INSTALAÇÃO DO AMBIENTE .NET

1 VISÃO GERAL DO .NET

1.1 Introdução

O .NET, um framework de software desenvolvido pela Microsoft e lançado inicialmente em 2002, representa um marco no desenvolvimento de software moderno. Esta plataforma multifacetada, desde sua concepção, tem sido uma força transformadora no mundo da tecnologia, oferecendo um conjunto abrangente de ferramentas, bibliotecas e linguagens de programação. Essa combinação poderosa não apenas simplifica a criação de aplicativos robustos e versáteis, mas também democratiza o desenvolvimento de software, tornando-o acessível em diversos ambientes e sistemas operacionais.

A ideia por trás do .NET era oferecer uma solução unificada que abrangesse uma variedade de necessidades de programação - desde o desenvolvimento de aplicações web e serviços baseados em nuvem até aplicativos móveis e de desktop. Essa abordagem integrada visava resolver alguns dos desafios mais críticos enfrentados pelos desenvolvedores, como a incompatibilidade entre plataformas e a complexidade na manutenção de código para diferentes sistemas operacionais.

O .NET surgiu em um momento em que o mundo da computação estava passando por rápidas mudanças, com a internet se tornando cada vez mais integrada à vida cotidiana e as empresas procurando expandir sua presença online. Nesse contexto, o .NET ofereceu uma plataforma altamente adaptável, capaz de atender às crescentes demandas por aplicações mais dinâmicas, interativas e conectadas.

A introdução do .NET marcou um ponto de inflexão na forma como os softwares eram desenvolvidos e implementados. Com sua arquitetura baseada

em componentes, suporte a múltiplas linguagens e um modelo de programação consistente, o .NET simplificou o processo de desenvolvimento, permitindo que os programadores se concentrassem mais na lógica de negócios e menos nos detalhes técnicos do ambiente de execução.

Além disso, a Microsoft, ao desenvolver o .NET, não se limitou a oferecer apenas um framework. Ela estabeleceu um ecossistema completo, incluindo ferramentas de desenvolvimento, como o Visual Studio, e serviços de suporte, como o .NET Framework Class Library e o Common Language Runtime, que juntos proporcionam uma experiência de desenvolvimento coerente e eficiente.

O .NET também desempenhou um papel crucial na promoção de práticas de desenvolvimento de software mais ágeis e colaborativas. Com sua estrutura modular e suporte a serviços baseados em nuvem, permitiu que as equipes de desenvolvimento trabalhassem de maneira mais sincronizada e responsiva, adaptando-se rapidamente às mudanças nos requisitos do projeto e nas tendências do mercado.

Desde seu lançamento, o .NET passou por várias evoluções significativas. Uma das mais notáveis foi a introdução do .NET Core, uma versão cross-platform e open-source do .NET, que ampliou ainda mais o alcance e a aplicabilidade do framework. Essa mudança refletiu o compromisso da Microsoft com as tendências emergentes na tecnologia e seu apoio à comunidade de código aberto.

Hoje, o .NET continua a ser uma plataforma vital para o desenvolvimento de software, com uma comunidade ativa de desenvolvedores e uma gama constante de atualizações e melhorias. A plataforma evoluiu para abraçar não apenas as necessidades tradicionais de desenvolvimento de software, mas também para se adaptar às novas demandas do mercado, como a computação em nuvem, big data, e a inteligência artificial.

1.2 Componentes fundamentais do .Net

O .NET é composto por vários componentes chave, cada um desempenhando um papel vital no desenvolvimento de software:

- **Common Language Runtime (CLR):** O CLR é o coração do .NET. Ele atua como um agente de execução que gerencia o código em tempo de execução, oferecendo serviços essenciais como gerenciamento de memória, threads e comunicação remota. O CLR também garante segurança e precisão no código através de um sistema rigoroso de tipagem e verificações de código. Uma característica importante do CLR é a compilação just-in-time (JIT), que converte o código gerido em linguagem de máquina nativa do sistema em que está sendo executado, melhorando assim o desempenho.
- **Base Class Library (BCL):** Esta biblioteca é uma vasta coleção de tipos reutilizáveis, orientada a objetos, que você pode utilizar para desenvolver aplicações. Ela inclui tudo, desde a gestão de cadeias de caracteres e coleções até a conectividade de base de dados e acesso a arquivos. As bibliotecas do .NET facilitam o desenvolvimento de uma ampla gama de aplicações e serviços, incluindo aplicações de consola, aplicações GUI do Windows (Windows Forms), Windows Presentation Foundation (WPF), ASP.NET para aplicações web, serviços do Windows e muito mais.
- **Framework Class Library (FCL**): Uma expansão da BCL, oferece mais funcionalidades, incluindo acesso a bancos de dados, desenvolvimento web. Uma coleção de classes reutilizáveis, interfaces e tipos de valor que facilitam e otimizam o processo de desenvolvimento e proporcionam acesso à funcionalidade do sistema. Essa biblioteca inclui tipos que realizam funções como representação de tipos de dados básicos e exceções, encapsulamento de estruturas de dados, execução de operações de I/O, acesso a informações sobre tipos carregados, invocação de verificações de segurança do .NET, e fornecimento de acesso a dados. As classes do FCL são organizadas em namespaces

para facilitar a busca e referência e podem ser usadas como estão ou derivadas para criar classes

- **ASP.NET:** uma plataforma versátil e dinâmica para o desenvolvimento de aplicações web e serviços web, destacando-se como um componente essencial para desenvolvedores que desejam criar soluções web eficientes, robustas e escaláveis. Ela proporciona um ambiente rico para a construção de websites interativos, aplicações web integradas e serviços web, oferecendo recursos avançados de segurança, gerenciamento de estado, e integração de back-end. ASP.NET é ideal para desenvolver aplicações que exigem alto desempenho, confiabilidade, e uma experiência de usuário refinada e interativa.
- **Interoperabilidade Linguística**: O .NET Framework oferece interoperabilidade entre diferentes linguagens de programação. Os compiladores que visam o .NET Framework emitem um código intermédio chamado Common Intermediate Language (CIL), que é depois compilado pelo CLR. Esta característica permite que rotinas escritas em uma linguagem sejam acessíveis por outras linguagens, o que promove a flexibilidade e a reutilização do código.
- **Compatibilidade de Versão e Execução Lado a Lado**: O .NET Framework é projetado para ser compatível com versões anteriores, permitindo que as aplicações desenvolvidas em uma versão específica do .NET Framework funcionem sem modificação em versões posteriores. Além disso, várias versões do .NET Framework podem coexistir no mesmo computador, permitindo a execução de diferentes aplicações em suas respectivas versões do framework.
- **Integração com Cloud e Big Data:** O .NET se adapta às tendências atuais de tecnologia, oferecendo integração com ambientes de cloud computing e big data, destacando-se em cenários que exigem escalabilidade e processamento de grandes volumes de dados.

1.3 Diferenças entre .NET Framework, .NET Core e .NET 7/8

Para entender as nuances entre .NET Framework, .NET Core e .NET 7/8, é essencial compreender a trajetória e a evolução do .NET ao longo dos anos. Cada uma dessas versões do .NET atende a diferentes necessidades e cenários de desenvolvimento.

1.3.1 .NET Framework

O .NET Framework, lançado em 2002 pela Microsoft, representou um avanço significativo no desenvolvimento de software, especialmente para aplicações Windows e web. Projetado para simplificar o processo de desenvolvimento, oferecia aos programadores uma plataforma consistente repleta de bibliotecas e funcionalidades. Sua estrutura monolítica significava que todas os seus componentes, desde a interface gráfica até o acesso a dados, estavam fortemente integrados, proporcionando uma experiência de desenvolvimento coesa.

Dentro do .NET Framework, diversas tecnologias chave se destacam. Por exemplo, para o desenvolvimento de aplicações de desktop, **Windows Forms** era essencial, oferecendo uma maneira fácil e eficiente de criar interfaces gráficas de usuário. No contexto web, **ASP.NET** se tornou a base para construir páginas web dinâmicas e aplicações web ricas, integrando-se perfeitamente com o **Internet Information Services** (IIS) para otimizar o desempenho e a segurança. Para o acesso e manipulação de dados, **ADO.NET** se mostrou crucial, permitindo interações eficientes com bases de dados e suportando tanto operações simples quanto transações mais complexas, graças à sua arquitetura desconectada.

No entanto, apesar de suas inúmeras vantagens, o **.NET Framework** tinha limitações notáveis. A mais proeminente era sua incompatibilidade com o desenvolvimento de aplicações **cross-platform**. Inicialmente, foi projetado exclusivamente para sistemas baseados em Windows, restringindo seu uso em ambientes de TI mais diversificados. Além disso, a natureza monolítica do .**NET**

Framework, embora benéfica para a integração de suas componentes, podia tornar as atualizações e a manutenção de sistemas mais complexos uma tarefa desafiadora. Com o tempo, essas limitações abriram caminho para novas implementações do .NET, como o .NET Core e o .NET 5+, que buscaram superar essas barreiras, oferecendo uma experiência de desenvolvimento mais moderna e adaptável.

1.3.2 .NET Core

O .NET Core, uma evolução significativa do **.NET Framework**, foi introduzido pela Microsoft como uma resposta às crescentes demandas por aplicações mais versáteis e adaptáveis. Esta nova plataforma se destacou por sua capacidade de suportar o desenvolvimento de aplicações **cross-platform,** o que representou um grande salto em relação ao seu predecessor, que era limitado a ambientes **Windows**.

