento de forma eficiente (LIMA, 2022). Esses desafios e a necessidade de ferramentas adequadas evidenciam a importância de uma abordagem sistemática e estruturada no processo de desenvolvimento de software. A adoção de uma taxonomia para classificar os IDEs pode auxiliar na categorização e organização de conceitos, técnicas e ferramentas relacionadas ao desenvolvimento de software, fornecendo uma base para a tomada de decisões da escolha de IDEs mais adequadas para as necessidade do desenvolvedor.

2.2 Taxonomia

A taxonomia é um campo de estudo dedicado à classificação, organização e categorização de elementos com base em suas características comuns. Surgido do grego, o termo ”taxonomia” é composto pelas palavras “táxis”, que significa ”ordem”, e ”nomos”, que se refere a ”lei” ou ”norma”. Juntas, essas palavras formam o termo "taxonomia", que representa a ciência da classificação. A taxonomia permite organizar e estruturar o conhecimento, identificando relações e padrões entre diferentes elementos (CAMPOS; GOMES, 2007).

Há diferentes tipos de taxonomia, sendo a biológica uma das mais conhecidas. Nesse ramo, os organismos são classificados em diferentes categorias, como espécie, gênero, família, ordem, classe, filo e reino, com base em características físicas, genéticas e evolutivas. Historicamente, a taxonomia passou a ter maior relevância a partir do século XVIII, com os estudos de Carl von Linné (Lineu), que propôs um sistema padronizado de nomenclatura para os seres vivos. Sua abordagem se destacou por nomear as classes com base em características gerais e observáveis, sem depender de propriedades específicas. Esse princípio reforça a importância dos nomes na definição e compreensão das categorias, sendo um aspecto essencial da classificação em diversas áreas do conhecimento (FRAIX-BURNET, 2016).

Nos sistemas de informação, a taxonomia também desempenha um papel importante na organização e recuperação de informações, sendo utilizada para estruturar conjuntos complexos de elementos de forma hierárquica e lógica (ARAUJO, 1985). Essa organização permite que conceitos sejam navegados em múltiplos níveis, como classes, subclasses e sub-subclasses, facilitando a busca, a análise e a tomada de decisão (CAMPOS M. L. A.; GOMES, 2008)

No contexto da Engenharia de Software, a taxonomia é aplicada para classificar métodos, ferramentas, práticas e atributos de sistemas, favorecendo a padronização e a comparabilidade entre soluções (GLASS; VESSEY, 1995). Por exemplo, (USMAN *et al.*, 2017b) propuseram uma taxonomia para classificação de técnicas de Engenharia de Requisitos, enquanto (BORGES *et al.*, 2020) desenvolveram uma taxonomia para categorizar defeitos de software. Esses trabalhos mostram que a adoção de taxonomias favorece a comunicação entre equipes e contribui para consolidar o conhecimento em domínios específicos.

Além disso, o uso de taxonomias em desenvolvimento de software também se aplica à classificação de ferramentas e ambientes, como demonstrado por (ZHANG *et al.*, 2018), que elaboraram uma taxonomia para ferramentas de análise de big data. Essa abordagem permite descrever e comparar soluções tecnológicas a partir de dimensões e atributos definidos,

favorecendo a organização sistemática das informações. Nesse sentido, a taxonomia também auxilia na identificação de relações entre diferentes tipos de dados, facilitando a descoberta de informações relacionadas e apoiando a tomada de decisão com base nas características e propriedades dos elementos classificados (ARAÚJO; PINTO, 2019).

2.2.1 Importância e Geração de Taxonomias

As taxonomias desempenham um papel fundamental tanto na pesquisa científica quanto na gestão da informação, pois auxiliam na análise e compreensão de problemas complexos. Glass e Vessey (2002, apud ANDRADE *et al.*, 2020), afirmam que a adoção de uma taxonomia bem estruturada contribui para a organização do conhecimento em uma determinada área, favorecendo o avanço científico. Miller e Ruth (2003, apud ANDRADE *et al.*, 2020) também reforçam que taxonomias são úteis como ferramentas para análise, discussão e ensino.

Além de promover uma melhor compreensão sobre o objeto de estudo, o uso de taxonomias facilita a identificação de divergências entre trabalhos anteriores. Sokal (1974, apud ANDRADE *et al.*, 2020) observa que abordagens iniciais classificavam elementos com base em apenas uma característica, muitas vezes escolhida arbitrariamente. Já as abordagens modernas consideram múltiplas características, enriquecendo a classificação, embora exijam métodos mais sofisticados.

No contexto da Engenharia de Software, a construção de taxonomias segue uma abordagem sistemática, que visa garantir rigor científico e validade na categorização de elementos. (NICKERSON *et al.*, 2013) cita que a formulação de uma taxonomia deve ser conduzida por meio de um processo iterativo, que combina tanto abordagens conceituais, fundamentadas na literatura, quanto abordagens empíricas, baseadas em dados observacionais ou de mercado.

O método propõe construir uma taxonomia a partir de dimensões, que representam as características gerais do objeto de estudo e de atributos que correspondem aos possíveis valores dessas dimensões. O processo começa com a definição de uma metacaracterística, que orienta toda a construção, e de uma condição de parada, que determina quando o desenvolvimento deve ser encerrado. A metacaracterística garante o alinhamento conceitual, e a condição de parada pode ser, por exemplo, a ausência de novas ideias ou o alcance de uma taxonomia útil e completa. Após essa definição inicial, o processo pode seguir um dos seguintes caminhos:

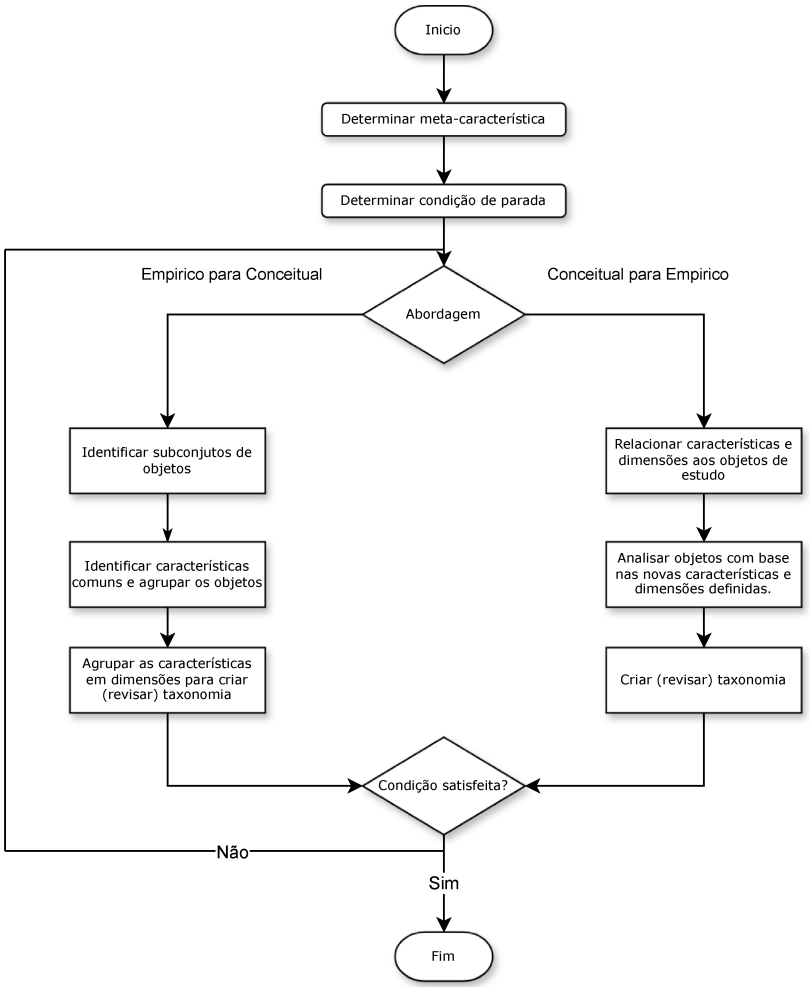
- Empírico para conceitual: Começa com a análise de dados reais. O pesquisador observa objetos, identifica características, agrupa em dimensões e propõe a taxonomia com base

nos padrões encontrados. Trata-se de uma abordagem indutiva.

- Conceitual para empírico: Tem como ponto de partida a literatura e o conhecimento teórico. O pesquisador define dimensões e atributos com base em estudos anteriores, aplica a estrutura aos objetos e ajusta conforme os resultados. Trata-se de uma abordagem dedutiva.

Esses ciclos iterativos são repetidos até que a condição de parada seja satisfeita. A cada nova iteração, a taxonomia é revisada e refinada, podendo alternar entre abordagens, de acordo com os dados e insights obtidos. A Figura 1 ilustra o processo iterativo de construção de taxonomias conforme proposto (NICKERSON *et al.*, 2013):

Figura 1 – Processo iterativo de desenvolvimento de taxonomias



Fonte: Adaptado de (NICKERSON *et al.*, 2013)

2.3 Ambiente de desenvolvimento Integrado - IDE

A literatura apresenta diferentes relatos sobre o surgimento dos ambientes de desenvolvimento integrado. Alguns autores apontam que, na década de 70, o ambiente Smalltalk, desenvolvido no Xerox Palo Alto Research Center (Xerox PARC), reunia editor, compilador e depurador em uma única interface, configurando-se como um precursor do conceito moderno de IDE (KAY, 1993). Essa integração permitia aos desenvolvedores escrever, executar e depurar código no mesmo ambiente, promovendo ganhos de produtividade e usabilidade.

Outras fontes destacam como marco importante o lançamento do *Turbo Pascal*[1], pela Borland, em 1983. Esse ambiente combinava editor e compilador integrados, possibilitando identificar e corrigir erros diretamente no código-fonte e acelerando o ciclo de desenvolvimento, o que contribuiu para a popularização da linguagem Pascal na época (BECK; GRAHAM, 1984).

No início da década de 1990, o *Visual Basic* (VB) [2], da Microsoft, realizou a combinação de programação visual e orientação a objetos em um único ambiente, representando um avanço no conceito de IDE ao integrar ferramentas de design gráfico e edição de código em uma única interface. Esta integração refletiu na programação e fez com que resultasse em um aumento na eficiência do desenvolvimento de software (NOLETO, 2021; LIMA, 2022). Entretanto, os IDEs não pararam de evoluir desde suas primeiras aparições. Ao longo dos anos, essas ferramentas continuaram a se aprimorar e incorporar novos recursos, buscando melhorar a experiência dos usuários e a eficiência no desenvolvimento de software.

