

RELATÓRIO DE ATIVIDADES

Uma Infraestrutura de e-Science voltada à Gestão da Qualidade de Água na Bacia do Rio Doce

Bolsistas: Henrique Almeida de Oliveira, Luiz Eduardo Marchiori

Orientadores: João Paulo Andrade Almeida; Julio Cesar Nardi; Victorio Albani de Carvalho

1. PROPOSTA DE PESQUISA

Durante as 25 primeiras semanas de projeto, até o dia 24 de junho, os bolsistas do projeto, juntamente com os orientadores, decidiram como seriam os passos do projeto. Inicialmente, revisou-se os principais conceitos da Web Semântica e como trabalhar com ontologias, em seguida conheceu-se a biblioteca python RDFlib, logo após se deu início a escrita dos algoritmos de busca e por fim a estruturação do guia prático da ontologia.

Inicialmente, para entender mais sobre o que ia ser feito, foi lido o livro Dados Abertos Conectados (ISOTANI, BITTENCOURT, 2015). Neste livro, possui os conceitos sobre dados abertos, taxonomia, padrões de RDF, triplas, OWL, ontologias e outros, que servem de base para trabalhar nessa área.

Depois de ter entendido a parte teórica inicial, foi feito uma aprofundação no software Protégé (STANFORD, 2014). Neste software, é possível visualizar os arquivos em formato turtle e também criar novas classes, indivíduos, validar dados, criar restrições e também executar outras formas de manipulação na ontologia.

Após essa primeira experiência no manuseio de ontologias, foi iniciada a escrita dos códigos de pesquisa na linguagem Python, tendo como base de dados a ontologia IntegraDoce (ALMEIDA, 2019). Os primeiros algoritmos criados implementavam os seguintes filtros: (i) - Pesquisa por intervalo de data, (ii) - Pesquisa por região geográfica, (iii) - Pesquisa por qualidade medida e (iv) - Pesquisa por qualidade medida.

O guia prático foi organizado no Jupyter Notebook (JUPYTER PROJECT, 2015) para poder ser disponibilizado de uma forma em que o usuário pudesse ver o conteúdo teórico e em seguida testar ele na prática, no código no bloco seguinte. Ele foi finalizado nas semanas após o dia 24 de junho e revisado até o fim do projeto ao final de julho, juntamente com o relatório final de atividades.

De acordo com a estrutura do relatório, é apresentado na seção 2 os conceitos observados no segundo e terceiro parágrafos desta seção de introdução. Na terceira seção, são apresentados os primeiros esforços no uso da biblioteca rdflib e os resultados presentes no guia prático.

2. TRABALHANDO COM ONTOLOGIAS

Como revisado no livro Dados Abertos Conectados (ISOTANI, BITTENCOURT, 2015), uma ontologia visa estabelecer sistematicamente uma linha conceitual, ou seja, em ciências da computação ela é a especificação explícita de uma conceitualização. Dessa forma, uma ontologia permite que tanto uma máquina como um ser humano entendam uma conceitualização feita através de uma ontologia.

Dessa forma, para a Web Semântica, cuja ideia original é estender a Web atual, a descrição dos dados e documentos atuais para que máquinas possam também entender e processar essa vasta coleção de informações é possível através do uso de ontologias, dada sua aplicabilidade para descrever dados conectados.

Durante o projeto, os bolsistas estudaram até a sétima semana no material de referência os livros de Dados Abertos e vídeos sobre ontologias. Após isso fez-se uso da ferramenta Protégé, até o fim da sétima semana, para implementar ontologias de teste, como a ontologia de Pizza (MIROIR, 2018).

2.1. REPRESENTAÇÃO DO CONHECIMENTO

Primeiramente foram revisados os conceitos que norteiam a Web-Semântica, entre eles a importância do uso de dados abertos. O grande volume de dados digitais existentes torna fundamental a sua estruturação (TIM BERNERS LEE, 2001), o método apresentado como solução para este é o framework RDF (Resource Description Framework) apresentado através de um grafo como o da Figura 1.



Figura 1 – Exemplo de um grafo RDF.

Um grafo normalmente tem diversas triplas como a da figura 1, em Dados Abertos Conectados (ISOTANI, BITTENCOURT, 2014) destaca que as informações de recursos, a estrutura <sujeito>, <predicado> e <objeto> do grafo, é responsável principalmente pela descrição semântica da informação dos recursos e é atribuída de uma URI (Uniform Resource Identification), que pode ser:

- URL (Uniform Resource Locator), onde basicamente define um endereço para um determinado recurso através de um protocolo existente;

- URN (Unified Resource Name) representa um nome para um determinado recurso, garantindo unicidade e persistência de forma global mesmo quando o recurso não está disponível;
- IRI (International Resource Identifier) que é uma generalização do URI. Diferente do URI que é baseado nos caracteres ASCII, o IRI amplia o número de caracteres para incluir Chineses (kanji) e Japoneses (hiragana e katakana), bem como os caracteres Cirílicos e Coreanos.

2.2 DESENVOLVIMENTO DE ONTOLOGIAS

Trabalhar por meio de ontologias é um dos modos encontrados de estruturar os dados abertos e, segundo Guimarães e Diniz (2015), a importância de criar uma Ontologia tem relação direta com a criação de uma Rede de Dados Abertos, para que os dados estruturados possam ser publicados e conectados pela web de modo padronizado, incentivando a inovação social facilitado pela web.

Meena Uni e Baskaran (2012) citam que a ontologia é uma especificação formal e explícita de uma conceitualização de um domínio de interesse que é utilizado como estrutura de comunicação entre pessoas e sistema, desse modo, quando se fala de dados abertos conectados, informações existentes na web podem ser combinadas para que um usuário consiga fazer pesquisas no futuro no domínio ao qual pertence a ontologia.

Desse modo, quando se deseja tratar uma grande quantidade de dados abertos, e esse número de dados é maior que a capacidade de interpretação humana, organizar o conhecimento relacionado a esses dados em uma ontologia. Tornar esses dados estruturados e conectados é uma tarefa que envolve gerenciar, coletar, modelar e padronizar para então consumir dados adequadamente, tornando possível automatizar os processos (BITTENCOURT; ISOTANI, 2015).

3. IMPLEMENTAÇÃO

O RDFLib é um pacote Python de código aberto para tratamento de triplas RDF (Resource Description Framework) utilizado durante todo o trabalho. Ele contém a maioria das funções necessárias para trabalhar com RDF como, por exemplo: analisadores e serializadores para RDF/XML, N3, NTriples, N-Quads, Turtle, TriX, Trig e JSON-LD.

Uma interface gráfica pode ser gerada pelo RDFLib, ela pode armazenar implementações para endpoints SPARQL na memória, também suporta consultas SPARQL 1.1 e instruções de atualização de mecanismos de extensão da função SPARQL. Para a

instalação do RDFLib é necessário importar a biblioteca pelo PIP (Package Installer for Python), o instalador de pacotes do Python. Um passo a passo da instalação do pacote pode ser consultado na documentação do RDFLib 6.1.1 em Getting started with RDFLib. (<https://rdflib.readthedocs.io/en/stable/gettingstarted.html>).

Para implementação dos algoritmos do projeto, foi necessário estudar as relações entre os objetos na ontologia. Todo objeto da ontologia doce se relaciona ainda com gUFO, uma ontologia desenvolvida por Almeida (2019) que serve de base para implementação de outras ontologias. Na Figura 2, é apresentado o diagrama utilizado na implementação das triplas das medições Insitu (realizadas no local).

