

Guia de instalação dos ambientes de desenvolvimento do SIRERC

https://github.com/sirercita/SIRERC_ProjetoPetrobrasdocs/

Tabela de Revisões

Revisão	Data	Autor(es)	Descrição
0.1.0	28/08/2023	André F. M. Caetano	Primeira versão.
0.2.0	04/09/2023	André F. M. Caetano	Nova seção para instalador.
0.3.0	25/09/2023	André F. M. Caetano	Nova seção <i>Gmsh</i> , atualização de versões.
0.4.0	27/04/2024	Nicolas F. C. Granese	Nova seção WSL, melhorias na seção compilação do código.
0.5.0	02/09/2024	A. Celio P. Mesquita	Atualização da instalação do ambiente de desenvolvimento

Resumo

O *front-end* de um software de simulação deve prover uma interface de usuário intuitiva e organizada que facilite a definição dos cenários a serem simulados.

SIRERC é uma interface humano-máquina (*HMI - Human-Machine Interface*) para definição de cenários em dinâmica de fluidos computacional (*Computational Fluid Dynamics* — *CFD*).

Este é o guia de instalação dos ambientes de desenvolvimentos do SIRERC.

Sumário

1	INSTALAÇÃO DO AMBIENTE WINDOWS	4
1.1	GitHub e LaTeX	4
1.2	Instalação do <i>WSL</i>	4
1.3	Instalação dos requisitos de desenvolvimento	5
1.3.1	Instalando o ambiente <i>ActiveState</i>	5
1.3.2	Instalação do compilador C++	7
1.3.3	Instalação do <i>CMake</i>	7
1.3.4	Instalando do <i>Git</i>	7
1.4	<i>Download</i> e instalação da última versão <i>Opensource</i> do <i>Qt Creator</i>	8
1.5	Instalação do <i>MongoDB</i>	12
1.5.1	Drivers de C++ para <i>MongoDB</i>	13
1.5.2	Configurar o driver	13
1.5.3	Compilar o driver	14
1.5.4	Instalar o driver	14
1.5.5	Criação de objetos no <i>MongoDB</i>	14
1.6	Instalação do VTK	16
1.6.1	Configurar o driver	16
1.6.2	Compilar o driver	16
1.6.3	Instalar o driver	17
1.6.4	Variáveis de Ambiente do VTK	17
1.7	Instalação do <i>Gmsh</i>	18
1.7.1	Configurar o driver	18
1.7.2	Compilar o driver	19
1.7.3	Instalar o driver	19
1.7.4	Variáveis de Ambiente do <i>Gmsh</i>	19
1.8	Compilação do SIRERC	20
1.9	Criação do Instalador	23

1 INSTALAÇÃO DO AMBIENTE WINDOWS

O SIRERC é uma aplicação *desktop* baseada na versão *open-source* do *framework Qt*, escolhido pela sua portabilidade, capacidade de gerar aplicações multiplataforma para sistemas MS *Windows*, *Linux* e *MacOS*. A linguagem de programação utilizada é C++.

Este manual foi atualizado por meio das instalações em duas máquina com *Windows 11 Pro* com versões diferentes.

1.1 GitHub e LaTeX

1. **GitHub Desktop**

- Baixe e instale o *GitHub Desktop* do endereço: <https://desktop.github.com/download/>
- Ao abrir a tela de instalação entre com o sua conta do *GitHub*.
- Clone ou atualize o projeto de interesse.

2. **LaTeX** [apenas para quem estiver encarregado de atualizar este manual.](https://www.texstudio.org/)

Baixe e instale um editor de *LaTeX*. Pode ser este <https://www.texstudio.org/> ou outro da sua preferência.

1.2 Instalação do WSL

O *WSL* (*Windows Subsystem for Linux*) permite que desenvolvedores executem um ambiente *GNU/Linux* diretamente no *Windows*, sem a necessidade de uma máquina virtual ou *dual-boot*.

Utilizar o *WSL* facilita o seguimento de processos e ferramentas já usados por colegas que desenvolvem em sistemas *Linux*, tornando a colaboração em projetos mais integrada e eficiente.

Na presente data o SIRERC depende do *WSL* para funcionar, já que o módulo *backend* executa em *Linux*. Isto não é automatizado pelo instalador do programa, e deve ser feito pelo usuário antecipadamente.

1. **Instalação do WSL:**

- Abra o *PowerShell* ou o *Prompt de Comando* do *Windows* no modo administrador clicando com o botão direito do mouse e selecionando "Executar como administrador".
- Insira o comando `wsl --install` e pressione `[Enter]`. Este comando instalará o *WSL* e a distribuição padrão do *Linux*. Após a instalação, reinicie o computador.

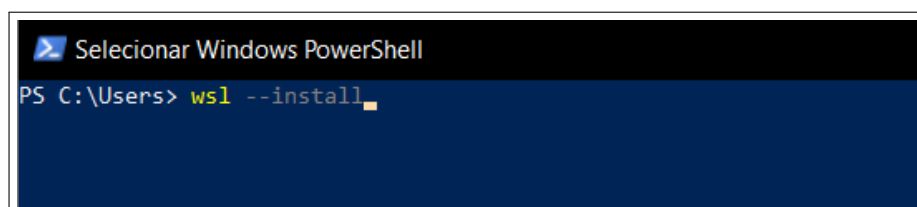


Figura 1: Comando para instalar o *WSL* no *Windows*

2. **Verificação da Instalação:**

- Após reiniciar o computador, abra novamente o *PowerShell* ou o *Prompt de Comando* e insira o comando `wsl` para verificar se o *WSL* foi instalado corretamente.
- Se o *WSL* estiver instalado corretamente, um prompt de linha de comando do *Linux* será exibido. Você pode usar esse ambiente para executar comandos do *Linux* e interagir com o sistema de arquivos do *Windows*.

1.3 Instalação dos requisitos de desenvolvimento

Primeiramente, será preciso instalar as versões corretas dos *drivers* do C++. Assim, quando se for instalar o *Qt Creator* e o *Qt 5.15.2*, as bibliotecas necessárias já estarão instaladas para permitir a configuração exata do projeto.

Há ainda, algumas linguagens de programação e ferramentas que serão utilizadas direta ou indiretamente nas compilações e no desenvolvimento do SIRERC.

- Perl
- Python
- Ruby
- Compilador C++ com suporte ao padrão C++11 e compatível com o *Qt 5.15.x*
- CMake
- Git

1.3.1 Instalando o ambiente *ActiveState*

ActiveState é uma empresa de software que fornece distribuições de linguagens de programação *open source*, como *Python*, *Perl*, *Ruby* e *Tcl*. Essas distribuições são pré-configuradas e testadas para garantir compatibilidade e desempenho em diferentes sistemas operacionais.

Em resumo, *ActiveState* é uma solução que simplifica o processo de desenvolvimento e implantação de aplicativos usando linguagens de programação *open source*.

Crie uma conta, uma organização e um projeto para cada linguagem de programação no *ActiveState*. A conta poderá ser o seu e-mail e a organização o seu nome.

Ao criar um projeto aparecerá esta tela para instalação do ambiente do *ActiveState*. Faça o download e a instalação.

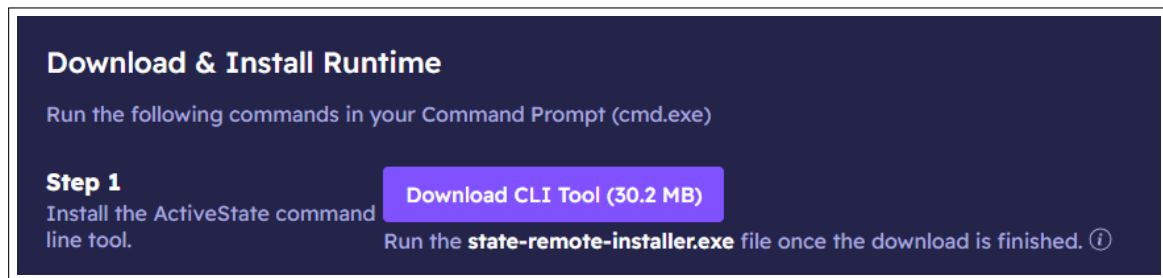


Figura 2: Instalação do ambiente *ActiveState*

Crie uma pasta para cada linguagem. Exemplo: *activeStatePython*, *activeStatePerl* e *activeStateRuby*.

Em cada uma dessas pastas digite os comandos copiados do site *ActiveState* (mantenha o ponto no final):

```
state checkout <sua organização no ActiveSate>/<linguagem e versão> .
state use <linguagem e versão>
```

```

C:\Users\Particular>cd activeStatePerl

C:\Users\Particular\activeStatePerl>state checkout celiomesquita-org/Perl-5.38.2-Windows .
Checking out project: celiomesquita-org/Perl-5.38.2-Windows

Resolving Dependencies ✓Done
Setting up the following dependencies:
└─ Utilities@1.00 (93 sub-dependencies)
└─ perl@5.38.2

■ Installing Runtime

Installing your runtime and dependencies.

Downloading                      95/95
Installing                      95/95
✓All dependencies have been installed and verified

```

Figura 3: Perl instalado com sucesso

```

C:\Users\Particular>mkdir activeStateRuby

C:\Users\Particular>cd activeStatePython

C:\Users\Particular\activeStatePython>state checkout celiomesquita-org/Python-3.11.9-Windows .
Checking out project: celiomesquita-org/Python-3.11.9-Windows

Resolving Dependencies ✓Done
Setting up the following dependencies:
└─ numpy@2.0.1
└─ python@3.11.9

■ Installing Runtime

Installing your runtime and dependencies.

Downloading                      9/9
Installing                      9/9
✓All dependencies have been installed and verified

```

Figura 4: Python instalado com sucesso.

```

C:\Users\Particular\activeStatePerl>cd ..

C:\Users\Particular>cd activeStateRuby

C:\Users\Particular\activeStateRuby>state checkout celiomesquita-org/Ruby-3.3.1-Windows .
Checking out project: celiomesquita-org/Ruby-3.3.1-Windows

Resolving Dependencies ✓Done
Setting up the following dependencies:
└─ ruby@3.3.1

■ Installing Runtime

Installing your runtime and dependencies.

Downloading                      8/8
Installing                      8/8
✓All dependencies have been installed and verified

```

Figura 5: Ruby instalado com sucesso.

