



O Guia Definitivo para Sobreviver a Sistemas Operacionais

Este PDF é estruturado para proporcionar uma compreensão acessível e prática dos sistemas operacionais, abrangendo desde a teoria básica até exercícios práticos que reforçam o aprendizado.

Table of Contents

Sistemas Operacionais	2
Capítulo 1	3
1.1 O que os Sistemas Operacionais fazem	4
1.1.1 Hardware	7
1.1.2 Software	9
1.1.3 Visões do Sistema	11
1.2 Operação do Computador	14
1.3 Estrutura de Armazenamento	19
1.4 Estrutura de Entrada e Saída	27
1.5 Arquitetura do Sistema	34
1.6 Estrutura do sistema operacional	40
1.7 Operações do Sistema Operacional	42
Caching	44
Questões 1	51
Prática 1	58
Respostas	127

Sistemas Operacionais

Bem-vindo ao nosso guia sobre Sistemas Operacionais! Nesta obra, exploraremos os conceitos fundamentais que regem o funcionamento dos sistemas que permitem que nossos dispositivos funcionem de maneira eficaz. Os sistemas operacionais são uma peça crucial da tecnologia moderna, servindo como intermediários entre o hardware e o software, gerenciando recursos, permitindo a execução de aplicativos e garantindo uma experiência de usuário fluida.

Este PDF é estruturado para proporcionar uma compreensão acessível e prática dos sistemas operacionais, abrangendo desde a teoria básica até exercícios práticos que reforçam o aprendizado.

Como utilizar este material

Para aproveitar ao máximo este guia, recomendamos a seguinte abordagem:

- 1. Estude os conceitos:** Leia atentamente cada seção, focando na compreensão dos conceitos fundamentais. Não se preocupe se não entender tudo de imediato; os sistemas operacionais são um tema complexo que se torna mais claro com o tempo e a prática.
- 2. Pratique regularmente:** Utilize os exercícios práticos fornecidos para reforçar seu aprendizado. A experiência prática é essencial para solidificar o conhecimento teórico.
- 3. Resolva as questões:** Tente responder às questões propostas ao final de cada seção. Isso ajudará a avaliar sua compreensão e identificar áreas que podem precisar de revisão.
- 4. Explore além do material:** Encorajamos você a pesquisar tópicos adicionais que despertem seu interesse. A área de sistemas operacionais é vasta e está em constante evolução.
- 5. Aplique o conhecimento:** Sempre que possível, relate o que você aprendeu com situações do dia a dia ou problemas reais de computação. Isso ajudará a contextualizar o conhecimento adquirido.

Capítulo 1

1.1 Introdução

Um **Sistema Operacional** é uma interface que interliga o **hardware** e o **software**, fazendo o gerenciamento dos dois mundos do computador.

- A inexistência, de um **SO** resulta em um impedimento de uma interação natural com o computador.

Como todo grande projeto complexo caímos em uma operação em que deve ser bem definida em como o **SO** irá trabalhar, quais são seus requisitos.

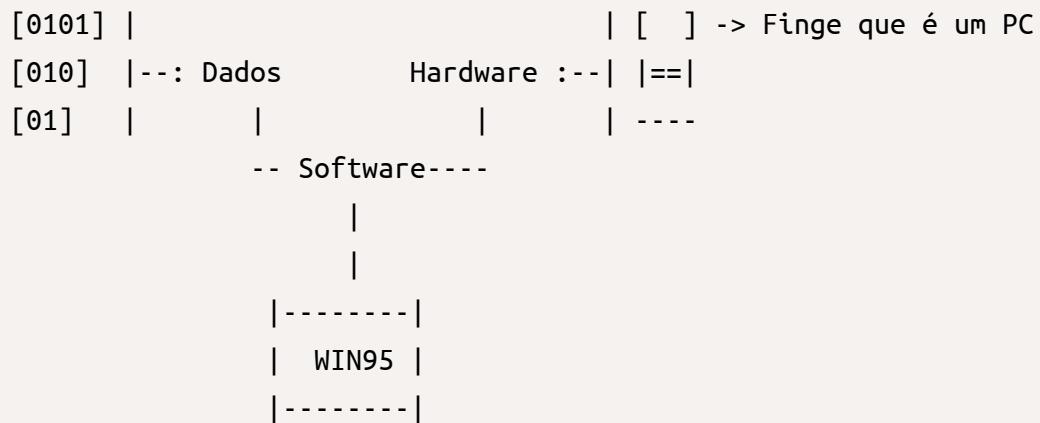
E o **SO** tem algumas funções básicas que é:

- dar partida no sistema
- gerenciar memória
- gerenciar dispositivos E/S

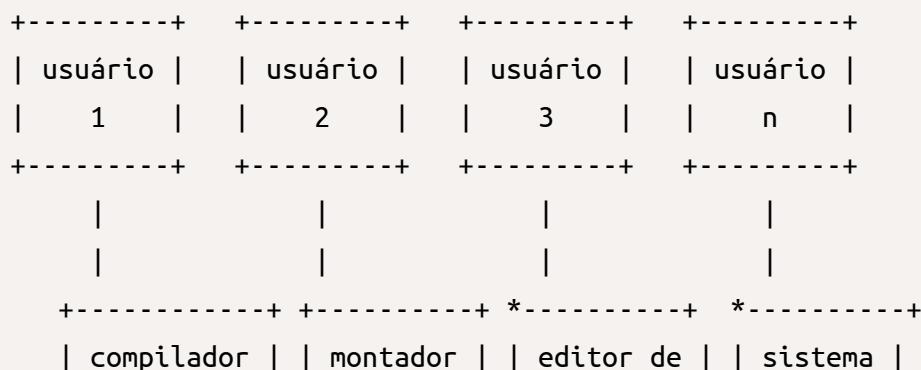
1.1 O que os Sistemas Operacionais fazem

Um **sistema computadorizado** ou só computador, pode ser *dividido em quatro partes*:

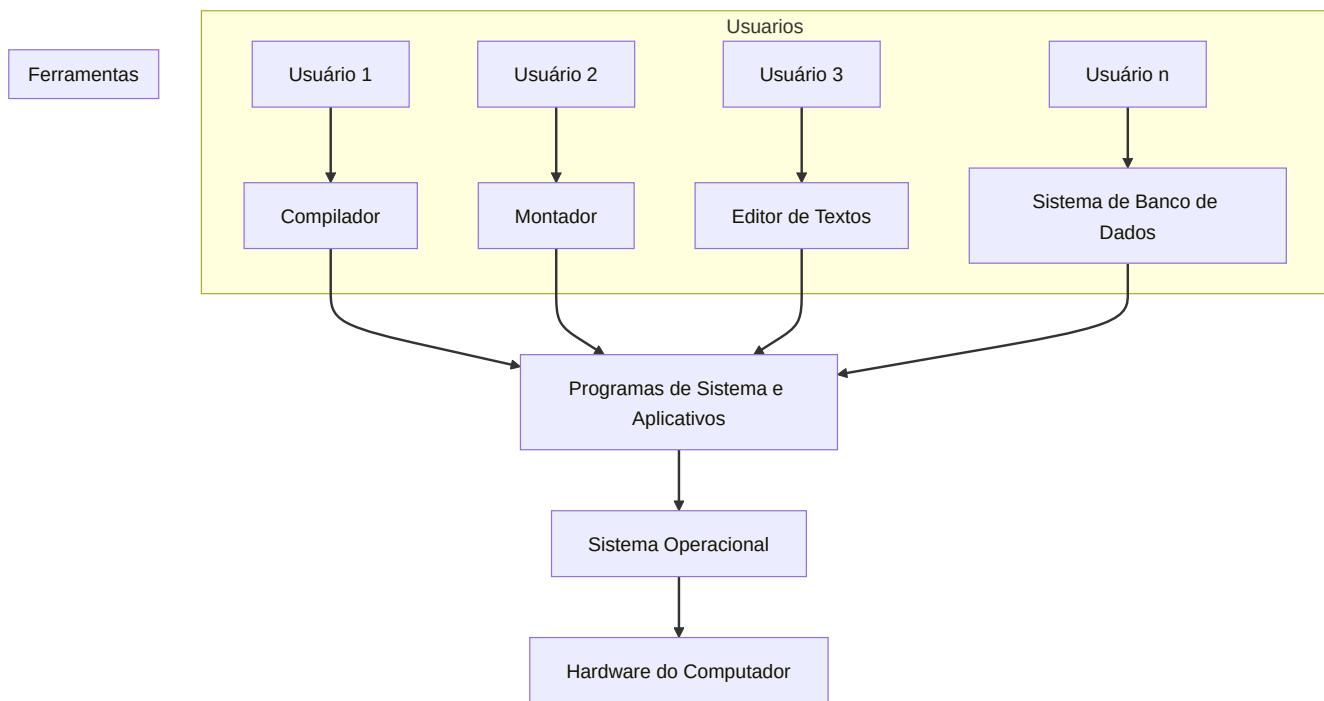
- Hardware ([1.1.1 Hardware](#))
 - Sistema Operacional
 - Software ([1.1.2 Software](#))
 - Usuários
- Também podemos considerar que um sistema computadorizado é composto por:



- Exemplo de Funcionamento de um Sistema Operacional



		textos	de banco
			de dados
+-----+ +-----+ +-----+ +-----+			
	+-----+		
	programas		
	de sistema		
	e aplicat-		
	ivos		
	+-----+		
	+-----+		
	sistema		
	operacional		
	+-----+		
	+-----+		
	hardware do		
	computador		
	+-----+		



1.1.1 Hardware

O hardware de um computador é como os blocos fundamentais de Minecraft que compõem o mundo do seu computador. Assim como você precisa de diferentes tipos de blocos para construir estruturas complexas em Minecraft, um computador precisa de vários componentes de hardware para funcionar.

Componentes Principais

Processador (CPU)

Pense no processador como o jogador em Minecraft. Assim como o jogador executa ações e toma decisões, a CPU processa instruções e realiza cálculos. É o cérebro do computador.

Memória RAM

A RAM é como o inventário do jogador em Minecraft. Ela armazena temporariamente informações que o processador precisa acessar rapidamente, assim como você mantém itens importantes no seu inventário para uso imediato.

Armazenamento (HDD/SSD)

O armazenamento é semelhante aos baús em Minecraft. HDDs e SSDs guardam dados a longo prazo, como programas e arquivos, assim como os baús armazenam itens que você não precisa carregar o tempo todo.

Placa-mãe

A placa-mãe é como o terreno em Minecraft onde você constrói. Ela conecta todos os outros componentes, permitindo que eles se comuniquem entre si.

Placa de Vídeo (GPU)

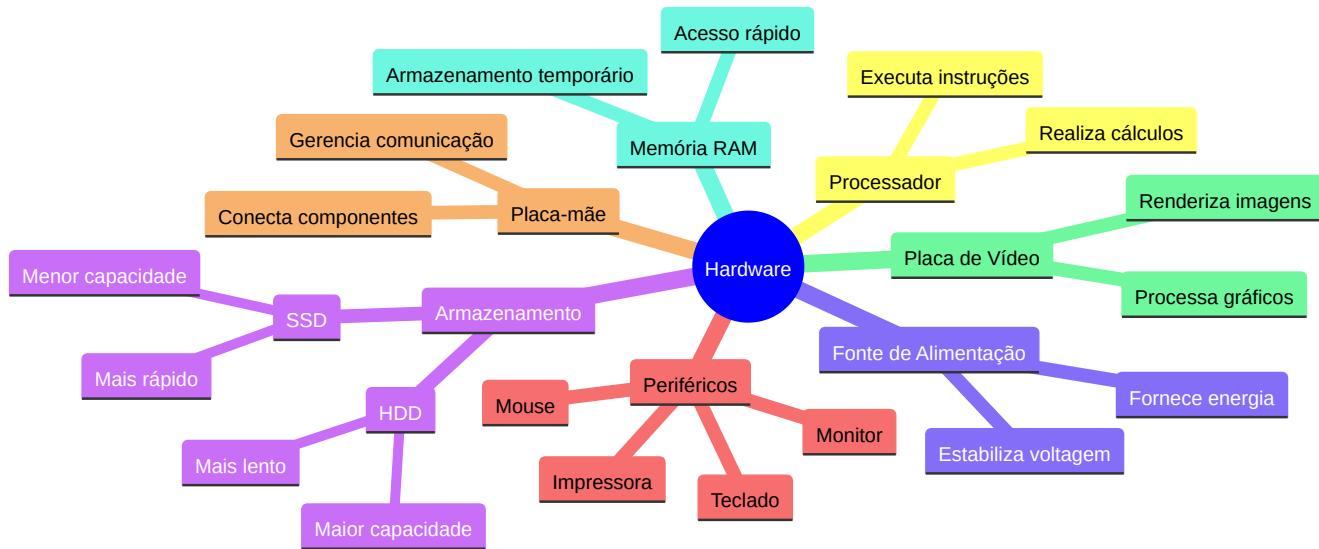
A GPU é como o mecanismo de renderização em Minecraft. Ela processa gráficos e imagens, tornando possível ver o mundo digital na sua tela.

Fonte de Alimentação

A fonte de alimentação é como a energia redstone em Minecraft. Ela fornece energia

para todos os componentes, mantendo tudo funcionando.

Mindmap do Hardware



Este mindmap ilustra os principais componentes de hardware de um computador, mostrando como eles se relacionam entre si, assim como diferentes estruturas em Minecraft se conectam para formar um mundo funcional.

Entender o hardware é essencial para compreender como os sistemas operacionais interagem com os componentes físicos do computador, gerenciando recursos e otimizando o desempenho, assim como um bom jogador de Minecraft gerencia seus recursos para construir e explorar eficientemente.

1.1.2 Software

Software é como o conjunto de regras e mecânicas que fazem o mundo de Minecraft funcionar. Assim como Minecraft tem diferentes tipos de mecânicas (como física, geração de mundo, interações de itens), um computador tem diferentes tipos de software que trabalham juntos para criar uma experiência funcional e interativa.

Tipos de Software

Sistema Operacional

O sistema operacional é como o modo de jogo em Minecraft (Sobrevivência, Criativo, etc.). Ele define as regras básicas de como o computador funciona e como os outros programas podem interagir com o hardware.

Aplicativos

Aplicativos são como os mods em Minecraft. Eles adicionam funcionalidades específicas ao sistema, permitindo que você realize tarefas como escrever documentos, navegar na internet ou editar imagens.

Drivers

Drivers são semelhantes aos comandos de bloco em Minecraft. Eles permitem que o sistema operacional se comunique com o hardware específico, assim como os comandos de bloco permitem interações complexas com o mundo do jogo.

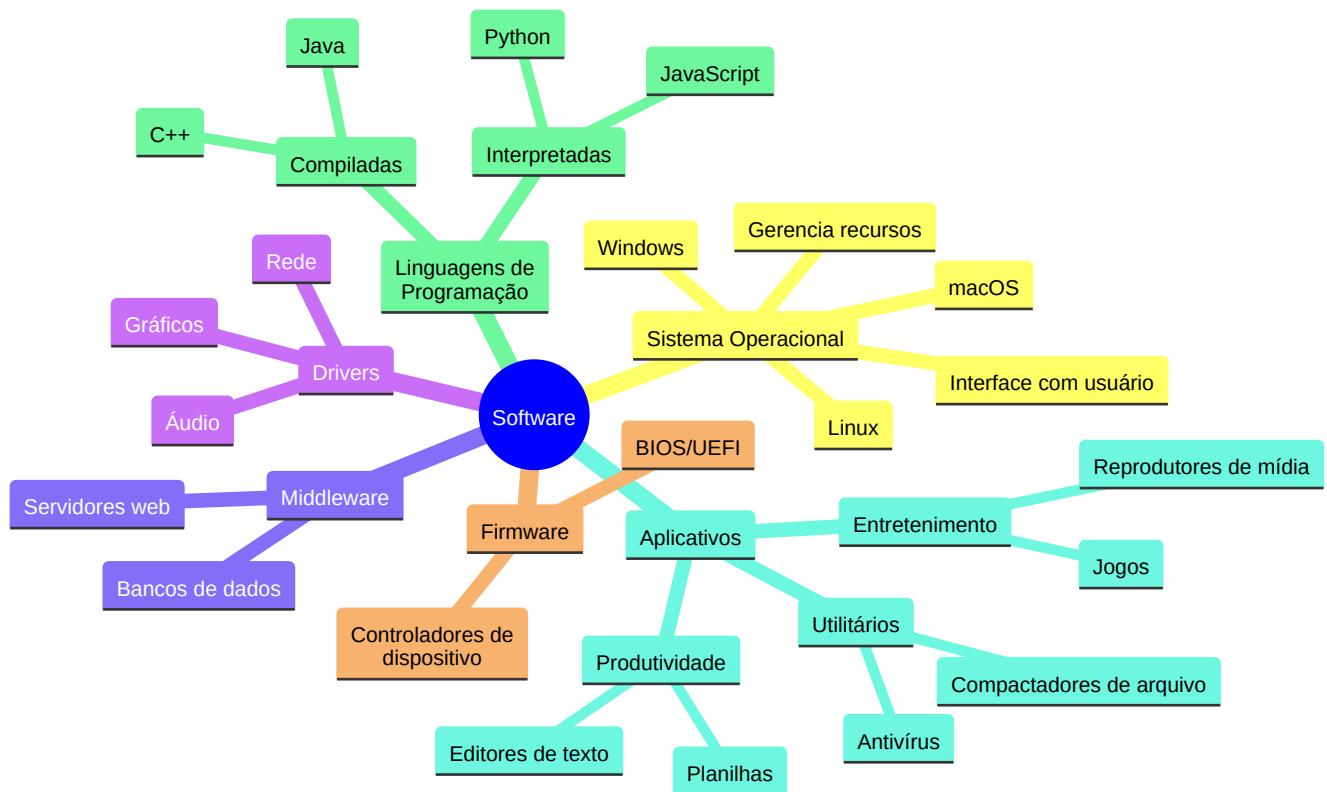
Firmware

O firmware é como as configurações internas dos blocos em Minecraft. É um software embutido no hardware que fornece instruções básicas para o funcionamento do dispositivo.

Linguagens de Programação

As linguagens de programação são como a linguagem de comandos em Minecraft. Elas permitem que os desenvolvedores criem software, assim como os comandos permitem aos jogadores criar comportamentos complexos no jogo.

Mindmap do Software



Este mindmap ilustra os principais tipos e categorias de software, mostrando como eles se relacionam e se organizam no ecossistema digital, assim como diferentes elementos se combinam para criar a experiência completa de Minecraft.

1.1.3 Visões do Sistema

Visão do Sistema: O Administrador do Servidor

Do ponto de vista do computador, o sistema operacional é como o administrador de um servidor Minecraft. Assim como um admin controla todos os aspectos do jogo, o sistema operacional gerencia intimamente o hardware do computador.

O Sistema Operacional como Alocador de Recursos

Imagine o sistema operacional como o sistema de plugins de um servidor Minecraft, responsável por gerenciar:

- 1. Tempo de CPU:** Como o dia e a noite no Minecraft, distribuindo tempo para cada processo.
- 2. Espaço de Memória:** Similar ao inventário dos jogadores, alocando espaço para programas.
- 3. Armazenamento de Arquivos:** Como baús no Minecraft, organizando e armazenando dados.
- 4. Dispositivos de E/S:** Portais para outros mundos, gerenciando a comunicação com dispositivos externos.

O sistema operacional deve alocar esses recursos de forma eficiente e justa, assim como um bom admin de Minecraft garante que todos os jogadores tenham acesso justo aos recursos do servidor.

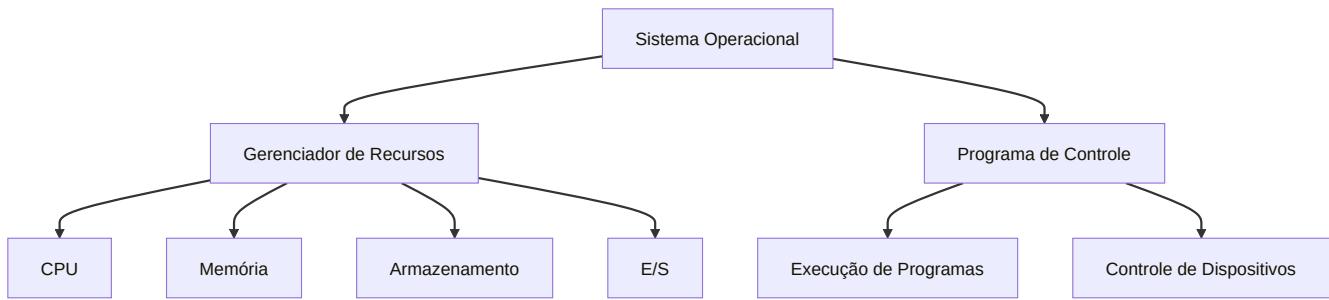
Unidades de Armazenamento: Os Blocos do Mundo Digital

- **Bit:** O bloco mais básico, como um grão de areia no Minecraft.
- **Byte:** 8 bits, como um bloco completo no Minecraft.
- **Word:** A unidade nativa do computador, como um chunk no Minecraft.
- **Kilobyte (KB):** 1.024 bytes, como uma pequena construção.

- **Megabyte (MB)**: 1.024² bytes, como uma vila inteira.
- **Gigabyte (GB)**: 1.024³ bytes, como um reino completo no Minecraft.

O Sistema Operacional como Programa de Controle

Assim como as regras e configurações de um servidor Minecraft, o sistema operacional controla a execução de programas e o uso de dispositivos para prevenir erros e uso indevido.



Este diagrama mostra como o Sistema Operacional, assim como o core de um servidor Minecraft, gerencia recursos e controla a execução de programas e dispositivos, mantendo todo o sistema funcionando harmoniosamente.

Visão do Usuário

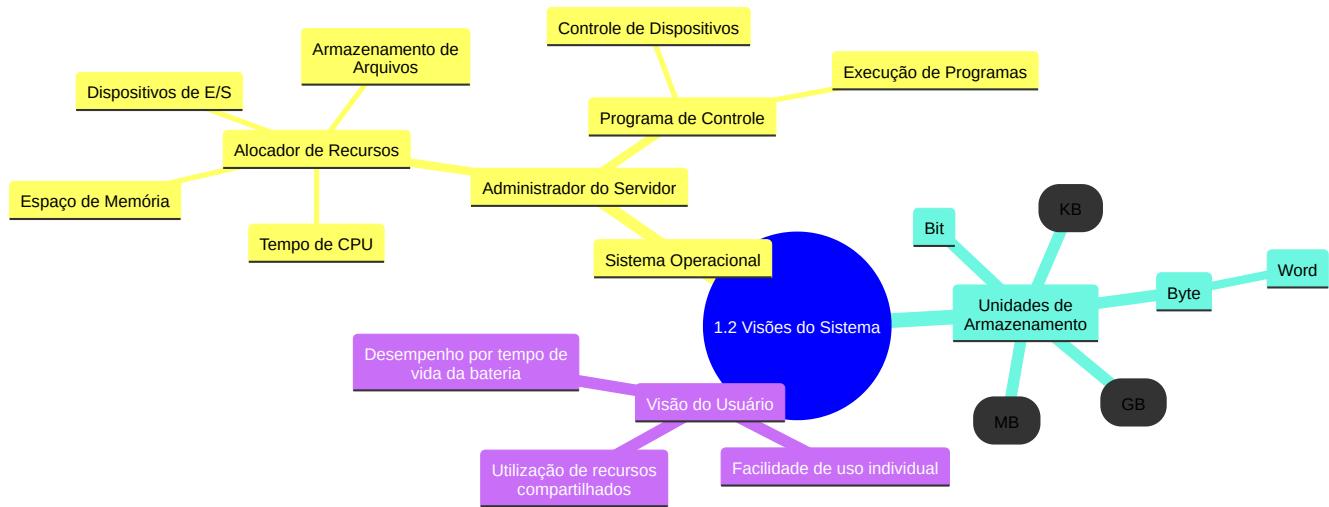
A visão do computador pelo usuário varia de acordo com a interface utilizada. Na maioria dos casos, os jogadores de Minecraft se sentam à frente de um computador, com um monitor, teclado, mouse e processador. Esse sistema foi projetado para que o jogador monopolize os recursos do computador.

O objetivo é proporcionar uma experiência mais rápida e imersiva no jogo. Nesse caso, o sistema operacional foi projetado principalmente para a facilidade de uso, com alguma atenção ao desempenho e pouca consideração à utilização de recursos – como a competição por espaço de memória e processamento.

É natural que o desempenho seja importante para o jogador; mas esses sistemas são otimizados para a experiência individual do jogador.

Em alguns casos, o jogador pode se conectar a um servidor remoto, permitindo que vários jogadores acessem o mesmo computador. Nesse caso, o sistema operacional foi projetado para um equilíbrio entre a facilidade de uso individual e a utilização de recursos compartilhados.

Alguns computadores podem ter pouca ou nenhuma visão do usuário. Por exemplo, os computadores embutidos nos consoles de jogos podem ter teclados numéricos e botões de luz indicadora, para mostrar o status do jogo, mas, em sua maioria, eles e seus sistemas operacionais são projetados para serem executados sem a intervenção do jogador.



1.2 Operação do Computador

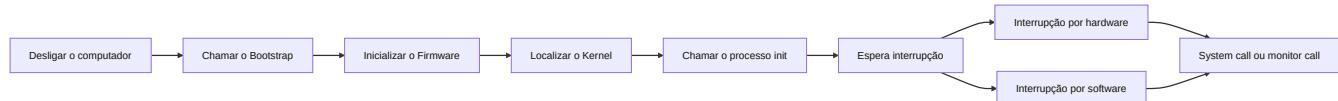
Ao desligar o computador e ligá-lo, o que acontece? Como ele "chama" o Sistema Operacional.

Para o computador começar a funcionar, ele chama um programa básico, chamado de **bootstrap**. Normalmente, este programa está alocado na memória apenas de leitura (ROM) ou é salvo na memória de somente leitura apagável programavelmente (**EEPROM**).

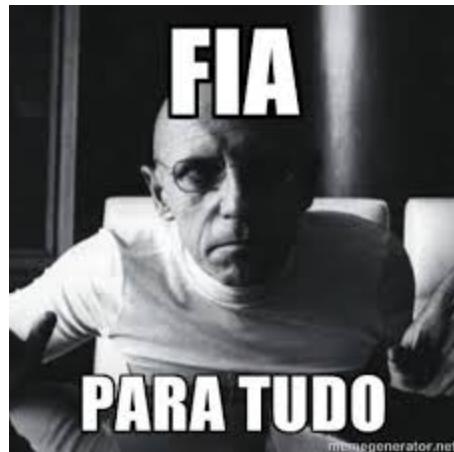
Este programa é conhecido como **Firmware**, pois está instalado diretamente no hardware, assim, ele inicializa todos os aspectos do sistema, desde os registradores da CPU até os dispositivos e o conteúdo na memória.

Para carregar o SO, ele precisa localizar o **Kernel**, que é o núcleo do sistema operacional. Assim que o Kernel é carregado na memória do computador, ele chama um processo chamado **init**, que espera uma interrupção do sistema ou do hardware. Os dois casos são:

- Se for pelo hardware, ele envia uma interrupção por sinal para a CPU, via normalmente o barramento do sistema;
- Se for por software, ele pode fazer de duas maneiras: chamando uma **system call** (chamada do sistema) ou usando um **monitor call** (monitor de chamada). Essas são operações especiais executadas para disparar uma interrupção, enviando um sinal para a CPU.



Quando a CPU recebe uma interrupção, ela para o que está fazendo e executa a rotina de tratamento correspondente:

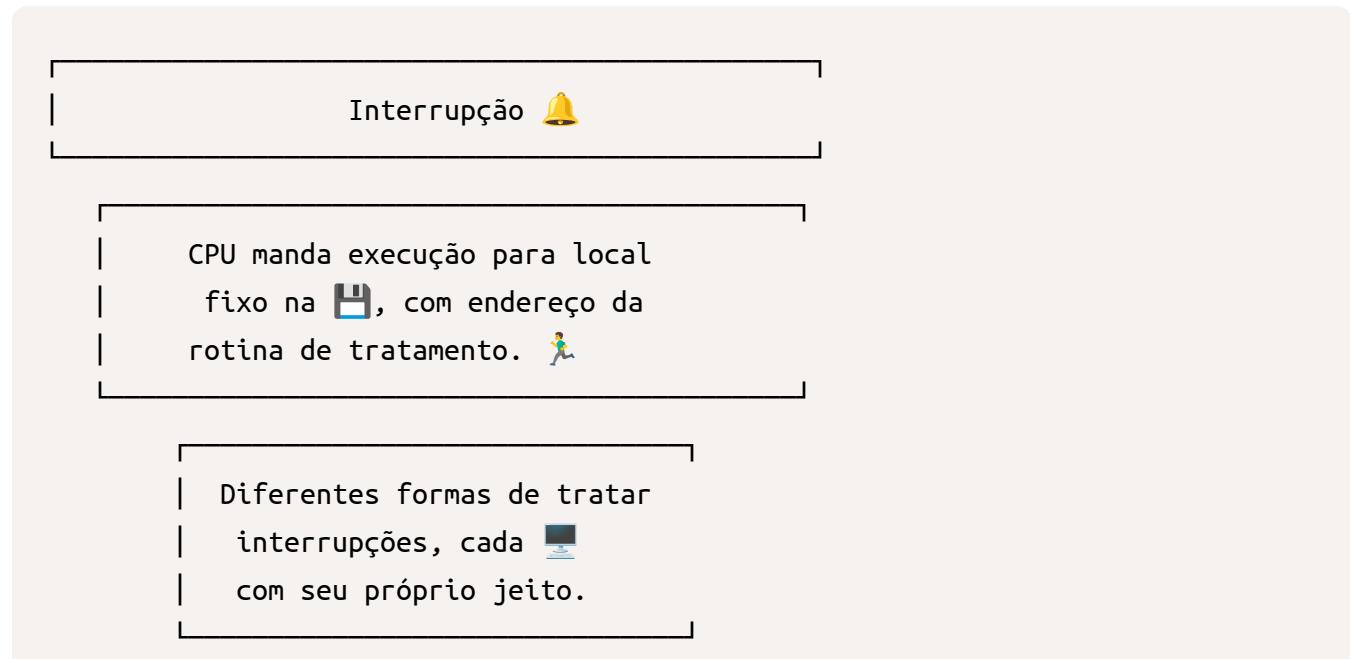


Meme fia para tudo

A CPU então manda a execução para uma **localização fixa na memória**, onde essa localização contém o **endereço inicial** da rotina para **atender a essa interrupção**.

Essas **interrupções** podem ser tratadas de diferentes maneiras, e cada computador possui seu próprio mecanismo. Um método simples para isso é tratar a transferência chamando uma rotina genérica. Para dar mais enfoque em velocidade pode ser usada uma **tabela de ponteiros a pontando para as interrupções**, já que elas devem ser predefinidas. **Essa tabela é armazenada em memoria baixa**, sendo ela a primeira parte ou locação da memoria.

Esse **vetor de interrupção** vai ser indexado exclusivamente pelo número do dispositivo, fornecido com a requisição da interrupção para gerar o endereço do tratamento da interrupção:



Método simples:
Transfere para uma
rotina genérica.  bootstrap

Método rápido:
 Tabela de
ponteiros para
interrupções, em
memória baixa. 

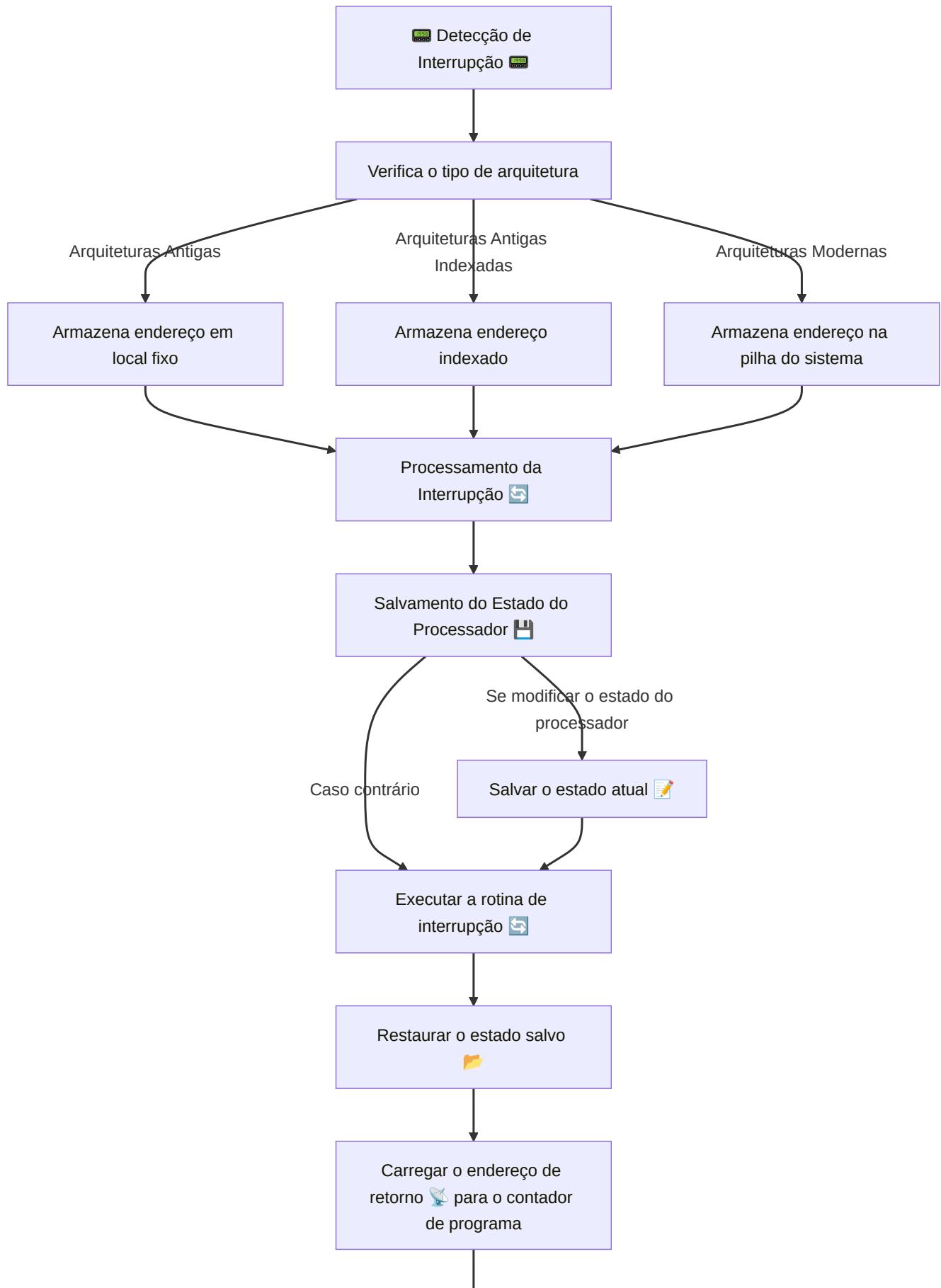
 Vetor usa
 dispositivo
para gerar
endereço do
tratamento. 

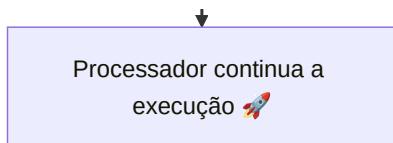
A arquitetura de interrupção **precisa salvar o endereço da instrução interrompida**, em projetos:

- Em alguns antigos armazenam o endereço da interrupção de **maneira fixa ou local indexado** por um numero do dispositivo;
- Em arquiteturas modernas, eles armazenam em **pilhas do sistema**;

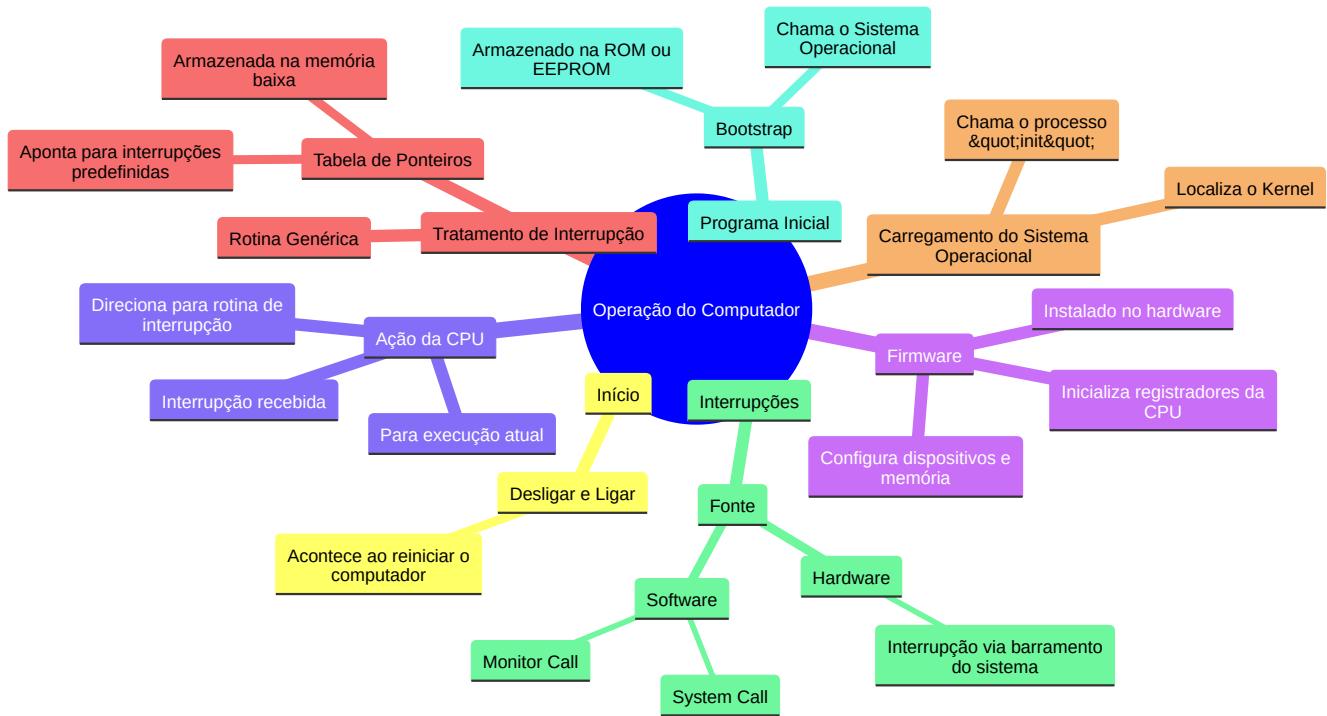
Se a rotina de interrupção precisar modificar algum estado do processador, por exemplo alterando os valores do **registrarador**:

- Ela vai **salvar** o estado atual, explicitamente;
- Depois **carregar e restaurar** esse estado para depois **retornar**;
- Em seguida será carregado para o **contador de programa** o **endereço do retorno** e o **processador** que foi **interrompido** continua como se nada tivesse acontecido;





Diagrama

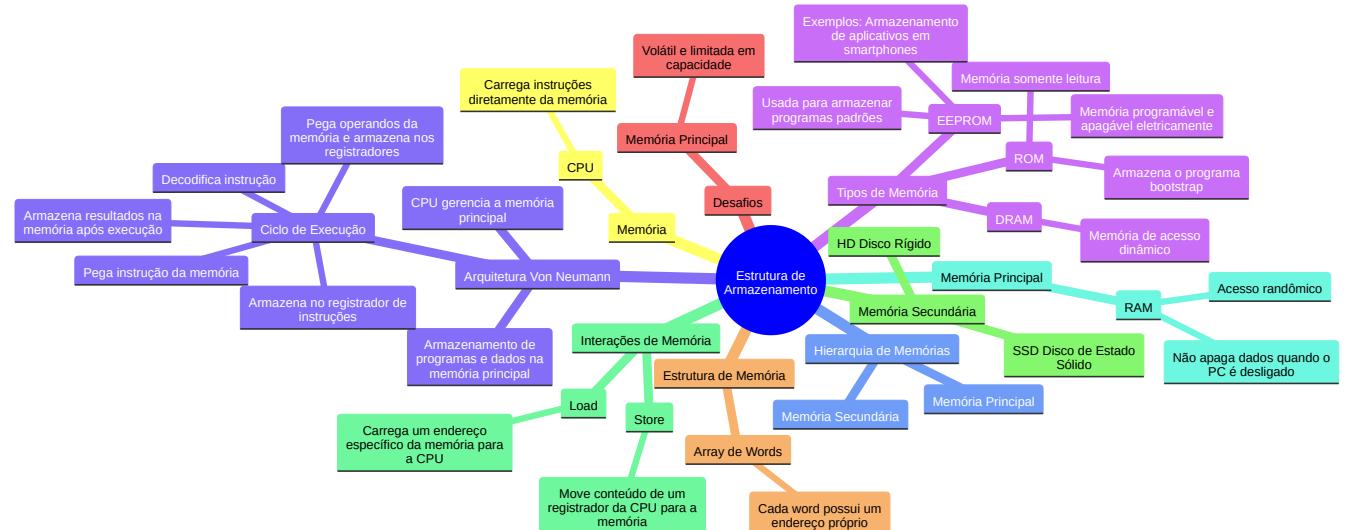


1.3 Estrutura de Armazenamento

Para os computadores que temos a **CPU** só consegue carregar instruções que vêm diretamente da memória.

- A memória não sendo nada, mas a **Memória Principal** - aquela cujo acesso é randômico, ou seja, desligar o PC não apaga os dados armazenados, que é a memória **RAM**.

Diagrama



i [🔗](#) Veja mais sobre tipos de memória em:

A memória RAM é comumente feita numa arquitetura de semicondutores chamada de **Dynamic Random Access Memory (DRAM)** ou, em português, **memória de acesso dinâmica**.

Um outro tipo de memória é aquela que só serve para leitura, assim como a mulher do seu amigo, apenas olhe. As conhecidas são:

- ROM (Read Only Memory) ==>** normalmente vem nos computadores e é usada para armazenar o programa bootstrap.

- Além disso, é usada por empresas de jogos para guardar os jogos, já que ela possui essa natureza imutável.
- **EEPROM** (Electrically Erasable Programmable Read Only Memory)
 - Por não ser modificado com frequência, essa memória costuma ser usada para armazenar programas padrões de modo estático.
 - Smartphones, por exemplo, utilizam a EEPROM de modo que as fabricantes armazenam nele os aplicativos de fábrica.

Quaisquer destas memórias utilizam **um array de words** ou **uma unidade de armazenamento**.

- Cada *word* possui seu próprio endereço.
- As interações se dão por instruções:
 - **load** - carrega um endereço específico da **memória principal** para um dos **registradores** da CPU.
 - **store** - move um conteúdo de um **registrador da CPU** para a **memória principal**.

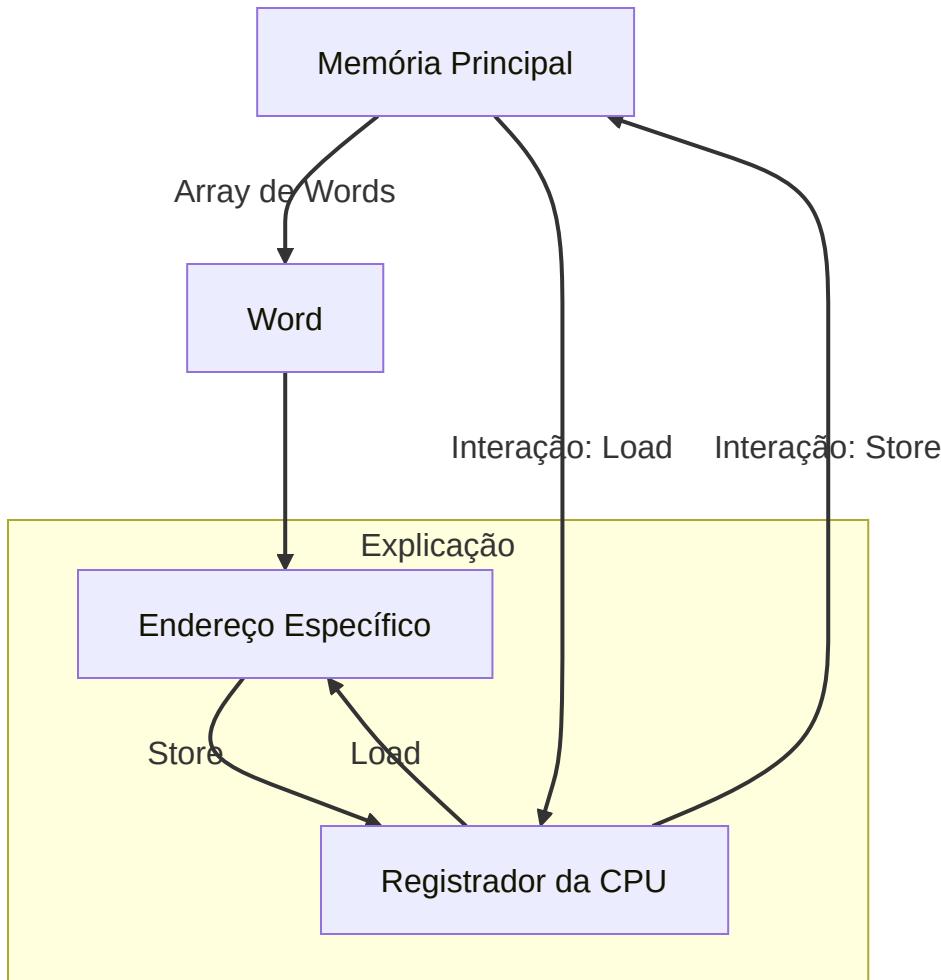


Ilustração de um esquema sobre instruções da CPU (`load` e `store`)

- A CPU carrega e armazena essas instruções tanto explicitamente (dizer para ela fazer) como de maneira automática - ela faz sozinha o carregamento da memória principal para serem executadas.

