

# Guia de instalação dos ambientes de desenvolvimento do SIRERC

[https://github.com/sirercita/SIRERC\\_ProjetoPetrobrasdocs/](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	01/09/2024	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

<b>1</b>	<b>INSTALAÇÃO DO AMBIENTE WINDOWS</b>	<b>4</b>
1.1	<i>Download</i> e instalação da última versão do <i>Qt</i> . . . . .	4
1.2	Variáveis de ambiente . . . . .	9
1.3	Instalação do <i>Visual Studio</i> . . . . .	11
1.4	Instalação do VTK . . . . .	15
1.5	Instalação do <i>Gmsh</i> . . . . .	20
1.6	Compilação do SIRERC . . . . .	24
1.7	Criação do Instalador . . . . .	28
1.8	Instalação do <i>WSL</i> . . . . .	34
<b>2</b>	<b>INSTALAÇÃO DO QT 5.15.2 NO WINDOWS</b>	<b>35</b>
2.1	Requisitos do Sistema . . . . .	35
2.2	Windows: . . . . .	35
2.3	BUILD! . . . . .	35
2.4	Dicas . . . . .	36
<b>3</b>	<b>INSTALAÇÃO DO AMBIENTE LINUX</b>	<b>38</b>

# 1 INSTALAÇÃO DO AMBIENTE WINDOWS

O SIREC 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 de uma máquina com *Windows 11 Pro*.

## 1.1 *Download* e instalação da última versão do *Qt*

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 1).

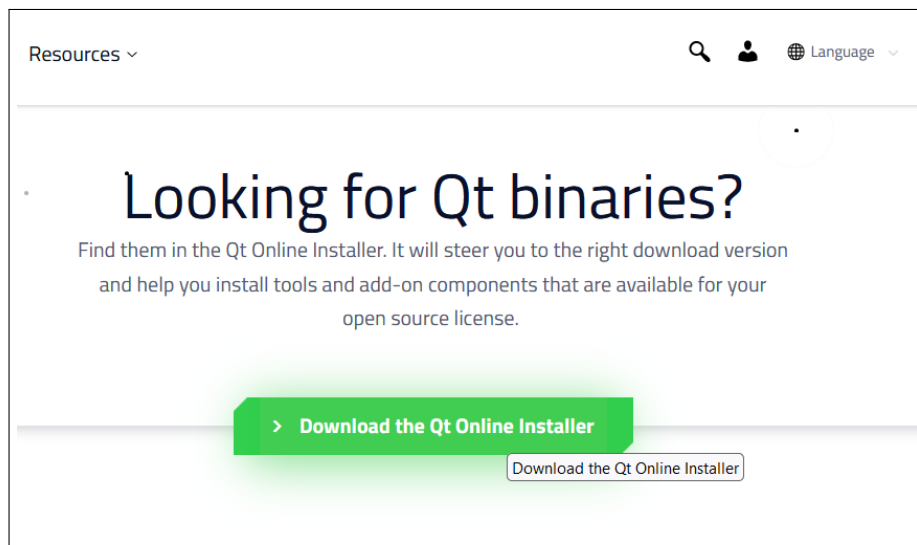


Figura 1: *Download* do instalador do *Qt*

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 2).



Figura 2: 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 3

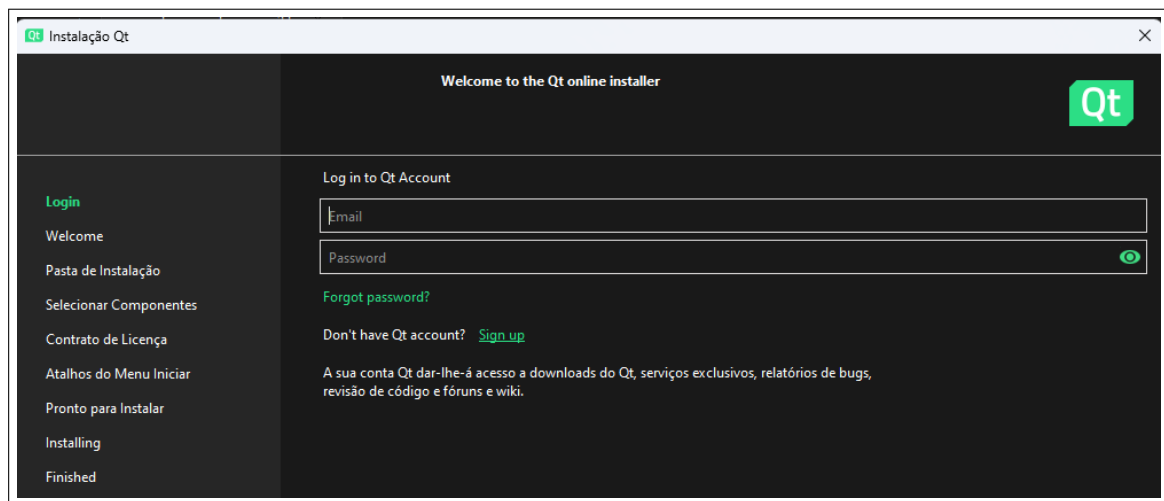


Figura 3: Digite seus dados a fim de continuar o processo de download do *Qt*.

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

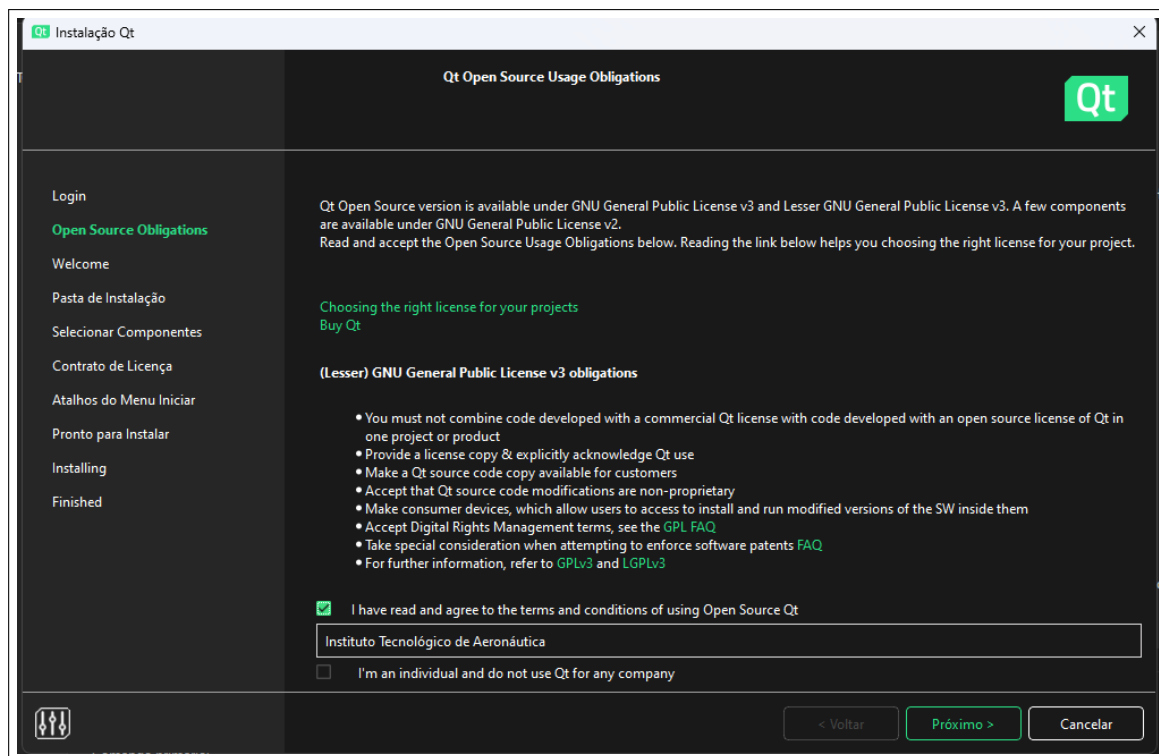


Figura 4: Marque que concorda e clique em **Próximo**.

8. Em seguida, será oferecida a possibilidade de escolher o caminho para instalação (Figura 5).

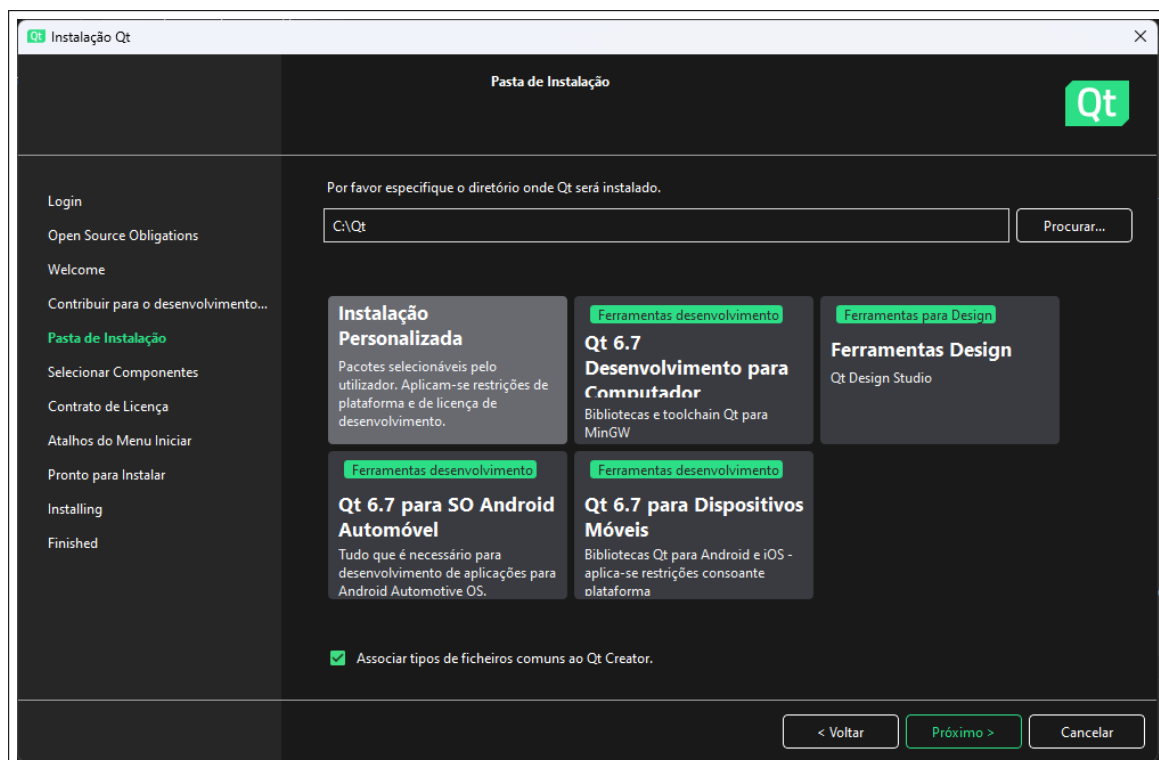


Figura 5: Escolha o caminho e clique em **Próximo**.

9. Em seguida, será oferecida a possibilidade de escolher os componentes da instalação (Figura 6). Caso não escolha todos os componentes, poderá instalá-los posteriormente.

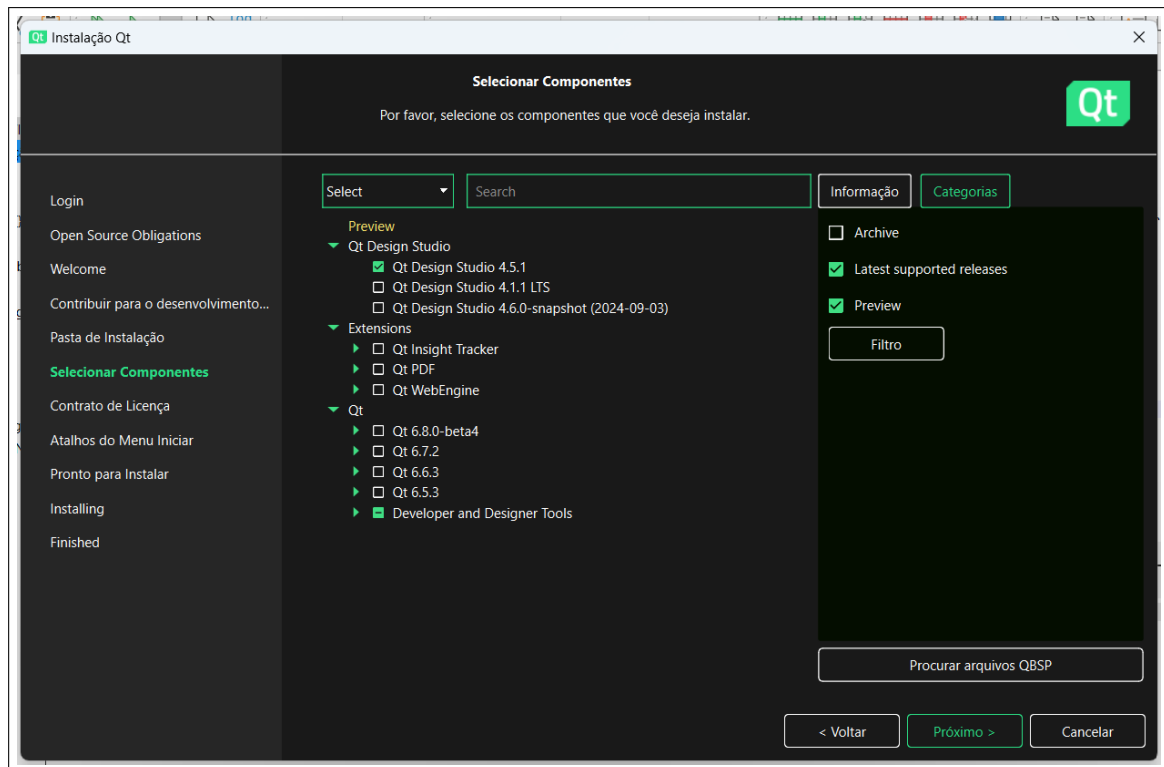


Figura 6: Escolha todos os componentes e clique em **Próximo**.