Diferentemente do .NET Framework, o .NET Core foi projetado com um foco especial na modularidade. Isso significava que os desenvolvedores tinham a liberdade de incluir apenas as bibliotecas e componentes necessários para suas aplicações, ao invés de depender de uma estrutura monolítica. Essa abordagem modular não apenas tornava as aplicações mais leves e eficientes, mas também facilitava a manutenção e atualização, já que os componentes individuais poderiam ser atualizados de forma independente.

Outra característica fundamental do .NET Core é sua portabilidade. Ao permitir o desenvolvimento de aplicações que podem ser executadas em diferentes sistemas operacionais, como **Windows**, **Linux** e **macOS**, o .NET Core abriu novos horizontes para os desenvolvedores, permitindo-lhes alcançar uma base de usuários mais ampla sem a necessidade de reescrever o código para cada plataforma.

Além disso, o .NET Core também trouxe melhorias significativas no desempenho. Otimizações no coletor de lixo (Garbage Collector), JIT (Just-In-Time) compilation e outras áreas-chave garantiram que as aplicações

construídas no .NET Core fossem mais rápidas e eficientes. Isso era particularmente benéfico para aplicações de alto desempenho e serviços de nuvem, onde a eficiência e a escalabilidade são críticas.

A adoção do .NET Core também refletiu uma mudança na filosofia de desenvolvimento da Microsoft, com um forte compromisso com o código aberto. Isso possibilitou que a comunidade de desenvolvedores contribuísse para a plataforma, promovendo inovação e melhorias contínuas.

Em suma, o .NET Core representou uma modernização significativa da plataforma .NET, abordando muitas das limitações do .NET Framework e introduzindo novos recursos e capacidades que eram essenciais para o desenvolvimento de software contemporâneo. Sua modularidade, portabilidade e desempenho aprimorado, juntamente com seu suporte ao código aberto, o tornaram uma escolha popular para desenvolvedores em todo o mundo.

1.3.3 .NET

Com o lançamento do .NET 5 e suas versões subsequentes (incluindo .NET 7/8), a Microsoft consolidou o .NET Framework e o .NET Core em uma única plataforma, simplesmente chamada .NET. Esta versão representa a unificação e a evolução das versões anteriores, visando simplificar a escolha da plataforma .NET e oferecer uma base única para todos os tipos de projetos .NET.

O avanço do .NET Framework para versões mais recentes como o .NET 7 e o .NET 8 reflete uma evolução contínua na busca por maior eficiência, flexibilidade e capacidade de atender às demandas crescentes do desenvolvimento moderno de software. Estas versões são parte do esforço contínuo da Microsoft para criar uma plataforma unificada e escalável que abrange não apenas aplicações **Windows**, mas também Linux, **macOS**, e desenvolvimento móvel com Xamarin, ampliando significativamente o alcance do .NET.

O .NET 7 e o .NET 8 continuam a tradição de inovação, trazendo melhorias significativas em termos de desempenho e produtividade. Eles

oferecem suporte aprimorado para desenvolvimento de aplicações **cloud-native**, **microserviços**, e **containerização**, aspectos cruciais na era atual de computação distribuída. Além disso, com a adoção crescente de práticas de **DevOps**, essas versões do .NET proporcionam integrações mais robustas com ferramentas de CI/CD (Continuous Integration/Continuous Delivery), facilitando a automatização do ciclo de vida de desenvolvimento de software.

Outra característica chave do .NET 7 e 8 é a ênfase na interoperabilidade e no desenvolvimento multiplataforma. Com a inclusão do MAUI (Multi-platform App UI), uma evolução do **Xamarin.Forms**, desenvolvedores têm à disposição um framework poderosa para criar interfaces de usuário que funcionam de forma consistente em diferentes plataformas. Isso significa que é possível desenvolver uma aplicação que tem uma ótima aparência e funcionamento tanto em iOS quanto em Android, além de desktops Windows e Mac, com uma base de código única.

Além disso, as atualizações de desempenho e as otimizações de memória continuam a ser um foco, garantindo que aplicações desenvolvidas com .NET 7 e 8 sejam mais rápidas e eficientes. Isso é particularmente relevante para aplicações que processam grandes volumes de dados ou que exigem alta capacidade de resposta, como aplicações financeiras ou jogos.

Em resumo, o .NET 7 e o .NET 8 representam um salto significativo na capacidade de desenvolver aplicações modernas, robustas e de alto desempenho, oferecendo aos desenvolvedores um conjunto de ferramentas e recursos que facilitam a criação de soluções inovadoras em uma variedade de plataformas e ambientes.

1.3.4 Conclusão

A trajetória do .NET, desde o .NET Framework até o .NET 7/8, ilustra a evolução constante da plataforma em resposta às demandas de um cenário de desenvolvimento de software em rápida mudança. O .NET Framework, com sua abordagem monolítica e foco no desenvolvimento Windows e web, estabeleceu

um forte alicerce, mas limitava-se ao ambiente Windows e enfrentava desafios na manutenção e atualizações. O advento do .NET Core marcou uma virada significativa, introduzindo modularidade, suporte a desenvolvimento **cross-platform,** desempenho aprimorado e um compromisso com o código aberto, atendendo às necessidades de aplicações mais versáteis e adaptáveis.

Com o .NET 7/8, a Microsoft unificou as melhores características do .NET Framework e .NET Core, criando uma plataforma única que abrange não apenas aplicações **Windows**, mas também **Linux**, **macOS** e desenvolvimento móvel. Essas versões mais recentes enfatizam a eficiência, flexibilidade, suporte a práticas de **DevOps**, e desenvolvimento multiplataforma com ferramentas como **MAUI**, destacando-se na era atual de computação distribuída e **cloud-native**. Além disso, continuam a tradição de melhorias no desempenho e na otimização de memória, essenciais para aplicações de alto desempenho.

Em conclusão, a evolução do .NET reflete um esforço contínuo para atender às demandas crescentes e diversificadas do desenvolvimento de software moderno, oferecendo aos desenvolvedores uma plataforma robusta, versátil e eficiente para criar soluções inovadoras em uma variedade de plataformas e ambientes.

2 PREPARANDO O AMBIENTE

Antes de embarcarmos na empolgante jornada de desenvolvimento com a plataforma .NET, é de suma importância estabelecer um ambiente de trabalho sólido e bem-preparado. Esta preparação inicial é um passo crítico que define o sucesso e a eficiência do desenvolvimento de software. Ela envolve uma série

de etapas essenciais, começando por entender e atender aos requisitos de sistema necessários para o funcionamento ideal da plataforma .NET.

Os requisitos de sistema para o .NET variam de acordo com a versão escolhida e podem incluir aspectos como sistema operacional compatível, memória disponível, espaço em disco e capacidades do processador. A conformidade com estes requisitos é fundamental para garantir que a plataforma .NET funcione de forma eficiente, permitindo que você aproveite ao máximo suas capacidades. É importante verificar esses requisitos no site oficial da Microsoft ou na documentação do .NET para garantir que seu sistema atende a todas as especificações necessárias.

Além disso, a escolha da versão correta do .NET para o seu projeto é um passo crucial. Com várias versões disponíveis, cada uma com suas próprias características e capacidades, é vital selecionar a que melhor se adequa às necessidades do seu projeto. Por exemplo, se você está planejando um projeto de longa duração que exige estabilidade e suporte contínuo, uma versão de Suporte de Longo Prazo (LTS) pode ser a escolha mais apropriada. Por outro lado, se o projeto requer as funcionalidades mais recentes e você está disposto a atualizar frequentemente, uma versão de Suporte a Termos Standard (STS) pode ser mais adequada.

Uma vez que os requisitos de sistema estejam atendidos e a versão correta do .NET esteja selecionada, é essencial também configurar adequadamente o ambiente de desenvolvimento. Isso inclui a instalação de ferramentas essenciais como o Visual Studio, o SDK do .NET e outras dependências necessárias para o desenvolvimento. Configurar corretamente o ambiente de desenvolvimento desde o início pode economizar tempo e evitar problemas técnicos no futuro.

Vamos abordar cada um desses dois aspectos fundamentais – requisitos de sistema e seleção de versão – para garantir que você tenha uma base sólida para o desenvolvimento de projetos .NET. Com um ambiente bem-preparado, você estará pronto para explorar as vastas possibilidades que o .NET oferece, desenvolvendo aplicações robustas, eficientes e inovadoras.

2.1 Requisitos de Sistema para .NET

O .NET é uma plataforma de desenvolvimento versátil, suportando uma ampla gama de aplicações, desde sites até aplicações móveis e de desktop. Para garantir que o .NET funcione sem problemas, seu sistema deve atender a alguns requisitos básicos:

- **Sistema Operacional:** O .NET é compatível com Windows, macOS e distribuições Linux. Certifique-se de que seu sistema operacional esteja atualizado para obter a melhor compatibilidade e segurança.
- **Memória e Processamento:** Embora o .NET seja eficiente, recomenda-se ter pelo menos **4GB** de RAM e um processador moderno para garantir uma experiência de desenvolvimento suave.
- **Espaço em Disco:** A instalação do .NET e dos ambientes de desenvolvimento associados, como o Visual Studio, requer um espaço considerável em disco. Certifique-se de ter pelo mínimo 20GB de espaço livre.
- **Ferramentas de Desenvolvimento:** Instale um IDE ou editor de código que suporte .NET, como Visual Studio, Visual Studio Code ou JetBrains Rider.

2.2 Versões do .NET

Cada produto desenvolvido pela Microsoft segue um ciclo de vida bem definido, que é crucial para a gestão e planejamento de TI. Este ciclo começa com o lançamento do produto e termina quando ele não é mais suportado, conhecido como 'fim do suporte'. Ter um entendimento claro das principais datas e fases deste ciclo pode ajudar significativamente os usuários e as empresas a tomar decisões informadas sobre quando é o momento ideal para atualizar, modificar ou substituir seu software.