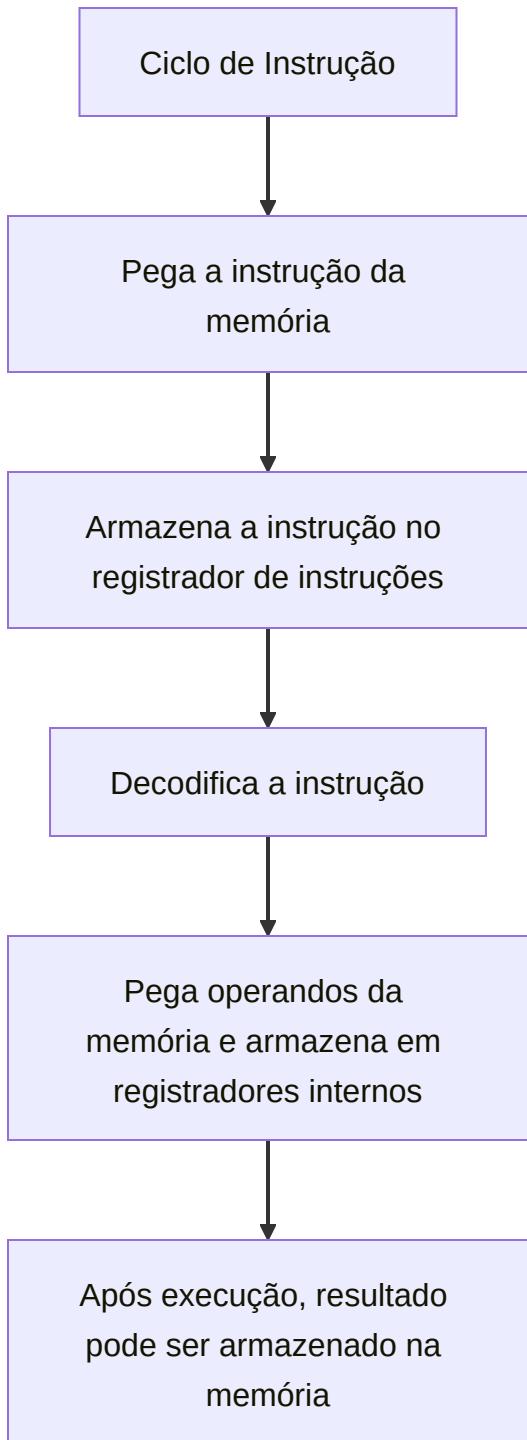
A arquitetura mais usada nos computadores modernos é a de **Von Neumann**. Essa arquitetura funciona da seguinte forma:

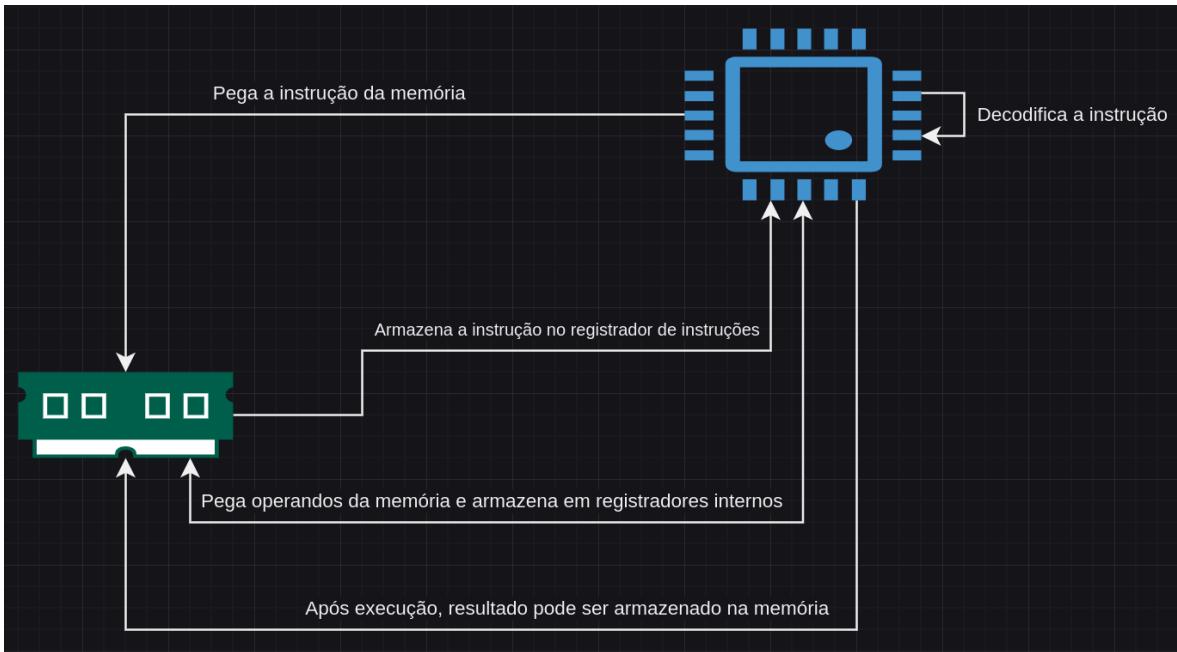
- Programas e dados são armazenados na memória principal.
- A CPU gerencia a memória principal.

Vamos para um ciclo de execução - quando uma instrução é dada:

1. Pega a instrução da memória.
2. Armazena essa instrução no **registrador de instruções**.
3. Essa instrução é então decodificada.
 1. Pode pegar operandos da memória e armazená-los em registradores internos.
4. Após a execução dos operandos, o resultado pode ser armazenado na memória.

Diagramas de Execução de Instrução





003 - Estrutura de Armazenamento

- i** A unidade de memória só consegue ver um fluxo de endereços de memória. Ela não sabe:
- Como são gerados (Gerados por contador de instruções, indexação, endereços literais e etc)
 - Para que servem
 - Se são instruções ou dados.

Seria bom, mas a vida não é um morango, a memória principal não consegue armazenar todos os dados e programas. Entretanto, não temos isso, já que:

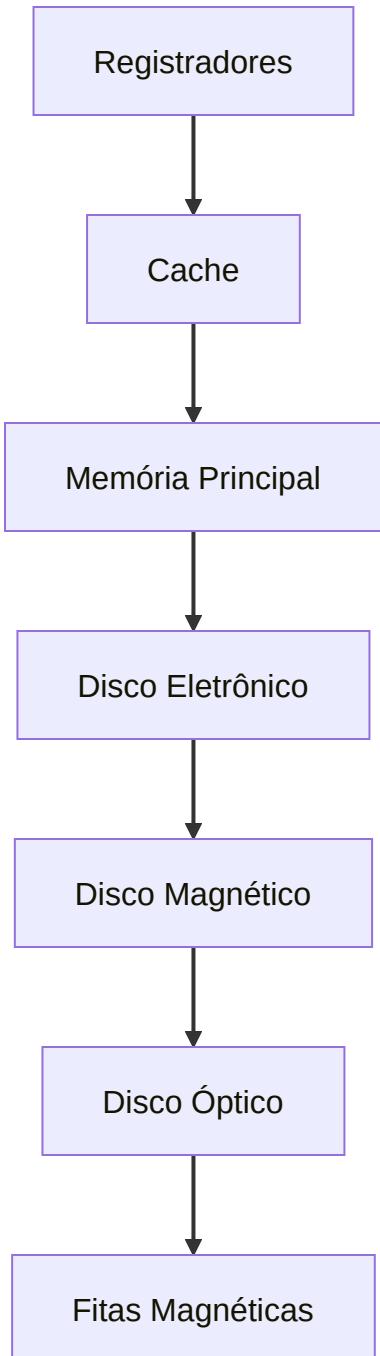
- A memória principal é volátil, ela perde os dados assim que a máquina é desligada.
- A memória principal possui um armazenamento irrisoriamente pequeno para armazenar todos os programas e dados.

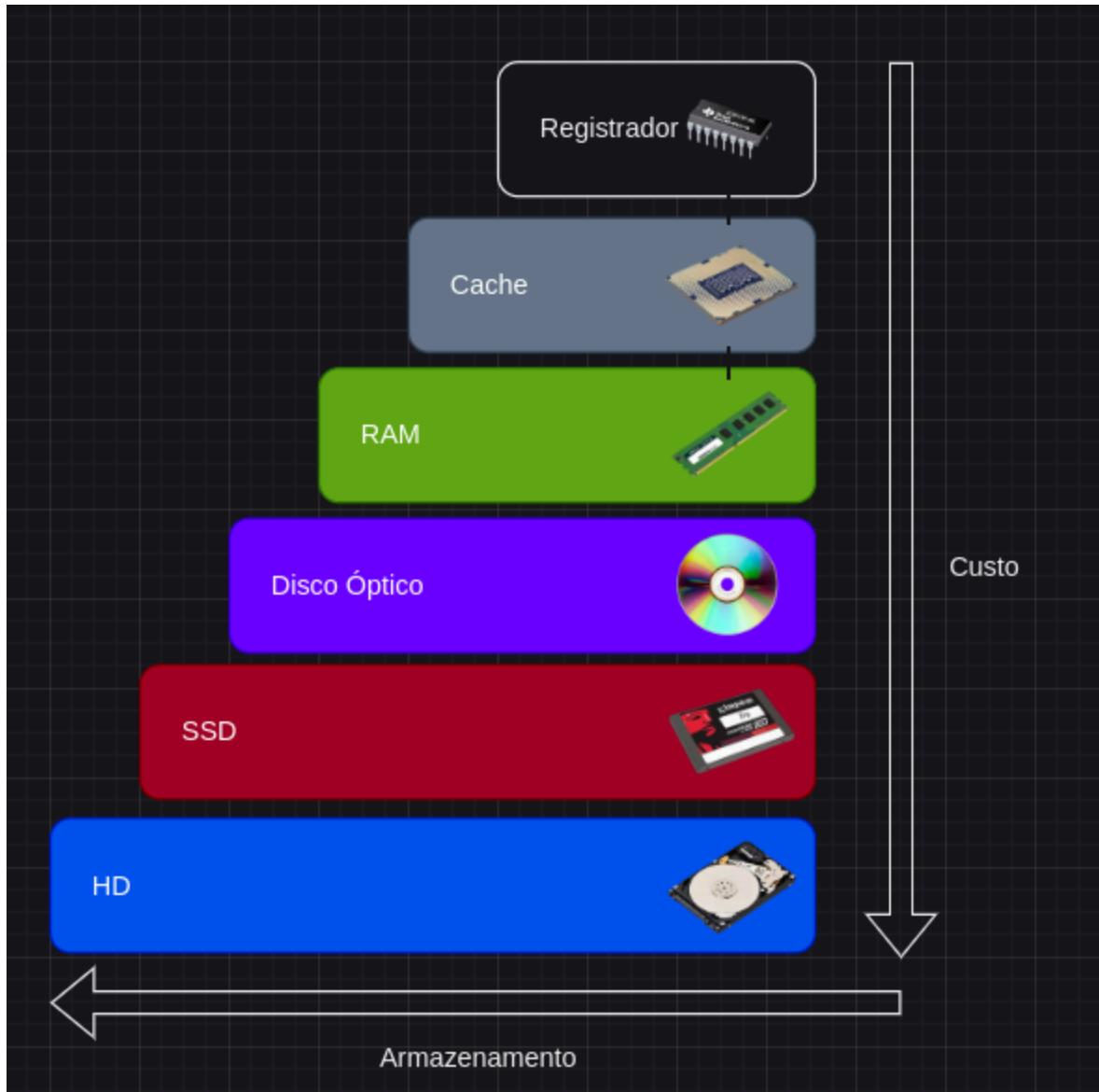
Assim, precisamos de outro tipo de memória chamado **memória secundária**, que tem o propósito de armazenar dados e programas de maneira permanente.

Um bom exemplo de memória secundária é o HD (Disco Rígido) e também temos outro tipo que está se tornando mais popular no mercado, o SSD (Disco de Estado Sólido).

No entanto, não há apenas dispositivos de armazenamento nessa hierarquia. Também podemos fazer uma hierarquia desses dispositivos, que é assim:

Diagramas de Dispositivos de Armazenamento:





003 - Estrutura de Armazenamento Hierarquia Dispositivos De Armazenamento

1.4 Estrutura de Entrada e Saída

Os dispositivos de Entrada e Saída (ou E/S), são um dos grandes pontos importantes para um Sistema Operacional, como podemos notar no armazenamento que possui grande importância para ser um dispositivo de E/S.

- Um outro ponto importante é que grande parte do código do SO é pensado para E/S;
 - Tanto por causa da **confiabilidade** como **desempenho**.

i Um sistema computadorizado para uso geral, consiste em:

- CPU
- Diversos tipos de controladores de dispositivos conectados por um barramento comum
- Cada controlador possui um tipo específico de dispositivo

Por exemplo, para o controlador SCSI (Small Computer-System Interface) podemos ter sete ou até mais dispositivos conectados ao mesmo controlador.

Cada controlador armazena **buffer local** e um **conjunto de registradores de uso especial**.

Os controladores tem duas funções básicas, que se baseiam:

- **Move** os dados para os dispositivos periféricos que controla.
- **Gerencia** o uso do buffer local.

Tais sistemas possuem um **driver de dispositivo** (driver de dispositivo) que serve como ponte entre o dispositivo e o sistema, permitindo que a **entrada dos dispositivos** tenha uma **saída uniforme** para o restante do sistema.

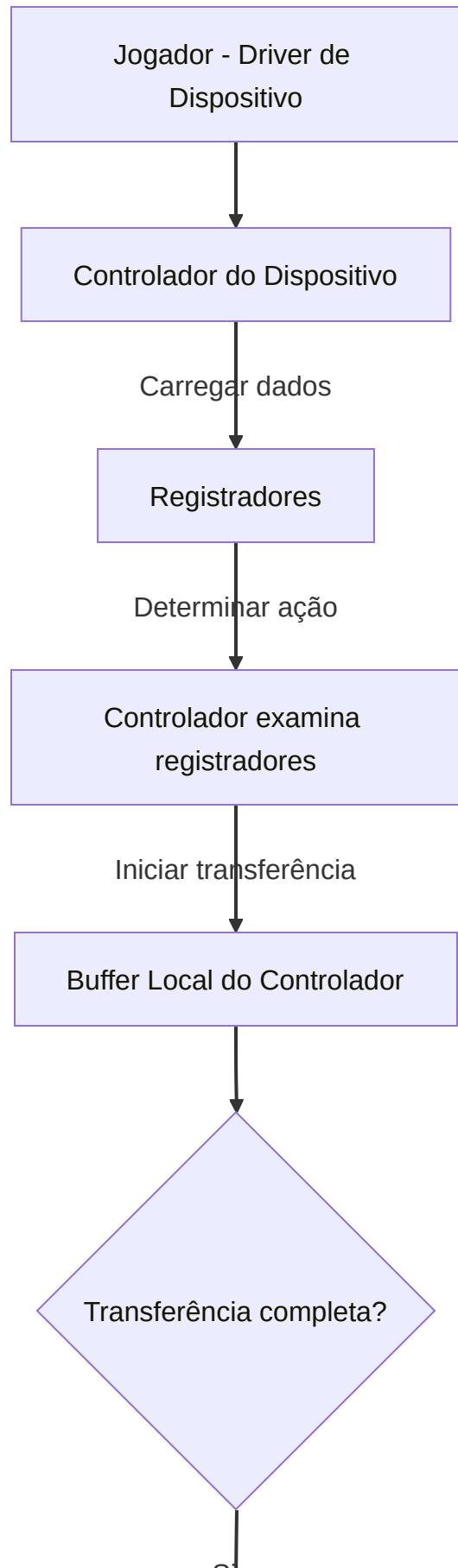
O funcionamento de uma operação de E/S:

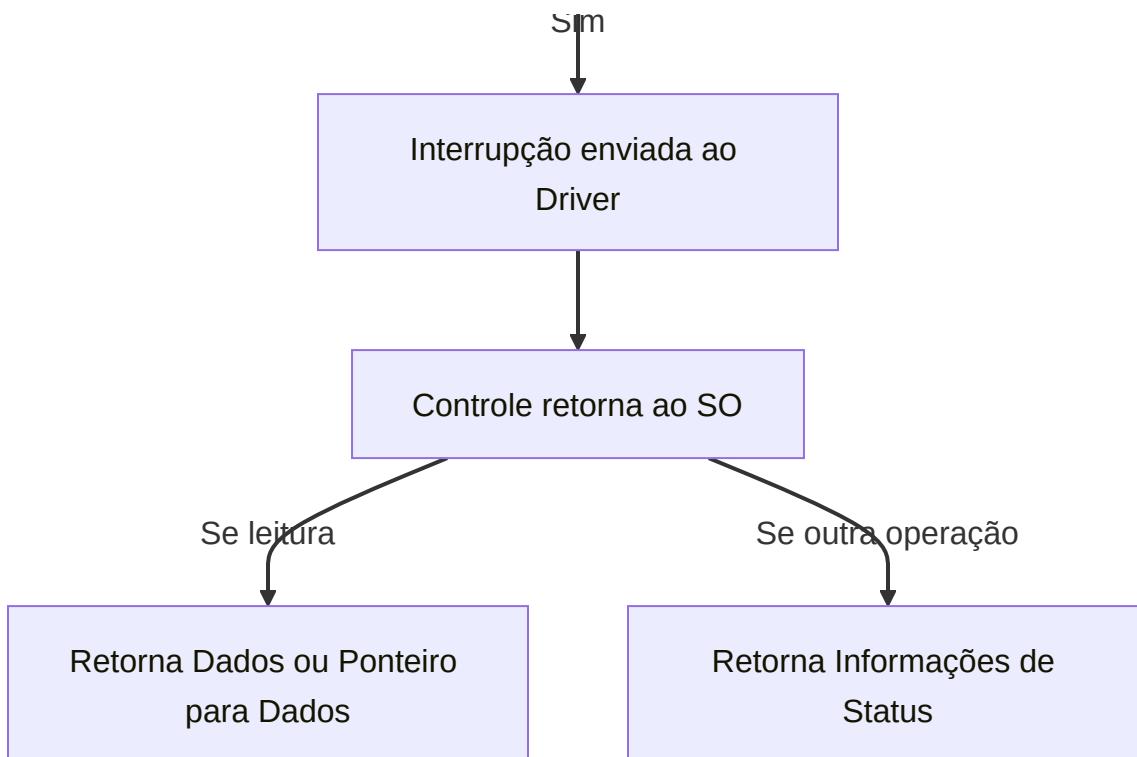
- O **driver de dispositivo** carrega os **registradores** apropriados para dentro do

controlador do dispositivo.

- O controlador examina o **conteúdo** que tem nos **registradores**, para determinar que ação deve ser tomada.
- O controlador começa a transferir os dados do dispositivo para o seu buffer local.
- Assim que a transferência está concluída, o **controlador de dispositivo** envia uma **interrupção** para o **driver de dispositivo** informando que a transferência foi concluída.
- O driver de dispositivo então retorna o controle diretamente para o SO, retornando os dados ou um ponteiro para esses dados, possivelmente, caso a operação seja de leitura.
 - Para outras operações, o driver retorna informações de status.

Representação:





- i** Para pequenas porções de dados, essa arquitetura de E/S por interrupção funciona bem, mas não funciona somente com isso há muito tempo, por isso, se usarmos essa forma para grandes volumes de dados como E/S de disco causa um **overhead** (que é uma sobrecarga).

Com esse grande problema, precisamos então de um outro dispositivo, um que armazene esses dados para que o acesso seja mais rápido, para isso usamos a **DAM** (Direct Access Memory ou Memória de Acesso Direto).

Logo o ciclo se torna assim:

- Depois de configurar buffers, ponteiros e contadores, o dispositivo de E/S, o controlador de dispositivo **move um bloco inteiro de dados** diretamente para ou do seu próprio buffer local para a memória.
- Somente **uma interrupção é feita por bloco**, para que seja avisado ao driver de dispositivo que a **transferência foi concluída**.

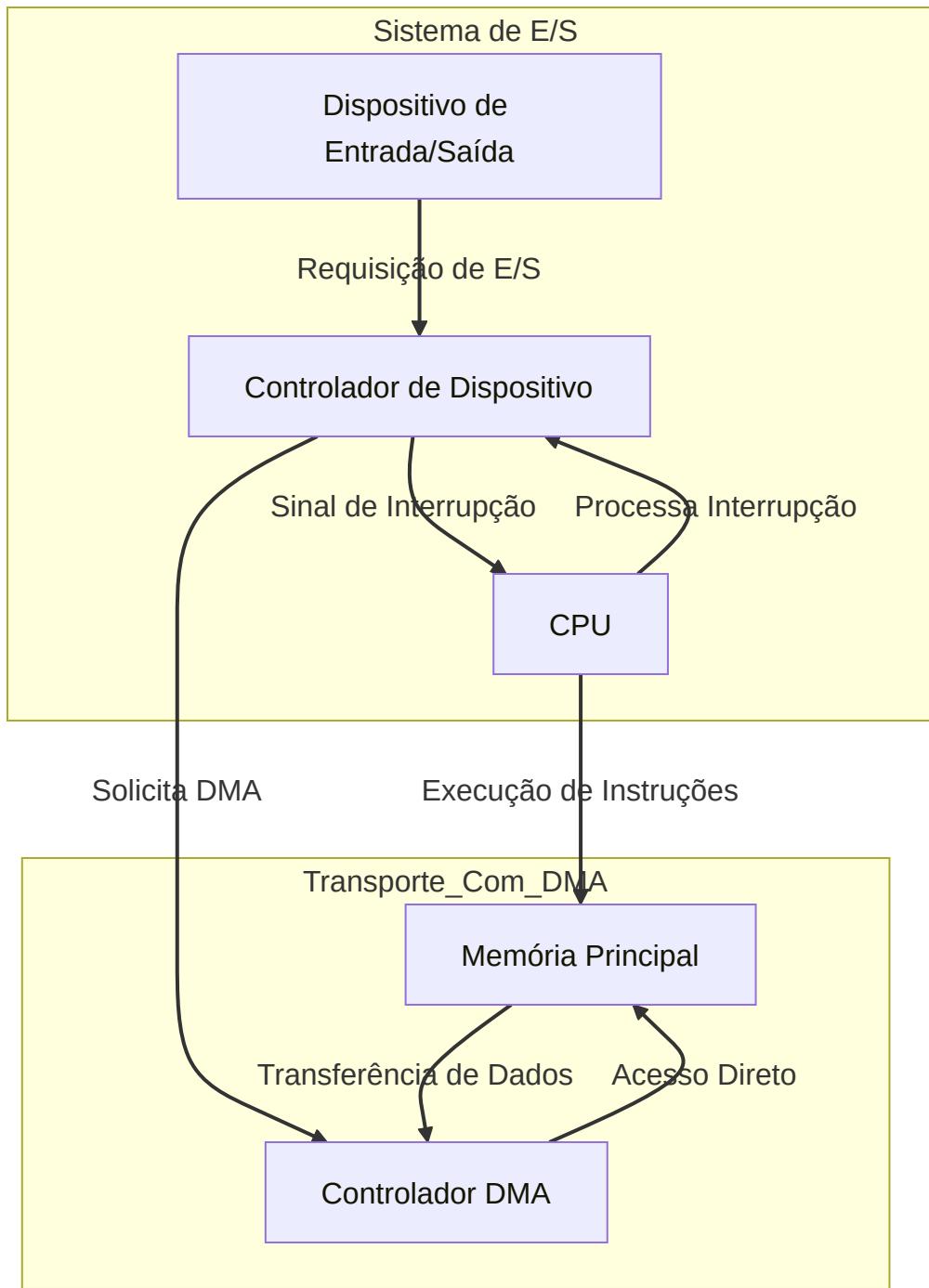
- i** Nesta etapa de transferência direta não ocorre intervenção da CPU, assim

apenas o controlador de dispositivo cuida dessa tarefa.

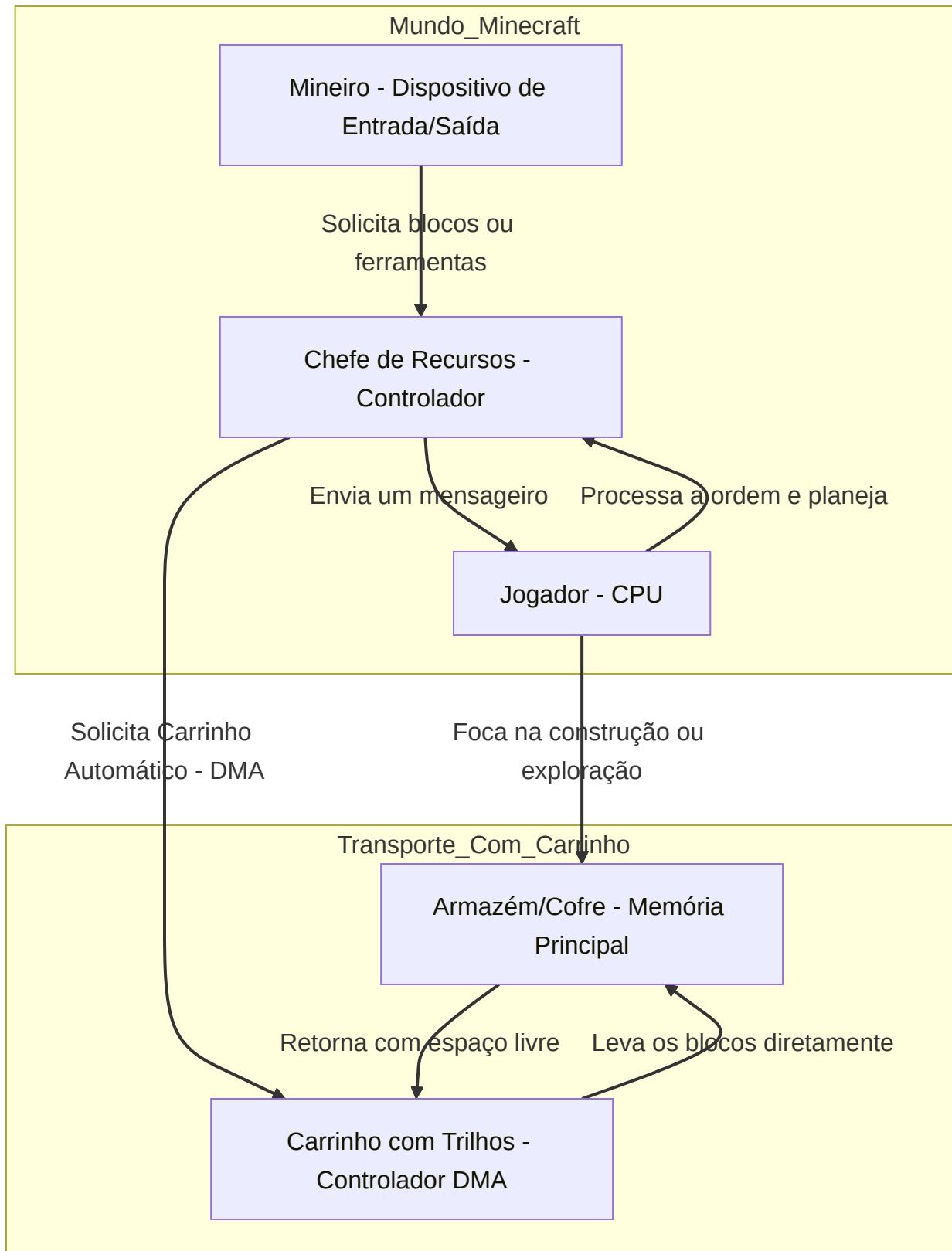
Para alguns sistemas não é utilizado essa arquitetura de barramento e sim de switch:

- Nesse tipo de sistema, os vários componentes do sistema podem interagir entre si ao mesmo tempo.
- Ao invés de competir por ciclos de um barramento compartilhado.
- Assim o **DMA** consegue ser ainda mais eficiente.

Representação da interação dos componentes num sistema:



- **Com Mineiro:**



1.5 Arquitetura do Sistema

Agora falaremos sobre a categorização dos sistemas computadorizados, que é feita com base no número de processadores que ele possui, ou seja, estamos nos referindo a computadores de uso geral.

1.5.1 Sistema Monoprocessador

Esses sistemas, como o nome diz, possuem um único processador e foram muito utilizados, desde PDAs até mainframes. Assim, esses sistemas contêm uma única CPU que pode realizar diversas instruções de uso geral, assim como os processos do usuário.

- A maioria dos sistemas utiliza um processador de uso específico, como, por exemplo, para processamento gráfico, com os controladores gráficos, ou nos mainframes, com os processadores de E/S.

Esses processadores específicos não executam processos do usuário e somente realizam instruções limitadas e especializadas.

- Em alguns casos, o sistema operacional controla esse componente, pois o sistema envia informações sobre sua próxima tarefa e monitora seu status.

Exemplo:

- Um processador controlador de disco recebe uma sequência de requisições da CPU principal.
- Implementa sua própria fila de disco e algoritmo de escalonamento.

- Com isso, há um alívio na carga de processamento do escalonamento de disco, que, de outra forma, seria delegado à CPU principal.

O sistema operacional não pode se comunicar diretamente com esses processadores, pois eles operam em um nível mais baixo. Um exemplo disso são os teclados, que

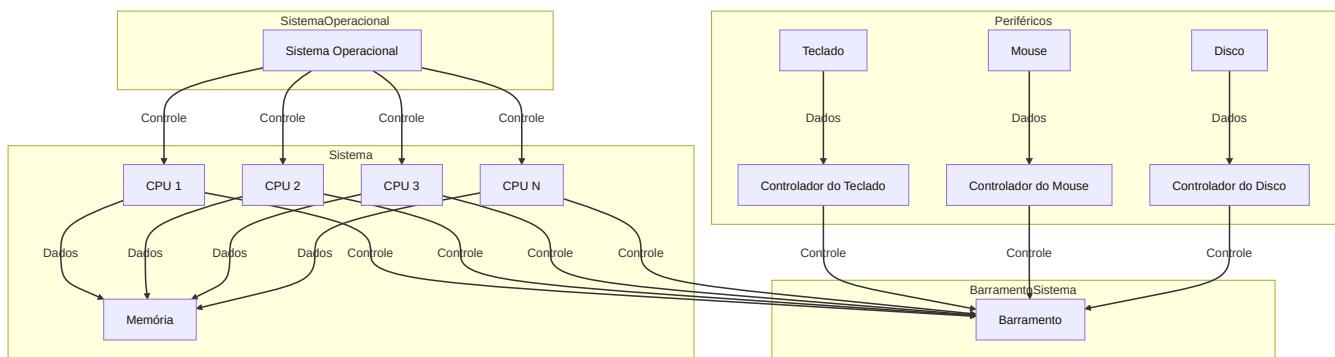
possuem um microprocessador responsável por converter os toques nas teclas em códigos que serão enviados para a CPU principal.

Assim, esses processadores realizam suas tarefas de forma anônima, pois não interagem diretamente com o sistema operacional.

Mesmo com o uso desses processadores específicos, o sistema ainda não é considerado multiprocessado.

Para que um sistema seja classificado como monoprocessador, ele deve possuir uma única CPU de uso geral. Os processadores mencionados anteriormente são de uso específico.

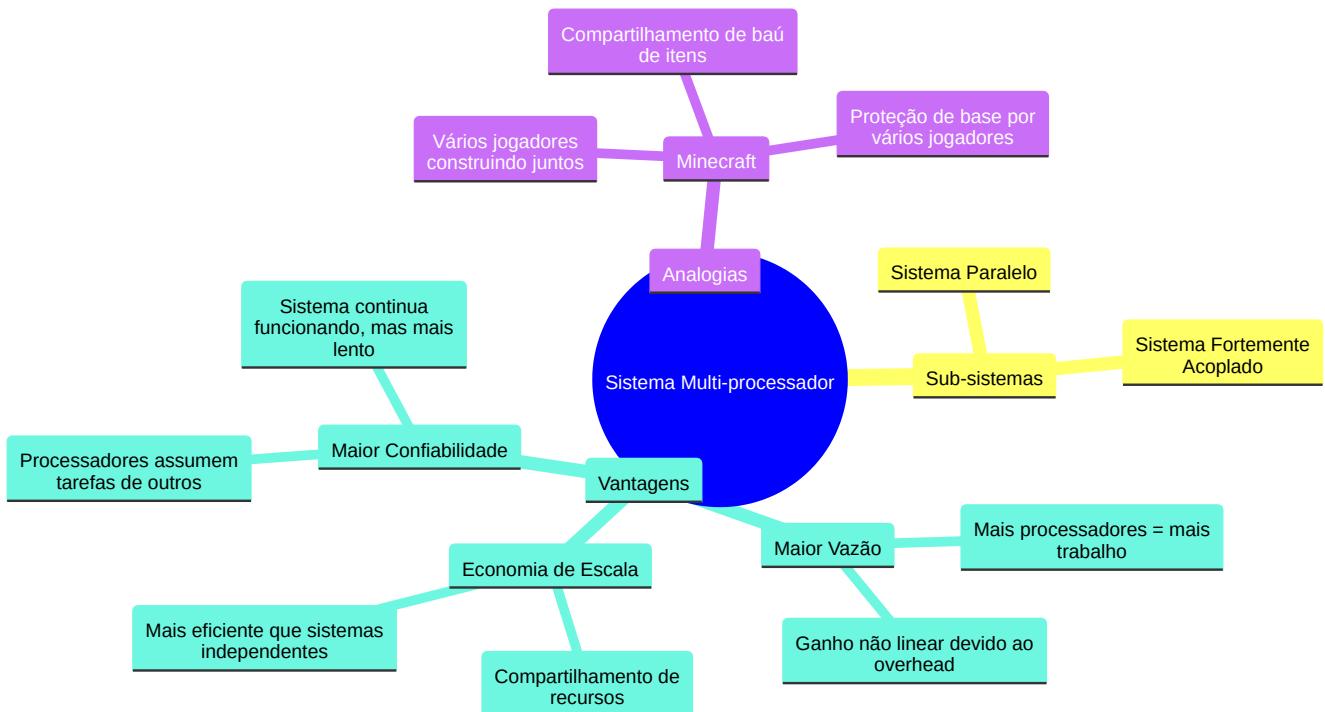
Diagrama



1.5.2 Sistema multi-processador

Esse tipo de sistema em que temos mais de um processador, de uso geral, dentro de um mesmo sistema computadorizado tem ganhado cada vez mais espaço por diversas razões o lugar dos sistema mono processador.

Os sistemas multiprocessados, ou também conhecidos como: **sistemas paralelos** (parallel system) ou **sistema fortemente acoplado** (tightly coupled system) fazem um compartilhamento perfeito de periféricos, relógio do computador, barramento do computador para vários processadores de modo que a comunicação entre eles é perfeita.



Podemos escalar **três grandes vantagens** acerca desse tipo de arquitetura para sistemas:

1. Maior vazão:

- Como ter vários jogadores trabalhando juntos em uma construção no Minecraft.
- Mais processadores = mais trabalho realizado em menos tempo.
- Porém, o ganho não é linear devido ao overhead de coordenação.

2. Economia de escala:

- Semelhante a compartilhar um baú de itens entre vários jogadores no Minecraft.
- Sistemas multiprocessados compartilham recursos (periféricos, armazenamento, energia).
- Mais eficiente que ter vários sistemas independentes.

3. Maior confiabilidade:

- Como ter vários jogadores protegendo uma base no Minecraft.
- Se um processador falha, os outros podem assumir suas tarefas.

- O sistema continua funcionando, apenas mais lento, em vez de travar completamente.

Estas vantagens tornam os sistemas multiprocessados cada vez mais populares, assim como servidores de Minecraft com vários jogadores oferecem uma experiência mais robusta e dinâmica.

Imagine construir um mundo no Minecraft. A **confiabilidade** do sistema é como a estabilidade do mundo: se algo der errado (bloco sumir, mob bugar), ele continua funcionando, mesmo que limitado. Isso é **degradação controlada** — como minerar com uma ferramenta pior se a melhor quebrar.

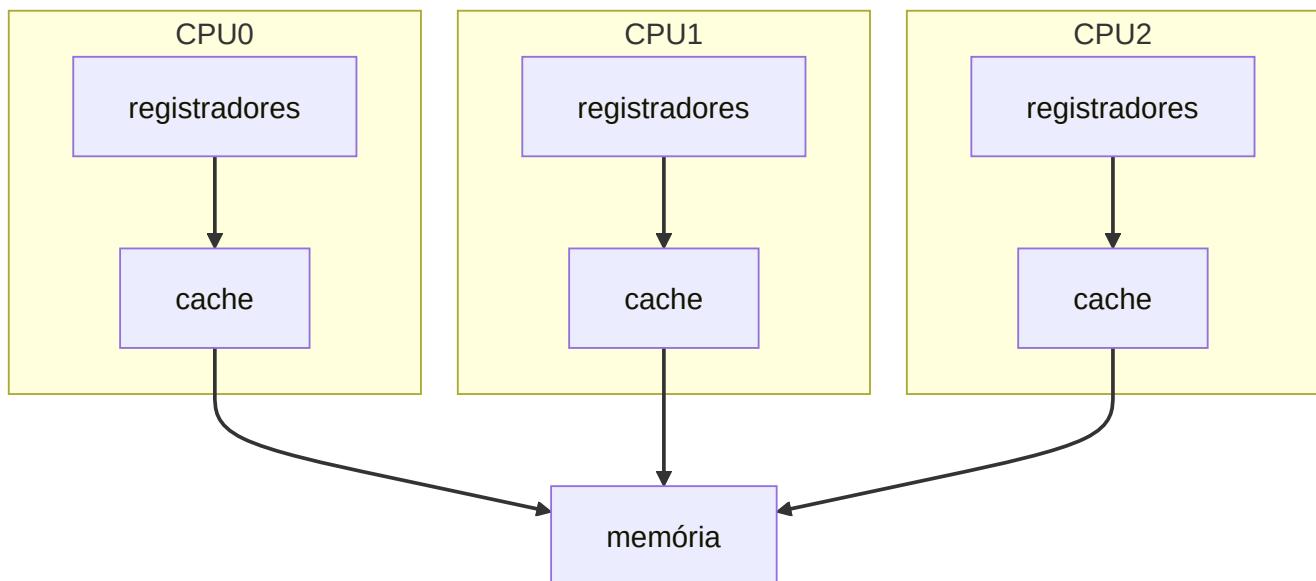
Sistemas **tolerantes a falhas** vão além: mesmo com falhas, funcionam sem interrupções. No Minecraft, seria um backup automático que restaura blocos destruídos por creepers sem você sair do jogo.

O **HP NonStop** é como um servidor com duplicação: dois jogadores (CPUs) constroem a mesma coisa ao mesmo tempo. Se um errar, o sistema corrige e transfere a tarefa para outro par, garantindo continuidade, mas com custo maior.

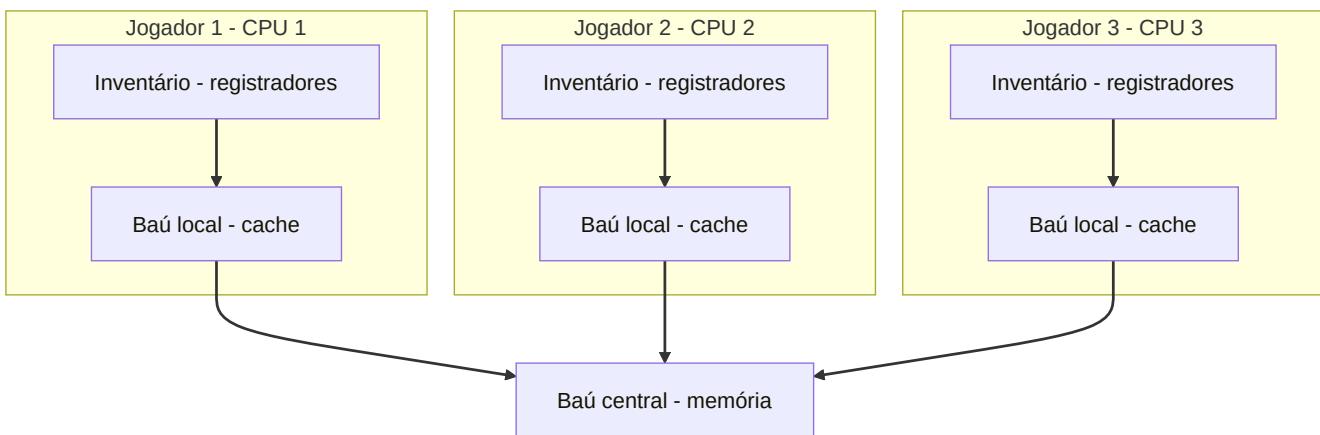
Já os **sistemas multiprocessados** são como vários jogadores trabalhando juntos:

1. Assimétrico: Um jogador mestre comanda os outros. Se ele sair, tudo pode parar.

2. Simétrico (SMP): Todos são iguais, compartilham recursos (baú/memória) e trabalham juntos sem perder desempenho. Sistemas como **Solaris**, **Windows** e **Linux** usam isso.



- Com Minecraft:



1.5.3 Sistemas em Clusters

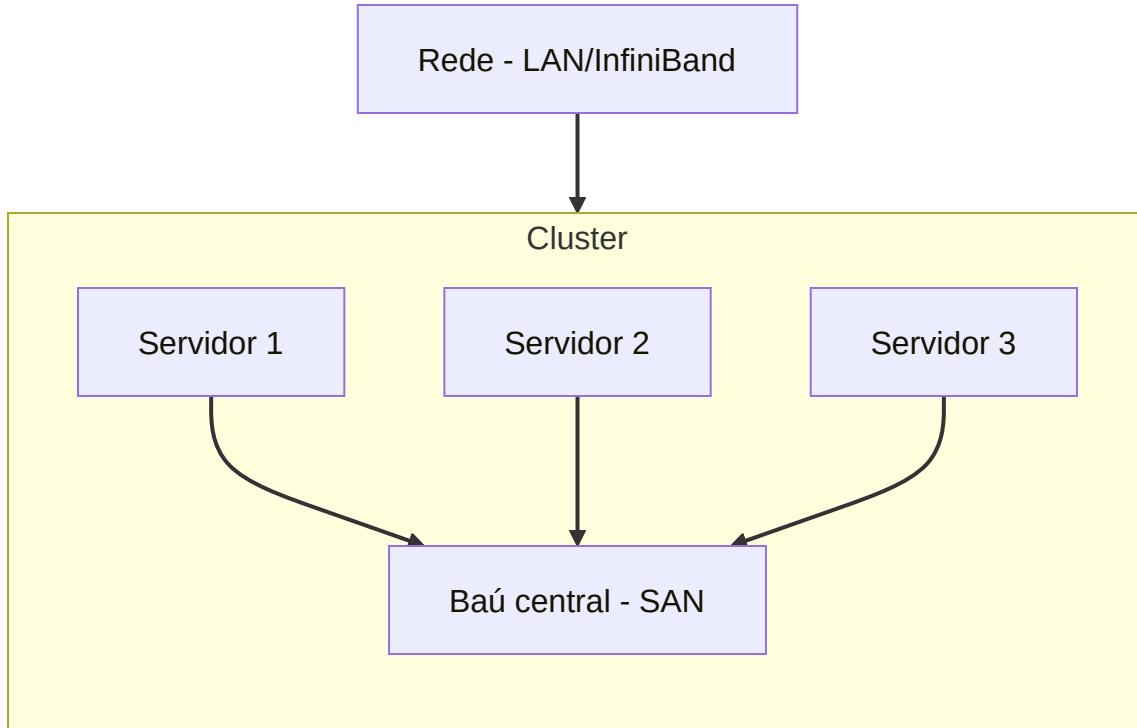
Resumo com analogias ao Minecraft:

Um **sistema em cluster** é como um grupo de servidores de Minecraft trabalhando juntos. Cada servidor (nó) é independente, mas eles estão conectados por uma rede (LAN ou conexão rápida) e compartilham armazenamento (como um baú central). O objetivo é garantir **alta disponibilidade e alto desempenho**.

- **Alta disponibilidade:** Se um servidor falhar (explodir como um creeper), outro assume seu lugar, mantendo o mundo (serviço) funcionando com pouca interrupção.
- **Modo assimétrico:** Um servidor fica de olho (hot-standby) enquanto o outro roda o jogo. Se o ativo falhar, o standby assume.
- **Modo simétrico:** Vários servidores rodam o jogo e se monitoram, usando todo o hardware de forma eficiente.
- **Alto desempenho:** Vários servidores podem trabalhar juntos para resolver tarefas complexas, como gerar chunks ou processar comandos em paralelo. Isso exige que o jogo (aplicação) seja dividido em partes que rodam simultaneamente em diferentes servidores.
- **Clusters paralelos:** Vários servidores acessam os mesmos dados (como um banco de dados compartilhado). Para evitar conflitos, um sistema de "trava" (DLM) garante que apenas um servidor modifique os dados por vez.

- **SANs (Storage-Area Networks):** É como um baú gigante conectado a todos os servidores. Se um servidor cair, outro pode pegar os itens (dados) e continuar o jogo.