Como, no passo anterior, não é mais possível optar pelo *Qt 5.15.2*, nos resta a opção da instalação a versão *offline*.

<https://www.qt.io/offline-installers>

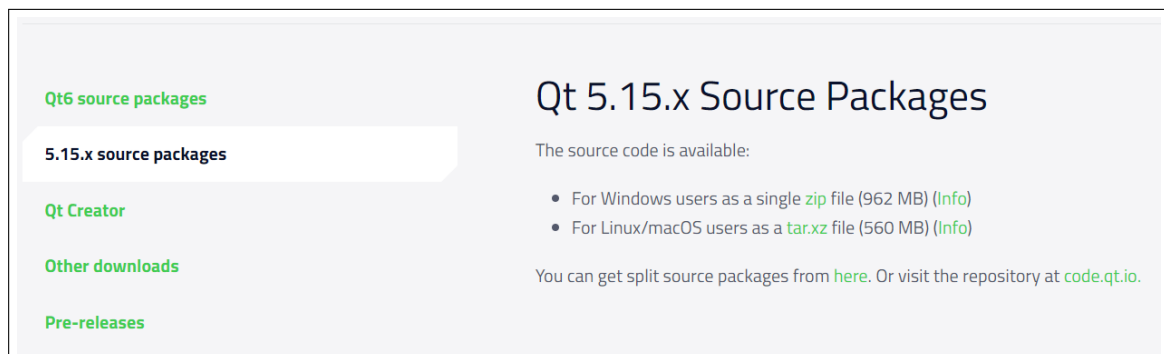


Figura 7: *Download* do instalador do Qt 5.15.x

Salve o arquivo compactado e continue com a instalação do *Qt Creator*. Posteriormente iremos demonstrar a instalação do *Qt 5.15.2*.

Nome:

Tipo:

pastas Salvar

Figura 8: Salve o arquivo compactado.

10. Em seguida, será oferecida a possibilidade de marcar que concorda com os termos do *CMake license agreement* (Figura 9). Marque e continue com o processo de instalação.

Login  
 Open Source Obligations  
 Welcome  
 Contribuir para o desenvolvimento...  
 Pasta de Instalação  
 Selecionar Componentes  
**Contrato de Licença**  
 Atalhos do Menu Iniciar  
 Pronto para Instalar  
 Installing  
 Finished

**CMake license agreement**  
 GPLv3 with Qt Company GPL Exception License Agreement  
 MICROSOFT SOFTWARE LICENSE TERMS MICROSOFT WINDOWS SOFTWARE DEVELOPMENT KIT (SDK) FOR WINDOWS 10  
 Ninja license agreement  
 OpenSSL 3 license agreement

CMake - Cross Platform Makefile Generator  
 Copyright 2000-2021 Kitware, Inc. and Contributors  
 All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name of Kitware, Inc. nor the names of Contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS

☒ I have read and agree to the terms contained in the license agreements.

< Voltar    Próximo >    Cancelar

Figura 9: Marque que concorda e clique em **Próximo**.

11. Aguarde a conclusão da instalação (Figura 10).



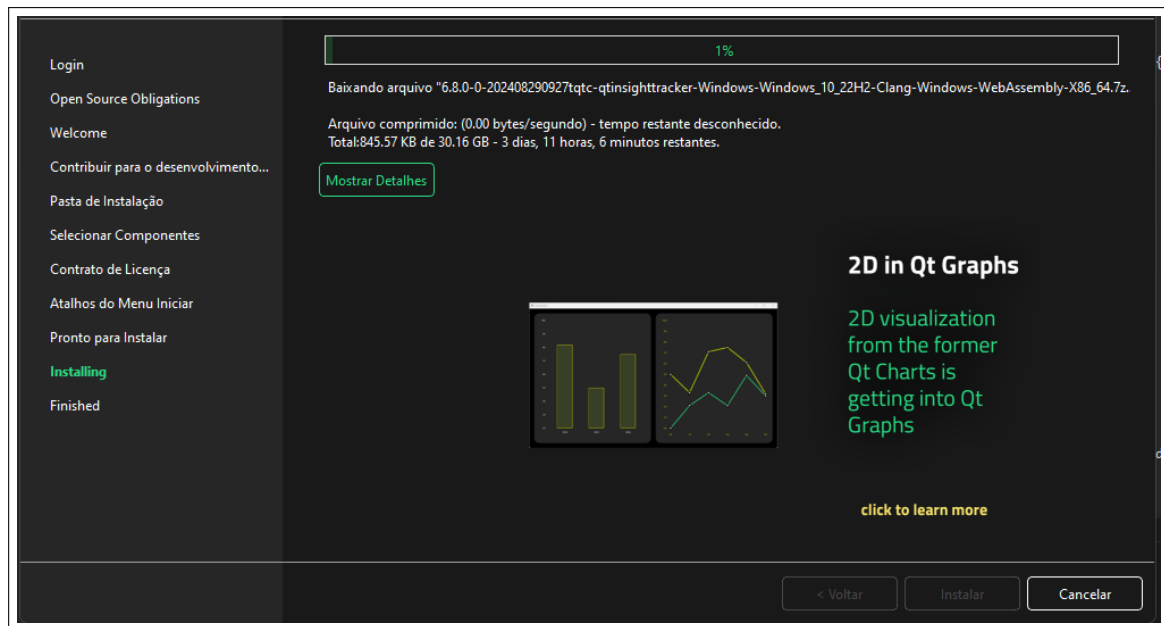


Figura 10: Andamento da instalação do *Qt Creator*.

## 1.2 Variáveis de ambiente

As variáveis de ambiente permitem que o ambiente de desenvolvimento localize corretamente as bibliotecas do Qt. Elas são especialmente importantes se o Qt foi instalado em um diretório não padrão no *Windows*, como fora de `C:\Program Files` ou `C:\Program Files (x86)`.

### 1. \*\*Criação da variável QTDIR:\*\*

- (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 **QTDIR**. O valor deve ser o caminho de instalação do Qt e do compilador Microsoft Visual C++ (*MSVC*) versão 2015 integrado. Exemplo: `C:\Qt\5.15.2\msvc2015_64`.

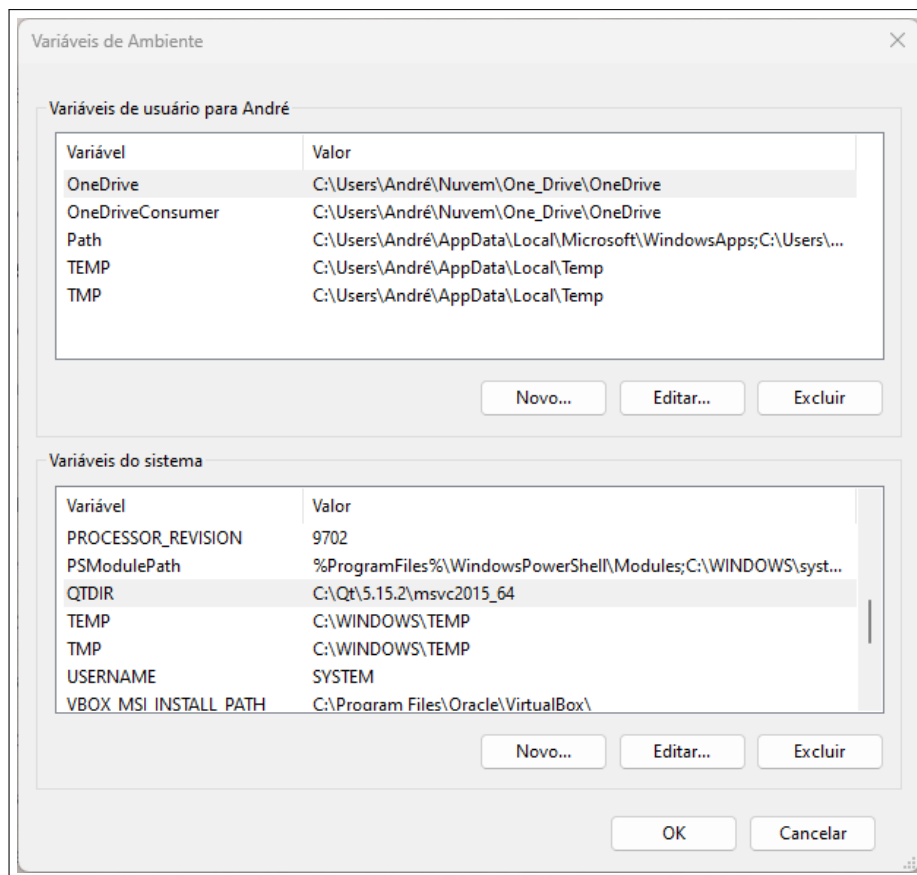


Figura 11: Criação da variável de ambiente *QTDIR*

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

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

```
%QTDIR%\bin
%QTDIR%\lib
```

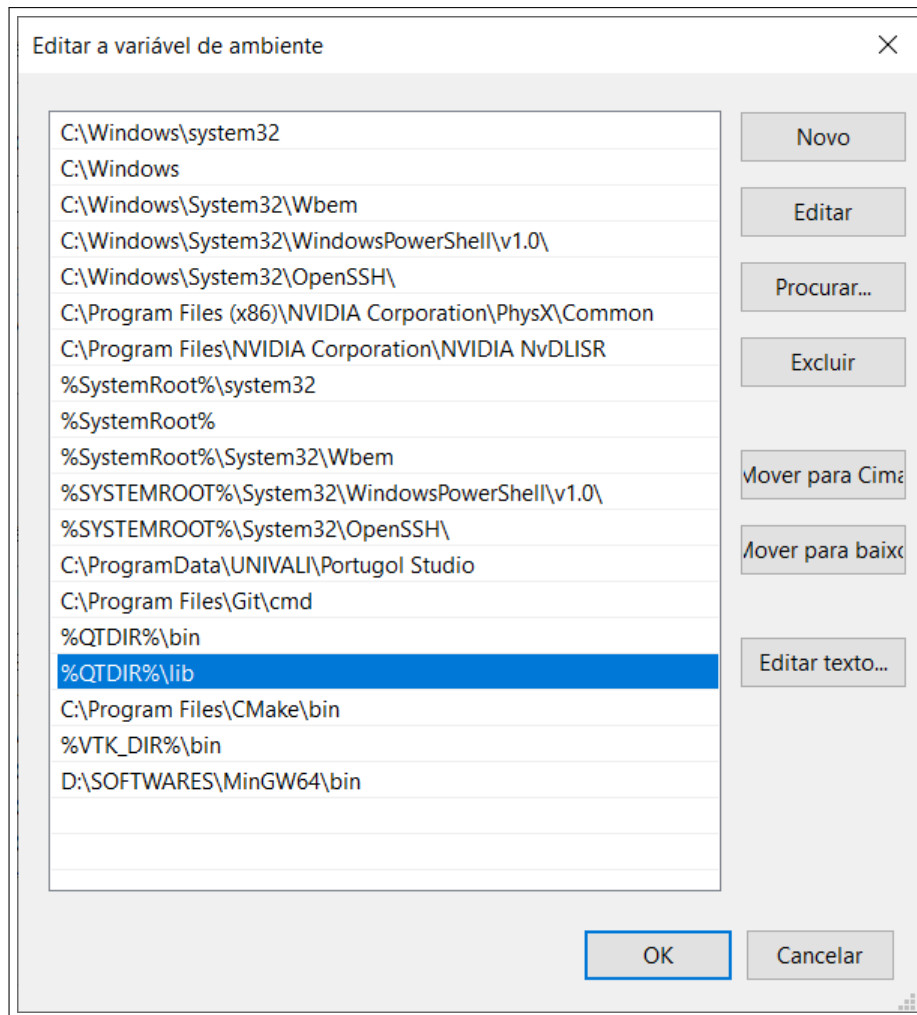


Figura 12: Novas variáveis dentro da lista em *Path*

### 3. **\*\*Reinicialização do sistema:\*\***


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

## 1.3 Instalação do *Visual Studio*

O *Visual Studio* é uma IDE desenvolvida e mantida pela *Microsoft*, amplamente utilizada em sistemas operacionais *Windows*. Ele utiliza o compilador *Microsoft Visual C++ (MSVC)*, uma alternativa ao *GNU Compiler Collection (GCC)* mais comum em ambientes *Linux*. Neste projeto, utilizamos a versão 2019 do *Visual Studio* e do compilador *MSVC*.

