## Resumo

uma ferramenta de desenvolvimento que debuga, executa e permite escrever o código.

IDE, ou ambiente de desenvolvimento integrado, é um software que combina ferramentas comuns de desenvolvimento em uma única interface gráfica do usuário (GUI), facilitando o desenvolvimento de aplicações. Um IDE geralmente consiste em:

* **Editor de código-fonte**: é um editor de texto que auxilia na criação de código de software por meio de funcionalidades como destaque da sintaxe com indicadores visuais, recurso de preenchimento automático específico da linguagem e verificação de bugs durante a criação.
* **Automação de compilação local**: são utilitários que automatizam tarefas simples e repetíveis durante a criação de uma compilação local do software usada pelo desenvolvedor. São tarefas como compilação de código-fonte em código binário, criação de pacotes de código binário e execução de testes automatizados.
* **Debugger**: é um programa usado para testar outros programas e mostrar graficamente a localização do bug no código original.

## **Para que serve um IDE?**

Os ambientes de desenvolvimento integrado ajudam os desenvolvedores a programar novas aplicações de forma rápida, já que os vários utilitários não precisam ser ajustados e integrados manualmente durante a configuração. Os desenvolvedores também não precisam passar horas aprendendo a usar cada uma das diferentes ferramentas, porque cada utilitário está localizado no mesmo workbench. Isso é especialmente útil quando há desenvolvedores novos em um projeto. Eles podem contar com o IDE para se atualizar em relação às ferramentas e fluxos de trabalho da equipe. Na verdade, o objetivo da maior parte das funcionalidades é economizar tempo: o preenchimento inteligente e a geração automática de código, por exemplo, eliminam a necessidade de digitar sequências inteiras.

## **Quais as vantagens de um IDE?**

Outras funcionalidades comuns aos IDEs têm o objetivo de ajudar os desenvolvedores a organizar seu fluxo de trabalho e solucionar problemas. Os IDEs analisam o código no momento em que está sendo escrito. Assim, bugs causados por erro humano são identificados em tempo real. Como todos os utilitários estão em uma única interface gráfica (GUI), os desenvolvedores podem executar as tarefas sem precisar trocar de aplicação. A maioria dos ambientes de desenvolvedor integrados também conta com destaque da sintaxe, usando indicadores visuais para diferenciá-la da gramática no editor de texto. Além disso, alguns IDEs incluem navegadores de classes e objetos, bem como diagramas de hierarquia de classes em determinadas linguagens.

É possível desenvolver aplicações sem um IDE. O desenvolvedor também pode, basicamente, compilar seu próprio IDE, integrando manualmente vários utilitários com um editor leve de textos, como Vim ou Emacs. O benefício dessa abordagem é o alto nível de personalização e controle que oferece aos desenvolvedores. No contexto empresarial, entretanto, a economia de tempo, a padronização do ambiente e as [funcionalidades de automação](https://www.redhat.com/pt-br/topics/automation) dos IDEs modernos geralmente superam outros benefícios.

Atualmente, a maioria das equipes empresariais de desenvolvimento escolhe o IDE pré-configurado que melhor serve ao seu caso de uso. A questão, portanto, não é decidir usar ou não um IDE, e sim qual usar.

## **Quais são os tipos de IDE?**

Existem inúmeros casos de usos técnicos e empresariais dos ambientes de desenvolvimento integrados, portanto são muitas as opções proprietárias e [open source](https://www.redhat.com/pt-br/about/open-source) no mercado. De modo geral, as características mais importantes que diferenciam os IDEs são:

* **A quantidade de linguagens compatíveis**: alguns IDEs são dedicados a uma linguagem específica, por isso acabam sendo melhores para determinado paradigma de programação. O IntelliJ, por exemplo, é conhecido principalmente como um IDE de Java. Outros IDEs suportam uma vasta gama de linguagens. O Eclipse, por exemplo, é compatível com Java, XML e Python, dentre outras.
* **Sistemas operacionais suportados:** o sistema operacional do desenvolvedor limitará sua escolha, exceto quando o IDE estiver na nuvem. Se a aplicação em desenvolvimento for destinada a um usuário final com um sistema operacional específico (Android ou iOS, por exemplo), isso também pode criar outra limitação.
* **Funcionalidades de automação**: a maioria dos IDEs incluem editor de texto, automação de compilação e debugger, e muitos são compatíveis com recursos adicionais, tais como refatoração, pesquisa de código e ferramentas de [integração e implantação contínua](https://www.redhat.com/pt-br/topics/devops/what-is-ci-cd) (CI/CD).
* **Impacto no desempenho do sistema**: pode ser importante considerar o volume de memória de um IDE se o desenvolvedor quiser executar, simultaneamente, outras aplicações que demandem muito processamento.
* **Plug-ins e extensões**: alguns IDEs permitem a personalização dos fluxos de trabalho, adaptando-se às necessidades e preferências do desenvolvedor.

### **IDE para desenvolvimento de aplicações mobile**

Praticamente todos os setores foram afetados pela crescente popularidade das aplicações para smartphones e tablets. Além das tradicionais aplicações web, muitas empresas também passaram a desenvolver [aplicações mobile](https://www.redhat.com/pt-br/topics/mobile). Um dos principais fatores no desenvolvimento de aplicações mobile é a escolha da plataforma. Por exemplo, se uma aplicação nova for usada em iOS, Android e páginas web, talvez seja melhor usar um IDE que seja compatível com diversas plataformas em vários sistemas operacionais.

### **IDE na nuvem**

IDEs fornecidos na nuvem como [Software como Serviço (SaaS)](https://www.redhat.com/pt-br/topics/cloud-computing/what-is-saas) oferecem benefícios exclusivos quando comparados a ambientes locais de desenvolvimento. Por exemplo, nas soluções SaaS, não existe a necessidade de fazer o download de softwares e configurar ambientes e dependências locais. Assim, os desenvolvedores podem começar a contribuir imediatamente com o projeto. Isso também viabiliza um nível de padronização dos ambientes da equipe, o que reduz problemas de incompatibilidade como "se funciona na minha máquina, por que não funciona na sua?". Além disso, já que o gerenciamento do ambiente de desenvolvimento é centralizado, os códigos não ficam no computador de uma só pessoa, o que ajuda com questões de propriedade intelectual e segurança.

O impacto dos processos em máquinas locais também é diferente. Geralmente, executar compilações e testar suites são processos intensos, por isso os desenvolvedores podem não conseguir usar as estações de trabalho durante essas operações. Um IDE SaaS consegue distribuir tarefas de longa duração sem monopolizar os recursos computacionais de uma máquina local. Os IDEs de nuvem também não costumam depender de plataforma, permitindo a conexão com diferentes fornecedores de nuvem.

#### **Leitura Recomendada**

* Press Release: [a Red Hat apresenta o primeiro IDE nativo em Kubernetes](https://www.redhat.com/pt-br/about/press-releases/red-hat-introduces-industrys-first-kubernetes-native-integrated-developer-environment-red-hat-codeready-workspaces)

#### **Aprenda mais**

* Webinar: [Como simplificar o desenvolvimento de containers com as ferramentas da Red Hat](https://www.redhat.com/pt-br/page-not-found)
* Artigo: [O que é middleware?](https://www.redhat.com/pt-br/topics/middleware/what-is-middleware)
* Artigo: [Por que escolher a Red Hat?](https://www.redhat.com/pt-br/topics/middleware/why-choose-red-hat-middleware)
* Artigo: [O que é CI/CD?](https://www.redhat.com/pt-br/topics/devops/what-is-ci-cd)
* Experimente grátis: [Red Hat®](https://developers.redhat.com/products/codeready-studio/download)