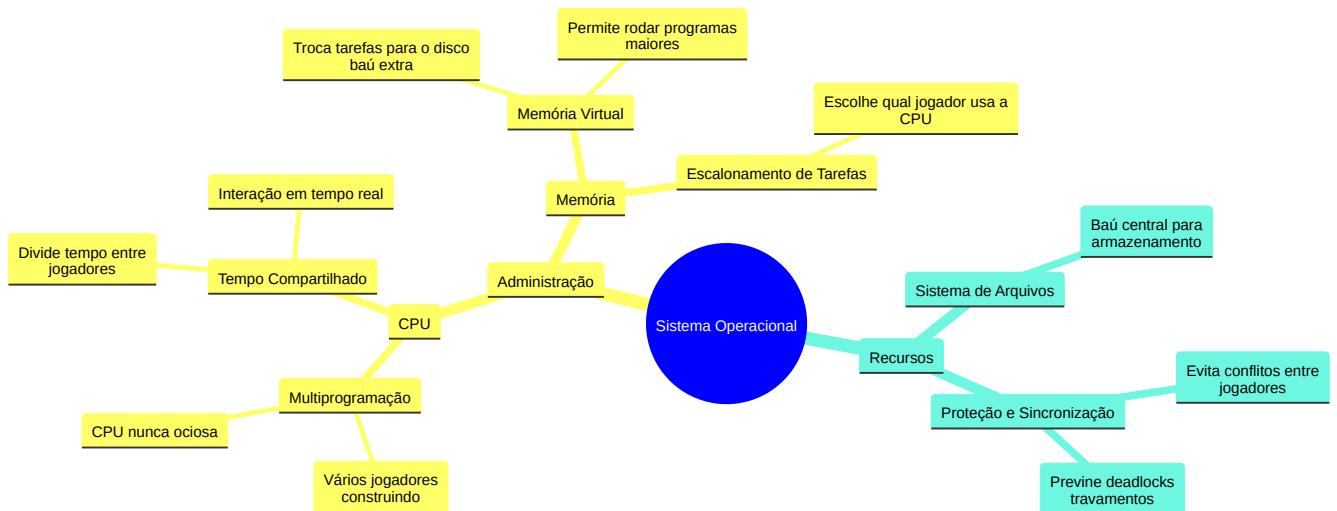
Resumo visual:



1.6 Estrutura do sistema operacional

Um **sistema operacional** é como o "administrador" de um servidor de Minecraft. Ele gerencia recursos (CPU, memória, dispositivos) e permite que vários programas (ou jogadores) funcionem ao mesmo tempo.

- **Multiprogramação:** É como ter vários jogadores construindo no mesmo mundo. Se um jogador precisa esperar (por exemplo, para minerar), o sistema passa para outro, mantendo a CPU sempre ocupada. Isso evita que o servidor fique ocioso.
- **Tempo compartilhado (time sharing):** É como dividir o tempo do servidor entre vários jogadores. Cada um recebe um pouco de atenção do servidor, mas tão rápido que parece que todos estão jogando ao mesmo tempo. Isso permite interação em tempo real, como digitar comandos e ver resultados imediatos.
- **Escalonamento de tarefas:** O sistema escolhe qual jogador (tarefa) deve usar o servidor (CPU) a seguir, garantindo que todos tenham uma chance justa.
- **Memória virtual:** Se o servidor não tem espaço para todos os jogadores (tarefas) na memória, ele "troca" alguns para o disco (como um baú extra) e os traz de volta quando necessário. Isso permite rodar programas maiores do que a memória física.
- **Sistema de arquivos:** É como o baú central do servidor, onde todos os itens (arquivos) são armazenados e organizados.
- **Proteção e sincronização:** O sistema garante que os jogadores (tarefas) não interfiram uns com os outros, evitando conflitos e travamentos (deadlocks).



1.7 Operações do Sistema Operacional

Resumo com analogias ao Minecraft:

O **sistema operacional** é como o "administrador" de um servidor de Minecraft, controlando tudo que acontece no mundo (sistema). Ele usa **interrupções** e **traps** para lidar com eventos, como um jogador tentando fazer algo que não deveria (erro) ou pedindo ajuda (chamada de sistema).

1. Modo Dual (Usuário e Kernel):

- **Modo Usuário:** Onde os jogadores (programas de usuário) operam. Eles têm permissão limitada, como construir ou minerar, mas não podem alterar o servidor diretamente.
- **Modo Kernel:** Onde o administrador (sistema operacional) opera. Ele tem controle total sobre o servidor, como gerenciar recursos, corrigir erros ou expulsar jogadores problemáticos.
- **Transição:** Quando um jogador precisa de algo que só o administrador pode fazer (como abrir um portal), ele faz uma **chamada de sistema**, e o servidor muda para o modo kernel temporariamente.

2. Proteção:

- O sistema operacional protege o servidor de jogadores mal-intencionados ou erros. Por exemplo, se um jogador tentar destruir o servidor (executar uma instrução privilegiada no modo usuário), o sistema bloqueia a ação e notifica o administrador.

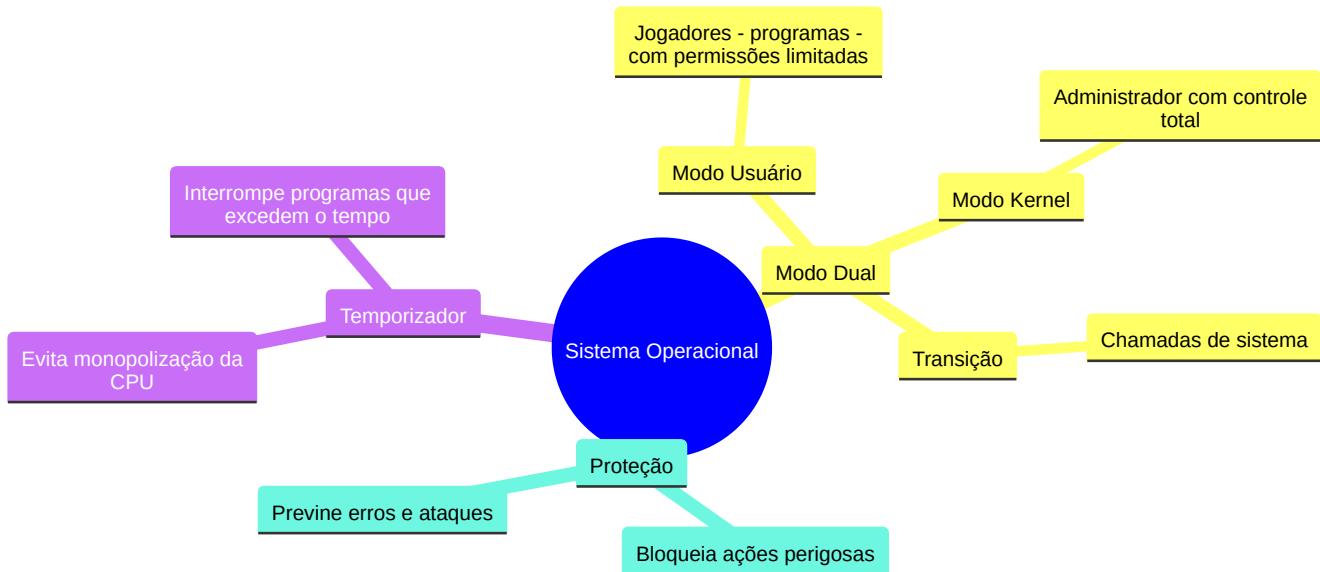
3. Temporizador:

- Para evitar que um jogador monopolize o servidor (loop infinito), o sistema usa um **temporizador**. Se um jogador ficar muito tempo sem ceder a vez, o sistema interrompe e passa o controle para outro jogador ou para o administrador.

4. Ciclo de Execução:

- O sistema operacional começa no modo kernel (administrador) ao ligar o servidor. Ele carrega os jogadores (programas) no modo usuário e alterna entre os modos conforme necessário, garantindo que tudo funcione sem problemas.

Resumo visual:



Em resumo, o sistema operacional é como um administrador de servidor de Minecraft, alternando entre modos para garantir que os jogadores (programas) possam jogar sem causar problemas, enquanto mantém o controle total sobre o sistema.

Caching

O entendimento de **caching** é fundamental para compreender como os sistemas computadorizados funcionam.

Primeiro, pensemos que as **informações são armazenadas** em algum **dispositivo de armazenamento**, como a memória principal. Conforme são utilizadas, essas informações **são copiadas para uma memória mais rápida de modo temporário** (até mesmo mais rápida que a memória principal). Essa memória é o **cache**.

Como funciona?

Primeiro, ao precisarmos de alguma informação, **o sistema busca no cache**:

- Se a informação estiver disponível, **usamos os dados** daí mesmo.
- Caso não esteja, **o sistema irá carregá-la** de uma memória mais lenta (principal ou até mesmo de massa, secundária) e, em seguida, **fará a cópia temporária para o cache**.
 - Dessa forma, em uma **nova tentativa de acesso ao dado**, ele **poderá ser encontrado no cache**, reduzindo significativamente as consultas lentas que seriam feitas à memória principal.

Indo além, registradores responsáveis pela comunicação com a memória principal, como registradores de índice, **são gerenciados por um algoritmo de alocação e substituição de registradores, implementado pelo programador ou compilador**.

Esses **algoritmos definem quais informações serão mantidas nos registradores e quais serão enviadas para a memória principal**.

Também existem hardwares projetados para **serem implementados inteiramente no hardware**.

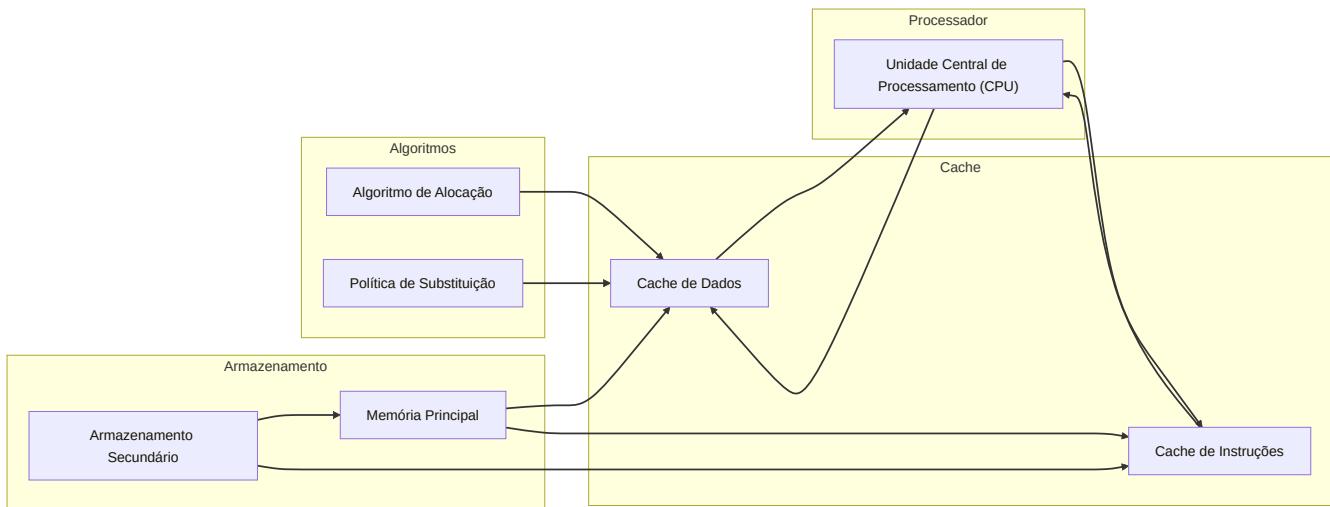
A maioria dos sistemas possui um **cache de instruções**, que serve para **armazenar as próximas instruções a serem executadas**.

Sem isso, a CPU levaria vários ciclos para buscar na memória principal a **próxima instrução a ser executada**.

Por essa e outras razões, a maioria dos sistemas possui vários caches de dados de alta velocidade, que estão no **topo da Hierarquia de Memórias**.

Mas, como os caches possuem um **tamanho reduzido**, o **gerenciamento de cache** se torna **fundamental**. Esse gerenciamento envolve alguns aspectos importantes, como:

- Definir o **tamanho do cache**.
- Estabelecer a **política de substituição**.



Esses fatores podem **melhorar o desempenho da memória cache**.

A **memória principal** pode ser vista como um **cache rápido para o armazenamento secundário**, pois os dados precisam ser copiados da memória secundária para a principal antes de serem utilizados.

De forma recíproca, para serem **movidos para a memória secundária**, os dados precisam estar **primeiro na memória principal**, garantindo proteção e integridade.

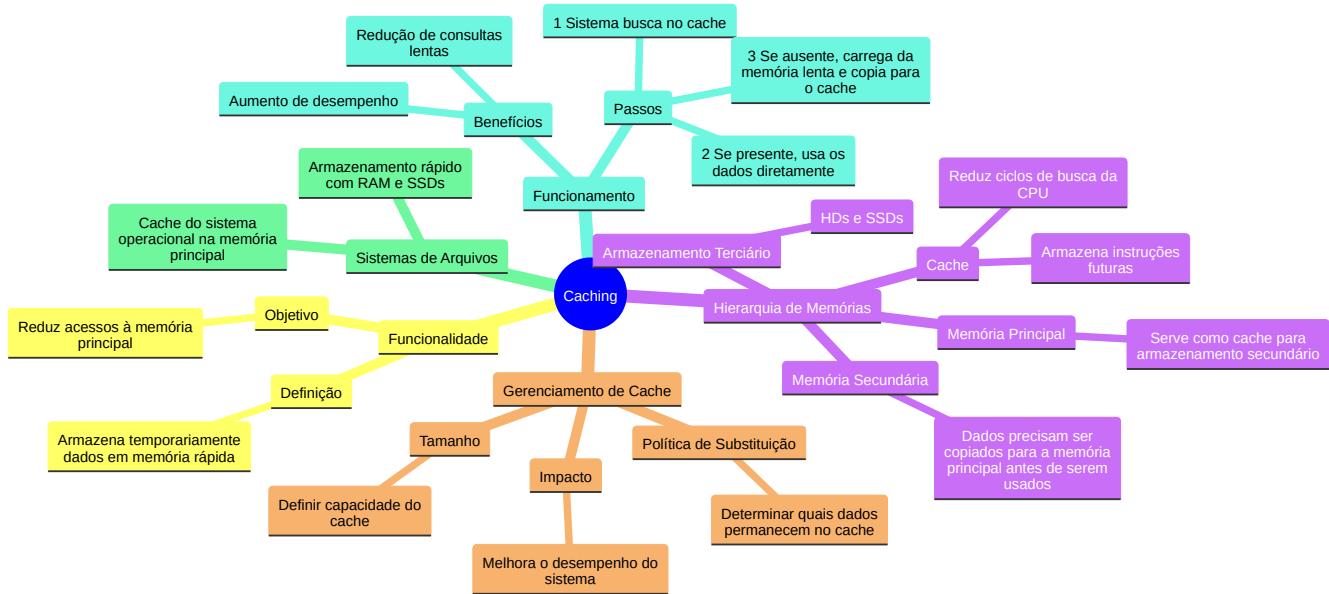
O sistema de arquivos vê os dados permanentemente gravados no armazenamento secundário de forma hierárquica, existindo *diversos níveis na hierarquia*:

- No nível mais alto -> o **sistema operacional** pode manter um **cache do sistema de arquivos na memória principal**.

Também é possível que memórias RAM, como discos de estado sólido (ou então discos eletrônicos de RAM), sejam usadas para **armazenamento de alta velocidade**, acessados pela **interface do sistema de arquivos**. Isso significa que a comunicação deve ser feita diretamente com o sistema de arquivos.

Atualmente, a maior parte do **armazenamento terciário** consiste em **HDs ou SSDs**.

Diagrama



Níveis e o Cache

Os **movimentos** de informações entre os **níveis da hierarquia de memórias** podem ser de dois tipos: **explícitos** e **implícitos**. Isso depende da arquitetura do **hardware** e do **software** que controla o sistema operacional.

Podemos exemplificar essa questão:

- A **transferência de dados entre a cache e a CPU e seus registradores** -> ocorre diretamente no **hardware**, sem intervenção do sistema operacional.
- A **transferência de dados do disco para a memória RAM**-> normalmente é controlada pelo **sistema operacional**.

Como, nessa estrutura hierárquica, os mesmos dados podem aparecer em diferentes níveis de armazenamento, vejamos um exemplo:

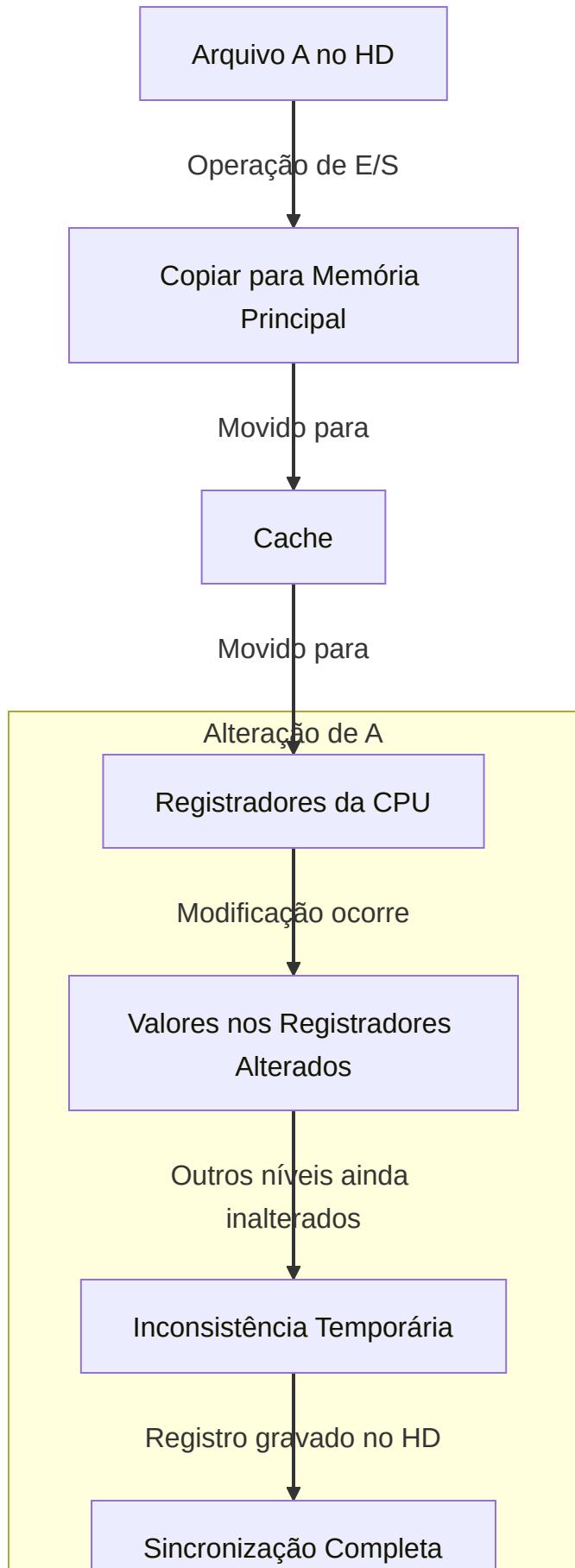
- Suponha que um texto no arquivo A precise ser alterado para um outro valor no arquivo B, que reside no HD.
- Antes da alteração, o sistema precisa emitir uma operação de E/S para copiar o bloco de disco contendo A para a memória principal.

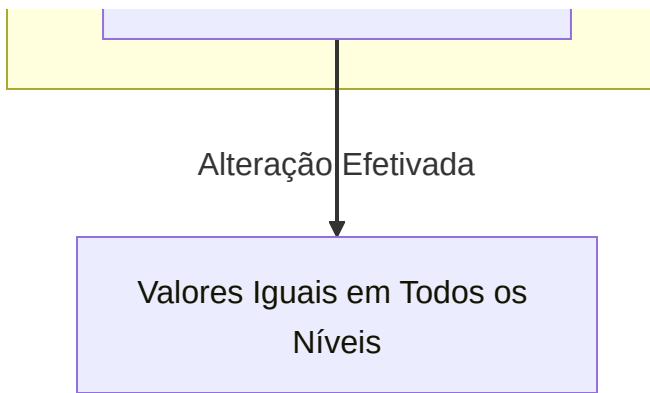
- Em seguida, o arquivo A será copiado para o cache e para os registradores internos da CPU.
- Assim, a cópia de A estará presente em vários níveis, conforme mostrado abaixo:



- Quando a alteração for feita nos registradores internos da CPU, os valores de A serão diferentes nos outros níveis de armazenamento, que permanecerão inalterados.
- Somente quando o registrador gravar a mudança no disco rígido (memória secundária), os valores nos diferentes níveis estarão sincronizados, tornando a alteração efetiva.

Diagrama





Threads, Cores e Sistemas Distribuídos

Em um ambiente com **apenas uma thread** (executando uma única tarefa por vez), esse esquema hierárquico funciona perfeitamente. Quando um valor é alterado nos registradores ou acessado no disco, o fluxo ocorre de forma linear, do nível mais alto ao mais baixo, sem conflitos de dados, pois todos os níveis são atualizados sequencialmente.

Entretanto, em **sistemas multicore**, onde múltiplas threads acessam simultaneamente o mesmo arquivo A, é necessário um **mecanismo de controle** para garantir que todos os núcleos acessem os valores atualizados de A. Caso contrário, podem ocorrer inconsistências, onde diferentes threads terão versões diferentes do mesmo dado, causando falhas no sistema.

Esse problema se torna ainda mais crítico em **sistemas multiprocessadores**, pois, além de registradores internos, cada processador pode possuir caches locais distintos.

Assim, A pode existir em diversos caches simultaneamente, e é essencial que a **versão mais recente de A seja refletida em todos os núcleos**.

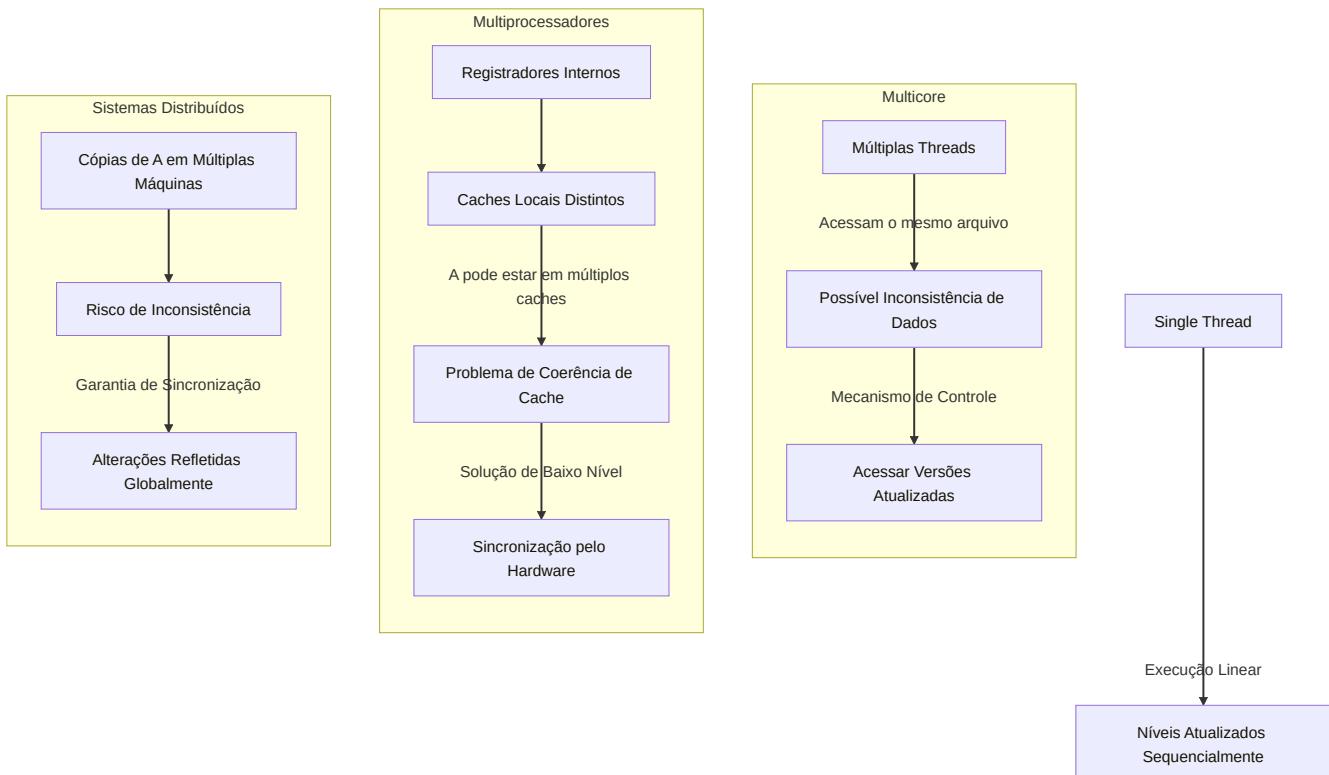
- Esse problema é conhecido como **Coerência de Cache**, e sua solução ocorre em **baixo nível**, diretamente no hardware.

A complexidade aumenta ainda mais em **sistemas distribuídos**, onde diversas cópias de A podem estar espalhadas por vários computadores conectados em rede.

Para evitar inconsistências, o sistema precisa garantir que **as alterações no arquivo sejam refletidas nas demais máquinas o mais rapidamente possível**.

- Existem diversas estratégias para resolver esse problema de sincronização em sistemas distribuídos.

Diagrama



Questões 1

Esta seção contém perguntas relacionadas ao assunto em questão, que podem ser usadas como referência para aprender ou revisar conhecimentos.

- i** Sugerimos que resolva estas questões para auxiliar em um melhor entendimento do conteúdo e assim melhor resolução dos quizzes

Pergunta 1

Dispositivos que utilizam Bluetooth e redes Wi-Fi se comunicam sem fio dentro de uma determinada área, formando uma:

- a) Rede de área metropolitana (MAN)
- b) Rede de área local (LAN)
- c) Rede de pequena área (SAN)
- d) Rede de longa distância (WAN)

Pergunta 2

O que fornece serviços adicionais para desenvolvedores ao intermediar a comunicação entre aplicativos e o sistema operacional?

- a) Virtualização
- b) Middleware
- c) Computação em nuvem
- d) Software de sistema

Pergunta 3

Os sistemas operacionais geralmente operam em dois modos distintos. Quais são eles?

- a) Modo físico e modo lógico
- b) Modo usuário e modo kernel
- c) Modo supervisor e modo secundário

- d) Modo protegido e modo comum

Pergunta 4

No contexto do Linux, uma versão personalizada do sistema operacional é conhecida como:

- a) Instalação (Installation)
- b) Distribuição (Distribution)
- c) LiveCD
- d) Virtual Machine

Pergunta 5

Qual das seguintes afirmações sobre dispositivos móveis é falsa?

- a) Eles geralmente possuem menos núcleos de processamento do que desktops.
- b) O consumo de energia é um fator crítico para dispositivos móveis.
- c) Dispositivos móveis sempre possuem mais capacidade de armazenamento que laptops.
- d) Algumas funcionalidades dos dispositivos móveis não estão presentes em desktops.

Pergunta 6

Sistemas embarcados geralmente executam um sistema operacional de:

- a) Tempo real
- b) Rede
- c) Clusterizado
- d) Multiprogramação

Pergunta 7

Quais são dois fatores importantes no design da memória cache?

- a) Consumo de energia e reutilização

- b) Política de tamanho e substituição
- c) Privilégios de acesso e controle
- d) Velocidade e volatilidade

Pergunta 8

De que forma um sistema operacional pode ser comparado a um governo?

- a) Ele executa todas as funções sozinho.
- b) Ele cria um ambiente para que outros programas possam operar.
- c) Ele prioriza as necessidades individuais dos usuários.
- d) Ele raramente funciona corretamente.

Pergunta 9

Sobre instruções privilegiadas, qual das afirmações a seguir é incorreta?

- a) Elas não podem ser executadas no modo usuário.
- b) Apenas podem ser executadas no modo kernel.
- c) São usadas para gerenciamento de interrupções.
- d) Nunca representam riscos ao sistema.

Pergunta 10

A menor unidade de execução dentro de um sistema operacional é chamada de:

- a) Sistema operacional
- b) Timer
- c) Bit de modo
- d) Processo

Pergunta 11

Qual das opções abaixo é um exemplo de um programa de sistema?

- a) Navegador Web

- b) Interpretador de comandos
- c) Planilha eletrônica
- d) Software de edição de imagem

Pergunta 12

Qual chamada de sistema do Windows é equivalente à `close()` do UNIX?

- a) `CloseHandle()`
- b) `close()`
- c) `Exit()`
- d) `CloseFile()`

Pergunta 13

As chamadas de sistema são responsáveis por:

- a) Fornecer uma interface para os serviços do sistema operacional
- b) Gerenciar apenas memória virtual
- c) Proteger exclusivamente processos críticos
- d) Impedir a execução de programas de terceiros

Pergunta 14

O que define o que será feito dentro de um sistema operacional?

- a) Política
- b) Mecanismo
- c) Estratégia
- d) Interface

Pergunta 15

No Windows, a chamada de sistema `CreateFile()` é utilizada para criar arquivos. Qual a chamada equivalente no UNIX?

- a) fork()
- b) open()
- c) createfile()
- d) ioctl()

Pergunta 16

Qual das opções **não** é uma categoria principal de chamadas de sistema?

- a) Segurança
- b) Proteção
- c) Controle de processos
- d) Comunicação

Pergunta 17

Microkernels utilizam ____ para comunicação interna.

- a) Chamadas de sistema
- b) Memória compartilhada
- c) Virtualização
- d) Passagem de mensagens

Pergunta 18

Um bloco de inicialização (bootstrap loader):

- a) É composto por vários blocos de disco
- b) Pode conter múltiplos cilindros de disco
- c) Normalmente é suficiente para carregar e iniciar o sistema operacional
- d) Apenas aponta para a localização do restante do sistema de boot

Pergunta 19

Qual é o sistema operacional utilizado em dispositivos iPhone e iPad?

- a) iOS
- b) UNIX
- c) Android
- d) Mac OS X

Pergunta 20

Para um programa SYSGEN de um sistema operacional, qual das informações abaixo é menos útil?

- a) Quantidade de memória disponível
- b) Configurações como tamanho de buffer e algoritmo de escalonamento de CPU
- c) Lista de aplicativos a serem instalados
- d) Arquitetura da CPU

Pergunta 21

Um sistema operacional pode ser classificado como um:

- a) Gerenciador de recursos
- b) Programa de usuário
- c) Firmware de inicialização
- d) Simulador de processos

Pergunta 22

O que é multitarefa em um sistema operacional?

- a) A execução de um único processo por vez
- b) A capacidade de executar múltiplos processos simultaneamente
- c) A execução de tarefas em tempo real sem atraso
- d) A execução de comandos administrativos pelo usuário

Pergunta 23

Qual das seguintes opções descreve um **hipervisor**?

- a) Um software responsável pela virtualização de hardware
- b) Um protocolo de comunicação em redes
- c) Um sistema operacional para servidores
- d) Um método de gerenciamento de arquivos

Pergunta 24

O que acontece quando um processo em execução tenta acessar uma página de memória que não está carregada?

- a) Ele é imediatamente encerrado pelo sistema operacional
- b) O sistema operacional gera uma interrupção e carrega a página necessária
- c) O sistema operacional ignora a solicitação
- d) O processo entra em um estado de loop infinito

Pergunta 25

O que é um deadlock em um sistema operacional?

- a) Um erro crítico no kernel
- b) Um estado onde dois ou mais processos ficam bloqueados indefinidamente
- c) Um método de gerenciamento de arquivos
- d) Uma forma de escalonamento de processos

Prática 1

Nesta prática, vamos criar e configurar máquinas virtuais.

Conhecendo ferramentas

Para fazer a virtualização de sistemas operacionais, temos alguns programas disponíveis:

- **Windows:**

- VirtualBox
- VMWare
- Parallels Desktop

- **Linux:**

- VirtualBox
- GNOME Boxes
- Virtual Machine Manager

i Há casos em que o VirtualBox falha por razões desconhecidas, então é melhor ter mais de uma opção disponível.

Vamos começar

Requisitos dos sistemas

Confira os requisitos dos sistemas operacionais que serão usados:

- Windows 10 (<https://support.microsoft.com/pt-br/windows/requisitos-do-sistema-do-windows-10-6d4e9a79-66bf-7950-467c-795cf0386715>)
- Ubuntu Desktop (<https://ubuntu.com/server/docs/system-requirements>)

- É importante verificar os requisitos mínimos para garantir que sua máquina virtual tenha recursos suficientes para executar o sistema operacional escolhido.

Instalando a ferramenta

Vamos usar o VirtualBox.

- Para Windows (<https://download.virtualbox.org/virtualbox/7.1.6/VirtualBox-7.1.6-167084-Win.exe>)
- Para Linux (https://www.virtualbox.org/wiki/Linux_Downloads)

- O VirtualBox é uma ferramenta de virtualização gratuita e de código aberto, adequada para usuários iniciantes e avançados.

Instalar ISOs (Windows e Ubuntu)

- Windows (<https://www.microsoft.com/pt-br/software-download/windows10ISO>)

- Na ISO oficial do Windows, o link acima, vêm todas as versões.

- Ubuntu Desktop (<https://ubuntu.com/download/desktop>)

- Certifique-se de baixar a versão mais recente e estável do Ubuntu Desktop para garantir compatibilidade e suporte.

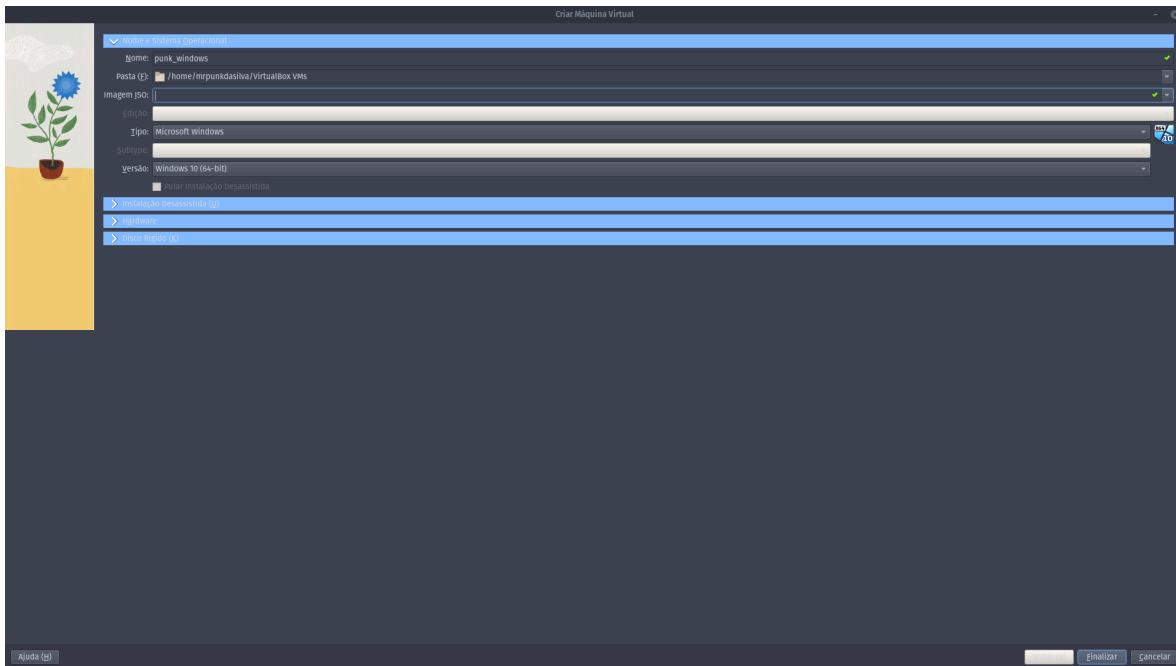
Criando Máquinas Virtuais

Windows

1. Com o VirtualBox aberto, pressione: **CTRL** + **N**

i Você pode acessar a opção também por: Machine > Add

2. Definir nome, sistema e versão:

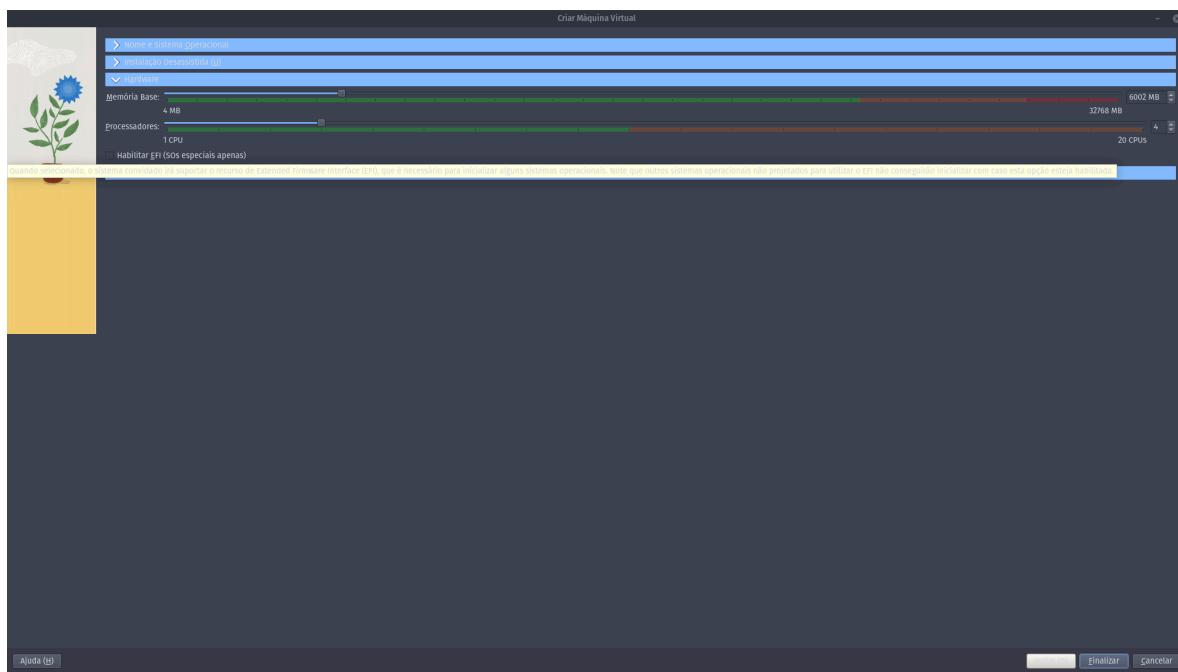


Tela de criação de máquina virtual no VirtualBox

i Escolha um nome descritivo para sua máquina virtual e selecione a versão correta do Windows que você planeja instalar.

3. Agora vamos definir a memória RAM. É bom deixarmos no mínimo 4GB

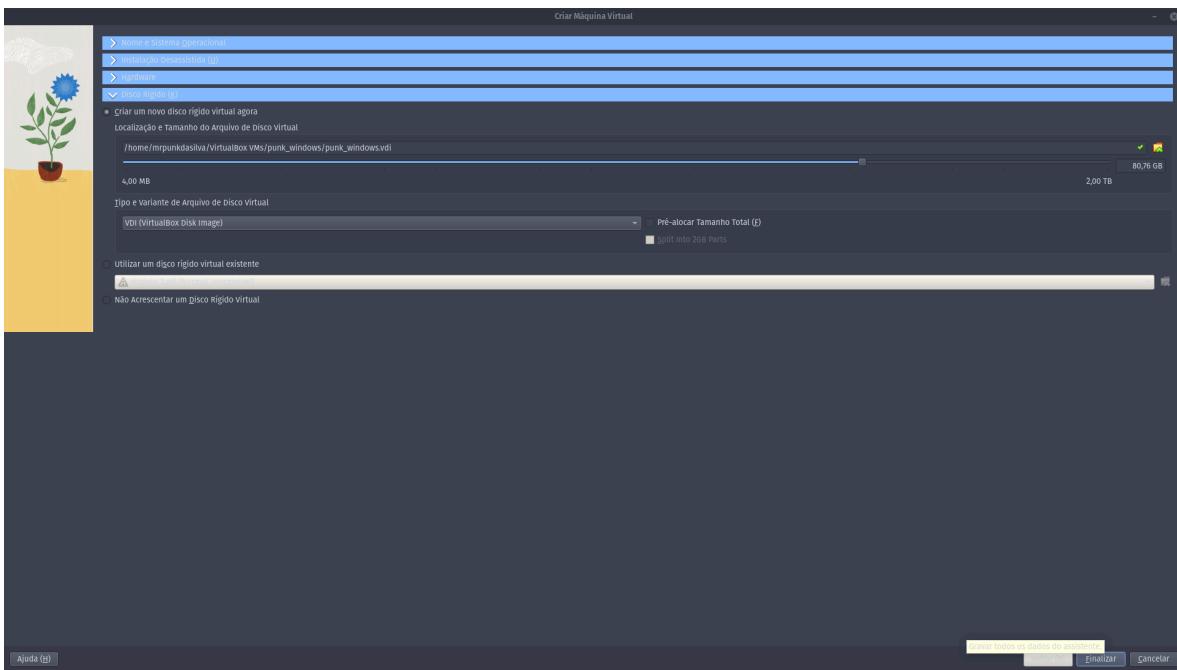
i Atente-se que computadores não lidam bem com números ímpares.



Configuração de memória RAM para a máquina virtual

- A quantidade de RAM alocada afetará diretamente o desempenho da sua máquina virtual. Certifique-se de deixar RAM suficiente para o seu sistema host também.

4. Agora definimos o espaço de armazenamento, disco rígido:

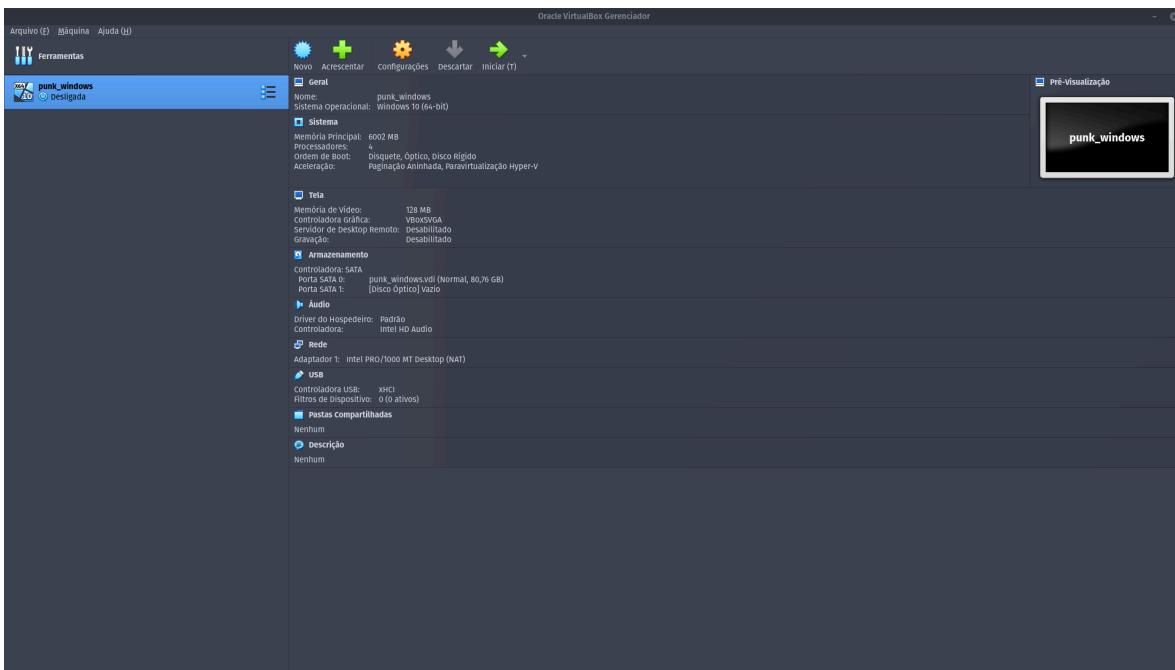


Configuração de disco rígido para a máquina virtual

- i** O tamanho do disco virtual deve ser suficiente para o sistema operacional e os programas que você planeja instalar. Você pode aumentar o tamanho posteriormente, mas é mais complicado diminuí-lo.

5. Com tudo criado, basta ir em **Finish**:

6. Temos então nossa primeira máquina virtual criada:



Máquina virtual Windows criada no VirtualBox

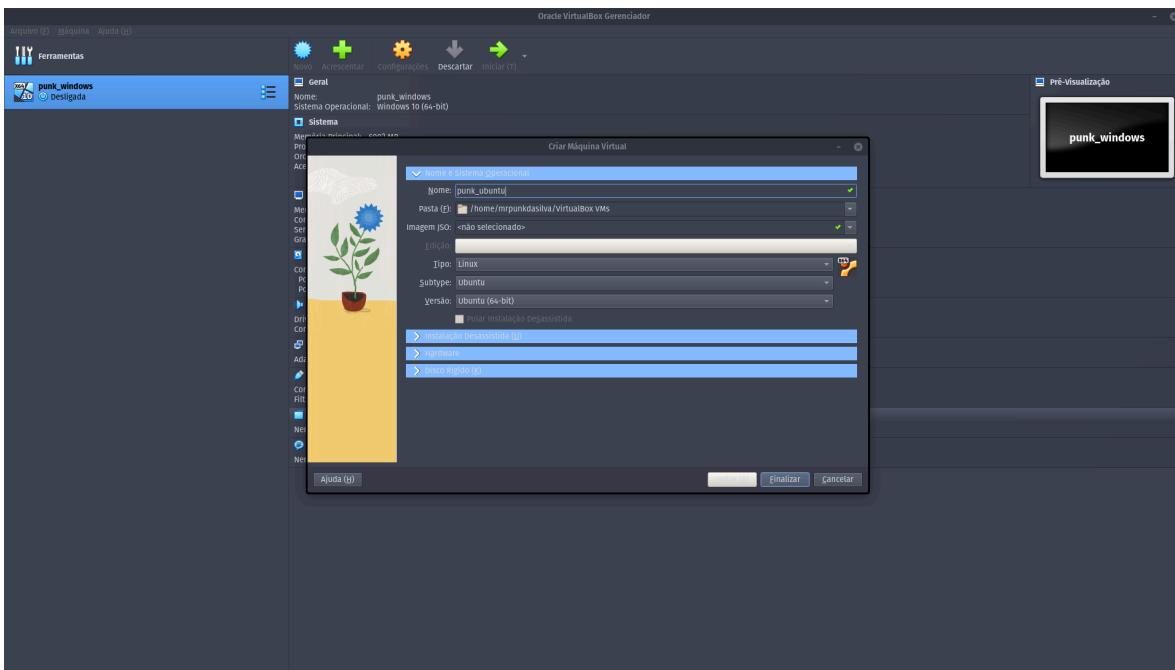
- i** Parabéns! Você criou sua primeira máquina virtual. Agora está pronta para receber o sistema operacional.