**O *Qt Creator* já inclui um compilador *MSVC* integrado, mas sua utilização é restrita ao contexto do *Qt Creator*. Recomendamos a instalação do *Visual Studio* e do *MSVC* globalmente para compilar bibliotecas de terceiros e outros projetos fora do *Qt Creator*.**

### 1. **\*\*Download do Visual Studio:\*\***

- (a) Acesse a página oficial de lançamentos do Visual Studio 2019 *Community* em <https://learn.microsoft.com/pt-br/visualstudio/releases/2019/release-notes>.
- (b) Clique no botão  para iniciar o *Download* do instalador.

## 2. \*\*Instalação do Visual Studio:\*\*

- (a) Após a finalização do *Download*, localize e execute o instalador.
- (b) Siga as instruções fornecidas pelo instalador para concluir a instalação do *Visual Studio*.
- (c) Durante a instalação, quando solicitado, selecione os módulos de C/C++ (vide Figura 13) para habilitar o suporte a projetos nessas linguagens.

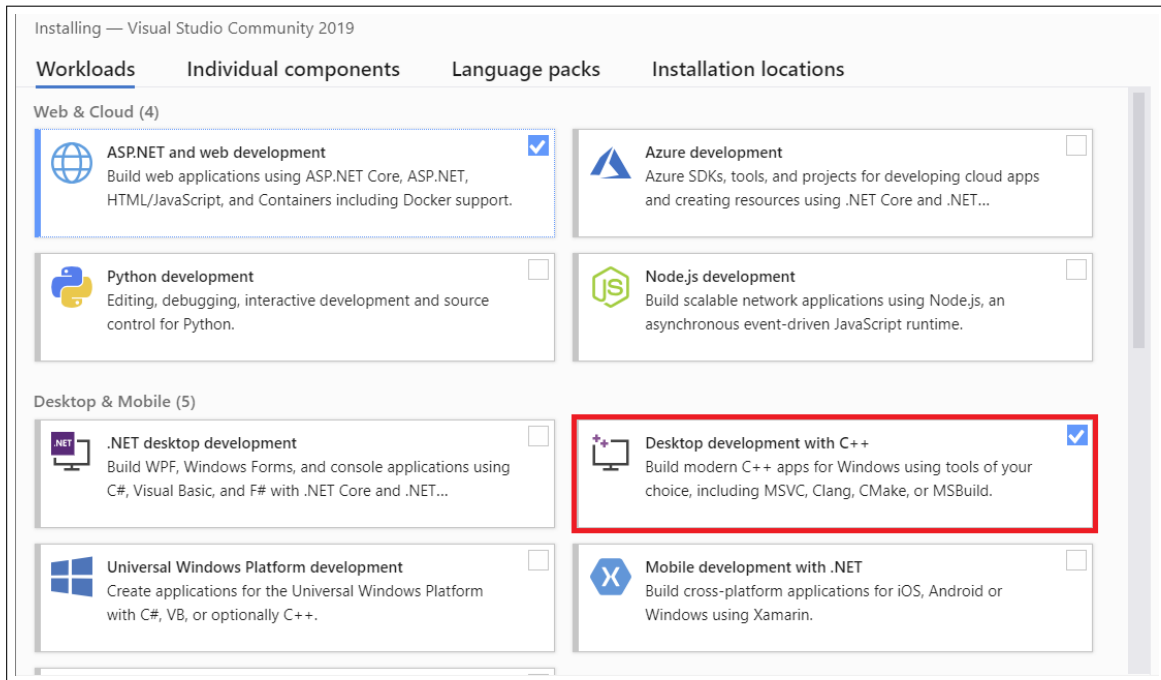


Figura 13: Módulo C/C++ habilitado durante a instalação do *Visual Studio*

**Caso não queira utilizar o Visual Studio (optando pelo *Qt Creator* ou pelo *VSCode*), você pode baixar e instalar as ferramentas de *Build* do Visual Studio 2019, versão 16.11.39.**

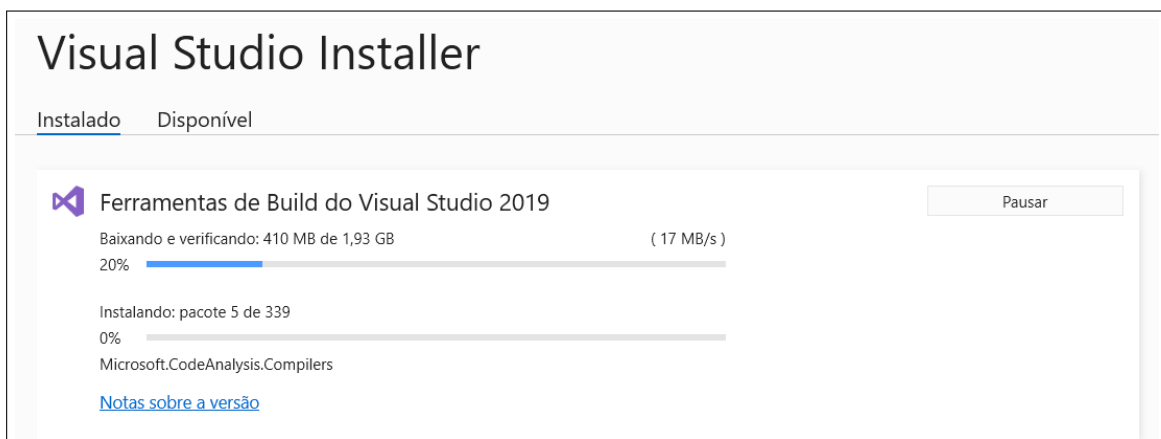


Figura 14: Visual Studio 2019 version 16.11.39

- 3. No *Qt Creator* ou no *VSCode*, configure o compilador compatível com o Qt 5.15.2, agora disponíveis após a instalação das ferramentas de *Build*.

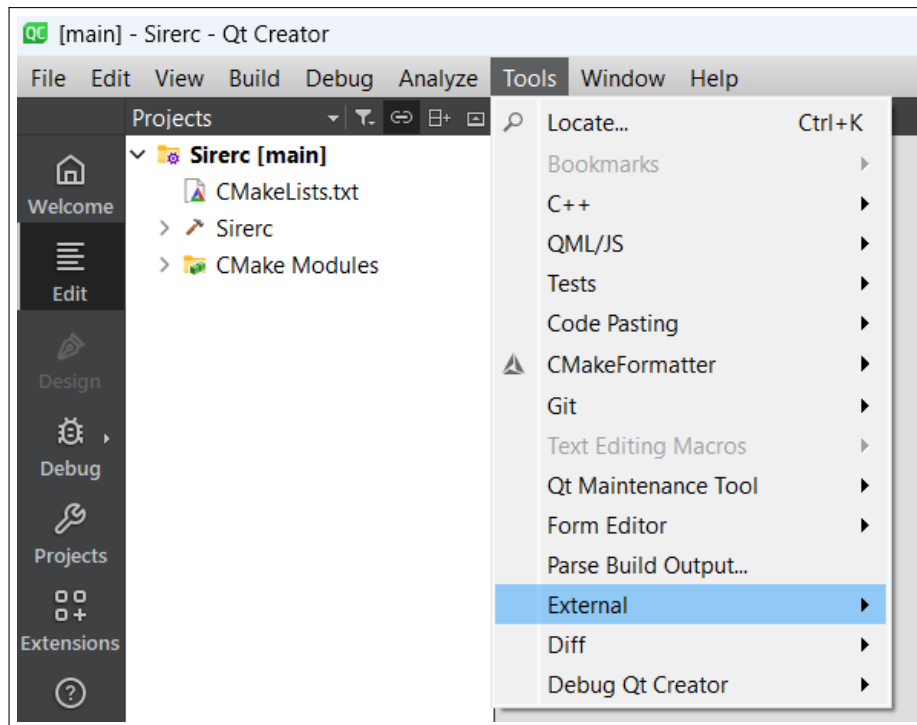


Figura 15: Configurar o *Qt Creator* para build

4. Escolha o kit de compilação baixado, o Visual Studio 2019 version 16.11.3, o Microsoft Visual C++ compiler, versão 16.11.x

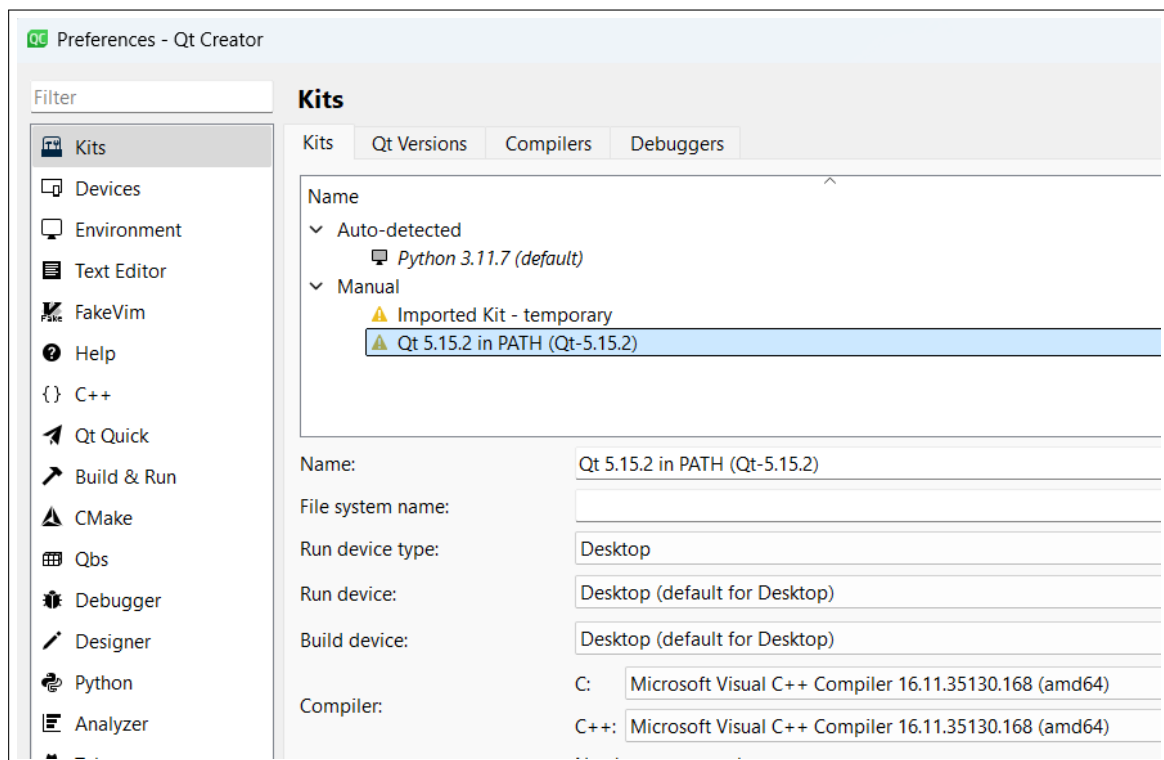
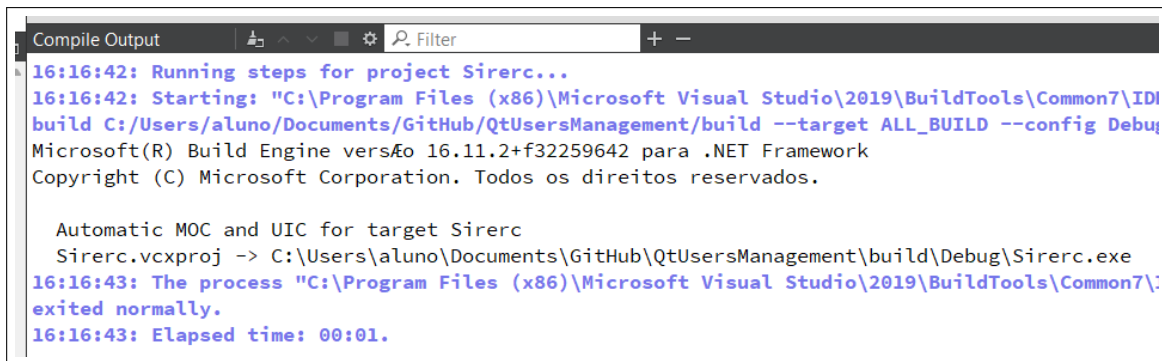


Figura 16: Escolha o kit de compilação baixado.



```
Compile Output
16:16:42: Running steps for project Sirerc...
16:16:42: Starting: "C:\Program Files (x86)\Microsoft Visual Studio\2019\BuildTools\Common7\IDE\
build C:/Users/aluno/Documents/GitHub/QtUsersManagement/build --target ALL_BUILD --config Debug
Microsoft(R) Build Engine vers o 16.11.2+f32259642 para .NET Framework
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Automatic MOC and UIC for target Sirerc
Sirerc.vcxproj -> C:\Users\aluno\Documents\GitHub\QtUsersManagement\build\Debug\Sirerc.exe
16:16:43: The process "C:\Program Files (x86)\Microsoft Visual Studio\2019\BuildTools\Common7\
exited normally.
16:16:43: Elapsed time: 00:01.
```

Figura 17: Compilação realizada sem erros.