Um ambiente de desenvolvimento integrado (IDE) é uma plataforma que reúne uma variedade de recursos e ferramentas destinados a facilitar o processo de desenvolvimento de software. Ele oferece uma interface unificada que combina funcionalidades como edição de código, compilação, depuração e testes, o que proporciona aos desenvolvedores um ambiente de trabalho completo e integrado. O objetivo principal de um IDE é agilizar o desenvolvimento de software, fornecendo recursos eficientes e automatizados, permitindo que os desenvolvedores concentrem-se na criação de soluções de software de forma mais produtiva e eficaz (SEBESTA, 2000). Um IDE também incorpora a técnica de Desenvolvimento Rápido de Aplicativos (RAD), que permite aos desenvolvedores tirar o máximo proveito do ambiente, desenvolvendo códigos com maior rapidez e facilidade. Essa abordagem visa aumentar a eficiência do processo de desenvolvimento e acelerar a entrega de soluções de software (CHAVES; SILVA, 2008;

[1] <https://turbopascal.org/>
[2] <https://learn.microsoft.com/pt-br/dotnet/visual-basic/>

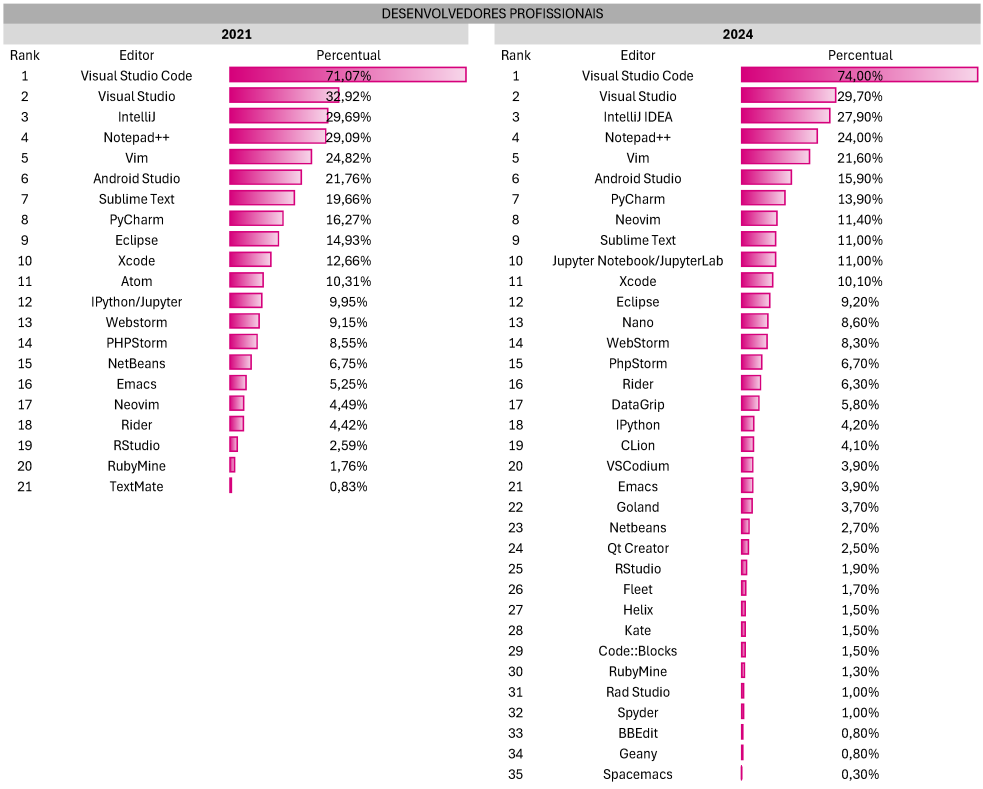
BEYNON-DAVIES *et al.*, 1999).

Há vários tipos e especializações de IDEs para atender às necessidades diversificadas dos desenvolvedores. Alguns são específicas para determinadas linguagens de programação, como *Eclipse*[3], um IDE gratuito que se destaca no ambiente *Java*[4], ou *PyCharm*[5], muito utilizado por programadores *Python*[6]. Outros são multiplataforma, como o *Visual Studio Code (VS Code)*[7], que oferecem suporte a várias linguagens e sistemas operacionais. Existem também IDEs direcionados a setores específicos, como *Android Studio*[8] e o *Xcode*[9], utilizados para o desenvolvimento de aplicativos Android e iOS, respectivamente.

Para entender a evolução da popularidade dos IDEs, foram consideradas duas pesquisas: a da *Stack Overflow* de 2021 e 2024 (STACKOVERFLOW, 2021; STACKOVERFLOW, 2024) e o *índice IDE* de 2025 (CARBONNELLE, 2025). Segundo dados da pesquisa da *Stack Overflow* de 2021, com respostas de 57.684 desenvolvedores e uma lista de 21 ferramentas distintas, o *VS Code* (71,07%) liderava a lista de ambientes de desenvolvimento mais utilizados, seguido por *Visual Studio*[10] (32,92%) e *IntelliJ IDEA*[11] (29,69%). Já na pesquisa de 2024, os resultados mantêm o *VS Code* como o IDE mais popular, agora com 74% de preferência entre os respondentes, seguido por *Visual Studio* (29,7%) e *IntelliJ IDEA* (27,9%). A lista completa apresenta um total de 35 ambientes distintos, o que reforça a diversidade de preferências entre os desenvolvedores. Entretanto, percebe-se que muitas das ferramentas citadas em 2021 permanecem presentes entre as mais utilizadas, evidenciando sua relevância contínua. A Figura 2 apresenta os resultados das pesquisas da *Stack Overflow* dos anos de 2021 e 2024, respectivamente.

[3] <https://www.eclipse.org/>
[4] <https://www.java.com/pt-BR/>
[5] <https://www.jetbrains.com/pt-br/pycharm/>
[6] <https://www.python.org/>
[7] <https://code.visualstudio.com/>
[8] <https://developer.android.com/studio?hl=pt-br>
[9] <https://developer.apple.com/xcode/>
[10] https://visualstudio.microsoft.com/
[11] https://www.jetbrains.com/

Figura 2 – Ambientes de desenvolvimento integrado mais utilizados em 2021 e 2024



DESENVOLVEDORES PROFISSIONAIS

2021			2024		
Rank	Editor	Percentual	Rank	Editor	Percentual
1	Visual Studio Code	71,07%	1	Visual Studio Code	74,00%
2	Visual Studio	32,92%	2	Visual Studio	29,70%
3	IntelliJ	29,69%	3	IntelliJ IDEA	27,90%
4	Notepad++	29,09%	4	Notepad++	24,00%
5	Vim	24,82%	5	Vim	21,60%
6	Android Studio	21,76%	6	Android Studio	15,90%
7	Sublime Text	19,66%	7	PyCharm	13,90%
8	PyCharm	16,27%	8	Neovim	11,40%
9	Eclipse	14,93%	9	Sublime Text	11,00%
10	Xcode	12,66%	10	Jupyter Notebook/JupyterLab	11,00%
11	Atom	10,31%	11	Xcode	10,10%
12	IPython/Jupyter	9,95%	12	Eclipse	9,20%
13	Webstorm	9,15%	13	Nano	8,60%
14	PHPStorm	8,55%	14	WebStorm	8,30%
15	NetBeans	6,75%	15	PhpStorm	6,70%
16	Emacs	5,25%	16	Rider	6,30%
17	Neovim	4,49%	17	DataGrip	5,80%
18	Rider	4,42%	18	IPython	4,20%
19	RStudio	2,59%	19	CLion	4,10%
20	RubyMine	1,76%	20	VSCodium	3,90%
21	TextMate	0,83%	21	Emacs	3,90%
			22	Goland	3,70%
			23	Netbeans	2,70%
			24	Qt Creator	2,50%
			25	RStudio	1,90%
			26	Fleet	1,70%
			27	Helix	1,50%
			28	Kate	1,50%
			29	Code::Blocks	1,50%
			30	RubyMine	1,30%
			31	Rad Studio	1,00%
			32	Spyder	1,00%
			33	BBEdit	0,80%
			34	Geany	0,80%
			35	Spacemacs	0,30%

Fonte: (STACKOVERFLOW, 2021; STACKOVERFLOW, 2024)

Complementando esse panorama, o índice IDE de 2025 que analisa os IDEs mais pesquisados via *Google Trends*, mostrou o *Visual Studio* como o ambiente mais pesquisado, seguido por *VS Code* e *Eclipse*. Abaixo, a Figura 3 apresenta os dados dessa pesquisa.

Figura 3 – Ambientes de desenvolvimento integrado mais pesquisados no Google

Worldwide, May 2025 :

Rank	Change	IDE	Share	1-year trend
1		Visual Studio	28,78%	1,50%
2		Visual Studio Code	15,24%	1,50%
3	↑	Android Studio	11,22%	1,30%
4		pyCharm	10,27%	-0,60%
5	↓	Eclipse	10,09%	-1,80%
6		IntelliJ	7,16%	-0,40%
7		NetBeans	3,56%	-0,50%
8		Xcode	2,97%	0,00%
9		RStudio	2,85%	-0,10%
10		Sublime Text	2,34%	-0,20%

Fonte: (CARBONNELLE, 2025)

Os dados da Figura 3 indicam que o *Visual Studio* lidera com 28,78% das buscas,

seguido por *VS Code* (15,24%), destacando a dominância dessas IDEs no mercado. Ferramentas como *pyCharm* e *Eclipse* ainda têm participação significativa, mas apresentaram queda no interesse. O crescimento do *Android Studio* e a estabilidade do *VS Code* sugerem que há uma maior demanda por desenvolvimento mobile e ambientes flexíveis. Já ferramentas como *NetBeans*, *Xcode* e *RStudio* aparecem com menor participação, indicando grupos específicos de uso.

A presença recorrente de algumas ferramentas nessas diferentes fontes reforça a importância de analisar não só sua popularidade, mas também suas características técnicas. Dessa forma, com o objetivo de aprofundar a compreensão sobre os ambientes mais utilizados e embasar a construção da taxonomia proposta, foi realizada uma interseção entre as ferramentas mencionadas nas pesquisas de 2021 e 2024 da *Stack Overflow*. A partir dessa lista foram consultadas as páginas e documentações oficiais de cada ferramenta para identificar suas principais funcionalidades. Essa análise permite observar quais características são mais comuns entre os ambientes que se mantêm relevantes ao longo do tempo, além de oferecer uma base concreta para a categorização dos IDEs segundo suas capacidades técnicas. É importante destacar que nem todas as ferramentas citadas são tecnicamente IDEs completas, algumas são editores de texto avançados que, com o uso de *plugins*, podem oferecer funcionalidades semelhantes às de um IDE. As Tabelas 1 e 2 apresentam o resultado dessa interseção com o mapeamento das funcionalidades:

Tabela 1 – Funcionalidades comuns das ferramentas presentes nas pesquisas de 2021 e 2024 - Parte 1

Ferramenta	**Compilador**	**Depurador**	**Realce de Sintaxe**	**Autocomp.**	**Git**
Visual Studio Code	X*	X*	X*	X*	X
Visual Studio	X	X	X	X	X
IntelliJ IDEA	X	X	X	X	X
PyCharm	X	X	X	X	X
Eclipse	X	X	X	X	X
Android Studio	X	X	X	X	X
Xcode	X	X	X	X	X
Sublime Text	X*		X	X*	X*
Vim	X*		X*	X*	X*
Notepad++	X*		X	X*	

Onde **X** são funcionalidades nativas da ferramenta, **X*** funcionalidades disponíveis por meio de plugins ou extensão e **Parcial** são funcionalidades disponíveis de forma limitada.