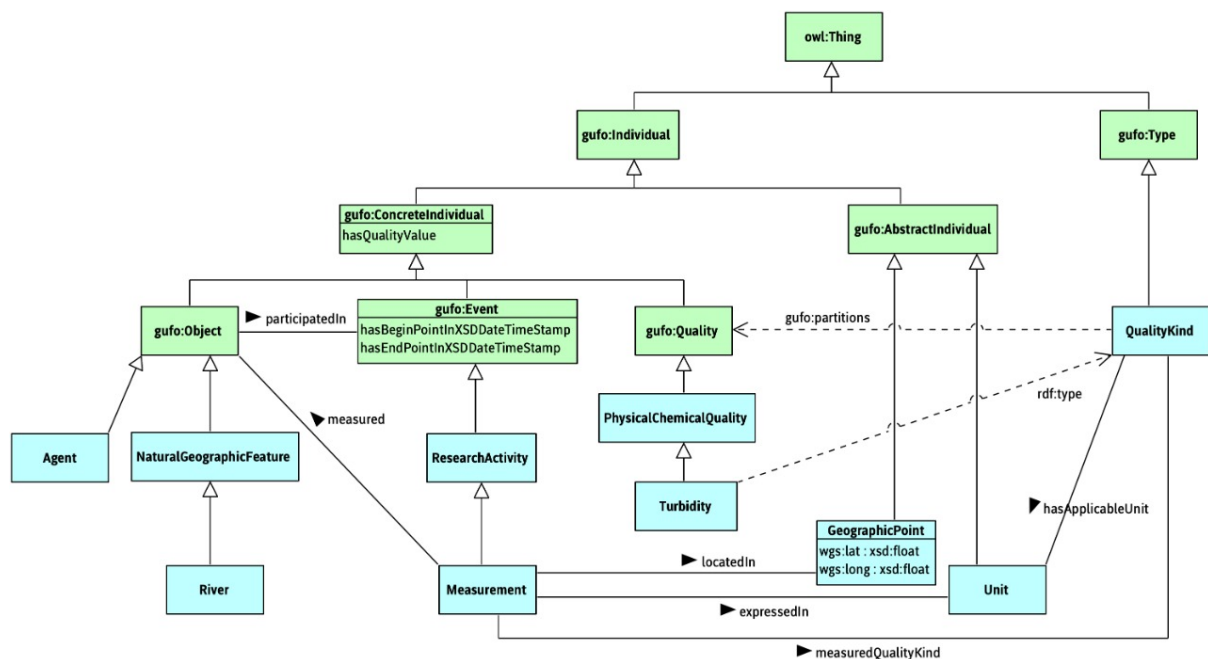


Figura 2 — Diagrama de medições Insitu

Da semana 8 (oito) à semana 24 (vinte e quatro), isto é, do dia 21 (vinte e um) de maio até o dia 17 (dezessete) de junho, foram desenvolvidos os algoritmos de busca para a IntegraDoce com os mecanismos de filtragem indicados por Rabbi (2019). O guia prático foi montado com base neste nas semanas seguintes, além dele foi montado um guia de instalação do Jupyter Notebook em Linux e Windows.

3.1 ALGORITMOS DE BUSCA

Após o aprendizado sobre o Colab Notebook, que foi descartado na segunda semana, foi utilizado como ambiente de desenvolvimento o Jupyter Notebook, com tutorial de instalação no Anexo I (para Windows) e no Anexo II (para Linux), e utilizada a biblioteca python RDFLib, os algoritmos começaram a ser montados levando primeiro em conta filtros

gerais. durante a décima e décima primeira semanas, os filtros elaborados foram estabelecidos pelos orientadores, mas após estas eles passaram a ser realizados com base no artigo de Rabbi (2019).

Os parâmetros de filtragem escolhidos foram: (i) localização geográfica da coleta da amostra de água; (ii) data da coleta da amostra de água; (iii) propriedade medida; e (iv) laboratório responsável pela medição. Os resultados obtidos no projeto para essas pesquisas são apresentados nas figuras de 3 a 6, os algoritmos de busca de cada um desses parâmetros.

```
def pesqTempo (graph, ini, fin):
# cria um grafo
    x = Graph()
# itera no gráfico, pegando o sujeito como predicado do tipo RDF e objeto da relação do tipo
Measurement
    for s1, p1, o1 in graph.triples((None, RDF.type, DOCE.Measurement)) :
# itera no gráfico, utilizando o sujeito obtido no laço anterior, pegando como predicado a relação
# "hasBeginPointInXSDDDateTimeStamp" e tendo como resultado o objeto. (data inicial)
        for s2, p2, o2 in graph.triples((s1, GUFO.hasBeginPointInXSDDDateTimeStamp, None)) :
# transforma a data inicial em string para realizar as comparações e retirar as incompatibilidades
do arquivo
            dateo2 = str(o2)

# sufixo de fuso horário GMT-2 removido por conflito com isoformat
            dateo2 = dateo2.replace('-0200', '')
            dateo2 = dateo2.replace('-0300', '')

# cria uma variavel de comparação utilizando a biblioteca isoformat do python
            dateComp = datetime.fromisoformat(dateo2)

# faz as comparações obtidas com as informações passadas pelo usuário da data inicial
            if ((dateComp.year >= ini.year) and (dateComp.year <= fin.year) and (dateComp.month >=
ini.month) and (dateComp.month <= fin.month) and (dateComp.day >= ini.day) and (dateComp.day <=
fin.day)) :

# itera no gráfico, utilizando o sujeito obtido no laço anterior, pegando como predicado a relação
# "hasEndPointInXSDDDateTimeStamp" (Propriedade de data final) e tendo como resultado o objeto.
(data final)
                for s21, p21, o21 in graph.triples((s1, GUFO.hasEndPointInXSDDDateTimeStamp, None)) :
# transforma a data inicial em string para fazer comparações e tirar incompatibilidades do arquivo
                    dateo2 = str(o21)
# sufixo de fuso horário GMT-2 removido por conflito com isoformat
                    dateo2 = dateo2.replace('-0200', '')
                    dateo2 = dateo2.replace('-0300', '')
# cria uma variavel de comparação utilizando a biblioteca isoformat do python
                    dateComp = datetime.fromisoformat(dateo2)
# faz as comparações obtidas com as informações passadas pelo usuário da data final
                    if ((dateComp.year >= ini.year) and (dateComp.year <= fin.year) and
(dateComp.month >= ini.month) and (dateComp.month <= fin.month) and (dateComp.day >= ini.day) and
(dateComp.day <= fin.day)) :
                        x.add((s1, p1, o1))
                        # laços para adicionar instâncias com sujeito a doce:measurement encontrada
                        for s3, p3, o3 in graph.triples((None, GUFO.participatedIn, s1)) :
                            x.add((s3, p3, o3))
                        for s3, p3, o3 in graph.triples((s1, DOCE.locatedIn, None)) :
                            x.add((s3, p3, o3))
                        for s3, p3, o3 in graph.triples((s1, DOCE.measured, None)) :
```

```

        x.add((s3, p3, o3))
    for s3, p3, o3 in graph.triples((s1, DOCE.measuredQualityKind, None)) :
        x.add((s3, p3, o3))
    for s3, p3, o3 in graph.triples((s1, DOCE.expressedIn, None)) :
        x.add((s3, p3, o3))
    for s3, p3, o3 in graph.triples((s1, GUFO.hasQualityValue, None)) :
        x.add((s3, p3, o3))

    x.add((s21, p21, o21))
    x.add((s2, p2, o2))
# retorna o grafo com todos os resultados obtidos.
return x