No universo dos produtos Microsoft, os clientes têm a flexibilidade de escolher entre duas categorias principais de suporte: Suporte de Longo Prazo (LTS) e Suporte a Termos Standard (STS). Ambas as categorias mantêm o padrão elevado de qualidade, garantindo confiabilidade e estabilidade. Contudo, a diferença chave entre elas está na duração do suporte oferecido.

As versões de Suporte de Longo Prazo (LTS) são projetadas pensando na estabilidade e na previsibilidade. Elas são ideais para organizações que precisam de um ambiente mais constante, com mudanças minimizadas. Essas versões recebem suporte completo e patches de segurança gratuitos por um período de três anos. Este suporte prolongado é ideal para sistemas críticos ou para ambientes onde mudanças frequentes podem levar a perturbações operacionais.

Por outro lado, as versões de Suporte a Termos Standard (STS) são voltadas para usuários e organizações que desejam acessar as funcionalidades mais recentes mais rapidamente. Estas versões recebem suporte e patches gratuitos por um período mais curto de 18 meses. A escolha por STS é frequentemente feita por aqueles que estão dispostos a adaptar-se rapidamente às novas tecnologias e que podem beneficiar-se de inovações e melhorias em um ritmo mais acelerado.

Durante o ciclo de vida de suporte de uma versão específica, é fundamental que os sistemas se mantenham atualizados com os patches de segurança e atualizações lançadas. Essas atualizações são essenciais para proteger os sistemas contra vulnerabilidades de segurança, melhorar a funcionalidade e garantir a compatibilidade com novos padrões e tecnologias. Manter os sistemas atualizados não apenas fortalece a segurança, mas também otimiza o desempenho e a estabilidade do software.

Compreender estas opções de suporte e manter-se atento às atualizações e ao fim do ciclo de vida do suporte é vital para a manutenção da integridade, segurança e eficiência dos sistemas de TI nas organizações.

2.2.1 Versões Atuais

Versão	Data Lançamento	Versão Atual	Tipo Versão	Fim Suporte
.NET 8	14/11/2023	8.0.0	LTS	10/11/2026
.NET 7	08/11/2022	7.0.14	STS	14/05/2024
.NET 6	08/11/2021	6.0.25	LTS	12/11/2024

Quadro 1 – Versões .Net
Fonte: Microsoft (2024)

2.2.2 Tipos de versão

- **Suporte de longo prazo (LTS):** As versões do LTS têm suporte por três anos após a versão inicial.
- **Suporte a Termos Standard (STS):** As versões STS são suportadas por seis meses após uma versão subsequente de STS ou LTS. Os lançamentos acontecem a cada 12 meses, então o período de suporte para STS é de 18 meses.

2.2.3 Cadência de versões do .NET

Uma nova versão principal do .NET é publicada todos os anos em novembro, permitindo que desenvolvedores, a comunidade e as empresas planejem seus roteiros. Até mesmo versões numeradas são versões LTS que obtêm suporte gratuito e patches por três anos.

As versões ímpares numeradas são versões STS que recebem suporte e patches gratuitos por 18 meses.

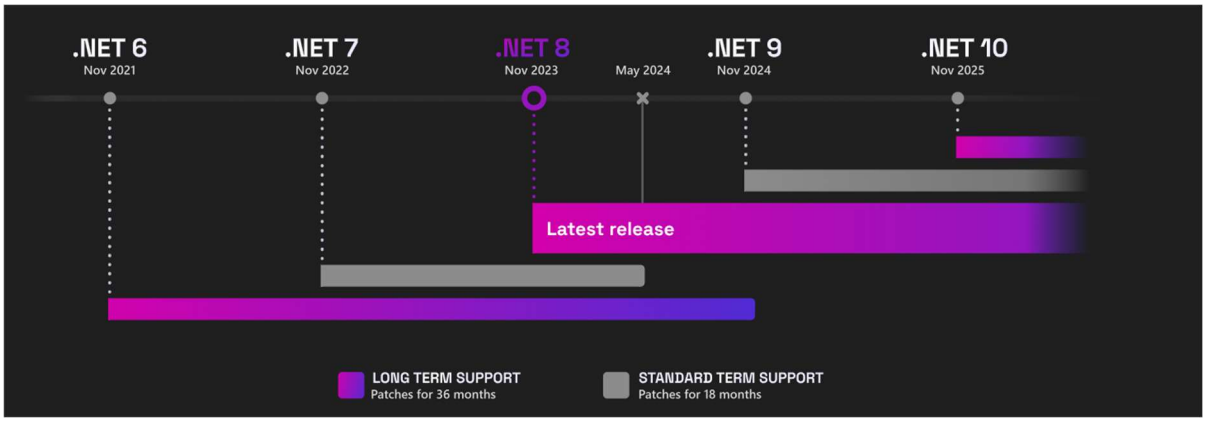

Figura 1 – Versões .Net
Fonte: Microsoft (2024)

2.3 Escolhendo a Versão Correta do .NET

- **Estabilidade vs. Recursos Mais Recentes:** Se você precisa de estabilidade e suporte a longo prazo, opte pelas versões LTS (Long-Term Support) do .NET. Para acessar os recursos mais recentes, escolha as versões atuais.
- **Compatibilidade com Projetos Existentes:** Se estiver trabalhando em um projeto existente, verifique a versão do .NET que ele utiliza e continue com ela, a menos que haja uma necessidade de atualização.
- **Requisitos do Projeto:** Considere os recursos específicos de cada versão do .NET e como eles se alinham com os requisitos do seu projeto.
- **Documentação e Comunidade:** Verifique a documentação disponível para a versão escolhida e a atividade da comunidade, pois isso pode facilitar o desenvolvimento e a solução de problemas.

3 PROCESSO DE INSTALAÇÃO DO .NET

Instalar o .NET é um processo direto, mas varia ligeiramente dependendo do seu sistema operacional. Abaixo, você encontrará um guia passo a passo para instalar o .NET em Windows, Mac e Linux.

3.1 Passo a Passo para Instalar o .NET

3.1.1 Windows

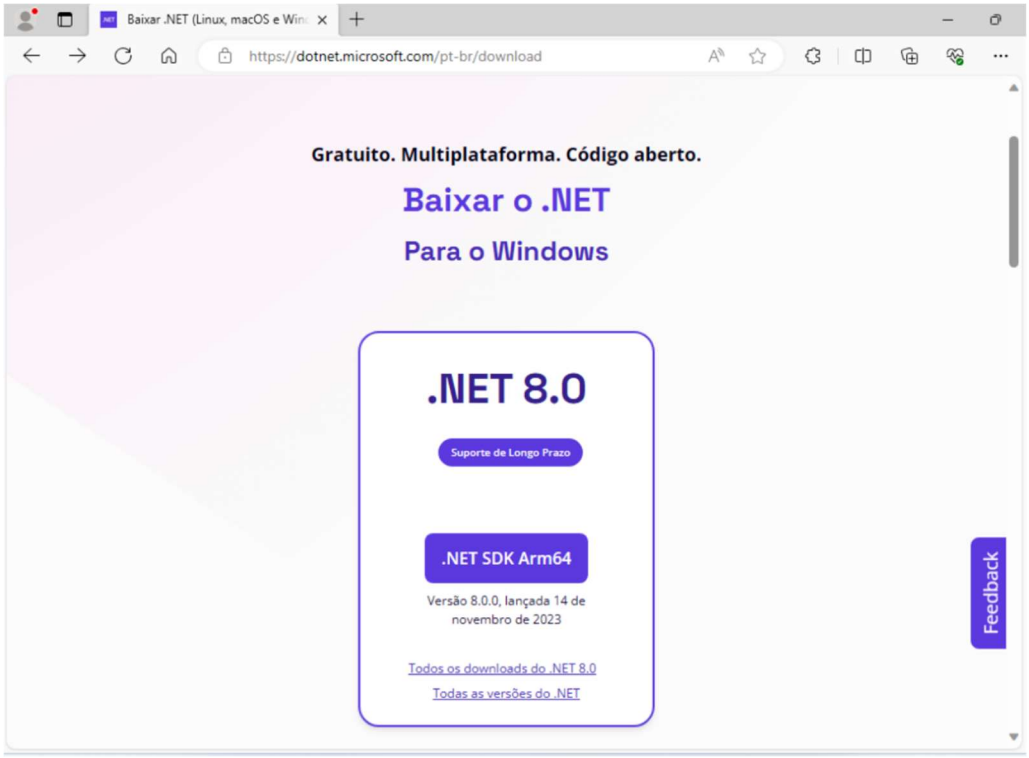

Figura 2 – Site para download .Net no Windows

- **Visite o Site Oficial:** Acesse o site do .NET *https://dotnet.microsoft.com/pt-br/downloads* e

Escolha a Versão: Selecione a versão do .NET que você deseja instalar. Para usuários do Windows, geralmente é recomendável a versão mais recente, a menos que você tenha requisitos específicos.

Baixe o Instalador: Baixe o instalador para Windows. Ele geralmente vem no formato de um executável (.exe).

Execute o Instalador: Dê um duplo clique no arquivo baixado e siga as instruções na tela para completar a instalação.