Embora a utilização dos IDEs traga diversos benefícios para os desenvolvedores,

Tabela 2 – Funcionalidades comuns das ferramentas presentes nas pesquisas de 2021 e 2024 - Parte 2

Ferramenta	**Multiling.**	**Extensões/Plugins**	**Terminal**	**Refat.**	**Snippets**
Visual Studio Code	X*	X	X	Parcial	X*
Visual Studio	X	X	X	X	X
IntelliJ IDEA	X	X	X	X	X
PyCharm	X	X	X	X	X
Eclipse	X	X	X	X	X
Android Studio	X	X	X	X	X
Xcode	Parcial	X	X	X	X
Sublime Text	X	X*	X*	X*	X
Vim	X	X*	X*	X*	X
Notepad++	X	X*	X*	X*	X*

Fonte: Elaborado pela autora

eles também podem apresentar algumas limitações. Alguns IDEs podem ser pagos, o que pode dificultar para aqueles que desejam utilizar ferramentas específicas, outros podem exigir elevados recursos de hardware, podendo impactar no desempenho em dispositivos menos potentes e, ao invés de agilizar no desenvolvimento do software, podem atrasar a execução das tarefas. A aprendizagem quanto à utilização das funcionalidades dessas ferramentas pode ser dificultosa para usuários iniciantes, podendo inicialmente gerar sobrecarga cognitiva e, consequentemente, maior tempo de aprendizado. Além disso, a personalização dos IDEs pode ser limitada ou complexa, em certos casos (NOLETO, 2021; BELITARDO, 2021).

3 METODOLOGIA

Neste capítulo, será apresentada a estrutura do trabalho a ser utilizada para realizar a pesquisa sobre a taxonomia de ambientes de desenvolvimento integrado. Este trabalho foi fundamentado no modelo proposto por (NICKERSON *et al.*, 2013) , no qual se estabelece um processo interativo e estruturado para a construção de taxonomias em sistemas de informação.

O capítulo está organizado nas seguintes seções: sobre tipo de pesquisa (Seção 3.1), construção da taxonomia (Seção 3.2) e método de coleta (Seção 3.3).

3.1 Tipo de pesquisa

Pesquisa aplicada, com abordagem exploratória e de natureza mista, estruturada em etapas conceituais e empíricas. Inicialmente, foram coletadas informações sobre as características de IDEs por meio de revisão da literatura e documentações oficiais, seguida por validação baseada em entrevistas com profissionais da área.

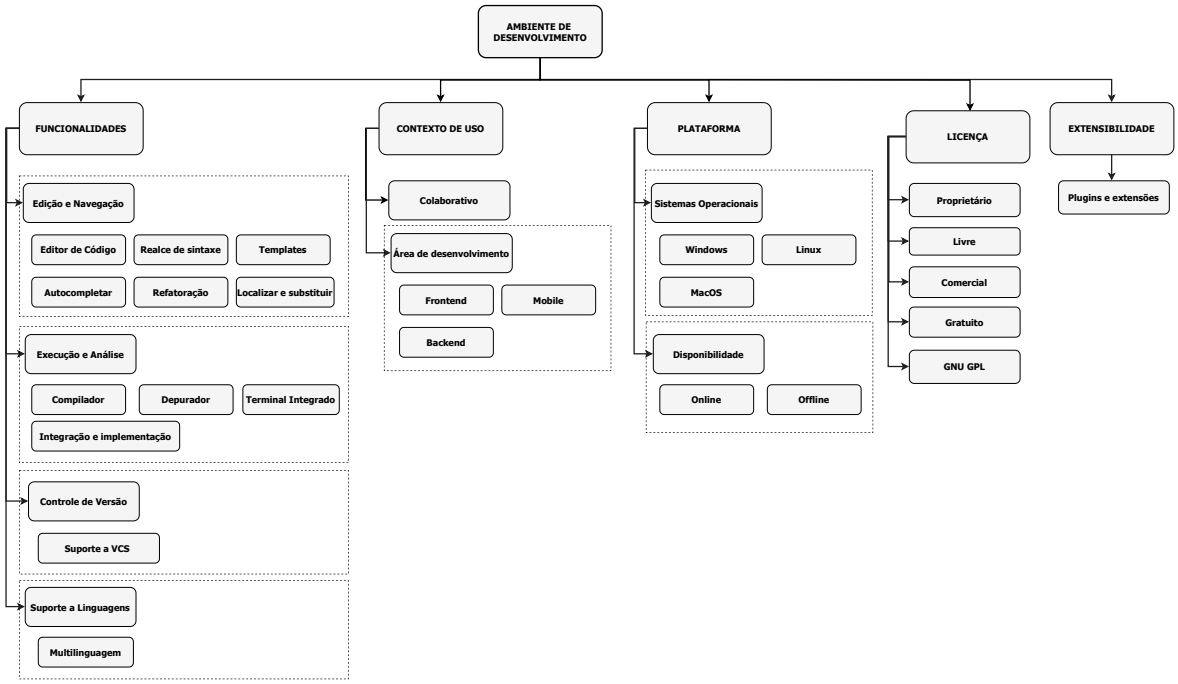
3.2 Construção da taxonomia

A construção da taxonomia seguiu a abordagem conceitual para empírica conforme descrita por (NICKERSON *et al.*, 2013). O processo iniciou com a definição da metacaracterística: "Entidades essenciais que caracterizam um ambiente de desenvolvimento integrado", que guiou toda a estrutura. Após a definição da metacaracterística, foram propostas as dimensões e atributos baseados na literatura, documentações e artigos técnicos. As entidades foram organizadas por categorias como: edição e navegação, execução e análise, controle de versão, suporte a linguagens, colaborativo, área de desenvolvimento, sistemas operacionais, disponibilidade, proprietário, livre, comercial, gratuito, GNU GPL e plugins e extensões.

A seguir será apresentada uma versão geral da estrutura da taxonomia proposta com a descrição das dimensões e suas respectivas categorias e subcategorias. Essa explicação tem como intuito detalhar como os elementos foram agrupados e quais entidades específicas compõem cada dimensão do ambiente de desenvolvimento integrado.

A estrutura está organizada em cinco grandes dimensões: funcionalidades, contexto de uso, plataforma, licença e extensibilidade, conforme representado na Figura 4.

Figura 4 – Versão proposta da taxonomia para IDE



Fonte: Elaborado pela autora

A dimensão **funcionalidades** reúne os recursos centrais de um IDE, organizados em subgrupos conforme seu propósito.

- **Edição e navegação:** essa categoria agrupa funcionalidades que otimizam a escrita e navegação do código;

  Editor de código: ferramenta básica de edição textual;

  Realce de sintaxe: colore elementos do código para facilitar a leitura;

  Templates: refere-se a trechos de código prontos que aceleram o desenvolvimento;

  Autocompletar: sugere comandos ou nomes durante a digitação de forma automática;

  Refatoração: permite reorganizar o código sem alterar seu comportamento;

  Localizar e substituir: busca e modifica trechos de código em larga escala.

- **Execução e Análise:** essa categoria refere-se a recursos ligados à compilação, testes e execução.

  Compilador: converte código-fonte para linguagem de máquina;

  Depurador: ajuda na identificação e correção de erros;

  Terminal integrado: executa comandos diretamente no IDE;

  Integração e implementação: inclui ferramentas de build, testes e deploy.

- **Controle de versão:** categoria que identifica ferramentas que oferecem recursos de versionamento.

  Suporte a VCS: integra sistemas como *Git*, permitindo versionamento e colaboração.
- **Suporte a Linguagens:** essa categoria classifica o IDE conforme linguagem específica ou não.

  Multilinguagem: indica suporte a múltiplas linguagens de programação.

A dimensão **contexo de uso** classifica o IDE conforme o ambiente ou foco de aplicação.

- **Colaborativo:** categoria que identifica ferramentas que oferecem recursos para trabalho em equipe.
- **Área de desenvolvimento:** categoria que subdivide os IDEs conforme o foco do desenvolvimento.

  Frontend: para interfaces e design;

  Backend: para lógica e banco de dados;

  Mobile: para apps iOS/Android.

A dimensão **plataforma** relaciona o IDE ao ambiente de execução e sistemas operacionais.

- **Colaborativo:**

  Windows, macOS, Linux: compatibilidade com OS específico.
- **Área de desenvolvimento:** subdivide conforme o foco do desenvolvimento.

  Online: executada via navegador;

  Offline: instalada localmente.

A dimensão **licença** agrupa os tipos de licenciamento da ferramenta

- **Proprietário:** software com código-fonte não disponível ou com restrições legais que impedem modificação, redistribuição ou uso sem autorização expressa;
- **Livre:** software com código-fonte acessível, permitindo uso, modificação e redistribuição, desde que respeitadas as condições da licença;
- **Comercial:** exige compra ou assinatura.
- **Gratuito:** distribuído sem custo ao usuário, mas que pode ter restrições de uso, modificação ou redistribuição;
- **GNU GPL:** licença livre com restrições de redistribuição.

A dimensão **extensibilidade** avalia a capacidade do ambiente ser expandido por meio de recursos adicionais.

