Nome do computador: ceosoft-031
Tipo: Desktop
Gerado por: alex

Relatório gerado em: 02/10/2025 - 17:29

INFORMAÇÕES DO SISTEMA

Versão do sistema operacional: Microsoft Windows 10 Pro (Version 10.0.19045) - 64 bits
Status do sistema operacional: Ativado

Versão do Office: Microsoft Office Professional Plus 2019 - pt-br 16.0.19231.20156
Status do Office: Ativado

SQL Server 1: Instância: Default | Versão: 16.0.1000.6 | Status: Running

Antivírus 1: Kaspersky | Status: Ativado
Antivírus 2: Windows Defender | Status: Desconhecido

Rede: Domínio: ceosoftwaread.net

INFORMAÇÕES DE HARDWARE

Memória RAM total: 4.0 GB
Pente de Memória 1: 1 GB | DDR2 | 800 MHz | DIMM
Fabricante: 7F94000000000000
Pente de Memória 2: 1 GB | DDR2 | 800 MHz | DIMM
Fabricante: 7F94000000000000
Pente de Memória 3: 1 GB | DDR2 | 800 MHz | DIMM
Fabricante: 7F94000000000000
Pente de Memória 4: 1 GB | DDR2 | 800 MHz | DIMM
Fabricante: 7F94000000000000

Processador 1: Intel(R) Core(TM)2 Duo CPU E7200 @ 2.53GHz
Cores: 2 físicos | 2 lógicos
Cache: L2: 3 MB | L3: N/A
Fabricante: GenuineIntel

Disco 1: WDC WDS240G2G0A-00JH30 ATA Device | Tamanho: 223.57 GB
Tipo: SSD | Interface: IDE
Unidade C: (Sem rótulo) | Total: 222.01 GB | Usado: 183.35 GB | Livre: 38.66 GB | Sistema: NTFS

Monitor 1: DEL | Modelo: DELL E198WFP | Serial: X763G88N1TRS
Monitor 2: GSM | Modelo: LG FULL HD | Serial: 302AZRD61974

Teclado conectado: SIM
Mouse conectado: SIM

Placa mãe: Dell Inc. | Modelo: 0CU409 | Serial: ..BR108198A60060.

Placa de Rede 1: Intel(R) 82562V-2 10/100 Network Connection | Fabricante: Intel | Velocidade: 100 Mbps | MAC: 00:1E:C9:1F:C7:59

Placa de Vídeo 1: BB Capture Driver | Fabricante: Blueberry Consultants | Memória: N/A | Tipo: Onboard (Integrada)
Placa de Vídeo 2: NVIDIA GeForce 210 | Fabricante: NVIDIA | Memória: 1 GB | Tipo: Offboard (Dedicada)

Impressora 1: OneNote (Desktop) | Serial/ID: NÃO OBTIDO | Fabricante: NÃO OBTIDO | Modelo: Send to Microsoft OneNote 16 Driver
Impressora 2: OneNote for Windows 10 | Serial/ID: NÃO OBTIDO | Fabricante: NÃO OBTIDO | Modelo: Microsoft Software Printer Driver
Impressora 3: Microsoft XPS Document Writer | Serial/ID: NÃO OBTIDO | Fabricante: NÃO OBTIDO | Modelo: Microsoft XPS Document Writer v4
Impressora 4: Microsoft Print to PDF | Serial/ID: NÃO OBTIDO | Fabricante: NÃO OBTIDO | Modelo: Microsoft Print To PDF
Impressora 5: Fax | Serial/ID: NÃO OBTIDO | Fabricante: NÃO OBTIDO | Modelo: Microsoft Shared Fax Driver

Impressora 6: AnyDesk Printer | Serial/ID: NÃO OBTIDO | Fabricante: NÃO OBTIDO | Modelo: AnyDesk v4 Printer Driver

ADMINISTRADORES

Administrador 1: CEOSOFT-031\Administrador
Administrador 2: CEOSOFT-031\CEOSOFTWARE
Administrador 3: CEOSOFTWAREAD\Admins. do domínio
Administrador 4: CEOSOFTWAREAD\alex
Administrador 5: CEOSOFTWAREAD\Alexandre.Amorim
Administrador 6: CEOSOFTWAREAD\Andre.Luiz
Administrador 7: CEOSOFTWAREAD\Angela.Joisciene
Administrador 8: CEOSOFTWAREAD\Angelo.Gabriel
Administrador 9: CEOSOFTWAREAD\Heberty.Richardy
Administrador 10: CEOSOFTWAREAD\Leandro.Magalhaes
Administrador 11: CEOSOFTWAREAD\Marco.Aurelio
Administrador 12: CEOSOFTWAREAD\Rafael.Freitas
Administrador 13: CEOSOFTWAREAD\Vinicius.Souza

SOFTWARES INSTALADOS

1. Adobe Acrobat (64-bit) | Versão: 25.001.20756 | Editor: Adobe
2. Adobe Refresh Manager | Versão: 1.8.0 | Editor: Adobe Systems Incorporated
3. AnyDesk | Versão: ad 6.0.8 | Editor: philandro Software GmbH
4. Asian Language And Spelling Dictionaries Support For Adobe Acrobat Reader | Versão: 23.008.20421 | Editor: Adobe Systems Incorporated
5. BB TestAssistant 2 | Versão: N/A | Editor: Blueberry Software (UK) Ltd.
6. Data Dynamics ActiveReports Professional 2 | Versão: 2.3.0.1261 | Editor: Data Dynamics, Ltd.
7. Driver do Microsoft OLE DB para SQL Server | Versão: 18.7.4.0 | Editor: Microsoft Corporation
8. EFD Contribuições PVA | Versão: PVA | Editor: N/A
9. ERP CEOSoftware 8_2 | Versão: 8.2 | Editor: CEOSoftware Sistemas de Informática
10. FileZilla 3.67.0 | Versão: 3.67.0 | Editor: Tim Kosse
11. Firebird 2.5.0.26074 (Win32) | Versão: 2.5.0.26074 | Editor: Firebird Project
12. GatewayComponents | Versão: 16.18.3 | Editor: Microsoft Corporation
13. GDR 1121 para SQL Server 2022 (KB5040936) (64-bit) | Versão: 16.0.1121.4 | Editor: Microsoft Corporation
14. GDR 1125 para SQL Server 2022 (KB5042211) (64-bit) | Versão: 16.0.1125.1 | Editor: Microsoft Corporation
15. GDR 1130 para SQL Server 2022 (KB5046057) (64-bit) | Versão: 16.0.1130.5 | Editor: Microsoft Corporation
16. GDR 1135 para SQL Server 2022 (KB5046861) (64-bit) | Versão: 16.0.1135.2 | Editor: Microsoft Corporation
17. GDR 1140 para SQL Server 2022 (KB5058712) (64-bit) | Versão: 16.0.1140.6 | Editor: Microsoft Corporation
18. GDR 1145 para SQL Server 2022 (KB5063756) (64-bit) | Versão: 16.0.1145.1 | Editor: Microsoft Corporation
19. GDR 1150 para SQL Server 2022 (KB5065221) (64-bit) | Versão: 16.0.1150.1 | Editor: Microsoft Corporation
20. Git | Versão: 2.51.0 | Editor: The Git Development Community
21. Google Chrome | Versão: 141.0.7390.54 | Editor: Google LLC
22. Gravador VSS da Microsoft para SQL Server 2022 | Versão: 16.0.1000.6 | Editor: Microsoft Corporation
23. Integration Services | Versão: 16.0.5491.7 | Editor: Microsoft Corporation
24. IRPF 2025 - Declaração de Ajuste Anual, Final de Espólio e Saída Definitiva do País | Versão: 1.0 | Editor: Receita Federal do Brasil
25. Kaspersky | Versão: 21.22.7.466 | Editor: Kaspersky
26. KPL.ONCLICK | Versão: 4.0.9 | Editor: KPL ONCLICK
27. Microsoft Edge | Versão: 140.0.3485.94 | Editor: Microsoft Corporation
28. Microsoft Edge WebView2 Runtime | Versão: 140.0.3485.94 | Editor: Microsoft Corporation
29. Microsoft Fabric Integration Runtime Preview | Versão: 5.54.9277.1 | Editor: Microsoft Corporation
30. Microsoft Help Viewer 2.3 | Versão: 2.3.28307 | Editor: Microsoft Corporation
31. Microsoft MPI (10.1.12498.18) | Versão: 10.1.12498.18 | Editor: Microsoft Corporation
32. Microsoft ODBC Driver 17 for SQL Server | Versão: 17.10.6.1 | Editor: Microsoft Corporation
33. Microsoft Office Access database engine 2007 (English) | Versão: 12.0.6612.1000 | Editor: Microsoft Corporation
34. Microsoft Office Professional Plus 2019 - pt-br | Versão: 16.0.19231.20156 | Editor: Microsoft Corporation
35. Microsoft Power BI Desktop (x64) | Versão: 2.145.1262.0 | Editor: Microsoft Corporation
36. Microsoft PowerBI Desktop (x64) | Versão: 2.145.1262.0 | Editor: Microsoft Corporation