<https://www.mongodb.com/try/download/community>

MongoDB Community Server Download

sirerc\_conn

sirerc\_db

sirerc\_collection

Install vcpkg if you don't already have it:

```
git clone https://github.com/microsoft/vcpkg
```

```
cd vcpkg
```

```
bootstrap-vcpkg.bat
```

Use vcpkg to install the required dependencies:

```
vcpkg install mongocxx:x64-windows
```

```
vcpkg integrate install
```

In your CMakeLists.txt, link the MongoDB C++ driver to your project:

```
find_package(mongocxx REQUIRED)
```

```
target_link_libraries(${PROJECT_NAME} PRIVATE mongocxx::mongocxx)
```

Baixe e extraia o driver do *MongoDB* para C.

Downloads

mongo-c-driver-1.27.6.tar

Abra a pasta em um terminal *Developer Command Prompt* (isto foi instalado com o Visual Studio).

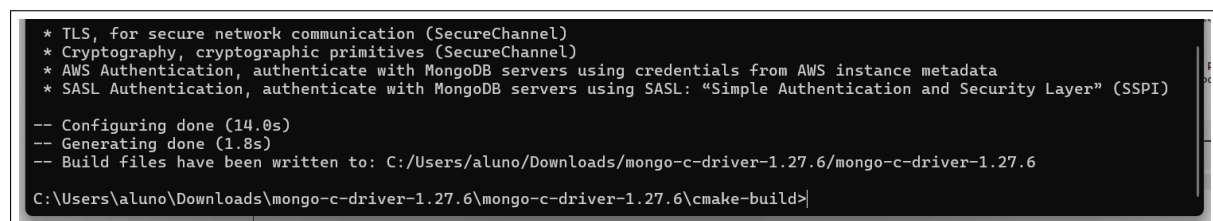
```
cd C:\Users\aluno\Downloads\mongo-c-driver-1.27.6\mongo-c-driver-1.27.6
```

```
mkdir cmake-build
```

```
cd cmake-build
```

```
cmake -G "NMake Makefiles" -DENABLE_AUTOMATIC_INIT_AND_CLEANUP=OFF
```

```
-DCMAKE_INSTALL_PREFIX="C:\Users\aluno\Downloads\mongo-c-driver-1.27.6\mongo-c-driver-1.27.6" ..
```



```
* TLS, for secure network communication (SecureChannel)
* Cryptography, cryptographic primitives (SecureChannel)
* AWS Authentication, authenticate with MongoDB servers using credentials from AWS instance metadata
* SASL Authentication, authenticate with MongoDB servers using SASL: "Simple Authentication and Security Layer" (SSPI)

-- Configuring done (14.0s)
-- Generating done (1.8s)
-- Build files have been written to: C:/Users/aluno/Downloads/mongo-c-driver-1.27.6/mongo-c-driver-1.27.6
C:\Users\aluno\Downloads\mongo-c-driver-1.27.6\mongo-c-driver-1.27.6\cmake-build>
```

Figura 18: Build do driver do *MongoDB* realizado com sucesso.

Agora instale do *driver* do *MongoDB*.

```
cd ..
nmake // Tenha paciência, pois irá demorar.
nmake install
```

```
ngoc-static-1.0-config.cmake
-- Installing: C:/Users/aluno/Downloads/mongo-c-driver-1.27.6/mongo-c-driver-1.27.6/lib/cmake/libmongoc-static-1.0/libmo
ngoc-static-1.0-config-version.cmake
-- Installing: C:/Users/aluno/Downloads/mongo-c-driver-1.27.6/mongo-c-driver-1.27.6/share/mongo-c-driver/COPYING
-- Installing: C:/Users/aluno/Downloads/mongo-c-driver-1.27.6/mongo-c-driver-1.27.6/share/mongo-c-driver/NEWS
-- Installing: C:/Users/aluno/Downloads/mongo-c-driver-1.27.6/mongo-c-driver-1.27.6/share/mongo-c-driver/README.rst
-- Installing: C:/Users/aluno/Downloads/mongo-c-driver-1.27.6/mongo-c-driver-1.27.6/share/mongo-c-driver/THIRD_PARTY_NOT
ICES
-- Generated uninstaller: C:/Users/aluno/Downloads/mongo-c-driver-1.27.6/mongo-c-driver-1.27.6/uninstall.cmd
-- Installing: C:/Users/aluno/Downloads/mongo-c-driver-1.27.6/mongo-c-driver-1.27.6/share/mongo-c-driver/uninstall.cmd
C:\Users\aluno\Downloads\mongo-c-driver-1.27.6\mongo-c-driver-1.27.6>
```

Figura 19: Driver do *MongoDB* instalado com sucesso.

## Sumário:

1. Faça o download e o build da libmongoc e da libbson manualmente.
2. Faça o build com o CMake (se não estive instalado, baixe o instalador.)
3. Faça a instalação das bibliotecas
4. Link essas bibliotecas no CMakeLists.txt do Sirerc.

## 1.4 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.

**Nesta seção, você aprenderá a compilar o VTK *from source*. Este processo pode ser demorado. Como alternativa, você pode obter o software já compilado por outros desenvolvedores do SIRERC e instalá-lo manualmente em um diretório do sistema operacional. Nesse caso, você pode pular para a Seção 4.1 deste guia.**

1. **\*\*Obtenção do código-fonte:\*\***
  - (a) 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>.
  - (b) Após o download, extraia o conteúdo do arquivo compactado para um diretório temporário no sistema.
2. **\*\*Preparação para compilação:\*\***
  - (a) Crie um diretório vazio chamado *build* dentro do diretório descompactado do código-fonte do VTK (vide Figura 20).

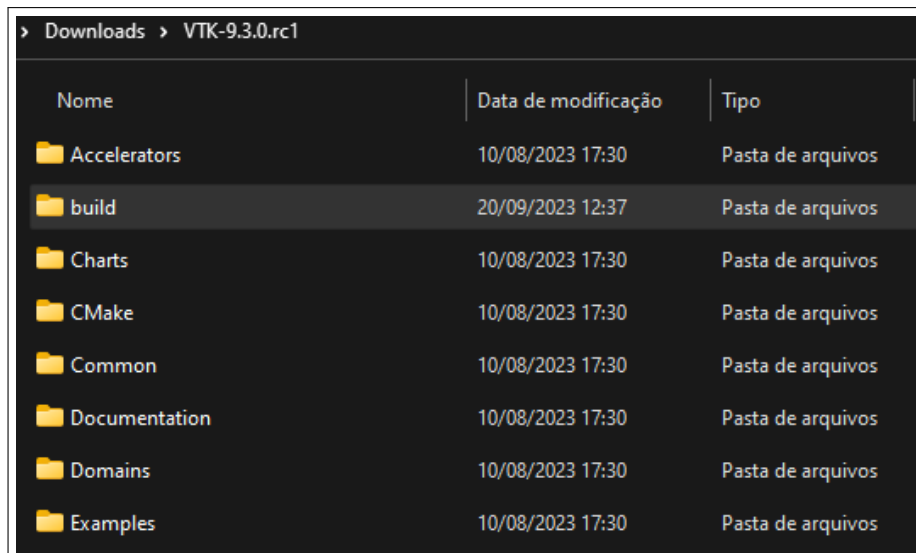


Figura 20: Criação do diretório *build* para compilação do VTK

- (b) Execute o *CMake* (interface visual). No campo *Where is the source code*, insira o caminho do código-fonte descompactado do VTK, algo como `C:\Users\username\Downloads\VTK-9.3.0.rc1` (vide Figura 21).

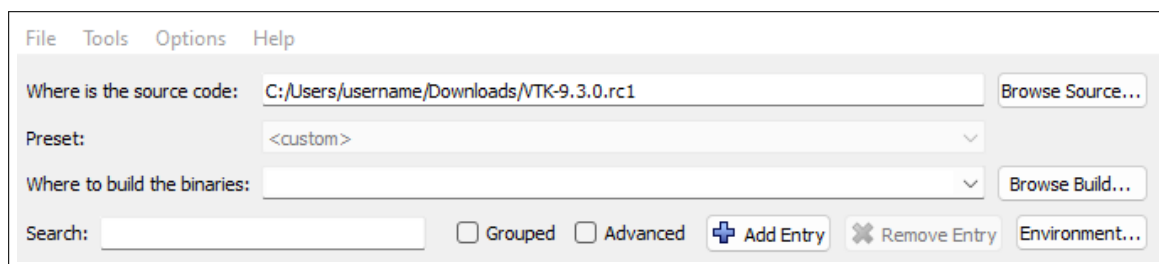


Figura 21: Configuração do diretório de código-fonte no *CMake*

- (c) No campo *Where to build the binaries*, insira o caminho do diretório *build* que você criou anteriormente, por exemplo: `C:\Users\username\Downloads\VTK-9.3.0.rc1\build` (vide Figura 22).

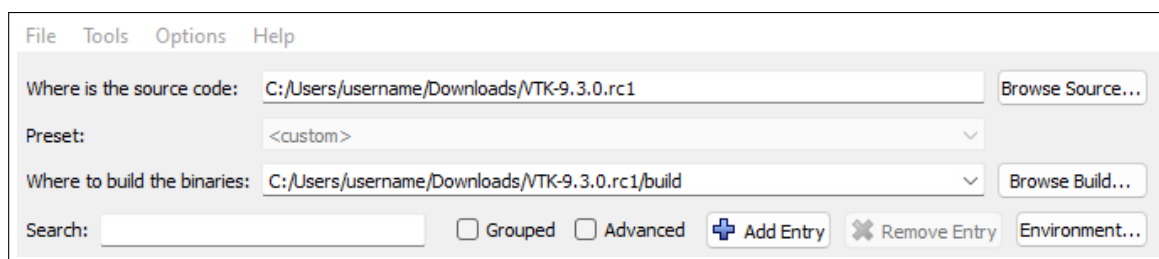


Figura 22: Configuração do diretório de código-fonte e *build* no *CMake*

### 3. \*\*Configuração do CMake:\*\*

- (a) Na parte inferior da tela do *CMake*, clique no botão **Configure**.
- (b) Na nova tela que se abrirá, selecione *Visual Studio 16 2019* como o gerador de projetos e escolha a arquitetura *x64* como plataforma. Em seguida, clique no botão **Finish** (vide Figura 23).