Ubuntu

1. Com o VirtualBox aberto, pressione: **CTRL** + **N**

- i** Você pode acessar a opção também por: Machine > Add

2. Definir nome, sistema e versão:

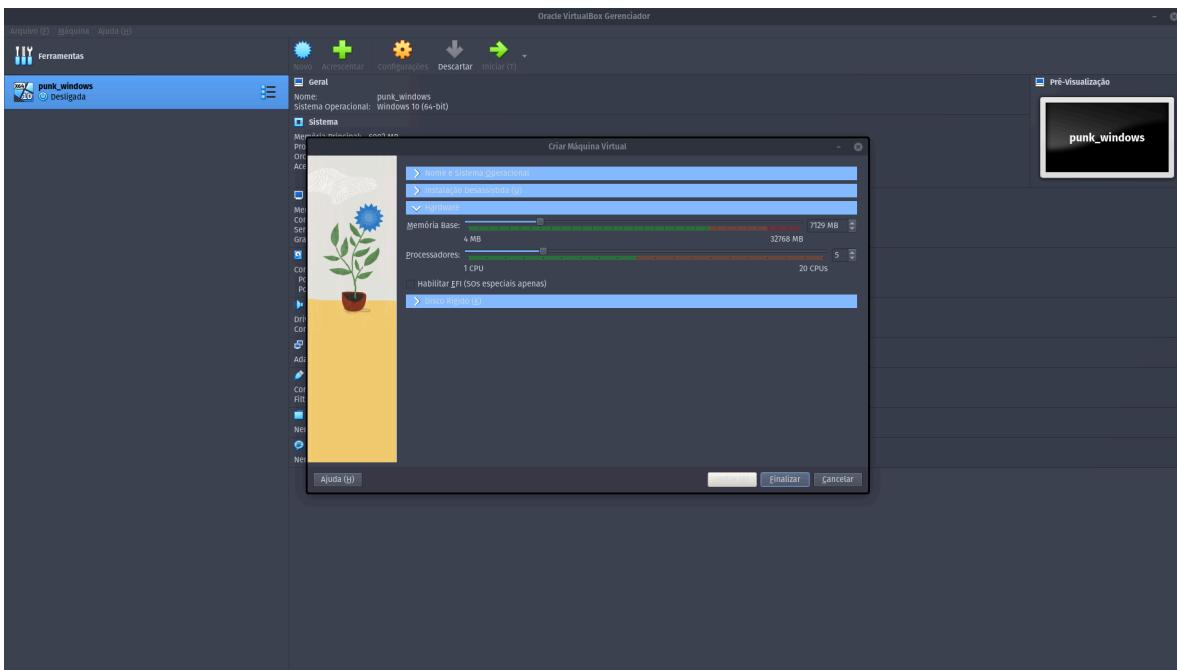


Tela de criação de máquina virtual Ubuntu no VirtualBox

- i** Certifique-se de selecionar "Ubuntu" como o tipo de sistema e escolha a versão correta que você planeja instalar.

3. Agora vamos definir a memória RAM. É bom deixarmos no mínimo 4GB

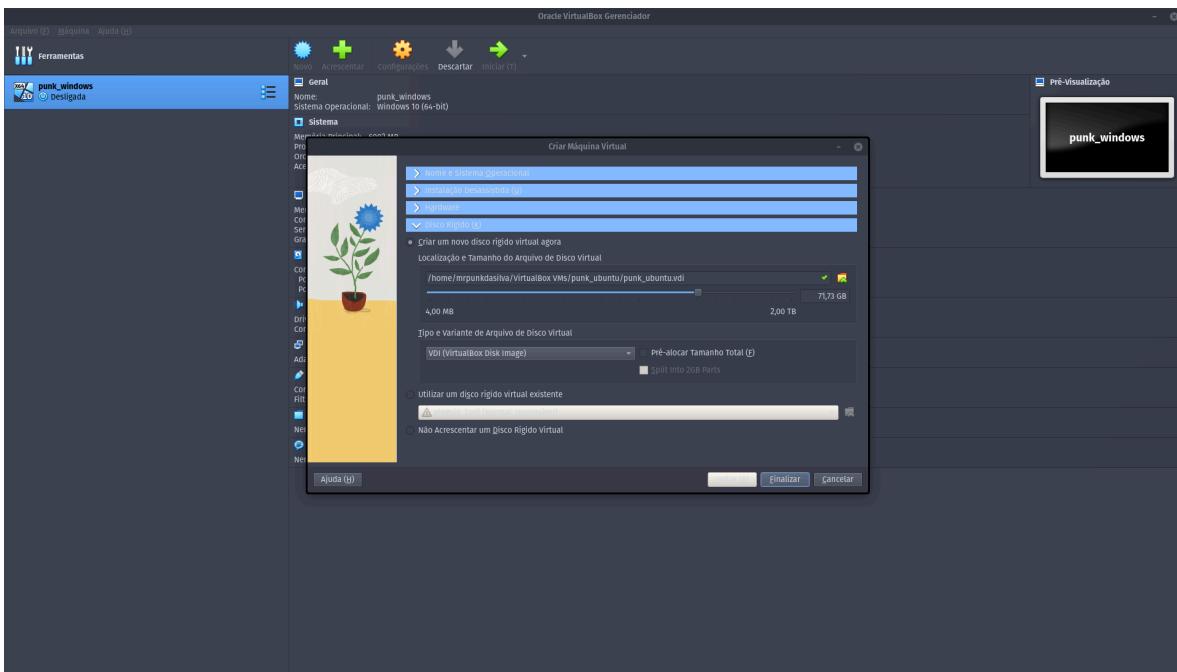
- i** Atente-se que computadores não lidam bem com números ímpares.



Configuração de memória RAM para a máquina virtual Ubuntu

- i** O Ubuntu geralmente requer menos RAM que o Windows, mas 4GB é uma boa quantidade para garantir um desempenho suave.

4. Agora definimos o espaço de armazenamento, disco rígido:

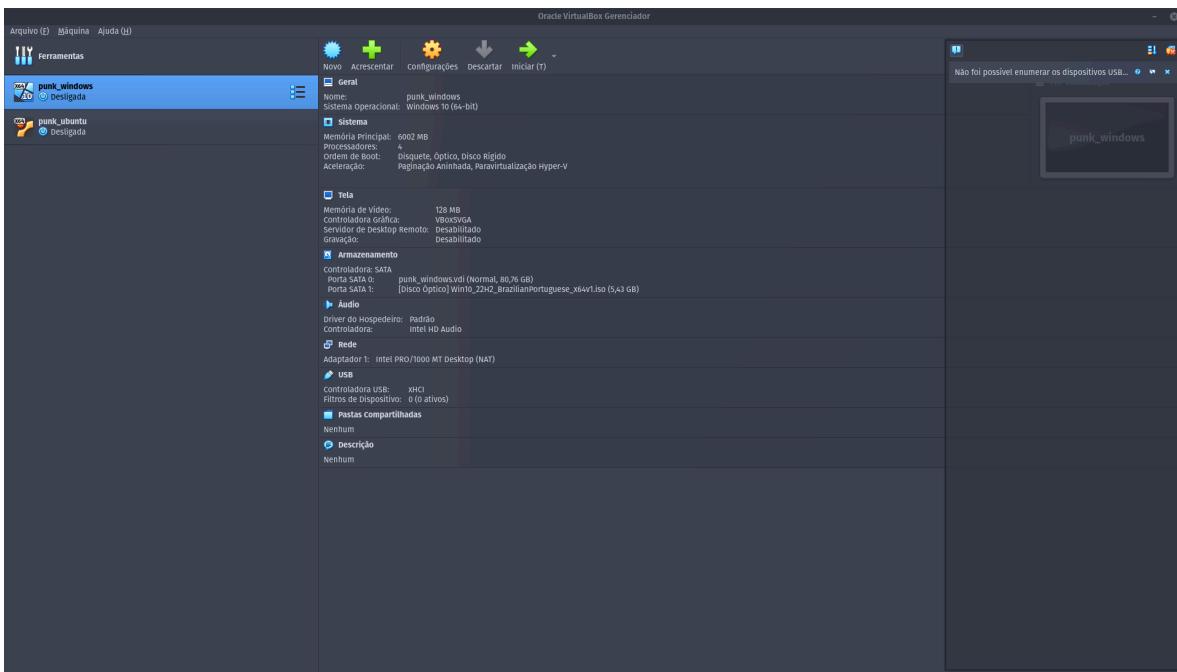


Configuração de disco rígido para a máquina virtual Ubuntu

- i** O Ubuntu geralmente requer menos espaço em disco que o Windows. 20GB é suficiente para uma instalação básica, mas considere alocar mais se planeja instalar muitos programas.

5. Com tudo criado, basta ir em **Finish**:

6. Temos então nossa primeira máquina virtual criada:



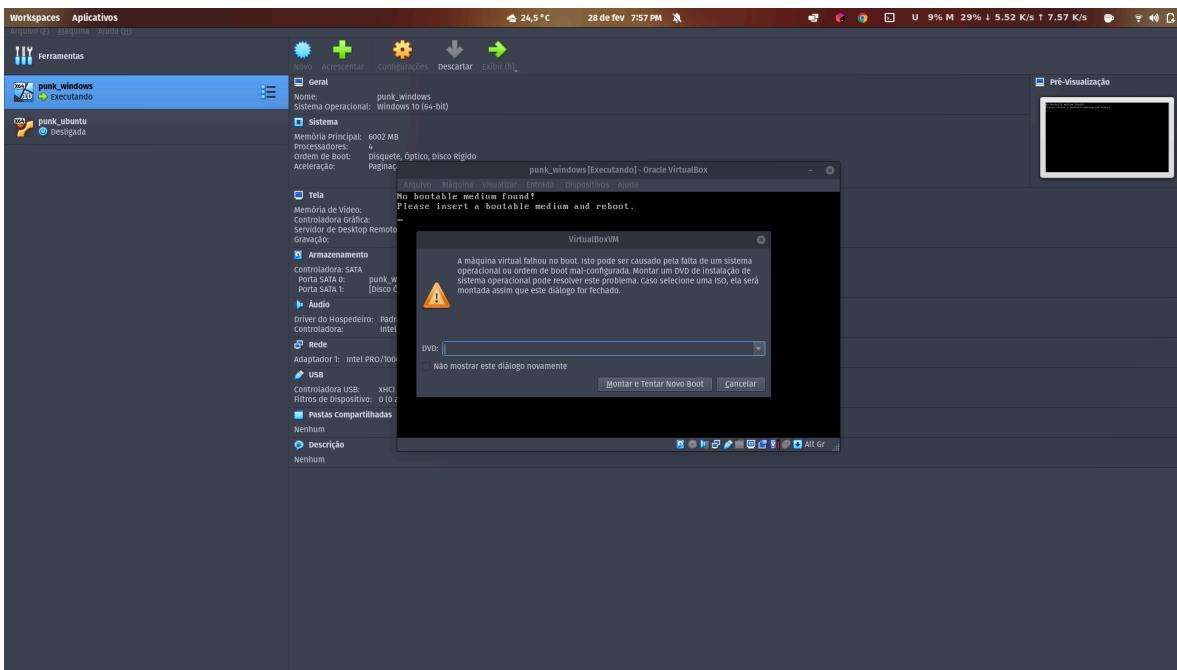
Máquina virtual Ubuntu criada no VirtualBox

- i** Sua máquina virtual Ubuntu está pronta para receber o sistema operacional. O próximo passo será iniciar a instalação do Ubuntu.

Logar nas VMs recém-criadas

Ao executar as máquinas, nenhum sistema será inicializado, já que não foi definida nenhuma ISO (Imagen de um Sistema Operacional). Assim, as máquinas ficam em seu estado puro, sem nenhum sistema operacional, e são inutilizáveis.

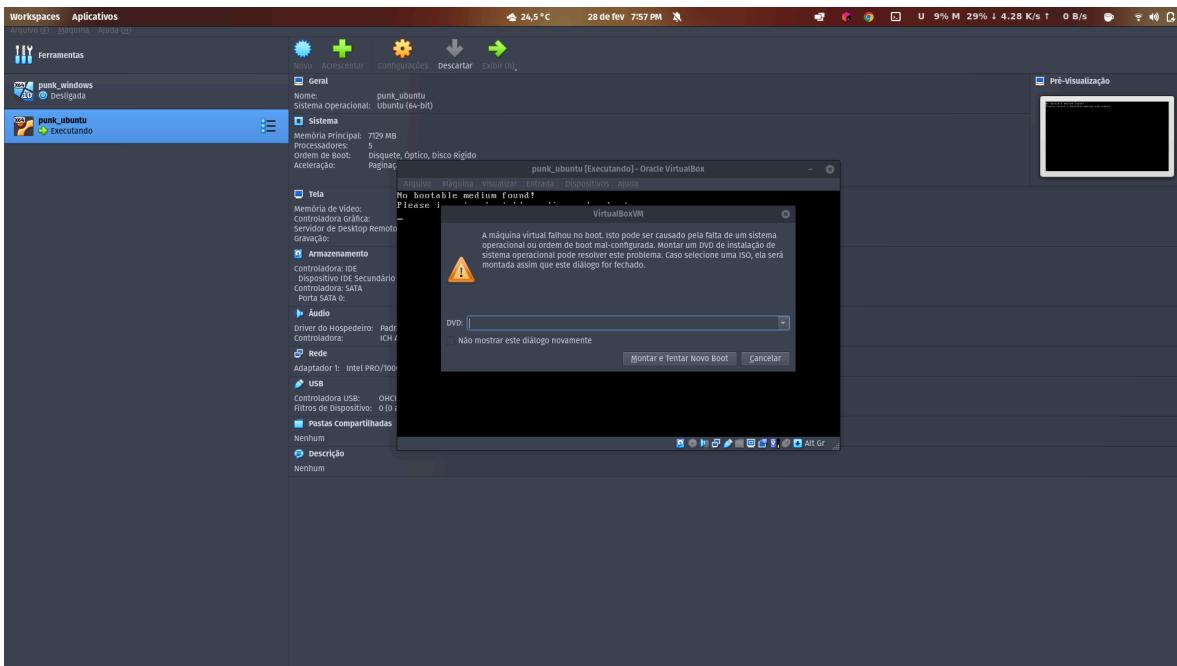
Windows



Tela inicial da máquina virtual Windows sem sistema operacional

- i** Esta tela indica que a máquina virtual está pronta, mas ainda não tem um sistema operacional instalado.

Ubuntu



Tela inicial da máquina virtual Ubuntu sem sistema operacional

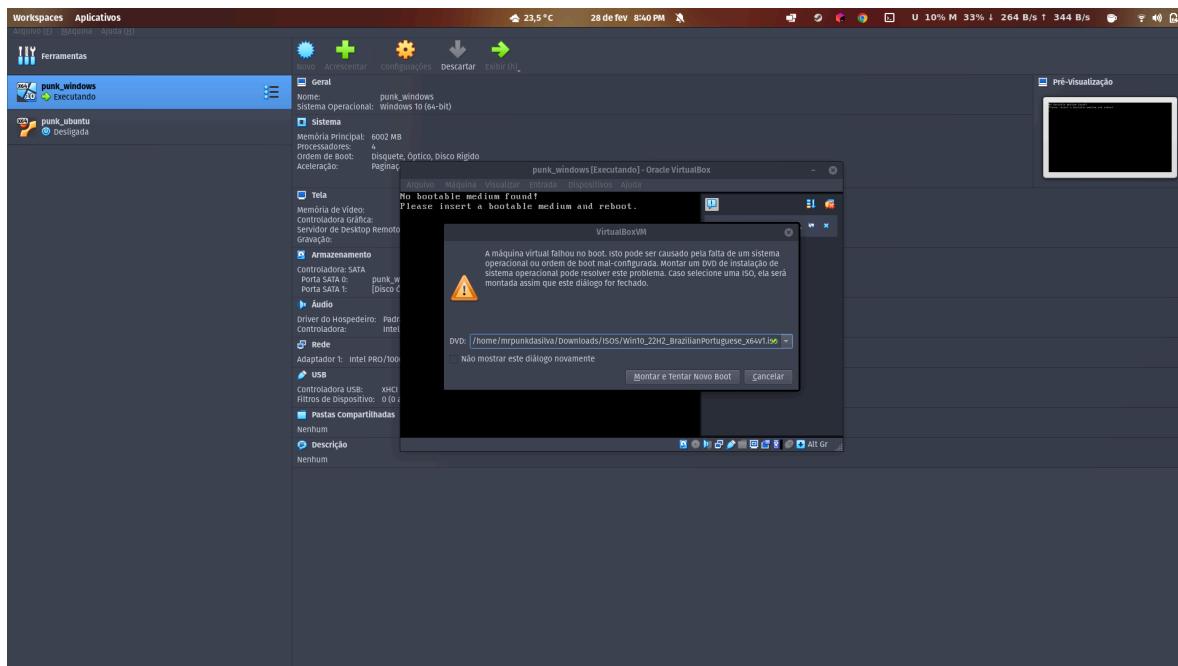
- i** Semelhante à máquina Windows, esta tela mostra que a máquina virtual Ubuntu está criada, mas ainda precisa de um sistema operacional.

Configurando VMs para os SOs

Para tornar as VMs utilizáveis, precisamos definir as ISOs que serão as imagens do sistema usadas para instalar o sistema operacional.

Windows

Com a máquina em execução e este pop-up aparecendo, selecionamos onde está a ISO do Windows que foi baixada nos passos anteriores:

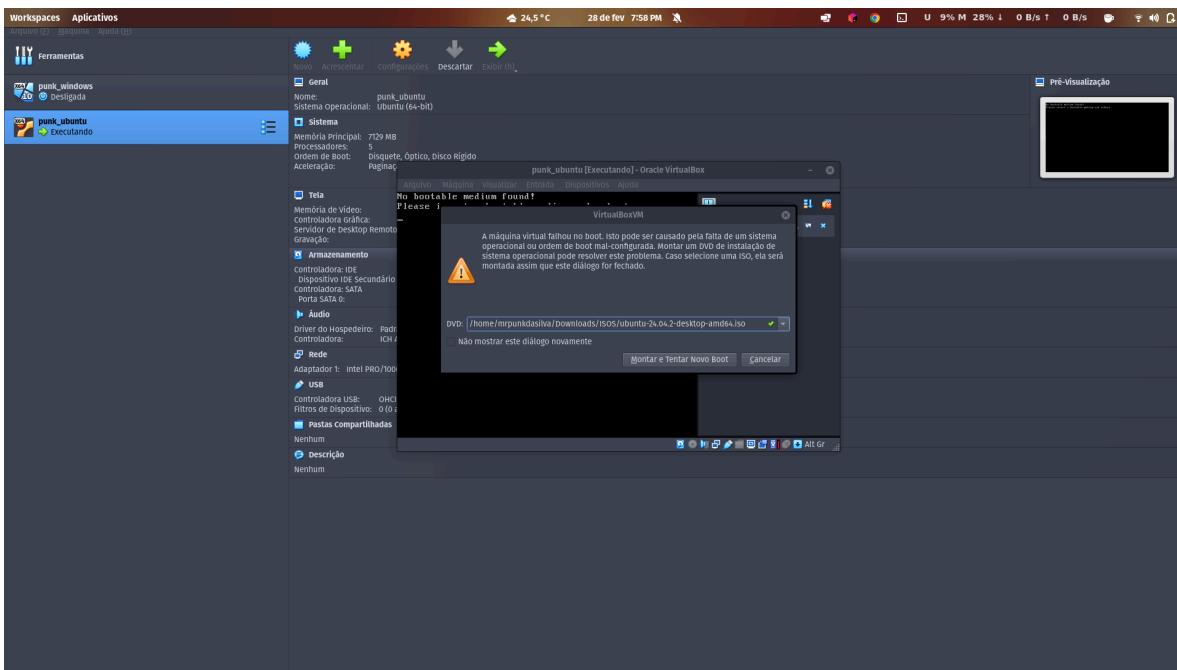


Seleção da ISO do Windows para instalação

- i** Certifique-se de selecionar a ISO correta do Windows que você baixou anteriormente. Isso iniciará o processo de instalação do Windows na sua máquina virtual.

Ubuntu

Com a máquina em execução e este pop-up aparecendo, selecionamos onde está a ISO do Ubuntu que foi baixada nos passos anteriores:



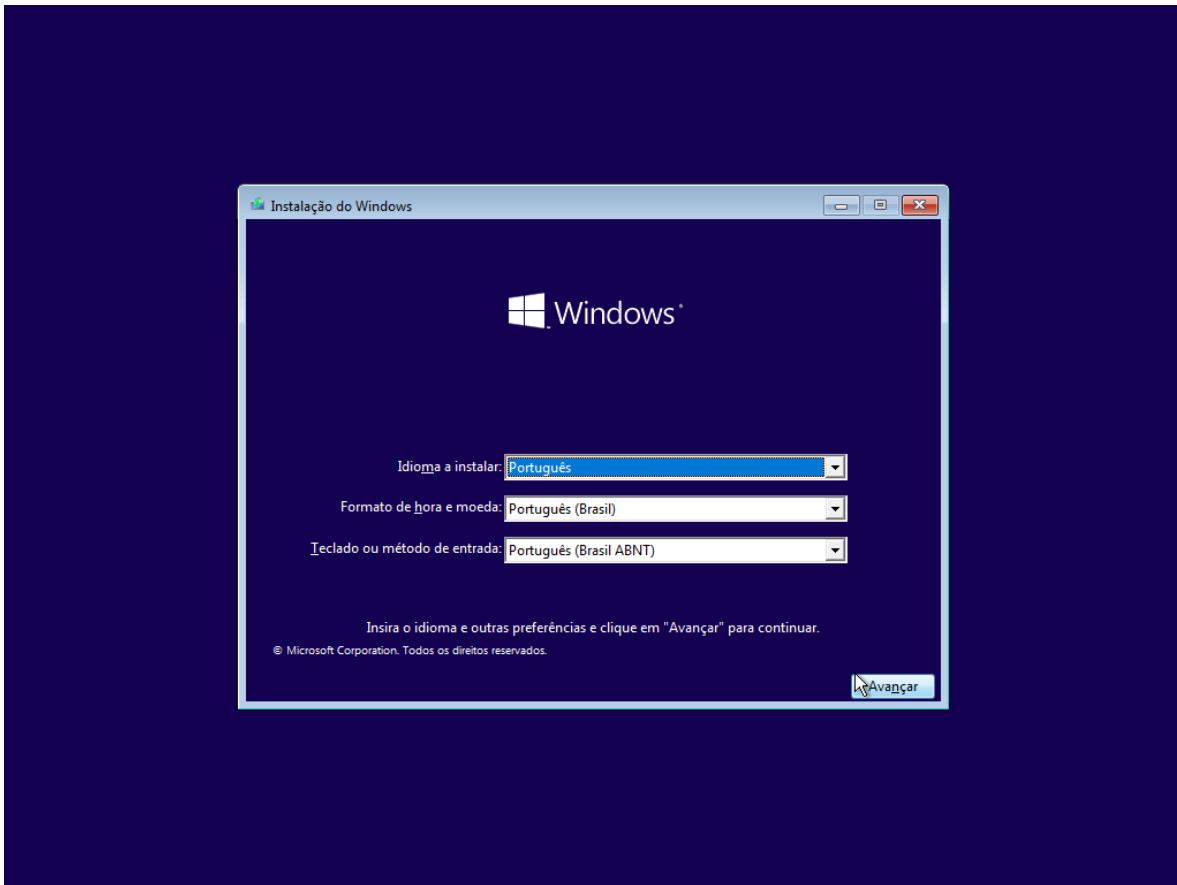
Seleção da ISO do Ubuntu para instalação

- i** Selecione a ISO do Ubuntu que você baixou. Isso iniciará o processo de instalação do Ubuntu na sua máquina virtual.

Logar nas VMs

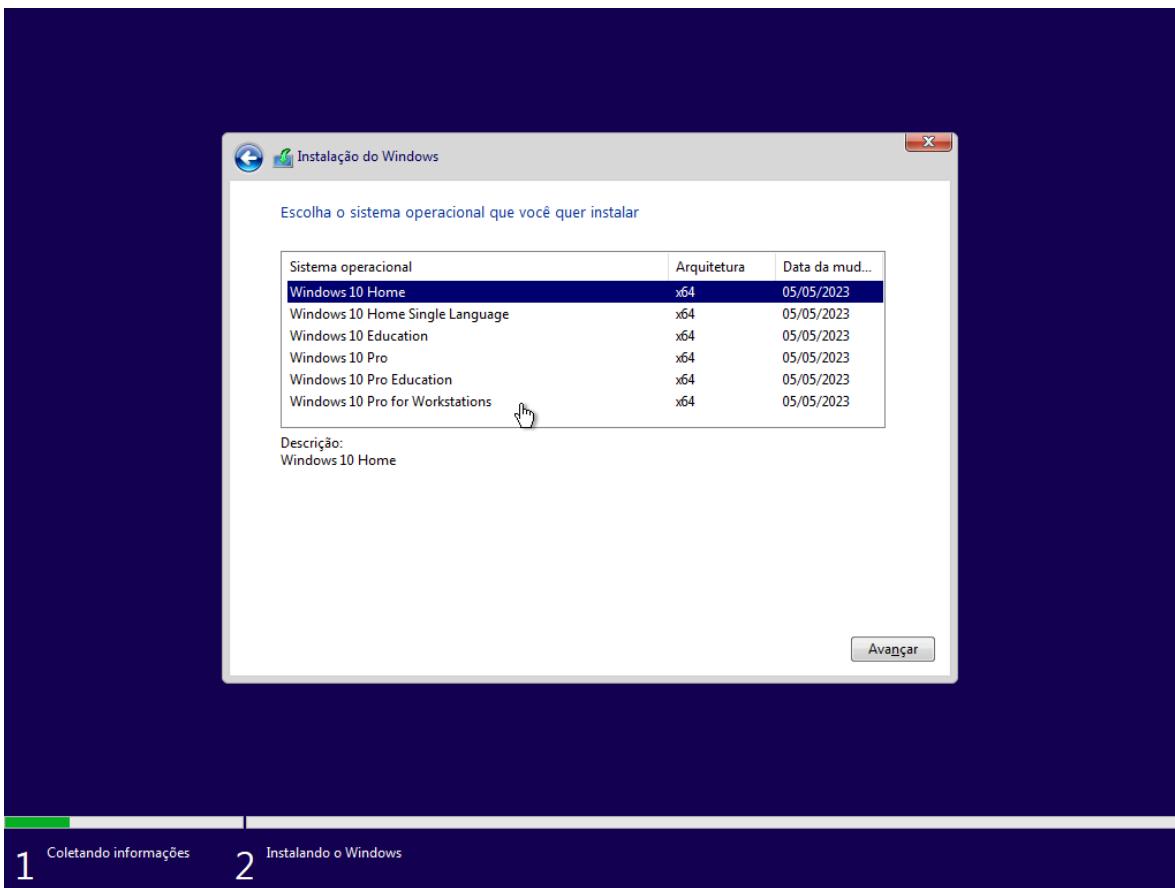
Agora, com tudo o que vimos, podemos fazer a instalação do sistema. Para isso, devemos logar ou entrar nas máquinas que criamos e então fazer as etapas de instalação do Windows e Ubuntu.

Windows



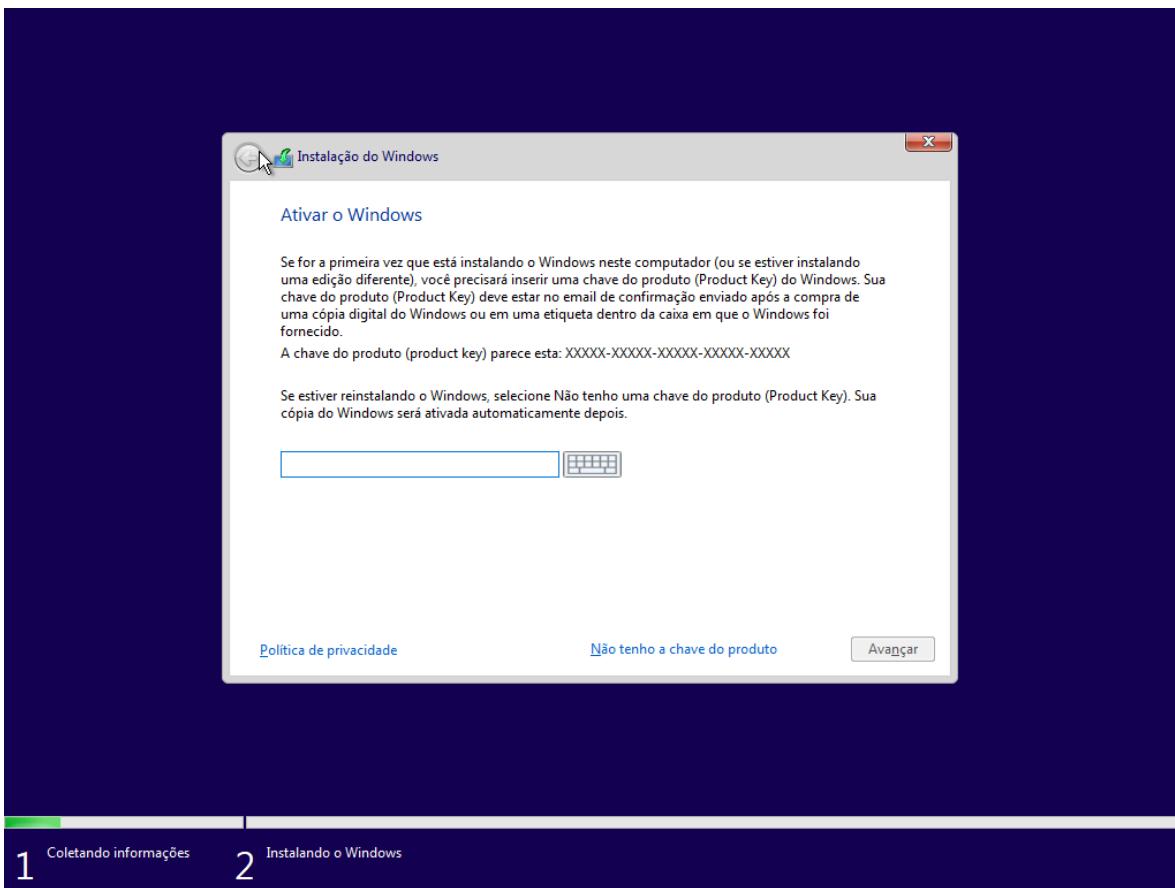
Tela inicial de instalação do Windows

- Esta é a tela inicial de instalação do Windows. A partir daqui, você seguirá as etapas para configurar seu sistema Windows.



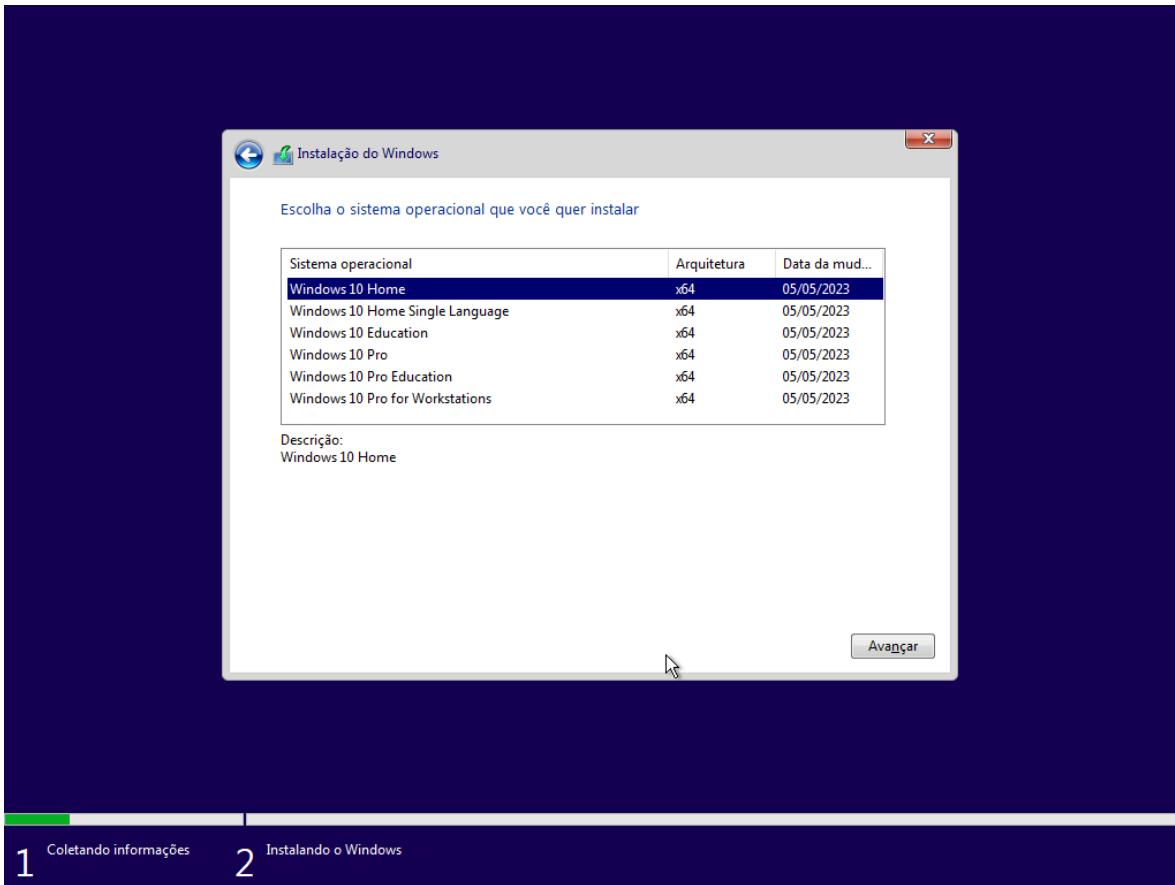
Seleção de idioma, formato de hora e moeda, e layout de teclado

- Escolha as opções que melhor se adequam à sua região e preferências de idioma.



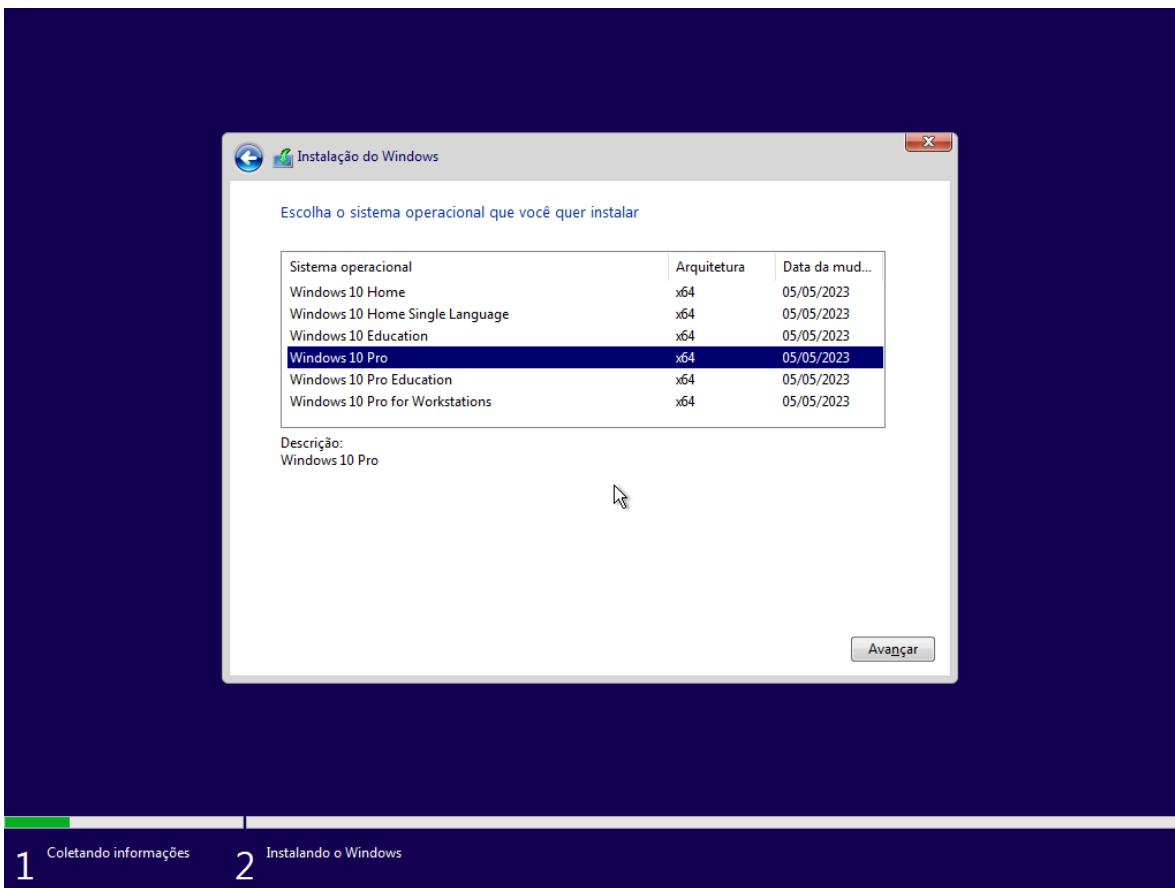
Botão "Instalar agora" para iniciar a instalação do Windows

- Clique em "Instalar agora" para começar o processo de instalação do Windows.



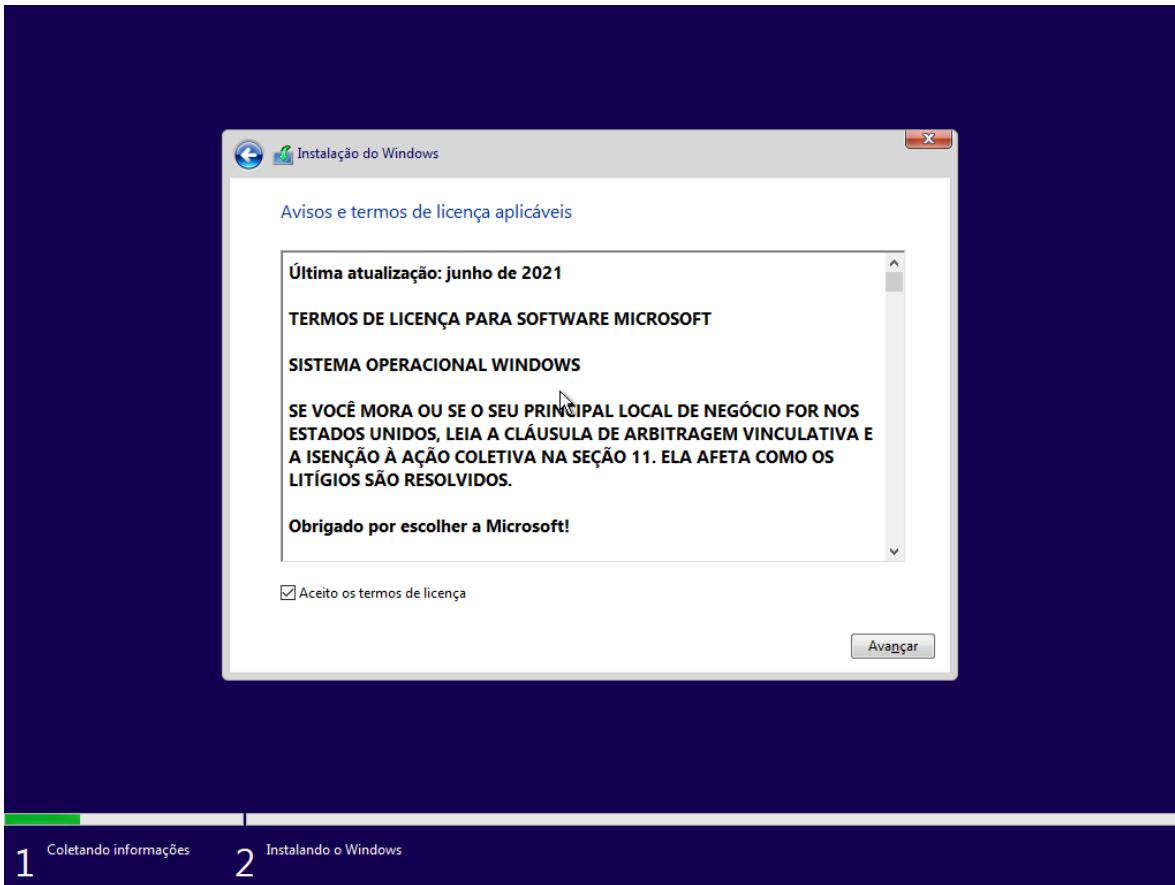
Inserção da chave do produto Windows

- Se você tiver uma chave do produto, insira-a aqui. Caso contrário, você pode pular esta etapa e ativar o Windows posteriormente.



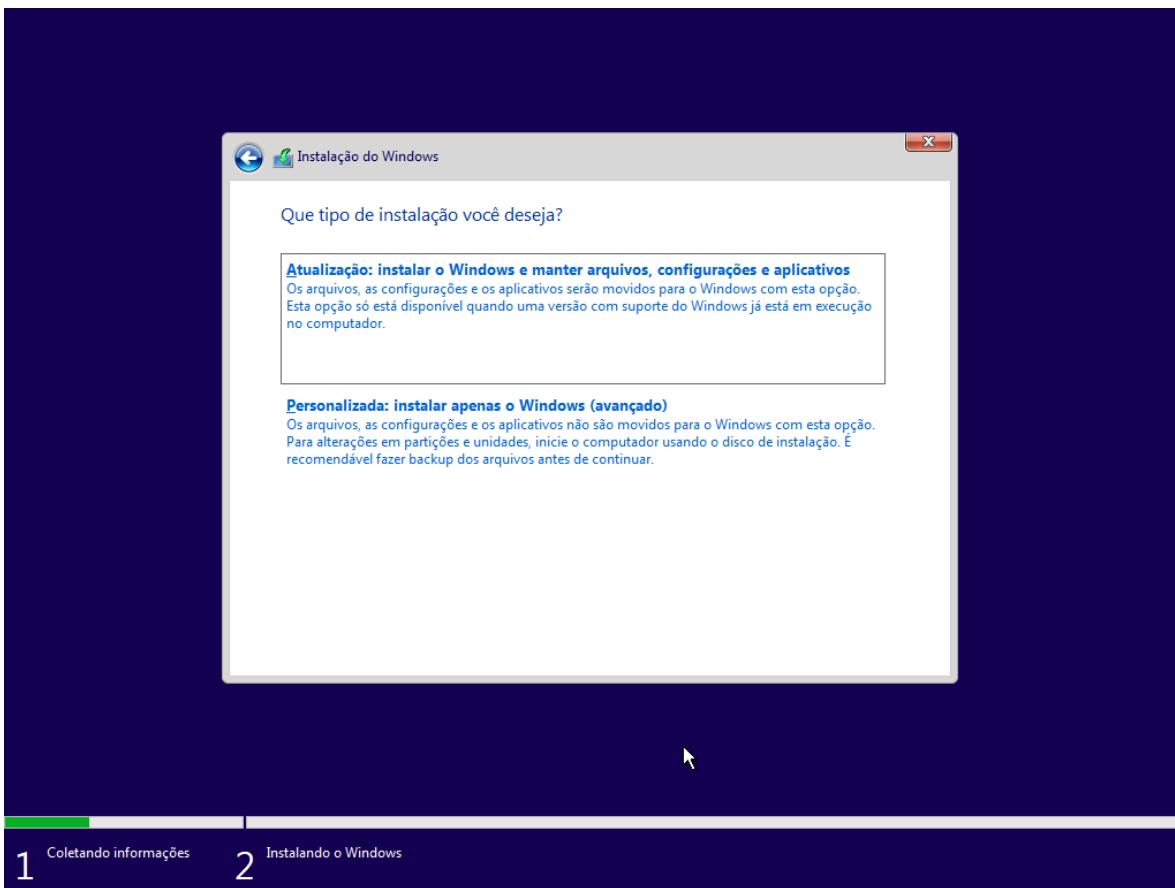
Seleção da versão do Windows a ser instalada

- Escolha a versão do Windows que deseja instalar. Para uso pessoal, "Windows 10 Home" geralmente é suficiente.