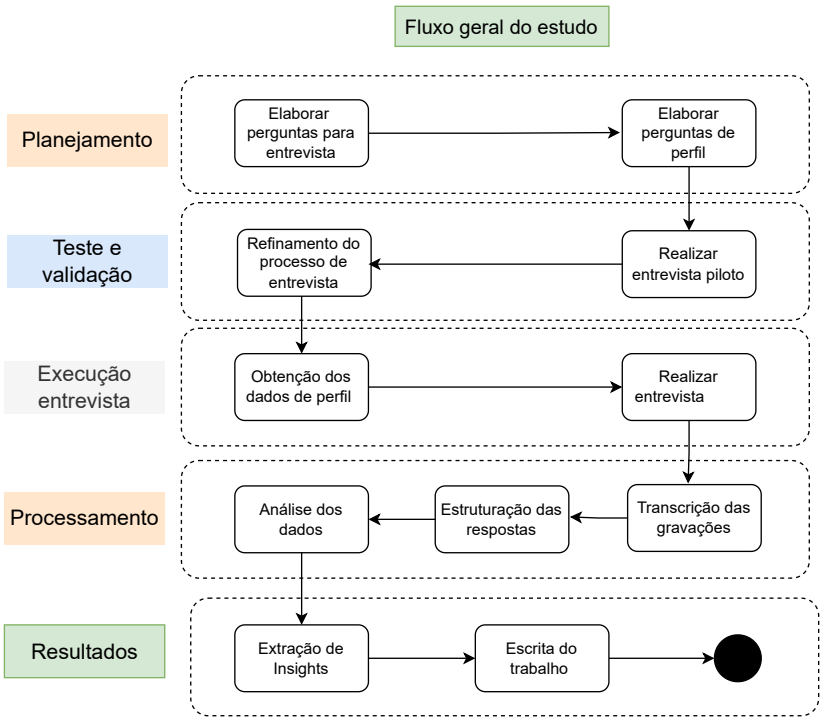
- **Plugins e extensões:** capacidade de adicionar novos recursos ao IDE via pacotes externos.

3.3 Método de coleta

3.3.1 Etapas de condução do estudo

O processo de entrevista foi estruturado em cinco etapas: planejamento, teste e validação, execução da entrevista, processamento e resultados, conforme apresenta a Figura 5. No primeiro momento, foi realizado um planejamento das questões a serem abordadas pelo entrevistador. Em seguida, em teste e validação, foi realizada uma entrevista piloto e um refinamento do método de entrevista. Posteriormente, foi realizada a execução da entrevista, sendo composta de um primeiro questionário para obter os dados de perfil e, em seguida, a entrevista com os candidatos. Finalizado o processo de obter os dados, iniciou-se a etapa de processamento, sendo dividida na parte de transcrição das gravações, estruturação das respostas em um formato de tabela e na análise de dados. Por fim, vem a etapa de resultados que consiste na obtenção dos insights e na escrita do trabalho.

Figura 5 – Fluxo de condução do estudo



Fonte: Elaborado pela autora

Nos próximos tópicos serão apresentados de forma detalhada os processos executados em cada etapa. Na (Seção 3.3.2) são apresentadas as perguntas elaboradas aos entrevistados. Na

(Seção 3.3.3) é detalhada a criação do formulário de perfil e sua necessidade. Na (Seção 3.3.4) é demonstrado como foi o processo da entrevista piloto para o aprimoramento do instrumento de coleta. Já na (Seção 3.3.5) é mostrado como ocorreu a execução das entrevistas. Na (Seção 3.3.6) são descritos os passos para processamento dos dados. Por fim, na (Seção 3.3.7) é abordada a análise realizada nos dados.

3.3.2 Definições das perguntas de entrevista

Na primeira etapa, foi definido um conjunto de perguntas com o intuito de identificar a percepção dos entrevistados em relação às principais funcionalidades em um IDE. A escolha das perguntas foi baseada nos principais pontos discutidos na literatura abordada neste estudo. Ao todo, foram formuladas seis perguntas, com o objetivo de captar a opinião de profissionais da área, permitindo uma análise mais precisa das funcionalidades essenciais de um IDE.

P1: Na sua experiência, quais são os critérios que você considera ao escolher uma IDE? Essa pergunta busca compreender os principais fatores que influenciam na escolha de um ambiente de desenvolvimento, sejam eles pessoais, técnicos ou contextuais.

P2: Como você percebe a evolução dos IDEs ao longo do tempo? Houve mudanças importantes nas que você usa ou usava? Essa pergunta busca captar a percepção sobre como os IDEs têm se transformado, e quais aspectos são percebidos como melhorias ou mudanças relevantes.

P3: Você acredita que existe um conjunto comum de características que toda IDE deveria ter? O objetivo é entender se o entrevistado tem uma visão consolidada sobre elementos fundamentais de um IDE contribuindo para avaliar a metacaracterística da taxonomia.

P4: Ao analisar esta taxonomia que estou propondo, ela representa bem os principais aspectos que você observa em uma IDE? Aqui, o objetivo é validar a estrutura da taxonomia do ponto de vista prático, entendendo se ela reflete a experiência do usuário com IDEs.

P5: Alguma parte da taxonomia parece desnecessária, incompleta ou mal organizada? Essa pergunta tem como objetivo identificar pontos de melhoria na estrutura proposta e captar sugestões para reorganização ou exclusão de categorias.

P6: Há alguma categoria ou conceito que você incluiria na taxonomia? Essa pergunta tem como intuito explorar possíveis contribuições adicionais para o refinamento da estrutura, com base na experiência prática do entrevistado.

3.3.3 Criação de formulário de perfil

Com o intuito de agilizar a condução das entrevistas e de ajudar a descrever a amostra, foi elaborado um formulário específico para a extração do perfil dos entrevistados. Essa etapa visa coletar informações como cargo, tempo de experiência, área de desenvolvimento e preferências para caracterizar e compreender o contexto profissional dos entrevistados. As perguntas foram elaboradas e disponibilizadas via *google forms*[1], sendo este composto de perguntas abertas e de múltipla escolha, como mostra a Tabela 3.

Tabela 3 – Formulário de perfil do entrevistado

Identificador	**Pergunta**
1	Nome
2	Área de atuação
3	Função ou cargo atual
4	Quantos anos de experiência na área de desenvolvimento ou áreas afins? [ ] Menos de 1 ano [ ] 1 a 3 anos [ ] 4 a 5 anos [ ] Mais de 5 anos
5	IDEs que utiliza ou já utilizou com frequência
6	Qual(is) linguagem(ns) de programação você costuma usar com essas ferramentas?

Fonte: Elaborado pela autora

3.3.4 Entrevista piloto

Antes da condução das entrevistas, foi realizada uma entrevista piloto com um participante com experiência de mais de 4 anos em desenvolvimento de software. O objetivo dessa etapa foi validar o modo de execução da entrevista, avaliando fatores como clareza das perguntas, dinâmica, tempo de duração e qualidade dos dados coletados. Com a realização da entrevista piloto, foi possível obter uma estimativa inicial de tempo, sendo observada uma duração de aproximadamente 20 minutos para cada entrevista, além de realizar ajustes no sentido da pergunta P5 do formulário de perfil do participante, deixando-a mais clara para o entrevistado e objetiva para o estudo.

[1] <https://docs.google.com/forms>

3.3.5 Realização das entrevistas

Após o preenchimento do formulário de perfil do participante, enviado via e-mail, foi marcado o agendamento das entrevistas, que foram conduzidas de forma virtual. Ao todo participaram sete profissionais, sendo três desenvolvedores de software, um engenheiro de dados, um *analitics engineer*, um analista de infraestrutura e um analista de planejamento. Para esta etapa, foi utilizada a ferramenta *Google Meet*[2] para a realização das entrevistas online e o software *OBS Studio*[3] para a gravação das mesmas.

Com base na entrevista piloto, destacada anteriormente, foi enviado um convite por e-mail para cada participante, com duração prevista de 20 minutos. Ao iniciar cada entrevista, os participantes foram informados sobre a confidencialidade das informações e o propósito educativo da pesquisa.

3.3.6 Processamento dos dados

O processamento dos dados foi dividido em duas etapas: transcrição dos áudios e estruturação dos dados.

Na primeira etapa cada entrevista foi transcrita usando a ferramenta *Gemini*[4], da *Microsoft*, integrada à plataforma *Google meet*. Posteriormente, as transcrições passaram por uma etapa de revisão manual, com o objetivo de corrigir eventuais erros e garantir maior fidelidade ao conteúdo original das falas.

Na etapa de estruturação, foi realizada uma limpeza e organização das respostas em um formato tabular, resultando em um arquivo no formato CSV. Como a pesquisa abrange uma amostragem reduzida, optou-se por esse processo manual de estruturação, o qual possibilitou um contato mais direto com os dados. Essa abordagem favoreceu a compreensão das respostas abertas e contribuiu para uma interpretação mais precisa e qualificada das falas dos participantes.

3.3.7 Análise dos dados

Para a análise dos dados, foi utilizado o ambiente *Google Colab*[5], que é um serviço gratuito da *Google* que permite desenvolver notebooks em linguagem *Python*[6] sem necessidade

[2] <https://meet.google.com/>
[3] <https://obsproject.com/>
[4] https://gemini.google/about/?hl=pt-BR
[5] https://colab.google/
[6] https://www.python.org/

de configuração, com recursos de computação e acessível através de um navegador web. Além disso, foram utilizadas bibliotecas complementares para obtenção e visualização dos resultados: a *NLTK*[7], usada para a remoção de *stopwords*, a *WordCloud*[8] para a criação de nuvens de palavras, sendo este o método utilizado como análise qualitativa para verificar os termos mais recorrentes respondidos pelos participantes e a *matplotlib*[9] para criação de gráficos de frequência e demais visualizações de dados quantitativos.

[7] https://www.nltk.org/
[8] https://pypi.org/project/wordcloud/
[9] https://matplotlib.org/

4 RESULTADOS

Este capítulo apresenta os principais resultados obtidos ao longo do desenvolvimento do trabalho. Na Seção 4.1, são descritos os resultados do processo de validação da taxonomia por meio de entrevistas com profissionais da área. Esta seção está dividida em duas partes: (i) os principais achados da pesquisa, com base nas respostas dos participantes e (ii) o refinamento da taxonomia a partir das sugestões e contribuições obtidas nas entrevistas, resultando na versão final da taxonomia.

4.1 Resultado da pesquisa

4.1.1 Perfil dos participantes

Conforme destacado na Seção 3.3, além do processo de entrevista, foi criado um formulário de perfil do entrevistado, com o intuito de coletar as características de cada participante. A pesquisa contou com a participação de profissionais de diversas áreas, sendo todos profissionais com experiência de mercado. A Tabela 4 apresenta o perfil do entrevistado, onde cada linha representa um participante, sendo a primeira coluna o identificador do entrevistado, seguido pela área de atuação, cargo profissional e tempo de experiência no mercado.