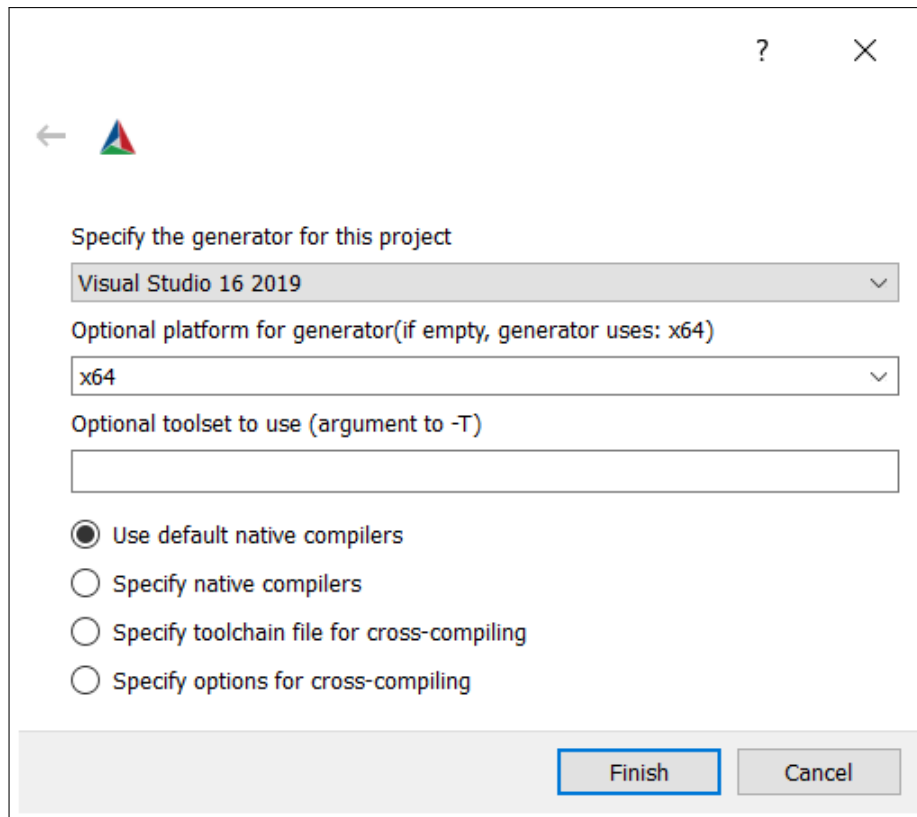


Figura 23: Definição do gerador de projeto e plataforma no *CMake*

- (c) Aguarde a conclusão do processo de configuração. Isso pode levar alguns minutos até que uma lista com várias variáveis em destaque vermelho apareça.
  - (d) Use o campo de busca *Search* para encontrar a variável `VTX_GROUP_ENABLE_Qt`. Altere o valor dessa variável para **YES** e clique em **Configure** novamente.
  - (e) Aguarde a reconfiguração do *CMake*.
  - (f) Opcionalmente, use o campo de busca *Search* para encontrar a variável `CMAKE_INSTALL_PREFIX`. Por padrão, essa variável está configurada para `C:/Program Files/VTX`. Você pode manter esse valor ou alterá-lo para outro diretório de sua escolha. Se fizer alterações, clique em **Configure** novamente e aguarde a reconfiguração.
  - (g) Clique no botão **Generate** para gerar o projeto para o *Visual Studio* 2019 no diretório `build`.
  - (h) Aguarde a conclusão do processo de geração do projeto.
4. **\*\*Compilação e instalação no Visual Studio:\*\***
- (a) Se o *Visual Studio* 2019 estiver corretamente instalado, clique no botão **Open Project** para carregar o projeto no *Visual Studio*. Se houver problemas, você pode abrir manualmente o arquivo `.sln` criado no diretório `build`.

**Se o diretório de instalação escolhido estiver em um dos diretórios protegidos do Windows, como `C:/Program Files/`, você precisará executar o *Visual Studio* como administrador para realizar a instalação. Caso contrário, ocorrerá um erro de permissão.**

- (b) Após o carregamento do projeto no *Visual Studio*, mude o tipo de *build* para *Release* na parte superior da interface e mantenha a arquitetura como *x64* (vide Figura 24).

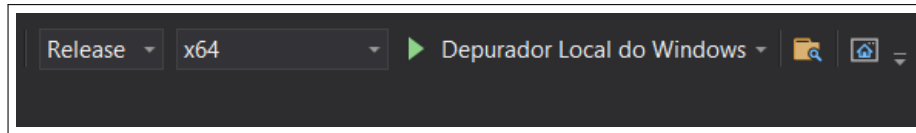


Figura 24: Definição do tipo de *build* e arquitetura no *Visual Studio*

- (c) No painel direito do *Visual Studio*, localize o submódulo *ALL\_BUILD*, clique com o botão direito sobre ele e selecione *Compilar* ou *Build* (vide Figura 25).

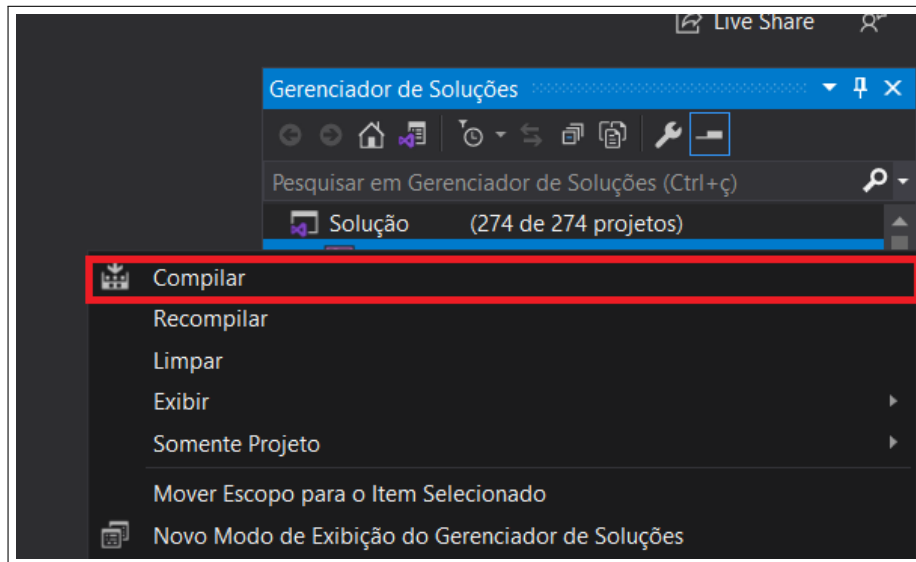


Figura 25: Compilação completa *ALL\_BUILD* do VTK

- (d) Aguarde a conclusão da compilação, que pode levar algum tempo.
- (e) Após a compilação, se não houver erros, localize o item *INSTALL* no painel direito, clique com o botão direito e selecione *Compilar* ou *Build*. Isso copiará os arquivos compilados para o diretório definido na variável *CMAKE\_INSTALL\_PREFIX*.

## Criação de 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. Exemplo: *C:\Program Files\VTK*.

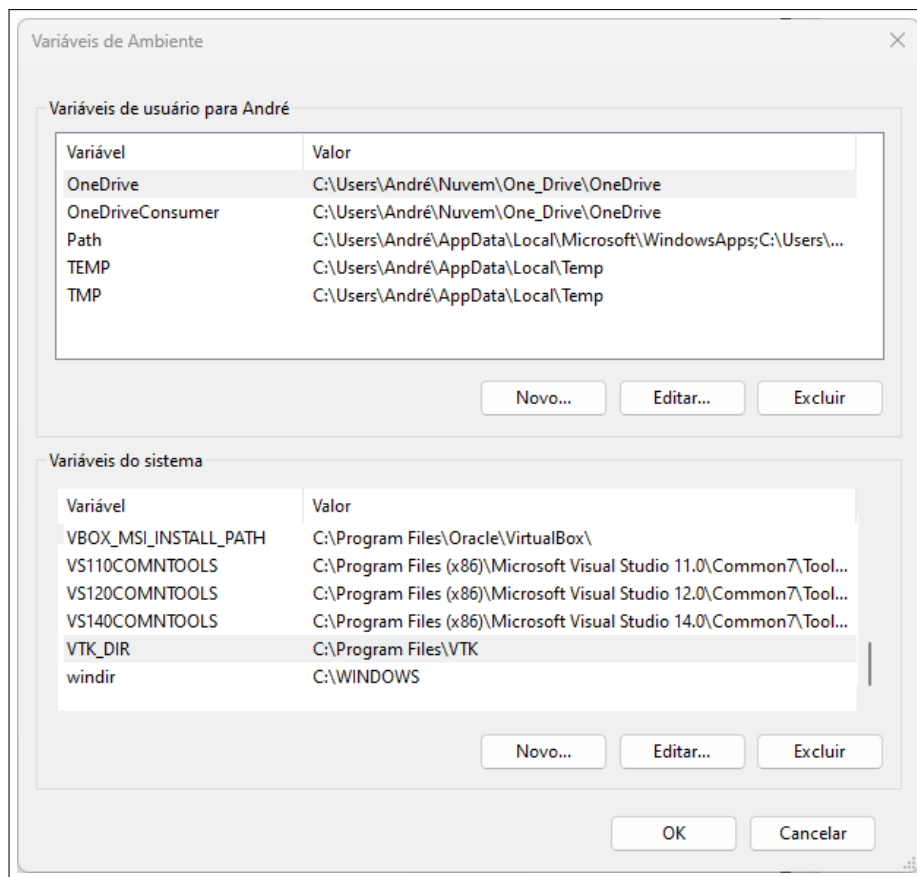


Figura 26: Criação da variável de ambiente *VTK\_DIR*

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

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

`%VTK_DIR%\bin`

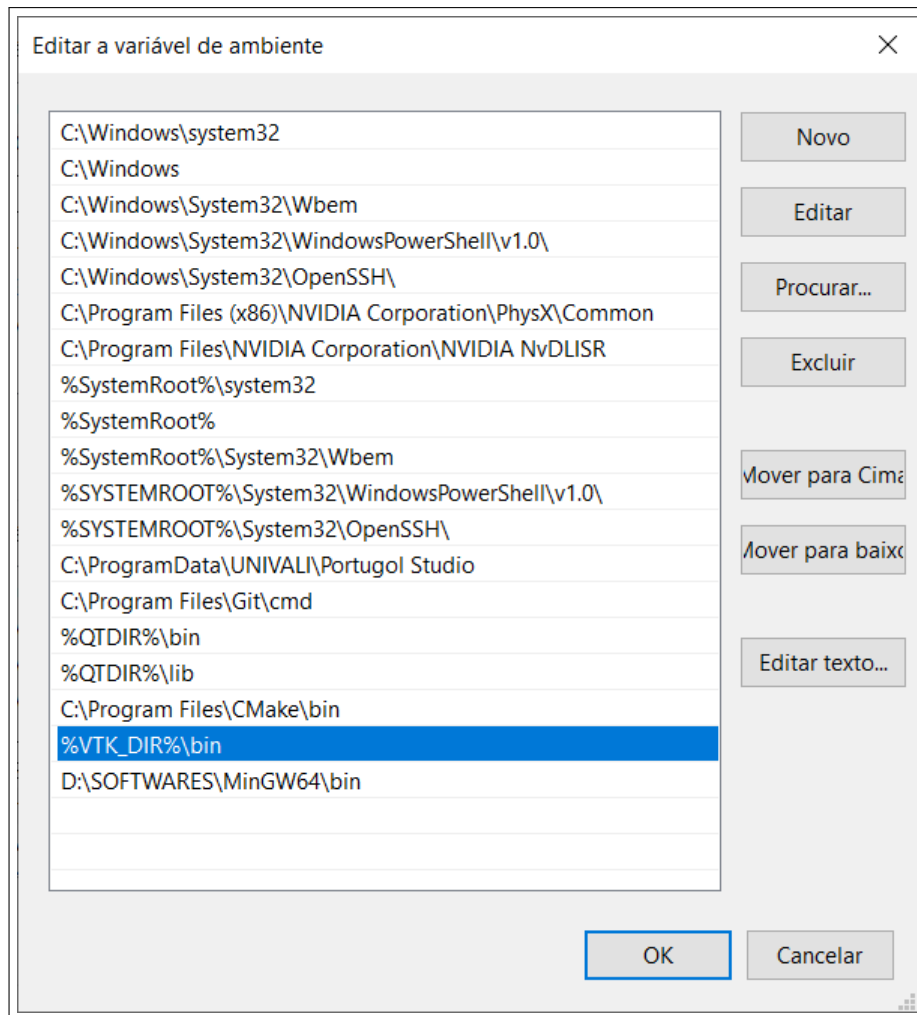


Figura 27: Novas variáveis dentro da lista em *Path*

### 3. \*\*Reinicialização do sistema:\*\*

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

## 1.5 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*.

**Neste processo, você aprenderá a compilar o *Gmsh* from source. Esse processo pode ser demorado. Como alternativa, você pode obter o software já compilado por outros desenvolvedores do SIRERC e instalá-lo manualmente em algum diretório do sistema operacional.**

1. **\*\*Obtenção do código-fonte:\*\***

- (a) 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.tgz>.
- (b) Após o download, extraia o conteúdo do arquivo compactado para um diretório temporário no sistema.

2. **\*\*Preparação para compilação:\*\***

- (a) Crie um diretório vazio chamado *build* dentro do diretório descompactado do código-fonte do *Gmsh* (vide Figura 28).

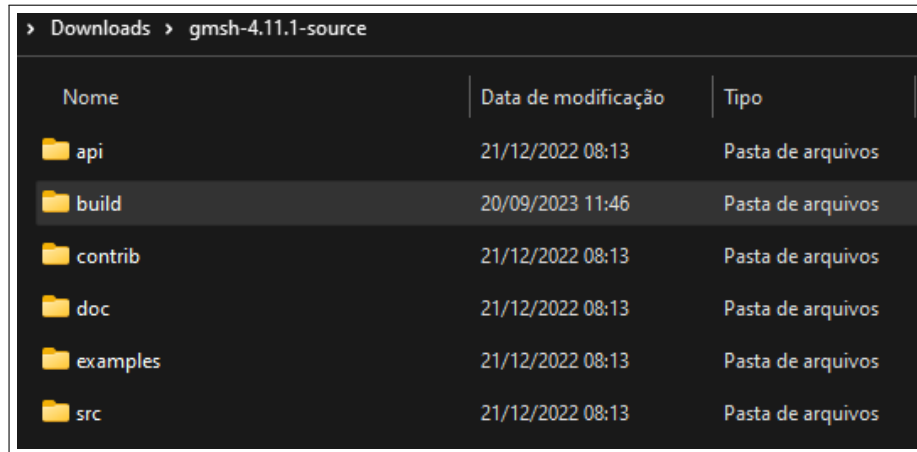


Figura 28: Criação do diretório *build* para compilação do *Gmsh*

- (b) Execute o *CMake* (interface visual). No campo *Where is the source code*, insira o caminho do código-fonte descompactado do *Gmsh*, algo como `C:\Users\username\Downloads\gmsh-4.11.1-source` (vide Figura 29).