```

Figura 3 — Pesquisa em intervalo de data

```

def pesqLocalizacao(graph, latMin, latMax, longMin, longMax):
    # Cria o grafo
    x = Graph()
    # Flag para indicar que não encontrou resultados
    flag = 1
    # Laço para pegar o sujeito e predicado como objeto utilizado a relação GeographicPoint
    for s1, p1, o1 in graph.triples((None, None, DOCE.GeographicPoint)) :
    # Obtém o valor de latitude e longitude
        lat = graph.value(s1, WGS.lat)
        long = graph.value(s1, WGS.long)
    # Comparação utilizada para saber se os dados passados pelo usuário contém na base de arquivo
        if lat < Literal(latMax, datatype=XSD.float) and lat > Literal(latMin, datatype=XSD.float) :
            if long < Literal(longMax, datatype=XSD.float) and long > Literal(longMin,
datatype=XSD.float) :
    # Laço para filtrar os resultados utilizando o predicado da relação locatedIn, e o objeto como o
sujeito do primeiro laço (relação GeographicPoint)
                for s4, p4, o4 in graph.triples((None, DOCE.locatedIn, s1)) :
    # Laço para adicionar o ponto geográfico que tem relação com a medição
                    for s5, p5, o5 in graph.triples((s4, RDF.type, DOCE.Measurement)) :
    # Laço para adicionar o agente que participou da medição.
                        for s6, p6, o6 in graph.triples((None, GUFO.participatedIn, s5)) :
    # Encontrou os resultados, começar a adicionar as triplas com sujeito a doce:measurement encontrada.
                            flag=0
                            x.add((s5, p5, o5))
                            x.add((s6, p6, o6))
                            x.add((s1, p1, o1))
    # Laços para adicionar instâncias com sujeito a doce:measurement encontrada.
                                for s6, p6, o6 in graph.triples((s5, DOCE.measuredQualityKind, None)) :
                                    x.add((s6, p6, o6))
                                    for s7, p7, o7 in graph.triples((s5, DOCE.expressedIn, None)) :
                                        x.add((s7, p7, o7))
                                    for s3, p3, o3 in graph.triples((s1, DOCE.measured, None)) :
                                        x.add((s3, p3, o3))
                                    for s7, p7, o7 in graph.triples((s5, DOCE.hasQualityValue, None)) :
                                        x.add((s7, p7, o7))
                                    for s7, p7, o7 in graph.triples((s5,
DOCE.hasBeginPointInXSDDateTimeStamp, None)) :
                                        x.add((s7, p7, o7))
    # Caso não encontre resultados com as informações passadas, é mostrado na tela que não foram
encontrados resultados.
        if flag :
            print("\nNão foram encontrados resultados")

    # Retorna o grafo resultante.
    return x

```

Figura 4 — Pesquisa em região geográfica

```

def pesqQualidade(exampleTriples, qualk):
# Cria o grafo
    x = Graph()
# Flag para indicar que não encontrou resultados
    flag = 1

# Laço para obter como sujeito, o predicado do tipo RDF e o objeto da relação Measurement
    for s1, p1, o1 in exampleTriple.triples((None, RDF.type, DOCE.Measurement)) :

# Laço utilizado para obter o objeto, tendo como sujeito o resultante do laço anterior e o predicado
# Da relação do tipo measuredQualityKind
    for s2, p2, o2 in exampleTriple.triples((s1, DOCE.measuredQualityKind, None)) :
# Cria uma variável med (medição) e o transforma em string
        med = str(o2)
        med = med.replace(str(DOCE), "")
# Faz a comparação da qualidade passada pelo usuário e da medição obtida
        if qualk in med :
# Laços para adicionar instâncias com sujeito a doce:measurement encontrada
            for s3,p3,o3 in exampleTriple.triples((None, GUFO.participatedIn, s1)) :
# Adiciona no grafo os resultados obtidos
                x.add((s1, p1, o1))
                x.add((s3, p3, o3))
            for s3,p3,o3 in exampleTriple.triples((s1, DOCE.locatedIn, None)) :
                x.add((s3, p3, o3))
            x.add((s2, p2, o2))
            for s3,p3,o3 in exampleTriple.triples((s1, DOCE.expressedIn, None)) :
                x.add((s3, p3, o3))
            for s3, p3, o3 in exampleTriple.triples((s1, DOCE.measured, None)) :
                x.add((s3, p3, o3))
            for s3,p3,o3 in exampleTriple.triples((s1, GUFO.hasQualityValue, None)) :
                x.add((s3, p3, o3))
            for s3,p3,o3 in exampleTriple.triples((s1, GUFO.hasBeginPointInXSDDateTimeStamp,
None)) :
                x.add((s3, p3, o3))
# Caso não encontre resultados com as informações passadas, é mostrado na tela.
        if flag :
            print("\nNão foram encontrados resultados")
# Retorna o grafo resultante
    return x

```

Figura 5 – Pesquisa por qualidade medida

```

def PesqAgent(exampleTriple, pesquisa):
# Cria o grafo
    x = Graph()
# Flag para indicar que não encontrou resultados
    flag = 1
# Laço para filtrar o sujeito, tendo como predicado o tipo RDF, e o objeto a relação Agent
    for s1, p1, o1 in exampleTriple.triples((None, RDF.type , DOCE.Agent)) :
# Cria uma variável de laboratório e transforma em string
        lab = str(s1)
        lab = lab.replace(str(DOCEEX), "")
# faz a comparação se o dado passado pelo usuário está presente na base de dados
        if pesquisa in lab :
# laço para filtrar o objeto, o sujeito é o resultante do primeiro laço e o predicado da relação
participatedIn
            for s2, p2, o2 in exampleTriple.triples((s1, GUFO.participatedIn, None)) :
# cria uma variável de laboratório e transforma em string
                med = str(o2)
                med = med.replace(str(DOCEEX), "")
# adicionar instâncias com sujeito a doce:measurement encontrada

```

```

x.add((o2, RDF.type, DOCE.Measurement))
x.add((s2, p2, o2))
for s3, p3, o3 in exampleTriple.triples((o2, DOCE.locatedIn, None)) :
    x.add((s3, p3, o3))
for s3, p3, o3 in exampleTriple.triples((o2, DOCE.measuredQualityKind, None)) :
    x.add((s3, p3, o3))
for s3, p3, o3 in graph.triples((s1, DOCE.measured, None)) :
    x.add((s3, p3, o3))
for s3, p3, o3 in exampleTriple.triples((o2, DOCE.expressedIn, None)) :
    x.add((s3, p3, o3))
for s3, p3, o3 in exampleTriple.triples((o2, GUF0.hasQualityValue, None)) :
    x.add((s3, p3, o3))
for s3, p3, o3 in exampleTriple.triples((o2, GUF0.hasBeginPointInXSDDateTimeStamp,
None)) :
    x.add((s3, p3, o3))
# Caso não encontre resultados com as informações passadas, é mostrado na tela.
if flag :
    print("\nNão foram encontrados resultados")
# retorna o grafo resultante
return x

```

Figura 6 — Função pesquisa por qualidade medida

Os algoritmos retornam uma nova ontologia em formato turtle que possui somente as medições com os filtros apresentados. Além disso, eles podem ser usados em sequência para realizar filtros a partir de outros filtros, como filtrar as medições em uma região em um período de tempo, em qualquer sequência.

3.2 GUIA PRÁTICO

O Guia prático foi montado com base nos algoritmos apresentados na seção 3.1 e também nas atividades desenvolvidas durante o projeto, de modo que são apresentados exemplos antes de introduzir os algoritmos de interações com a ontologia Integra Doce. Ele apresenta inicialmente o projeto e as tecnologias utilizadas, bem como os seus participantes e objetivos.

Antes de apresentar os algoritmos apresentados na seção anterior, o guia tutorial apresenta a ontologia e suas triplas de grafos de relações, exemplificando o diagrama apresentado na Figura 2. Em seguida, são mostrados os primeiros algoritmos com funções básicas da biblioteca RDFLib, e, por fim, os algoritmos de interação com a Integradoce.

4. CONCLUSÃO

A proposta da pesquisa foi concluída, foi feito um tutorial para usuários que possuem pouca experiência com RDFLib e Python, com o propósito de entender o que cada função faz, o que é o RDFLib, o que são triplas, como navegar nas triplas e como utilizar o RDFLib para iterar nas triplas.

Foi utilizado o Jupyter Notebook para realizar o tutorial e a plataforma GitHub para armazenar o arquivo do notebook (onde contém o tutorial para o usuário), um trecho da base de dados da ontologia e arquivos com tutorial para instalação do notebook no Linux e no Windows.

O projeto apresenta alguns códigos possíveis para esses filtros citados anteriormente. Os algoritmos podem ser adaptados para qualquer outra ontologia desde que seja citado o projeto nas referências. Outros trabalhos poderão ser acrescentados aos materiais de referência no tutorial, dado a frequência de uso do RDFLib utilizando a linguagem Python para manipular ontologias.