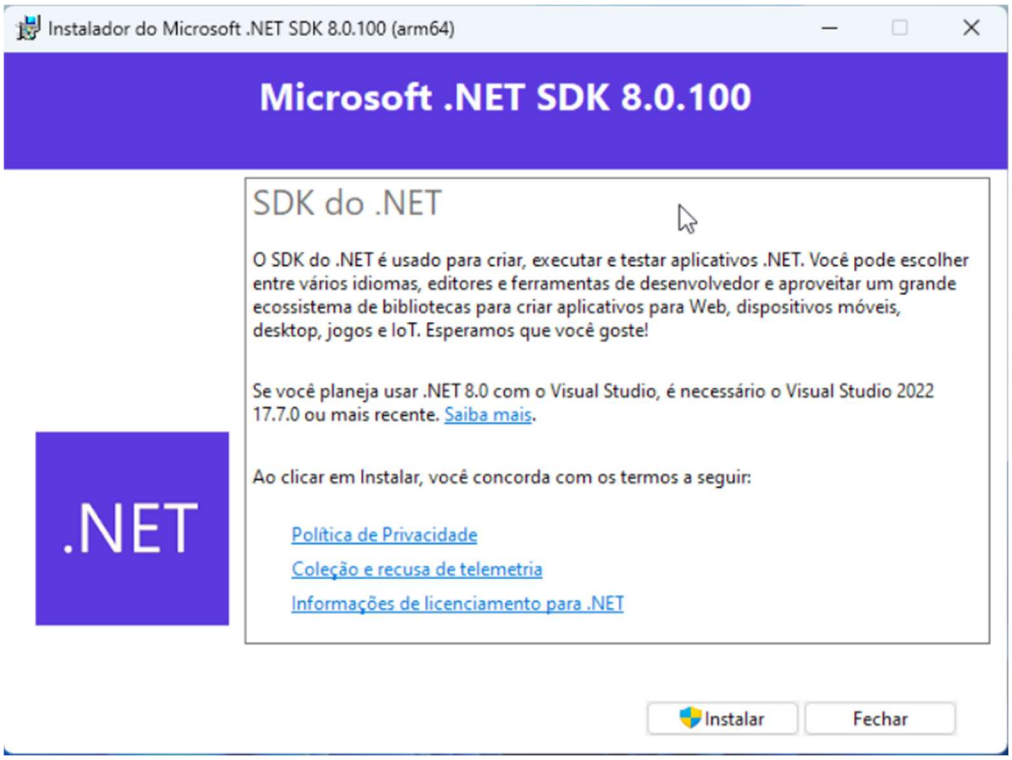

Figura 3 – Tela inicial da instalação no Windows

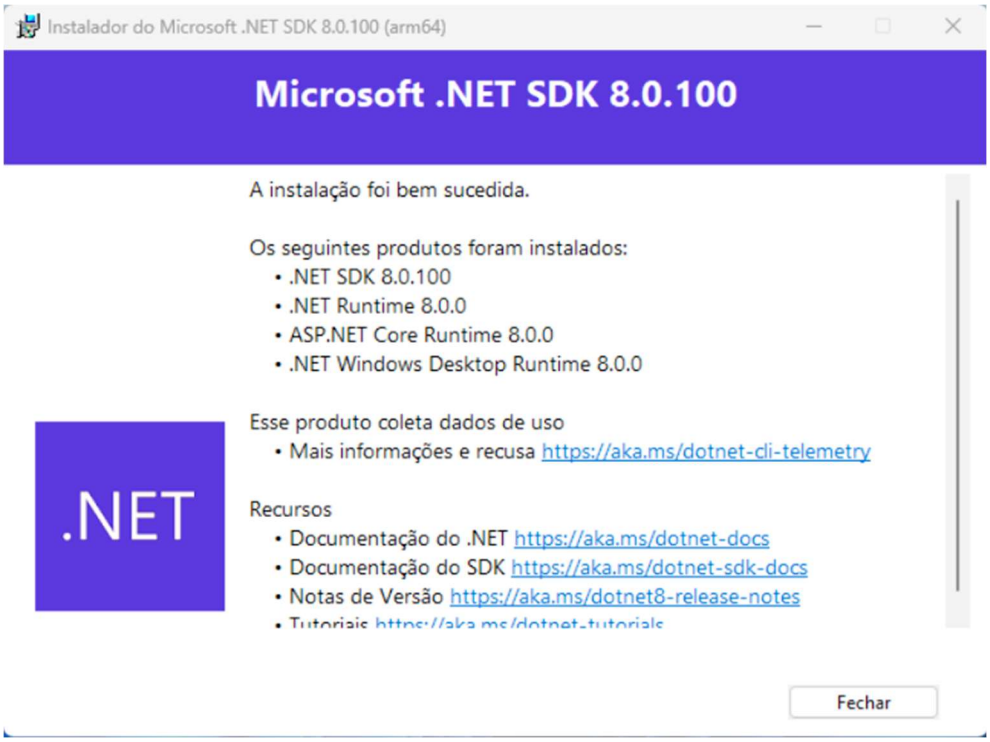

Figura 4 – Tela de conclusão da instalação Windows

Verifique a Instalação: Após a instalação, abra o prompt de comando e digite *dotnet --version* para verificar se o .NET foi instalado corretamente.

```
C:\Users>dotnet -version
```

Comando de prompt 1 – Verificação da instalação Windows
Fonte: Elaborado pelo autor (2024)

Resultado

Figura 5 – Verificação instalação Windows

3.1.2 Mac

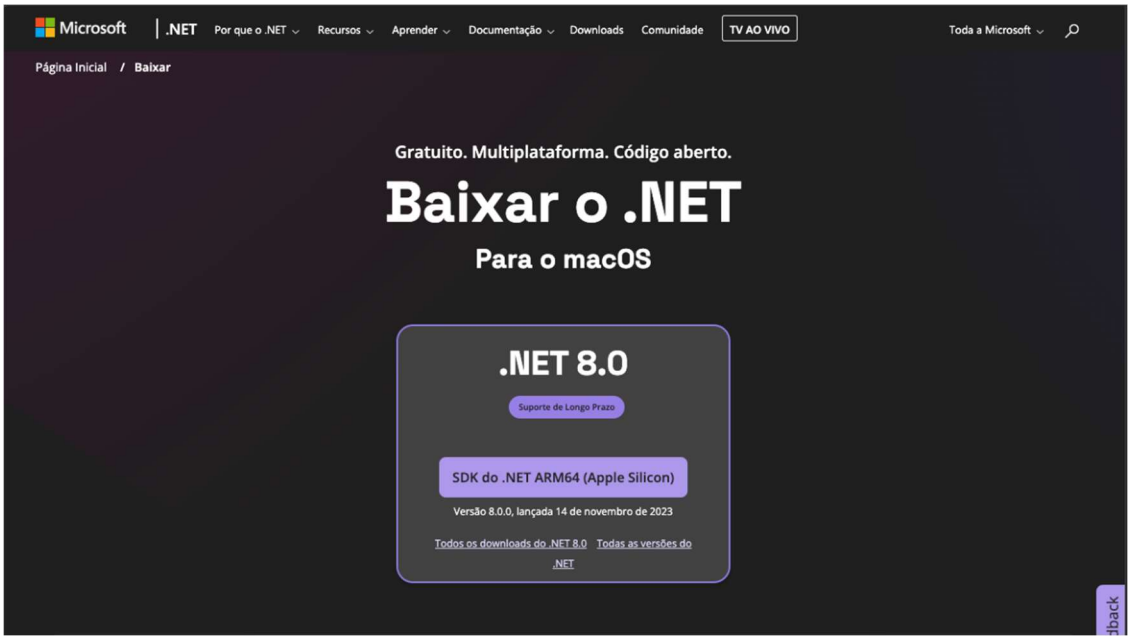

Figura 6 – Site para download .Net no macOS

- **Visite o Site Oficial:** Acesse o site do .NET *https://dotnet.microsoft.com/pt-br/downloads* e
- **Escolha a Versão para macOS:** Selecione a versão apropriada para o seu Mac. Isso geralmente será claramente marcado na página de download.
- **Baixe o Instalador:** Baixe o pacote de instalação para **macOS**, que geralmente é um arquivo .pkg.
- **Instale o .NET:** Abra o arquivo baixado e siga as instruções para instalar o .NET no seu Mac.

Figura 7 – Tela inicial da instalação no macOS

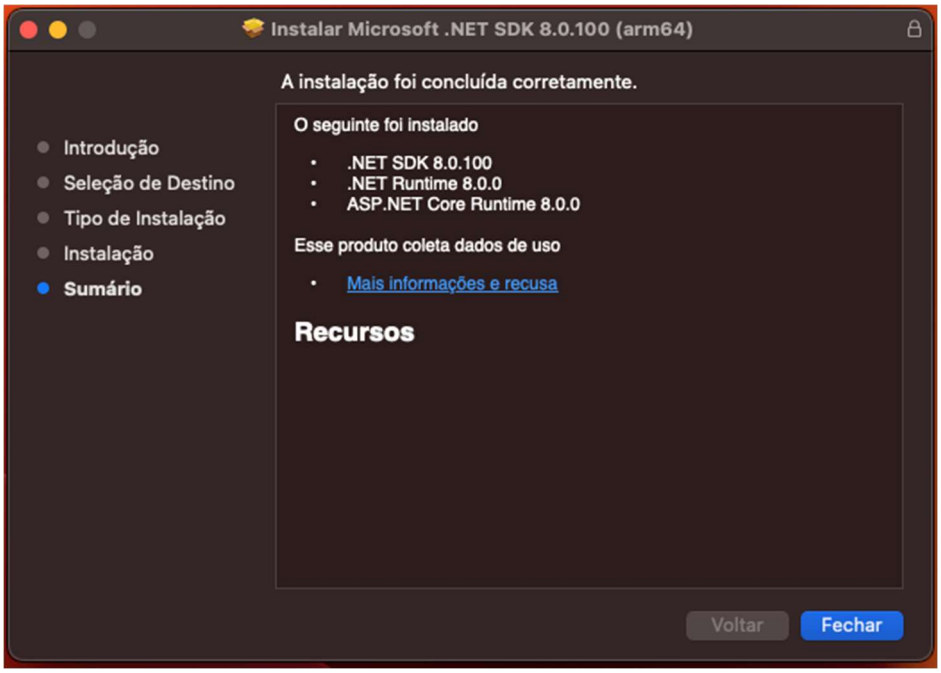

Figura 8 – Tela de conclusão da instalação macOS

- **Verifique a Instalação:** Abra o Terminal e digite *dotnet --version* para confirmar a instalação.

```
~]# dotnet -version
```