37. Microsoft Project - pt-br | Versão: 16.0.19231.20156 | Editor: Microsoft Corporation
38. Microsoft SQL Server 2012 Native Client | Versão: 11.4.7001.0 | Editor: Microsoft Corporation
39. Microsoft SQL Server 2019 Setup (English) | Versão: 15.0.4013.40 | Editor: Microsoft Corporation
40. Microsoft SQL Server 2022 (64-bit) | Versão: N/A | Editor: N/A
41. Microsoft SQL Server 2022 RsFx Driver | Versão: 16.0.1000.6 | Editor: Microsoft Corporation
42. Microsoft SQL Server 2022 Setup (English) | Versão: 16.0.1150.1 | Editor: Microsoft Corporation
43. Microsoft SQL Server Management Studio - 20.2 | Versão: 20.2.30.0 | Editor: Microsoft Corporation
44. Microsoft Teams Meeting Add-in for Microsoft Office | Versão: 1.25.04401 | Editor: Microsoft
45. Microsoft Visual C++ 2013 x86 Additional Runtime - 12.0.40664 | Versão: 12.0.40664 | Editor: Microsoft Corporation
46. Microsoft Visual C++ 2013 x86 Minimum Runtime - 12.0.40664 | Versão: 12.0.40664 | Editor: Microsoft Corporation
47. Microsoft Visual C++ 2022 X64 Additional Runtime - 14.38.33135 | Versão: 14.38.33135 | Editor: Microsoft Corporation
48. Microsoft Visual C++ 2022 X64 Minimum Runtime - 14.38.33135 | Versão: 14.38.33135 | Editor: Microsoft Corporation
49. Microsoft Visual C++ 2022 X86 Additional Runtime - 14.38.33135 | Versão: 14.38.33135 | Editor: Microsoft Corporation
50. Microsoft Visual C++ 2022 X86 Minimum Runtime - 14.38.33135 | Versão: 14.38.33135 | Editor: Microsoft Corporation
51. Microsoft Visual Studio Tools for Applications 2019 | Versão: 16.0.31110 | Editor: Microsoft Corporation
52. Microsoft Visual Studio Tools for Applications 2019 x64 Hosting Support | Versão: 16.0.31110 | Editor: Microsoft Corporation
53. Microsoft Visual Studio Tools for Applications 2019 x86 Hosting Support | Versão: 16.0.31110 | Editor: Microsoft Corporation
54. MSI to redistribute MS Visual Studio Runtime libraries for x64 | Versão: 14.37.32822 | Editor: The Firebird Project
55. MSI to redistribute MS Visual Studio Runtime libraries for x86 | Versão: 14.37.32822 | Editor: The Firebird Project
56. MSI to redistribute MS VS2005 CRT libraries | Versão: 8.0.50727.42 | Editor: The Firebird Project
57. MSI to redistribute MS VS2010 CRT libraries | Versão: 10.0.30319.1 | Editor: The Firebird Project
58. Navegador para SQL Server 2022 | Versão: 16.0.1000.6 | Editor: Microsoft Corporation
59. Notepad++ (64-bit x64) | Versão: 8.7 | Editor: Notepad++ Team
60. NVIDIA Install Application | Versão: 2.1002.172.1382 | Editor: NVIDIA Corporation
61. Office 16 Click-to-Run Extensibility Component | Versão: 16.0.19231.20072 | Editor: Microsoft Corporation
62. Office 16 Click-to-Run Extensibility Component 64-bit Registration | Versão: 16.0.19231.20072 | Editor: Microsoft Corporation
63. Office 16 Click-to-Run Licensing Component | Versão: 16.0.19231.20138 | Editor: Microsoft Corporation
64. Office 16 Click-to-Run Localization Component | Versão: 16.0.19231.20072 | Editor: Microsoft Corporation
65. On-premises data gateway | Versão: 3000.274.3 | Editor: Microsoft Corporation
66. OpenSSL 3.4.1 (64-bit) | Versão: 3.4.1 | Editor: OpenSSL Win64 Installer Team
67. OpenVPN 2.6.11-I002 amd64 | Versão: 2.6.1102 | Editor: OpenVPN, Inc.
68. OpenVPN Connect | Versão: 3.5.1 | Editor: OpenVPN Inc.
69. Pacote de Idiomas do Microsoft Help Viewer 2.3 – PTB | Versão: 2.3.28107 | Editor: Microsoft Corporation
70. Pacote de Idiomas do SQL Server Management Studio - Português | Versão: 20.2.30.0 | Editor: Microsoft Corp.
71. Painel de controle da NVIDIA 342.01 | Versão: 342.01 | Editor: NVIDIA Corporation
72. Provedor Microsoft Analysis Services OLE DB | Versão: 16.0.5143.0 | Editor: Microsoft Corporation
73. RustDesk | Versão: 1.2.7 | Editor: RustDesk
74. Shell isolado do Visual Studio 2017 para o pacote de idiomas do SSMS – Português (Brasil) | Versão: 15.0.28307.421 | Editor: Microsoft Corporation
75. Sped Contábil 10.3.3 | Versão: 10.3.3 | Editor: Receita Federal do Brasil
76. Sped Fiscal 2.1.1-SNAPSHOT | Versão: 2.1.1-SNAPSHOT | Editor: N/A
77. SQL Lite | Versão: 1.27.3 | Editor: Friendship Solutions
78. SQL Server 2019 Serviço de Linguagem T-SQL da Microsoft | Versão: 15.0.2000.5 | Editor: Microsoft Corporation
79. SQL Server 2022 Advanced Analytics | Versão: 16.0.1000.6 | Editor: Microsoft Corporation
80. SQL Server 2022 Batch Parser | Versão: 16.0.1000.6 | Editor: Microsoft Corporation
81. SQL Server 2022 Common Files | Versão: 16.0.1000.6 | Editor: Microsoft Corporation
82. SQL Server 2022 Connection Info | Versão: 16.0.1000.6 | Editor: Microsoft Corporation
83. SQL Server 2022 Database Engine Services | Versão: 16.0.1000.6 | Editor: Microsoft Corporation
84. SQL Server 2022 Database Engine Shared | Versão: 16.0.1000.6 | Editor: Microsoft Corporation
85. SQL Server 2022 DMF | Versão: 16.0.1000.6 | Editor: Microsoft Corporation
86. SQL Server 2022 Full text search | Versão: 16.0.1000.6 | Editor: Microsoft Corporation
87. SQL Server 2022 Shared Management Objects | Versão: 16.0.1000.6 | Editor: Microsoft Corporation
88. SQL Server 2022 Shared Management Objects Extensions | Versão: 16.0.1000.6 | Editor: Microsoft Corporation
89. SQL Server 2022 SQL Diagnostics | Versão: 16.0.1000.6 | Editor: Microsoft Corporation
90. SQL Server 2022 XEvent | Versão: 16.0.1000.6 | Editor: Microsoft Corporation
91. SQL Server Management Studio | Versão: 20.2.30.0 | Editor: Microsoft Corp.

92. SSMS Post Install Tasks | Versão: 20.2.30.0 | Editor: Microsoft Corporation
93. TeamViewer | Versão: 15.68.5 | Editor: TeamViewer
94. TortoiseSVN 1.14.8.29723 (64 bit) | Versão: 1.14.29723 | Editor: TortoiseSVN
95. Visual Studio 2017 Isolated Shell for SSMS | Versão: 15.0.28307.421 | Editor: Microsoft Corporation
96. WinRAR 7.13 (64-bit) | Versão: 7.13.0 | Editor: win.rar GmbH
97. XAMPP | Versão: 8.2.12-0 | Editor: Apache Friends

INFORMAÇÕES DE REDE

Adaptador 1: Intel(R) 82562V-2 10/100 Network Connection
IP: 192.168.0.114 | Gateway: 192.168.0.1 | DNS: 192.168.0.3, 192.168.0.4, 8.8.8.8 | MAC: 00:1E:C9:1F:C7:59

SEGURANÇA DO SISTEMA

Firewall:
Perfil: Domain | Status: Ativado
Perfil: Private | Status: Ativado
Perfil: Public | Status: Ativado

Windows Update:
Nenhuma atualização pendente

LOG DE DEBUG (CSINFO)