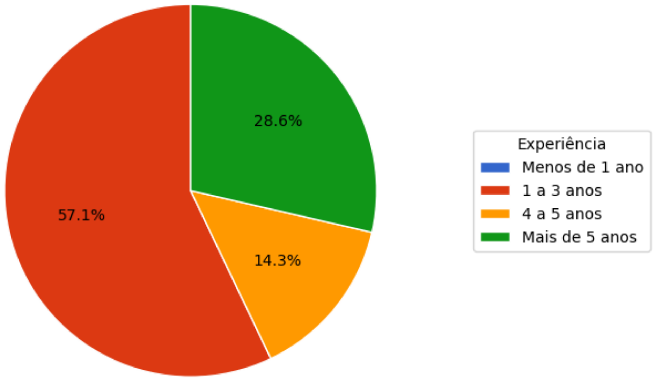
Tabela 4 – Formulário de perfil do entrevistado

ID	Área	Cargo	Experiência
01	Desenvolvimento de Software	Eng. de Software	4 a 5 anos
02	Desenvolvimento de Software	Dev. Fullstack	1 a 3 anos
03	Desenvolvimento de Software	Dev. de Software	mais de 5 anos
04	Dados	Analytics Engineer	1 a 3 anos
05	Dados	Eng. de Dados	1 a 3 anos
06	Infraestrutura de TI	Analista Infraestrutura	1 a 3 anos
07	PPCP + Int. de Dados	Analista Planejamento	mais de 5 anos

Fonte: Elaborado pela autora

A Figura 6 representa o percentual de experiência dos profissionais que participaram da entrevista, sendo evidente que a pesquisa abrangeu profissionais experientes, sendo todos com atuação há mais de um ano de mercado e mais de 28% atuam a mais de cinco anos no mercado. Evidenciando um público com considerável vivência na área e trajetórias consolidadas, conferindo uma maior credibilidade das respostas.

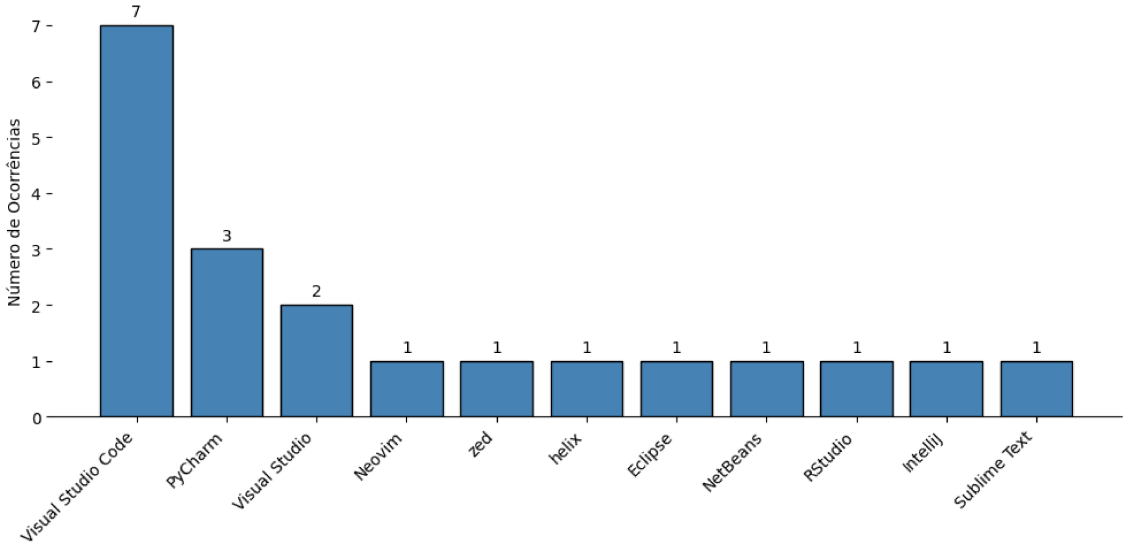
Figura 6 – Tempo de experiência dos entrevistados



Fonte: Elaborado pela autora

A Figura 7 mostra os principais ambientes de desenvolvimento utilizados pelos profissionais entrevistados, evidenciando que determinadas ferramentas são amplamente preferidas no mercado. Entre elas, destacam-se, *Visual Studio Code*, *PyCharm* e *Visual Studio*. A presença desses IDEs permite identificar padrões que se relacionam com as características previstas na taxonomia proposta, além de serem amplamente presentes no mercado, como mostrou a pesquisa destacada na Seção 2.3

Figura 7 – Ambientes utilizados pelos entrevistados



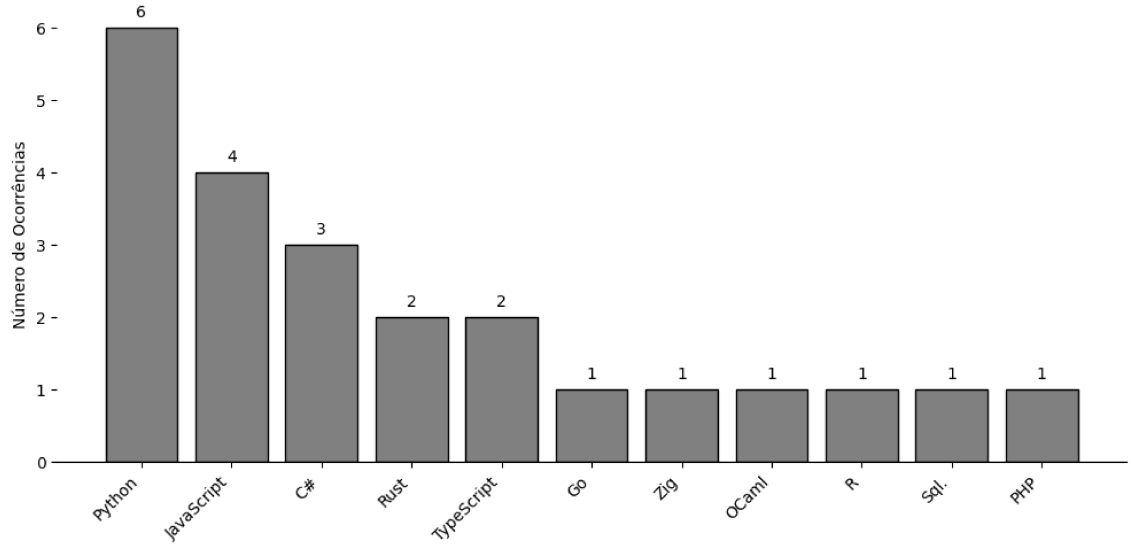
Fonte: Elaborado pela autora

Complementando a análise das ferramentas de desenvolvimento utilizadas pelos participantes, também foi investigado quais linguagens de programação são mais utilizadas em seus fluxos de trabalho. A diversidade de linguagens citadas evidencia a relevância da característica multilinguagem, já destacada na taxonomia proposta, e reforça a necessidade de

que os IDEs ofereçam suporte a diferentes ecossistemas.

A Figura 8 apresenta a distribuição das linguagens citadas pelos entrevistados. Observa-se que, assim como nos resultados obtidos sobre IDEs, existe um núcleo de tecnologias amplamente dominantes e um conjunto de ferramentas de uso mais restrito, porém importante para contextos especializados. Essa variedade confirma que a escolha de um IDE frequentemente está associada ao ecossistema de linguagens com o qual o profissional trabalha, fortalecendo o argumento de que o suporte a múltiplas linguagens é um critério decisivo na seleção dessas ferramentas.

Figura 8 – Linguagens de programação utilizadas pelos entrevistados



Fonte: Elaborado pela autora

4.1.1.1 P1: Na sua experiência, quais são os critérios que você considera ao escolher um IDE?

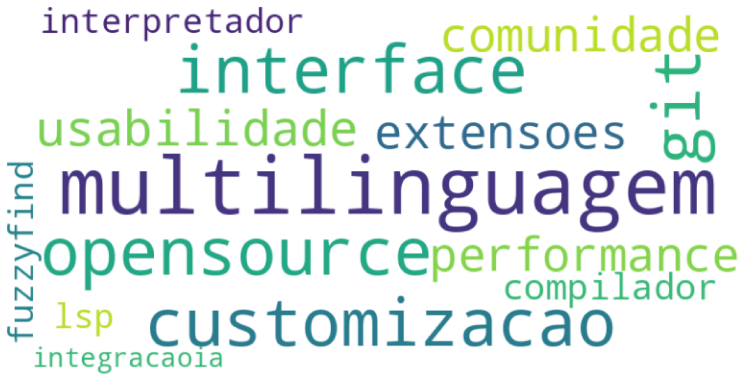
A Figura 9 ilustra os principais critérios destacados pelos participantes ao escolher um IDE. A análise realizada a partir das respostas dos entrevistados permitiu identificar os critérios mais recorrentes por meio de uma nuvem de palavras.

Ao todo foram mencionados 14 critérios distintos, entre os mais mencionados estão: multilinguagem, indicando que a compatibilidade com diversas linguagens continua sendo diferenciais valorizados. Em seguida, surge o open source, reforçando a preferência por ferramentas gratuitas. A customização e interface foram outros critérios amplamente citados, abrangendo tanto aspectos visuais quanto funcionais das ferramentas.

Além disso, os entrevistados destacaram a importância da integração com ferramentas externas, como Git e a presença de extensões que ajudam na produtividade, pois permitem ampliar as funcionalidades das ferramentas de acordo com as necessidades do projeto. Também houve destaque para a preferência de IDE com recursos integrados, como compilador e interpretador, mostrando que a centralização de funcionalidades em um único ambiente contribui para um fluxo de trabalho mais eficiente.

Os entrevistados ressaltaram que ferramentas com apoio da comunidade, boa documentação e funcionalidades como Language Server Protocol (LSP), fuzzy find e integração com IA são pontos importantes na escolha da ferramenta, pois contribuem para uma produção mais precisa e abrangente.

Figura 9 – Nuvem de palavras da análise da pergunta P1



Fonte: Elaborado pela autora

4.1.1.2 P2: Como você percebe a evolução dos IDEs ao longo do tempo? Houve mudanças importantes nas que você usa ou usava?

A Figura 10 ilustra as principais palavras e termos mencionados pelos participantes ao refletirem sobre a evolução dos ambientes de desenvolvimento integrado ao longo do tempo. Durante a entrevista, os profissionais destacaram uma evolução significativa nas ferramentas, que passaram de editores simples a plataformas mais completas e adaptáveis, como mostra nuvem de palavras.

Dentre os termos mais comentados, destaca-se as extensões, que tornam os IDEs mais componetizadas, como citado em uma das entrevistas. Essa modularização foi apontada como principal ponto de evolução, pois permite que os desenvolvedores escolham apenas os recursos necessários, melhorando a experiência e otimizando a performance.

Outro ponto relevante que a nuvem também revela é a crescente adoção de Inteligência Artificial, trazendo recursos como autocompletar inteligente, dicas de código e até documentação automática. Essas funcionalidades foram apontadas como um importante avanço, impactando diretamente na produtividade, pois reduzem o esforço cognitivo e aumentam a eficiência no dia a dia. Além disso, foi destacada a evolução no desempenho e na estabilidade dos IDEs, sendo cada vez mais capazes de executar grandes projetos. Também houve menções sobre a capacidade de um IDE acompanhar as tendências do mercado.

A nuvem mostra um vocabulário variado sobre à evolução dos IDEs. Os participantes destacaram que a evolução desses ambientes vão além da simples adição de novas funcionalidades, também envolve adaptação, integração e personalização. Esses aspectos reforçam a relevância de critérios como extensibilidade, colaboração e versionamento, já contemplados na taxonomia proposta.