Comando de prompt 2 – Verificação da instalação macOS
Fonte: Elaborado pelo autor (2024)

Resultado

Figura 9 – Verificação instalação macOS

3.1.3 Linux

- **Abra o Terminal**: Inicie seu terminal.
- **Adicione o Repositório do Microsoft Package**: Dependendo da sua distribuição, você precisará adicionar o repositório do Microsoft Package à sua lista de fontes.
- **Atualize o Gerenciador de Pacotes**: Execute um comando de atualização (por exemplo, *sudo apt-get update* para distribuições baseadas em Debian).
- **Instale o .NET SDK ou Runtime**: Instale o .NET SDK para desenvolvimento ou apenas o runtime se você só precisar executar aplicações .NET. Use o comando apropriado, como *sudo apt-get install dotnet-sdk-6.0* (altere o número da versão conforme necessário).
- **Verifique a Instalação**: No terminal, digite *dotnet --version* para verificar a instalação.

4 INSTALACAO DO VISUAL STUDIO WINDOWS

Iremos instalar o Visual Studio Community que é uma versão gratuita, rica em recursos e extensível do Visual Studio, uma suíte de desenvolvimento integrada (IDE) criada pela Microsoft. É destinado a desenvolvedores individuais, pequenas equipes, estudantes e entusiastas de programação, oferecendo uma plataforma robusta para desenvolvimento de software em diversas linguagens de programação.

4.1 Passo a Passo para instalação da IDE Visual Studio Community

4.1.1 Download do Instalador

Para obter a versão mais recente e autêntica do Visual Studio, visite o site oficial da Microsoft em https://visualstudio.microsoft.com/pt-br/vs/community/. Este link o direcionará para a edição "Community" do Visual Studio, que é uma versão gratuita, rica em recursos e ideal para desenvolvedores individuais, acadêmicos, pesquisadores e projetos de código aberto.

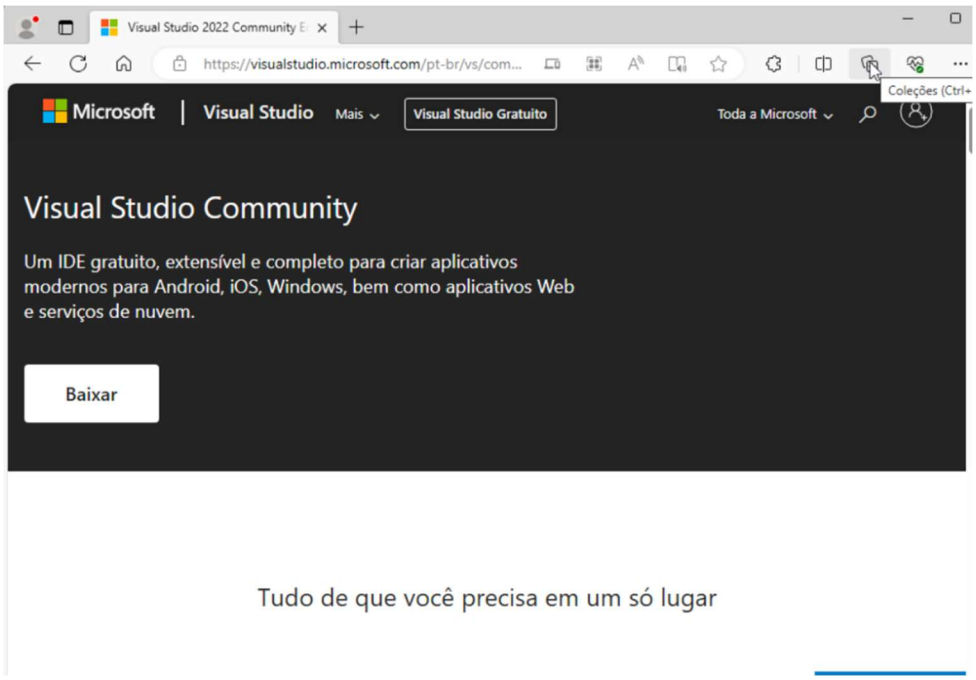

Figura 10 – Site para download do Visual Studio

4.1.2 Instalação

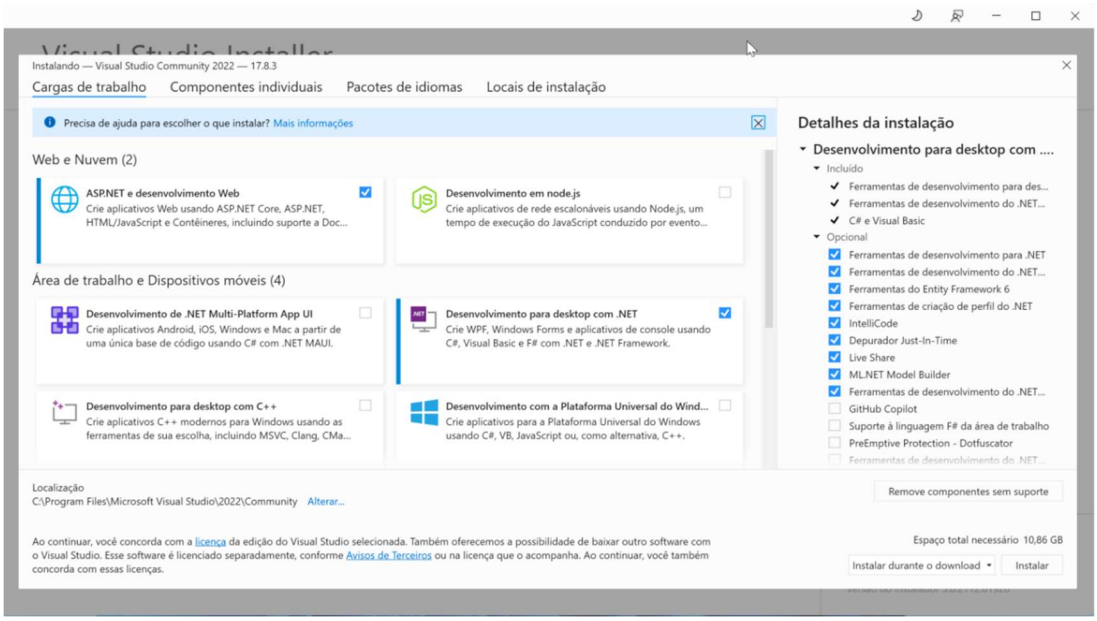

Figura 11 – Cargas de Trabalho

Durante a instalação, você pode personalizar seu ambiente de desenvolvimento selecionando as "Cargas de trabalho" desejadas. Estas são conjuntos de ferramentas e recursos relacionados a tipos específicos de projetos, como desenvolvimento web ou para dispositivos móveis.

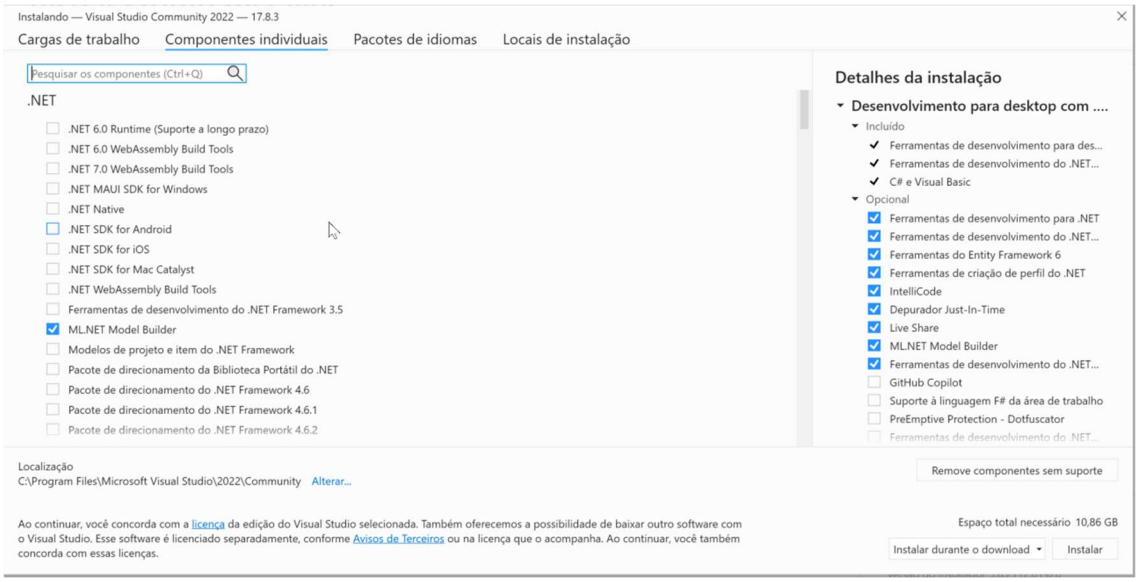

Figura 12 – Componentes individuais

Você também pode escolher "Componentes individuais" se precisar de ferramentas ou SDKs específicos não incluídos nas cargas de trabalho padrão. Por exemplo, você pode selecionar o ".NET 6.0 Runtime" ou o "ML.NET Model Builder" dependendo das suas necessidades de desenvolvimento.