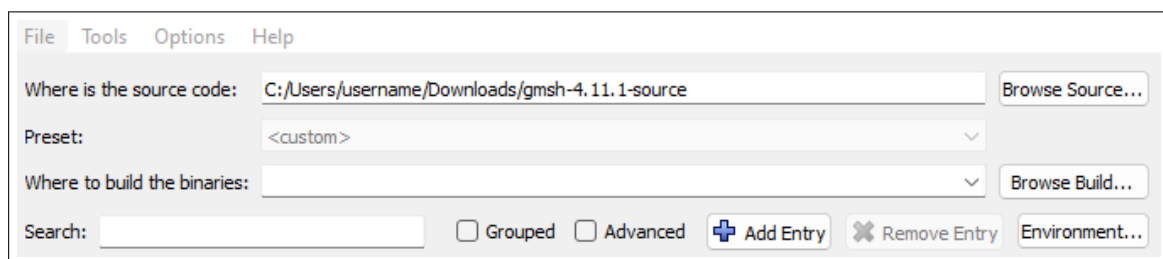


Figura 29: Configuração do diretório de código-fonte no *CMake*

- (c) No campo *Where to build the binaries*, insira o caminho do diretório *build* que você criou anteriormente, por exemplo: `C:\Users\username\Downloads\gmsh-4.11.1-source\build` (vide Figura 30).

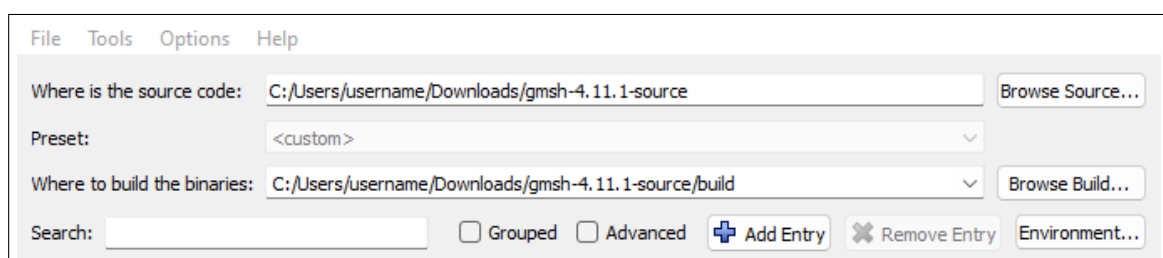


Figura 30: Configuração do diretório de código-fonte e *build* no *CMake*

### 3. \*\*Configuração do CMake:\*\*

- (a) Na parte inferior da tela do *CMake*, clique no botão **Configure**.
- (b) Na nova tela que se abrirá, selecione *Visual Studio 16 2019* como o gerador de projetos e escolha a arquitetura x64 como plataforma. Em seguida, clique no botão **Finish** (vide Figura 31).

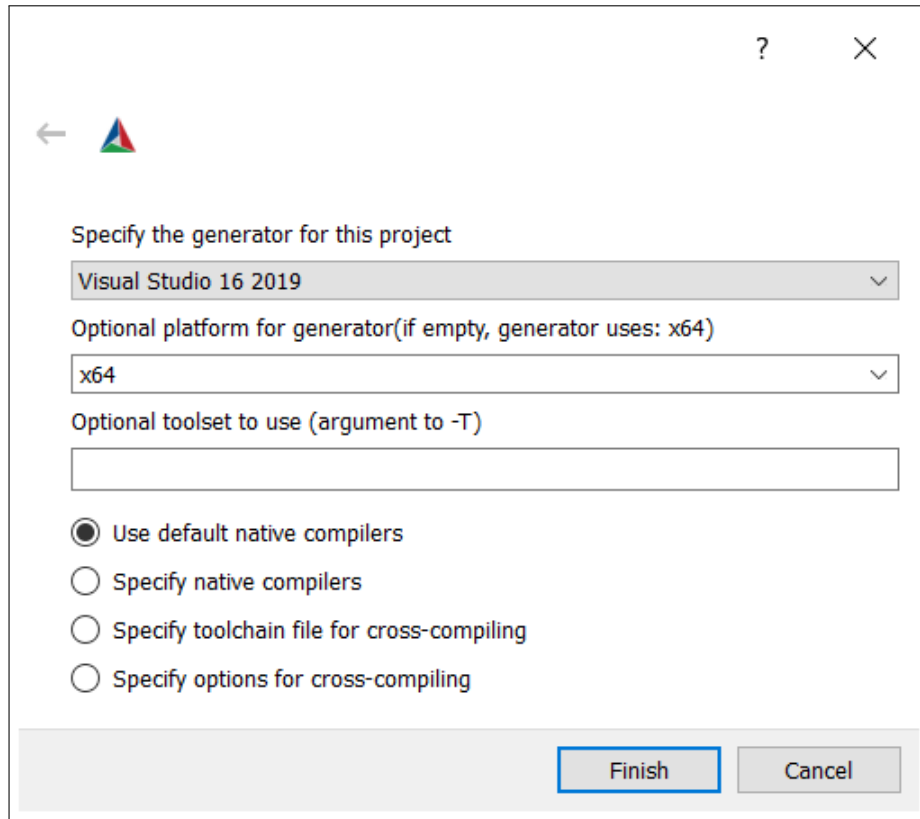


Figura 31: Definição do gerador de projeto e plataforma no *CMake*

- (c) Aguarde a conclusão do processo de configuração. Isso pode levar alguns minutos até que uma lista com várias variáveis em destaque vermelho apareça.
  - (d) Use o campo de busca *Search* para encontrar as seguintes variáveis e configure-as conforme indicado:
    - **ENABLE\_OPENMP = False**
    - **ENABLE\_PRIVATE\_API = True**
    - **ENABLE\_BUILD\_DYNAMIC = True**
    - **ENABLE\_TESTS = False**
  - (e) Opcionalmente, utilize o campo de busca *Search* para encontrar a variável *CMAKE\_INSTALL\_PREFIX*. Por padrão, essa variável está configurada para *C:/Program Files/\gmsh{}*. Você pode manter esse valor ou alterá-lo para outro diretório de sua escolha.
  - (f) Na parte inferior da tela do *CMake*, clique no botão **Configure** novamente.
  - (g) Clique no botão **Generate** para gerar o projeto para o *Visual Studio 2019* no diretório *build*.
  - (h) Aguarde a conclusão do processo de geração do projeto.
- ### 4. \*\*Compilação e instalação no Visual Studio:\*\*

- (a) Se o *Visual Studio* 2019 estiver corretamente instalado, clique no botão **Open Project** para carregar o projeto no *Visual Studio*. Se houver problemas, você pode abrir manualmente o arquivo *.sln* criado no diretório *build*.

**Se o diretório de instalação escolhido estiver em um dos diretórios protegidos do Windows, como C:/Program Files/, você precisará executar o *Visual Studio* como administrador para realizar a instalação. Caso contrário, ocorrerá um erro de permissão.**

- (b) Após o carregamento do projeto no *Visual Studio*, mude o tipo de *build* para *Release* na parte superior da interface e mantenha a arquitetura como *x64* (vide Figura 32).

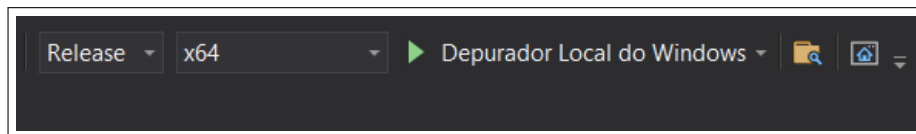


Figura 32: Definição do tipo de *build* e arquitetura no *Visual Studio*

- (c) No painel direito do *Visual Studio*, localize o submódulo *ALL\_BUILD*, clique com o botão direito sobre ele e selecione *Compilar* ou *Build* (vide Figura 33).

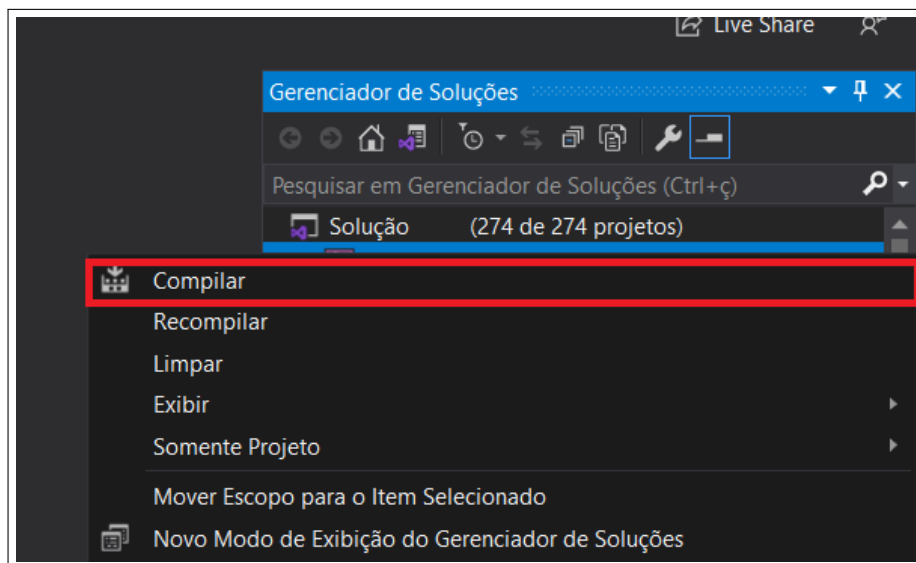


Figura 33: Compilação completa *ALL\_BUILD* do *Gmsh*

- (d) Aguarde a conclusão da compilação, que pode levar algum tempo.
- (e) Após a compilação, se não houver erros, localize o item *INSTALL* no painel direito, clique com o botão direito e selecione *Compilar* ou *Build*. Isso copiará os arquivos compilados para o diretório definido na variável *CMAKE\_INSTALL\_PREFIX*.

## Variáveis de Ambiente do *Gmsh*

### 1. \*\*Criação das variáveis GMSH\_INC e GMSH\_LIB:\*\*

- (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 **GMSH\_INC**. O valor deve ser o caminho do diretório de *headers* do GMSH. Exemplo: `C:\Program Files\gmsh\include`.
- (e) Em seguida, crie outra variável chamada **GMSH\_LIB**. O valor deve ser o caminho do diretório de *libs* do GMSH. Exemplo: `C:\Program Files\gmsh\lib`.

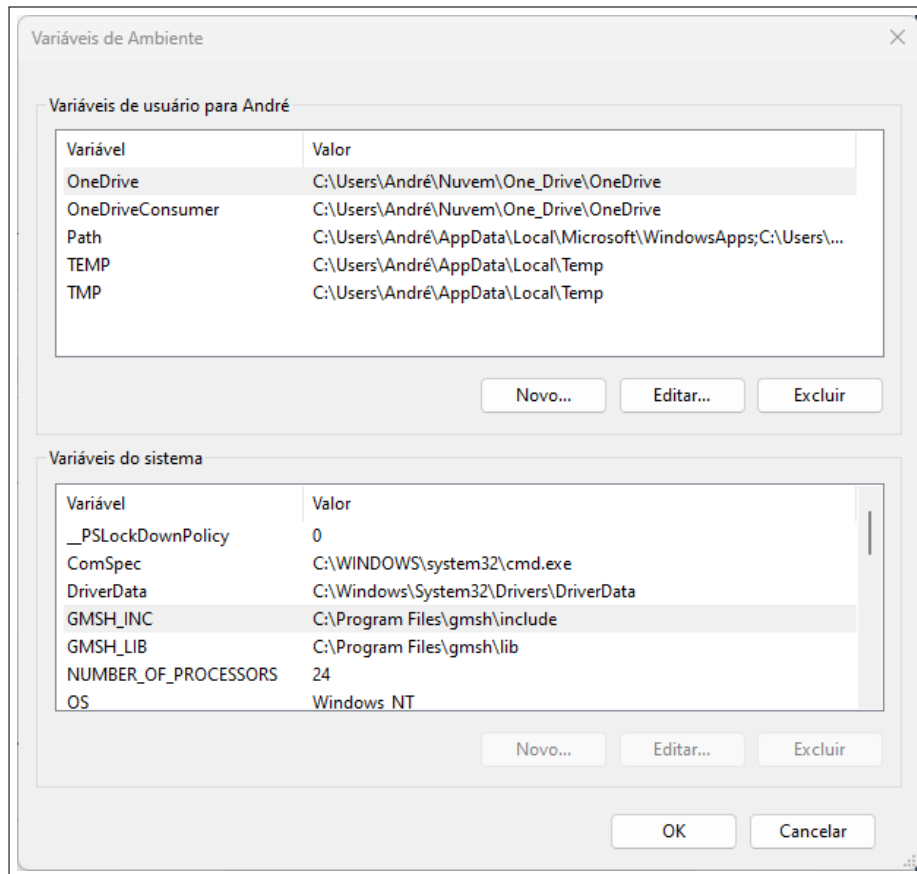


Figura 34: Criação das variáveis de ambiente *GMSH\_INC* e *GMSH\_LIB*

## 2. **\*\*Reinicialização do sistema:\*\***

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

## 1.6 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/sirercita/SIRERC\\_ProjetoPetrobras](https://github.com/sirercita/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 35).