1.3.2 Instalação do compilador C++

Faça o download dos drivers do Microsoft Visual Studio C++ (Visual C++ Redistributable for Visual Studio 2019) do seguinte endereço:

<https://learn.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-160>

Opte pela versão: https://aka.ms/vs/17/release/vc_redist.x64.exe

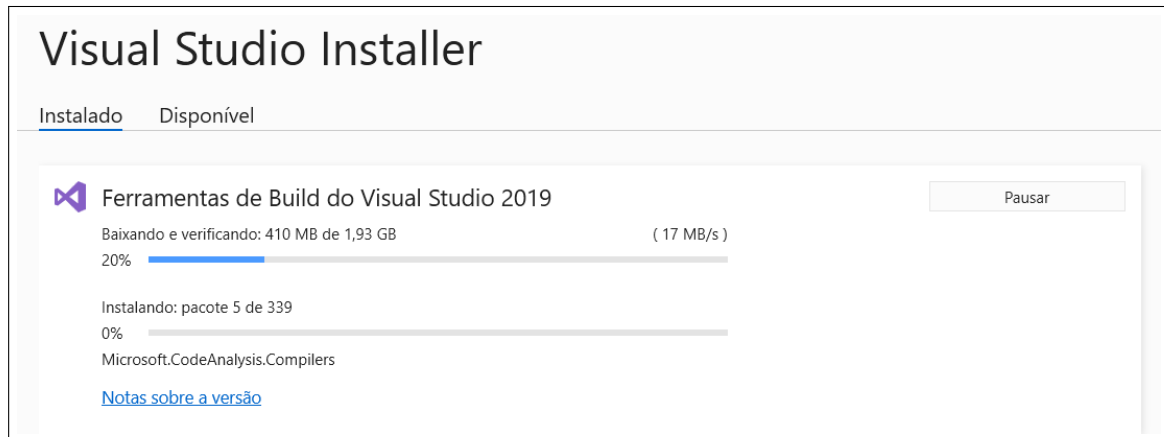


Figura 6: Visual Studio 2019 versão 16.11.39

A partir deste ponto, todas as compilações serão baseadas em Visual Studio 16 2019.

1.3.3 Instalação do CMake

CMake é um sistema para automatizar tarefas de compilação de arquivos de código criados em C e C++.

1. Baixe e instale o CMake por meio do endereço: <https://cmake.org/download/>.
2. Instale a versão *Windows x64 Installer*.
3. Marque a opção: *Add CMake to the PATH environment variable*.

1.3.4 Instalando do Git

Acesse o site oficial do Git: <https://git-scm.com/download/win>

Selecione esta versão: *64-bit Git for Windows Setup*.

Ao instalar, não se preocupe com a experiência para decidir, utilize todas as opções oferecidas pelo instalador.

Se o Git já estiver instalado, mas ainda não for reconhecido, talvez o caminho (PATH) não tenha sido configurado corretamente.

Adicione o Git ao PATH

1. Pressione Win + Pause/Break (Win + I, se for em um notebook) para abrir as Propriedades do Sistema.
2. Clique em Configurações avançadas do sistema.
3. Na aba Avançado, clique em Variáveis de Ambiente.
4. Na seção Variáveis do sistema, localize a variável chamada Path e clique em Editar.

5. Clique em Novo e adicione o caminho completo para o diretório bin do Git, por exemplo:

C:\Program Files\Git\bin

6. Clique em OK em todas as janelas para salvar as mudanças.

Importante ressaltar que, até agora, estivemos instalando somente a infraestrutura. A seguir virão as ferramentas de desenvolvimento.

1.4 *Download* e instalação da última versão *Opensource* do *Qt Creator*

1. Acesse a página de *Download* do *Qt* em: <https://www.qt.io/download-open-source>
2. Role a página até atingir a parte de baixo, onde deve visualizar o botão de *Download*.
3. Clique no botão para iniciar o *Download* do instalador do *Qt* (ver Figura 7).

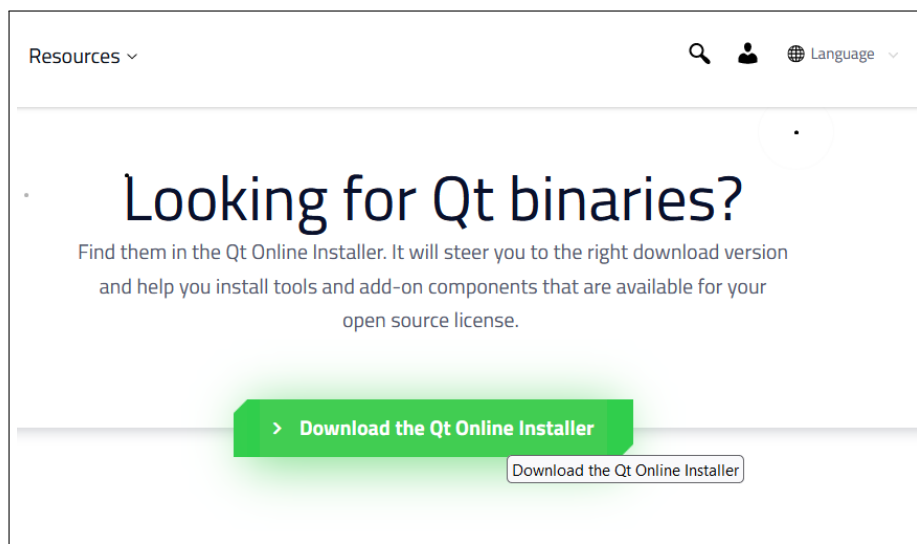


Figura 7: *Download* do instalador do *Qt*

4. Também será possível escolher o *mirror*, local de origem do instalador.

```
.\qt-online-installer-windows-x64-4.8.0.exe --mirror https://mirrors.ocf.berkeley.edu/qt/
```

5. Na próxima tela, escolha a opção do instalador para sistemas operacionais *Windows* e clique no botão para *Download* (vide Figura 8).



Figura 8: Escolha do instalador por sistema operacional

- Na instalação online, a próxima tela oferece a possibilidade de registro de e-mail no ambiente do *Qt*, conforme a Figura 9

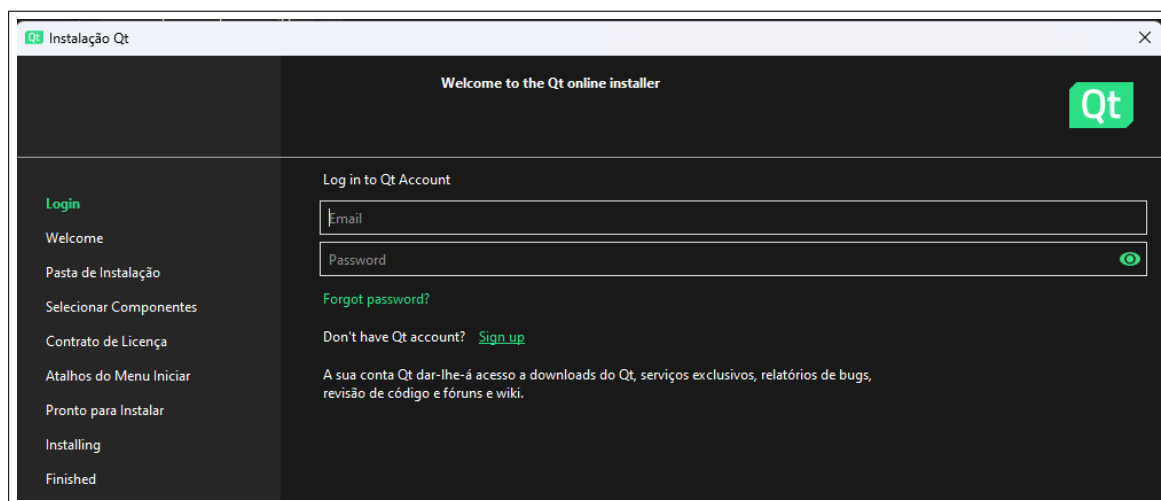


Figura 9: Digite seus dados a fim de continuar o processo de download do *Qt*.

- Após o registro, será oferecida a possibilidade de concordar com os termos. Marque que concorda e clique em **Próximo**.

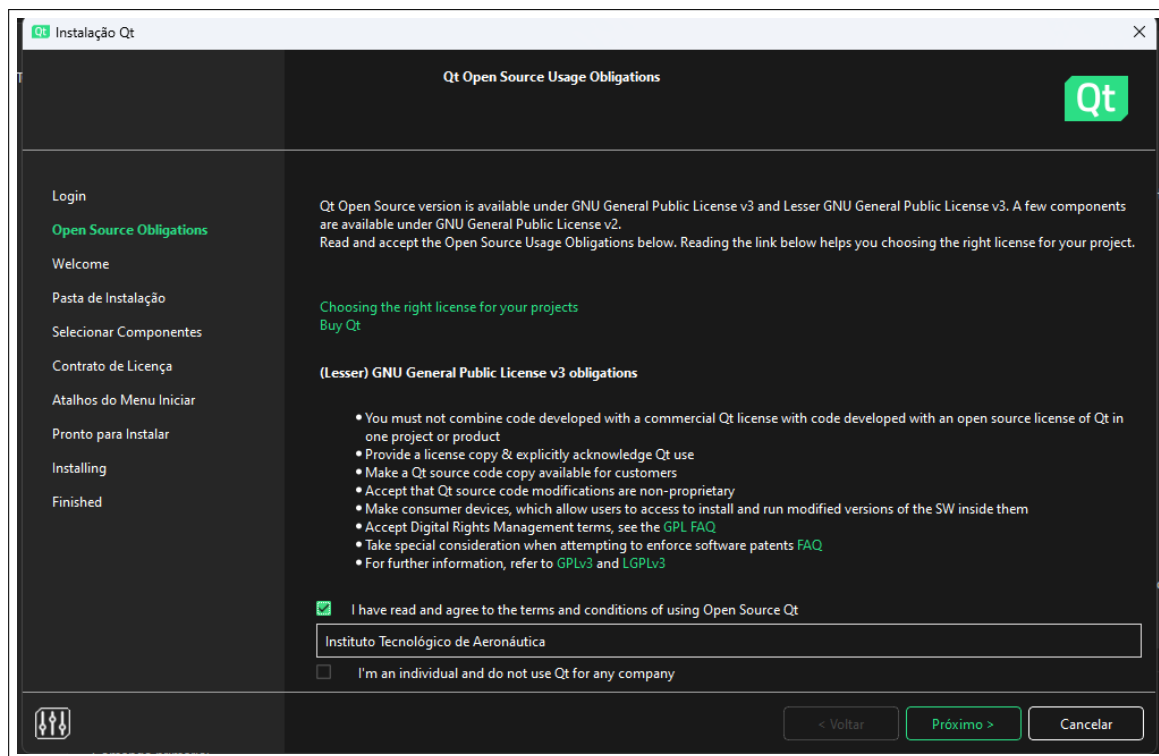


Figura 10: Marque que concorda e clique em **Próximo**.