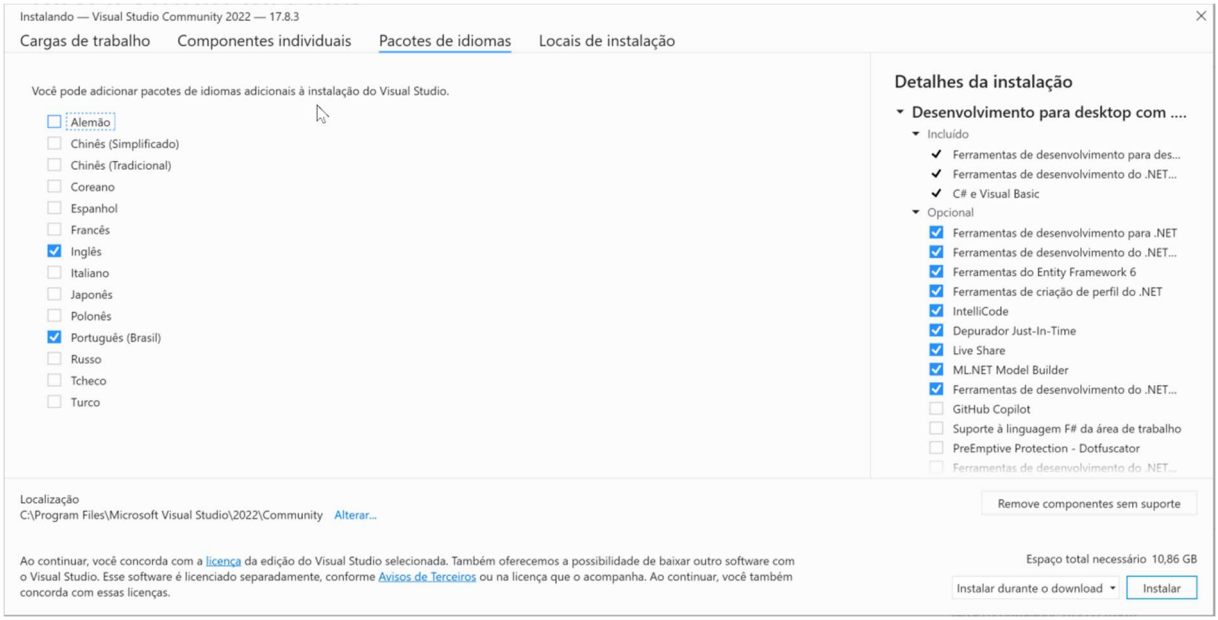

Figura 13 – Pacotes de idiomas

Outra opção disponível é a seleção de "Pacotes de idiomas". O Visual Studio suporta vários idiomas, e você pode marcar aqueles que deseja instalar, como inglês ou português (Brasil).

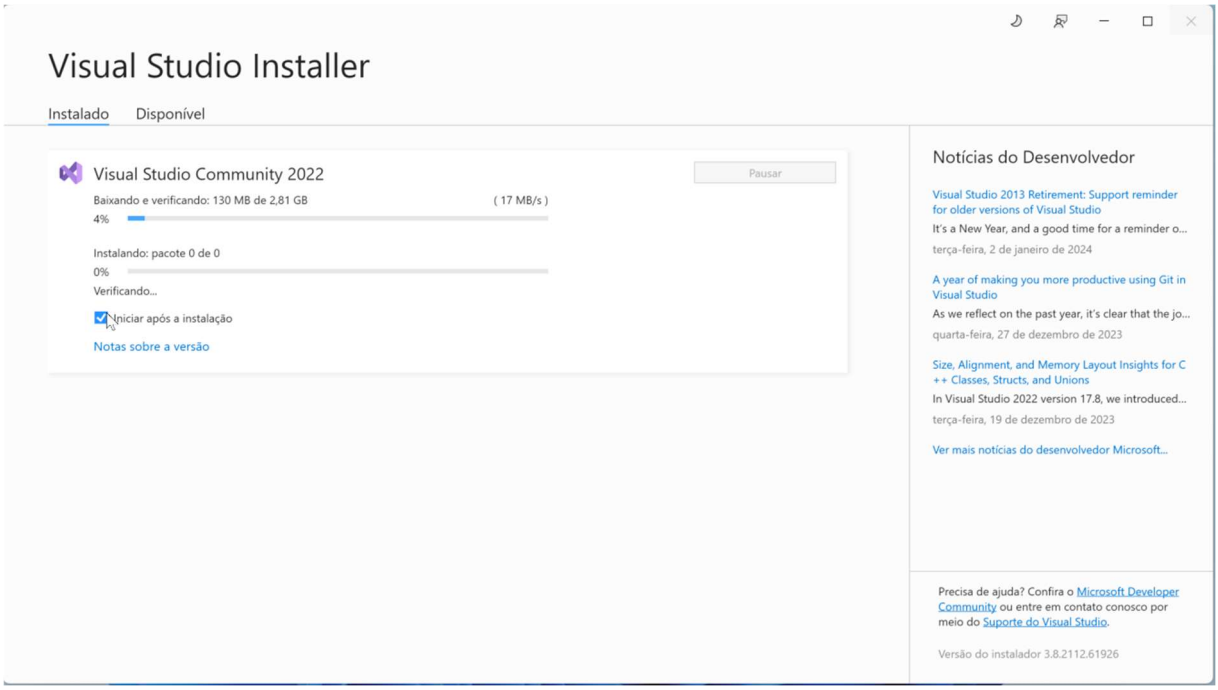

Figura 14 – Progresso da instalação

Após o início da instalação do Visual Studio, teremos uma tela inicial que mostra o progresso do download. Por exemplo, você pode ver algo como "Baixando e verificando: 130 MB de 2,81 GB (4%)" e "Instalando: pacote 0 de 0 (0%)". Isso indica que a instalação está em progresso. É possível também observar uma opção para "Iniciar após a instalação", o que permite que o Visual Studio seja aberto automaticamente após a conclusão da instalação.

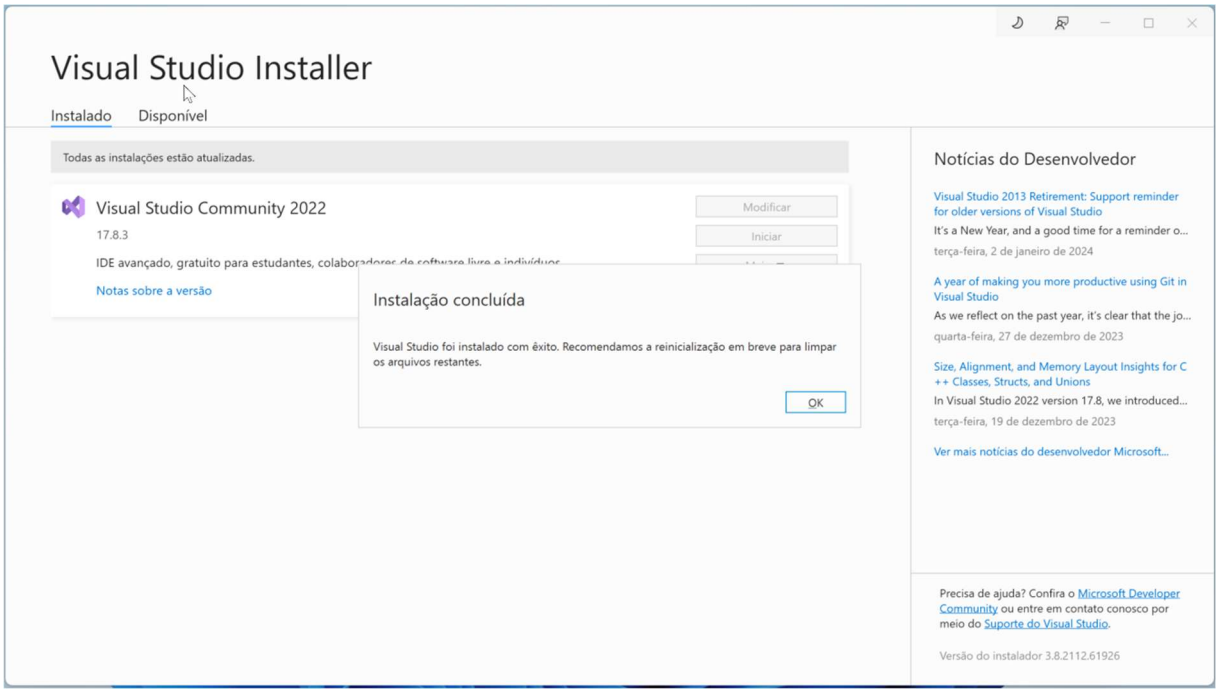

Figura 15 – Instalação Concluída

Após a instalação dos componentes selecionados, o Visual Studio Installer apresentará uma tela confirmando o sucesso do processo. A mensagem "Instalação concluída" indica que o Visual Studio Community 2022 versão XX.XX foi instalado corretamente em seu sistema. O instalador recomenda a reinicialização do computador para limpar quaisquer arquivos temporários ou restantes que foram usados durante a instalação. É uma boa prática seguir esta recomendação para garantir que todas as configurações sejam aplicadas corretamente e para evitar potenciais conflitos com arquivos antigos.