Figura 10 – Nuvem de palavras da análise da pergunta P2

Fonte: Elaborado pela autora

4.1.1.3 *P3: Você acredita que existe um conjunto comum de características que toda IDE deveria ter?*

A Figura 11 ilustra as principais palavras e termos mencionados pelos participantes ao responderem à pergunta 3. Essa pergunta investigou a existência de um conjunto comum de características essenciais a qualquer IDE.

Entre os termos mais recorrentes na nuvem de palavras, destacam-se as funcionalidades de depurador, terminal embutido, autocompletar, realce de sintaxe e versionamento, sendo essas características de grande importância para uma melhor produtividade e desempenho das atividades profissionais.

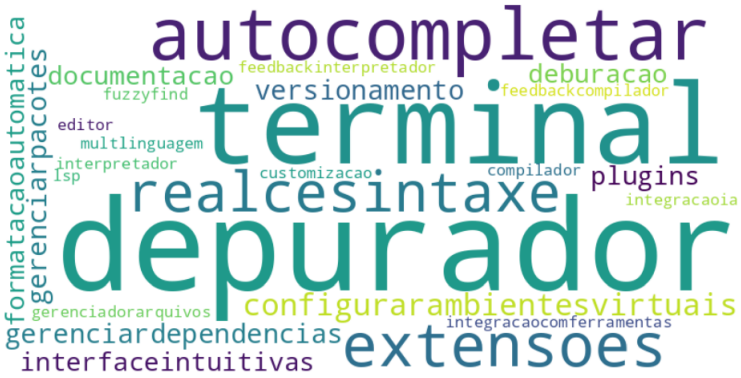
Outro ponto bastante destacado e relevante pela maioria dos participantes foi a capacidade de customização, como modificar temas e adaptar o ambiente conforme o estilo de trabalho individual.

Outro fator destacado foi uso de extensões, pois possibilita utilizar recursos extras, como reconhecimento de linguagens e sintaxes, além de integrar o IDE a outras ferramentas.

A criação de ambientes virtuais também foi mencionada como recurso necessário, pois permite um controle mais eficaz sobre bibliotecas e dependências.

Por fim, os entrevistados ressaltaram a importância de interfaces intuitivas, considerando que esse fator auxilia na curva de aprendizado, na resolução de problemas e na manutenção do ambiente de desenvolvimento.

Figura 11 – Nuvem de palavras da análise da pergunta P3



Fonte: Elaborado pela autora

4.1.1.4 P4: Ao analisar esta taxonomia que estou propondo, ela representa bem os principais aspectos que você observa em um IDE?

A pergunta 4 foi elaborada com o objetivo de validar a estrutura da taxonomia do ponto de vista prático de cada entrevistado. Após a apresentação da taxonomia proposta, todos os participantes concordaram que a taxonomia representa os principais aspectos observados em um IDE. Os participantes reconheceram que as funcionalidades apresentadas refletem os critérios utilizados no momento da escolha ou análise de um IDE.

4.1.1.5 P5: Alguma parte da taxonomia parece desnecessária, incompleta ou mal organizada?

Os entrevistados afirmaram que a taxonomia proposta está bem estruturada e não apresenta elementos desnecessários. A organização foi avaliada como clara, sem menções

significativas a redundância ou ausências críticas.

Houve uma observação referente a categoria *colaborativo* dentro da dimensão *contexto de uso*, com uma dúvida sobre o significado do termo, se ele se refere a edição de código por múltiplos desenvolvedores em tempo real ou se o termo possuía um escopo mais amplo.

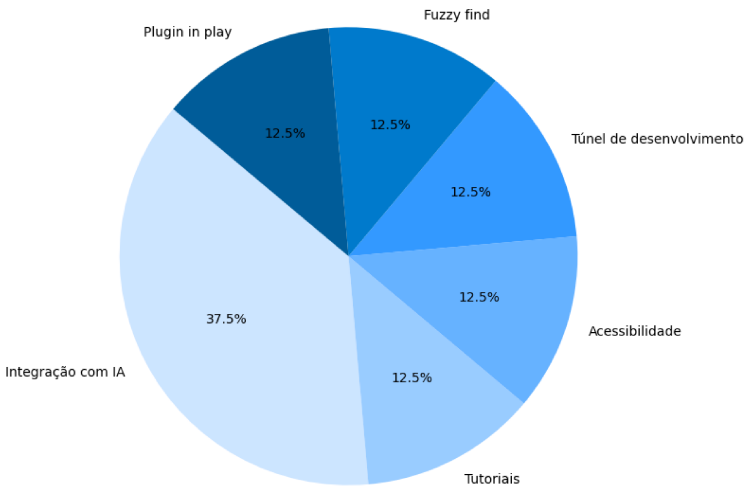
Essa dúvida sugere que a categoria pode ter uma descrição mais precisa, a fim de evitar interpretações ambíguas, especialmente considerando as diferentes formas de colaboração em um ambiente de desenvolvimento.

Esse e outros ajustes são apresentados na Figura 13, presente na Seção 4.1.2, dedicada ao refinamento da taxonomia.

4.1.1.6 P6: Há alguma categoria ou conceito que você incluiria na taxonomia?

A Figura 12 apresenta as sugestões mencionadas pelos participantes em relação a possíveis aprimoramentos na taxonomia proposta. As sugestões foram identificadas a partir da análise qualitativa das entrevistas e representam pontos considerados relevantes para ajustar a taxonomia às demandas atuais dos profissionais.

Figura 12 – Análise da pergunta P6 - sugestão de melhoria para a taxonomia



Fonte: Elaborado pela autora

A sugestão mais recorrente foi a integração com Inteligência Artificial (IA), mencionada por 37,5% dos participantes. Apesar da possibilidade de ser possível utilizar via extensões, os participantes destacaram a importância crescente de IA como funcionalidade nativa em IDEs modernas, citando o ambiente de desenvolvimento *Cursor*[1] que já é integrado à IA, justificando sua inclusão como item próprio na taxonomia.

As demais sugestões aparecem com frequência equivalente a (12,5%) refletindo aspectos complementares à estrutura proposta.

- Fuzzy find: também conhecida como busca inteligente, é citada como fundamental para agilidade na navegação entre arquivos e conteúdos.
- Plugin in play: ativação de um plugin sem a necessidade da reinicialização do IDE, ou seja, reforço da importância de suporte a extensões;
- Túnel de desenvolvimento: recomendação voltada principalmente ao desenvolvimento mobile;
- Tutoriais integrados: recursos informativos para facilitar o uso do IDE, especialmente por inciantes.
- Acessibilidade: funcionalidades como navegação por teclado, promovendo inclusão.

Esses pontos destacam a sugestão dos participantes em tornar os IDEs mais adaptáveis e produtivos, alinhando a taxonomia a um cenário de constante evolução.

As sugestões de melhoria para a taxonomia, mencionadas nessa pergunta, serão discutidas na seção a seguir.

4.1.2 Refinamento da taxonomia

Durante a etapa de validação da taxonomia, foi realizada a pergunta: "há alguma categoria ou conceito que você incluiria na taxonomia?". A partir das respostas, surgiram sugestões como: Integração com Inteligência Artificial (IA), plugin in play, tutoriais integrados, túnel de desenvolvimento e acessibilidade.

A primeira sugestão, sendo a mais recorrente encontrada durante as entrevistas e resultados, refere-se à inclusão da funcionalidade de integração com IA. Essa proposta reflete uma tendência emergente no mercado de desenvolvimento de software, em que IDEs passam a incorporar recursos de IA para apoiar a escrita e a revisão de código, sugerir otimizações, automatizar tarefas e detectar problemas de forma proativa. Embora os participantes tenham

[1] https://cursor.com/

mencionado que essa funcionalidade ainda não é amplamente consolidada entre as ferramentas atuais, a frequência com que foi citada e o crescente movimento de integração observado em soluções recentes justificam sua incorporação como uma nova dimensão da taxonomia. Assim, foi criada a dimensão Integração com Inteligência Artificial, dedicada a funcionalidades que utilizam algoritmos e modelos de IA para potencializar o trabalho do desenvolvedor.

Em relação ao plugin in play, embora relevante, seus aspectos já são contemplados na categoria extensões e plugins, encontrados na dimensão extensibilidade, tornando desnecessária a inclusão.

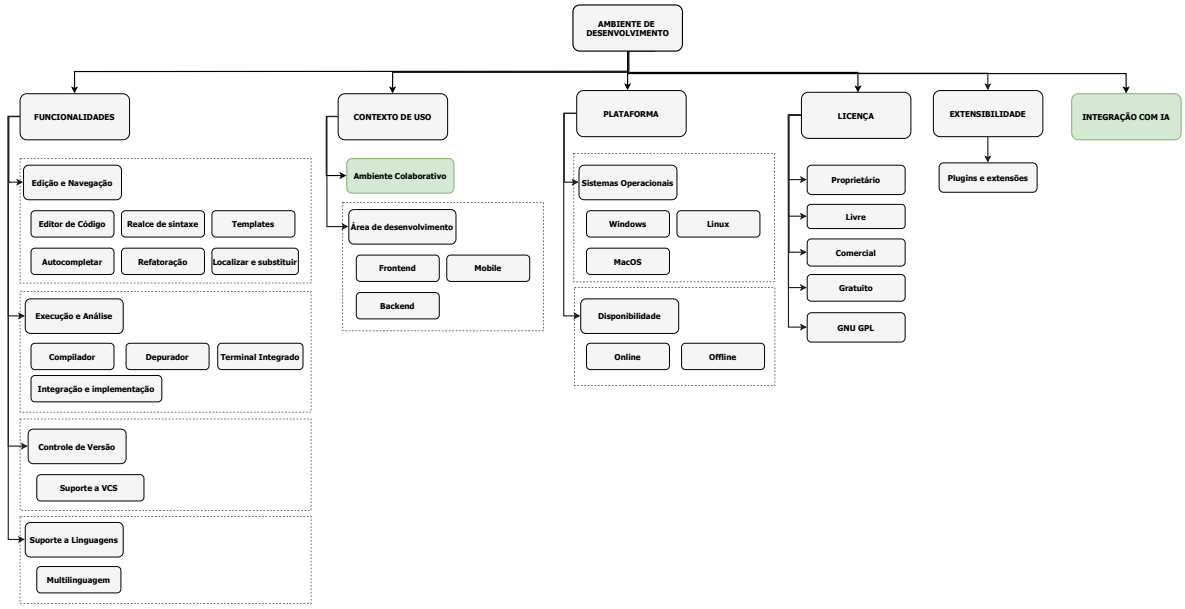
Os tutoriais integrados estão relacionando a experiência de aprendizado, não sendo um elemento amplamente presente entre os IDEs. O mesmo acontece para o túnel de desenvolvimento, sendo uma funcionalidade associada a um preview do projeto de forma remota, tratando-se de um diferencial específico, não necessariamente sendo um componente estruturante.