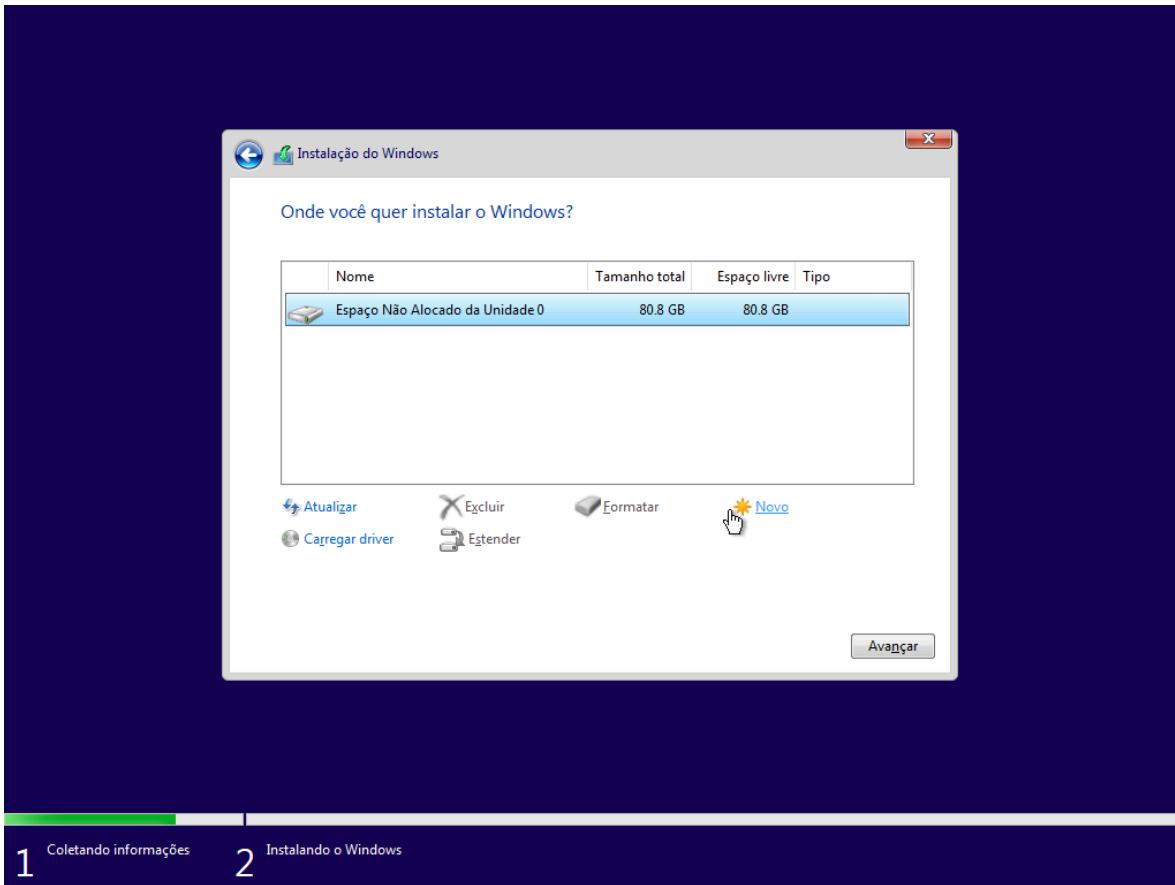
Aceitação dos termos de licença do Windows

- i Leia e aceite os termos de licença para prosseguir com a instalação.



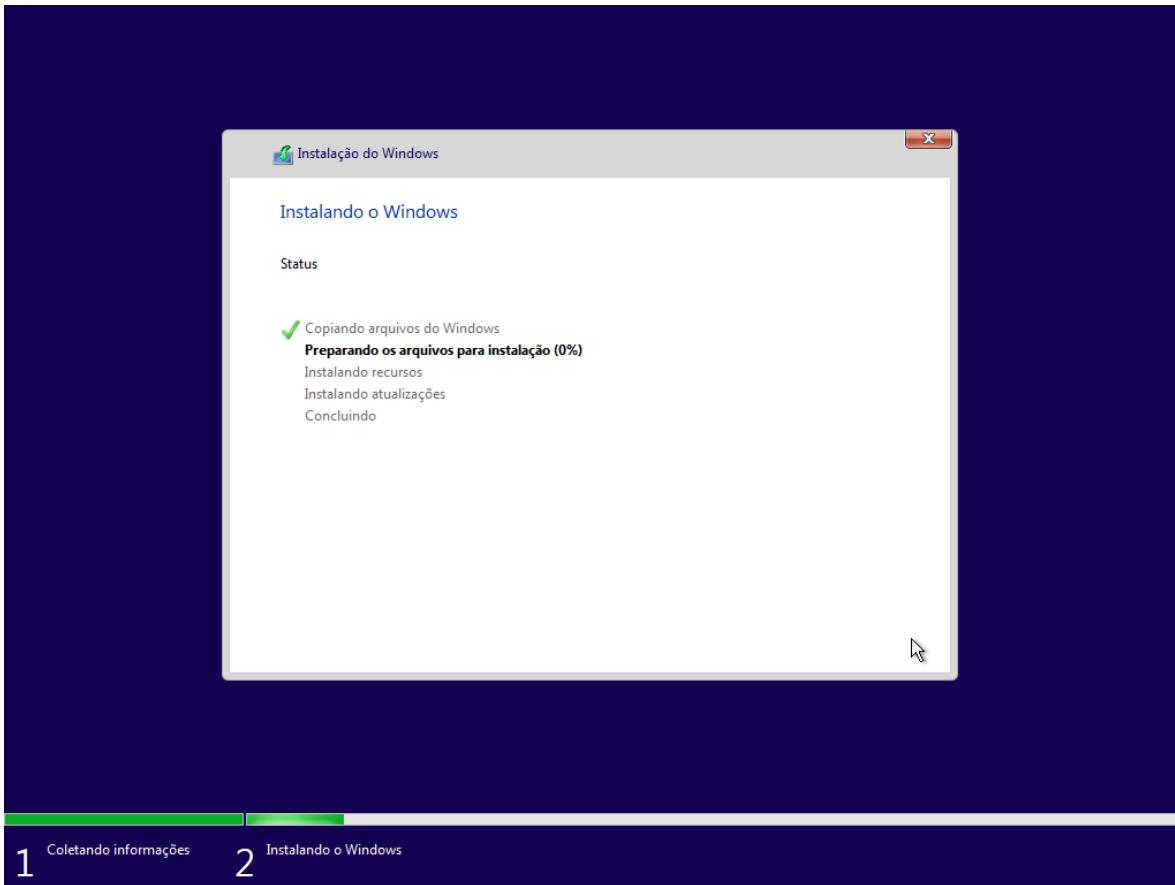
Escolha do tipo de instalação: Atualização ou Personalizada

- Para uma nova instalação em uma máquina virtual, escolha "Instalação Personalizada".



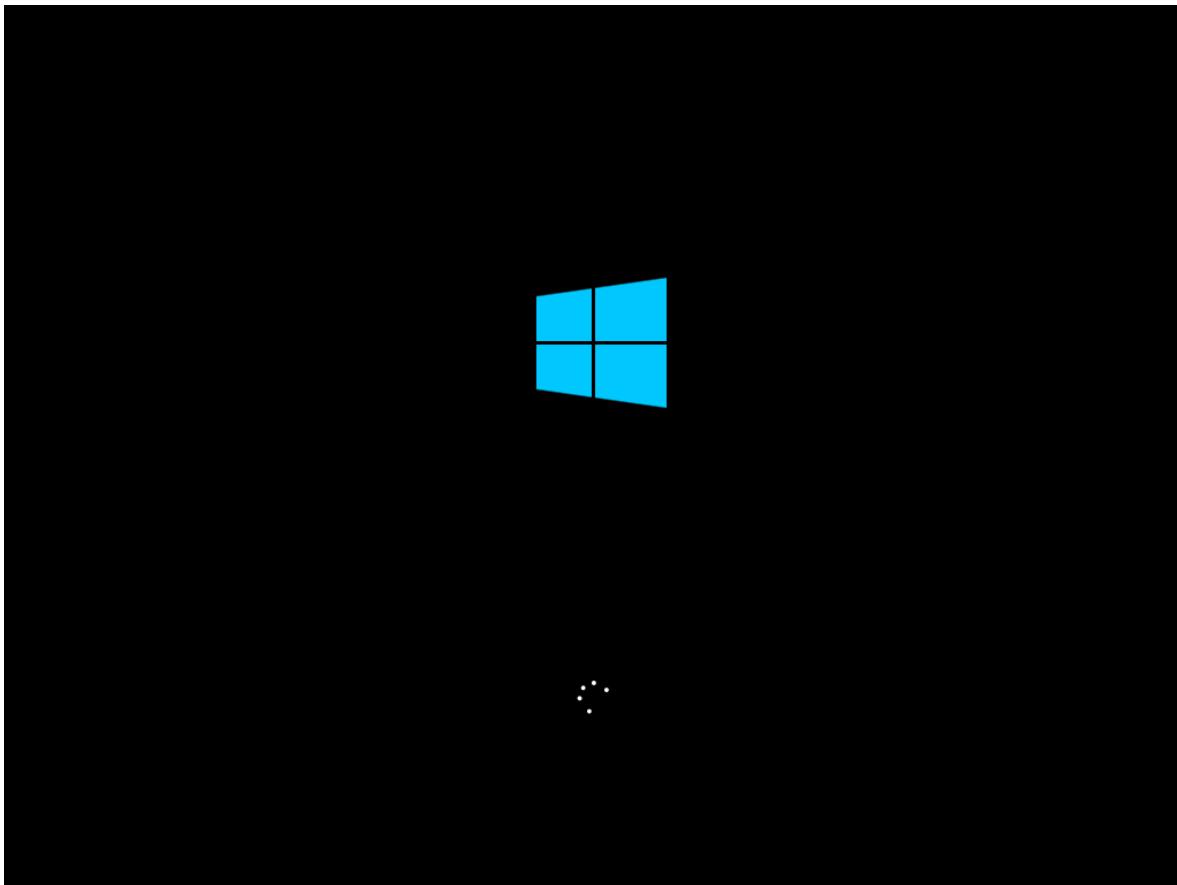
Seleção do disco onde o Windows será instalado

- Selecione o disco virtual que você criou anteriormente para instalar o Windows.



Progresso da instalação do Windows

- i** A instalação do Windows está em andamento. Isso pode levar alguns minutos.



Reinicialização do sistema após a instalação inicial

⚠️ Após a instalação inicial, o sistema irá

Vamos começar

Requisitos dos sistemas

Confira os requisitos dos sistemas operacionais que serão usados:

- Windows 10 (<https://support.microsoft.com/pt-br/windows/requisitos-do-sistema-do-windows-10-6d4e9a79-66bf-7950-467c-795cf0386715>)
- Ubuntu Desktop (<https://ubuntu.com/server/docs/system-requirements>)

Instalando a ferramenta

Vamos usar o VirtualBox.

- Para Windows (<https://download.virtualbox.org/virtualbox/7.1.6/VirtualBox-7.1.6-167084-Win.exe>)
- Para Linux (https://www.virtualbox.org/wiki/Linux_Downloads)

Instalar ISOs (Windows e Ubuntu)

- Windows (<https://www.microsoft.com/pt-br/software-download/windows10ISO>)

i Na ISO oficial do Windows, o link acima, vêm todas as versões.

- Ubuntu Desktop (<https://ubuntu.com/download/desktop>)

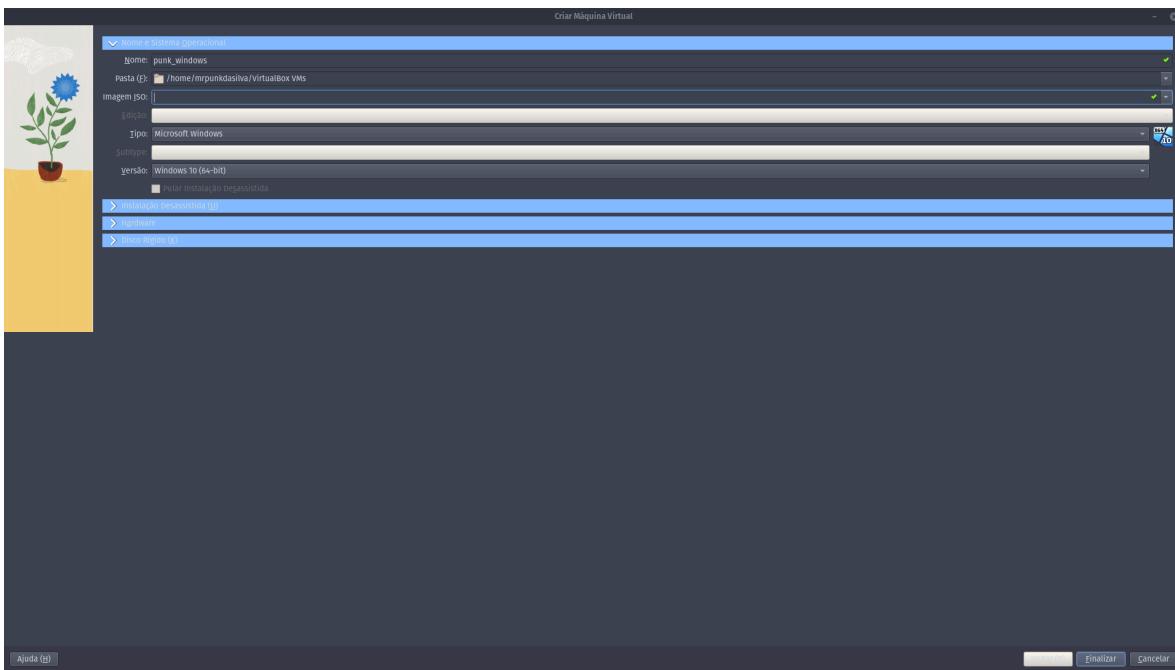
Criando Máquinas Virtuais

Windows

1. Com o VirtualBox aberto, pressione: **CTRL** + **N**

i Você pode acessar a opção também por: Machine > Add

2. Definir nome, sistema e versão:

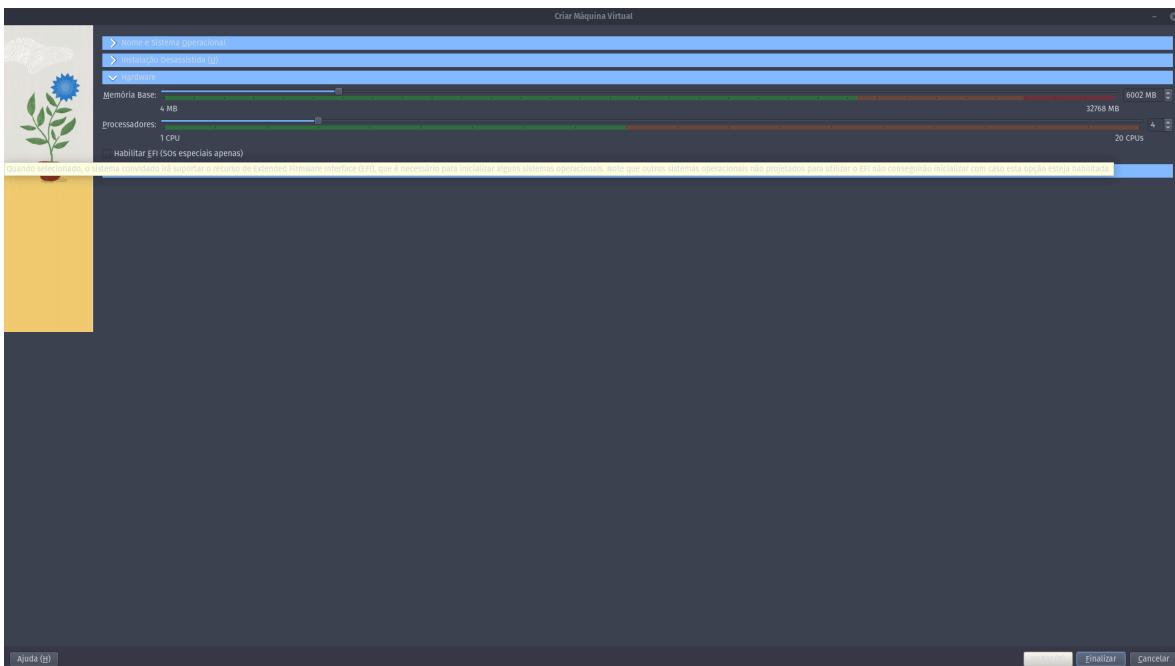


Tela de criação de máquina virtual no VirtualBox

3. Agora vamos definir a memória RAM. É bom deixarmos no mínimo 4GB

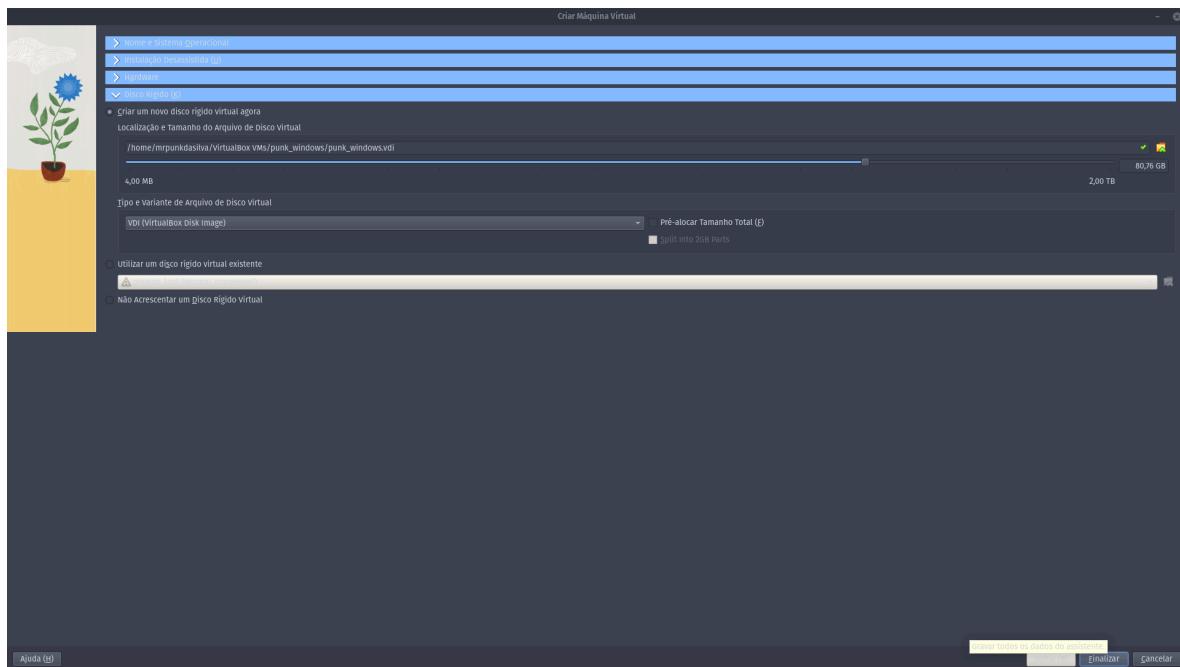


Atente-se que computadores não lidam bem com números ímpares.



Configuração de memória RAM para a máquina virtual

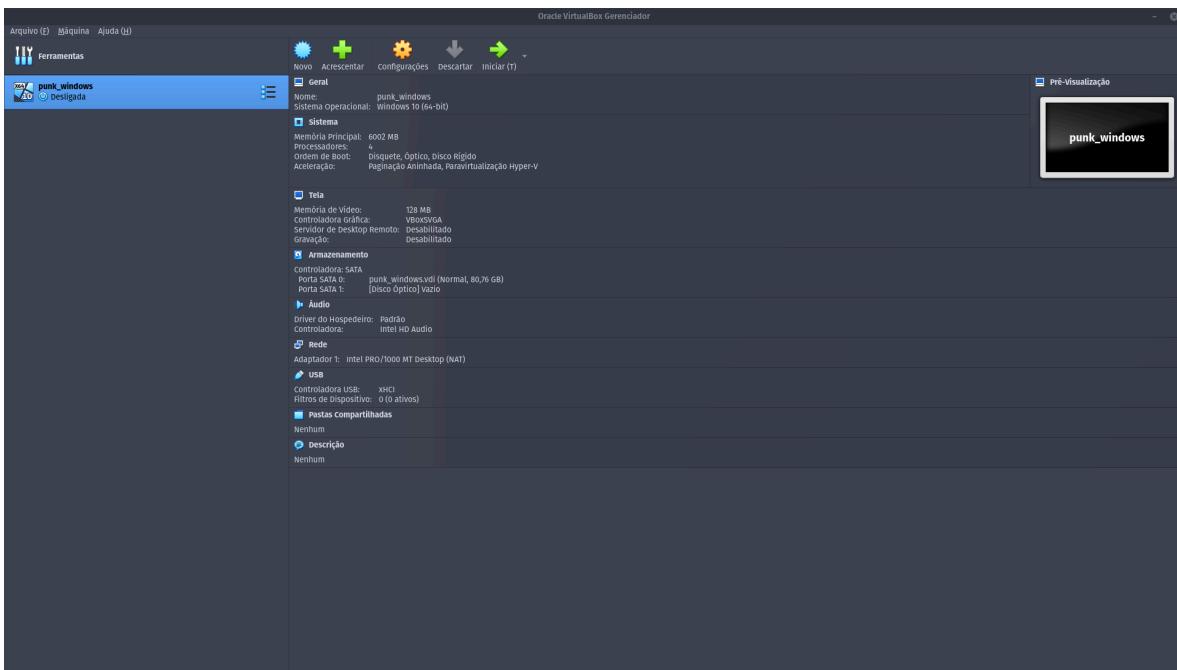
4. Agora definimos o espaço de armazenamento, disco rígido:



Configuração de disco rígido para a máquina virtual

5. Com tudo criado, basta ir em **Finish**:

6. Temos então nossa primeira máquina virtual criada:



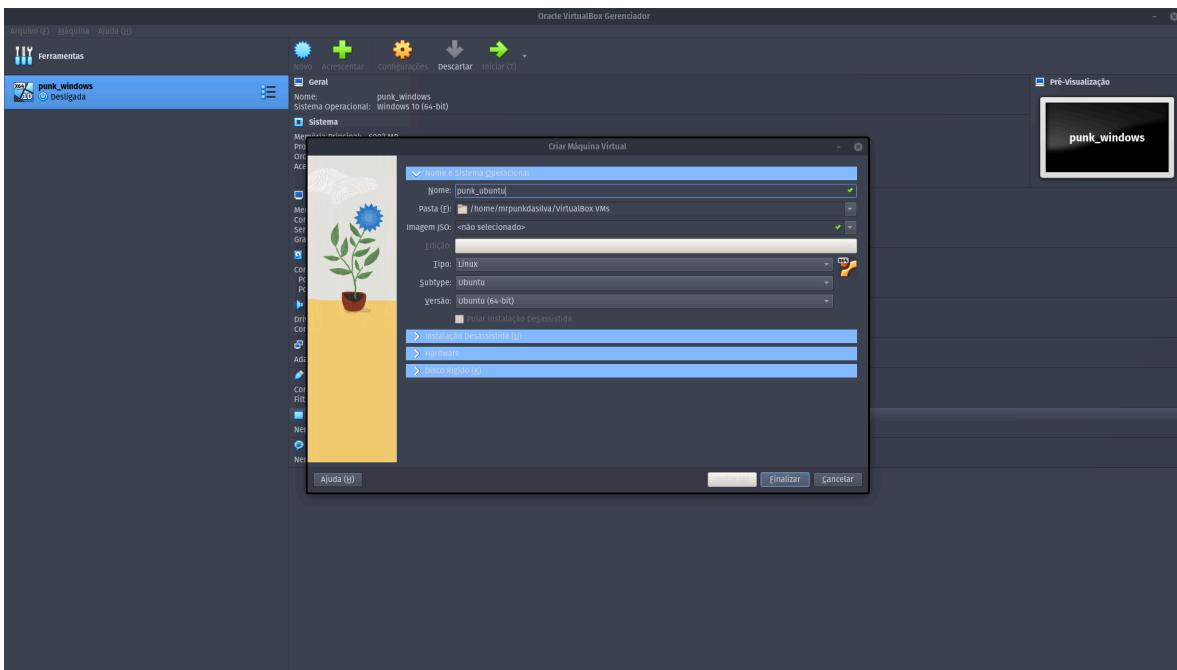
Máquina virtual Windows criada no VirtualBox

Ubuntu

1. Com o VirtualBox aberto, pressione: **CTRL** + **N**

i Você pode acessar a opção também por: Machine > Add

2. Definir nome, sistema e versão:

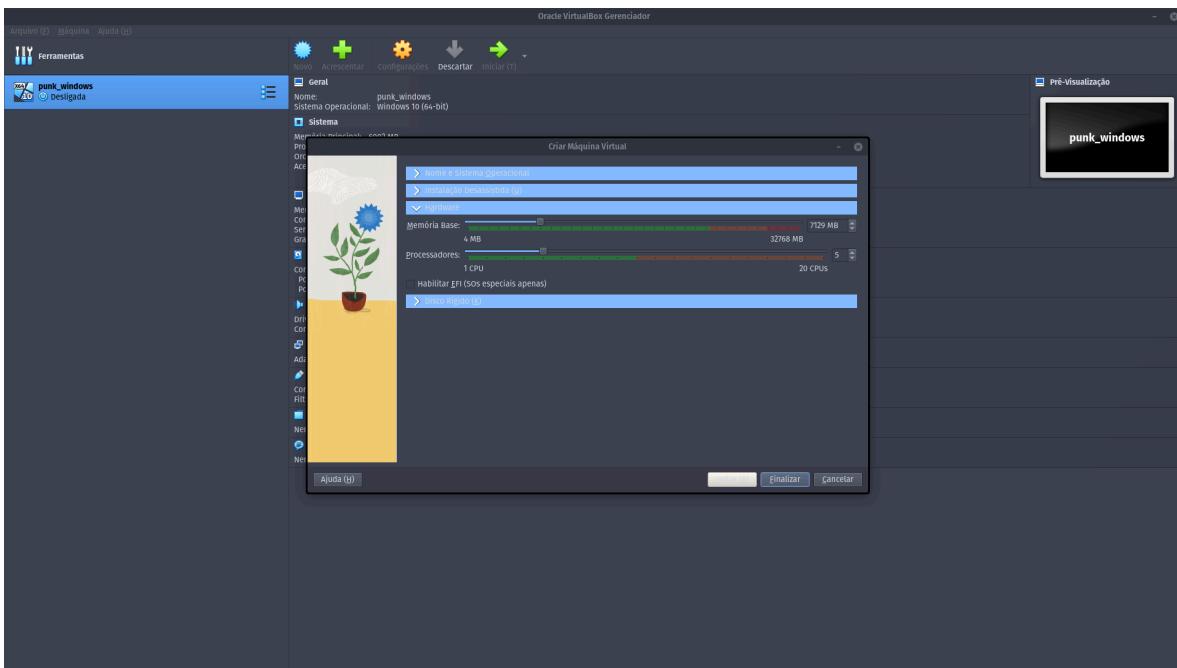


Tela de criação de máquina virtual Ubuntu no VirtualBox

3. Agora vamos definir a memória RAM. É bom deixarmos no mínimo 4GB

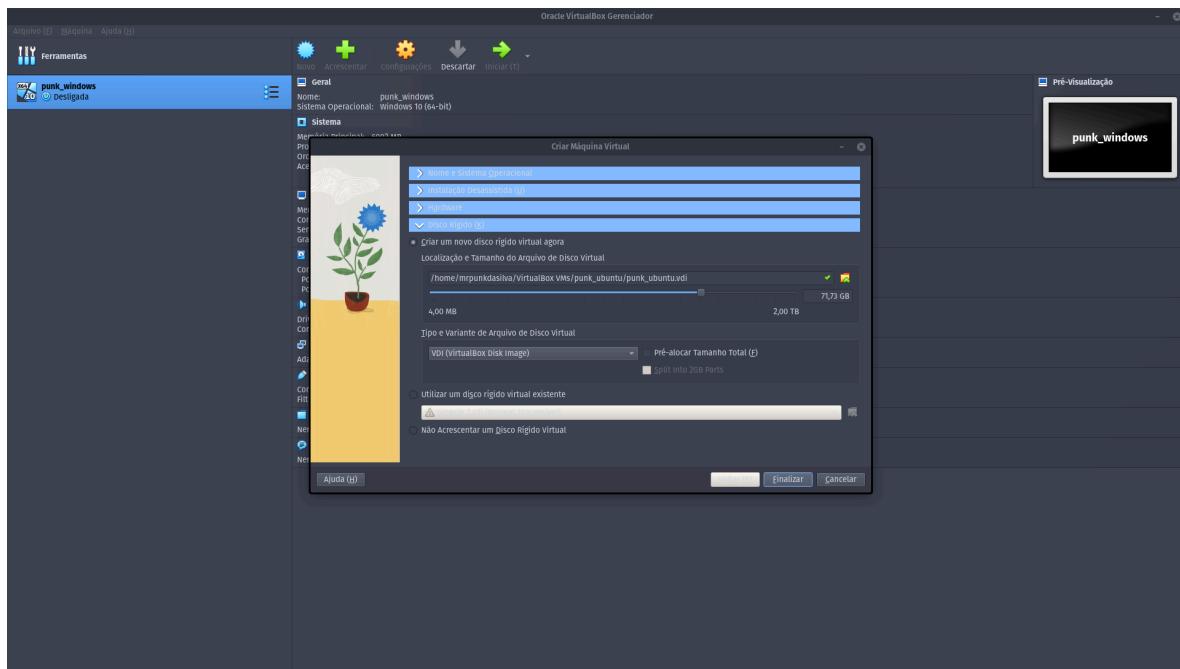


Atente-se que computadores não lidam bem com números ímpares.



Configuração de memória RAM para a máquina virtual Ubuntu

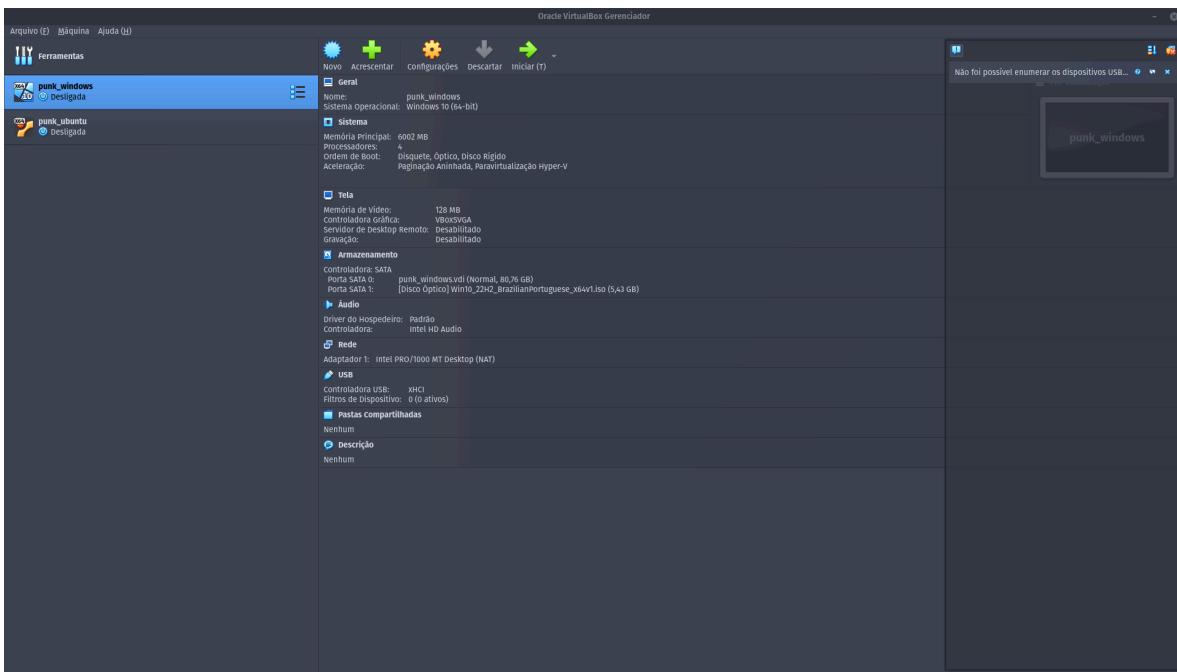
4. Agora definimos o espaço de armazenamento, disco rígido:



Configuração de disco rígido para a máquina virtual Ubuntu

5. Com tudo criado, basta ir em **Finish**:

6. Temos então nossa primeira máquina virtual criada:

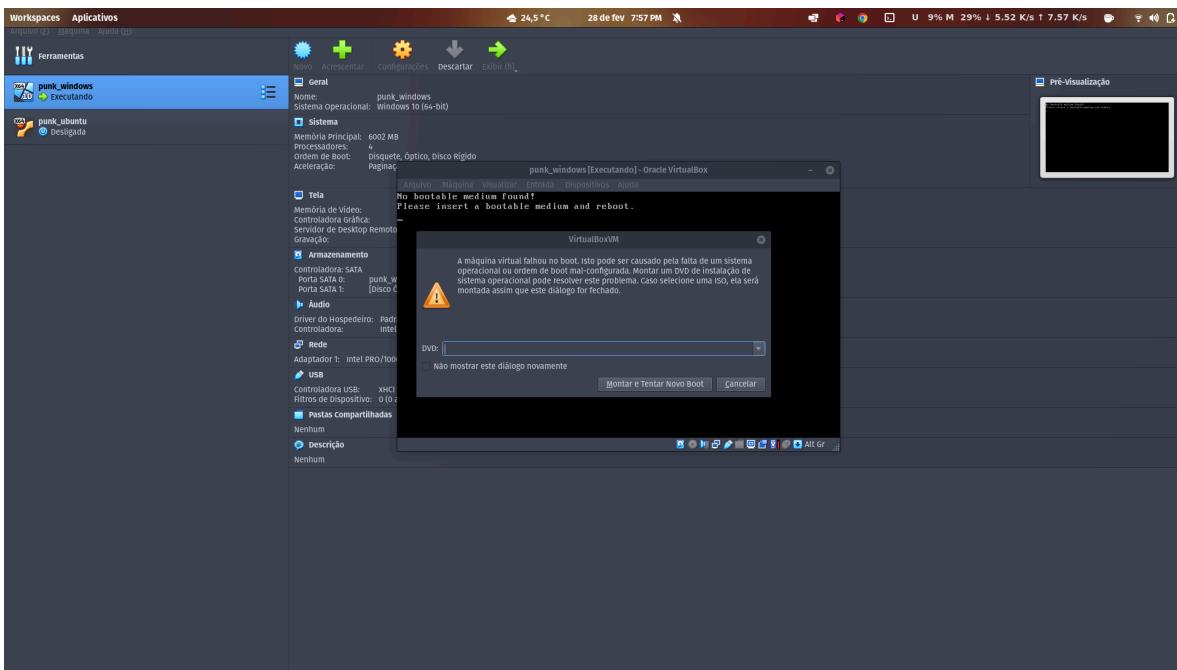


Máquina virtual Ubuntu criada no VirtualBox

Logar nas VMs recém-criadas

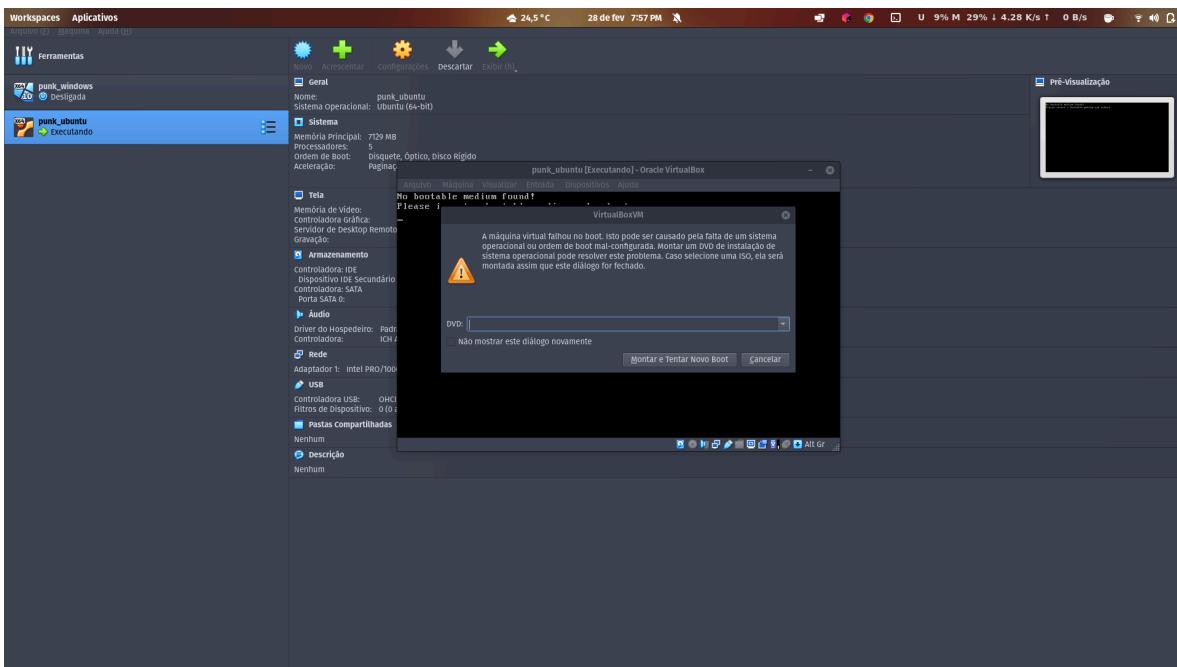
Ao executar as máquinas, nenhum sistema será inicializado, já que não foi definida nenhuma ISO (Imagem de um Sistema Operacional). Assim, as máquinas ficam em seu estado puro, sem nenhum sistema operacional, e são inutilizáveis.

Windows



Tela inicial da máquina virtual Windows sem sistema operacional

Ubuntu



Tela inicial da máquina virtual Ubuntu sem sistema operacional

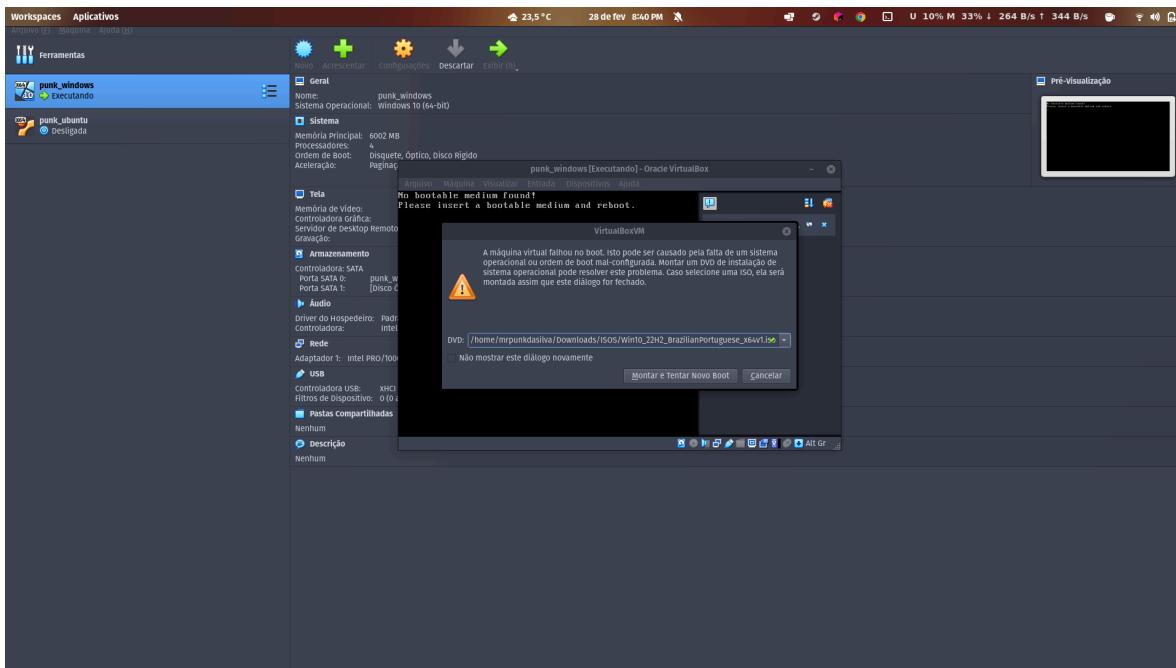
Configurando VMs para os SOs

Para tornar as VMs utilizáveis, precisamos definir as ISOs que serão as imagens do

sistema usadas para instalar o sistema operacional.

Windows

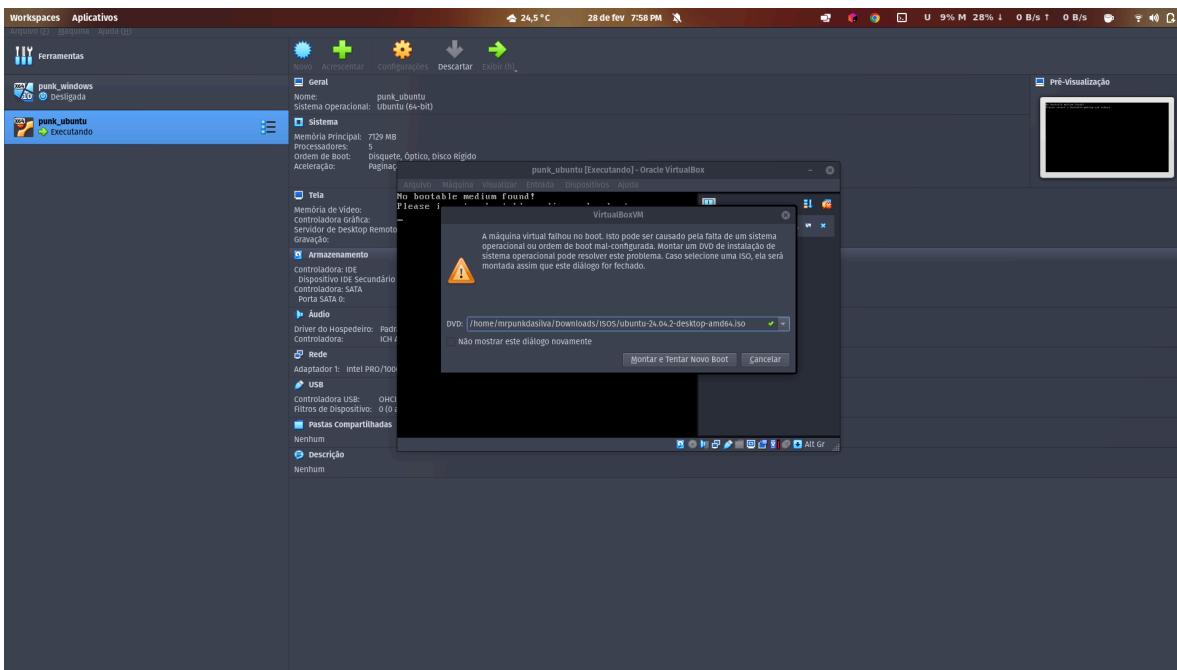
Com a máquina em execução e este pop-up aparecendo, selecionamos onde está a ISO do Windows que foi baixada nos passos anteriores:



Seleção da ISO do Windows para instalação

Ubuntu

Com a máquina em execução e este pop-up aparecendo, selecionamos onde está a ISO do Ubuntu que foi baixada nos passos anteriores:

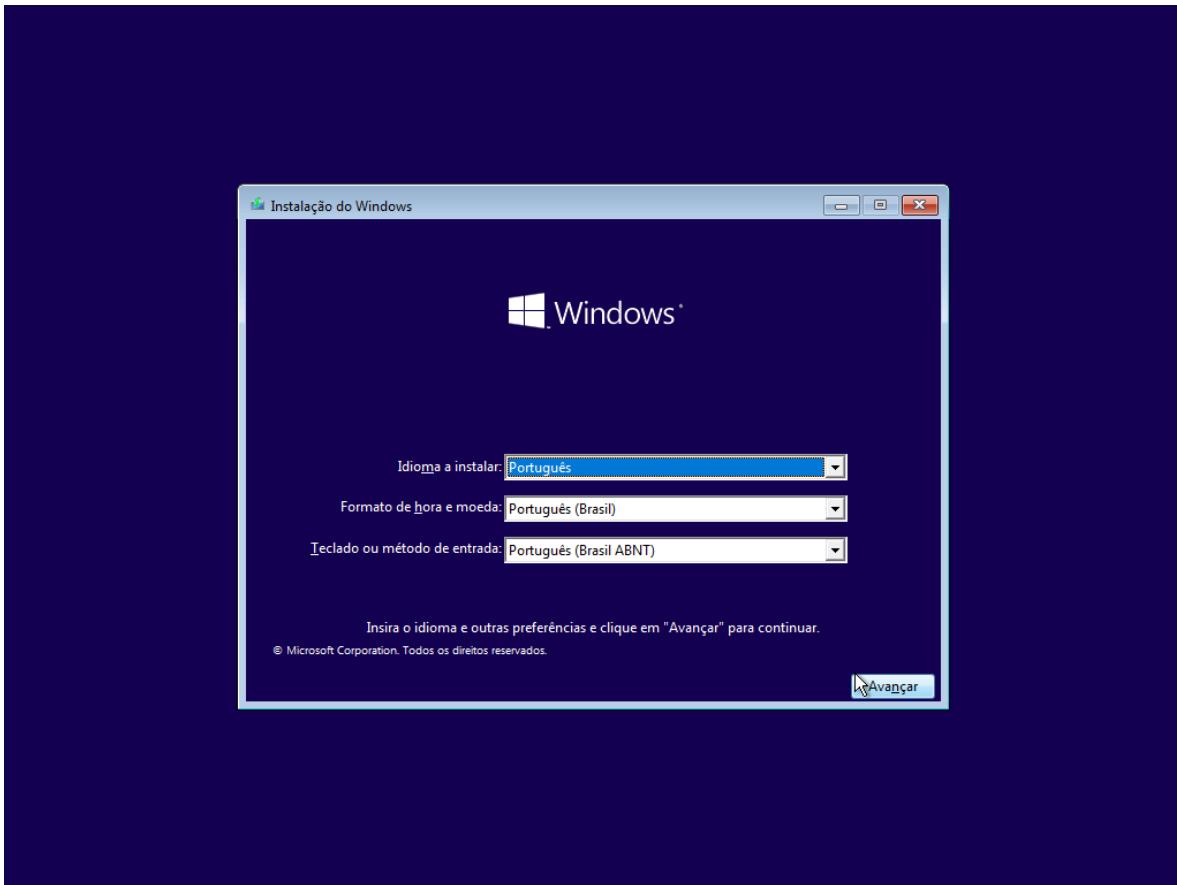


Seleção da ISO do Ubuntu para instalação

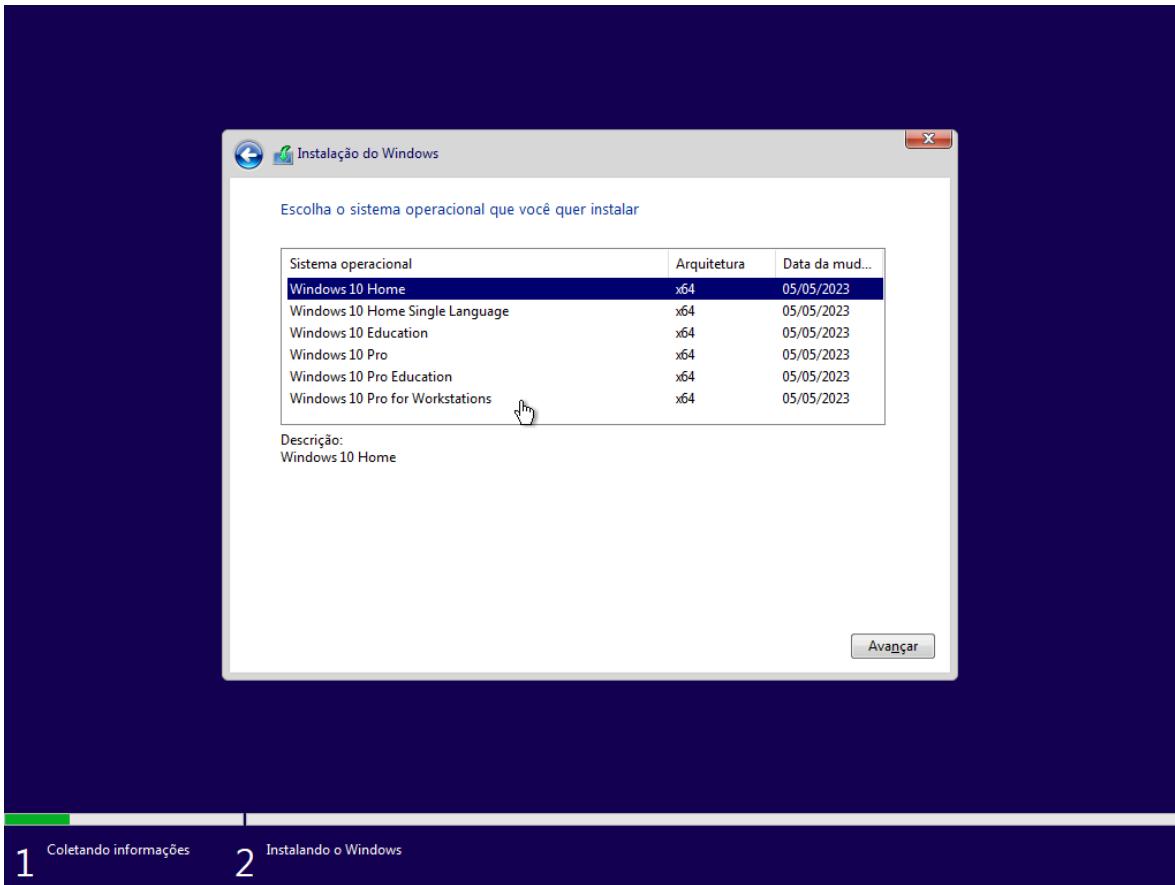
Logar nas VMs

Agora, com tudo o que vimos, podemos fazer a instalação do sistema. Para isso, devemos logar ou entrar nas máquinas que criamos e então fazer as etapas de instalação do Windows e Ubuntu.