8. Em seguida, será oferecida a possibilidade de escolher o caminho para instalação (Figura 11).

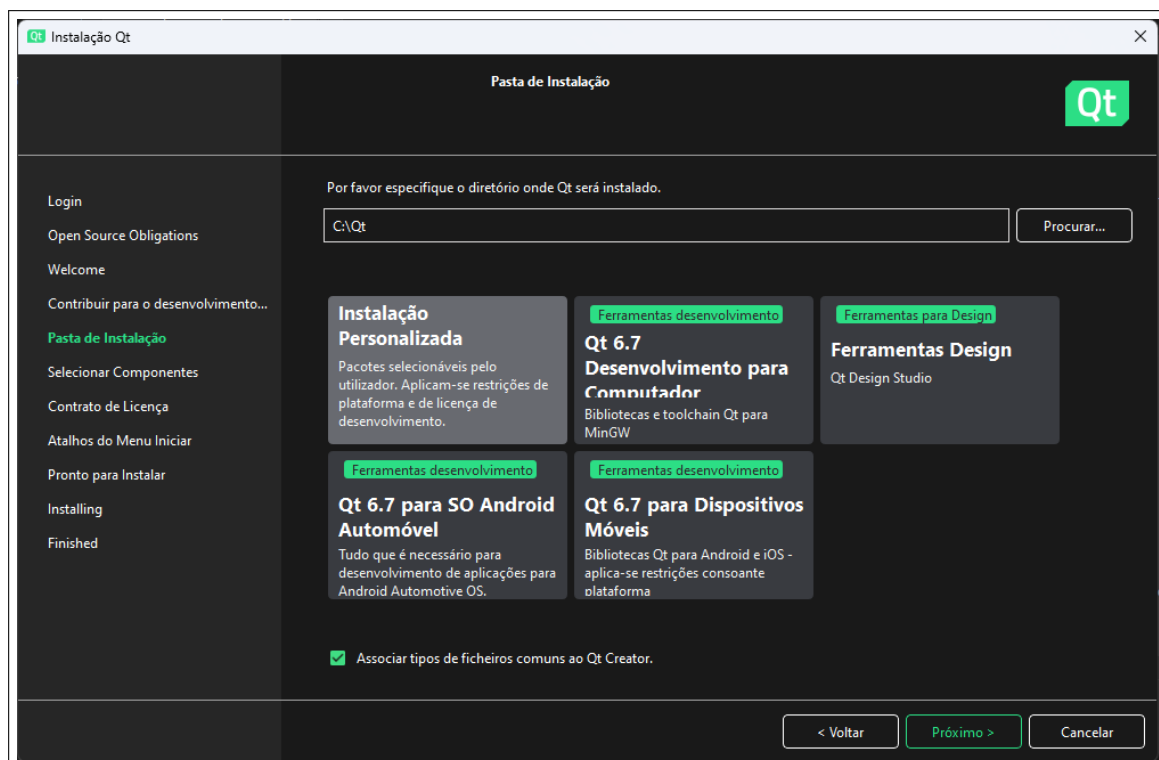


Figura 11: Escolha o caminho e clique em **Próximo**.

9. Em seguida, será oferecida a possibilidade de escolher os componentes da instalação (Figura 12). Caso não escolha todos os componentes, poderá instalá-los posteriormente.

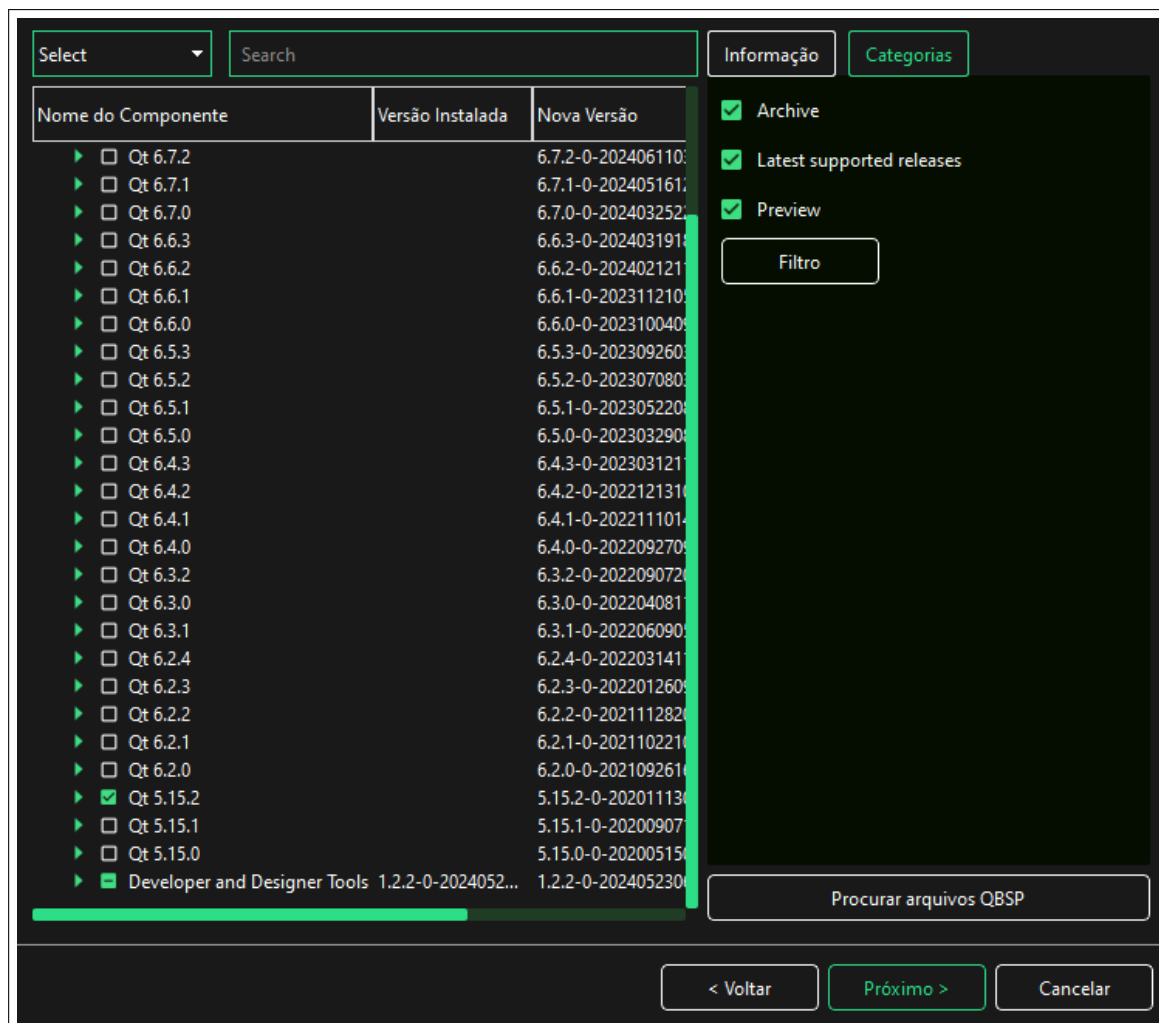


Figura 12: Marque *Archive*, clique em **Filtro**, marque *Qt 5.15.2* e clique em **Próximo**.

- Em seguida, será oferecida a possibilidade de marcar que concorda com os termos do *CMake license agreement* (Figura 13). Marque e continue com o processo de instalação.

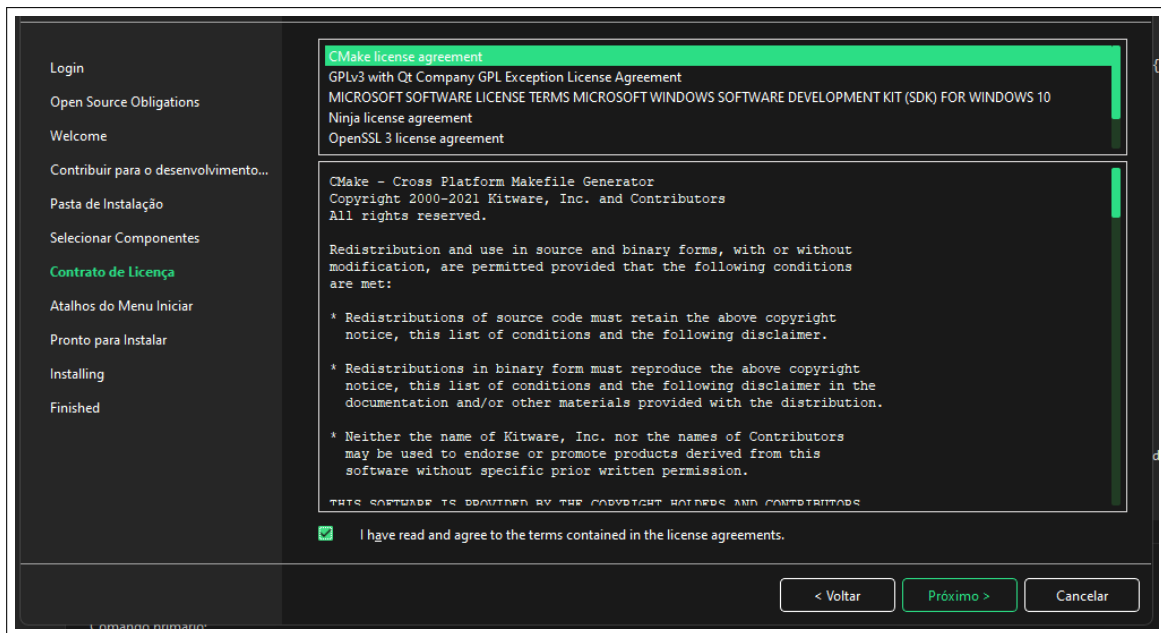


Figura 13: Marque que concorda e clique em **Próximo**.

11. Aguarde a conclusão da instalação (Figura 14).

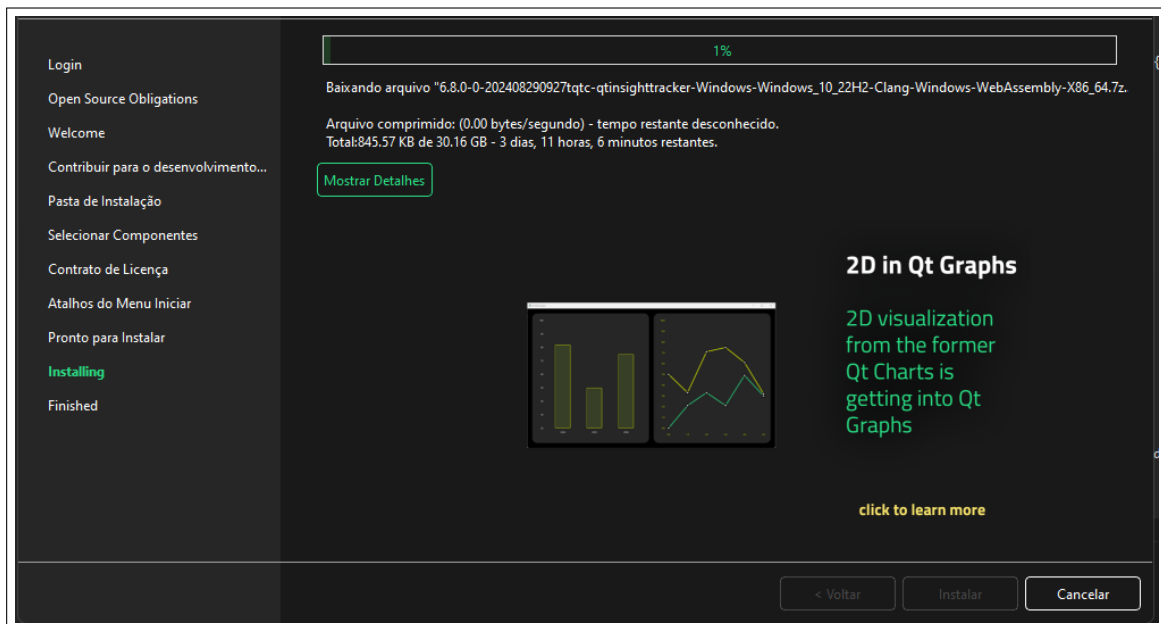


Figura 14: Andamento da instalação do *Qt Creator* e do *Qt 5.15.2*.