--- CSInfo debug entry ---
TIMESTAMP_UTC: 2025-10-02T20:28:47.815829Z
COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $adapters = Get-WmiObject Win32_NetworkAdapterConfiguration | Where-Object { $_.IPEnabled -eq $true }\n $out = @()\n foreach ($a in $adapters) {\n $ip = $a.IPAddress[0]\n $gw = $a.DefaultIPGateway[0]\n $dns = $a.DNSServerSearchOrder -join '', ''\n $mac = $a.MACAddress\n $desc = $a.Description\n $out += [PSCustomObject]@{\n IP = $ip; Gateway = $gw; DNS = $dns; MAC = $mac; Descricao = $desc\n }\n }\n $out | ConvertTo-Json -Compress\n } -ErrorAction SilentlyContinue']
COMPUTER: ceosoft-031
TIMEOUT: 20
DURATION_SECONDS: 5.658945322036743
RETURN_CODE: 0
OUTPUT:
{"IP":"192.168.0.114","Gateway":"192.168.0.1","DNS":"192.168.0.3, 192.168.0.4, 8.8.8.8","MAC":"00:1E:C9:1F:C7:59","Descricao":"Intel(R) 82562V-2 10/100 Network Connection"}
--- CSInfo debug entry ---
TIMESTAMP_UTC: 2025-10-02T20:28:57.250449Z
COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n try {\n $profiles = Get-NetFirewallProfile\n $out = @()\n foreach ($p in $profiles) {\n $out += [PSCustomObject]@{\n Perfil = $p.Name; Ativado = $p.Enabled\n }\n }\n $out | ConvertTo-Json -Compress\n } catch { "NÃO OBTIDO" }\n } -ErrorAction SilentlyContinue']
COMPUTER: ceosoft-031
TIMEOUT: 20
DURATION_SECONDS: 9.431214094161987
RETURN_CODE: 0
OUTPUT:
[{"Perfil":"Domain","Ativado":1},{"Perfil":"Private","Ativado":1},{"Perfil":"Public","Ativado":1}]
--- CSInfo debug entry ---
TIMESTAMP_UTC: 2025-10-02T20:29:26.172860Z
COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n try {\n $session = New-Object -ComObject Microsoft.Update.Session\n $searcher = $session.CreateUpdateSearcher()\n $result = $searcher.Search("IsInstalled=0")\n $count = $result.Updates.Count\n if ($count -eq 0) { "Nenhuma atualização pendente" }\n else { "$count atualizações pendentes" }\n } catch { "NÃO OBTIDO" }\n } -ErrorAction SilentlyContinue']
COMPUTER: ceosoft-031
TIMEOUT: 60
DURATION_SECONDS: 28.912227630615234
RETURN_CODE: 0
OUTPUT:
Nenhuma atualização pendente
--- CSInfo debug entry ---
TIMESTAMP_UTC: 2025-10-02T20:29:30.533132Z
COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n Get-Process | Select-Object -First 10 Name,Id,CPU | ConvertTo-Json -Compress\n } -ErrorAction SilentlyContinue']
COMPUTER: ceosoft-031
TIMEOUT: 20
DURATION_SECONDS: 4.357950925827026
RETURN_CODE: 0
OUTPUT:
[{"Name":"Acrobat","Id":8568,"CPU":26.828125},{"Name":"Acrobat","Id":10816,"CPU":3.34375},{"Name":"AcroCEF","Id":5500,"CPU":0.59375},{"Name":"AcroCEF","Id":7204,"CPU":0.8125},{"Name":"AcroCEF","Id":7664,"CPU":3.921875},{"Name":"AcroCEF","Id":10068,"CPU":0.234375},{"Name":"AcroCEF","Id":10208,"CPU":3.234375},{"Name":"AcroCEF","Id":15172,"CPU":3.953125},{"Name":"AdobeCollabSync","Id":3192,"CPU":0.1875},{"Name":"AdobeCollabSync","Id":7156,"CPU":0.21875}]
--- CSInfo debug entry ---
TIMESTAMP_UTC: 2025-10-02T20:29:31.922862Z
COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $names = @("wuauserv",

"WinDefend", "BITS", "Spooler", "LanmanServer", "LanmanWorkstation")\n $out = @()\n foreach ($n in $names) {\n $s = Get-Service -Name $n -ErrorAction SilentlyContinue\n if ($s) {\n $out += [PSCustomObject]@{Nome=$s.Name; Status=$s.Status}\n }\n }\n $out | ConvertTo-Json -Compress\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 1.3891193866729736

RETURN_CODE: 0

OUTPUT:

[{"Nome":"wuauserv","Status":4},{"Nome":"WinDefend","Status":1},{"Nome":"BITS","Status":1},{"Nome":"Spooler","Status":4},{"Nome":"LanmanServer","Status":4},{"Nome":"LanmanWorkstation","Status":4}]

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:29:33.291838Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n try {\n $namespace = "root\\\\SecurityCenter2"\n $firewallProducts = Get-WmiObject -Namespace $namespace -Class FirewallProduct -ErrorAction SilentlyContinue\n $out = @()\n foreach ($fw in $firewallProducts) {\n $out += $fw.displayName\n }\n if ($out.Count -gt 0) {\n $out -join ", "\n } else {\n "Windows Firewall (padrão)"\n }\n } catch {\n "NÃO OBTIDO"\n }\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 1.3683748245239258

RETURN_CODE: 0

OUTPUT:

Windows Firewall (padrão)

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:29:34.699312Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { Get-CimInstance Win32_Battery | ConvertTo-Json -Compress } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 1.4045500755310059

RETURN_CODE: 0

OUTPUT:

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:29:36.014728Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { (Get-CimInstance Win32_SystemEnclosure | Select-Object -ExpandProperty ChassisTypes | ConvertTo-Json -Compress) } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 1.314915657043457

RETURN_CODE: 0

OUTPUT:

3

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:29:37.519257Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { (Get-CimInstance Win32_OperatingSystem | Select-Object -Property Caption,Version,OSArchitecture | ConvertTo-Json -Compress) } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 1.5035574436187744

RETURN_CODE: 0

OUTPUT:

{"Caption":"Microsoft Windows 10 Pro","Version":"10.0.19045","OSArchitecture":"64 bits"}

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:29:39.607788Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n try {\n # Usar slmgr diretamente\n $slmgrResult = & cscript //nologo C:\\Windows\\System32\\slmgr.vbs /xpr\n $output = $slmgrResult -join "`n"\n \n # Verificar diferentes padrões de texto para ativação\n if ($output -match "ativada permanentemente|permanently activated|permanently|permanente") {\n "Ativado"\n } elseif ($output -match "grace period|período de carência|carência") {\n "Período de carência"\n } elseif ($output -match "notification|notificação") {\n "Período de notificação"\n } elseif ($output -match "not activated|não ativado|não está ativado") {\n "Não ativado"\n } else {\n # Se não conseguir interpretar, retornar a saída original limpa\n ($output -replace "`r", "" -replace "`n", " ").Trim()\n }\n } catch {\n try {\n # Método alternativo usando Get-WmiObject\n $licenses = Get-WmiObject -Class SoftwareLicensingProduct | Where-Object {\n $_.Name -like "*Windows*" -and $_.PartialProductKey -ne $null\n } | Select-Object -First 1\n \n if ($licenses) {\n switch ($licenses.LicenseStatus) {\n 1 { "Ativado" }\n 0 { "Não licenciado" }\n 2 { "Período de carência" }\n 3 { "OOT (Out of Tolerance)" }\n 4 { "OOB (Out of Box)" }\n 5 { "Notificação" }\n 6 { "Carência estendida" }\n default { "Status desconhecido: $($licenses.LicenseStatus)" }\n }\n } else {\n "Nenhuma licença encontrada"\n }\n } catch {\n "Erro na verificação"\n }\n }\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 2.0877203941345215

RETURN_CODE: 0

OUTPUT:

Ativado

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:29:41.255501Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $apps = Get-ItemProperty HKLM:\\Software\\Microsoft\\Windows\\CurrentVersion\\Uninstall* -ErrorAction SilentlyContinue | Where-Object {$_.DisplayName -like "*Office*" -or $_.DisplayName -like "*Microsoft 365*"}\n if ($apps) { ($apps | Select-Object -First 1).DisplayName + " " + ($apps | Select-Object -First 1).DisplayVersion } else { "Não encontrado" }\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 1.64705228805542

RETURN_CODE: 0

OUTPUT:

Microsoft Office Professional Plus 2019 - pt-br 16.0.19231.20156

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:30:01.343289Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n try {\n # Verificar Office através do registro\n $officeApps = @("Word", "Excel", "PowerPoint", "Outlook")\n $activated = $false\n \n foreach ($app in $officeApps) {\n try {\n $comObject = New-Object -ComObject "$app.Application"\n if ($comObject) {\n $activated = $true\n $comObject.Quit()\n break\n }\n } catch {}\n }\n \n if ($activated) {\n "Ativado"\n } else {\n # Método alternativo: verificar chaves de ativação no registro\n $regPaths = @(\n "HKLM:\\SOFTWARE\\Microsoft\\Office*\\Registration",\n "HKLM:\\SOFTWARE\\WOW6432Node\\Microsoft\\Office*\\Registration"\n)\n \n $foundActivation = $false\n foreach ($path in $regPaths) {\n try {\n $items = Get-ChildItem $path -ErrorAction SilentlyContinue\n if ($items) {\n $foundActivation = $true\n break\n }\n } catch {}\n }\n \n if ($foundActivation) {\n "Possivelmente ativado"\n } else {\n "Não detectado"\n }\n }\n } catch {\n "NÃO OBTIDO"\n }\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 20.08710265159607

EXCEPTION: Command '['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n try {\n # Verificar Office através do registro\n $officeApps = @("Word", "Excel", "PowerPoint", "Outlook")\n $activated = $false\n \n foreach ($app in $officeApps) {\n try {\n $comObject = New-Object -ComObject "$app.Application"\n if ($comObject) {\n $activated = $true\n $comObject.Quit()\n break\n }\n } catch {}\n }\n \n if ($activated) {\n "Ativado"\n } else {\n # Método alternativo: verificar chaves de ativação no registro\n $regPaths = @(\n "HKLM:\\SOFTWARE\\Microsoft\\Office*\\Registration",\n "HKLM:\\SOFTWARE\\WOW6432Node\\Microsoft\\Office*\\Registration"\n)\n \n $foundActivation = $false\n foreach ($path in $regPaths) {\n try {\n $items = Get-ChildItem $path -ErrorAction SilentlyContinue\n if ($items) {\n $foundActivation = $true\n break\n }\n } catch {}\n }\n \n if ($foundActivation) {\n "Possivelmente ativado"\n } else {\n "Não detectado"\n }\n }\n } catch {\n "NÃO OBTIDO"\n }\n } -ErrorAction SilentlyContinue']' timed out after 20 seconds

OUTPUT:

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:30:13.543497Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n try {\n # Verificar Office através do registro\n $officeApps = @("Word", "Excel", "PowerPoint", "Outlook")\n $activated = $false\n \n foreach ($app in $officeApps) {\n try {\n $comObject = New-Object -ComObject "$app.Application"\n if ($comObject) {\n $activated = $true\n $comObject.Quit()\n break\n }\n } catch {}\n }\n \n if ($activated) {\n "Ativado"\n } else {\n # Método alternativo: verificar chaves de ativação no registro\n $regPaths = @(\n "HKLM:\\SOFTWARE\\Microsoft\\Office*\\Registration",\n "HKLM:\\SOFTWARE\\WOW6432Node\\Microsoft\\Office*\\Registration"\n)\n \n $foundActivation = $false\n foreach ($path in $regPaths) {\n try {\n $items = Get-ChildItem $path -ErrorAction SilentlyContinue\n if ($items) {\n $foundActivation = $true\n break\n }\n } catch {}\n }\n \n if ($foundActivation) {\n "Possivelmente ativado"\n } else {\n "Não detectado"\n }\n }\n } catch {\n "NÃO OBTIDO"\n }\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 11.898736715316772

RETURN_CODE: 0

OUTPUT:

Ativado

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:30:22.606420Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $sqlInstances = @()\n \n # Buscar serviços do SQL Server\n try {\n $services = Get-WmiObject -Class Win32_Service -Filter "Name LIKE \'%SQL%\'" -ErrorAction SilentlyContinue\n $sqlServices = $services | Where-Object { $_.Name -match "MSSQL\\$" -or $_.Name -eq "MSSQLSERVER" }\n \n foreach ($service in $sqlServices) {\n $instanceName = if ($service.Name -eq "MSSQLSERVER") { "Default" } else { $service.Name -replace "MSSQL\\$", "" }\n $status = $service.State\n \n # Tentar obter versão do registro\n $version = ""\n try {\n if ($instanceName -eq "Default") {\n $regPath = "HKLM:\\SOFTWARE\\Microsoft\\Microsoft SQL Server\\MSSQL*\\MSSQLServer\\CurrentVersion"\n } else {\n $regPath = "HKLM:\\SOFTWARE\\Microsoft\\Microsoft SQL Server\\MSSQL*\\MSSQLServer\\CurrentVersion"\n }\n \n $versionKeys = Get-ChildItem "HKLM:\\SOFTWARE\\Microsoft\\Microsoft SQL Server\\" -ErrorAction SilentlyContinue | \n Where-Object { $_.Name -match "MSSQL\\d+\\." }\n \n foreach ($key in $versionKeys) {\n $currentVersionPath = Join-Path $key.PSPath "MSSQLServer\\CurrentVersion"\n if (Test-Path $currentVersionPath) {\n $versionReg = Get-ItemProperty $currentVersionPath -ErrorAction SilentlyContinue\n if ($versionReg.CurrentVersion) {\n $version = $versionReg.CurrentVersion\n break\n }\n }\n }\n } catch {}\n \n $sqlInstances += [PSCustomObject]@{\n Instance = $instanceName\n Status = $status\n Version = $version\n }\n }\n } catch {}\n \n # Se não encontrou serviços, tentar pelo registro\n if ($sqlInstances.Count -eq 0) {\n try {\n $regKeys = Get-ChildItem "HKLM:\\SOFTWARE\\Microsoft\\Microsoft SQL Server\\" -ErrorAction SilentlyContinue | \n Where-Object { $_.Name -match "MSSQL\\d+\\." }\n \n foreach ($key in $regKeys) {\n $setupPath = Join-Path $key.PSPath "Setup"\n if (Test-Path $setupPath) {\n $setup = Get-ItemProperty $setupPath -ErrorAction SilentlyContinue\n if ($setup.SqlProgramDir) {\n $sqlInstances += [PSCustomObject]@{\n Instance = $setup.Edition\n Status = "Unknown"\n Version = $setup.Version\n }\n }\n }\n } catch {}\n }\n \n $sqlInstances | ConvertTo-Json -Compress\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 9.062405586242676

RETURN_CODE: 0

OUTPUT:

{"Instance":"Default","Status":"Running","Version":"16.0.1000.6"}

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:30:25.363825Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $antivirusList = @()\n \n # Método 1: Windows Security Center (funciona no Windows 10/11)\n try {\n $namespace = "root\\SecurityCenter2"\n $antivirusProducts = Get-WmiObject -Namespace $namespace -Class AntiVirusProduct -ErrorAction SilentlyContinue\n \n foreach ($av in $antivirusProducts) {\n $name = $av.displayName\n $state = $av.productState\n \n # Decodificar o estado do produto\n $hex = [Convert]::ToString($state, 16).PadLeft(6, "0")\n $s2 = $hex.Substring(2,2) \n \n $enabled = "Desconhecido"\n \n # Verificar se está ativado (bit 4 e 5)\n if ([Convert]::ToInt32($s2, 16) -band 0x10) {\n $enabled = "Ativado"\n } elseif ([Convert]::ToInt32($s2, 16) -band 0x00) {\n $enabled = "Desativado"\n }\n \n $antivirusList += [PSCustomObject]@{\n Name = $name\n Enabled = $enabled\n }\n }\n } catch {}\n \n # Método 2: Verificar pelo registro (programas instalados)\n if ($antivirusList.Count -eq 0) {\n try {\n $uninstallKeys = @(\n "HKLM:\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Uninstall*",\n "HKLM:\\SOFTWARE\\WOW6432Node\\Microsoft\\Windows\\CurrentVersion\\Uninstall*"\n)\n \n $antivirusNames = @(\n "Avast", "AVG", "Kaspersky", "Norton", "McAfee", "Bitdefender", \n "ESET", "Trend Micro", "F-Secure", "Panda", "Sophos", "Malwarebytes",\n "Windows Defender", "Microsoft Defender", "Symantec", "Avira",\n "Quick Heal", "Comodo", "360 Total Security", "Baidu Antivirus"\n)\n \n foreach ($key in $uninstallKeys) {\n $programs = Get-ItemProperty $key -ErrorAction SilentlyContinue\n foreach ($program in $programs) {\n $displayName = $program.DisplayName\n if ($displayName) {\n foreach ($avName in $antivirusNames) {\n if ($displayName -like "*$avName*") {\n $antivirusList += [PSCustomObject]@{\n Name = $displayName\n Enabled = "Instalado"\n }\n break\n }\n }\n }\n }\n } catch {}\n }\n \n # Método 3: Verificar Windows Defender especificamente\n if

($antivirusList.Count -eq 0) {\n try {\n $defenderStatus = Get-MpComputerStatus -ErrorAction SilentlyContinue\n if ($defenderStatus) {\n $enabled = if ($defenderStatus.RealTimeProtectionEnabled) { "Ativado" } else { "Desativado" }\n \n $antivirusList += [PSCustomObject]@{\n Name = "Windows Defender"\n Enabled = $enabled\n }\n }\n } catch {}\n }\n \n # Se nada foi encontrado, verificar se há pelo menos o Windows Defender básico\n if ($antivirusList.Count -eq 0) {\n try {\n $defenderService = Get-Service -Name "WinDefend" -ErrorAction SilentlyContinue\n if ($defenderService) {\n $status = if ($defenderService.Status -eq "Running") { "Em execução" } else { $defenderService.Status }\n $antivirusList += [PSCustomObject]@{\n Name = "Windows Defender (Serviço)"\n Enabled = $status\n }\n }\n } catch {}\n }\n \n # Remover duplicatas baseado no nome\n $uniqueList = @()\n $seenNames = @()\n foreach ($av in $antivirusList) {\n $cleanName = $av.Name -replace \'\\s*\\d+.*$\', \'\' -replace \'\\s*\\(.*\\)\', \'\' -replace \'\\s+\', \' \'\n $cleanName = $cleanName.Trim()\n \n if ($seenNames -notcontains $cleanName) {\n $seenNames += $cleanName\n $uniqueList += [PSCustomObject]@{\n Name = $cleanName\n Enabled = $av.Enabled\n }\n }\n \n $uniqueList | ConvertTo-Json -Compress\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 2.756863832473755

RETURN_CODE: 0

OUTPUT:

[{"Name":"Kaspersky","Enabled":"Ativado"},{"Name":"Windows Defender","Enabled":"Desconhecido"}]

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:30:28.153505Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $computer = Get-CimInstance -ClassName Win32_ComputerSystem\n if ($computer.PartOfDomain) {\n "Domínio: $($computer.Domain)"\n } else {\n "Workgroup: $($computer.Workgroup)"\n }\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 2.788787841796875

RETURN_CODE: 0

OUTPUT:

Domínio: ceosoftwaread.net

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:30:31.207607Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { (Get-CimInstance Win32_ComputerSystem | Select-Object -Property TotalPhysicalMemory | ConvertTo-Json -Compress) } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 3.0535898208618164

RETURN_CODE: 0

OUTPUT:

{"TotalPhysicalMemory":4293050368}

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:30:32.780096Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $memoryModules = @()\n \n try {\n $modules = Get-CimInstance Win32_PhysicalMemory -ErrorAction SilentlyContinue\n \n foreach ($module in $modules) {\n $manufacturer = $module.Manufacturer\n $partNumber = $module.PartNumber\n $serialNumber = $module.SerialNumber\n $capacity = $module.Capacity\n $speed = $module.Speed\n $memoryType = $module.MemoryType\n $formFactor = $module.FormFactor\n $deviceLocator = $module.DeviceLocator\n $bankLabel = $module.BankLabel\n \n # Converter capacidade para GB\n $capacityGB = if ($capacity) { [math]::Round($capacity / 1GB, 2).ToString() + " GB" } else { "N/A" }\n \n # Converter velocidade\n $speedMHz = if ($speed) { $speed.ToString() + " MHz" } else { "N/A" }\n \n # Converter tipo de memória\n $memTypeText = switch ($memoryType) {\n 20 { "DDR" }\n 21 { "DDR2" }\n 22 { "DDR2 FB-DIMM" }\n 24 { "DDR3" }\n 26 { "DDR4" }\n 34 { "DDR5" }\n default { \n # Tentar deduzir pelo speed (heurística) se $memoryType for 0, null ou desconhecido\n if ($speed -and $speed -ge 2133) { "DDR4" }\n elseif ($speed -and $speed -ge 1066) { "DDR3" } \n elseif ($speed -and $speed -ge 533) { "DDR2" }\n elseif ($speed -and $speed -ge 200) { "DDR" }\n else { "Desconhecido" }\n }\n }\n \n # Converter fator de forma\n $formFactorText = switch ($formFactor) {\n 8 { "DIMM" }\n 12 { "SO-DIMM" }\n 13 { "Micro-DIMM" }\n default { "Form $formFactor" }\n }\n \n $memoryModules += [PSCustomObject]@{\n Manufacturer = if ($manufacturer) { $manufacturer.Trim() } else { "N/A" }\n PartNumber = if ($partNumber) { $partNumber.Trim() } else { "N/A" }\n SerialNumber = if ($serialNumber) { $serialNumber.Trim() } else { "N/A" }\n Capacity = $capacityGB\n Speed = $speedMHz\n MemoryType = $memTypeText\n FormFactor = $formFactorText\n Location = if ($deviceLocator) { $deviceLocator } else { $bankLabel }\n }\n }\n \n } catch {}\n \n $memoryModules | ConvertTo-Json -Compress\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 1.5709645748138428

RETURN_CODE: 0

OUTPUT:

[{"Manufacturer":"7F94000000000000","PartNumber":"SG564288FG8N6KF-Z1","SerialNumber":"5C00580B","Capacity":"1 GB","Speed":"800 MHz","MemoryType":"DDR2","FormFactor":"DIMM","Location":"DIMM1"},{"Manufacturer":"7F94000000000000","PartNumber":"SG564288FG8N6KF-Z1","SerialNumber":"2F00580B","Capacity":"1 GB","Speed":"800 MHz","MemoryType":"DDR2","FormFactor":"DIMM","Location":"DIMM2"},{"Manufacturer":"7F94000000000000","PartNumber":"SG564288FG8N6KF-Z1","SerialNumber":"F000C00B","Capacity":"1 GB","Speed":"800 MHz","MemoryType":"DDR2","FormFactor":"DIMM","Location":"DIMM3"},{"Manufacturer":"7F94000000000000","PartNumber":"SG564288FG8N6KF-Z1","SerialNumber":"3500C00B","Capacity":"1 GB","Speed":"800 MHz","MemoryType":"DDR2","FormFactor":"DIMM","Location":"DIMM4"}]

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:30:38.142070Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $processorInfo = @()\n \n try {\n $processors = Get-CimInstance Win32_Processor -ErrorAction SilentlyContinue\n \n foreach ($cpu in $processors) {\n $name = $cpu.Name\n $manufacturer = $cpu.Manufacturer\n $architecture = $cpu.Architecture\n $cores = $cpu.NumberOfCores\n $logicalProcessors = $cpu.NumberOfLogicalProcessors\n $maxClockSpeed = $cpu.MaxClockSpeed\n $currentClockSpeed = $cpu.CurrentClockSpeed\n $l2CacheSize = $cpu.L2CacheSize\n $l3CacheSize = $cpu.L3CacheSize\n $socket = $cpu.SocketDesignation\n \n # Converter arquitetura para texto\n $archText = switch ($architecture) {\n 0 { "x86" }\n 1 { "MIPS" }\n 2 { "Alpha" }\n 3 { "PowerPC" }\n 6 { "Intel Itanium" }\n 9 { "x64" }\n default { "Desconhecida ($architecture)" }\n }\n \n # Converter velocidades de MHz para GHz\n $maxSpeed = if ($maxClockSpeed) { [math]::Round($maxClockSpeed / 1000, 2).ToString() + " GHz" } else { "N/A" }\n $currentSpeed = if ($currentClockSpeed) { [math]::Round($currentClockSpeed / 1000, 2).ToString() + " GHz" } else { "N/A" }\n \n # Converter cache para KB/MB\n $l2Cache = if ($l2CacheSize) { \n if ($l2CacheSize -ge 1024) { [math]::Round($l2CacheSize / 1024, 2).ToString() + " MB" }\n else { $l2CacheSize.ToString() + " KB" }\n } else { "N/A" }\n \n $l3Cache = if ($l3CacheSize) { \n if ($l3CacheSize -ge 1024) { [math]::Round($l3CacheSize / 1024, 2).ToString() + " MB" }\n else { $l3CacheSize.ToString() + " KB" }\n } else { "N/A" }\n \n $processorInfo += [PSCustomObject]@{\n Name = $name\n Manufacturer = $manufacturer\n Architecture = $archText\n Cores = $cores\n LogicalProcessors = $logicalProcessors\n MaxSpeed = $maxSpeed\n CurrentSpeed = $currentSpeed\n L2Cache = $l2Cache\n L3Cache = $l3Cache\n Socket = $socket\n }\n }\n } catch {}\n \n $processorInfo | ConvertTo-Json -Compress\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 5.361061334609985

RETURN_CODE: 0

OUTPUT:

{"Name":"Intel(R) Core(TM)2 Duo CPU E7200 @ 2.53GHz","Manufacturer":"GenuineIntel","Architecture":"x64","Cores":2,"LogicalProcessors":2,"MaxSpeed":"2,53 GHz","CurrentSpeed":"2,53 GHz","L2Cache":"3 MB","L3Cache":"N/A","Socket":"Socket 775"}

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:30:43.869306Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $disks = @()\n try {\n $physicalDisks = Get-CimInstance Win32_DiskDrive -ErrorAction SilentlyContinue\n foreach ($disk in $physicalDisks) {\n $model = $disk.Model\n $size = [math]::Round($disk.Size / 1GB, 2)\n $mediaType = $disk.MediaType\n $interface = $disk.InterfaceType\n \n # Tentar determinar se é SSD ou HDD\n $diskType = "HDD"\n if ($mediaType -like "*SSD*" -or $model -like "*SSD*" -or $model -like "*Solid State*") {\n $diskType = "SSD"\n } elseif ($mediaType -like "*Fixed*" -or $interface -eq "SCSI") {\n # Usar SMART para detectar SSD (método alternativo)\n try {\n $smartData = Get-WmiObject -Namespace root\\wmi -Class MSStorageDriver_FailurePredictData -ErrorAction SilentlyContinue | Where-Object {$_.InstanceName -like "*$($disk.PNPDeviceID)*"}\n if ($smartData) {\n $diskType = "SSD"\n }\n } catch {}\n }\n \n # Obter informações de espaço das partições associadas ao disco físico\n $totalUsed = 0\n $totalFree = 0\n $partitions = ""\n \n try {\n # Método mais direto: obter partições pelo índice do disco\n $associatedPartitions = Get-CimInstance -Query "ASSOCIATORS OF {Win32_DiskDrive.DeviceID=\'$($disk.DeviceID)\'} WHERE AssocClass=Win32_DiskDriveToDiskPartition" -ErrorAction SilentlyContinue\n \n foreach ($partition in $associatedPartitions) {\n $logicalDisks = Get-CimInstance -Query "ASSOCIATORS OF {Win32_DiskPartition.DeviceID=\'$($partition.DeviceID)\'} WHERE AssocClass=Win32_LogicalDiskToDiskPartition" -ErrorAction SilentlyContinue\n \n foreach ($logicalDisk in $logicalDisks) {\n if ($logicalDisk.Size) {\n $driveSize = [math]::Round($logicalDisk.Size / 1GB, 2)\n $driveFree = [math]::Round($logicalDisk.FreeSpace / 1GB, 2)\n $driveUsed = $driveSize - $driveFree\n $totalUsed += $driveUsed\n $totalFree += $driveFree\n if ($partitions) { $partitions += ", " }\n $partitions += "$($logicalDisk.DeviceID) ($driveFree GB livre de $driveSize GB)"\n }\n }\n }\n \n # Se o método acima não funcionou, tentar método alternativo\n if ($totalFree -eq 0 -and $totalUsed -eq 0) {\n $diskPartitions = Get-CimInstance Win32_DiskPartition -ErrorAction SilentlyContinue | Where-Object { $_.DiskIndex -eq $disk.Index }\n foreach ($partition in $diskPartitions) {\n $logicalDisks = Get-CimInstance Win32_LogicalDiskToDiskPartition -ErrorAction SilentlyContinue | Where-Object { $_.Antecedent -like "*$($partition.DeviceID)*" }\n foreach ($logicalDiskRel in $logicalDisks) {\n $deviceId = ($logicalDiskRel.Dependent -split

\"'\')[1]\n $drive = Get-CimInstance Win32_LogicalDisk -ErrorAction SilentlyContinue | Where-Object { $_.DeviceID -eq $deviceId }\n if ($drive -and $drive.Size) {\n $driveSize = [math]::Round($drive.Size / 1GB, 2)\n $driveFree = [math]::Round($drive.FreeSpace / 1GB, 2)\n $driveUsed = $driveSize - $driveFree\n $totalUsed += $driveUsed\n $totalFree += $driveFree\n if ($partitions) { $partitions += ", " }\n $partitions += "$($drive.DeviceID) ($driveFree GB livre de $driveSize GB)"\n }\n }\n }\n } catch {}\n \n $espacoLivre = if ($totalFree -gt 0) { "$totalFree GB" } else { "N/A" }\n $espacoUsado = if ($totalUsed -gt 0) { "$totalUsed GB" } else { "N/A" }\n $particoesInfo = if ($partitions) { $partitions } else { "N/A" }\n \n $disks += [PSCustomObject]@{\n Modelo = $model\n Tamanho = "$size GB"\n EspacoUsado = $espacoUsado\n EspacoLivre = $espacoLivre\n Particoes = $particoesInfo\n Tipo = $diskType\n Interface = $interface\n }\n }\n } catch {}\n $disks | ConvertTo-Json -Compress\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 5.726258754730225