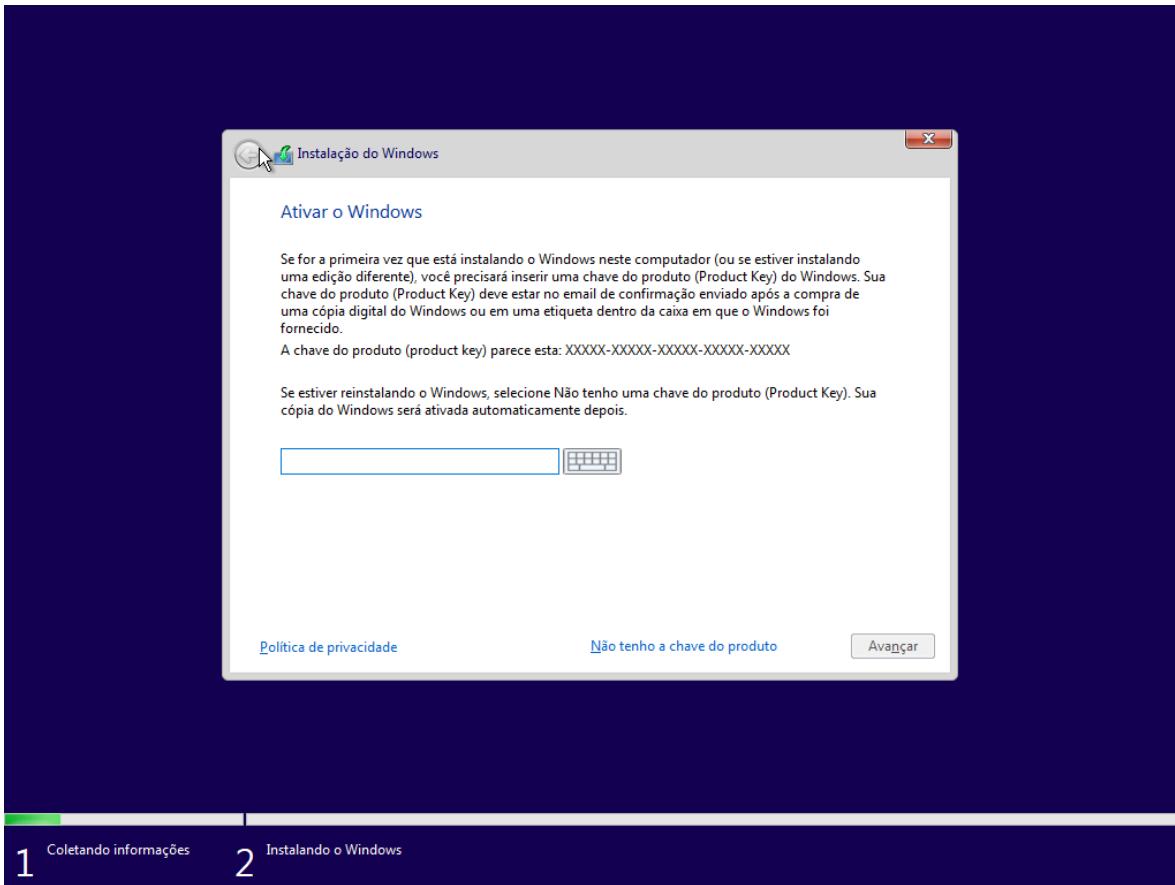
Windows



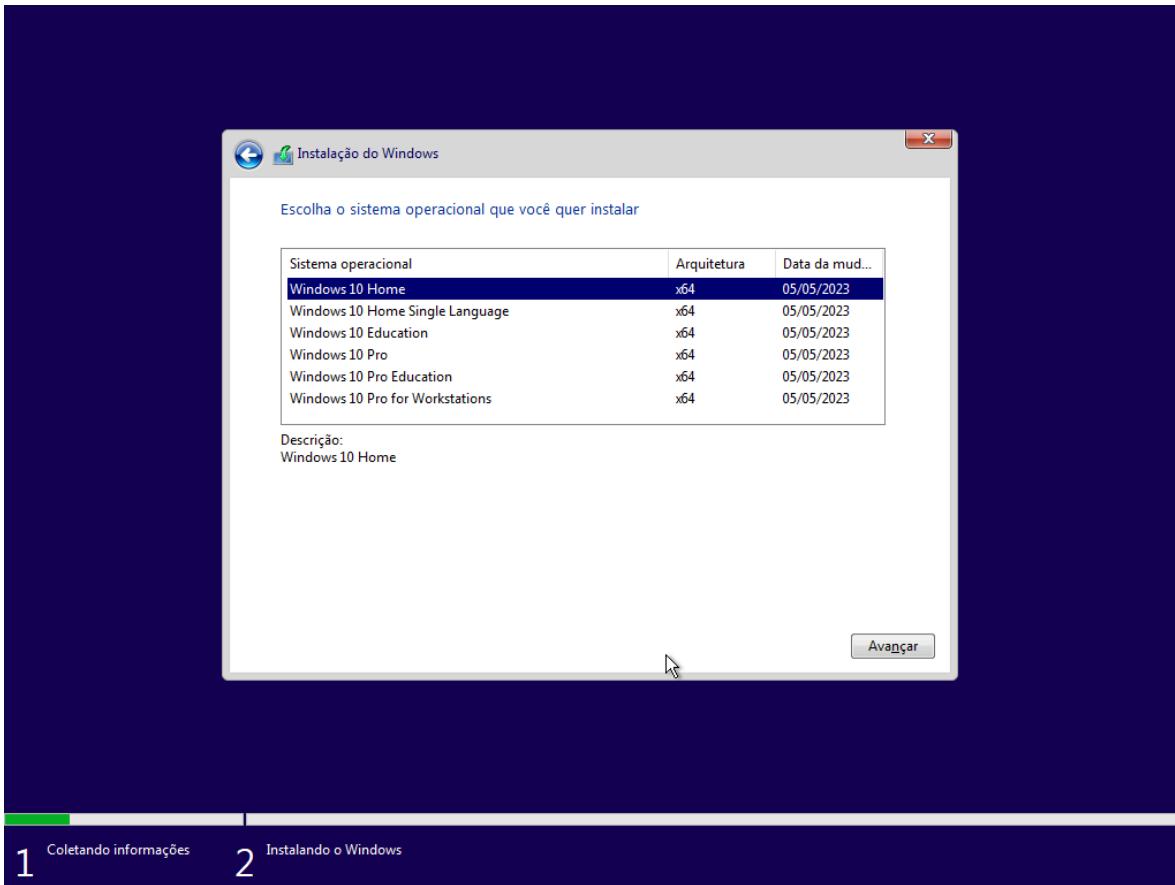
Tela inicial de instalação do Windows



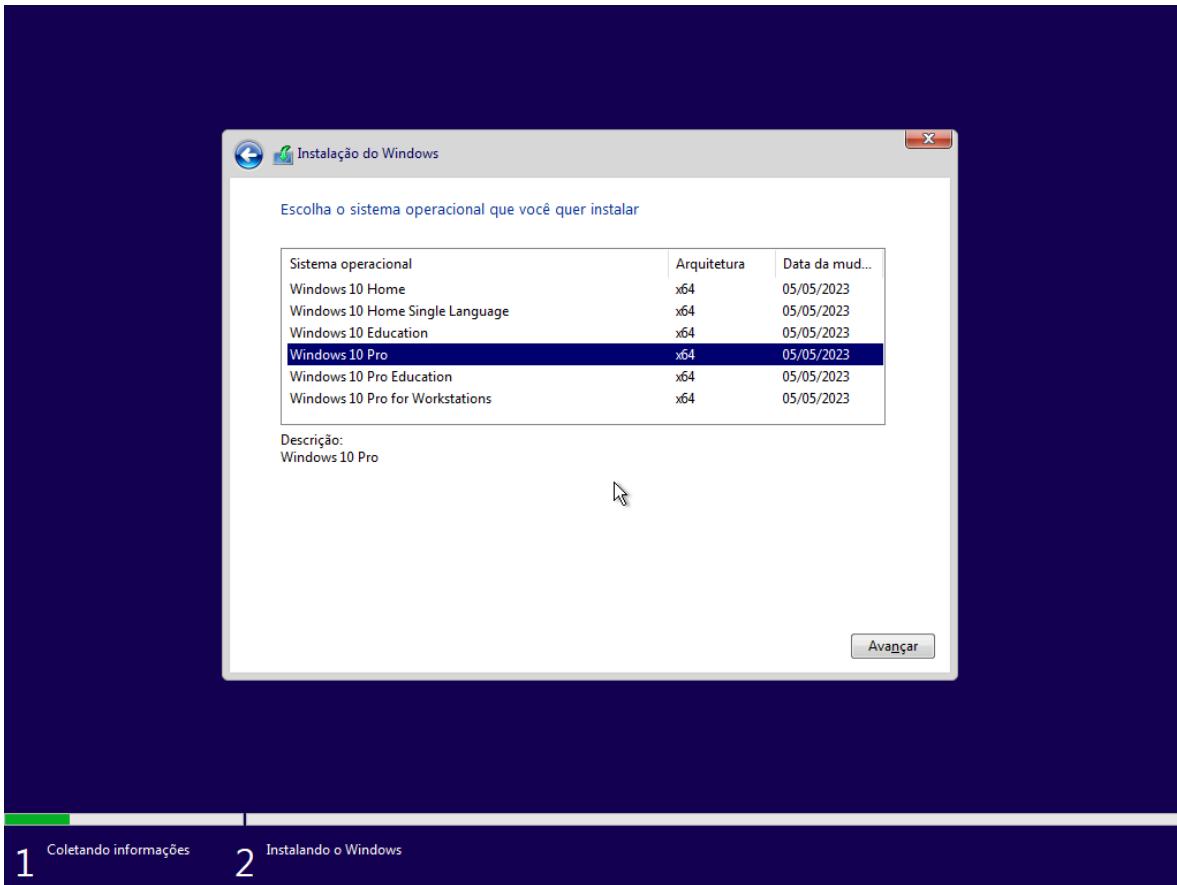
Seleção de idioma, formato de hora e moeda, e layout de teclado



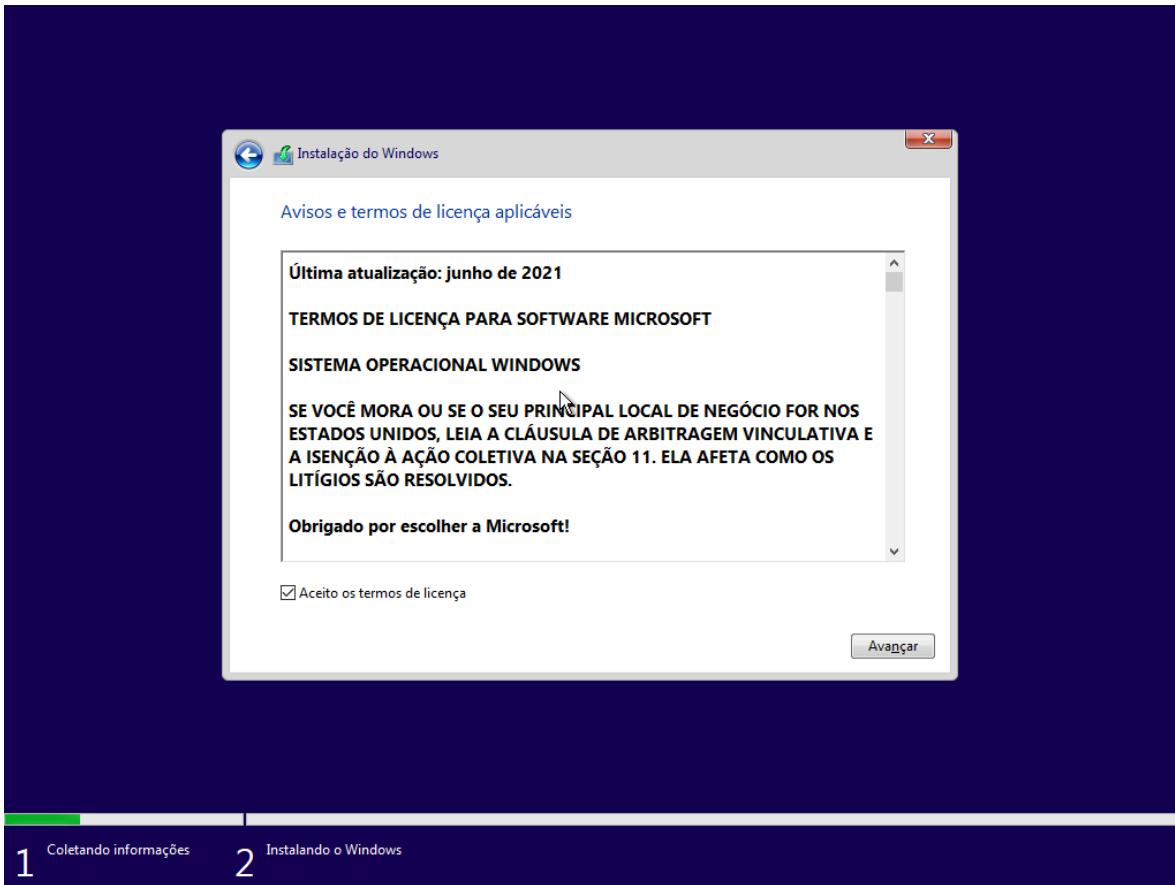
Botão "Instalar agora" para iniciar a instalação do Windows



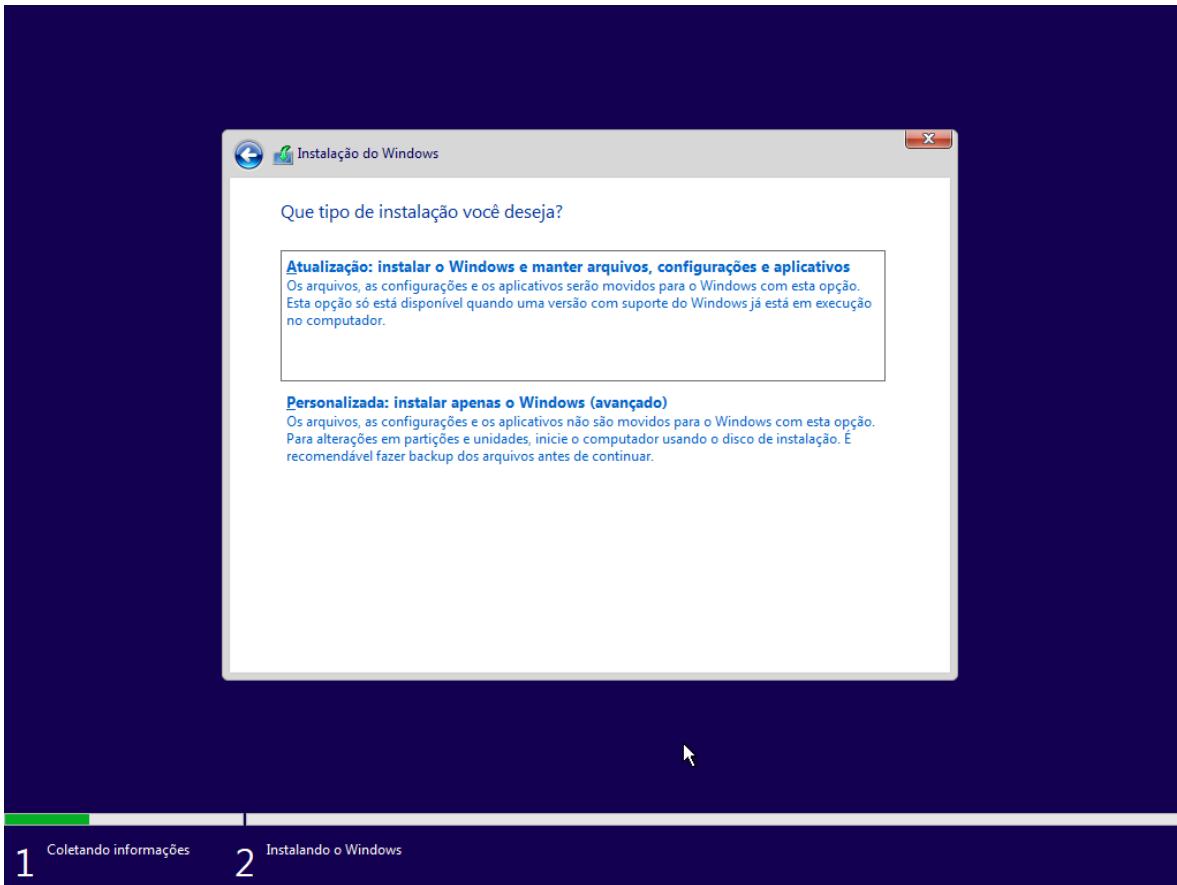
Inserção da chave do produto Windows



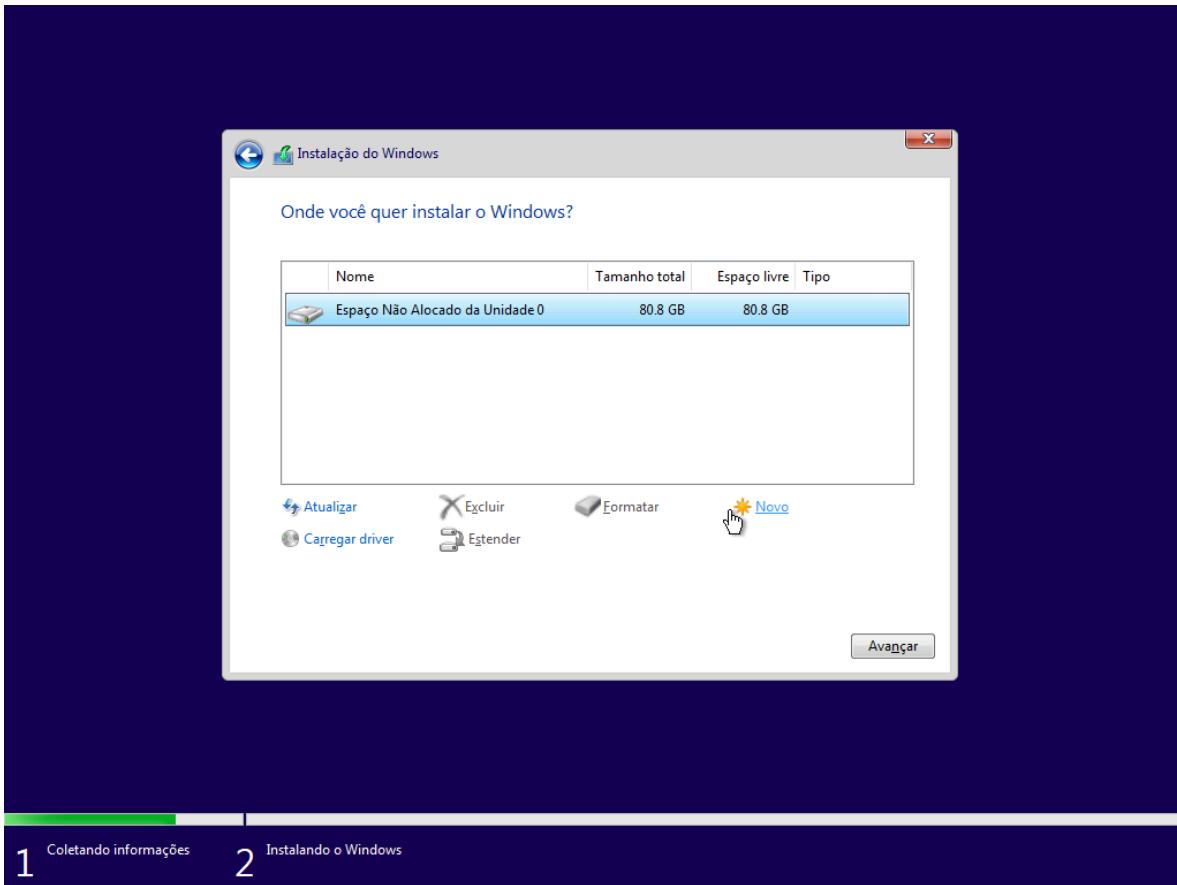
Seleção da versão do Windows a ser instalada



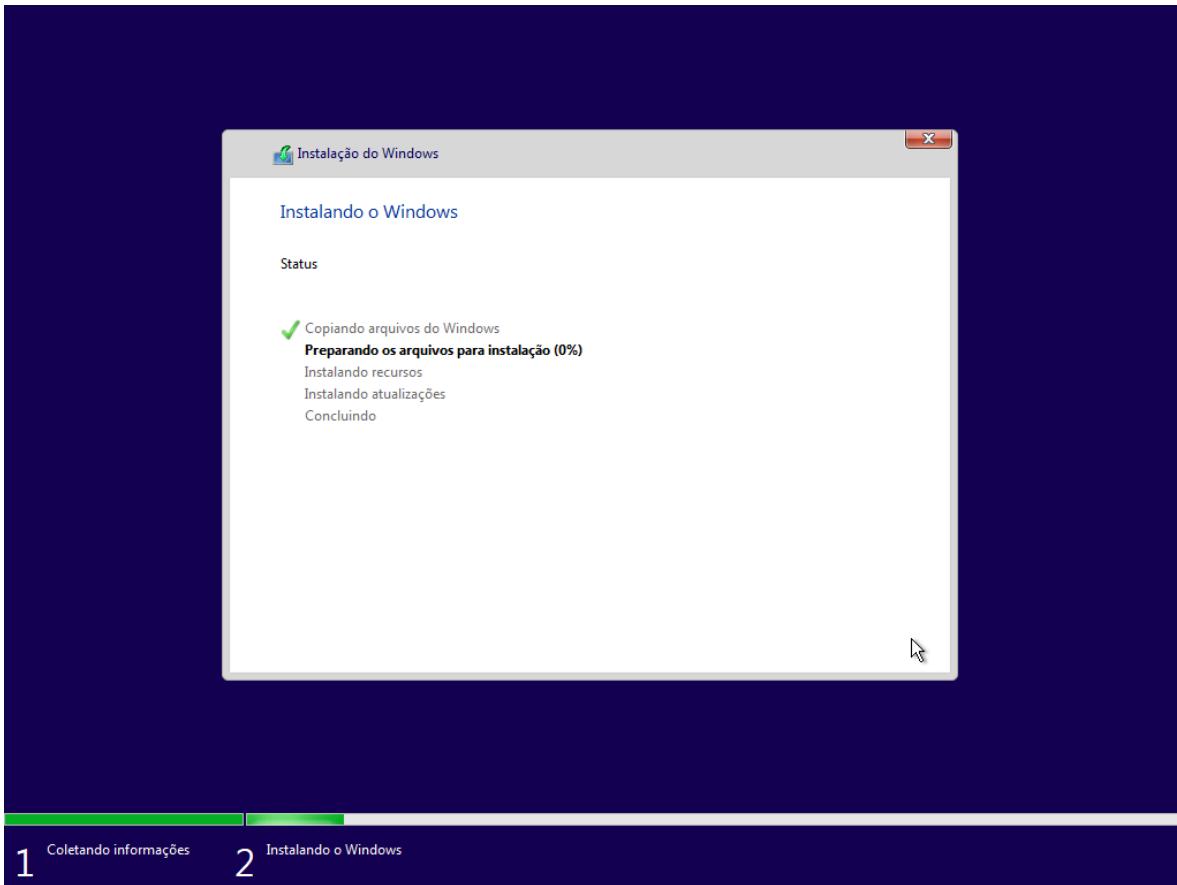
Aceitação dos termos de licença do Windows



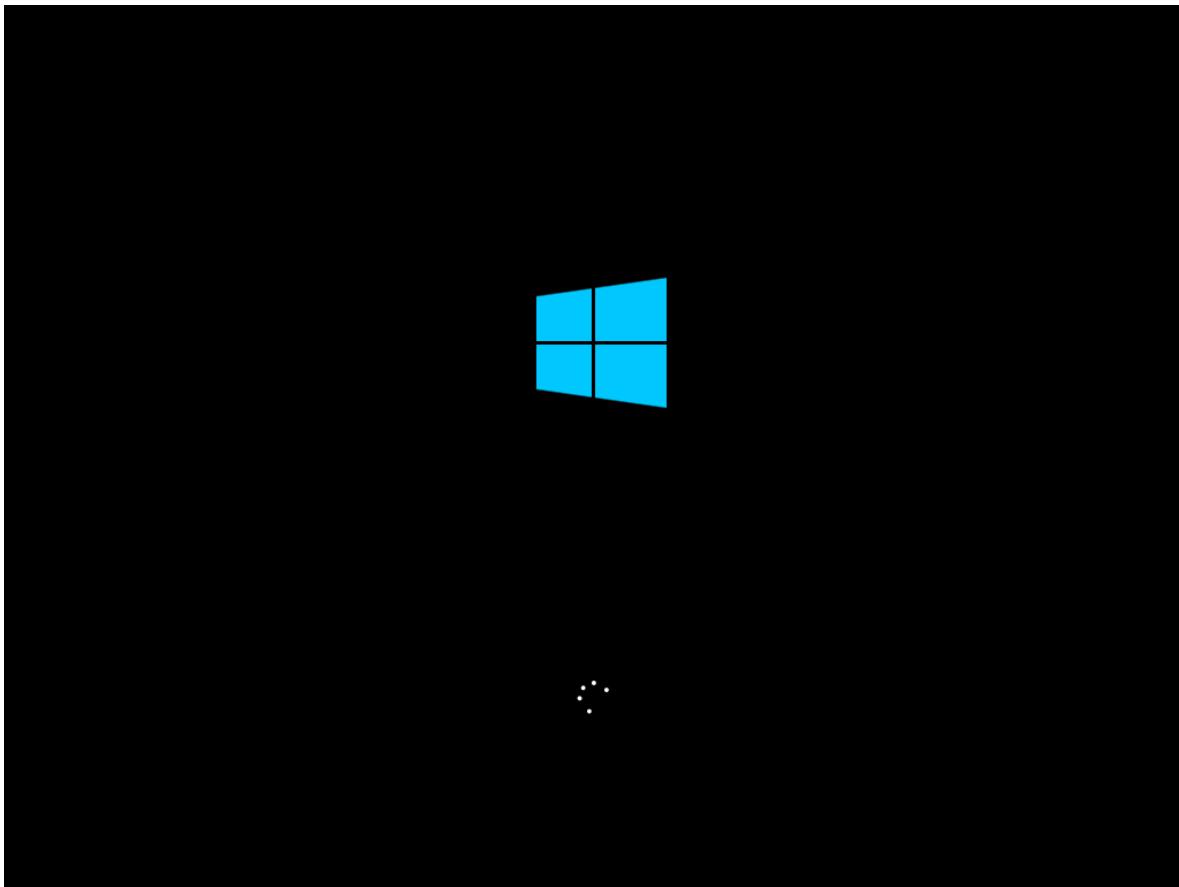
Escolha do tipo de instalação: Atualização ou Personalizada



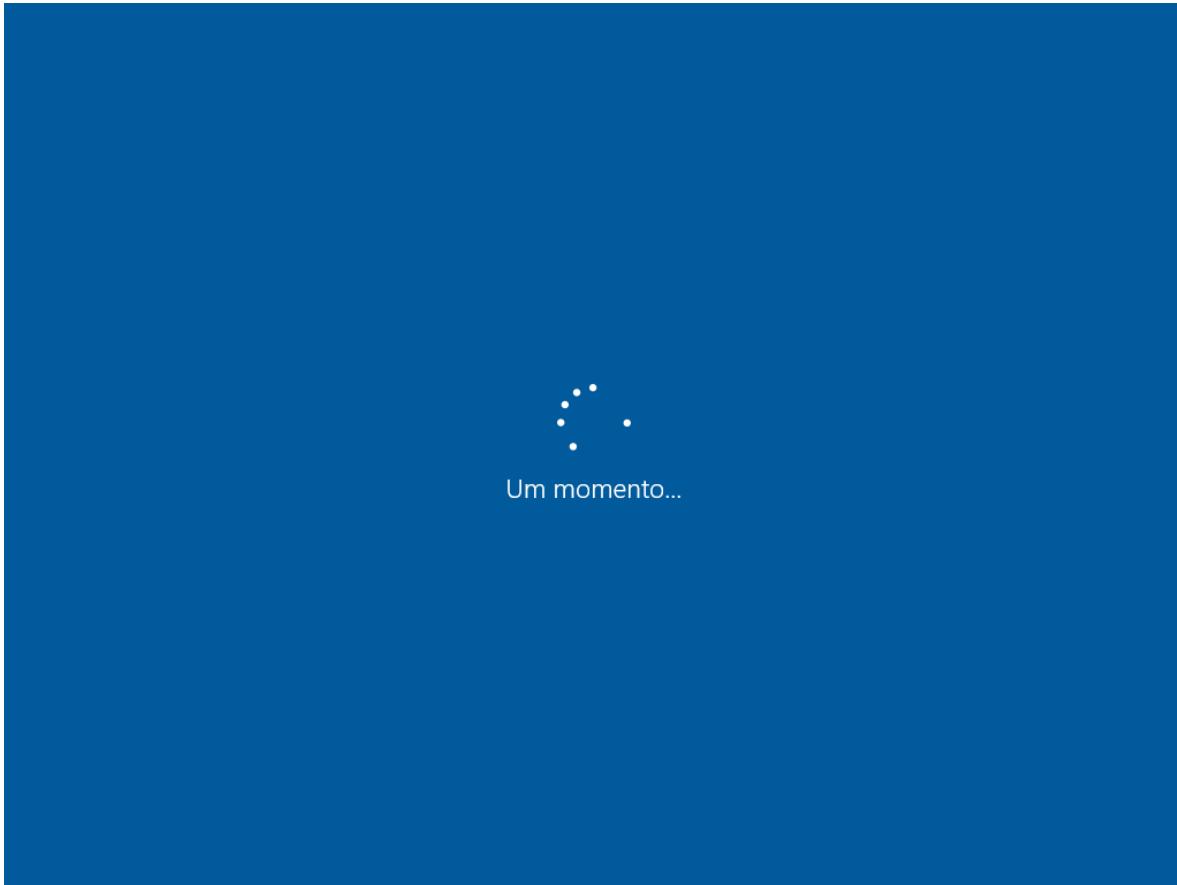
Seleção do disco onde o Windows será instalado



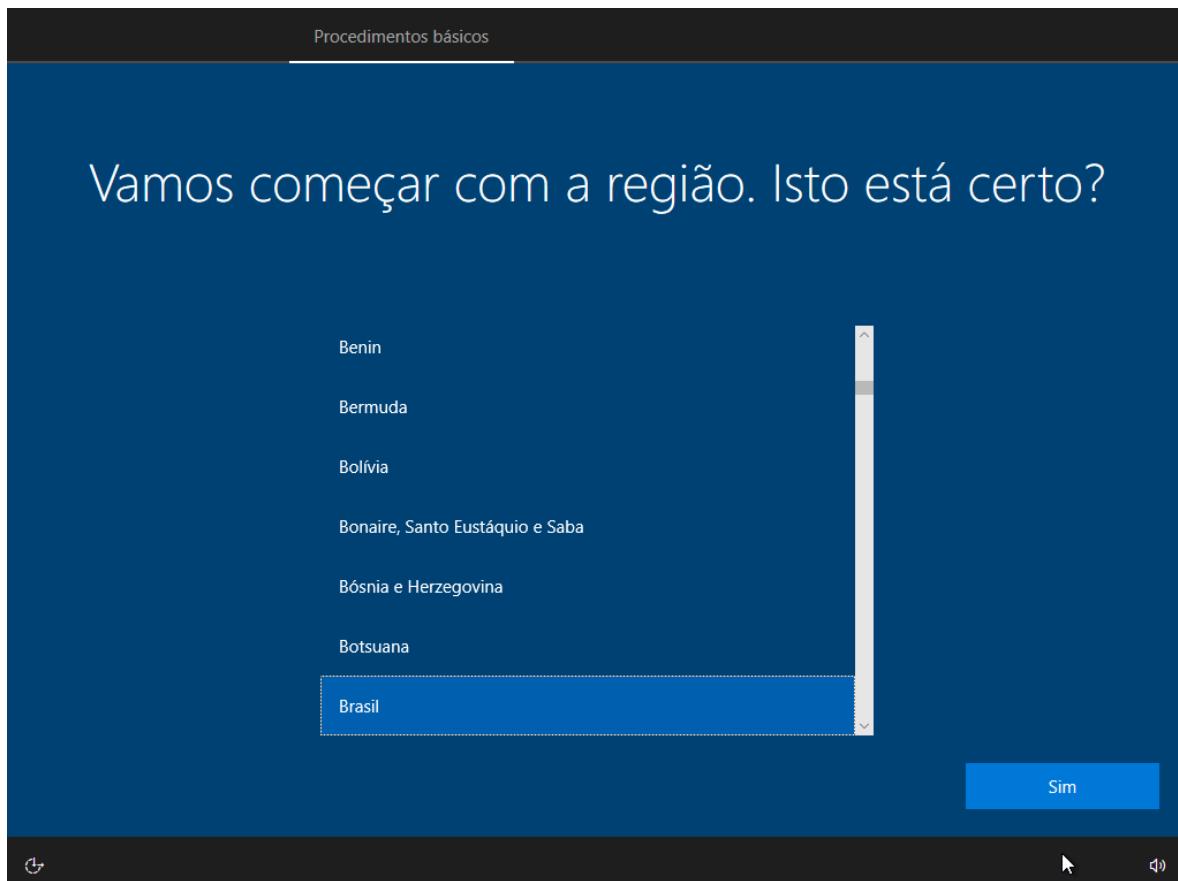
Progresso da instalação do Windows



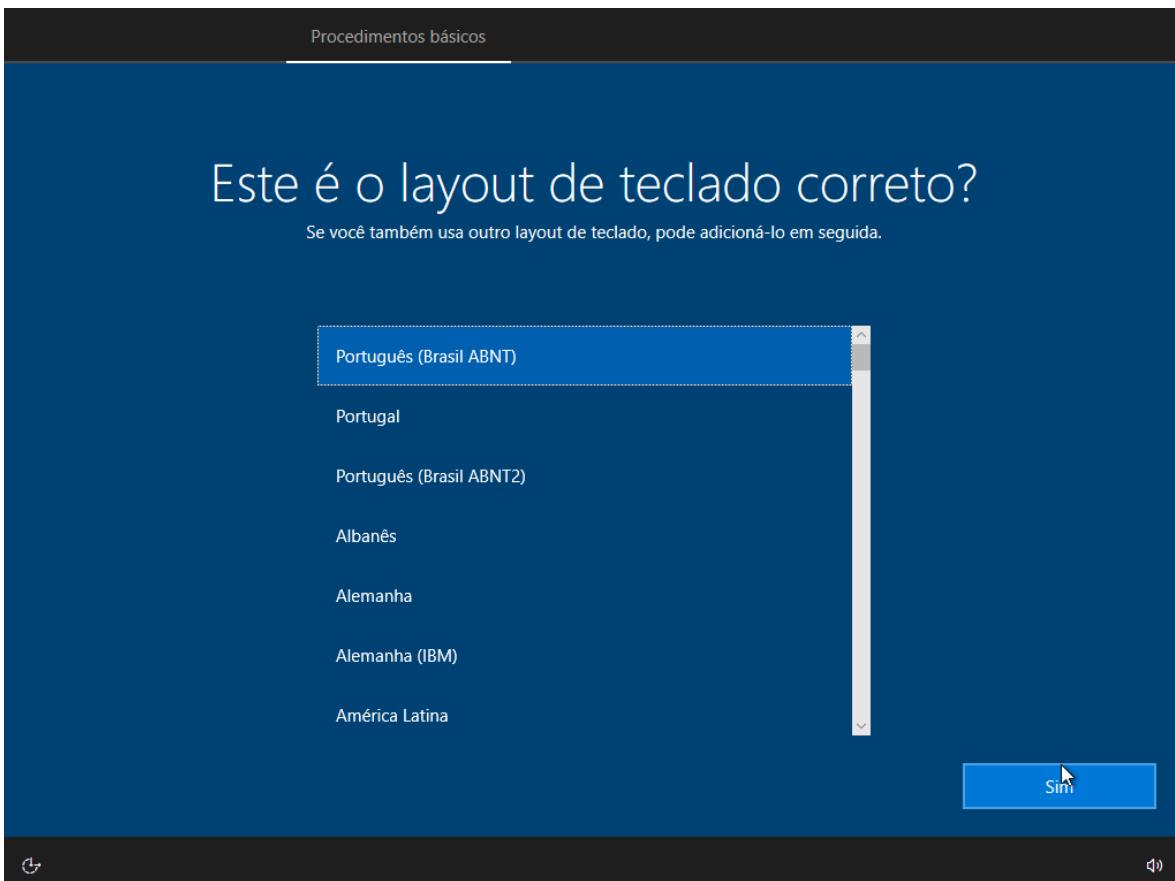
Reinicialização do sistema após a instalação inicial



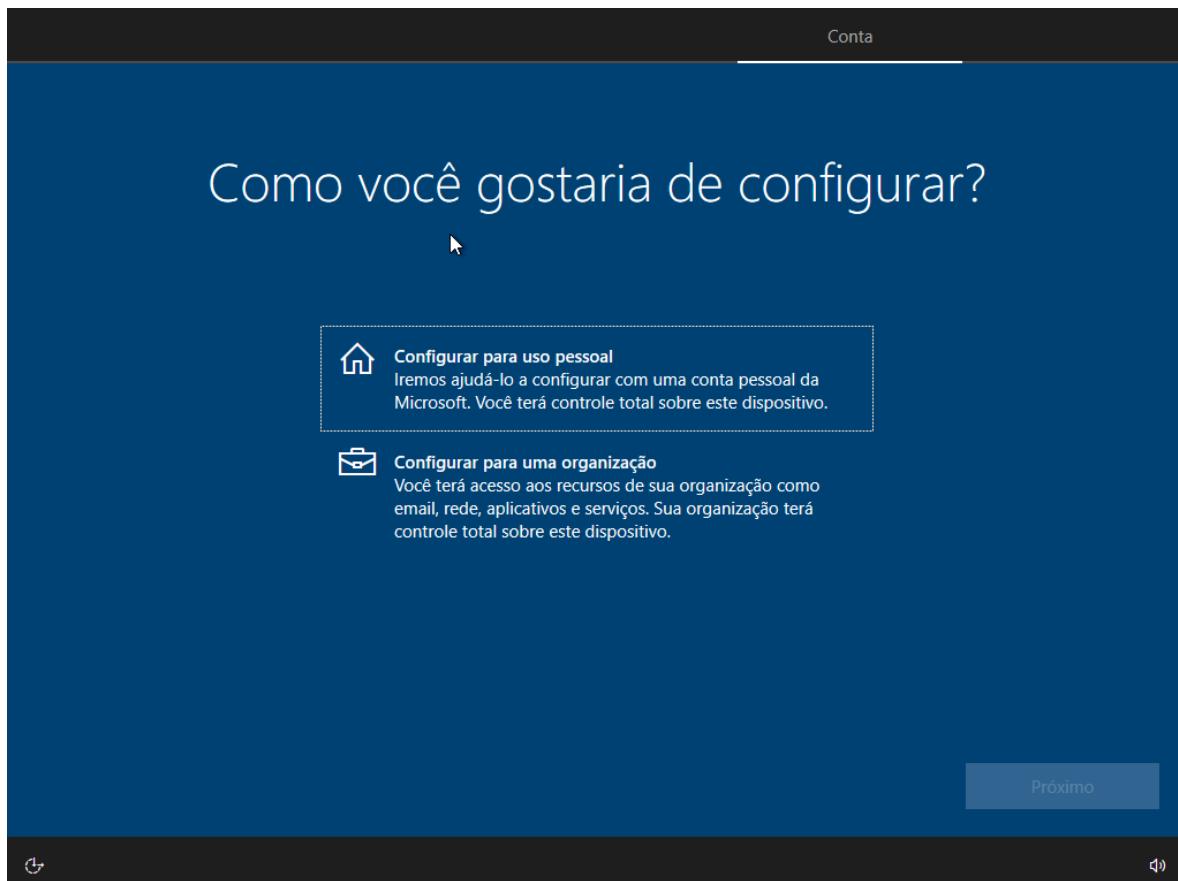
Configuração inicial: Seleção de região



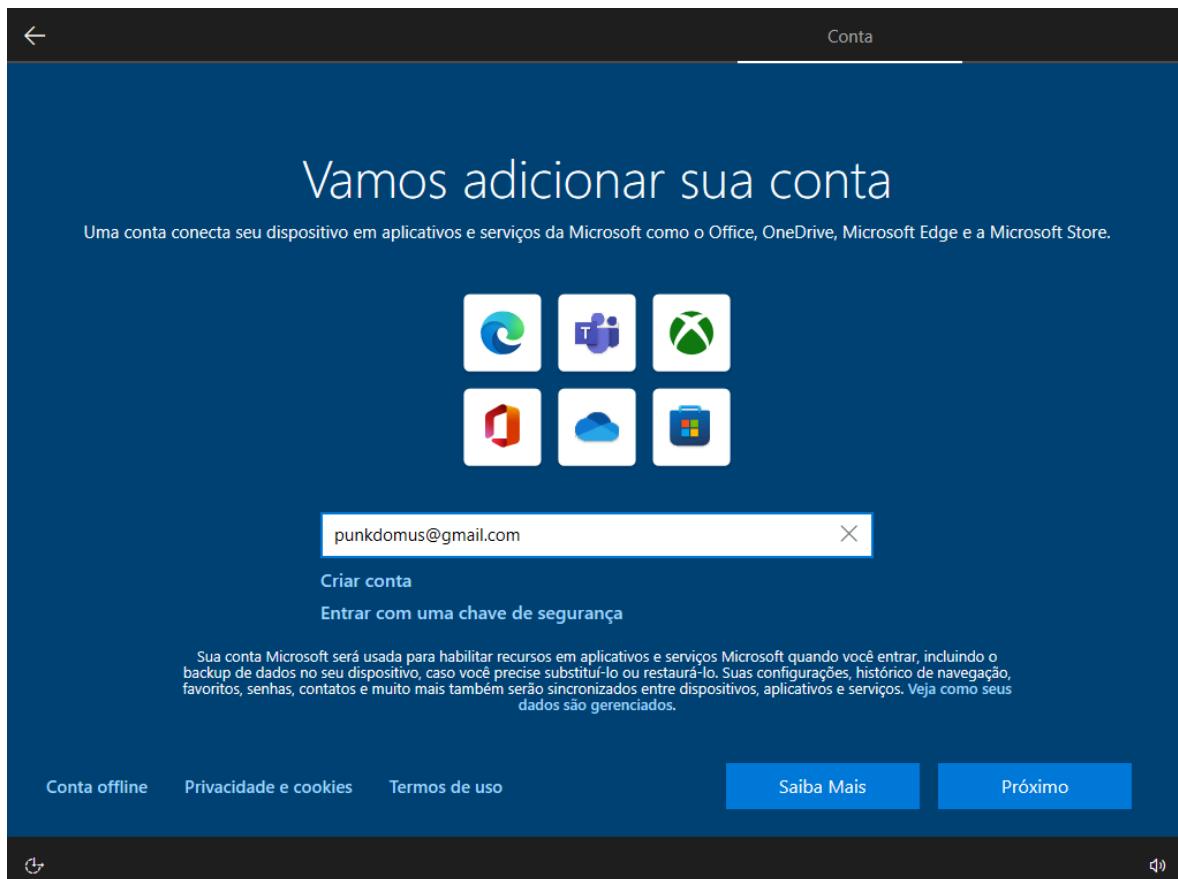
Configuração inicial: Confirmação do layout de teclado