Em relação à sugestão de acessibilidade, observou-se que seu uso ainda é limitado, não sendo padrão entre as ferramentas avaliadas. Com isso, optou-se por não incluí-la na taxonomia neste momento. Entretanto, é válido ressaltar a crescente presença de recursos de acessibilidade entre as ferramentas, possibilitando um possível estudo futuro abordando esse aspecto.

Outro ponto evidenciado nas entrevistas virtuais foram as observações espontâneas por parte dos participantes, a fim de ampliar a compreensão sobre o estudo. Sendo um dos aspectos mencionados o entendimento sobre a categoria *colaborativo*, pertencente à dimensão *contexto de uso* da taxonomia, conforme discutido na Seção 4.1.1.5. Para evitar ambiguidade, optou-se pela renomeação da categoria para *ambiente colaborativo*, especificando que essa categoria está relacionada à colaboração em equipe.

A Figura 13 apresenta a versão final da taxonomia para ambientes de desenvolvimento integrado, contendo as modificações discutidas. A estrutura mantém a organização por dimensões principais, agora incluindo a dimensão Integração com Inteligência Artificial e a categoria renomeada *ambiente colaborativo*, destacada na imagem. As demais sugestões de funcionalidades foram mantidas fora da estrutura, conforme justificativas apresentadas anteriormente.

Figura 13 – Versão final da taxonomia para IDE



Fonte: Elaborado pela autora

4.2 Exemplos de uso da taxonomia

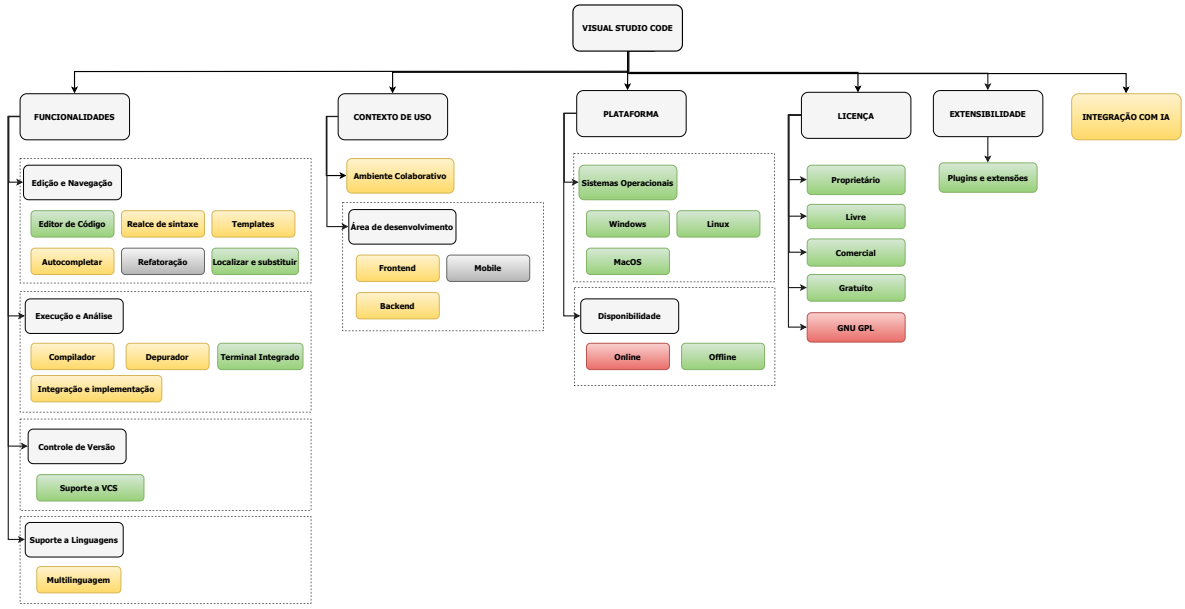
Para ilustrar a aplicabilidade da taxonomia proposta, nesta seção são apresentados dois exemplos baseados nos ambientes de desenvolvimento mais citados pelos entrevistados presentes na Seção 4.1.1, Figura 7 , sendo eles: *Visual Studio Code* e *Pycharm*. As Figuras 14 e 15 apresentam o uso da taxonomia para esses IDEs.

É válido destacar que por mais que a documentação do *Visual Studio Code* denomine-o como um editor de código-fonte, sua arquitetura expansível permite que torne-se um IDE completo, por meio do uso de extensões. Dessa forma, sua inclusão foi considerada nessa análise, pelo uso notório relatado pelos entrevistados e evidenciado nas pesquisas.

Os recursos estão dispostos por cores, indicando se são nativos (verde), disponíveis via extensão (amarelo), parcialmente suportados (cinza escuro) ou não suportados (vermelho).

É importante destacar que alguns IDEs adotam modelos híbridos, combinando software livre e proprietário. O núcleo pode ser aberto, mas a versão oficial inclui componentes proprietários. Embora gratuitas, frequentemente estão associadas a serviços ou versões comerciais. Por isso, podem ter classificações simultâneas na categoria licença.

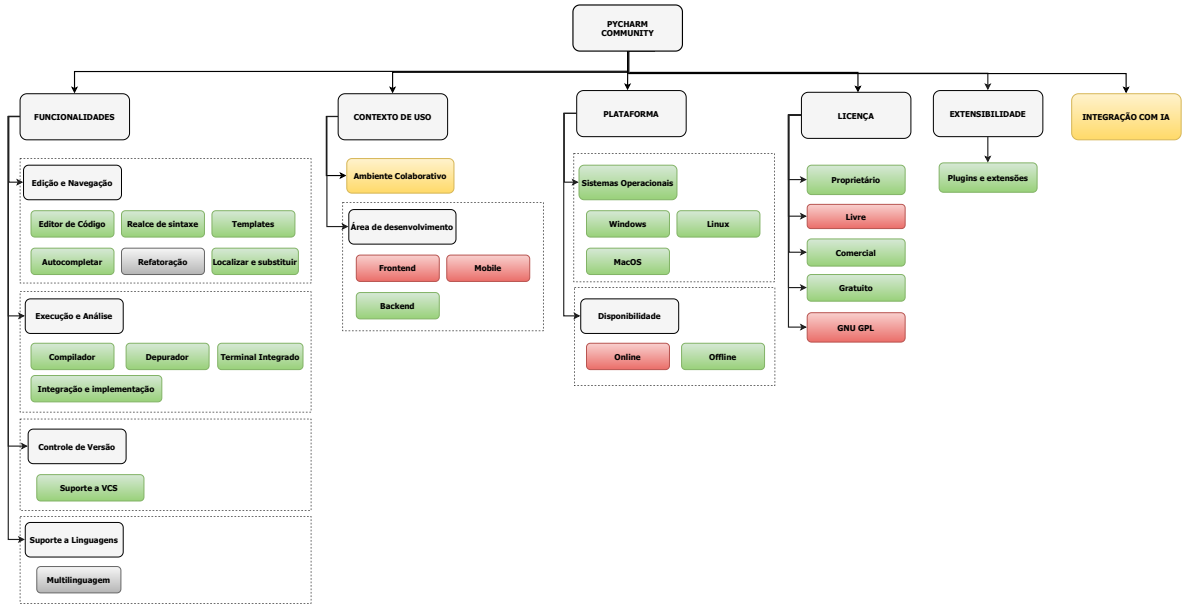
Figura 14 – Aplicação da taxonomia ao *Visual Studio Code*



Fonte: Elaborado pela autora

Os itens destacados em amarelo representam características presentes no *Visual Studio Code*, mas que necessitam de extensões para seu completo funcionamento. Os itens em cinza escuro, como refatoração e mobile, indicam os recursos que, mesmo utilizando extensões, ainda permanecem limitados ou, no caso do mobile, exigem configurações. Sendo assim, classificados como parciais.

Figura 15 – Aplicação da taxonomia ao *Pycharm (community)*



Fonte: Elaborado pela autora

A ferramenta *Pycharm* oferece uma variedade de funcionalidades nativas. Entretanto, sua versão gratuita possui limitações quanto ao suporte à multilinguagem, mesmo com o uso de extensões. Sendo assim, classificada como parcialmente suportada.

Os exemplos de uso apresentados demonstram o funcionamento do uso da taxonomia, evidenciando sua utilização em cenários práticos. Permite descrever, analisar e comparar diferentes ambientes de desenvolvimento integrado, tornando-a uma ferramenta de apoio à decisão para profissionais, acadêmicos e professores que necessitam selecionar ferramentas adequadas à sua necessidade.

5 TRABALHOS RELACIONADOS

O estudo (ANDRADE *et al.*, 2020) propõe a construção de uma taxonomia para classificar os atributos interpretáveis associados aos dados de trajetórias. O estudo foi conduzido a partir de quatro etapas principais: levantamento dos atributos interpretáveis utilizados em diferentes domínios de trajetórias, coleta dos dados de trajetória que serviram de base para a definição da taxonomia, etapa de geração dos atributos calculados e por último, a etapa de avaliação da taxonomia, conforme a metodologia proposta. Com isso, o estudo resultou no desenvolvimento de uma taxonomia robusta e aplicável a trabalhos de aprendizado de máquina no contexto de trajetórias, além de servir como base de referência para pesquisas na área.

Em (JUNIOR *et al.*, 2017), é abordada a construção de uma taxonomia para a classificação dos atores relacionados à corretagem em Infraestrutura como Serviço (IaaS). O estudo foca na migração de recursos virtuais entre provedores IaaS e tem como objetivo auxiliar os clientes na seleção de soluções para o gerenciamento de mídias computacionais. A construção da taxonomia foi fundamentada em referências da literatura sobre computação em nuvem, bem como em artigos e documentações técnicas. Como resultado, foi proposto um modelo que classifica os corretores de nuvem de acordo com o tipo de serviço prestado e a área de atuação no ambiente de nuvem computacional.

O estudo de (USMAN *et al.*, 2017a) realizou um mapeamento sistemático sobre o uso de taxonomias na Engenharia de Software e revisou um método de desenvolvimento de taxonomias. A busca inicial retornou 1.517 resultados únicos, dos quais 270 estudos primários foram selecionados. Os resultados indicam um interesse crescente na publicação e utilização de taxonomias na área, especialmente nas subáreas de construção, projeto, requisitos e manutenção. O trabalho evidencia a relevância e o potencial de aplicação das taxonomias, mas aponta que a maioria é desenvolvida de forma ad hoc, com procedimentos pouco descritos, e que raramente são estendidas ou revisadas.