Esta instalação precisará fazer o download de muitos arquivos e a sua instalação. Portanto será bastante demorada.

1.5 Instalação do *MongoDB*

I "Sub judice"

Com a finalidade de controlar os projetos dos usuários do SIRERC, poderemos utilizar o banco de dados não relacional *MongoDB*, que armazena as informações no formato de objetos *json*.

Isto tem potencial de contribuir para a segurança dos dados industriais.

Baixe e instale o *MongoDB* a partir do endereço: <https://www.mongodb.com/try/download/community>.

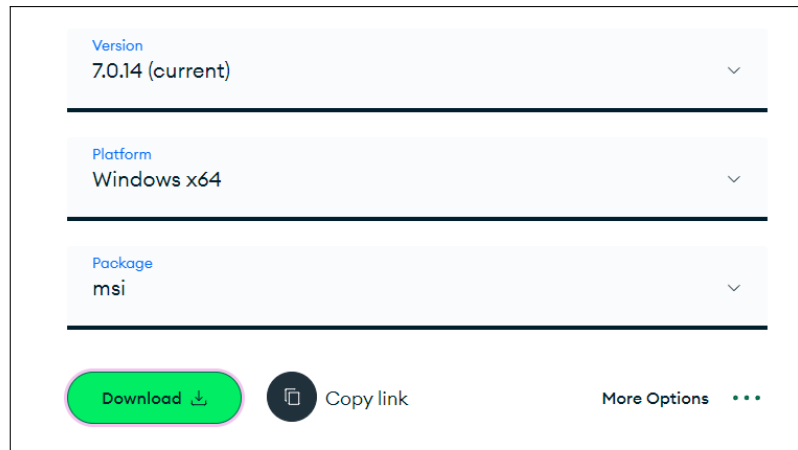


Figura 15: Clique em *Download*.

Clique na opção *Complete* e continue a instalação.

Agora com o *MongoDB* instalado, vamos instalar os drivers de C++ para *MongoDB*, a serem utilizados pelo *Qt Creator*.

1.5.1 Drivers de C++ para *MongoDB*

Baixe a versão mais recente do driver **mongocxx**. O ponto de partida mais confiável para criar o driver **mongocxx** é o *tarball* (arquivo compactado) da versão mais recente.

Os lançamentos do **mongocxx** terá links para o *tarball* de versão da versão que você deseja instalar.

Comandos deste manual grafados em mais de uma linha, precisam ser editados para somente uma linha, antes de colar no terminal.

Baixe e descompacte do endereço: <https://github.com/mongodb/mongo-cxx-driver/releases/tag/r3.10.2>

Abra o *Developer Command Prompt*.

```
cd C:/Users/<seu usuário>/Downloads/
```

1.5.2 Configurar o driver

```
cd mongo-cxx-driver-r3.10.2/mongo-cxx-driver-r3.10.2
cmake -G "Visual Studio 16 2019" -A x64 -DCMAKE_CXX_STANDARD=17
-DCMAKE_BUILD_TYPE=Release
```

Caso o comando **cmake** não seja reconhecido, coloque este endereço no **path** `C:/Program Files/CMake/bin` e repita os comandos acima.

1.5.3 Compilar o driver

Agora que os arquivos de build foram gerados, você pode iniciar a compilação do driver. No Prompt de Comando do Desenvolvedor do Visual Studio ou no terminal que você está usando, execute o seguinte comando para compilar o projeto:

```
cmake --build . --config Release
```

Esse comando compilará o driver no modo Release. Como pode ser bastante demorado, você pode, antes de iniciar, tentar conseguir uma versão já compilada de algum colega.

1.5.4 Instalar o driver

Crie a pasta **Drivers** no seu computador, se não existir.

Execute o seguinte comando para instalar o driver:

```
cmake --install . --config Release --prefix "C:/Drivers/mongo-cxx-driver"
```

Agora os arquivos devem estar corretamente instalados no diretório **C:/Drivers/mongo-cxx-driver**. A partir daqui, você pode usar o **mongo-cxx-driver** em seus projetos C++ vinculando as bibliotecas e incluindo os cabeçalhos instalados.

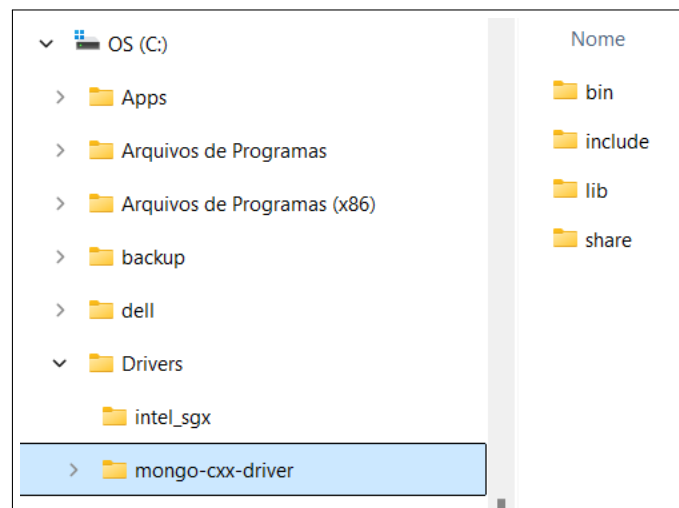


Figura 16: Drivers de C++ para *MongoDB* instalados com sucesso.

1.5.5 Criação de objetos no *MongoDB*

O *MongoDB* vem com uma ferramenta para administração de objetos **json**, o *Compass*.

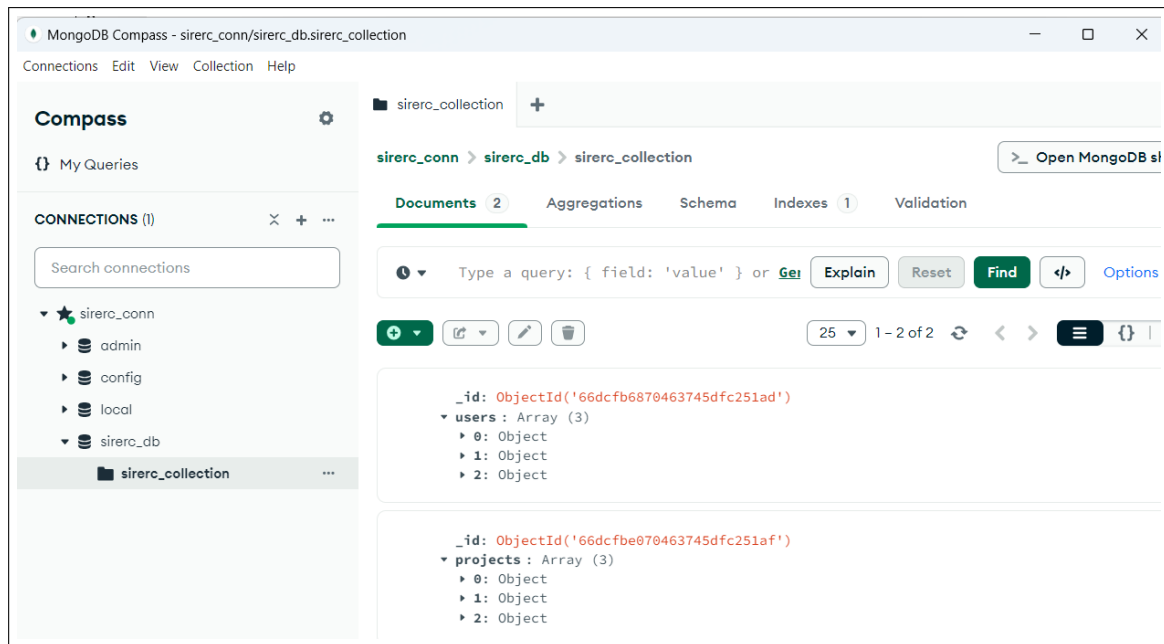


Figura 17: *MongoDB Compass*

Na Figura 17, pode-se observar que foram criados os objetos:

1. `sirerc_conn`
2. `sirerc_db`
3. `sirerc_collection`

No `sirerc_collection`, criamos duas coleções: *users* e *projects*.

A coleção *users* será utilizada para associar usuários e seus projetos. A coleção *projects* será utilizada para registrar os projetos e suas informações.

1.6 Instalação do VTK

O *Visualization Toolkit* (VTK) é um software multiplataforma utilizado para computação gráfica 3D, visualização e processamento de imagens. No SIRERC, o VTK é usado para visualizar os resultados gráficos após a execução do simulador, relacionados ao problema de dinâmica de fluidos.

1. Baixe o código-fonte da versão 9.3.0 RC1 do VTK, atualmente em uso no SIRERC, no seguinte endereço: <https://gitlab.kitware.com/vtk/vtk/-/archive/v9.3.0.rc1/vtk-v9.3.0.rc1.zip>.
2. Descompacte no diretório Downloads.
 - (a) Caso deseje, você pode descompactar com o botão direito do mouse sobre o arquivo **.zip** ou **.tar.gz**.
 - (b) Também é possível, em um terminal do *Windows*, digitar WSL, acessando o terminal do *Linux*, e colar este comando: `tar -xvf vtk-v9.3.0.rc1.zip`.
3. Abra o *Developer Command Prompt*.

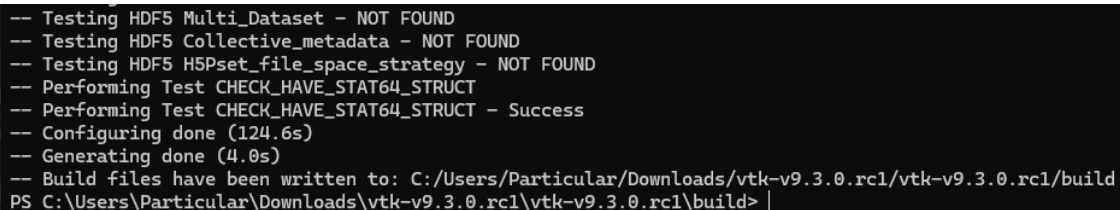
1.6.1 Configurar o driver

Na extração, serão criados dois diretórios: `vtk-v9.3.0.rc1/vtk-v9.3.0.rc`

```
cd Downloads/vtk-v9.3.0.rc1/vtk-v9.3.0.rc1
mkdir build
cd build

cmake -G "Visual Studio 16 2019" -A x64 -DCMAKE_CXX_STANDARD=17
-DCMAKE_BUILD_TYPE=Release ..
```

Os dois pontos no final fazem com que o comando se refira ao diretório pai. Caso o comando **cmake** não seja reconhecido, coloque este endereço no **path** `C:/Program Files/CMake/bin` e repita o *CMake* acima.



```
-- Testing HDF5 Multi_Dataset - NOT FOUND
-- Testing HDF5 Collective_metadata - NOT FOUND
-- Testing HDF5 H5Pset_file_space_strategy - NOT FOUND
-- Performing Test CHECK_HAVE_STAT64_STRUCT
-- Performing Test CHECK_HAVE_STAT64_STRUCT - Success
-- Configuring done (124.6s)
-- Generating done (4.0s)
-- Build files have been written to: C:/Users/Particular/Downloads/vtk-v9.3.0.rc1/vtk-v9.3.0.rc1/build
PS C:\Users\Particular\Downloads\vtk-v9.3.0.rc1\vtk-v9.3.0.rc1\build> |
```

Figura 18: VTK drivers configurados para o Visual Studio 16 2019 com sucesso.