ANEXO I – TUTORIAL DE INSTALAÇÃO DO ANACONDA (WINDOWS)

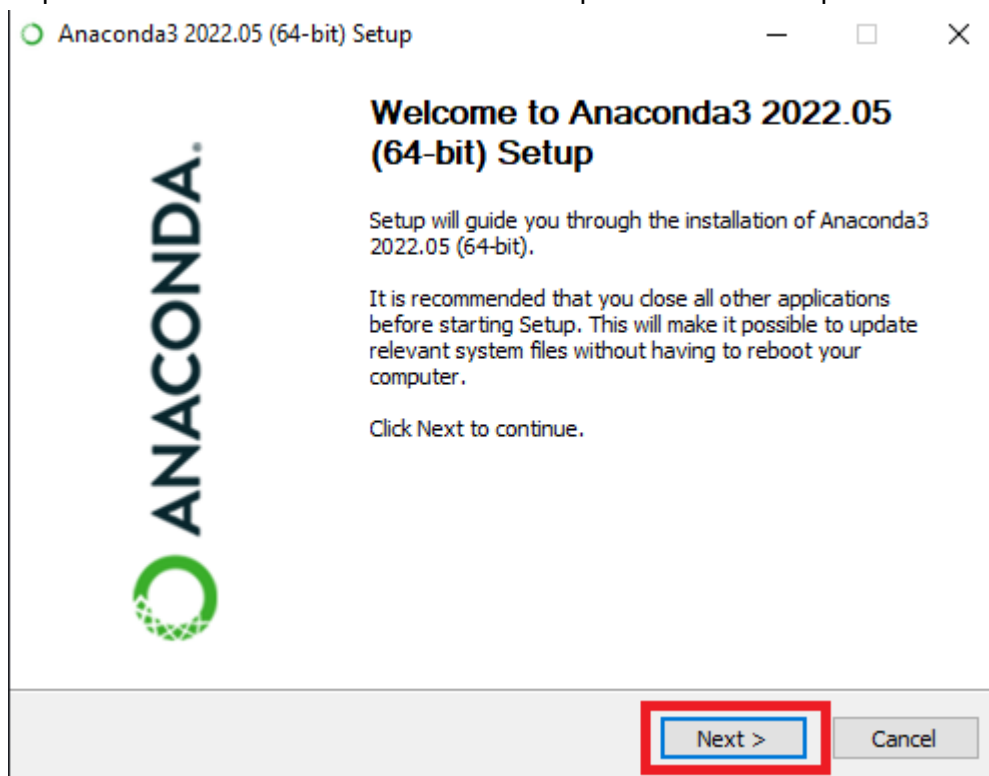
VERSÃO: 1.0

Instalando o Jupyter Notebook no Windows

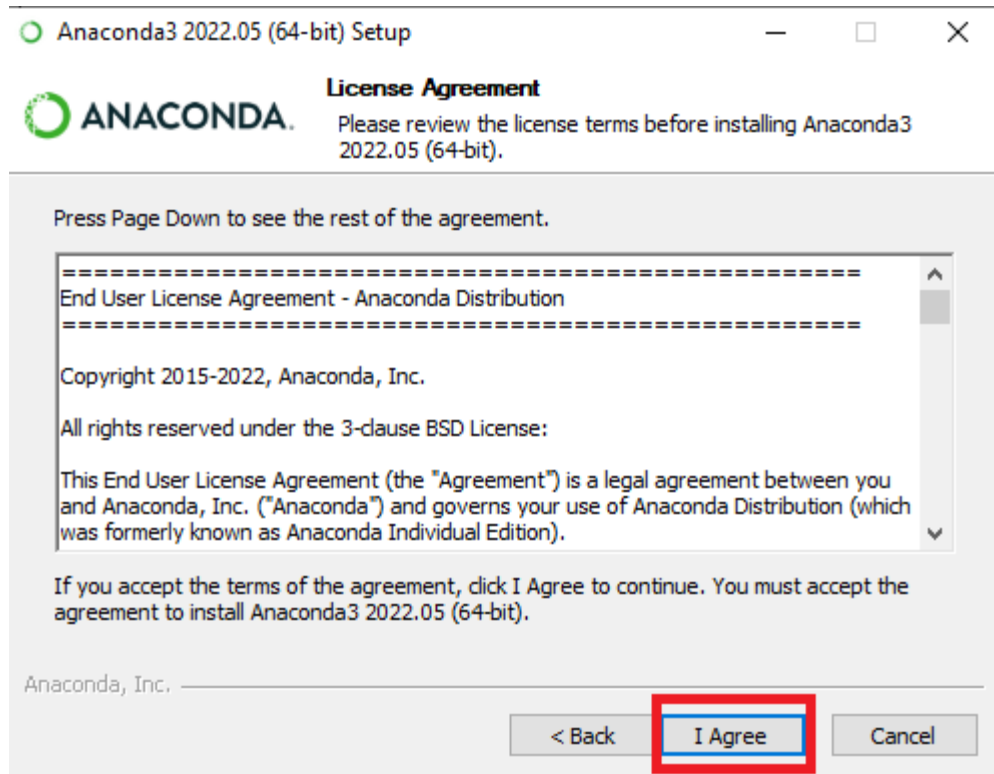
Essa seção inclui instruções como instalar o Jupyter Notebook. Existem várias interfaces de usuário do Jupyter que podem ser usadas, mas a recomendada é utilizar o programa chamado Anaconda para instalar o python e o notebook.

O Anaconda instala o python, o jupyter notebook e outros pacotes utilizados para computação científica e ciência de dados. Para efetuar a instalação deste programa, siga os seguintes passos:

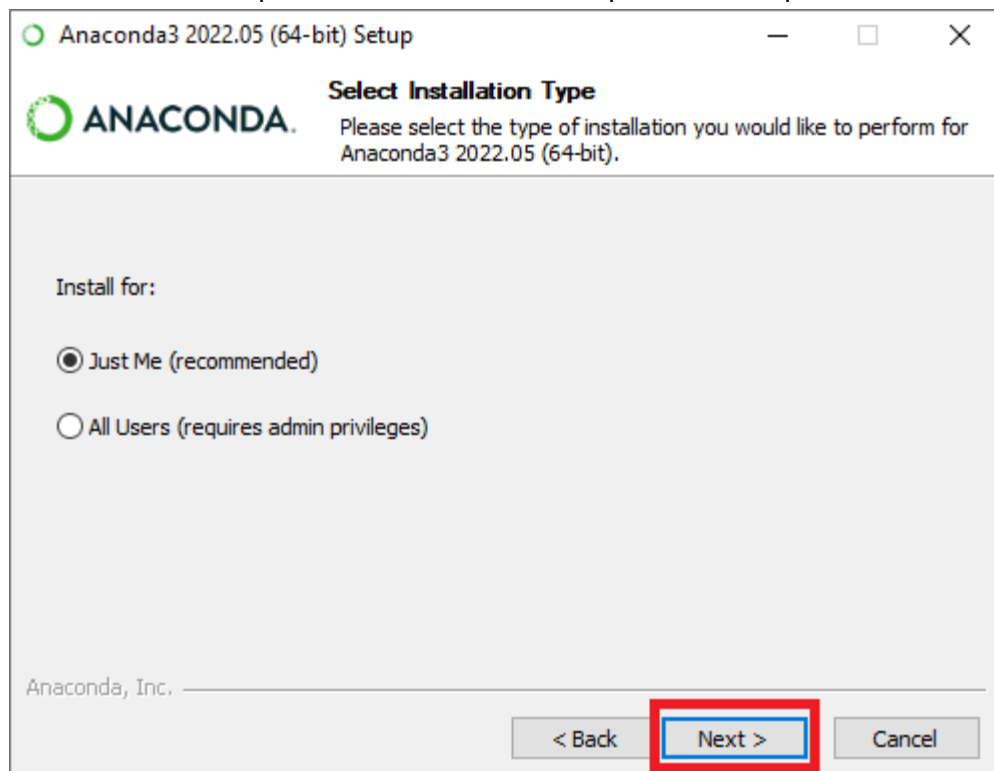
- 1- Faça o download do Anaconda diretamente pelo site ou através do endereço https://repo.anaconda.com/archive/Anaconda3-2022.05-Windows-x86_64.exe – 594MB.
- 2- Depois de ter feito o download execute o arquivo baixado e clique em *Next*.