RETURN_CODE: 0

OUTPUT:

{"Modelo":"WDC WDS240G2G0A-00JH30 ATA Device","Tamanho":"223.57 GB","EspacoUsado":"N/A","EspacoLivre":"N/A","Particoes":"N/A","Tipo":"SSD","Interface":"IDE"}

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:30:46.220904Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $drives = @()\n \n try {\n $logicalDrives = Get-CimInstance Win32_LogicalDisk -Filter "DriveType=3" -ErrorAction SilentlyContinue\n foreach ($drive in $logicalDrives) {\n $deviceId = $drive.DeviceID\n $size = if ($drive.Size) { [math]::Round($drive.Size / 1GB, 2) } else { 0 }\n $freeSpace = if ($drive.FreeSpace) { [math]::Round($drive.FreeSpace / 1GB, 2) } else { 0 }\n $usedSpace = $size - $freeSpace\n $fileSystem = $drive.FileSystem\n $volumeName = $drive.VolumeName\n \n $drives += [PSCustomObject]@{\n Drive = $deviceId\n Size = $size\n Used = $usedSpace\n Free = $freeSpace\n FileSystem = $fileSystem\n Label = if ($volumeName) { $volumeName } else { "Sem rótulo" }\n }\n }\n \n # Ordenar por letra da unidade\n $sortedDrives = $drives | Sort-Object Drive\n \n } catch {\n $sortedDrives = @()\n }\n \n $sortedDrives | ConvertTo-Json -Compress\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 2.350855827331543

RETURN_CODE: 0

OUTPUT:

{"Drive":"C:","Size":222.01,"Used":183.35,"Free":38.66,"FileSystem":"NTFS","Label":"Sem rótulo"}

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:30:48.210412Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', "[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $s = @()\n try {\n $mons = Get-WmiObject -Namespace root\\wmi -Class WmiMonitorID -ErrorAction SilentlyContinue\n foreach ($m in $mons) {\n $arr = $m.SerialNumberID\n $manuf = ($m.ManufacturerName | ForEach-Object {[char]$_}) -join ''\n $model = ($m.UserFriendlyName | ForEach-Object {[char]$_}) -join ''\n $serial = ($arr | ForEach-Object {[char]$_}) -join ''\n $s += [PSCustomObject]@{Fabricante=$manuf; Modelo=$model; Serial=$serial}\n }\n } catch {}\n $s | ConvertTo-Json -Compress\n } -ErrorAction SilentlyContinue"]

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 1.988553524017334

RETURN_CODE: 0

OUTPUT:

[{"Fabricante":"DEL\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000","Modelo":"DELL E1 98WFP\u0000","Serial":"X763G88N1TRS\u0000\u0000\u0000\u0000\u0000"},{"Fabricante":"GSM\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000","Modelo":"LG FULL HD\u0000\u0000\u0000\u0000","Serial":"302AZRD61974\u0000\u0000\u0000\u0000\u0000"}]

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:30:51.555018Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $hasKeyboard = $false\n $hasMouse = $false\n \n try {\n $keyboards = Get-WmiObject -Class Win32_Keyboard -ErrorAction SilentlyContinue\n if ($keyboards) { $hasKeyboard = $true }\n \n $mice = Get-WmiObject -Class Win32_PointingDevice -ErrorAction SilentlyContinue\n if ($mice) { $hasMouse = $true }\n } catch {}\n \n [PSCustomObject]@{\n HasKeyboard = $hasKeyboard\n HasMouse = $hasMouse\n } | ConvertTo-Json -Compress\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 3.3439321517944336

RETURN_CODE: 0

OUTPUT:

{"HasKeyboard":true,"HasMouse":true}

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:30:53.113699Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { (Get-CimInstance Win32_BaseBoard | Select-Object Manufacturer,Product,SerialNumber | ConvertTo-Json -Compress) } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 1.5577881336212158

RETURN_CODE: 0

OUTPUT:

{"Manufacturer":"Dell Inc.","Product":"0CU409","SerialNumber":"..BR108198A60060."}

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:30:54.828961Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $networkAdapters = @()\n \n try {\n # Obter adaptadores de rede ativos\n $adapters = Get-CimInstance Win32_NetworkAdapter -Filter "NetConnectionStatus=2" -ErrorAction SilentlyContinue\n \n foreach ($adapter in $adapters) {\n $name = $adapter.Name\n $manufacturer = $adapter.Manufacturer\n $speed = "N/A"\n $macAddress = $adapter.MACAddress\n \n # Tentar obter velocidade\n if ($adapter.Speed) {\n $speedMbps = [math]::Round($adapter.Speed / 1000000, 0)\n $speed = "$speedMbps Mbps"\n }\n \n # Verificar se é adaptador físico (não virtual)\n if ($adapter.PhysicalAdapter -eq $true -or $name -notlike "*Virtual*" -and $name -notlike "*Loopback*" -and $macAddress) {\n $networkAdapters += [PSCustomObject]@{\n Name = $name\n Manufacturer = $manufacturer\n Speed = $speed\n MACAddress = $macAddress\n }\n }\n }\n \n # Se não encontrou pelo método acima, tentar método alternativo\n if ($networkAdapters.Count -eq 0) {\n $netAdapters = Get-NetAdapter -Physical -ErrorAction SilentlyContinue | Where-Object { $_.Status -eq "Up" }\n foreach ($net in $netAdapters) {\n $speed = if ($net.LinkSpeed) { \n $speedValue = $net.LinkSpeed\n if ($speedValue -ge 1000000000) {\n [math]::Round($speedValue / 1000000000, 1).ToString() + " Gbps"\n } else {\n [math]::Round($speedValue / 1000000, 0).ToString() + " Mbps"\n }\n } else { "N/A" }\n \n $networkAdapters += [PSCustomObject]@{\n Name = $net.Name\n Manufacturer = $net.DriverProvider\n Speed = $speed\n MACAddress = $net.MacAddress\n }\n }\n }\n } catch {}\n \n $networkAdapters | ConvertTo-Json -Compress\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 1.7147269248962402

RETURN_CODE: 0

OUTPUT:

{"Name":"Intel(R) 82562V-2 10/100 Network Connection","Manufacturer":"Intel","Speed":"100 Mbps","MACAddress":"00:1E:C9:1F:C7:59"}

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:30:56.357558Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $videoCards = @()\n \n try {\n $cards = Get-CimInstance Win32_VideoController -ErrorAction SilentlyContinue\n \n foreach ($card in $cards) {\n $name = $card.Name\n $manufacturer = $card.AdapterCompatibility\n $memory = "N/A"\n $driver = $card.DriverVersion\n $type = "Desconhecido"\n \n # Determinar memória de vídeo\n if ($card.AdapterRAM -and $card.AdapterRAM -gt 0) {\n $memoryGB = [math]::Round($card.AdapterRAM / 1GB, 2)\n $memory = "$memoryGB GB"\n }\n \n # Tentar determinar se é onboard ou offboard\n if ($name -like "*Intel*" -and ($name -like "*HD Graphics*" -or $name -like "*UHD Graphics*" -or $name -like "*Iris*")) {\n $type = "Onboard (Integrada)"\n } elseif ($name -like "*AMD*" -and $name -like "*Radeon*" -and ($name -like "*Vega*" -or $name -like "*APU*")) {\n $type = "Onboard (Integrada)"\n } elseif ($name -like "*NVIDIA*" -or $name -like "*AMD Radeon RX*" -or $name -like "*GeForce*" -or $name -like "*Quadro*") {\n $type = "Offboard (Dedicada)"\n } elseif ($card.PNPDeviceID -like "*PCI\\VEN_*") {\n $type = "Offboard (Dedicada)"\n } else {\n $type = "Onboard (Integrada)"\n }\n \n $videoCards += [PSCustomObject]@{\n Name = $name\n Manufacturer = $manufacturer\n Memory = $memory\n Driver = $driver\n Type = $type\n }\n }\n } catch {}\n \n $videoCards | ConvertTo-Json -Compress\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 1.52760648727417

RETURN_CODE: 0

OUTPUT:

[{"Name":"BB Capture Driver","Manufacturer":"Blueberry Consultants","Memory":"N/A","Driver":"3.40.0.0","Type":"Onboard (Integrada)"},{"Name":"NVIDIA GeForce 210 ","Manufacturer":"NVIDIA","Memory":"1 GB","Driver":"21.21.13.4201","Type":"Offboard (Dedicada)"}]

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:30:58.383813Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $o = @()\n try {\n $pr = Get-CimInstance Win32_Printer -ErrorAction SilentlyContinue\n foreach ($p in $pr) {\n $name = $p.Name\n $pnp = $p.PNPDeviceID\n $serial = ""\n $manuf = $p.Manufacturer\n $model = $p.DriverName\n if ($pnp) {\n try {\n $prop = Get-PnpDeviceProperty -InstanceId $pnp -KeyName \'DEVPKEY_Device_SerialNumber\' -ErrorAction SilentlyContinue\n if ($prop) { $serial = $prop.Data }\n } catch {}\n }\n if (-not $serial) { $serial = $pnp }\n $o += [PSCustomObject]@{Name=$name; Serial=$serial; Fabricante=$manuf; Modelo=$model}\n }\n } catch {}\n $o | ConvertTo-Json -Compress\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 2.025501012802124