1.6.2 Compilar o driver

Agora que os arquivos de *build* foram gerados, você pode iniciar a compilação do VTK. Execute o seguinte comando para compilar o projeto:

```
cmake --build . --config Release
```

Esse comando compilará o *driver* no modo *Release*, que depois será instalado em `C:/Drivers/VTK`.

1.6.3 Instalar o driver

1. Crie a pasta **Drivers** no seu computador, se não existir.
2. Execute o seguinte comando para instalar o driver:

```
cmake --install . --config Release --prefix "C:\Drivers\VTK"
```

```
-- Installing: C:\Drivers\VTK\lib\cmake\vtk-9.3\patches\3.22\FindMPI\test_mpi.f90.in
-- Installing: C:\Drivers\VTK\lib\cmake\vtk-9.3\patches\3.22\FindMPI.cmake
-- Installing: C:\Drivers\VTK\lib\cmake\vtk-9.3\patches\3.23\FindPython\Support.cmake
-- Installing: C:\Drivers\VTK\lib\cmake\vtk-9.3\patches\3.23\FindPython3.cmake
-- Installing: C:\Drivers\VTK\lib\cmake\vtk-9.3\patches\99\FindHDF5.cmake
-- Installing: C:\Drivers\VTK\lib\cmake\vtk-9.3\patches\99\FindOpenGL.cmake
-- Installing: C:\Drivers\VTK\lib\cmake\vtk-9.3\vtk-config.cmake
-- Installing: C:\Drivers\VTK\lib\cmake\vtk-9.3\vtk-config-version.cmake
-- Installing: C:\Drivers\VTK\lib\cmake\vtk-9.3\vtk-prefix.cmake
-- Installing: C:\Drivers\VTK\lib\cmake\vtk-9.3\VTK-vtk-module-find-packages.cmake
-- Installing: C:\Drivers\VTK\share\licenses\VTK\Copyleft.txt
PS C:\Users\Particular\Downloads\vtk-v9.3.0.rc1\vtk-v9.3.0.rc1\build> |
```

Figura 19: VTK drivers gerados com sucesso.

1.6.4 Variáveis de Ambiente do VTK

1. ****Criação da variável VTK_DIR:****

- (a) Abra o *Painel de Controle* do *Windows* e navegue até *Sistema e Segurança > Sistema > Configurações avançadas do sistema*.
- (b) Clique em *Variáveis de Ambiente...*
- (c) Na seção *Variáveis de sistema*, clique em *Novo...*
- (d) Crie uma nova variável chamada **VTK_DIR**. O valor deve ser o caminho do diretório onde o VTK foi instalado.

2. ****Atualização da variável Path:****

- (a) Ainda na tela de variáveis de ambiente, localize e selecione a variável **Path**, depois clique em **Editar...**
- (b) Na nova tela que se abrirá, clique em **Novo** para adicionar a seguinte entrada à lista existente:

```
%VTK_DIR%\bin
```

3. ****Reinicialização do sistema:****

- (a) Faça logout e login ou reinicie o sistema para que as novas variáveis sejam carregadas corretamente.

1.7 Instalação do *Gmsh*

O *Gmsh* é um gerador de malha de elementos finitos 3D de código aberto, com um mecanismo CAD integrado e pós-processador. No SIRERC, o *Gmsh* é utilizado para gerar a malha que representa o domínio a ser simulado, como poços de petróleo.

Seu objetivo de design é fornecer uma ferramenta de malha rápida, leve e fácil de usar com entrada paramétrica e recursos avançados de visualização. O *Gmsh* é construído em torno de quatro módulos: geometria, malha, solucionador e pós-processamento. A especificação de qualquer entrada para esses módulos é feita interativamente usando a interface gráfica do usuário ou em arquivos de texto ASCII usando a própria linguagem de *script* do *Gmsh*.

1. Baixe o código-fonte da versão 4.11.1 do *Gmsh*, atualmente em uso no SIRERC, no seguinte endereço: <https://gmsh.info/src/gmsh-4.11.1-source.tgzp>.
2. Descompacte no diretório Downloads.
3. Abra o *Developer Command Prompt*.

1.7.1 Configurar o driver

Na extração, serão criados dois diretórios: `vtk-v9.3.0.rc1/vtk-v9.3.0.rc`

Copie e cole no terminal as linhas do comando do *CMake*, uma a uma, separadas por um espaço, antes de executar.

```
cd Downloads\gmsh-4.11.1-source\gmsh-4.11.1-source
mkdir build
cd build

cmake -G "Visual Studio 16 2019" -A x64 -DCMAKE_CXX_STANDARD=17
-DMAKE_BUILD_TYPE=Release
-DENABLE_OPENMP=False
-DENABLE_PRIVATE_API=True
-DENABLE_BUILD_DYNAMIC=True
-DENABLE_TESTS=False
```

```

-- Gmsh 4.11.1 has been configured for Windows64
--
-- * Build options: 64Bit ALGLIB[contrib] ANN[contrib] Bamg Blossom DIntegration
Kbipack MathEx[contrib] Mesh Metis[contrib] Netgen NoSocklenT NoVsnprintf ONELAB
lugins Post QuadMeshingTools QuadTri Solver TetGen/BR Voro++[contrib] WinslowUntan
-- * Build type: Release
-- * C compiler: C:/Program Files (x86)/Microsoft Visual Studio/2019/Community/VC
4/cl.exe
-- * C++ compiler: C:/Program Files (x86)/Microsoft Visual Studio/2019/Community/
x64/cl.exe
-- * Install prefix: C:/Program Files/gmsh
--
-- Configuring done (11.3s)
-- Generating done (0.3s)
-- Build files have been written to: C:/Users/Particular/Downloads/gmsh-4.11.1-sou
PS C:\Users\Particular\Downloads\gmsh-4.11.1-source\gmsh-4.11.1-source\build> |

```

Figura 20: *Gmsh* drivers configurados para o Visual Studio 16 2019 com sucesso.

1.7.2 Compilar o driver

Agora que os arquivos de *build* foram gerados, você pode iniciar a compilação do *Gmsh*. Execute o seguinte comando para compilar o projeto:

```
cmake --build . --config Release
```

Esse comando compilará o *driver* no modo *Release*, que depois será instalado em *C:/Drivers/Gmsh*.

1.7.3 Instalar o driver

1. Execute o seguinte comando para instalar o driver:

```
cmake --install . --config Release --prefix "C:\Drivers\Gmsh"
```

```

-- Up-to-date: C:\Drivers\Gmsh/share/doc/gmsh/examples/simple_geo/sphere.geo
-- Up-to-date: C:\Drivers\Gmsh/share/doc/gmsh/examples/simple_geo/splines.geo
-- Up-to-date: C:\Drivers\Gmsh/share/doc/gmsh/examples/simple_geo/square_regular.geo
-- Up-to-date: C:\Drivers\Gmsh/share/doc/gmsh/examples/simple_geo/tower.geo
-- Up-to-date: C:\Drivers\Gmsh/share/doc/gmsh/examples/simple_geo/tower.i1
-- Up-to-date: C:\Drivers\Gmsh/share/doc/gmsh/examples/simple_geo/tower.i2
-- Up-to-date: C:\Drivers\Gmsh/share/doc/gmsh/examples/simple_geo/tower.i3
-- Up-to-date: C:\Drivers\Gmsh/share/doc/gmsh/examples/simple_geo/tower.i4
-- Up-to-date: C:\Drivers\Gmsh/share/doc/gmsh/examples/simple_geo/tower.i5
-- Up-to-date: C:\Drivers\Gmsh/share/doc/gmsh/examples/simple_geo/transfinite.geo
-- Up-to-date: C:\Drivers\Gmsh/share/doc/gmsh/examples/struct/Exists_GetForced.geo
-- Up-to-date: C:\Drivers\Gmsh/share/doc/gmsh/examples/struct/struct.geo
PS C:\Users\Particular\Downloads\gmsh-4.11.1-source\gmsh-4.11.1-source\build>

```

Figura 21: *Gmsh* drivers instalados com sucesso em *C:/Drivers/Gmsh*.

1.7.4 Variáveis de Ambiente do *Gmsh*

1. Tecle Win + I para acessar as configurações do *Windows* e localize *variáveis de ambiente*.
2. Na seção *Variáveis de sistema*, clique em *Novo....*

3. Crie uma nova variável chamada *GMSH_INC*. O valor deve ser o caminho dos drivers do GMSH. Exemplo: *C:\Drivers\Gmsh\include*.
4. Em seguida, crie outra variável chamada *GMSH_LIB*. Exemplo: *C:\Drivers\Gmsh\lib*.