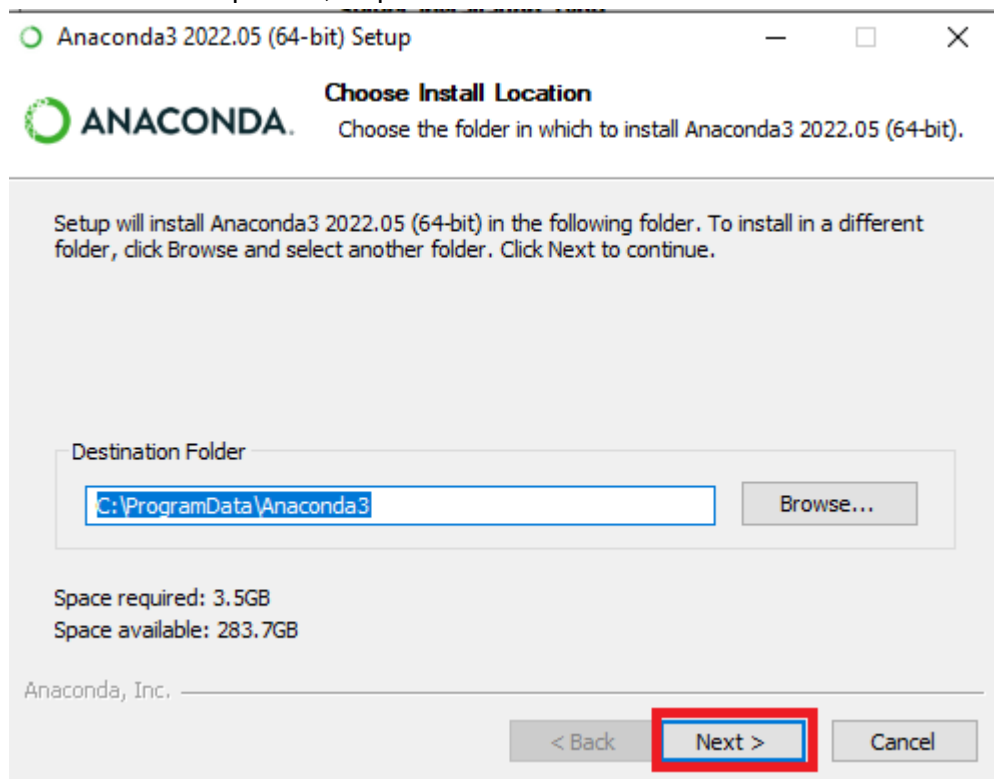
- 3- Se concordar com os termos, clique em *I Agree*.



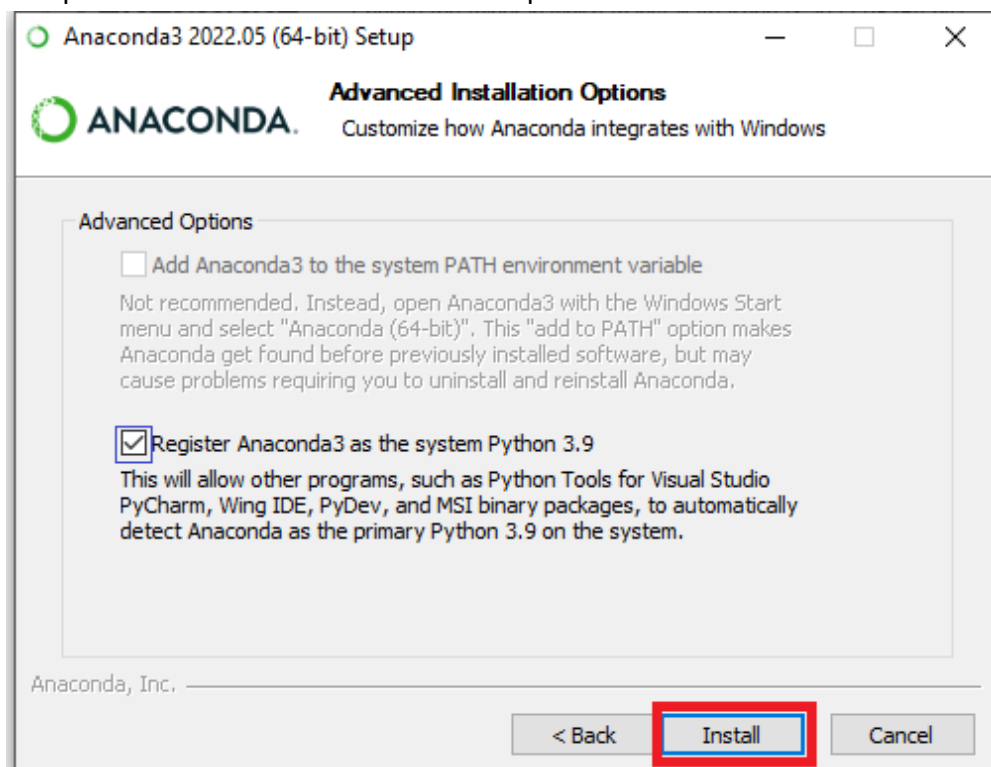
- 4- Você irá escolher se o programa será instalado apenas para o usuário que está sendo utilizado, ou para todos os usuários. Depois disso clique em *Next*.



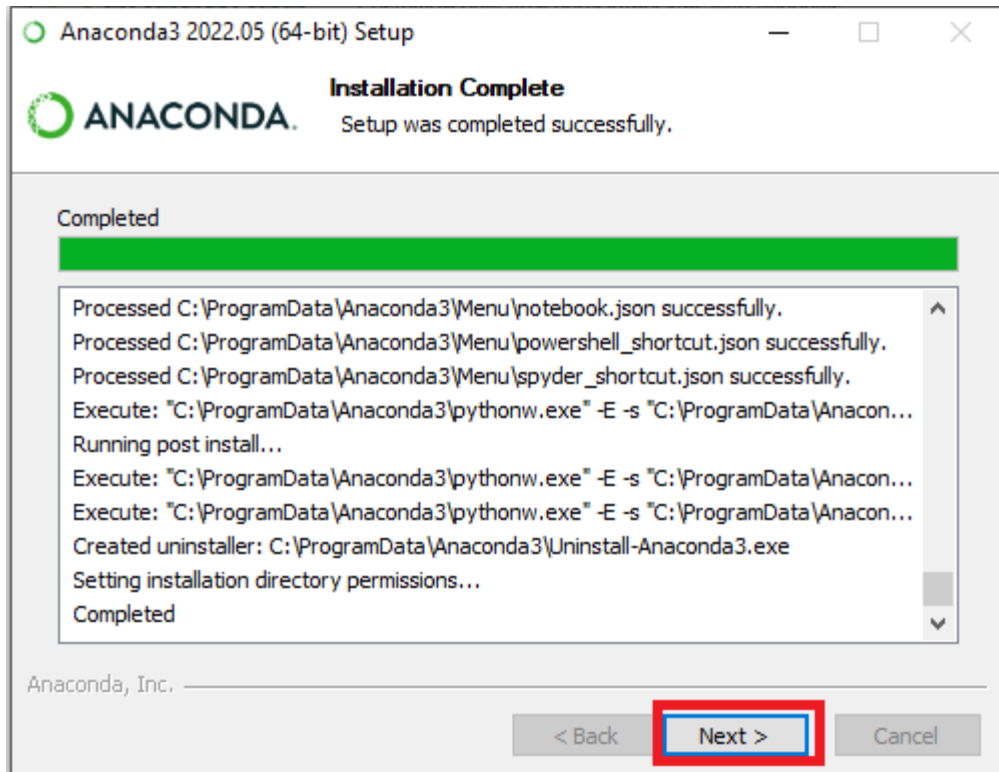
- 5- Selecione a pasta de destino onde irá ocorrer a instalação do Anaconda, caso queira deixar o diretório padrão, clique em *Next*.