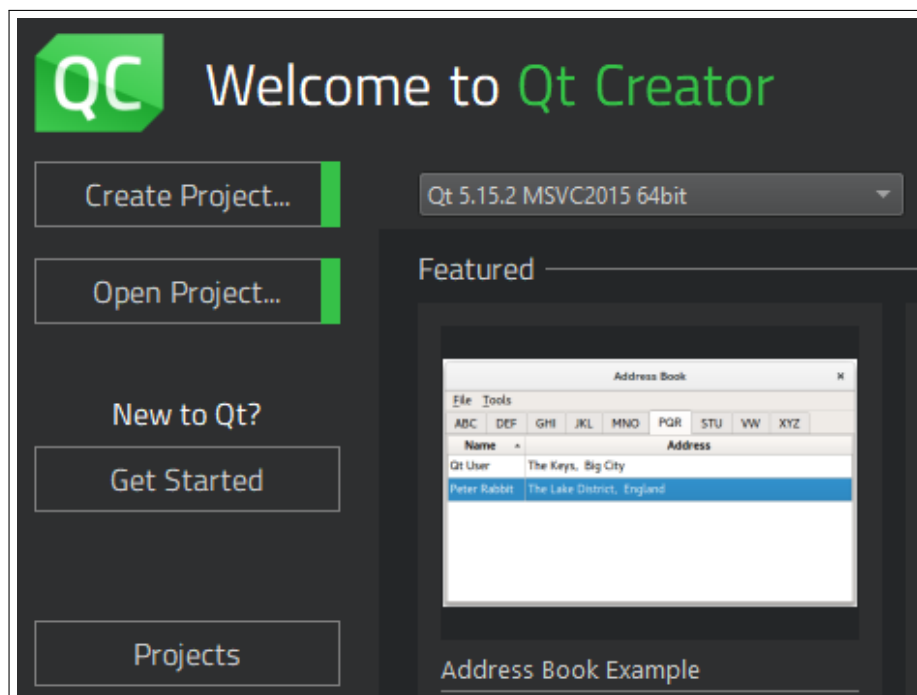



Figura 35: Tela inicial do *Qt Creator* e escolha de versão

- (b) Clique no botão , 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 36). Esse alerta pode ser ignorado clicando em **OK**. Ele aparecerá apenas uma vez.

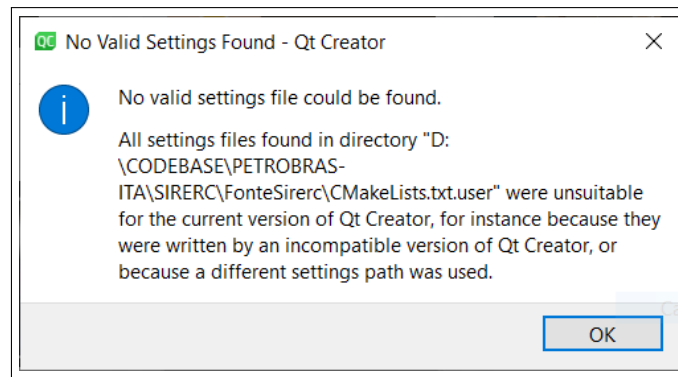


Figura 36: Alerta de primeira utilização do *Qt Creator*

- (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 37) e clique em **Configure Project** na parte inferior da tela.

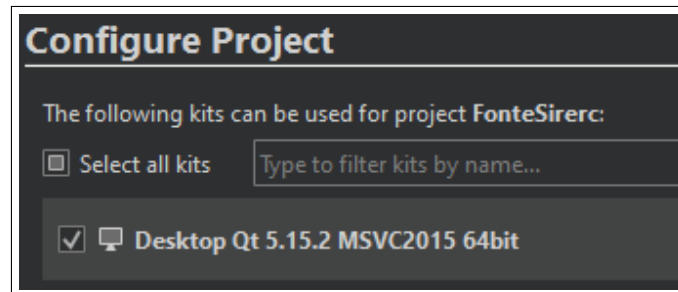


Figura 37: Seleção do tipo de compilador e plataforma no *Qt Creator*

## 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 38). 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*.

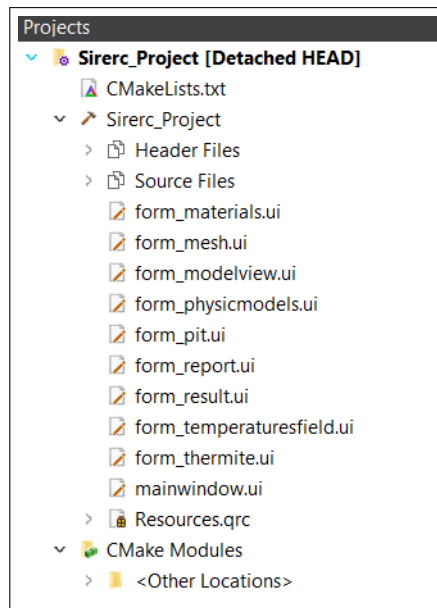


Figura 38: Árvore de diretórios do SIRERC

- (b) Na parte inferior esquerda do *Qt Creator*, altere o tipo de *build* de *Debug* para *Release* e aguarde a reconfiguração (vide Figura 39).

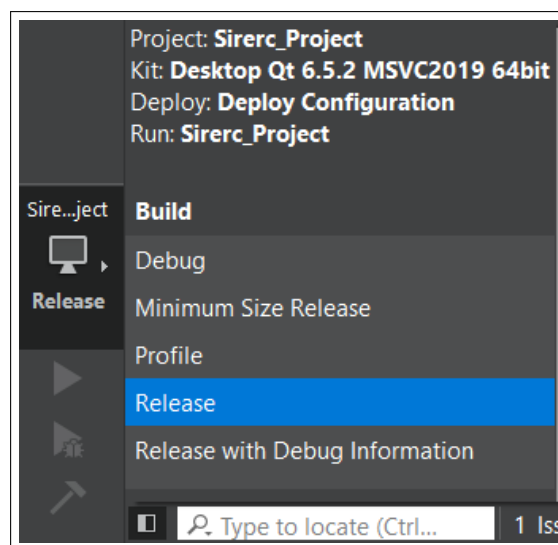



Figura 39: Mudança de *build* para *Release* no *Qt Creator*

## 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 40).

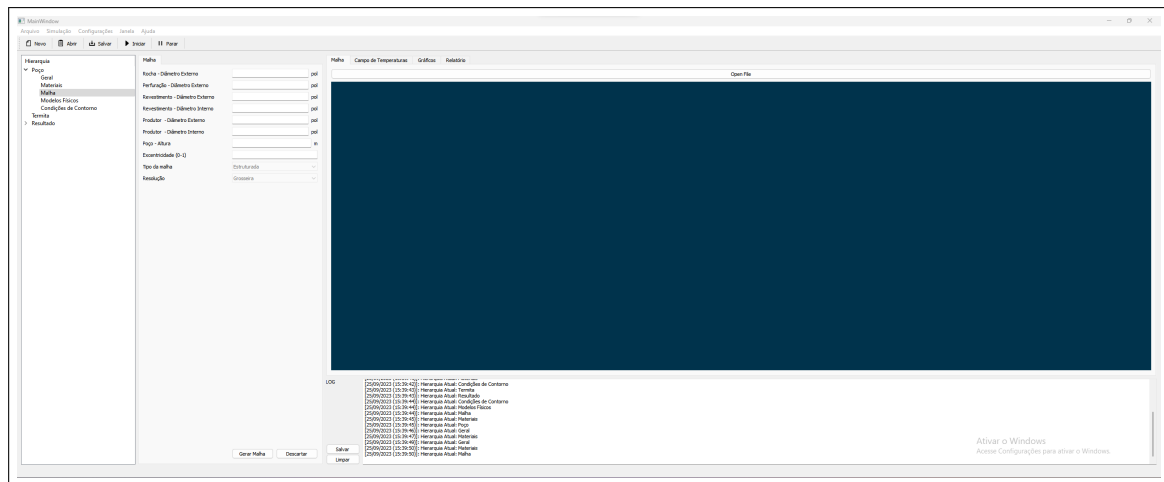


Figura 40: SIRERC em execução

## 1.7 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:\*\*

- 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 41).

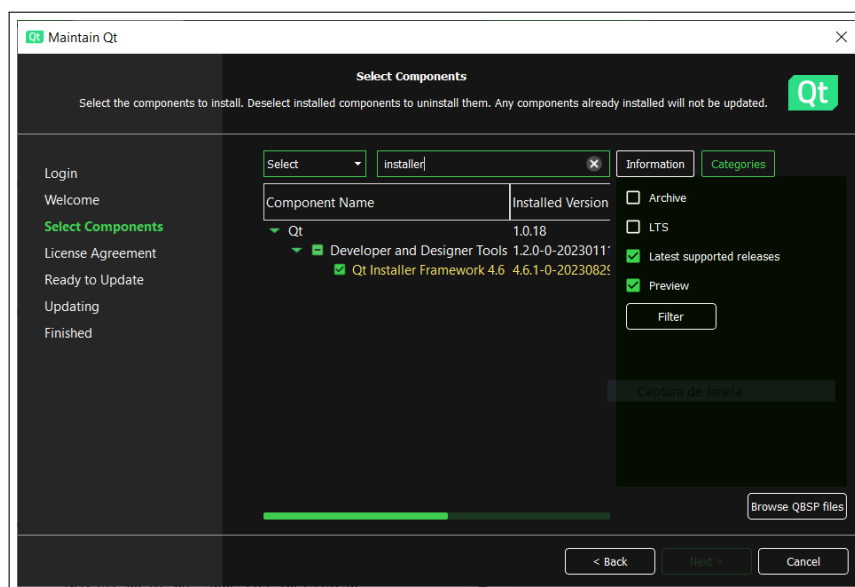
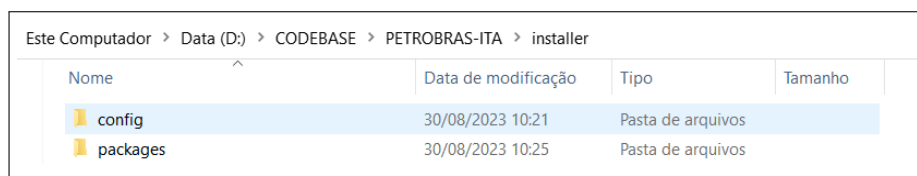


Figura 41: Instalação do componente QtInstallerFramework

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

- Após a instalação, o QtInstallerFramework estará disponível no diretório  $\${QtDirectory} / \text{Tools} / \text{QtInstallerFramework}$ , onde  $\${QtDirectory}$  representa o caminho do sistema de arquivos onde o Qt está instalado.

- (b) Crie um diretório temporário em qualquer local do sistema de arquivos para preparar o SIRERC para *deployment*.
- (c) Dentro deste diretório, crie dois subdiretórios chamados `config` e `packages`, ambos inicialmente vazios (vide Figura 42).



Nome	Data de modificação	Tipo	Tamanho
config	30/08/2023 10:21	Pasta de arquivos	
packages	30/08/2023 10:25	Pasta de arquivos	

Figura 42: Diretório inicial para geração do instalador

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

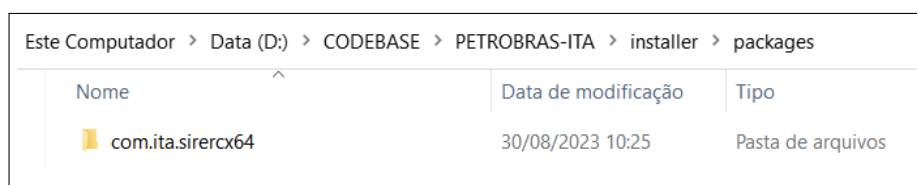
- (a) No diretório `config`, crie um arquivo chamado `config.xml` com o seguinte conteúdo:

```
<?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>
```

- (b) 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 43).



Nome	Data de modificação	Tipo
com.ita.sirercx64	30/08/2023 10:25	Pasta de arquivos

Figura 43: 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 44). O primeiro conterá executáveis, bibliotecas e outros arquivos necessários para o funcionamento do SIRERC. O segundo conterá informações complementares do *package*.