RETURN_CODE: 0

OUTPUT:

[{"Name":"OneNote (Desktop)","Serial":null,"Fabricante":null,"Modelo":"Send to Microsoft OneNote 16 Driver"},{"Name":"OneNote for Windows 10","Serial":null,"Fabricante":null,"Modelo":"Microsoft Software Printer Driver"},{"Name":"Microsoft XPS Document Writer","Serial":null,"Fabricante":null,"Modelo":"Microsoft XPS Document Writer v4"},{"Name":"Microsoft Print to PDF","Serial":null,"Fabricante":null,"Modelo":"Microsoft Print To PDF"},{"Name":"Fax","Serial":null,"Fabricante":null,"Modelo":"Microsoft Shared Fax Driver"},{"Name":"AnyDesk Printer","Serial":null,"Fabricante":null,"Modelo":"AnyDesk v4 Printer Driver"}]

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:31:01.032907Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $adminUsers = @()\n \n try {\n # Obter membros do grupo de administradores\n $adminGroup = Get-LocalGroup -Name "Administradores" -ErrorAction SilentlyContinue\n if (-not $adminGroup) {\n $adminGroup = Get-LocalGroup -Name "Administrators" -ErrorAction SilentlyContinue\n }\n \n if ($adminGroup) {\n $members = Get-LocalGroupMember -Group $adminGroup -ErrorAction SilentlyContinue\n foreach ($member in $members) {\n $name = $member.Name\n $objectClass = $member.ObjectClass\n $principalSource = $member.PrincipalSource\n \n # Remover o nome do computador/domínio do nome do usuário\n if ($name -like "**") {\n $name = ($name -split "\\\\")[-1]\n }\n \n $adminUsers += $name\n }\n }\n \n # Ordenar por nome em ordem crescente e remover duplicatas\n $sortedUsers = $adminUsers | Sort-Object | Get-Unique\n \n } catch {\n # Método alternativo usando WMI se o método acima falhar\n try {\n $group = Get-WmiObject -Class Win32_Group -Filter "Name=\'Administrators\' OR Name=\'Administradores\'" -ErrorAction SilentlyContinue | Select-Object -First 1\n if ($group) {\n $members = Get-WmiObject -Class Win32_GroupUser -ErrorAction SilentlyContinue | Where-Object { $_.GroupComponent -like "*$($group.Name)*" }\n foreach ($member in $members) {\n $userPath = $member.PartComponent\n if ($userPath -match \'Name="([^"]+)"\') {\n $userName = $matches[1]\n $sortedUsers += $userName\n }\n }\n $sortedUsers = $sortedUsers | Sort-Object | Get-Unique\n }\n } catch {\n $sortedUsers = @()\n }\n }\n \n $sortedUsers | ConvertTo-Json -Compress\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031

TIMEOUT: 20

DURATION_SECONDS: 2.6482269763946533

RETURN_CODE: 0

OUTPUT:

["CEOSOFT-031\\Administrador","CEOSOFT-031\\CEOSOFTWARE","CEOSOFTWAREAD\\Admins. do domínio","CEOSOFTWAREAD\\alex","CEOSOFTWAREAD\\Alexandre.Amorim","CEOSOFTWAREAD\\Andre.Luiz","CEOSOFTWAREAD\\Angela.Joisciene","CEOSOFTWAREAD\\Angelo.Gabriel","CEOSOFTWAREAD\\Heberty.Richardy","CEOSOFTWAREAD\\Leandro.Magalhaes","CEOSOFTWAREAD\\Marco.Aurelio","CEOSOFTWAREAD\\Rafael.Freitas","CEOSOFTWAREAD\\Vinicius.Souza"]

--- CSInfo debug entry ---

TIMESTAMP_UTC: 2025-10-02T20:31:03.374874Z

COMMAND: ['powershell', '-NoProfile', '-NonInteractive', '-ExecutionPolicy', 'Bypass', '-Command', '[Console]::OutputEncoding = [System.Text.Encoding]::UTF8; Invoke-Command -ComputerName ceosoft-031 -ScriptBlock { \n $softwareList = @()\n \n try {\n $uninstallKeys = @(\n "HKLM:\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Uninstall*",\n "HKLM:\\SOFTWARE\\WOW6432Node\\Microsoft\\Windows\\CurrentVersion\\Uninstall*"\n)\n \n foreach ($key in $uninstallKeys) {\n $programs = Get-ItemProperty $key -ErrorAction SilentlyContinue\n foreach ($program in $programs) {\n $displayName = $program.DisplayName\n $version = $program.DisplayVersion\n $publisher = $program.Publisher\n \n if ($displayName -and $displayName.Trim() -ne "") {\n # Filtrar alguns itens desnecessários\n if ($displayName -notlike "*Update*" -and \n $displayName -notlike "*Hotfix*" -and\n $displayName -notlike "*Security Update*" -and\n $displayName -notlike "KB*" -and\n $displayName -ne "Microsoft Visual C++ 2019 X64 Additional Runtime" -and\n $displayName -notlike "*Redistributable*") {\n \n $softwareList += [PSCustomObject]@{\n Name = $displayName\n Version = if ($version) { $version } else { "N/A" }\n Publisher = if ($publisher) { $publisher } else { "N/A" }\n }\n }\n }\n }\n \n # Remover duplicatas e ordenar por nome\n $uniqueSoftware = $softwareList | Group-Object Name | ForEach-Object { $_.Group | Select-Object -First 1 }\n $sortedSoftware = $uniqueSoftware | Sort-Object Name\n \n } catch {\n $sortedSoftware = @()\n }\n \n $sortedSoftware | ConvertTo-Json -Compress\n } -ErrorAction SilentlyContinue']

COMPUTER: ceosoft-031
TIMEOUT: 20
DURATION_SECONDS: 2.3414037227630615
RETURN_CODE: 0
OUTPUT:

[{"Name":"Adobe Acrobat (64-bit)","Version":"25.001.20756","Publisher":"Adobe"},{"Name":"Adobe Refresh Manager","Version":"1.8.0","Publisher":"Adobe Systems Incorporated"},{"Name":"AnyDesk","Version":"ad 6.0.8","Publisher":"philandro Software GmbH"},{"Name":"Asian Language And Spelling Dictionaries Support For Adobe Acrobat Reader","Version":"23.008.20421","Publisher":"Adobe Systems Incorporated"},{"Name":"BB TestAssistant 2","Version":"N/A","Publisher":"Blueberry Software (UK) Ltd."},{"Name":"Data Dynamics ActiveReports Professional 2","Version":"2.3.0.1261","Publisher":"Data Dynamics, Ltd."},{"Name":"Driver do Microsoft OLE DB para SQL Server","Version":"18.7.4.0","Publisher":"Microsoft Corporation"},{"Name":"EFD Contribuições PVA","Version":"PVA","Publisher":"N/A"},{"Name":"ERP CEOSoftware 8_2","Version":"8.2","Publisher":"CEOSoftware Sistemas de Informática"},{"Name":"FileZilla 3.67.0","Version":"3.67.0","Publisher":"Tim Kosse"},{"Name":"Firebird 2.5.0.26074 (Win32)","Version":"2.5.0.26074","Publisher":"Firebird Project"},{"Name":"GatewayComponents","Version":"16.18.3","Publisher":"Microsoft Corporation"},{"Name":"GDR 1121 para SQL Server 2022 (KB5040936) (64-bit)","Version":"16.0.1121.4","Publisher":"Microsoft Corporation"},{"Name":"GDR 1125 para SQL Server 2022 (KB5042211) (64-bit)","Version":"16.0.1125.1","Publisher":"Microsoft Corporation"},{"Name":"GDR 1130 para SQL Server 2022 (KB5046057) (64-bit)","Version":"16.0.1130.5","Publisher":"Microsoft Corporation"},{"Name":"GDR 1135 para SQL Server 2022 (KB5046861) (64-bit)","Version":"16.0.1135.2","Publisher":"Microsoft Corporation"},{"Name":"GDR 1140 para SQL Server 2022 (KB5058712) (64-bit)","Version":"16.0.1140.6","Publisher":"Microsoft Corporation"},{"Name":"GDR 1145 para SQL Server 2022 (KB5063756) (64-bit)","Version":"16.0.1145.1","Publisher":"Microsoft Corporation"},{"Name":"GDR 1150 para SQL Server 2022 (KB5065221) (64-bit)","Version":"16.0.1150.1","Publisher":"Microsoft Corporation"},{"Name":"Git","Version":"2.51.0","Publisher":"The Git Development Community"},{"Name":"Google Chrome","Version":"141.0.7390.54","Publisher":"Google LLC"},{"Name":"Gravador VSS da Microsoft para SQL Server 2022","Version":"16.0.1000.6","Publisher":"Microsoft Corporation"},{"Name":"Integration Services","Version":"16.0.5491.7","Publisher":"Microsoft Corporation"},{"Name":"IRPF 2025 - Declaração de Ajuste Anual, Final de Espólio e Saída Definitiva do País ","Version":"1.0","Publisher":"Receita Federal do Brasil"},{"Name":"Kaspersky","Version":"21.22.7.466","Publisher":"Kaspersky"},{"Name":"KPL.ONCLICK","Version":"4.0.9","Publisher":"KPL ONCLICK"},{"Name":"Microsoft Edge","Version":"140.0.3485.94","Publisher":"Microsoft Corporation"},{"Name":"Microsoft Edge WebView2 Runtime","Version":"140.0.3485.94","Publisher":"Microsoft Corporation"},{"Name":"Microsoft Fabric Integration Runtime Preview","Version":"5.54.9277.1","Publisher":"Microsoft Corporation"},{"Name":"Microsoft Help Viewer 2.3","Version":"2.3.28307","Publisher":"Microsoft Corporation"},{"Name":"Microsoft MPI (10.1.12498.18)","Version":"10.1.12498.18","Publisher":"Microsoft Corporation"},{"Name":"Microsoft ODBC Driver 17 for SQL Server","Version":"17.10.6.1","Publisher":"Microsoft Corporation"},{"Name":"Microsoft Office Access database engine 2007 (English)","Version":"12.0.6612.1000","Publisher":"Microsoft Corporation"},{"Name":"Microsoft Office Professional Plus 2019 - pt-br","Version":"16.0.19231.20156","Publisher":"Microsoft Corporation"},{"Name":"Microsoft Power BI Desktop (x64)","Version":"2.145.1262.0","Publisher":"Microsoft Corporation"},{"Name":"Microsoft PowerBI Desktop (x64)","Version":"2.145.1262.0","Publisher":"Microsoft Corporation"},{"Name":"Microsoft Project - pt-br","Version":"16.0.19231.20156","Publisher":"Microsoft Corporation"},{"Name":"Microsoft SQL Server 2012 Native Client ","Version":"11.4.7001.0","Publisher":"Microsoft Corporation"},{"Name":"Microsoft SQL Server 2019 Setup (English)","Version":"15.0.4013.40","Publisher":"Microsoft Corporation"},{"Name":"Microsoft SQL Server 2022 (64-bit)","Version":"N/A","Publisher":"N/A"},{"Name":"Microsoft SQL Server 2022 RsFx Driver","Version":"16.0.1000.6","Publisher":"Microsoft Corporation"},{"Name":"Microsoft SQL Server 2022 Setup (English)","Version":"16.0.1150.1","Publisher":"Microsoft Corporation"},{"Name":"Microsoft SQL Server Management Studio - 20.2","Version":"20.2.30.0","Publisher":"Microsoft Corporation"},{"Name":"Microsoft Teams Meeting Add-in for Microsoft Office","Version":"1.25.04401","Publisher":"Microsoft"},{"Name":"Microsoft Visual C++ 2013 x86 Additional Runtime - 12.0.40664","Version":"12.0.40664","Publisher":"Microsoft Corporation"},{"Name":"Microsoft Visual C++ 2013 x86 Minimum Runtime - 12.0.40664","Version":"12.0.40664","Publisher":"Microsoft Corporation"},{"Name":"Microsoft Visual C++ 2022 X64 Additional Runtime - 14.38.33135","Version":"14.38.33135","Publisher":"Microsoft Corporation"},{"Name":"Microsoft Visual C++ 2022 X64 Minimum Runtime - 14.38.33135","Version":"14.38.33135","Publisher":"Microsoft Corporation"},{"Name":"Microsoft Visual C++ 2022 X86 Additional Runtime - 14.38.33135","Version":"14.38.33135","Publisher":"Microsoft Corporation"},{"Name":"Microsoft Visual C++ 2022 X86 Minimum Runtime - 14.38.33135","Version":"14.38.33135","Publisher":"Microsoft Corporation"},{"Name":"Microsoft Visual Studio Tools for Applications 2019","Version":"16.0.31110","Publisher":"Microsoft Corporation"},{"Name":"Microsoft Visual Studio Tools for Applications 2019 x64 Hosting Support","Version":"16.0.31110","Publisher":"Microsoft Corporation"},{"Name":"Microsoft Visual Studio Tools for Applications 2019 x86 Hosting Support","Version":"16.0.31110","Publisher":"Microsoft Corporation"},{"Name":"MSI to redistribute MS Visual Studio Runtime libraries for x64","Version":"14.37.32822","Publisher":"The Firebird Project"},{"Name":"MSI to redistribute MS Visual Studio Runtime libraries for x86","Version":"14.37.32822","Publisher":"The Firebird Project"},{"Name":"MSI to redistribute MS VS2005 CRT libraries","Version":"8.0.50727.42","Publisher":"The Firebird Project"},{"Name":"MSI to redistribute MS VS2010 CRT libraries","Version":"10.0.30319.1","Publisher":"The Firebird Project"},{"Name":"Navegador para SQL Server 2022","Version":"16.0.1000.6","Publisher":"Microsoft Corporation"},{"Name":"Notepad++ (64-bit x64)","Version":"8.7","Publisher":"Notepad++ Team"},{"Name":"NVIDIA Install Application","Version":"2.1002.172.1382","Publisher":"NVIDIA Corporation"},{"Name":"Office 16 Click-to-Run Extensibility Component","Version":"16.0.19231.20072","Publisher":"Microsoft Corporation"},{"Name":"Office 16 Click-to-Run Extensibility

Component 64-bit Registration","Version":"16.0.19231.20072","Publisher":"Microsoft Corporation"},{"Name":"Office 16 Click-to-Run Licensing Component","Version":"16.0.19231.20138","Publisher":"Microsoft Corporation"},{"Name":"Office 16 Click-to-Run Localization Component","Version":"16.0.19231.20072","Publisher":"Microsoft Corporation"},{"Name":"On-premises data gateway","Version":"3000.274.3","Publisher":"Microsoft Corporation"},{"Name":"OpenSSL 3.4.1 (64-bit)","Version":"3.4.1","Publisher":"OpenSSL Win64 Installer Team"},{"Name":"OpenVPN 2.6.11-I002 amd64","Version":"2.6.1102","Publisher":"OpenVPN, Inc."},{"Name":"OpenVPN Connect","Version":"3.5.1","Publisher":"OpenVPN Inc."},{"Name":"Pacote de Idiomas do Microsoft Help Viewer 2.3 – PTB","Version":"2.3.28107","Publisher":"Microsoft Corporation"},{"Name":"Pacote de Idiomas do SQL Server Management Studio - Português","Version":"20.2.30.0","Publisher":"Microsoft Corp."},{"Name":"Painel de controle da NVIDIA 342.01","Version":"342.01","Publisher":"NVIDIA Corporation"},{"Name":"Provedor Microsoft Analysis Services OLE DB","Version":"16.0.5143.0","Publisher":"Microsoft Corporation"},{"Name":"RustDesk","Version":"1.2.7","Publisher":"RustDesk"},{"Name":"Shell isolado do Visual Studio 2017 para o pacote de idiomas do SSMS – Português (Brasil)","Version":"15.0.28307.421","Publisher":"Microsoft Corporation"},{"Name":"Sped Contábil 10.3.3","Version":"10.3.3","Publisher":"Receita Federal do Brasil"},{"Name":"Sped Fiscal 2.1.1-SNAPSHOT","Version":"2.1.1-SNAPSHOT","Publisher":"N/A"},{"Name":"SQL Lite","Version":"1.27.3","Publisher":"Friendship Solutions"},{"Name":"SQL Server 2019 Serviço de Linguagem T-SQL da Microsoft","Version":"15.0.2000.5","Publisher":"Microsoft Corporation"},{"Name":"SQL Server 2022 Advanced Analytics","Version":"16.0.1000.6","Publisher":"Microsoft Corporation"},{"Name":"SQL Server 2022 Batch Parser","Version":"16.0.1000.6","Publisher":"Microsoft Corporation"},{"Name":"SQL Server 2022 Common Files","Version":"16.0.1000.6","Publisher":"Microsoft Corporation"},{"Name":"SQL Server 2022 Connection Info","Version":"16.0.1000.6","Publisher":"Microsoft Corporation"},{"Name":"SQL Server 2022 Database Engine Services","Version":"16.0.1000.6","Publisher":"Microsoft Corporation"},{"Name":"SQL Server 2022 Database Engine Shared","Version":"16.0.1000.6","Publisher":"Microsoft Corporation"},{"Name":"SQL Server 2022 DMF","Version":"16.0.1000.6","Publisher":"Microsoft Corporation"},{"Name":"SQL Server 2022 Full text search","Version":"16.0.1000.6","Publisher":"Microsoft Corporation"},{"Name":"SQL Server 2022 Shared Management Objects","Version":"16.0.1000.6","Publisher":"Microsoft Corporation"},{"Name":"SQL Server 2022 Shared Management Objects Extensions","Version":"16.0.1000.6","Publisher":"Microsoft Corporation"},{"Name":"SQL Server 2022 SQL Diagnostics","Version":"16.0.1000.6","Publisher":"Microsoft Corporation"},{"Name":"SQL Server 2022 XEvent","Version":"16.0.1000.6","Publisher":"Microsoft Corporation"},{"Name":"SQL Server Management Studio","Version":"20.2.30.0","Publisher":"Microsoft Corp."},{"Name":"SSMS Post Install Tasks","Version":"20.2.30.0","Publisher":"Microsoft Corporation"},{"Name":"TeamViewer","Version":"15.68.5","Publisher":"TeamViewer"},{"Name":"TortoiseSVN 1.14.8.29723 (64 bit)","Version":"1.14.29723","Publisher":"TortoiseSVN"},{"Name":"Visual Studio 2017 Isolated Shell for SSMS","Version":"15.0.28307.421","Publisher":"Microsoft Corporation"},{"Name":"WinRAR 7.13 (64-bit)","Version":"7.13.0","Publisher":"win.rar GmbH"},{"Name":"XAMPP","Version":"8.2.12-0","Publisher":"Apache Friends"}]