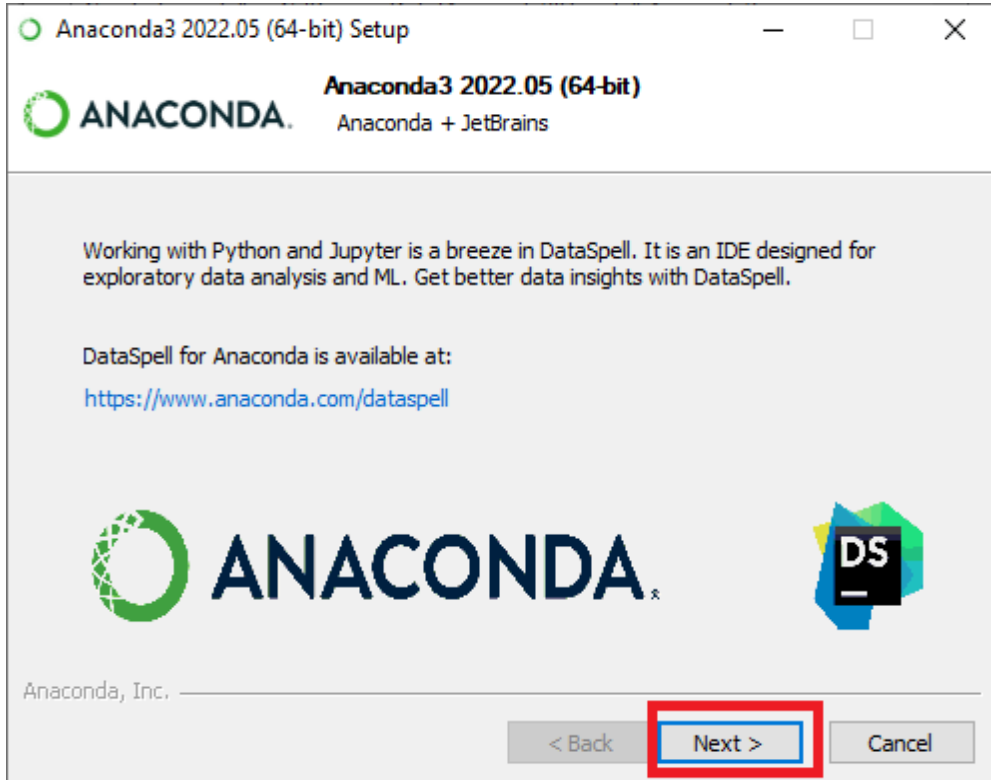
- 6- Marque a caixa destacada em azul e clique em *Install*.



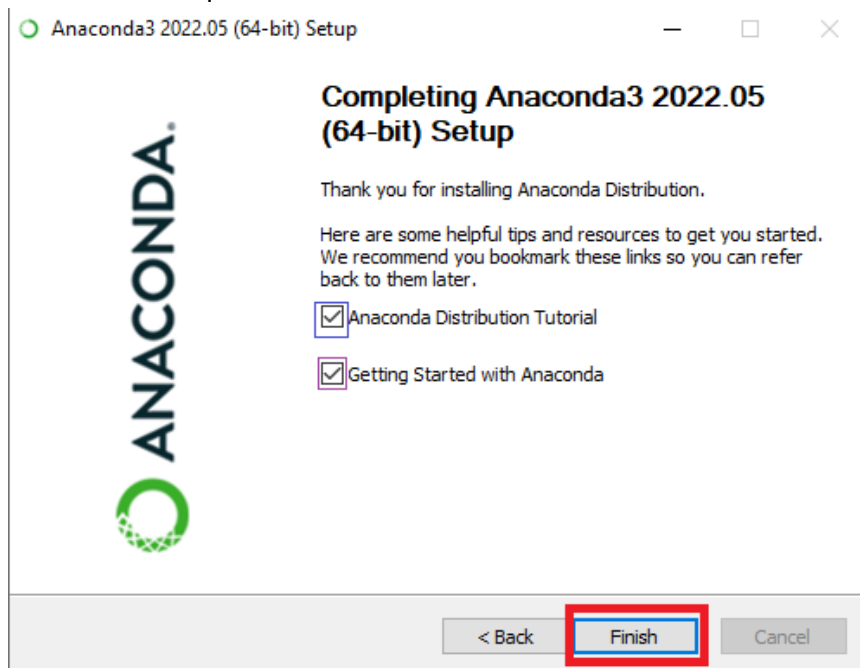
- 7- Depois de acabar o processo de instalação, clique em *Next*.



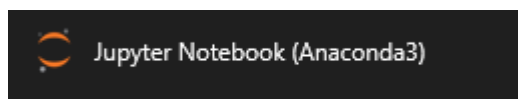
- 8- Clique em *Next*.



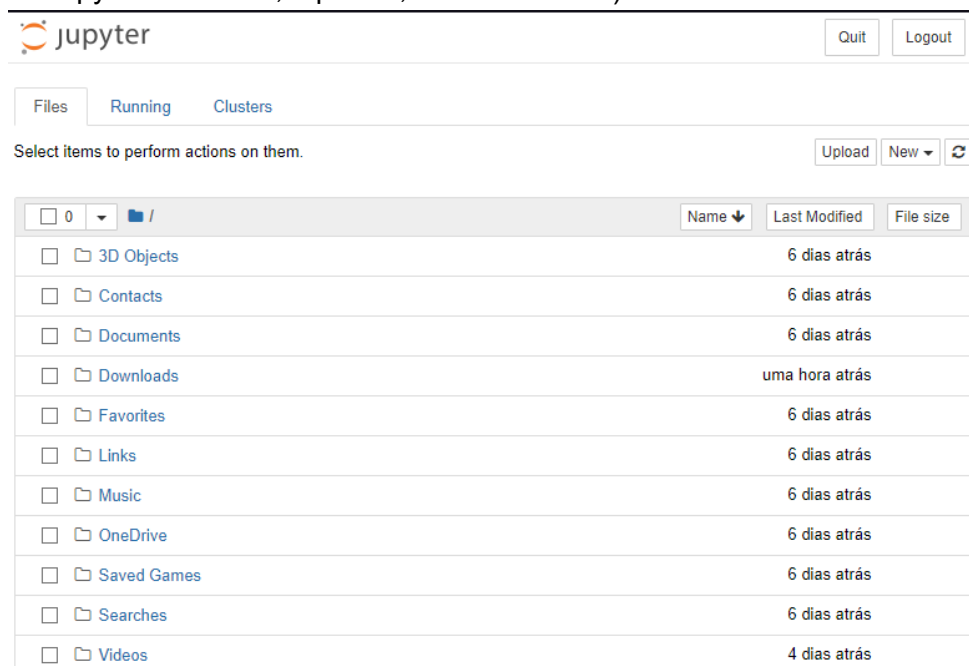
- 9- Caso não queira ver o tutorial do Anaconda, desmarque a caixa destacada de azul. Caso não queira que o Anaconda abra a aba do Getting Started depois de clicar em *Finish*, desmarque a caixa destacada em roxo.



- 10- Abra o arquivo chamado Jupyter Notebook (Anaconda 3)



- 11- Depois disso, irá abrir uma aba no seu navegador padrão com o sistema de arquivos do Jupyter Notebook, e pronto, está instalado :)



ANEXO II – TUTORIAL DE INSTALAÇÃO DO ANACONDA (LINUX)

VERSÃO: 1.0

Instalando o Jupyter Notebook no Linux

Essa seção inclui instruções como instalar o Jupyter Notebook. Existem várias interfaces de usuário do Jupyter que podem ser usadas, mas a recomendada é utilizar o programa chamado Anaconda para instalar o Python e o Notebook.

O Anaconda instala a versão mais recente do python, o jupyter notebook e outros pacotes utilizados para computação científica e ciência de dados. Para efetuar a instalação deste programa, siga os seguintes passos:

- 1- Certifique-se que possui as dependências necessárias para instalação do Anaconda ao executar o comando correto no prompt de acordo com a sua distribuição Linux.