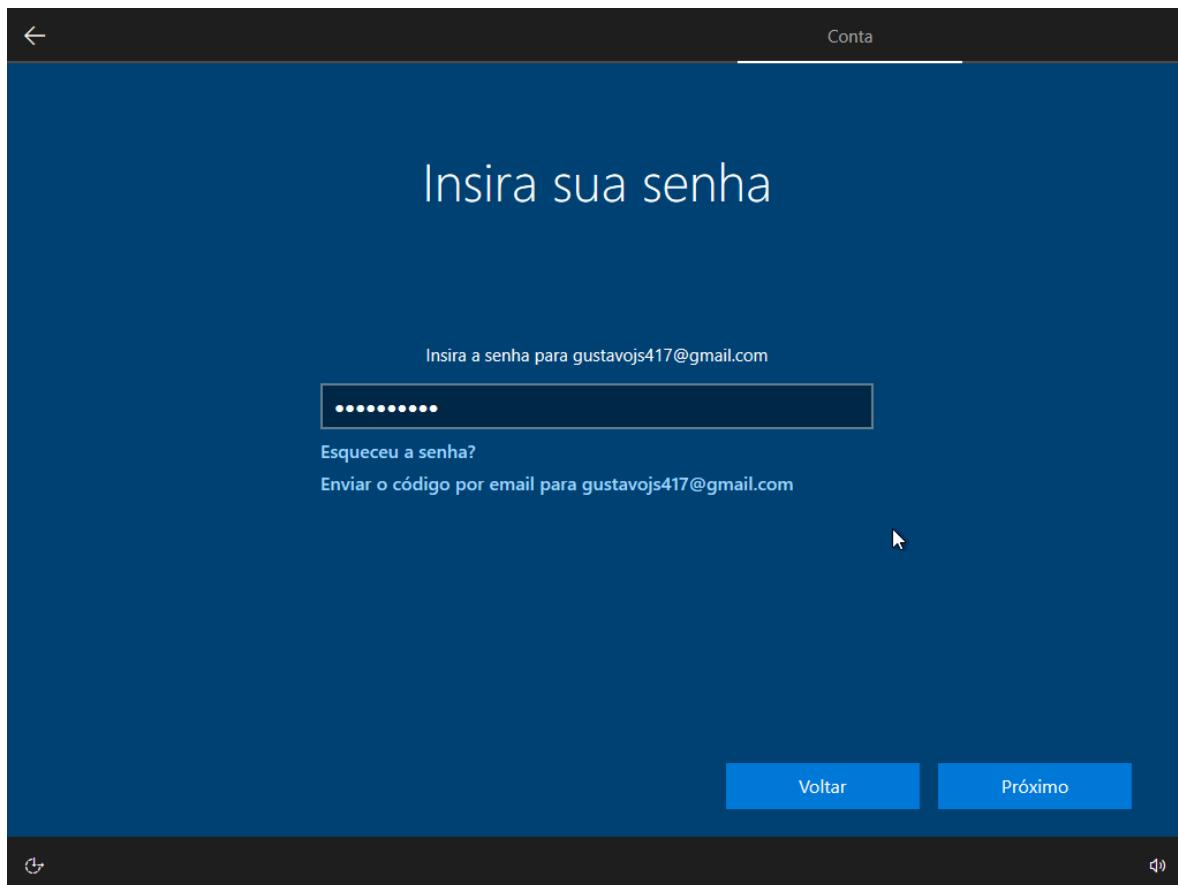
Configuração inicial: Opção de adicionar um segundo layout de teclado



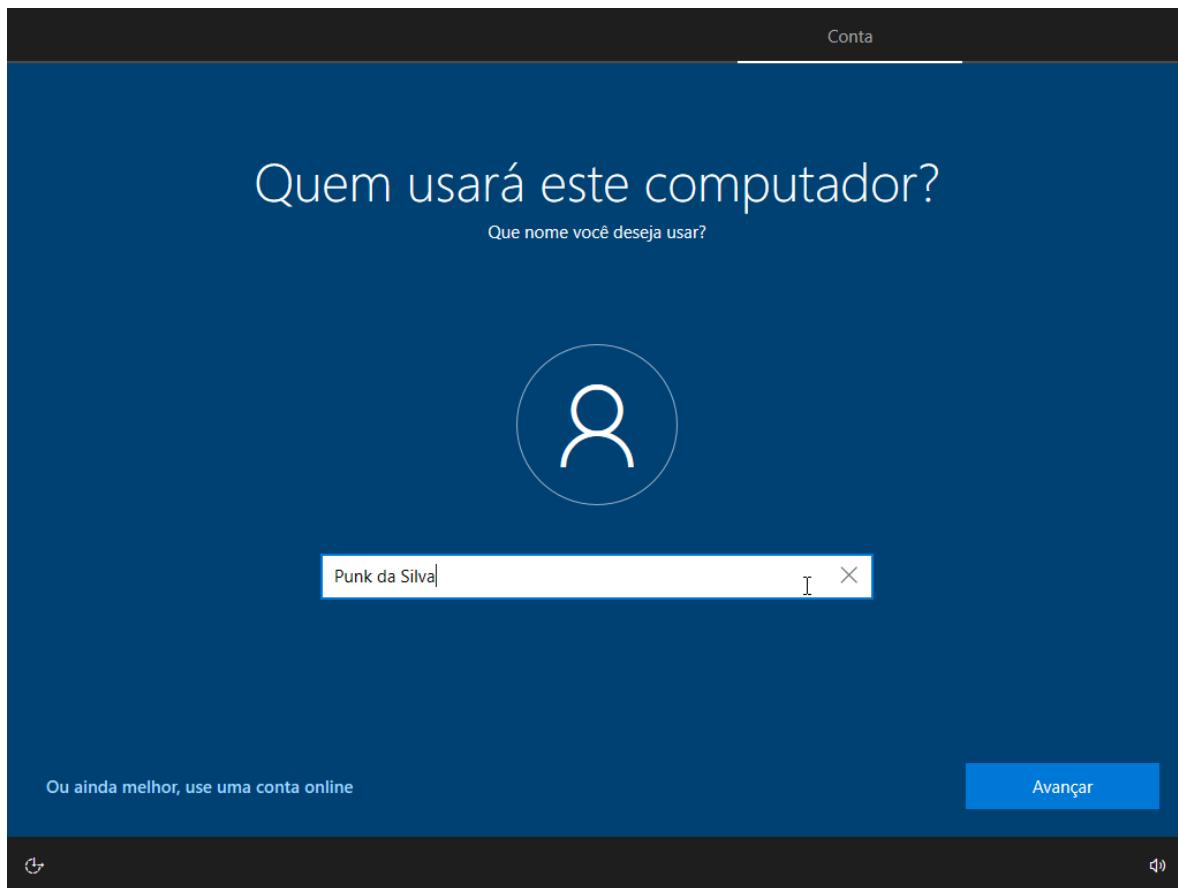
Configuração de rede: Conexão à internet



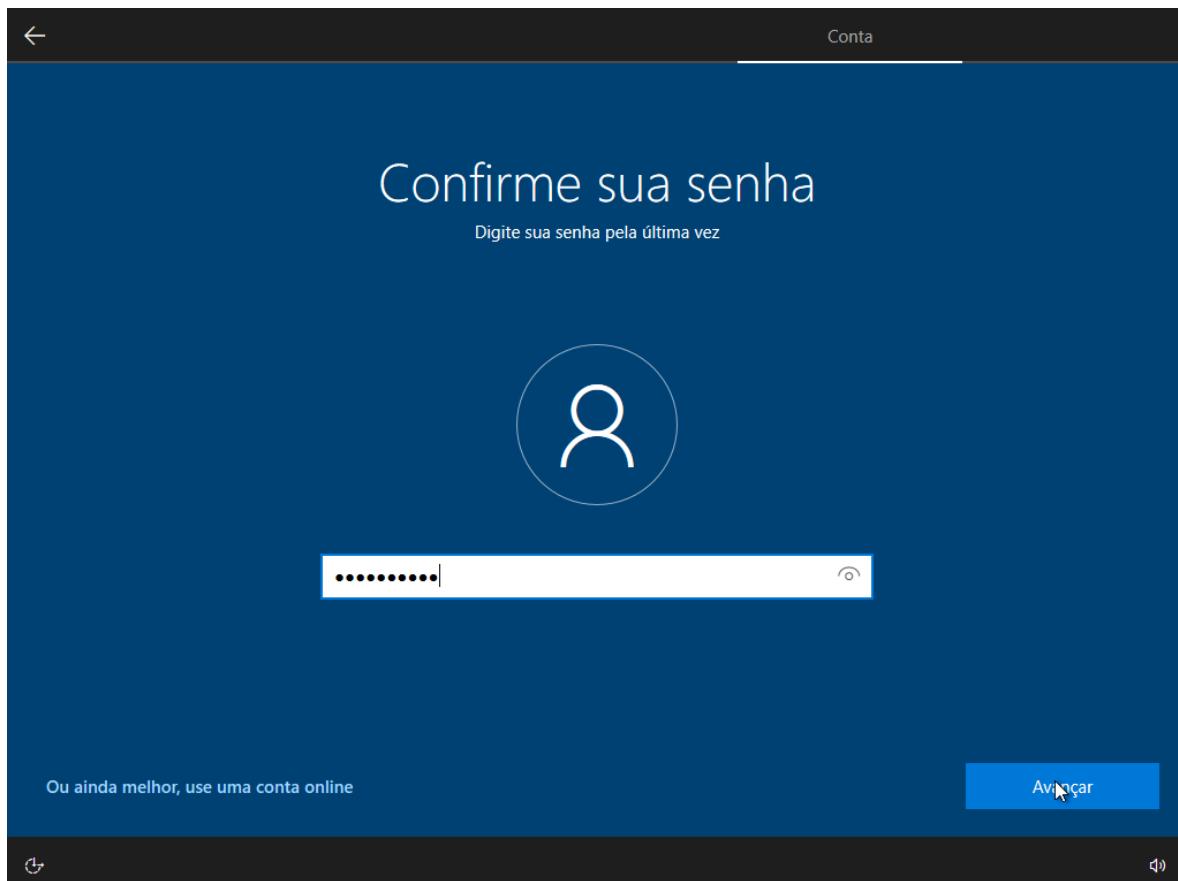
Configuração de conta: Opções de login



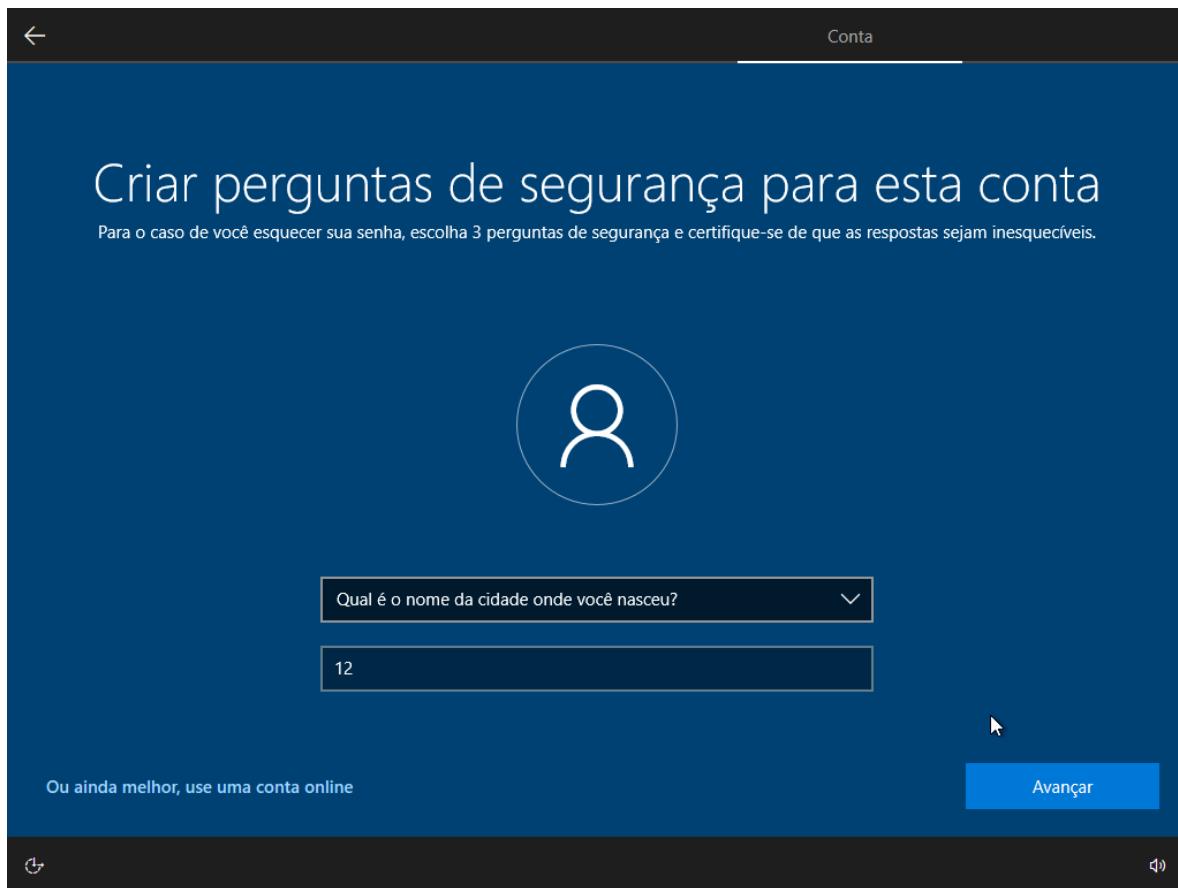
Definição de senha para a conta local



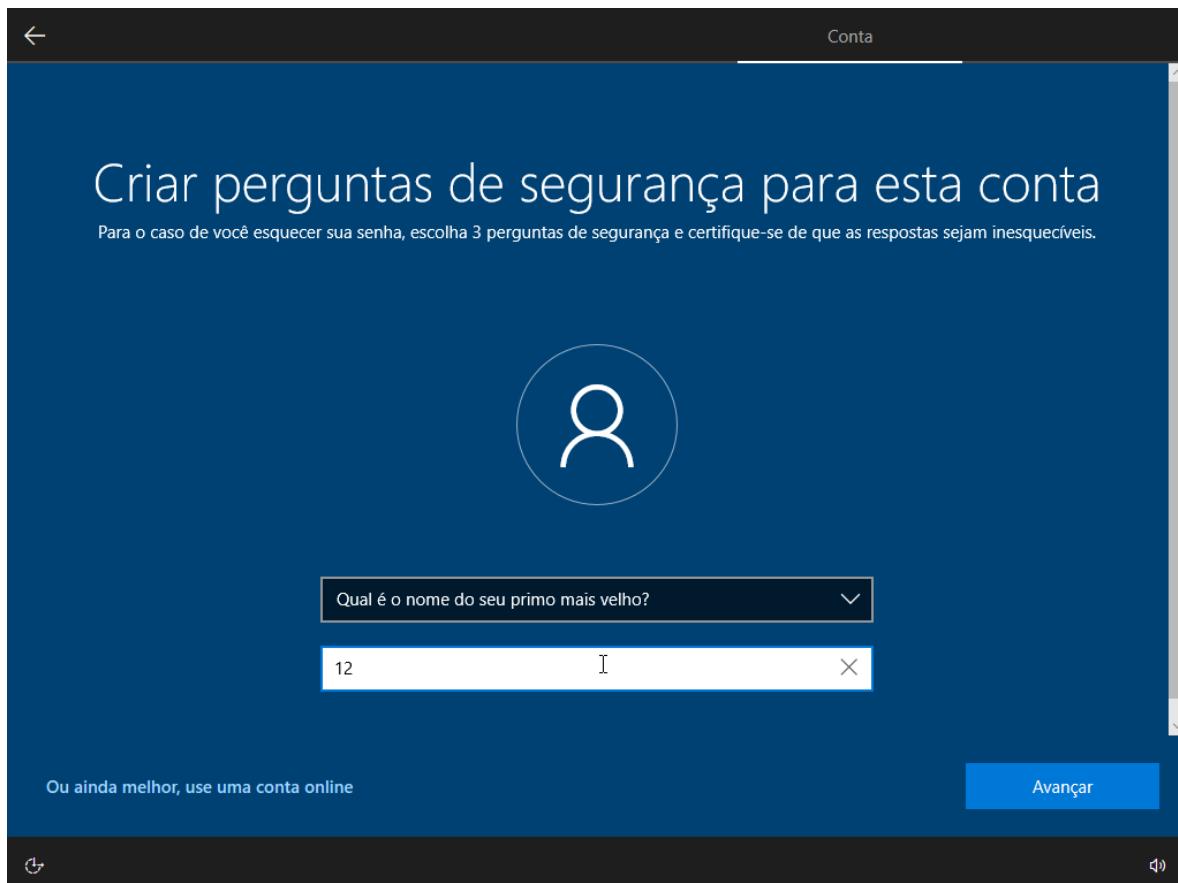
Configuração de perguntas de segurança



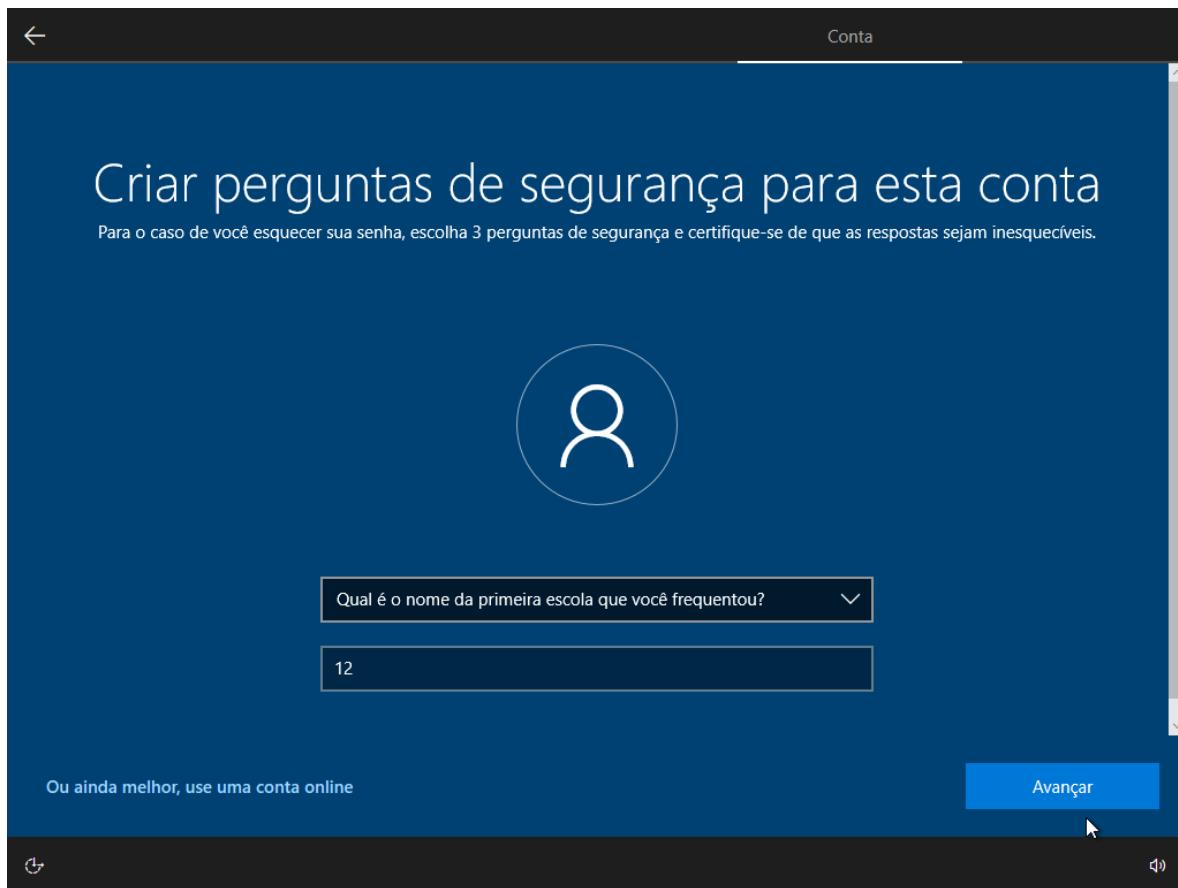
Configurações de privacidade: Escolha de permissões



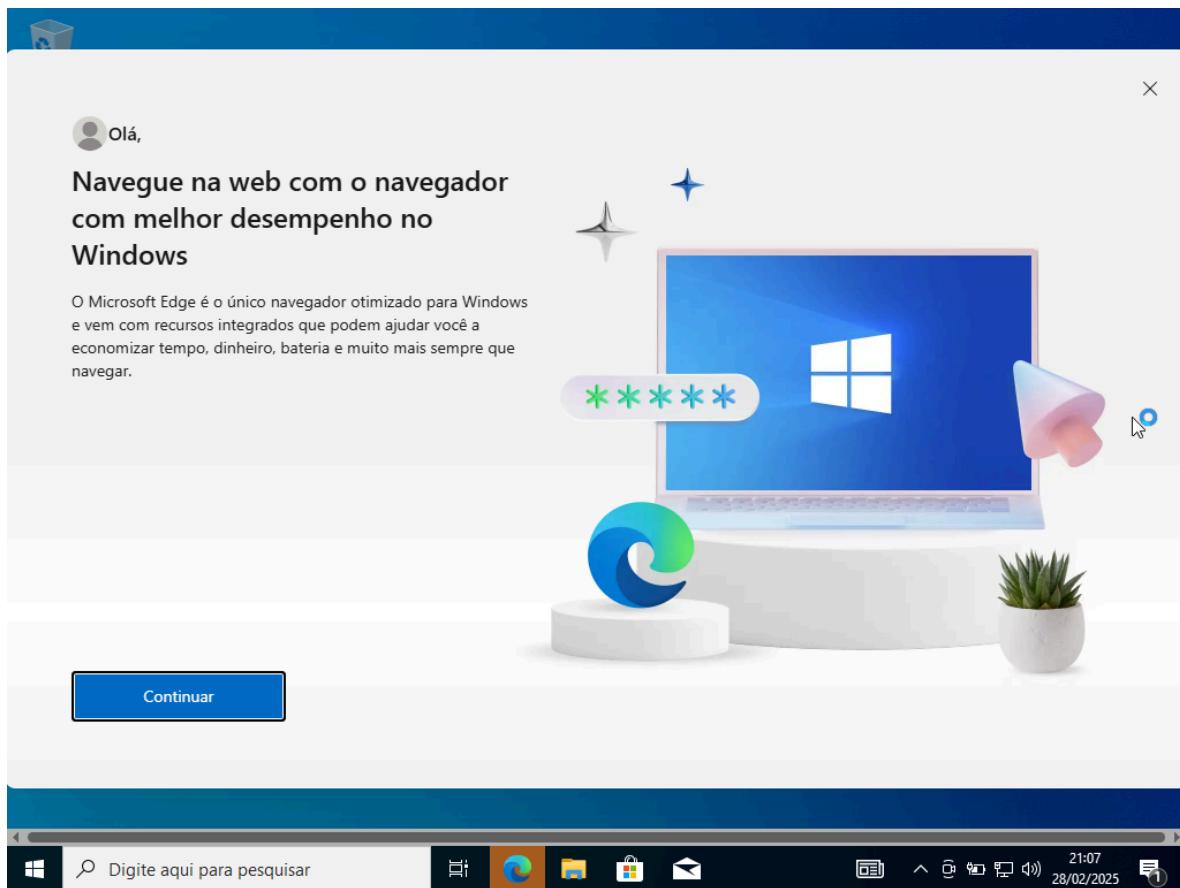
Configuração de experiência personalizada



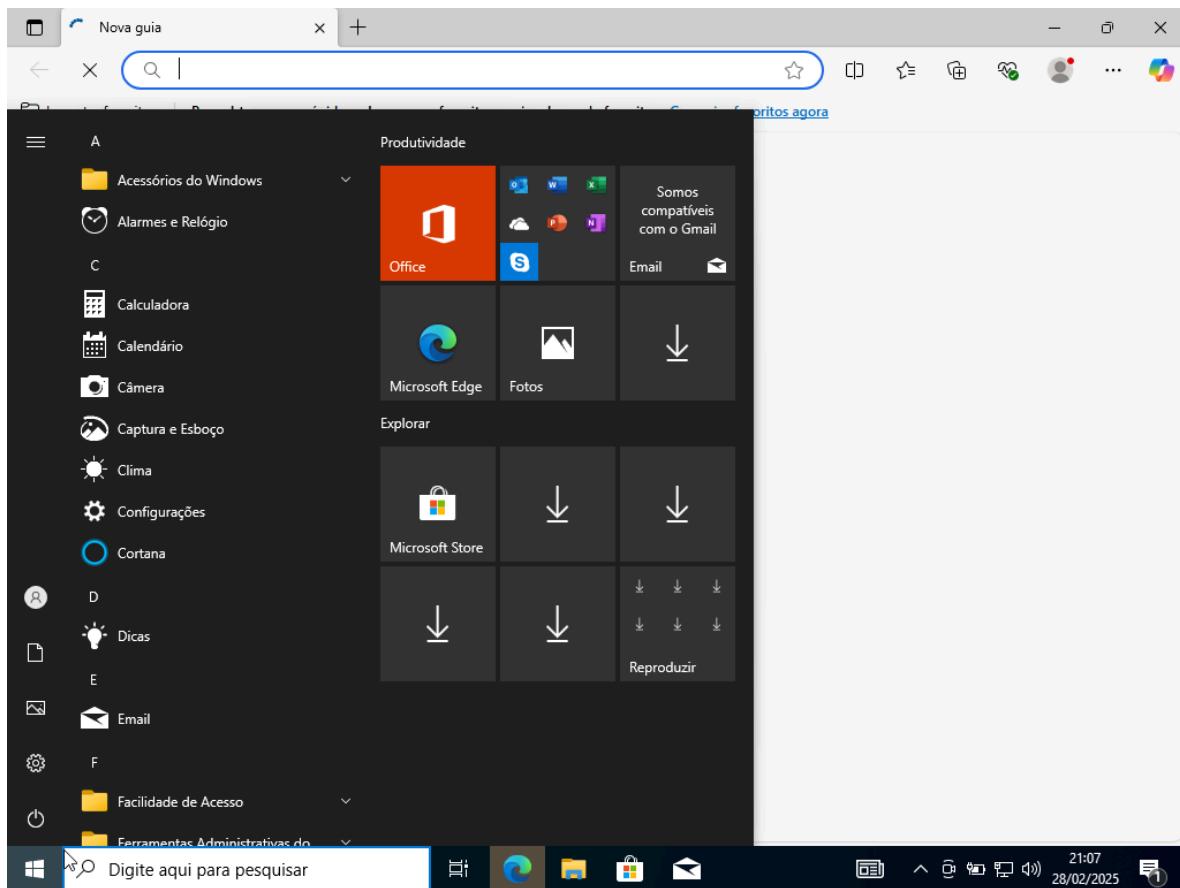
Configuração do assistente digital Cortana



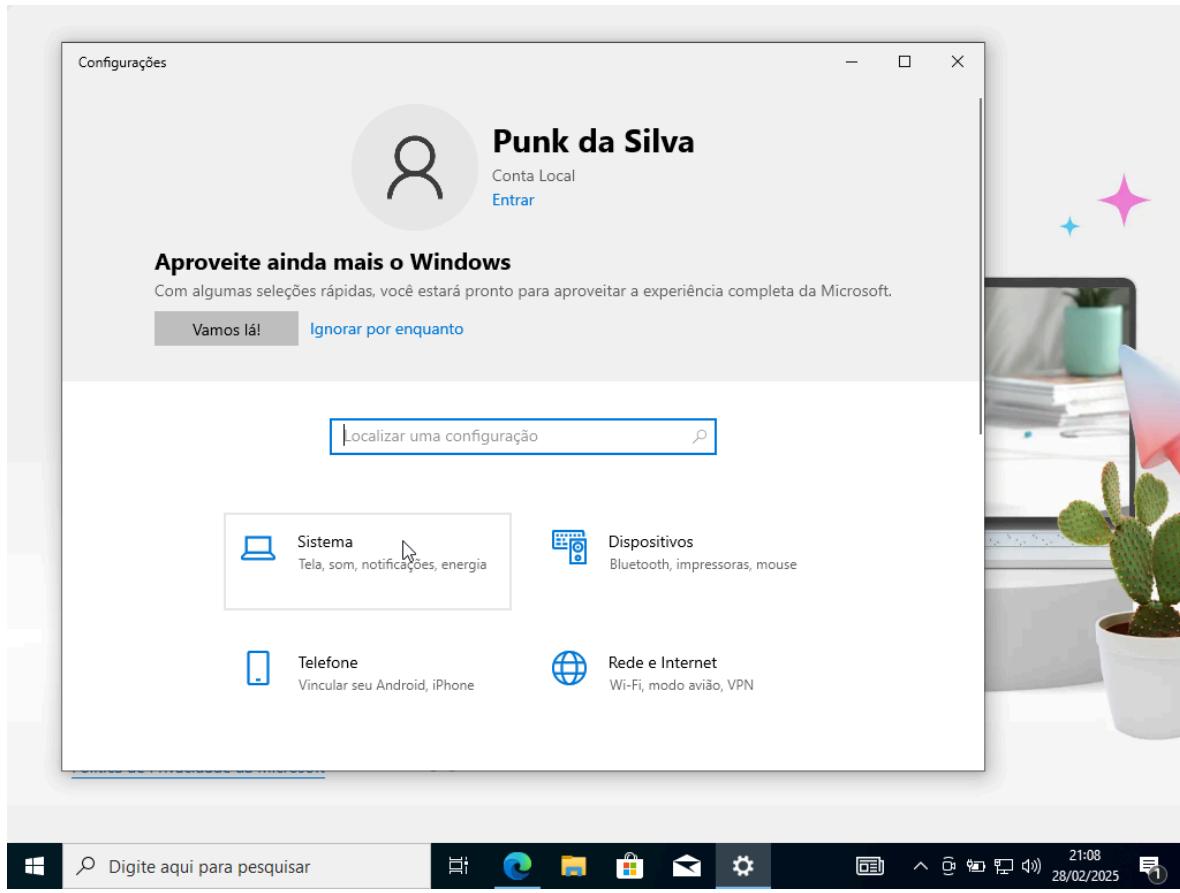
Finalização da configuração do Windows



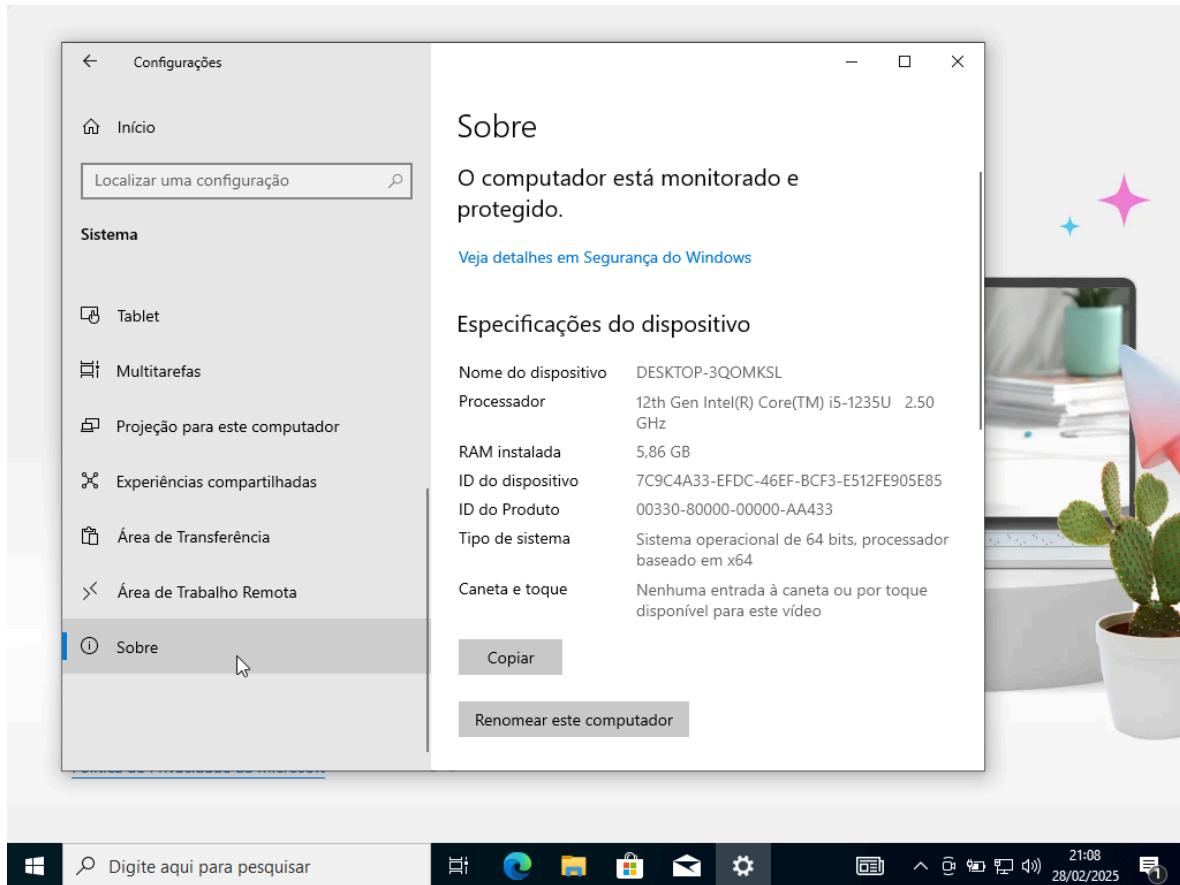
Área de trabalho do Windows após a instalação completa



Menu Iniciar do Windows recém-instalado e digite "Configurações"



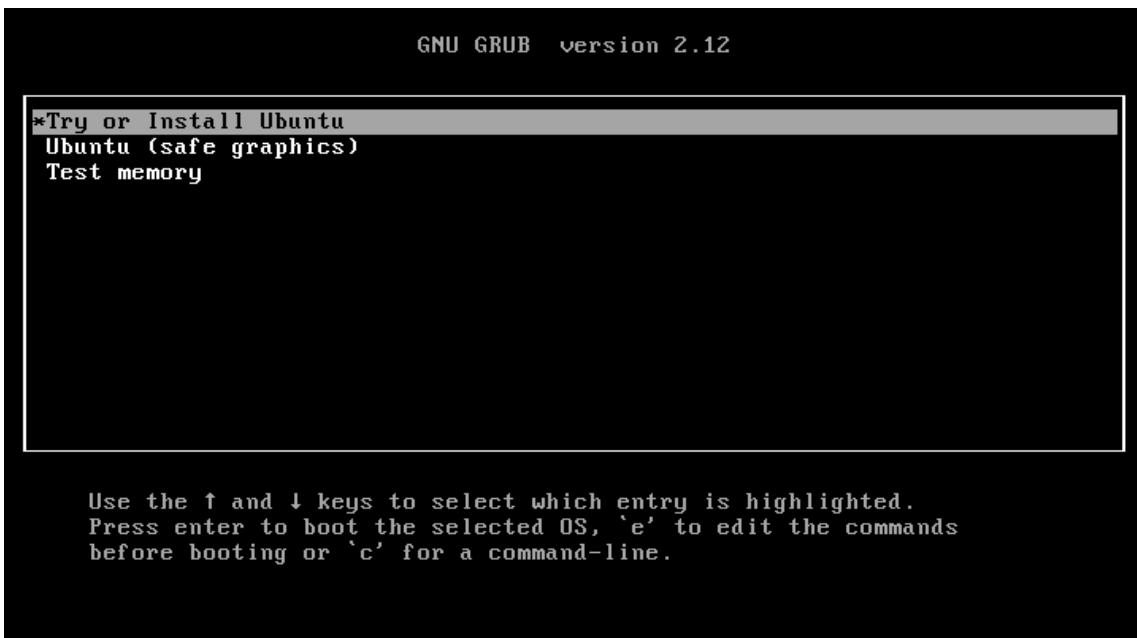
Vá em "Sistema"



Na seção "Sobre" do sistema podemos ver as configurações da máquina que foi instalada

Ubuntu

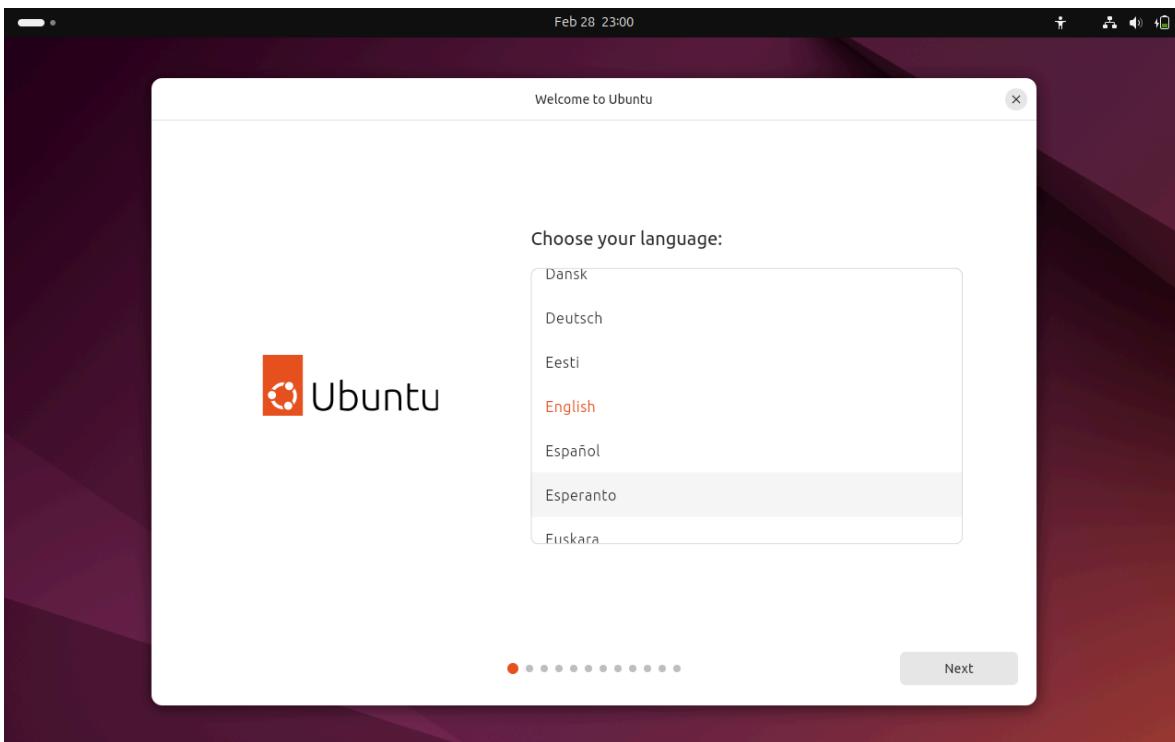
- Configurações de instalação:



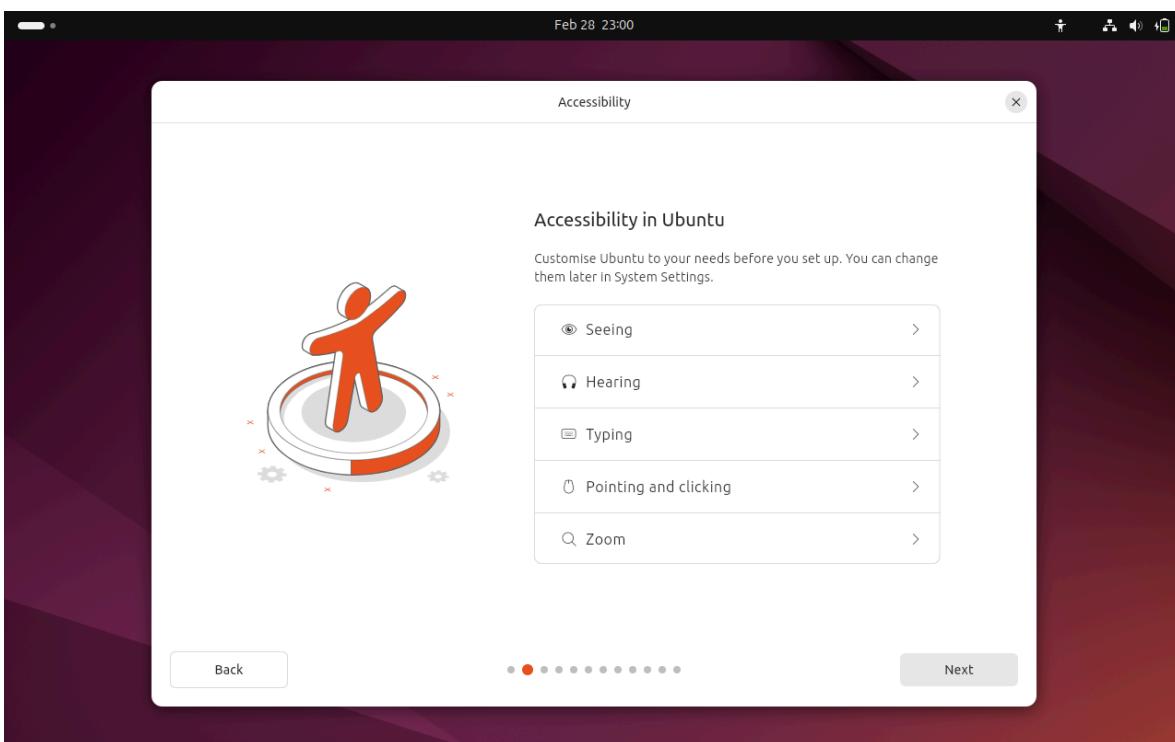
Assim que rodarmos a máquina, vai aparecer a tela do GRUB e então selecionamos a primeira opção

- i** O GRUB (Grand Unified Bootloader) é o menu de inicialização que aparece quando você inicia o sistema Ubuntu. Ele permite que você escolha entre diferentes opções de inicialização.
- A primeira opção geralmente é "Ubuntu", que inicia o sistema normalmente.
 - Outras opções podem incluir modos de recuperação ou versões anteriores do kernel.

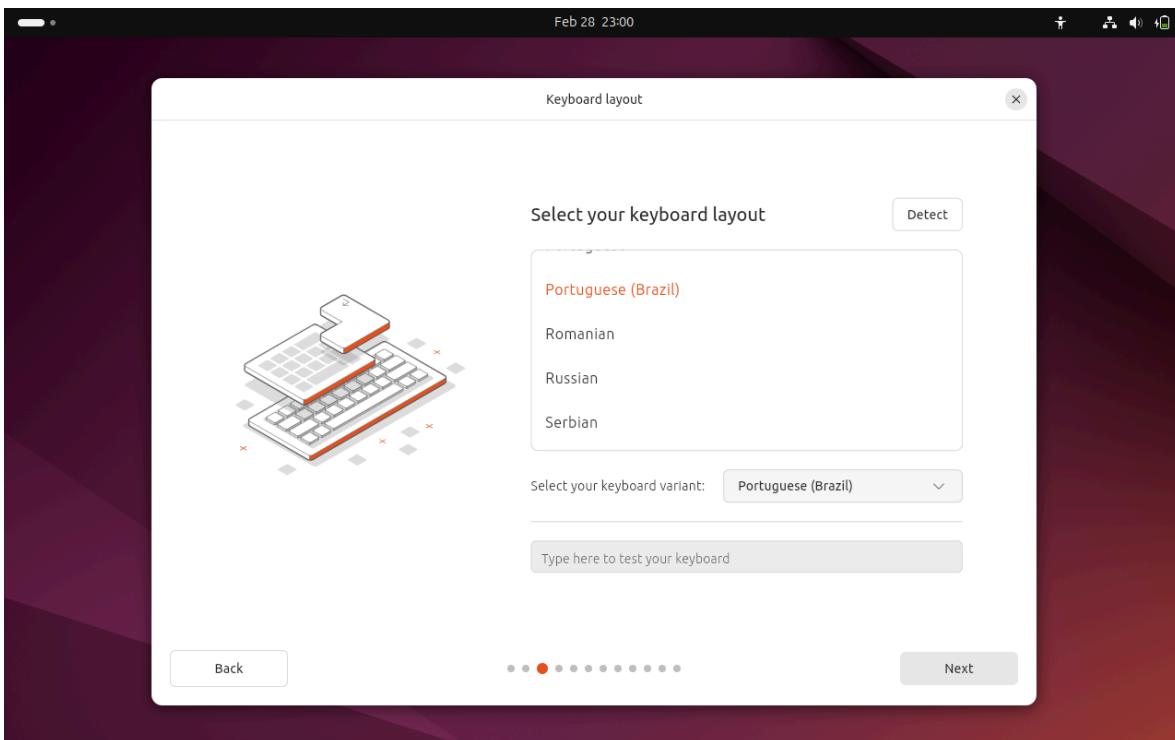
Para a instalação padrão, selecione a primeira opção "Ubuntu" e pressione Enter para continuar.