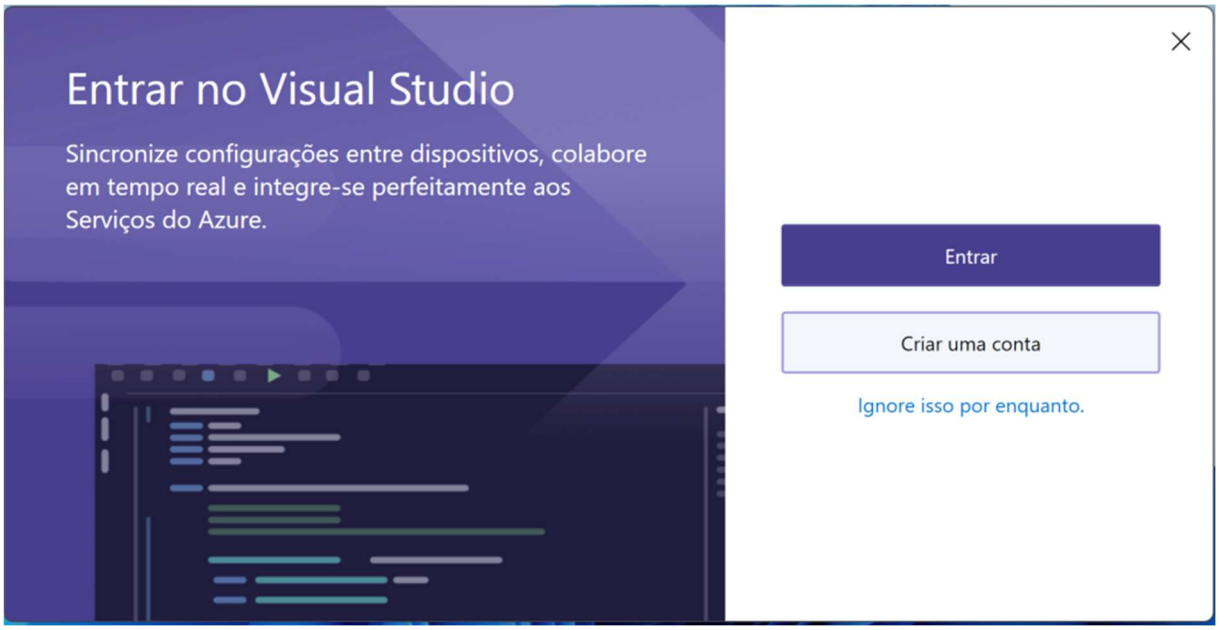

Figura 16 – Login Visual Studio

O próximo passo é a tela de login, que permite "Entrar no Visual Studio". Fazer login sincroniza as configurações entre dispositivos, colabora em tempo real e integra-se aos Serviços do Azure.

Opções de Login

- **Entrar:** Se você já tem uma conta da Microsoft, pode usar suas credenciais para entrar e sincronizar suas configurações.
- **Criar uma conta:** Se você é novo no Visual Studio ou não tem uma conta da Microsoft, pode criar uma conta.
- **Ignorar isso por enquanto:** Se você preferir não entrar imediatamente, pode optar por ignorar o login e continuar para a IDE.

Fazer login é altamente recomendado para aproveitar os recursos de colaboração e serviços em nuvem, mas não é obrigatório para começar a usar o Visual Studio.

4.1.3 Personalização do Visual Studio

Após a instalação, o Visual Studio convida você a "Personalizar sua experiência" para otimizar o layout e os atalhos de teclado de acordo com seu fluxo de trabalho. Você pode selecionar um tema de cores que se adapte ao seu estilo, com opções como "Escuro", "Claro", "Azul" e "Azul (Contraste Extra)". Essa personalização inicial ajuda a criar um ambiente de desenvolvimento confortável e produtivo, e as configurações podem ser alteradas a qualquer momento.

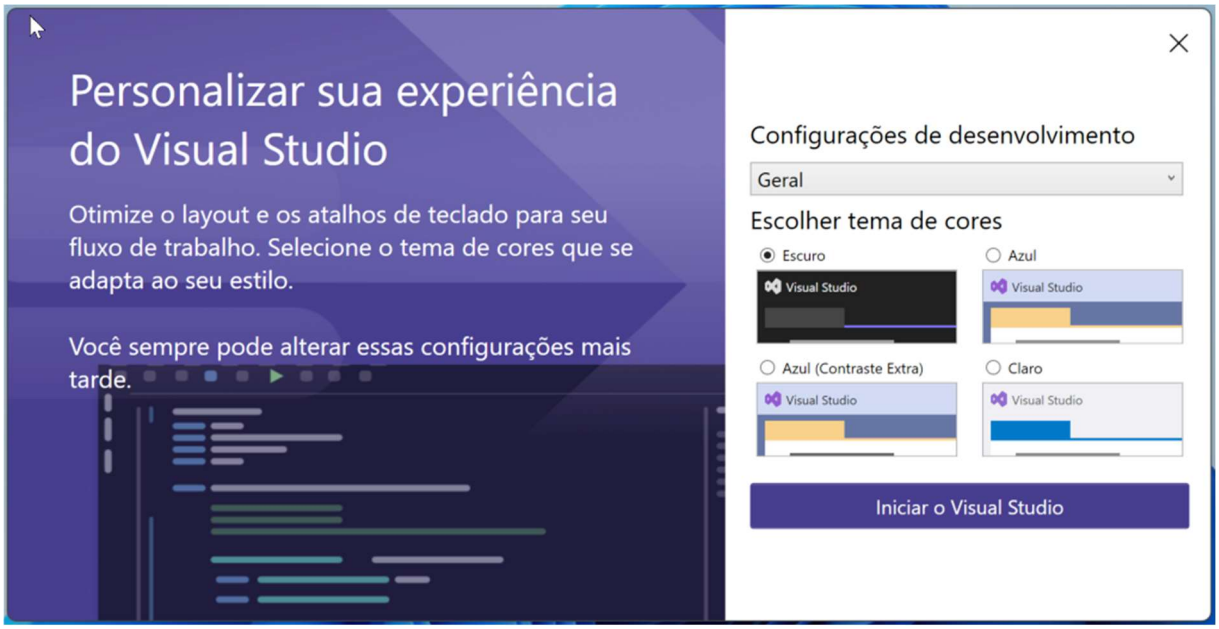

Figura 17 – Personalização Visual Studio

Escolhendo um Tema de Cores

- **Escuro**: Ideal para desenvolvedores que preferem um tema que seja fácil para os olhos, especialmente em ambientes com pouca luz.
- **Azul**: Uma variante do tema escuro com um toque de cor azul, oferecendo um visual distinto.
- **Azul (Contraste Extra):** Semelhante ao tema azul, mas com maior contraste para melhorar a legibilidade.

- **Claro**: Para aqueles que preferem um fundo claro, este tema oferece um design mais tradicional.

Depois de selecionar o tema desejado, você pode iniciar o Visual Studio clicando no botão "Iniciar o Visual Studio". Parabéns, o Visual Studio foi instalado com sucesso. Em breve, exploraremos as funcionalidades e ferramentas desta poderosa IDE."

Figura 18 – Site Visual Studio Code

4.2 Alternativas para o Visual Studio

O Visual Studio da Microsoft é amplamente reconhecido como uma das **IDEs** (Integrated Development Environments) mais poderosas e abrangentes disponíveis para desenvolvedores. No entanto, dependendo das necessidades

específicas do projeto, preferências pessoais ou restrições de orçamento, você pode estar buscando alternativas. Aqui falaremos somente de duas alternativas, mas lembrando que voce pode utilizar qualquer editor de texto para o desenvolvimento

4.2.1 Visual Studio Code

O Visual Studio Code, comumente conhecido como VS Code, é um editor de código-fonte desenvolvido pela Microsoft. Disponível gratuitamente, o VS Code se tornou excepcionalmente popular entre desenvolvedores devido à sua leveza, eficiência e capacidade de suporte a uma ampla gama de linguagens de programação e plataformas. Baixe em: https://code.visualstudio.com/

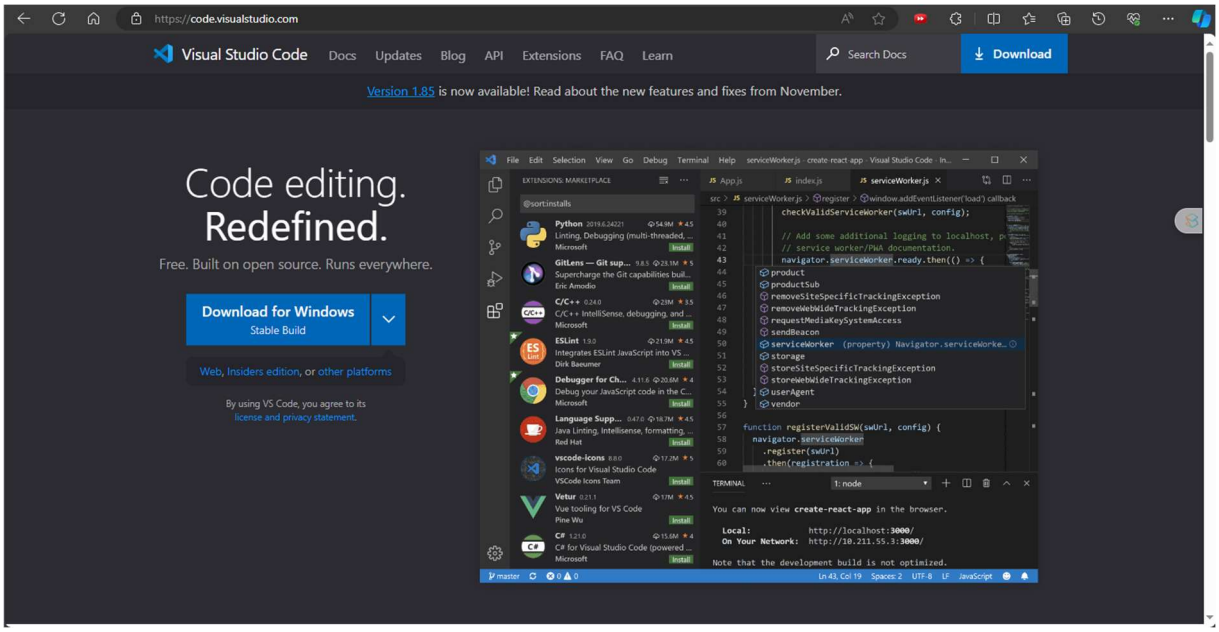

Figura 19 – Site Visual Studio Code

4.2.1.1 Características Principais

- **Multiplataforma:** Funciona em Windows, Linux e macOS, tornando-o acessível para uma ampla base de usuários.