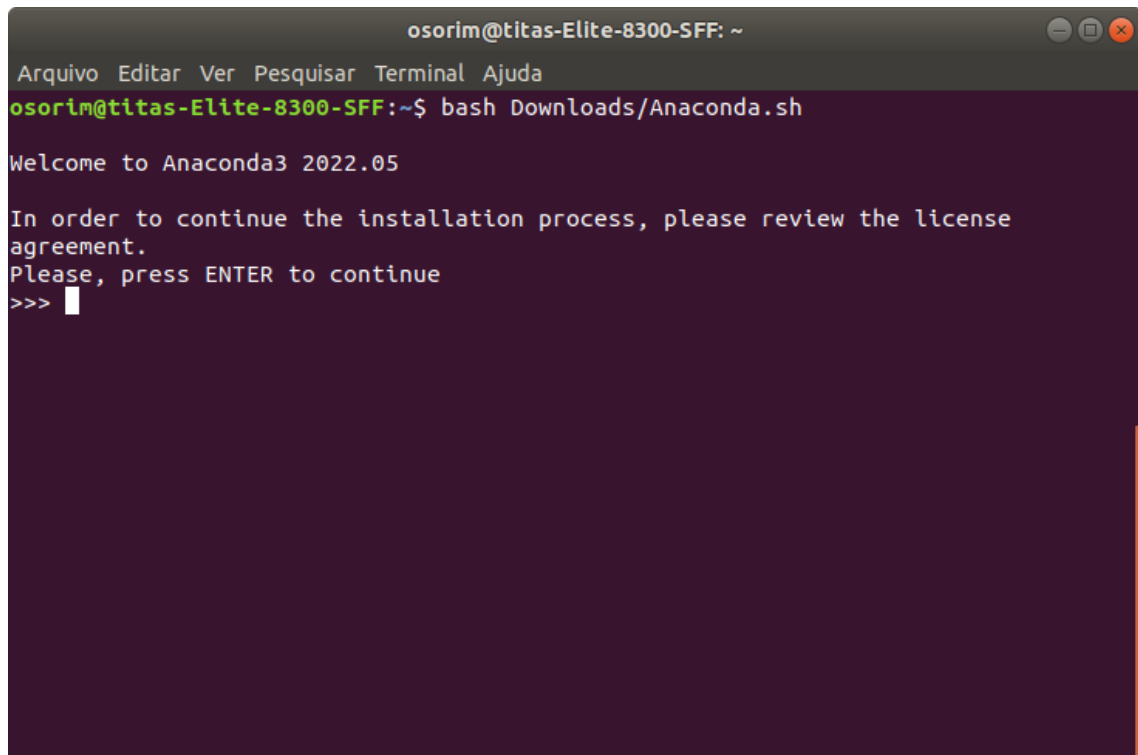
Prerequisites

To use GUI packages with Linux, you will need to install the following extended dependencies for Qt:

Debian	<code>apt-get install libgl1-mesa-glx libegl1-mesa libxrandr2 libxrandr2 libxss1 libxcursor1 libxcomposite1 libasound2 libxi6 libxtst6</code>
RedHat	<code>yum install libXcomposite libXcursor libXi libXtst libXrandr alsa-lib mesa-libEGL libXdamage mesa-libGL libXScrnSaver</code>
ArchLinux	<code>pacman -Sy libxau libxi libxss libxtst libxcursor libxcomposite libxdamage libxfixes libxrandr libxrender mesa-libgl alsa-lib libglvnd</code>
OpenSuse/SLES	<code>zypper install libXcomposite1 libXi6 libXext6 libXau6 libX11-6 libXrandr2 libXrender1 libXss1 libXtst6 libXdamage1 libXcursor1 libxcb1 libasound2 libX11-xcb1 Mesa-libGL1 Mesa-libEGL1</code>
Gentoo	<code>emerge x11-libs/libXau x11-libs/libxcb x11-libs/libX11 x11-libs/libXext x11-libs/libXfixes x11-libs/libXrender x11-libs/libXi x11-libs/libXcomposite x11-libs/libXrandr x11-libs/libXcursor x11-libs/libXdamage x11-libs/libXScrnSaver x11-libs/libXtst media-libs/alsa-lib media-libs/mesa</code>

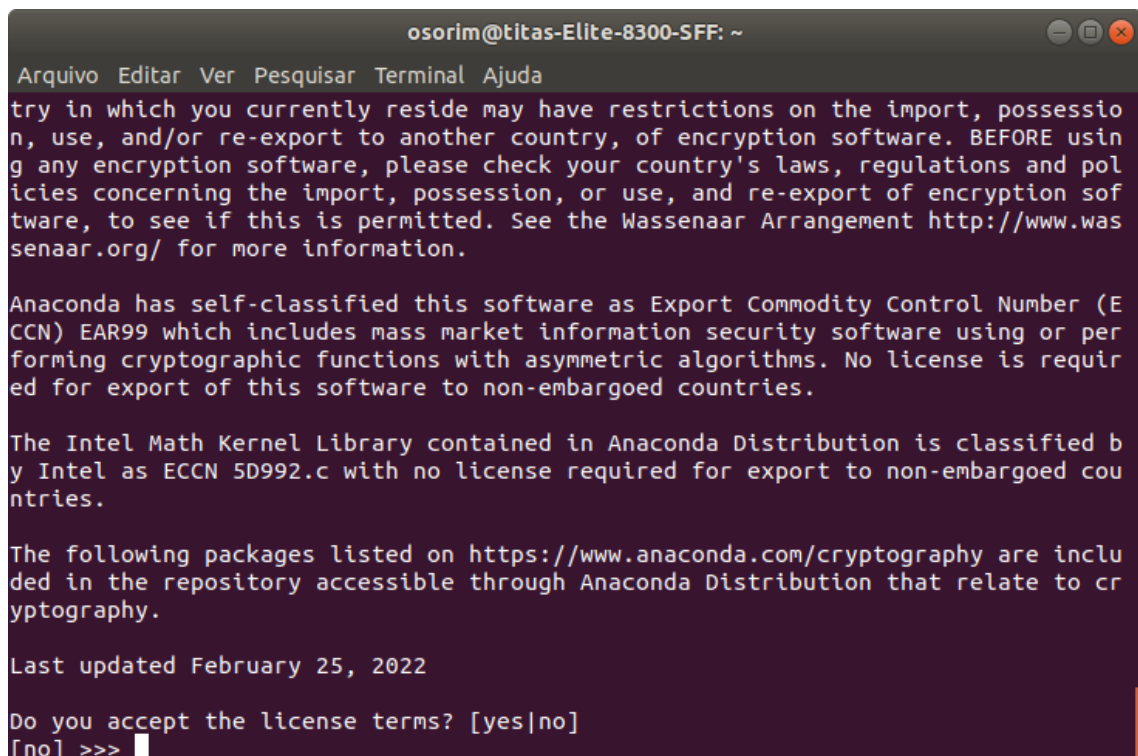
- 2- Após executar o comando, faça download do Anaconda para Linux diretamente do site ou pelo endereço <https://www.anaconda.com/download/#linux>.