Este Computador > Data (D:) > CODEBASE > PETROBRAS-ITA > installer > packages > com.ita.sirercx64	
Nome	Data de modificação
data	30/08/2023 10:30
meta	30/08/2023 10:28

Figura 44: 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 64Bits</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 45).

Este Computador > Data (D:) > CODEBASE > PETROBRAS-ITA > installer > packages > com.ita.sirercx64 > meta			
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 45: Arquivos no diretório *meta*

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

- (a) No diretório **data**, crie um subdiretório chamado **win64** (vide Figura 46). Copie para este diretório o arquivo *.exe* gerado pelo *Qt Creator* ou outra IDE usada para desenvolver o SIRERC (vide Figura 47). Este arquivo normalmente está localizado nos diretórios de build criados durante o desenvolvimento.


Este Computador > Data (D:) > CODEBASE > PETROBRAS-ITA > installer > packages > com.ita.sirercx64 > data			
Nome	Data de modificação	Tipo	Tamanho
 win64	04/09/2023 14:42	Pasta de arquivos	

Figura 46: Criação do subdiretório *win64* dentro de *data*


Este Computador > Data (D:) > CODEBASE > PETROBRAS-ITA > installer > packages > com.ita.sirercx64 > data > win64			
Nome	Data de modificação	Tipo	Tamanho
 Sirerc_Project	04/09/2023 10:09	Aplicativo	8.942 KB

Figura 47: 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 48).

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 48: 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 49).





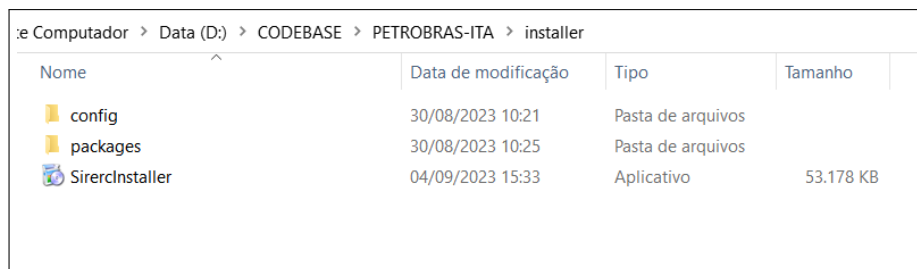


Figura 50: Instalador gerado para o SIRERC

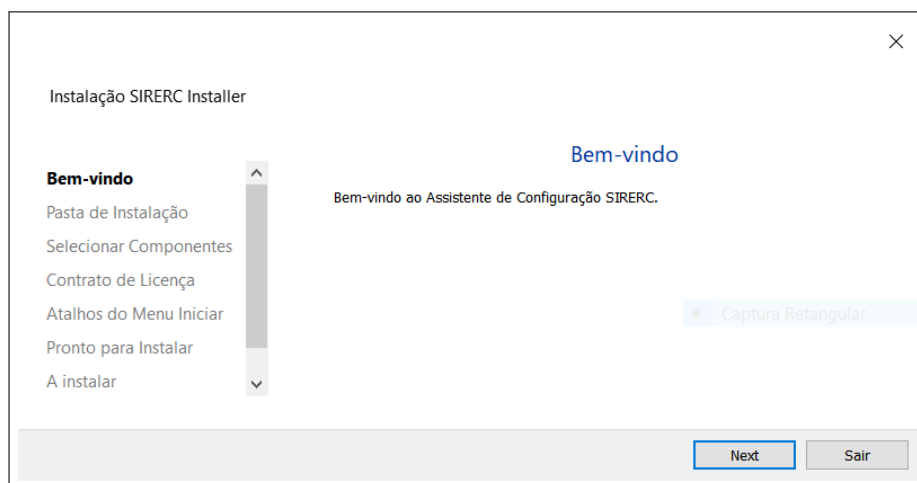


Figura 51: Execução do instalador do SIRERC

## 1.8 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.

### 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.

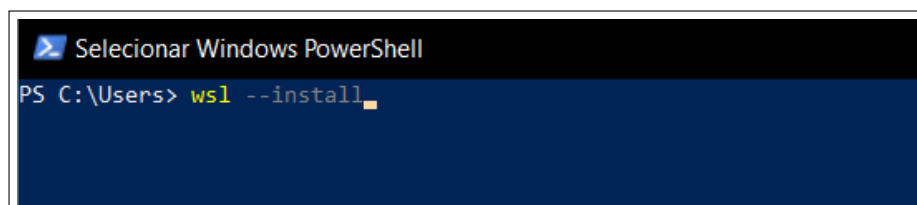


Figura 52: Comando para instalar o WSL no Windows

### 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.

- (b) 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*.

## 2 INSTALAÇÃO DO QT 5.15.2 NO WINDOWS

### 2.1 Requisitos do Sistema

- Perl 5.8 ou posterior
- Python 2.7 ou posterior
- Compilador C++ com suporte ao padrão C++11

Para outros requisitos específicos da plataforma, consulte a seção "Configurando sua máquina" em: [http://wiki.qt.io/Get\\_The\\_Source](http://wiki.qt.io/Get_The_Source)

### 2.2 Windows:

Abra um prompt de comando.

Certifique-se de que as seguintes ferramentas possam ser encontradas no caminho:

- \* Compilador suportado (Visual Studio 2012 ou posterior, MinGW-builds gcc 4.9 ou posterior)
- \* Perl versão 5.12 ou posterior [http://www.activestate.com/activeperl/]
- \* Python versão 2.7 ou posterior [http://www.activestate.com/activepython/]
- \* Ruby versão 1.9.3 ou posterior [http://rubyinstaller.org/]

```
cd <caminho>\<pacote_fonte>
configure -prefix %CD%\qtbases <license> -nomake tests
nmake // jom // mingw32-make
```

Para acelerar o bootstrap do qmake com MSVC, pode ser útil passar `-make-tool jom` na linha de comando do `configure`. Se você não usar `jom`, adicionar `"/MP"` à variável de ambiente `CL` é uma boa ideia.

### 2.3 BUILD!

Um processo típico de `'configure; make'` é usado.

**Algumas opções relevantes do configure (veja `configure -help`):**

- `-release` Compila e vincula o Qt com a depuração desativada.
- `-debug` Compila e vincula o Qt com a depuração ativada.
- `-nomake tests` Desativa a construção de testes para acelerar a compilação.
- `-nomake examples` Desativa a construção de exemplos para acelerar a compilação.
- `-confirm-license` Reconhece automaticamente a licença LGPL 2.1.

**Exemplo para uma construção de release:**

(adapte o parâmetro `'-jN'` conforme necessário para o seu sistema)

```
./configure -prefix $PWD/qtbases <license>
make -j4
```

## Exemplo para uma construção de desenvolvedor:

(habilita mais autotestes, constrói versão de depuração das bibliotecas, ...)

```
./configure -developer-build <license>
make -j4
```

Veja a saída de `./configure -help` para a documentação sobre várias opções para configurar.

Os exemplos acima irão construir quaisquer módulos Qt5 que tenham sido habilitados por padrão no sistema de construção.

É possível construir módulos selecionados com suas dependências executando um `make module-<foo>`. Por exemplo, para construir apenas `qtdeclarative`, e os módulos dos quais ele depende:

```
./configure -prefix $PWD/qtbase <license>
make -j4 module-qtdeclarative
```

Isso pode economizar muito tempo se você estiver interessado apenas em um subconjunto do Qt5.

## 2.4 Dicas

O repositório de submódulos `qtrepotools` contém scripts úteis para desenvolvedores e engenheiros de liberação. Considere adicionar `qtrepotools/bin` à sua variável de ambiente `PATH` para acessá-los.

A ferramenta `qt5_tool` no `qtrepotools` possui alguns recursos adicionais que podem ser de interesse. Experimente `qt5_tool -help`.

## Construindo o Qt5 a partir do git

1. Install Required Tools Make sure you have the following tools installed:

Visual Studio: Qt recommends using a specific Visual Studio version for compatibility. For *Qt 5.15.2*, you can use Visual Studio 2017 or 2019. During installation, ensure the C++ development components are installed.

<https://visualstudio.microsoft.com/pt-br/vs/older-downloads/>

Perl: Download and install Strawberry Perl from Strawberry Perl. Qt requires Perl for its build system.

Python: Install Python 3 from the official Python website. Make sure Python is added to the `PATH`.

Ruby: Download and install Ruby from [rubyinstaller.org](http://rubyinstaller.org).

Git: Since you've already cloned the repository, ensure Git is available in your `PATH`.

CMake: You'll need CMake to configure and build Qt.

2. Download *Qt 5.15.2* Source

If you haven't already, you can download the *Qt 5.15.2* source code from the official Qt website or use the version you've cloned.

```
git clone git://code.qt.io/qt/qt5.git
cd qt5
git checkout 5.15.2 // compilar exatamente esta versão
```

Now you're all set to proceed with the compilation process. You can follow the steps to configure and build *Qt 5.15.2* as mentioned earlier

3. Configure *Qt 5.15.2*

Open the Developer Command Prompt for Visual Studio with administrative privileges, navigate to the qt5 directory, and run the configure script with the appropriate options.

```
cd C:\...\qt5
perl init-repository // bem demorado. Tenha paciência.
```

Abra um terminal no **Prompt de Comando do Desenvolvedor** Visual Studio.

```
configure.bat -prefix "C:\Users\aluno\Documents\qt5\Qt5.15.2" -platform win32-msvc2019 -re
-prefix
```

```
"C:\...\qt5
```

: The directory where Qt will be installed after building. -platform win32-msvc2019: Specifies Visual Studio 2019 for building. -release: Build a release version. -opensource: Use the open-source license. -confirm-license: Automatically confirm the open-source license agreement. -nomake examples -nomake tests: Skip building examples and tests to speed up the process. -skip qtwebengine: Skips the Qt WebEngine module, which can be complex to build.

The -prefix option in the configure.bat command specifies where Qt will be installed after the build process, so the directory

```
C:\Users\aluno\Documents\qt5\Qt5.15.2
```

doesn't need to exist yet. It will be created during the installation step (nmake install) if it doesn't exist.

Here's what happens: -prefix

```
C:\Users\aluno\Documents\qt5\Qt5.15.2
```

: This tells Qt where to install its built binaries, libraries, and tools once the build process is complete. Qt will automatically create the *Qt 5.15.2* directory during the installation phase. What you should do: You don't need to manually create the directory. When the nmake install command is executed after the build, it will place the files in the specified -prefix directory. So you can safely proceed with the configure.bat command as it is.

To summarize:

Run the

```
init-repository
```

command to initialize the submodules. Use the same configure.bat command with the -prefix option pointing to

```
C:\Users\aluno\Documents\qt\Qt5.15.2
```

. The *Qt 5.15.2* folder will be created automatically during the installation phase.

Run vcvarsall.bat with the Correct Arguments Open the Visual Studio Developer Command Prompt or use the current command prompt where you're running the script.

Run the following command to set up the environment for 64-bit development (for Visual Studio 2022 Community):

```
"C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Auxiliary\Build\vcvarsall.bat" a
```

After Setting Up the Environment: Once you have set up the environment, you can proceed with your Qt build:

Navigate to your Qt directory:

```
cd C:\Users\aluno\Documents\qt5
```

Run the nmake command to start building Qt:

```
cd qtbase
configure.bat
nmake // Tenha paciência. Irá demorar.
nmake install
```

### 3 INSTALAÇÃO DO AMBIENTE LINUX

```
sudo apt-get install ninja-build # For Ubuntu or Debian
```

```
sudo add-apt-repository universe
sudo apt-get update
sudo apt-get install qtbase5-dev qtbase5-dev-tools libqt5charts5 libqt5charts5-dev
```

```
sudo apt-get install build-essential cmake git libgl1-mesa-dev libxt-dev
sudo apt-get install qtchooser qt5-qmake
```

```
sudo apt-get install gmesh libgmsh-dev
```

```
// clonando, compilando e instalando o Gmesh
% git clone https://gitlab.onelab.info/gmesh/gmesh.git
git clone https://github.com/live-clones/gmesh.git
cd gmesh
mkdir build
cd build
cmake ..
make -j$(nproc)
sudo make install
```

First, remove all the VTK 9.1.0 packages from your system.

You can use the following command to purge VTK 9.1.0 and its related packages:

```
sudo apt-get purge 'libvtk9*' vtk9 python3-vtk9
sudo apt-get autoremove
sudo apt-get clean
```

```
// clonando, compilando e instalando o VTK
git clone https://gitlab.kitware.com/vtk/vtk.git
cd vtk
git checkout v9.3.1
mkdir build
cd build
cmake .. -DVTK_QT_VERSION=5 -DVTK_Group_Qt=ON -DVTK_Group_Rendering=ON -DVTK_Group_StandAlone=ON
make -j$(nproc)
sudo make install
```

No Sirerc, a fim de passar a utilizar o VTK 9.3.1

```
rm -rf build/CMakeCache.txt
```

```
cmake ..
```