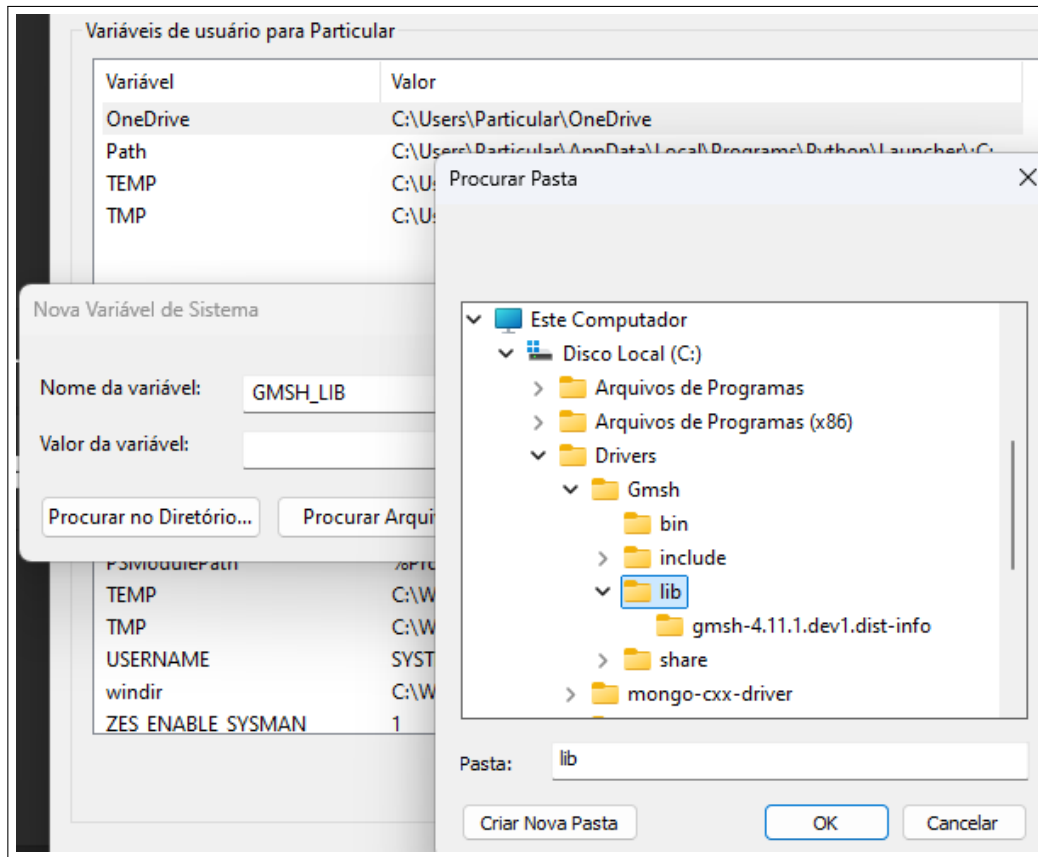


Figura 22: Configuração da variável *GMSH_LIB*.

Faça logout e login ou reinicie o sistema para que as novas variáveis sejam carregadas corretamente.

1.8 Compilação do **SIRERC**

Neste ponto, o ambiente está pronto para compilar o código do **SIRERC**. Este guia foi testado com a versão do sistema disponível na branch *development* do projeto no GitHub em 25/04/2024.

1. ****Clonando o repositório:****
 - (a) Faça o *clone* do repositório Git disponível em https://github.com/sirercitafront/SIRERC_ProjetoPetrobras usando a ferramenta de sua preferência, como *Sourcetree* ou *Git Bash*.
2. ****Checkout do commit de referência:****
 - (a) Faça o *checkout* do *commit* de referência usando a ferramenta de sua preferência. O *hash* do *commit* de referência é *3f635a7*.
3. ****Preparação do diretório:****
 - (a) Navegue até o diretório do **SIRERC** no sistema de arquivos e, na pasta principal, exclua definitivamente todos os diretórios cujo nome inicie com **build-**. Esses diretórios são gerados automaticamente pelo *Qt Creator*.

- (b) Ainda no diretório do SIRERC, remova o arquivo `FonteSirerc\CMakeLists.txt.user`. Esse arquivo é gerado automaticamente pelo *Qt Creator* e depende da instalação de cada usuário.

4. ****Configuração no *Qt Creator*:**

- (a) Inicie o *Qt Creator*, a IDE instalada durante a configuração do Qt. Na tela inicial, selecione a versão 5.15.2 do Qt (vide Figura 23).

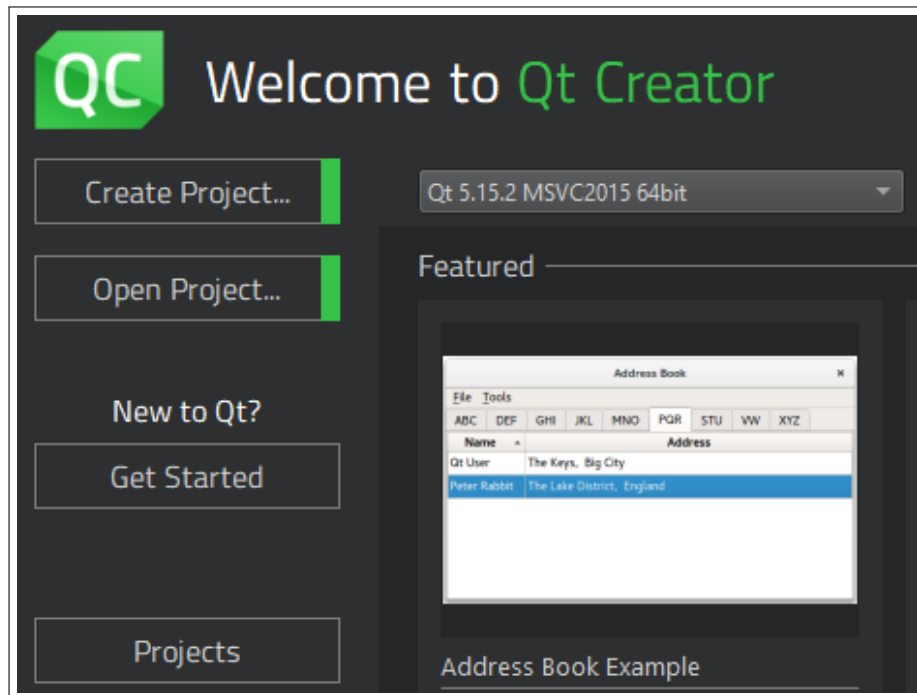


Figura 23: Tela inicial do *Qt Creator* e escolha de versão

- (b) Clique no botão `Open Project...`, navegue até o diretório onde está o código do SIRERC e selecione o arquivo `FonteSirerc/CMakeLists.txt` para abrir o projeto.
- (c) Ao abrir o projeto, o *Qt Creator* pode exibir um alerta relacionado às configurações locais que precisam ser refeitas automaticamente (vide Figura 24). Esse alerta pode ser ignorado clicando em `OK`. Ele aparecerá apenas uma vez.

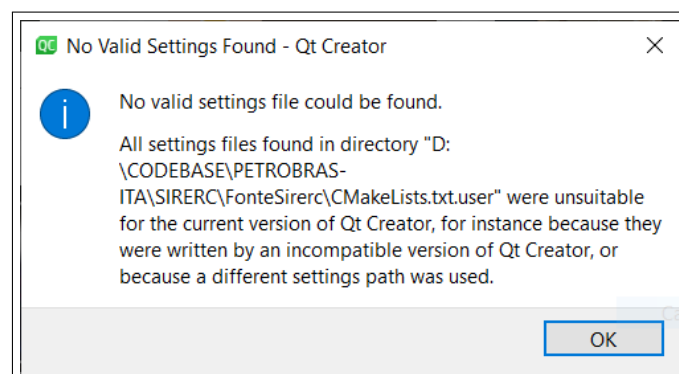


Figura 24: Alerta de primeira utilização do *Qt Creator*

- (d) Na tela seguinte, o *Qt Creator* solicitará que você selecione os compiladores e arquiteturas a serem suportados no projeto. Mantenha selecionada apenas a versão com o compilador *MSVC2015* 64-bit (vide Figura 25) e clique em `Configure Project` na parte inferior da tela.

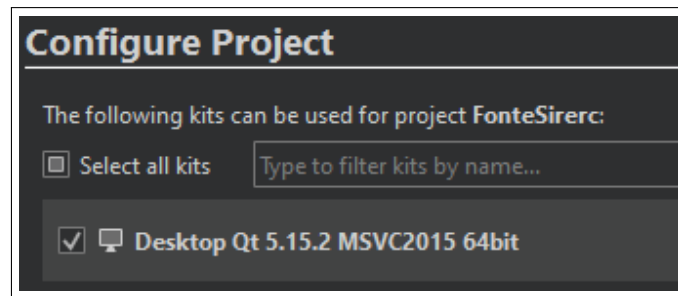


Figura 25: Seleção do tipo de compilador e plataforma no *Qt Creator*

5. ****Configuração do projeto:****

- (a) Aguarde o *Qt Creator* finalizar o carregamento do projeto. Se tudo estiver correto, você verá a árvore de diretórios do projeto (vide Figura 26). Se houver algum problema, verifique se o campo *Build Directory* aponta para a pasta **build-FonteSirerc-Desktop_Qt_5_15_2_\msvc{ }2015_64bit-Release** correta. Modifique-o se necessário e, em seguida, clique na aba *Initial Configuration* e reconfigure o projeto clicando em *Re-configure with Initial Parameters*.

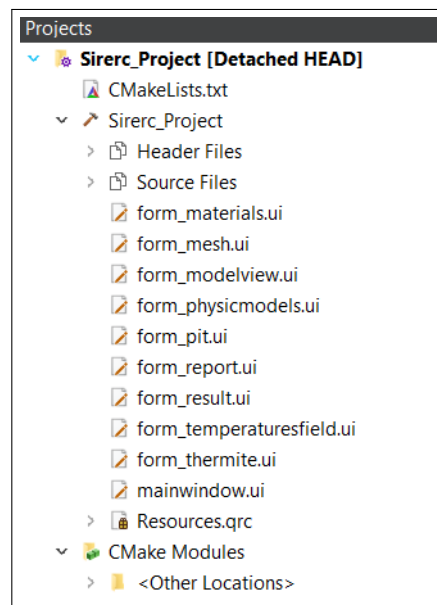


Figura 26: Árvore de diretórios do SIRERC

- (b) Na parte inferior esquerda do *Qt Creator*, altere o tipo de *build* de *Debug* para *Release* e aguarde a reconfiguração (vide Figura 27).

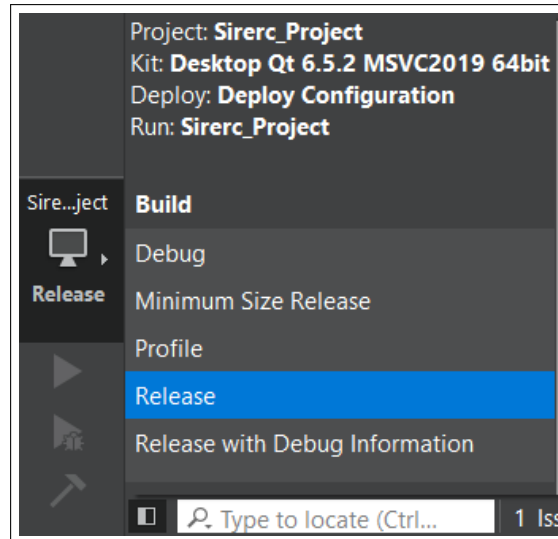



Figura 27: Mudança de *build* para *Release* no *Qt Creator*

6. **Execução do SIRERC:**

- (a) Neste ponto, o SIRERC pode ser executado clicando no botão  no *Qt Creator*. A tela principal do SIRERC deverá ser exibida após alguns segundos de inicialização (vide Figura 28).