A Tabela 5 apresenta uma síntese dos trabalhos relacionados, selecionados por abordarem propostas de taxonomias ou classificações em contextos distintos. Cada linha corresponde a um dos estudos analisados, enquanto as colunas destacam aspectos relevantes como a metodologia empregada, o foco principal da pesquisa e os principais resultados obtidos.

Tabela 5 – Resumo dos trabalhos relacionados

Trabalho	**Metodologia**	**Objeto de estudo**	**Resultados**
Presente estudo (2025)	Revisão da literatura, análise de documentação técnica e entrevistas com desenvolvedores	atributos e funcionalidades que compõem ambientes de desenvolvimento integrado (IDEs)	Proposta de uma taxonomia para classificar IDEs
(ARAÚJO; PINTO, 2019)	Etapas de levantamento, coleta, geração de atributos e avaliação	Atributos interpretáveis em dados de trajetórias	Desenvolvimento de uma taxonomia para aplicações em aprendizado de máquina
(JUNIOR *et al.*, 2017)	Revisão da literatura e análise de documentação técnica	Atores na corretagem em nuvem IaaS	Modelo de classificação dos corretores de nuvem com base no tipo de serviço e área de atuação
(USMAN *et al.*, 2017a)	Estudo de mapeamento sistemático de literatura	Taxonomias aplicadas à Engenharia de Software	Identificou tendências e lacunas no uso de taxonomias em ES e revisou método para seu desenvolvimento sistemático

Fonte: Elaborado pela autora

6 CONCLUSÕES E TRABALHOS FUTUROS

O presente trabalho teve como objetivo propor uma taxonomia para IDE através de uma revisão da literatura e pela percepção de profissionais de tecnologia sobre as principais características que compõem um ambiente de desenvolvimento integrado. O trabalho foi desenvolvido seguindo a metodologia (NICKERSON *et al.*, 2013) e entrevistas, onde foram questionados os profissionais sobre os principais aspectos necessários para classificar um IDE. Esse processo contou com um total de sete participantes, entrevistados no mês de julho de 2025.

Da análise dos resultados, pode-se evidenciar que a customização das ferramentas foi um aspecto relevante tratado pela maioria dos entrevistados, sendo o autocompletar, extensões e os plugins funcionalidades essenciais para tal feito. Os entrevistados citaram que esses recursos garantem maior produtividade e melhor desempenho no desenvolvimento de suas atividades.

Outro ponto mencionado foi o fato de o ambiente permitir uma diversidade de linguagens, possibilitando o uso da mesma ferramenta em diferentes projetos. Essa questão foi vista como favorável, pois elimina a necessidade de utilizar múltiplas ferramentas no dia a dia.

A capacidade de versionamento foi outro fator bastante citado durante as entrevistas, sendo considerada muito importante para o controle e organização dos trabalhos ao longo do desenvolvimento. Os profissionais destacaram a importância da integração com Sistemas de Controle de Versão (SCVs) como o *Git*[1], pois contribui para um fluxo de trabalho colaborativo e eficiente.

O terminal integrado foi outro ponto destacado, pois sua presença no próprio IDE evita a necessidade de alternância entre janelas ou ferramentas externas. Essa ferramenta foi destacada como prática e ágil.

Com base nas entrevistas, as categorias propostas na taxonomia foram reconhecidas e citadas pelos participantes, deixando claro um alinhamento entre a taxonomia proposta e a vivência dos profissionais da área. Além disso, a aprovação dos participantes em relação ao modelo proposto, tanto em sua estrutura quanto aos pontos abordados, indica que ele é compreensível, representativo e aplicável ao mercado. Portanto, os insights obtidos nos resultados reforçam a consistência da taxonomia proposta, validando-a como uma representação adequada e coerente com os requisitos analisados pelos profissionais da área.

Como trabalhos futuros, é proposta uma ampliação do estudo com uma amostragem maior de entrevistados, contemplando uma diversidade de perfis e níveis de experiência, a fim de

[1] https://github.com/

fortalecer a confiabilidade dos insights extraídos e possibilitar a identificação de novos padrões. Também se sugere a realização de revisões periódicas na taxonomia, visando acompanhar a evolução das ferramentas e a incorporação de novas funcionalidades, garantindo que o modelo se mantenha atualizado e representativo.

Adicionalmente, sugere-se o desenvolvimento de um catálogo digital de IDEs fundamentado na taxonomia construída neste estudo. Esse catálogo reuniria informações organizadas de acordo com as dimensões e atributos definidos, permitindo buscas filtradas e comparações diretas entre diferentes ferramentas. Além de servir como um repositório de referência para profissionais, pesquisadores e estudantes, o catálogo poderia ser atualizado continuamente, incorporando novos ambientes e funcionalidades identificados tanto na literatura quanto em estudos empíricos. Tal iniciativa contribuiria para a disseminação do conhecimento, auxiliando na escolha de ferramentas mais adequadas a diferentes contextos de desenvolvimento.

REFERÊNCIAS

ANDRADE, L. H. d. *et al.* Trajtax: uma taxonomia para o domínio de trajetórias. Universidade Federal de Campina Grande, 2020.

ARAÚJO, I. M. d.; PINTO, V. B. Taxonomia nas áreas da biblioteconomia e da ciência da informação: uma revisão sistemática. Páginas a&b, 2019.

ARAUJO, V. M. R. H. d. A organização espacial da informação científica e tecnológica no brasil. Ibict, 1985.

BECK, K.; GRAHAM, D. Turbo pascal: the design decisions. **Byte**, v. 9, n. 12, p. 118–130, 1984.

BELITARDO, M. G. da S. **IDE's no mundo da programação**. 2021. Disponível em: <https://csguaratingueta.medium.com/ides-no-mundo-da-programa%C3%A7%C3%A3o-2179d89f5480>.

BEYNON-DAVIES, P.; CARNE, C.; MACKAY, H.; TUDHOPE, D. Rapid application development (rad): an empirical review. **European Journal of Information Systems**, Taylor & Francis, v. 8, n. 3, p. 211–223, 1999.

BORGES, A.; MOREIRA, A.; ARAúJO, J.; SILVA, A. A taxonomy for software defect classification in software engineering. **Journal of Systems and Software**, Elsevier, v. 170, p. 110738, 2020.

CAMPOS, M. d. A.; GOMES, H. E. Taxonomia e classificação: a categorização como princípio. **Encontro Nacional de Pesquisa e Pós-Graduação em Ciência da Informação (Enancib)**, v. 8, 2007.

CAMPOS M. L. A.; GOMES, H. E. **Taxonomia e classificação: a categorização como princípio.** 2008. Urlhttp://hdl.handle.net/20.500.11959/brapci/172684.

CARBONNELLE, P. **Top IDE index**. 2025. Disponível em: <https://pypl.github.io/IDE.html>.

CHAVES, A. M.; SILVA, G. proposta de uma arquitetura de software e funcionalidades para implementação de um ambiente integrado de desenvolvimento para a linguagem php. **I Jornada Científica e VI FIPA do CEFET Bambuí**, v. 31, p. 32–36, 2008.

FOWLER, M. Refactoring: Improving the design of existing code. In: **11th European Conference. Jyväskylä, Finland**. [S.l.: s.n.], 1997.

FRAIX-BURNET, D. Concepts of classification and taxonomy phylogenetic classification. **EAS Publications Series**, EDP Sciences, v. 77, p. 221–257, 2016. ISSN 1638-1963. Disponível em: <http://dx.doi.org/10.1051/eas/1677010>.

GLASS, R. L.; VESSEY, I. Contemporary application-domain taxonomies. **IEEE Software**, IEEE, v. 12, n. 4, p. 63–76, 1995.

JR, H. E. **Engenharia de software na prática**. [S.l.]: Novatec Editora, 2010.

JUNIOR, E. C.; MIERS, C. C.; KOSLOVSKI, G. P. Uma taxonomia para corretagem de nuvens iaas baseada na migração de infraestruturas virtuais. **Revista Brasileira de Computação Aplicada**, v. 9, n. 1, p. 15–30, 2017.

KAY, A. C. The early history of smalltalk. In: ACM. **The second ACM SIGPLAN conference on History of programming languages**. [S.l.], 1993. p. 69–95.

LIMA, G. **Saiba tudo sobre o IDE - Integrated Development Environment**. 2022. Disponível em: <https://www.alura.com.br/artigos/o-que-e-uma-ide#:~:text=O%20Turbo%20Pascal%20lan%C3%A7ou%20a,primeiro%20IDE%20real%20da%20hist%C3%B3ria.>

MATLOFF, N. **The Art of Debugging**. San Francisco: No Starch Press, 2012.

MEGA, G.; KON, F. God: Um depurador simbólico para sistemas de objetos distribuídos. **12º Salão de Ferramentas do Simpósio Brasileiro de Engenharia de Software**, 2005.

NICKERSON, R. C.; VARSHNEY, U.; MUNTERMANN, J. A method for taxonomy development and its application in information systems. **European Journal of Information Systems**, Taylor & Francis, v. 22, n. 3, p. 336–359, 2013.

NOLETO, C. **IDE: saiba o que é um ambiente de desenvolvimento integrado!** 2021. Disponível em: <https://blog.betrybe.com/tecnologia/ide/>.

OLIVEIRA, L. M. S. Caracterização do conceito de modularidade no desenvolvimento de linguagens de programação. 2017.

PRESSMAN, R. S. **Software engineering: a practitioner's approach**. [S.l.]: McGraw-Hill, 2010.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software-9**. [S.l.]: McGraw Hill Brasil, 2021.

SEBESTA, R. W. **Conceitos de Linguagem de Programação. 4º edição**. [S.l.]: Porto Alegre, Bookman, 2000.

SOUSA, A. P. Integração continua de codigo. **Revista Ada Lovelace**, 2022.

STACKOVERFLOW. **Developer survey**. 2021. Disponível em: <https://insights.stackoverflow.com/survey/2021#most-popular-technologies-new-collab-tools-prof>.

STACKOVERFLOW. **Developer survey**. 2024. Disponível em: <https://insights.stackoverflow.com/survey/2024#most-popular-technologies-new-collab-tools-prof>.

USMAN, M.; BÖRSTLER, J.; MENDES, E.; BRITTO, R. Taxonomies in software engineering: A systematic mapping study and a revised method for developing taxonomies. **Information and Software Technology**, Elsevier, v. 85, p. 43–59, 2017.

USMAN, M.; BRITTO, R.; BöRSTLER, J.; MENDES, E. A taxonomy of requirements engineering and software design concepts for self-adaptive systems. **Information and Software Technology**, Elsevier, v. 91, p. 114–129, 2017.

ZHANG, H.; MIAO, C.; WANG, H. Y.; LI, Y.; LEUNG, C. Classifying software tools for big data analysis: a taxonomy. **Journal of Big Data**, Springer, v. 5, n. 1, p. 26, 2018.