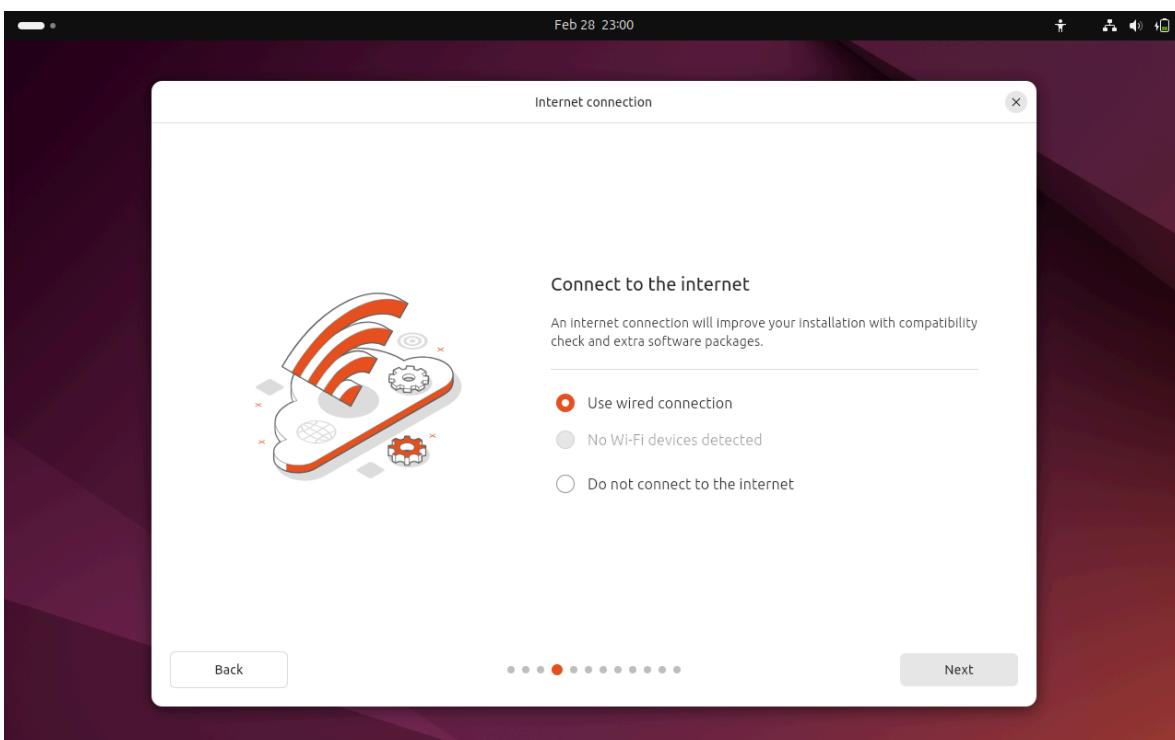
Selecionamos o idioma



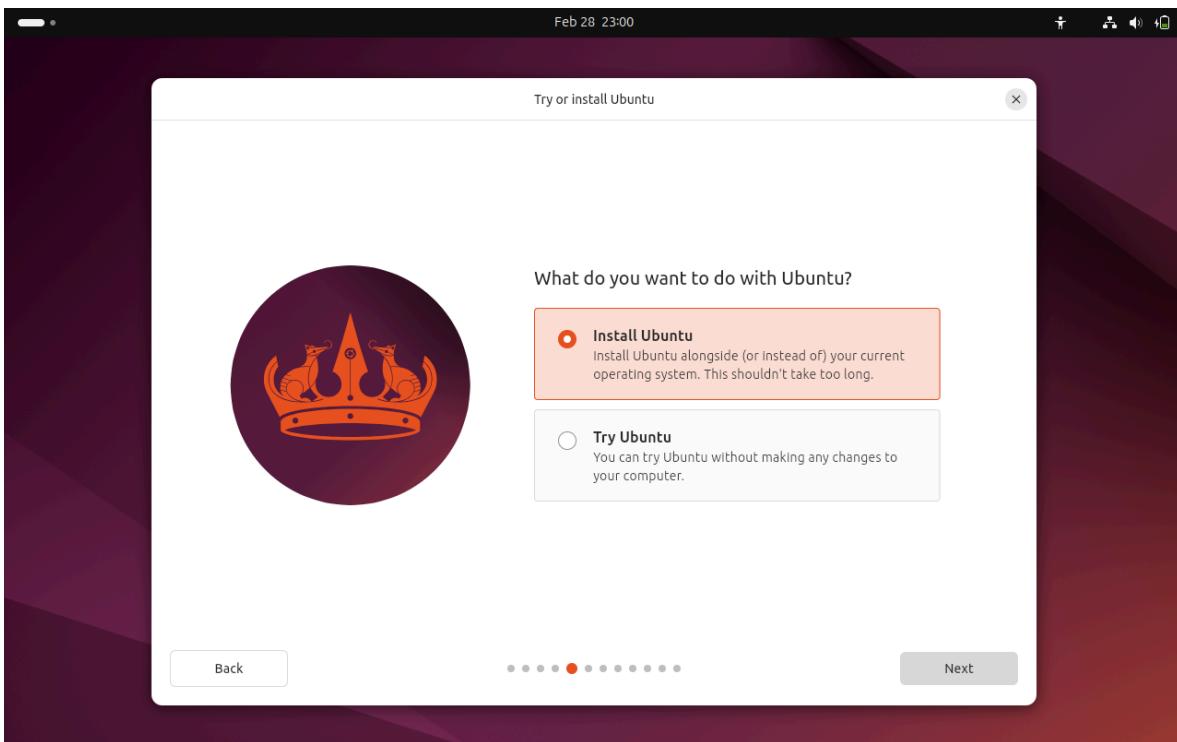
Na parte de Acessibilidade, é só se for necessário



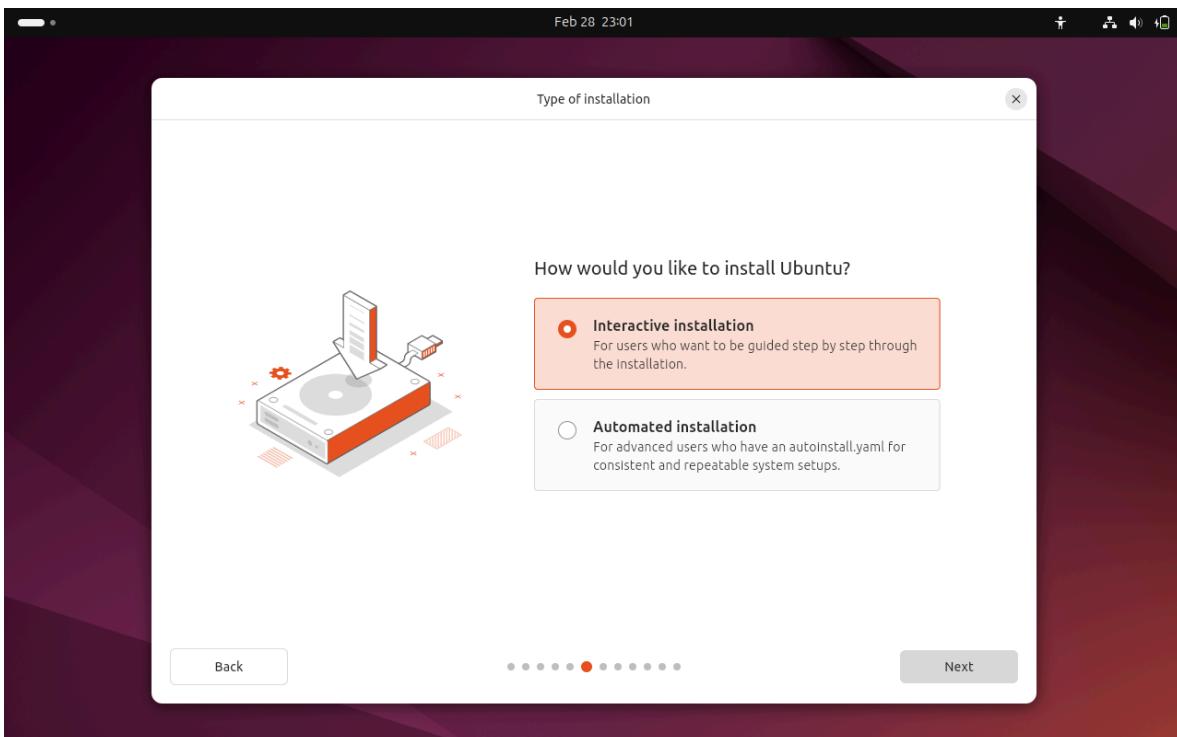
Selecionaremos agora o layout do teclado



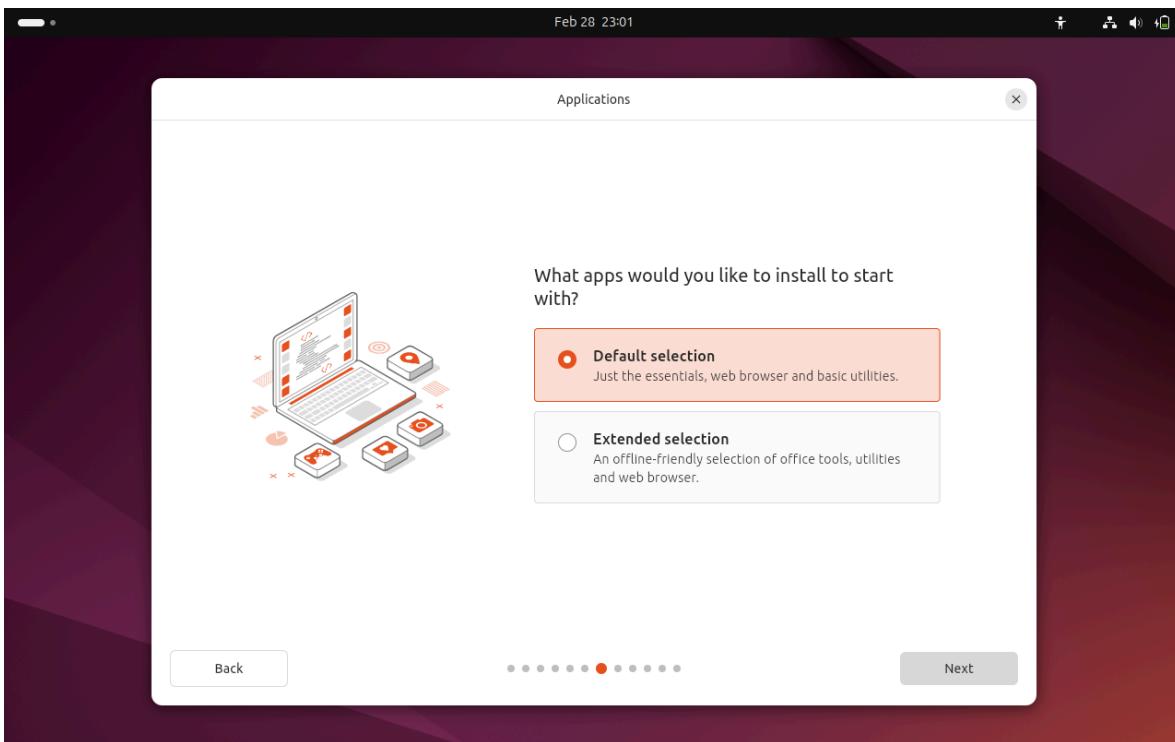
Deixe na forma padrão de conexão



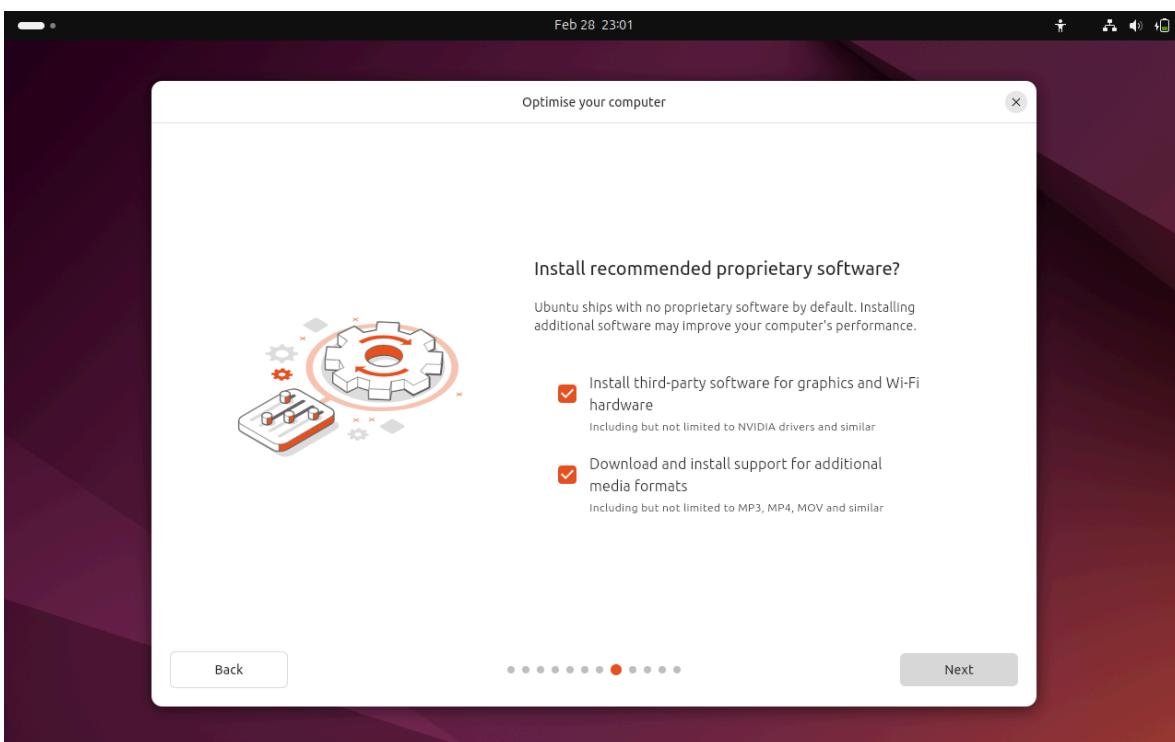
Aperte em "Next" ou "Próximo", deixe selecionada a forma padrão de instalação



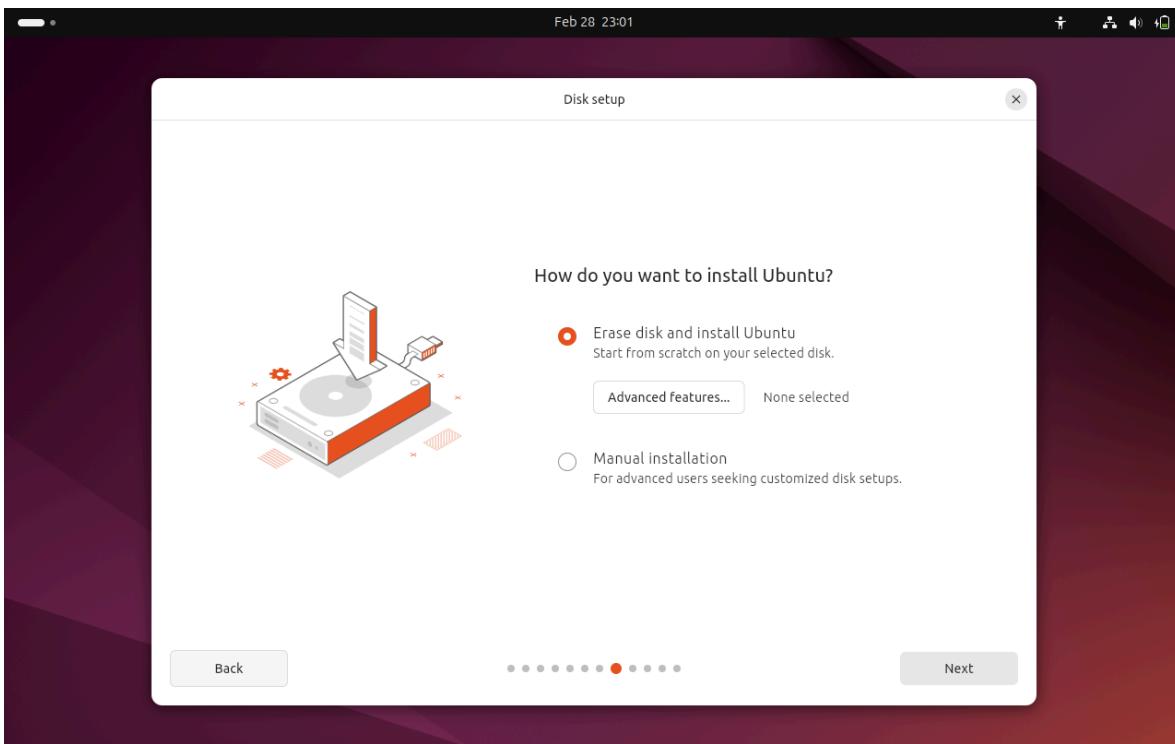
Deixe na forma interativa de instalação, ou seja, primeira opção



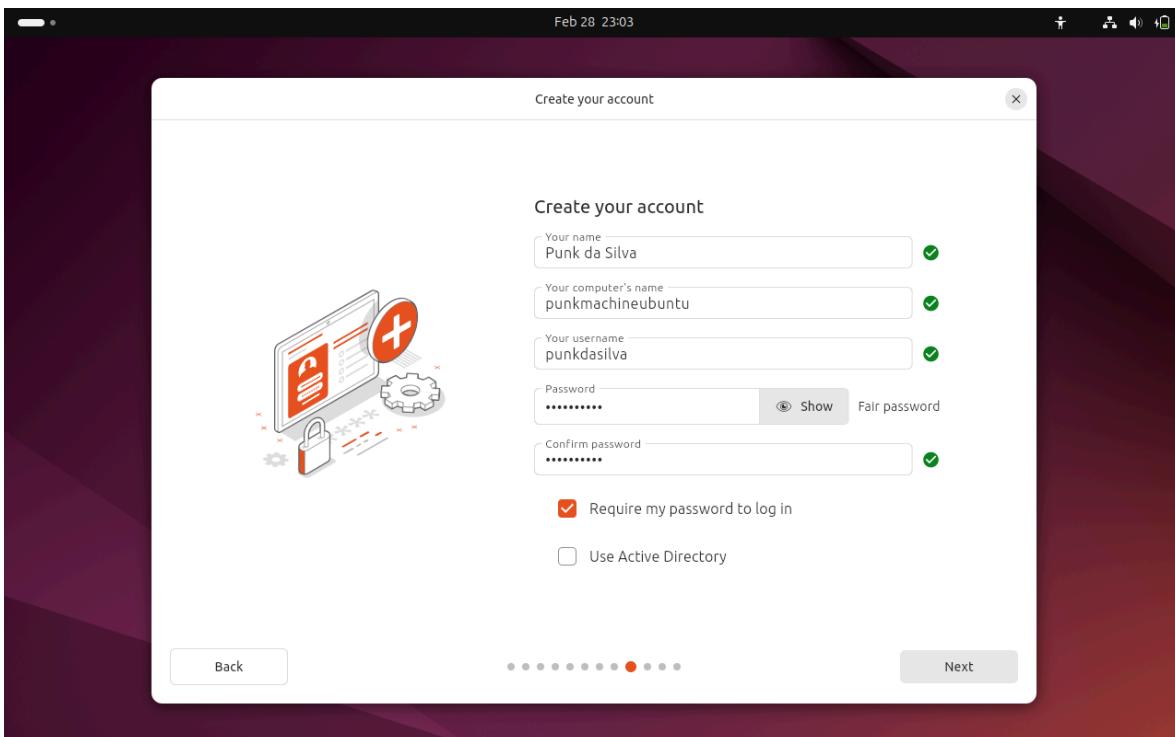
Para a próxima parte, é onde definimos se queremos que ao instalar o Ubuntu sejam instalados aplicativos adicionais, além dos básicos como navegador e outros utilitários. Neste caso, deixe na opção padrão



Nesta etapa, selecione todas as opções, que são para instalar softwares de terceiros e download de formatos de mídia adicionais



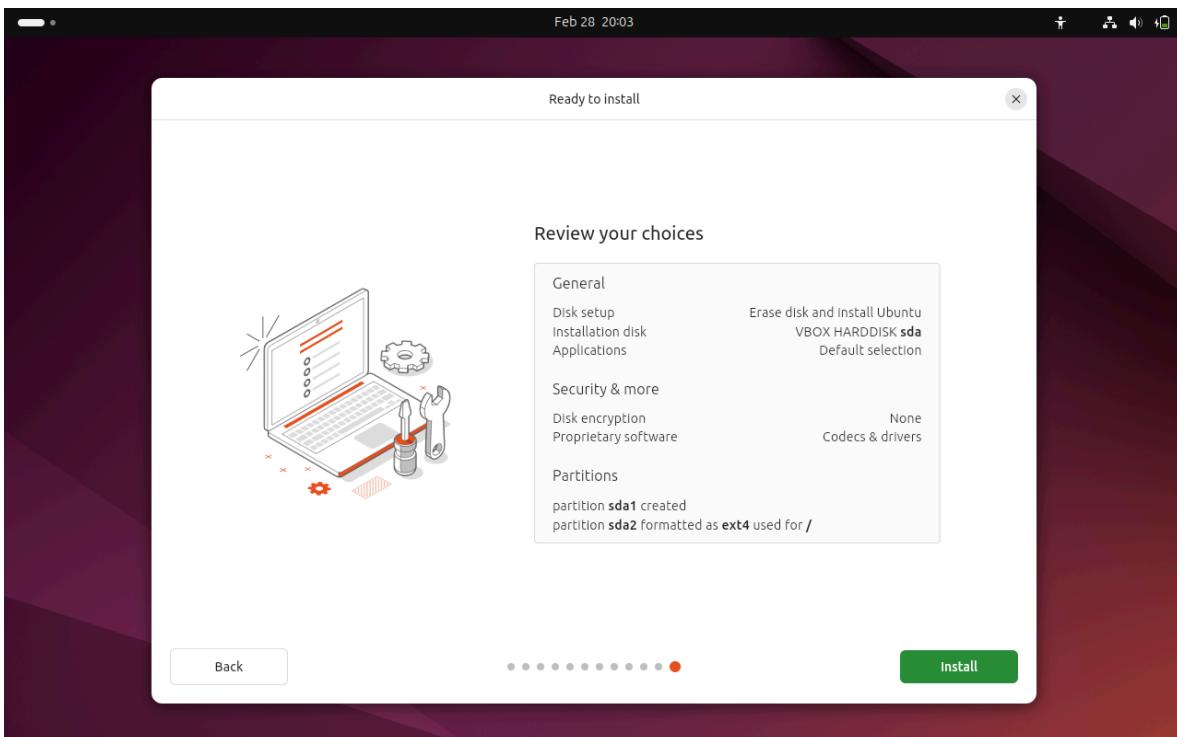
Nesta parte é a definição de se iremos instalar limpando o disco ou se faremos o particionamento do disco. Deixe a opção como padrão e aperte em next



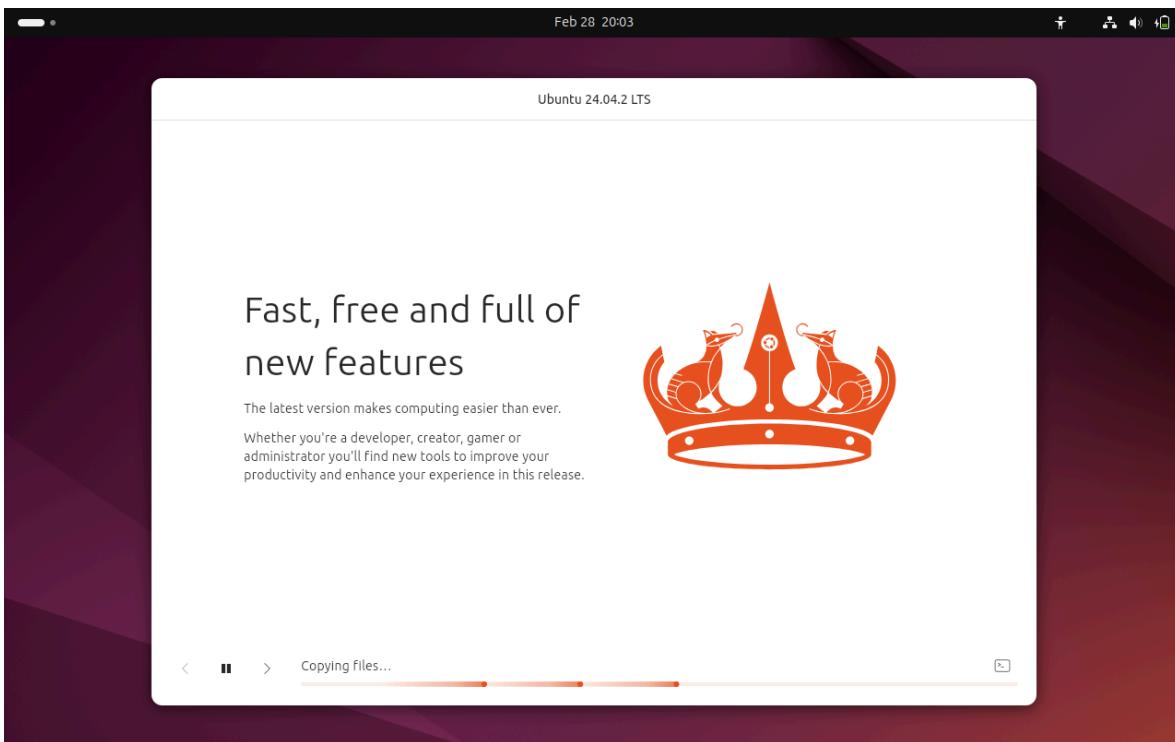
Agora na parte de criação de conta, defina suas credenciais



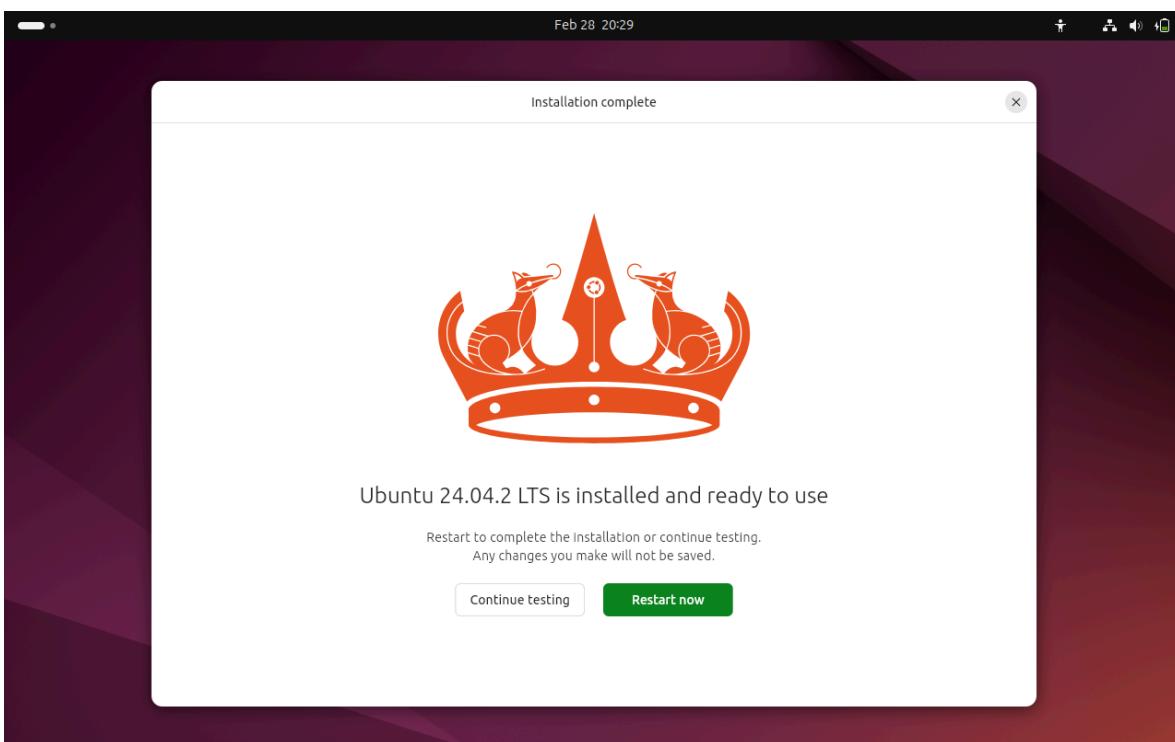
Agora é só fazermos a configuração do fuso horário, ou seja, do tempo que o computador irá usar



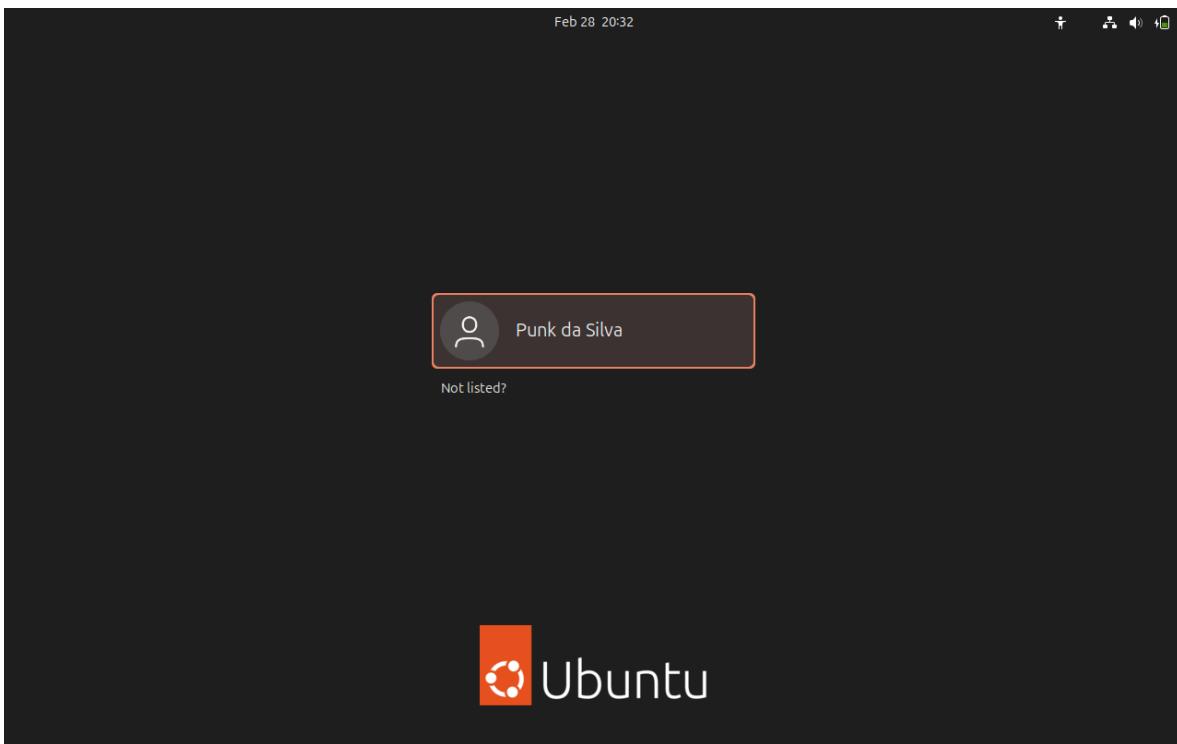
Nesta página será apenas para mostrar o resumo da instalação, uma visão geral das configurações para instalação



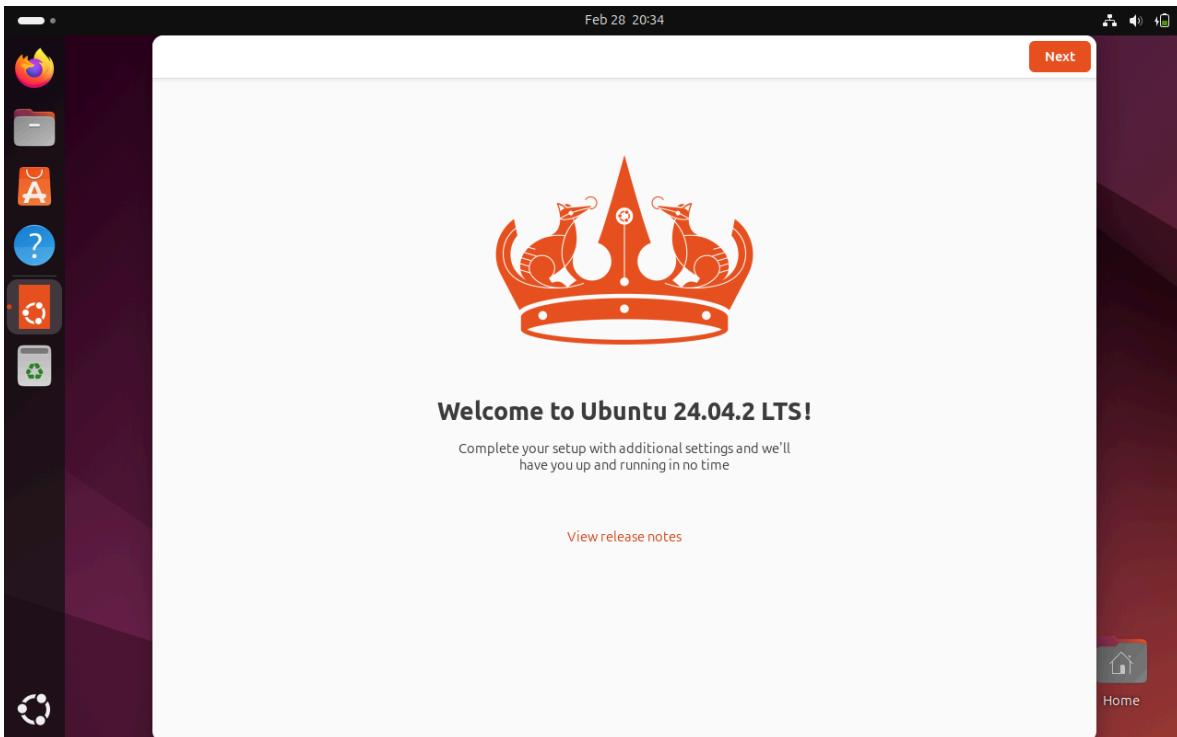
Aqui é a etapa em que o sistema será instalado, pode demorar uns 30 minutos



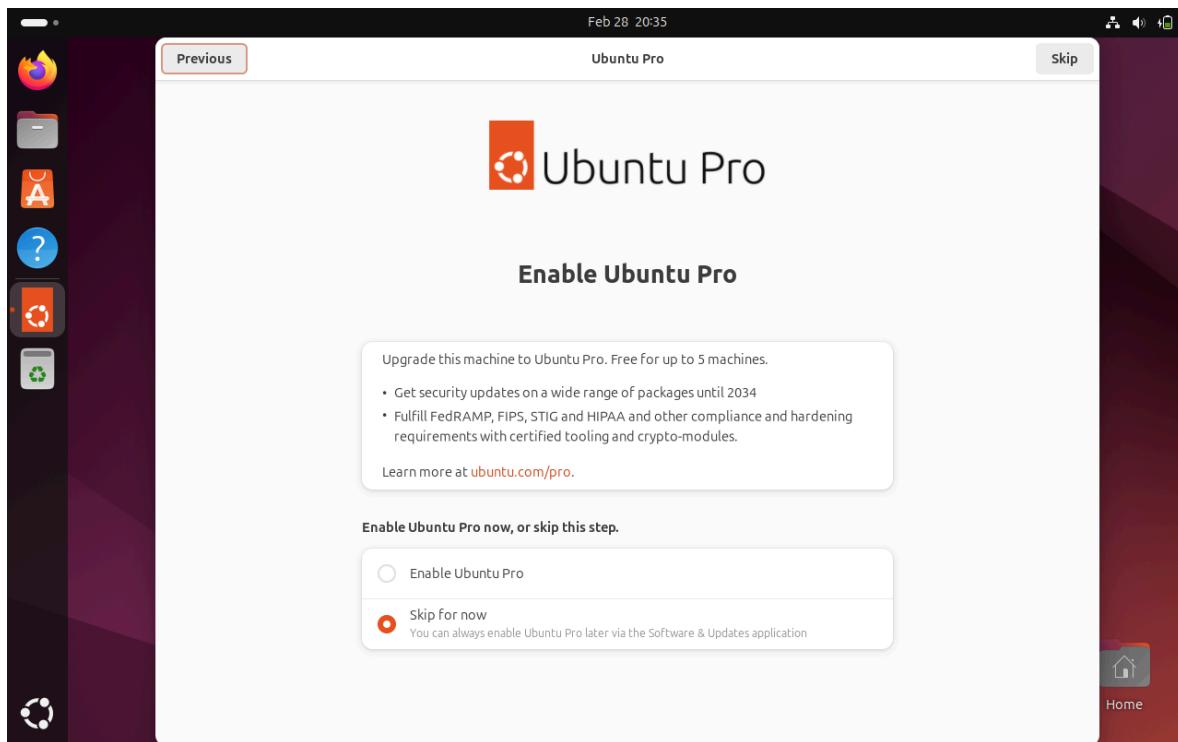
Com a etapa anterior concluída, o sistema já foi instalado e já podemos reiniciar a máquina



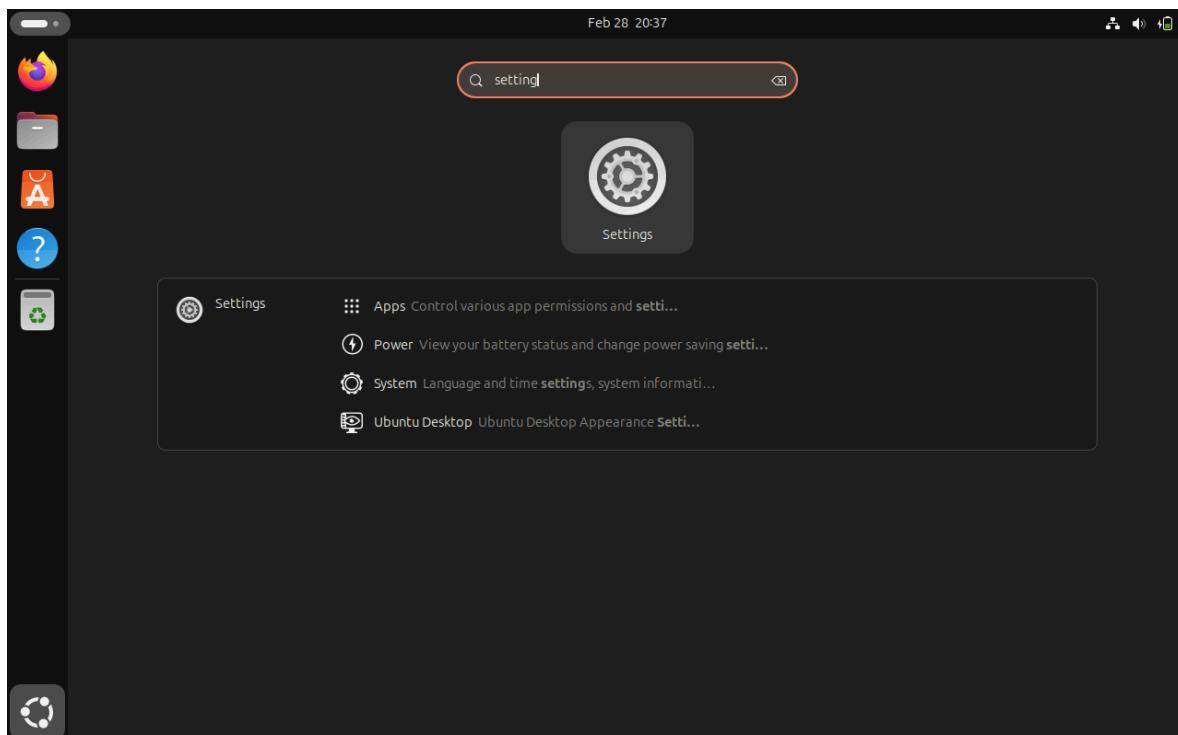
Ao reiniciarmos a máquina, aparece então a tela de login do usuário que foi criado



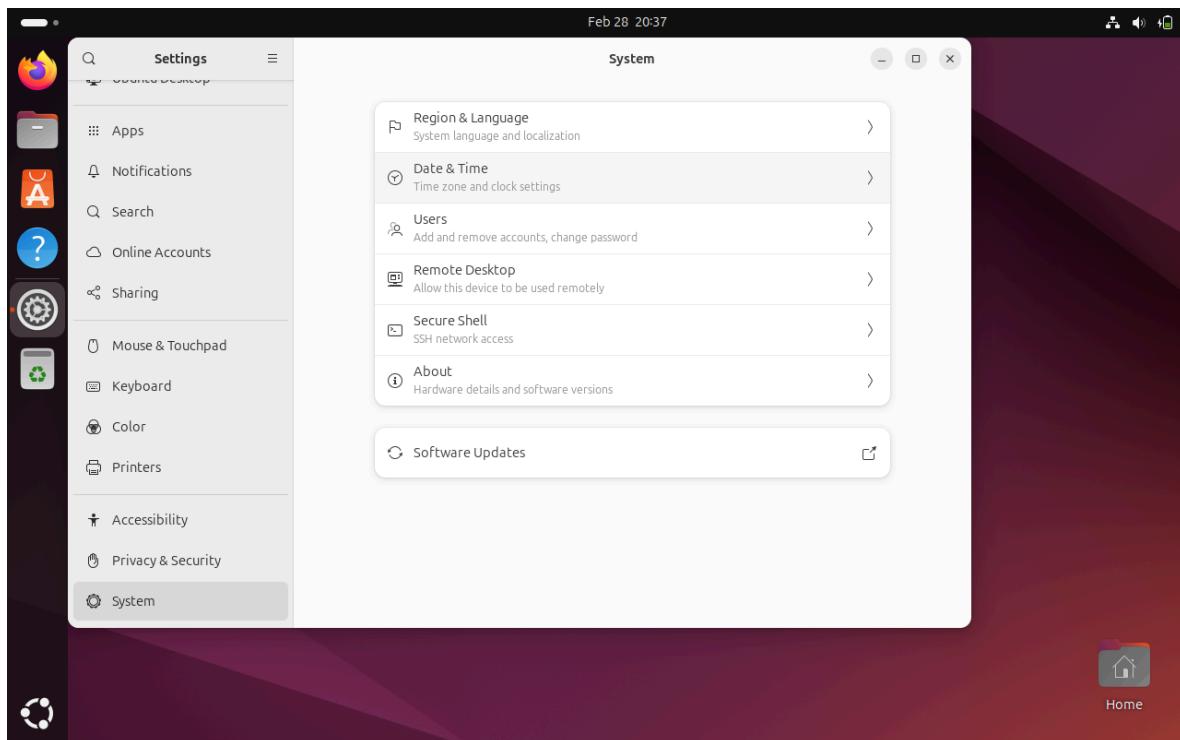
Ao logarmos, temos a visão desta tela



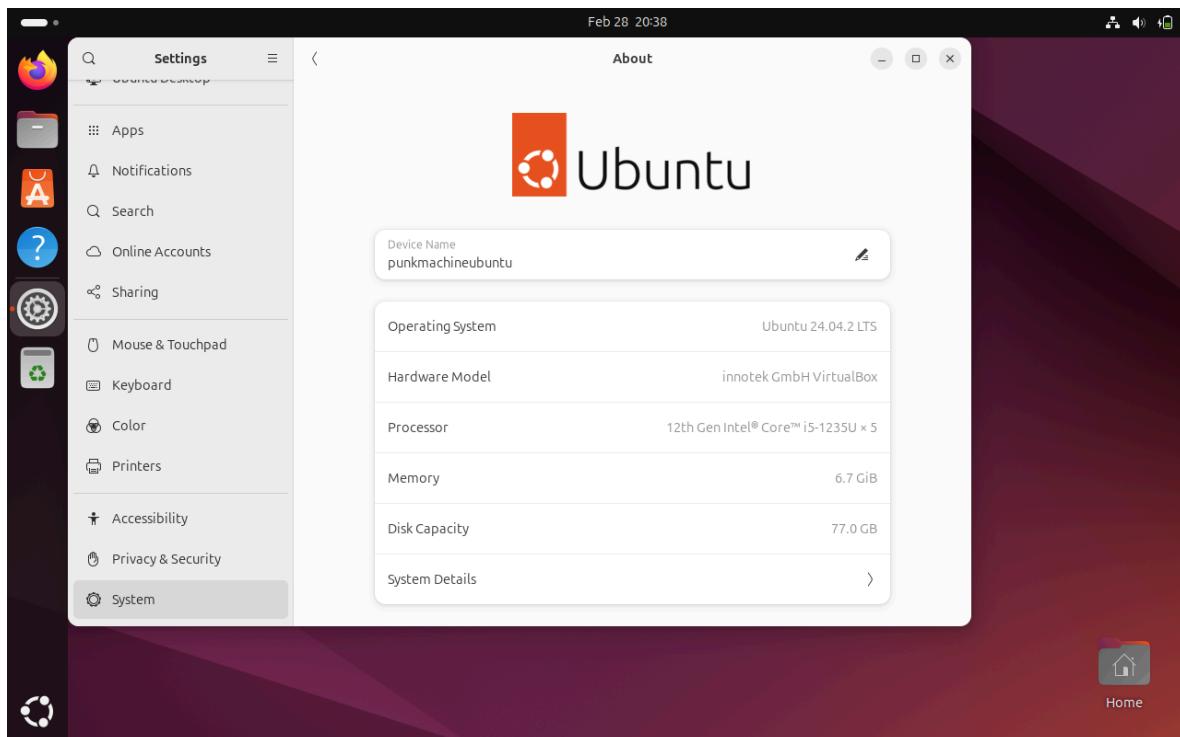
Nessa parte, basta passarmos adiante - clique em Skip ou Prosseguir



Aperte a tecla Super do seu teclado e digite "settings" ou "configurações" e aperte no primeiro item



Vá na parte inferior do menu lateral esquerdo e aperte em "System" ou "Sistema" e em seguida aperte em "About" ou "Sobre"



Assim podemos visualizar as informações do sistema que está instalado

Respostas

#	Resposta Correta
1	b
2	b
3	b
4	b
5	c
6	b
7	d
8	b
9	d
10	d
11	d
12	a
13	d
14	d
15	b

16	a
17	d
18	d
19	a
20	c
21	a
22	b
23	a
24	b
25	b