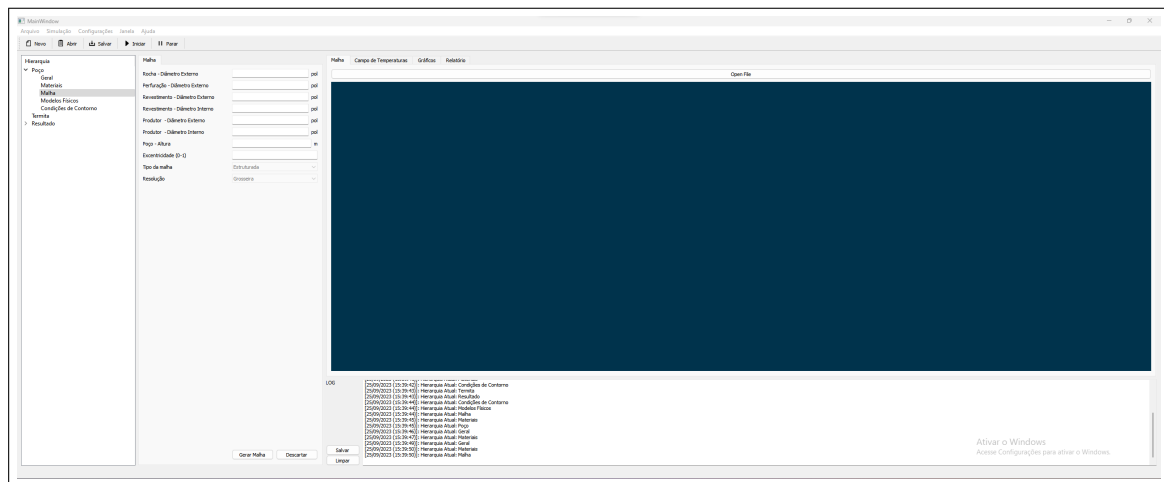


Figura 28: SIRERC em execução

1.9 Criação do Instalador

Um instalador *offline* simplifica o processo de instalação ao fornecer uma interface amigável e incluir todas as dependências necessárias em um único arquivo. Projetos baseados no Qt, como o SIRERC, podem utilizar o módulo `QtInstallerFramework` para gerar instaladores multiplataforma.

1. **Instalação do `QtInstallerFramework`:

- (a) O `QtInstallerFramework` está disponível como um componente opcional durante a instalação inicial do Qt ou pode ser instalado posteriormente usando a ferramenta `QtMaintenanceTool`. Em ambos os casos, selecione o módulo na lista de componentes disponíveis e prossiga com a instalação (vide Figura 29).

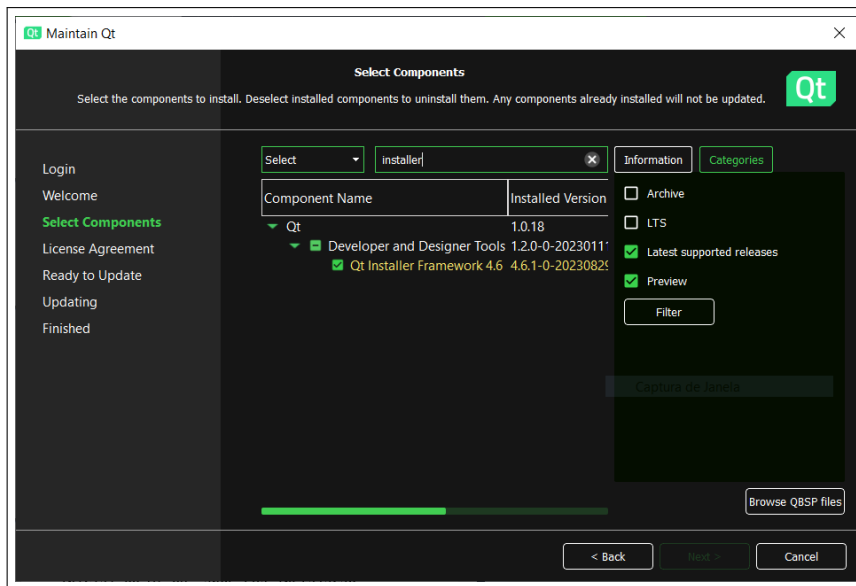


Figura 29: Instalação do componente QtInstallerFramework

2. **Preparação do ambiente de deployment:**

- Após a instalação, o QtInstallerFramework estará disponível no diretório `${QtDirectory}/Tools/QtInstallerFramework`, onde `${QtDirectory}` representa o caminho do sistema de arquivos onde o Qt está instalado.
- Crie um diretório temporário em qualquer local do sistema de arquivos para preparar o SIRERC para *deployment*.
- Dentro deste diretório, crie dois subdiretórios chamados `config` e `packages`, ambos inicialmente vazios (vide Figura 30).

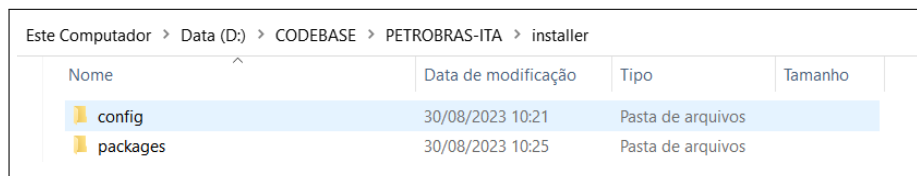


Figura 30: Diretório inicial para geração do instalador

3. **Configuração do arquivo config.xml:**

- No diretório `config`, crie um arquivo chamado `config.xml` com o seguinte conteúdo:

```
<?xml version="1.0" encoding="UTF-8"?>
<Installer>
  <Name>SIRERC</Name>
  <Version>1.0.0</Version>
  <Title>SIRERC Installer</Title>
  <Publisher>ITA</Publisher>
  <StartMenuDir>SIRERC</StartMenuDir>
  <TargetDir>@HomeDir@/SIRERC</TargetDir>
</Installer>
```

- Verifique as *tags* aceitas neste arquivo na documentação do Qt em <https://doc.qt.io/qtinstallerframework/ifw-globalconfig.html>. O conteúdo apresentado é apenas um exemplo; verifique se há uma versão mais completa deste arquivo em uso no SIRERC.

4. **Configuração dos pacotes:**

- (a) Um *package* é um módulo que contém uma versão específica de certas partes do *software*. Assim como na interface do instalador do Qt, onde é possível escolher módulos a serem instalados, você pode modularizar o SIRERC.
- (b) No diretório `packages`, crie um subdiretório chamado `com.ita.sirercx64` (vide Figura 31).

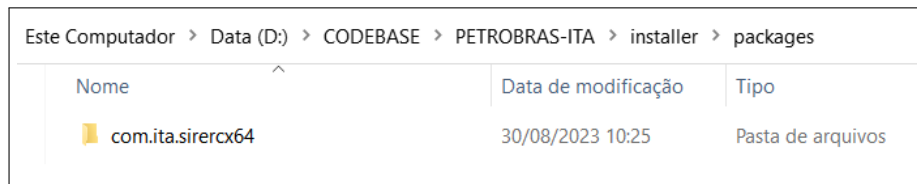


Figura 31: Criação do diretório para pacote 64 bits do SIRERC

- (c) Dentro do diretório `com.ita.sirercx64`, crie dois subdiretórios chamados `data` e `meta` (vide Figura 32). O primeiro conterá executáveis, bibliotecas e outros arquivos necessários para o funcionamento do SIRERC. O segundo conterá informações complementares do *package*.

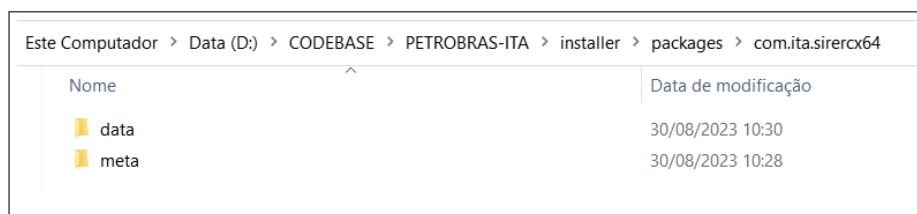


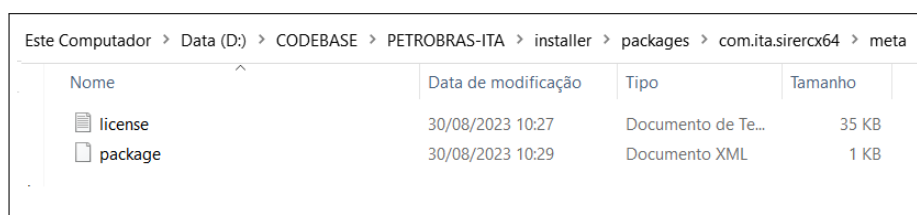
Figura 32: Subdiretórios *data* e *meta* criados

- (d) No diretório `meta`, crie um arquivo chamado `license.txt` contendo as informações da licença associada ao *software*. A *GNU General Public License* (GPL) pode ser obtida em <https://www.gnu.org/licenses/gpl-3.0.txt>. Verifique qual licença está em uso atualmente no SIRERC.
- (e) Ainda no diretório `meta`, crie um arquivo chamado `package.xml` para definir os comportamentos do instalador e mensagens informativas. O conteúdo deve ser semelhante ao seguinte:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package>
  <DisplayName>SIRERC 64 Bits</DisplayName>
  <Description>x64 Version</Description>
  <Version>1.0.0</Version>
  <ReleaseDate>2023-08-30</ReleaseDate>
  <Licenses>
    <License name="License Information" file="license.txt" />
  </Licenses>
  <Default>true</Default>
</Package>
```

- (f) Verifique as *tags* aceitas neste arquivo na documentação do Qt em <https://doc.qt.io/qtinstallerframework/ifw-component-description.html#package-information-file-syntax>. O conteúdo apresentado é apenas um exemplo; verifique se há uma versão mais completa deste arquivo em uso no SIRERC.

(g) Isso conclui a configuração dos arquivos no diretório *meta* (vide Figura 33).

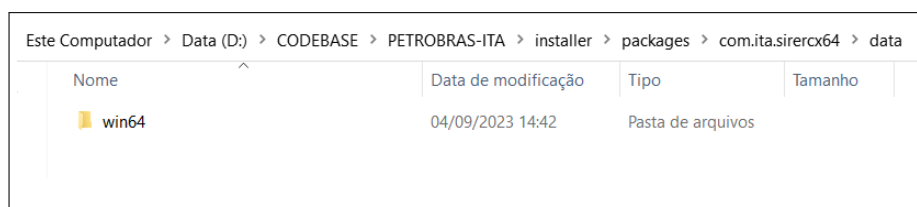


Nome	Data de modificação	Tipo	Tamanho
license	30/08/2023 10:27	Documento de Te...	35 KB
package	30/08/2023 10:29	Documento XML	1 KB

Figura 33: Arquivos no diretório *meta*

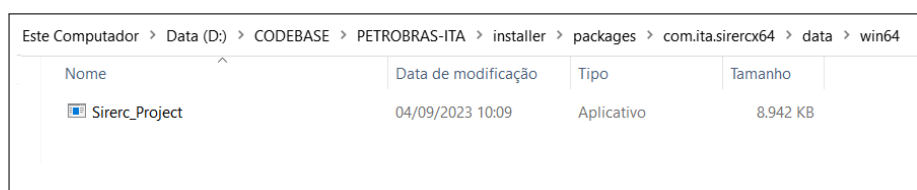
5. **Preparação dos arquivos de deployment:**

(a) No diretório *data*, crie um subdiretório chamado *win64* (vide Figura 34). Copie para este diretório o arquivo *.exe* gerado pelo *Qt Creator* ou outra IDE usada para desenvolver o SIRERC (vide Figura 35). Este arquivo normalmente está localizado nos diretórios de build criados durante o desenvolvimento.



Nome	Data de modificação	Tipo	Tamanho
win64	04/09/2023 14:42	Pasta de arquivos	

Figura 34: Criação do subdiretório *win64* dentro de *data*



Nome	Data de modificação	Tipo	Tamanho
Sirerc_Project	04/09/2023 10:09	Aplicativo	8.942 KB

Figura 35: Arquivo *.exe* no diretório *win64*

- (b) O diretório *win64* deve conter todos os executáveis, bibliotecas e dependências necessárias para executar o SIRERC. O Qt possui uma ferramenta de *deploy* que traz todas essas dependências para o diretório a partir do arquivo executável *.exe*. Para acessar essa ferramenta, abra um terminal no *Powershell* do *Windows* e navegue até o diretório *win64*.
- (c) No terminal *Powershell*, certifique-se de estar no diretório que contém o executável do SIRERC copiado anteriormente. Execute o comando a seguir, substituindo a variável `${QtDirectory}` pelo caminho onde o Qt está instalado e usando o nome correto do arquivo *.exe*:

```
${QtDirectory}\5.15.2\msvc2015_64\bin\windeployqt.exe Sirerc_Project.exe --compile
```

(d) Aguarde a conclusão do processo de *deploy* e verifique se há uma grande quantidade de novos arquivos e diretórios dentro do diretório *win64* (vide Figura 36).

te Computador > Data (D:) > CODEBASE > PETROBRAS-ITA > installer > packages > com.ita.sirercx64 > data > win64

Nome	Data de modificação	Tipo	Tamanho
generic	04/09/2023 14:50	Pasta de arquivos	
geometryloaders	04/09/2023 14:50	Pasta de arquivos	
iconengines	04/09/2023 14:50	Pasta de arquivos	
imageformats	04/09/2023 14:50	Pasta de arquivos	
networkinformation	04/09/2023 14:50	Pasta de arquivos	
platforms	04/09/2023 14:50	Pasta de arquivos	
renderers	04/09/2023 14:50	Pasta de arquivos	
sceneparsers	04/09/2023 14:50	Pasta de arquivos	
styles	04/09/2023 14:50	Pasta de arquivos	
tls	04/09/2023 14:50	Pasta de arquivos	
translations	04/09/2023 14:50	Pasta de arquivos	
D3Dcompiler_47.dll	11/03/2014 07:54	Extensão de aplica...	4.077 KB
opengl32sw.dll	04/06/2020 04:50	Extensão de aplica...	20.150 KB
Qt6Concurrent.dll	06/07/2023 16:05	Extensão de aplica...	35 KB
Qt6Core.dll	06/07/2023 16:05	Extensão de aplica...	5.618 KB
Qt6Gui.dll	06/07/2023 16:05	Extensão de aplica...	7.820 KB
Qt6Network.dll	06/07/2023 16:06	Extensão de aplica...	1.342 KB
Qt6OpenGL.dll	06/07/2023 16:06	Extensão de aplica...	1.881 KB
Qt6OpenGLWidgets.dll	06/07/2023 16:06	Extensão de aplica...	59 KB
Qt6Pdf.dll	07/07/2023 17:05	Extensão de aplica...	5.331 KB
Qt6ShaderTools.dll	07/07/2023 00:26	Extensão de aplica...	3.219 KB
Qt6Svg.dll	07/07/2023 00:26	Extensão de aplica...	356 KB
Qt6Widgets.dll	06/07/2023 16:06	Extensão de aplica...	5.883 KB
Qt63DAnimation.dll	07/07/2023 21:15	Extensão de aplica...	489 KB
Qt63DCore.dll	07/07/2023 21:15	Extensão de aplica...	513 KB
Qt63DExtras.dll	07/07/2023 21:15	Extensão de aplica...	715 KB
Qt63DInput.dll	07/07/2023 21:15	Extensão de aplica...	378 KB
Qt63DLogic.dll	07/07/2023 21:15	Extensão de aplica...	71 KB
Qt63DRender.dll	07/07/2023 21:15	Extensão de aplica...	2.441 KB
Sirerc_Project	04/09/2023 10:09	Aplicativo	8.942 KB

Figura 36: Diretório *win64* após o *deploy*

6. **Resolução de dependências extras:**

- Na data de escrita deste documento, a ferramenta de *deploy* não consegue encontrar algumas dependências extras do SIRERC, como as bibliotecas do VTK. Essas e outras dependências devem ser rastreadas usando ferramentas como a disponível em <https://github.com/lucasg/Dependencies>.
- A ferramenta *Dependencies* possui uma interface de usuário que permite fornecer um arquivo executável *.exe* e receber uma lista de todas as dependências (*.dlls*) necessárias para o funcionamento deste executável (vide Figura 37).

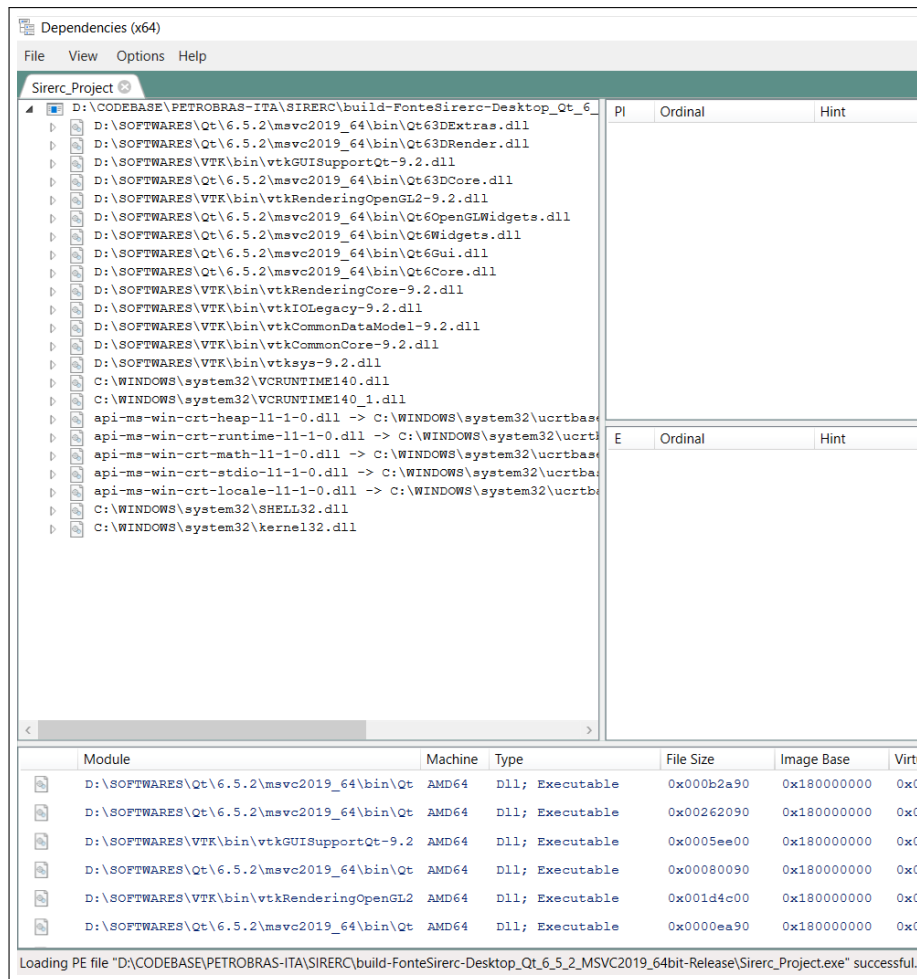


Figura 37: Dependências do SIRERC encontradas com a ferramenta *Dependencies*

- (c) Copie todas as dependências necessárias para o diretório `win64`, incluindo bibliotecas do VTK, por exemplo.

Este processo deve ser realizado com cautela, pois sem todas as dependências necessárias, o SIRERC não funcionará corretamente após a instalação.

7. **Geração do instalador:**

- (a) Após a resolução de todas as dependências, você pode gerar o instalador. Para isso, abra novamente um terminal *Powershell* e acesse o diretório principal do instalador, onde estão os diretórios `config` e `packages`.
- (b) Nesse diretório, execute o seguinte comando para gerar o instalador, substituindo a variável `${QtDirectory}` pelo caminho onde o Qt está instalado:

```
${QtDirectory}\Tools\QtInstallerFramework\4.6\bin\binarycreator.exe -c config\conf
```

- (c) Aguarde a geração do instalador. Se o processo for bem-sucedido, o terminal *Powershell* não exibirá nenhuma saída, mas você encontrará um novo executável *SirercInstaller.exe* no diretório do instalador (vide Figura 38). Este é o arquivo que deve ser fornecido para instalação nas máquinas dos usuários (vide Figura 39).

Meu Computador > Data (D:) > CODEBASE > PETROBRAS-ITA > installer

Nome	Data de modificação	Tipo	Tamanho
config	30/08/2023 10:21	Pasta de arquivos	
packages	30/08/2023 10:25	Pasta de arquivos	
SirercInstaller	04/09/2023 15:33	Aplicativo	53.178 KB

Figura 38: Instalador gerado para o SIRERC

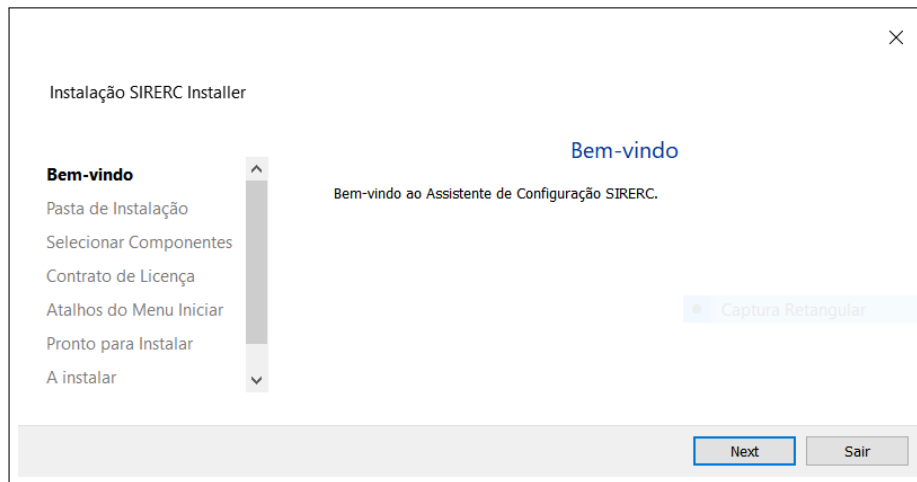


Figura 39: Execução do instalador do SIRERC