- **Suporte de Linguagem**: Oferece suporte para uma vasta gama de linguagens, incluindo, mas não se limitando a Python, Java, JavaScript, C++, Go, PHP e mais.
- **Extensões e Personalização**: Uma das maiores vantagens do VS Code é sua capacidade de personalização. Através de um extenso mercado de extensões, os usuários podem adicionar funcionalidades, temas e ferramentas de acordo com as necessidades específicas.
- **Integração com Git**: O VS Code possui uma excelente integração com o Git, oferecendo recursos nativos para controle de versão.
- **Depuração**: Possui recursos de depuração integrados, que são facilmente configuráveis e potentes.
- **Terminal Integrado**: Permite o acesso a um terminal diretamente na interface do usuário, facilitando a execução de comandos e scripts.
- **Interface Customizável**: A interface do usuário é altamente customizável, permitindo aos usuários ajustar o layout, os temas e os atalhos de acordo com suas preferências.

4.2.1.2 Principais Tipos de Plugins

- **Suporte a Linguagens de Programação**: Existem plugins para praticamente todas as linguagens de programação populares, como Python, JavaScript, Java, C#, PHP, Ruby, Go e muitas outras. Essas extensões oferecem funcionalidades como realce de sintaxe, snippets de código, intellisense (completamento de código inteligente), e refatoração de código.
- **Ferramentas de Desenvolvimento Web**: Para desenvolvedores web, existem extensões que melhoram o trabalho com HTML, CSS, JavaScript, frameworks como Angular, React, Vue.js, e bibliotecas como jQuery. Estes plugins podem oferecer desde a formatação automática de código até a visualização ao vivo de páginas web.

- **Integração com Controle de Versão**: Extensões como o GitLens aprimoram a integração com sistemas de controle de versão, fornecendo insights detalhados sobre mudanças no código, histórico de commits e muito mais, diretamente no editor.
- **Ferramentas de Colaboração**: Plugins como o Live Share permitem a colaboração em tempo real entre desenvolvedores, possibilitando o compartilhamento de sessões de codificação, terminais, depurações e muito mais.
- **Ferramentas de Automação e Linting**: Extensões como ESLint, Prettier e outras ferramentas de linting e formatação ajudam a manter a qualidade e a consistência do código, além de automatizar tarefas repetitivas.
- **Suporte para Desenvolvimento em Containers**: Plugins como o Docker e o Kubernetes facilitam o trabalho com containers, permitindo gerenciar, implantar e depurar aplicações containerizadas diretamente do VS Code.
- **Customização da Interface**: Existe uma ampla gama de temas e ícones que os usuários podem instalar para personalizar a aparência do editor, melhorando a experiência visual e de usabilidade.

4.2.1.3 Por que o utilizar?

O VS Code é ideal para desenvolvedores que procuram um editor leve, mas poderoso, que possa ser adaptado para diferentes linguagens de programação e frameworks. É particularmente popular entre desenvolvedores web e aqueles que trabalham em projetos que requerem múltiplas linguagens. Além de ser um editor leve e poderoso, destaca-se pela sua incrível adaptabilidade proporcionada por um vasto ecossistema de plugins (extensões). Estas extensões permitem que os desenvolvedores personalizem e ampliem as funcionalidades do editor de acordo com suas necessidades específicas em diferentes linguagens de programação e frameworks.

4.2.2 Rider

O Rider é uma IDE (Integrated Development Environment) da JetBrains focada em desenvolvimento .NET. É uma opção paga, mas oferece um conjunto robusto de ferramentas e funcionalidades que a tornam uma escolha atraente para desenvolvedores profissionais. Baixe a avaliação gratuita por 30 dias em: https://www.jetbrains.com/pt-br/rider/

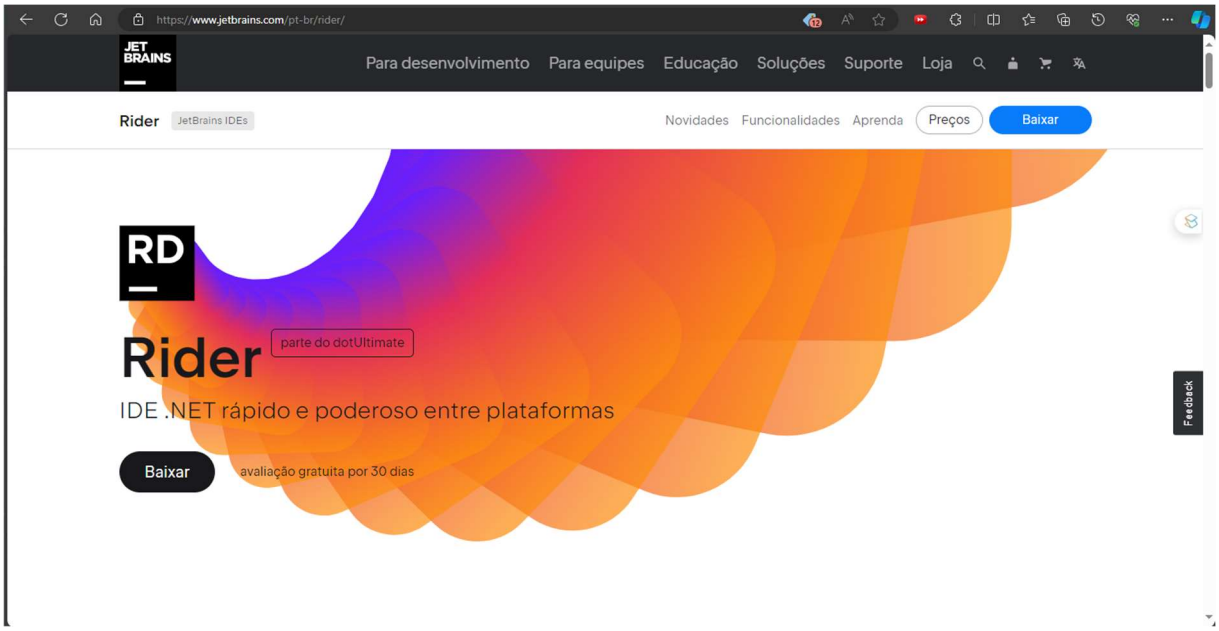

Figura 20 – Site Jetbrains Rider

4.2.2.1 Características Principais

- **Suporte .NET:** O Rider oferece suporte abrangente para desenvolvimento .NET, incluindo aplicações .NET Framework, .NET Core.
- **Compatibilidade com Plataformas:** Disponível para Windows, macOS e Linux, oferece um ambiente de desenvolvimento consistente em diferentes sistemas operacionais.

- **Integração com Ferramentas JetBrains**: Integra-se perfeitamente com outras ferramentas da JetBrains, como ReSharper, oferecendo recursos avançados de refatoração e análise de código.
- **Suporte a Múltiplas Linguagens**: Além do C#, o Rider suporta outras linguagens como F#, VB.NET, JavaScript, TypeScript, SQL, entre outras.
- **Ferramentas de Depuração e Teste**: Possui ferramentas de depuração poderosas e suporte para várias estruturas de teste.
- **Interface de Usuário Intuitiva**: A interface do Rider é intuitiva e customizável, com suporte para temas e layouts diversos.
- **Integração com Controle de Versão**: Oferece suporte integrado para sistemas de controle de versão, incluindo Git.

4.2.2.1 Por que o utilizar?

O Rider é ideal para desenvolvedores que trabalham predominantemente com tecnologias .NET e buscam uma IDE que ofereça um suporte abrangente e funcionalidades avançadas de desenvolvimento. É uma escolha popular entre profissionais que desejam uma ferramenta robusta que ofereça alta produtividade e eficiência no desenvolvimento .NET.

5 CONCLUSÃO

Concluímos assim nosso guia de instalação e configuração do ambiente .NET. Foi um prazer guiá-lo por este processo essencial, proporcionando as bases para o seu crescimento e desenvolvimento em programação. Este guia é apenas o começo de uma jornada emocionante e enriquecedora no mundo do .NET. Aguardo ansiosamente por nossos próximos encontros nos capítulos seguintes, onde exploraremos ainda mais as capacidades e possibilidades deste

poderoso ambiente de desenvolvimento. Até lá, desejo-lhe uma experiência gratificante e inovadora com o .NET!

REFERÊNCIAS

MICROSOFT. Documentação sobre os conceitos básicos do .NET. Disponível em: < https://learn.microsoft.com/pt-br/dotnet/fundamentals/>. Acesso em: 07 jan. 2024.

MICROSOFT. **Política de Suporte do .NET e do .NET Core**. Disponível em: < https://dotnet.microsoft.com/pt-br/platform/support/policy/dotnet-core>. Acesso em: 07 jan. 2024.