- 3- Após fazer o download, execute o comando *bash* com o endereço do arquivo baixado.



```
osorim@titas-Elite-8300-SFF: ~  
Arquivo Editar Ver Pesquisar Terminal Ajuda  
osorim@titas-Elite-8300-SFF:~$ bash Downloads/Anaconda.sh  
  
Welcome to Anaconda3 2022.05  
  
In order to continue the installation process, please review the license  
agreement.  
Please, press ENTER to continue  
>>> 
```

- 4- Após concordar com a licença, basta seguir pressionando a tecla enter. Deve aparecer novamente esse termo, onde deverá ser digitados *yes*.



```
osorim@titas-Elite-8300-SFF: ~  
Arquivo Editar Ver Pesquisar Terminal Ajuda  
try in which you currently reside may have restrictions on the import, possession,  
n, use, and/or re-export to another country, of encryption software. BEFORE using  
any encryption software, please check your country's laws, regulations and policies  
concerning the import, possession, or use, and re-export of encryption software,  
to see if this is permitted. See the Wassenaar Arrangement http://www.wassenaar.org/ for more information.  
  
Anaconda has self-classified this software as Export Commodity Control Number (ECCN)  
EAR99 which includes mass market information security software using or performing  
cryptographic functions with asymmetric algorithms. No license is required for  
export of this software to non-embargoed countries.  
  
The Intel Math Kernel Library contained in Anaconda Distribution is classified by  
Intel as ECCN 5D992.c with no license required for export to non-embargoed countries.  
  
The following packages listed on https://www.anaconda.com/cryptography are included in the repository accessible  
through Anaconda Distribution that relate to cryptography.  
  
Last updated February 25, 2022  
  
Do you accept the license terms? [yes|no]  
[no] >>> 
```

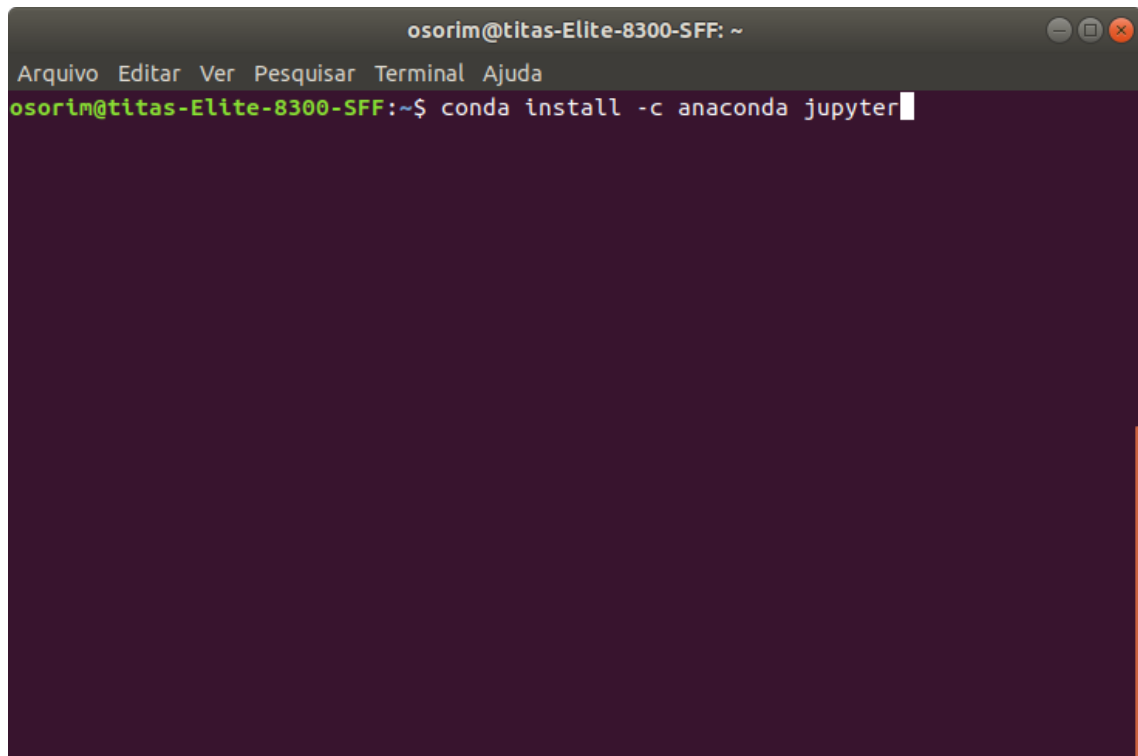

- 5- Após concordar com a licença, você deverá especificar o destino da pasta, que por padrão é a pasta pessoal do usuário.

```
osorim@titas-Elite-8300-SFF: ~  
Arquivo Editar Ver Pesquisar Terminal Ajuda  
forming cryptographic functions with asymmetric algorithms. No license is requir  
ed for export of this software to non-embargoed countries.  
  
The Intel Math Kernel Library contained in Anaconda Distribution is classified b  
y Intel as ECCN 5D992.c with no license required for export to non-embargoed cou  
ntries.  
  
The following packages listed on https://www.anaconda.com/cryptography are inclu  
ded in the repository accessible through Anaconda Distribution that relate to cr  
yptography.  
  
Last updated February 25, 2022  
  
Do you accept the license terms? [yes|no]  
[no] >>> yes  
  
Anaconda3 will now be installed into this location:  
/home/osorim/anaconda3  
  
- Press ENTER to confirm the location  
- Press CTRL-C to abort the installation  
- Or specify a different location below  
  
[/home/osorim/anaconda3] >>> 
```

- 6- Quando a instalação terminar, o instalador vai pedir se você deseja inicializar o Anaconda3, você poderá digitar *no* já que nosso objetivo é instalar o Jupyter.

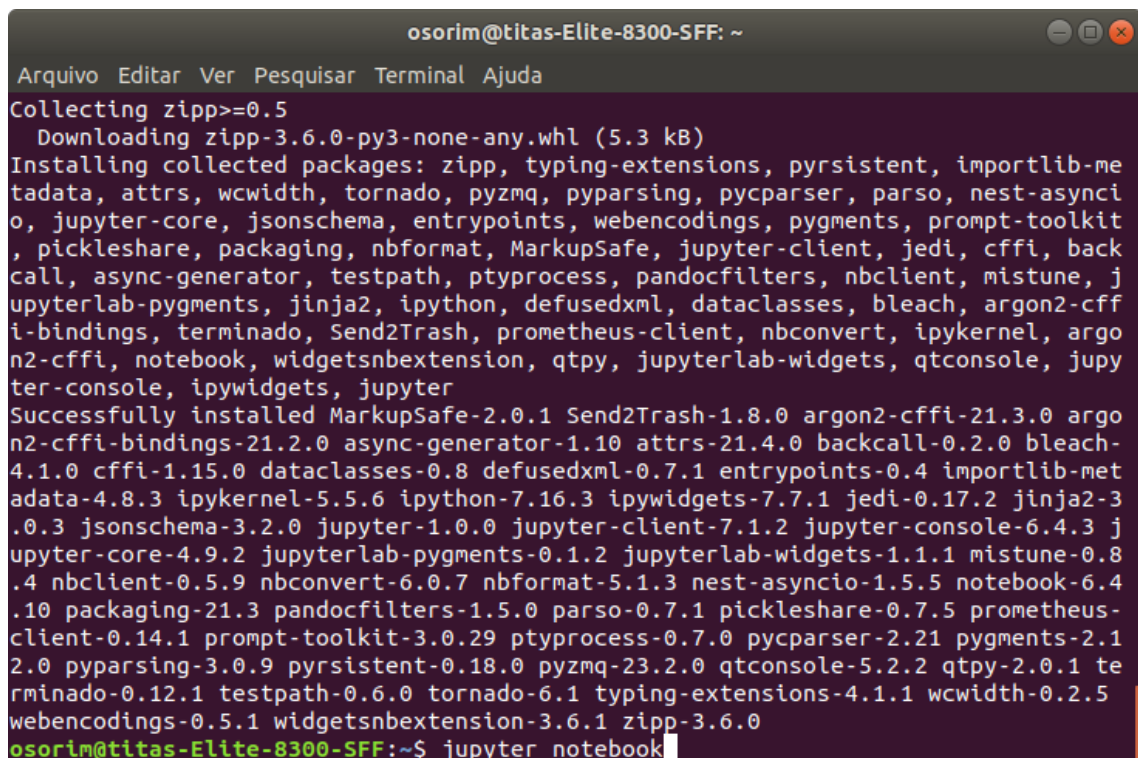
```
osorim@titas-Elite-8300-SFF: ~  
Arquivo Editar Ver Pesquisar Terminal Ajuda  
zope.interface      pkgs/main/linux-64::zope.interface-5.4.0-py39h7f8727e_0  
zstd                pkgs/main/linux-64::zstd-1.4.9-haebb681_0  
  
Preparing transaction: done  
Executing transaction: \  
  
    Installed package of scikit-learn can be accelerated using scikit-learn-inte  
lex.  
    More details are available here: https://intel.github.io/scikit-learn-intele  
x  
  
    For example:  
  
    $ conda install scikit-learn-intele  
    $ python -m sklearn my_application.py  
  
done  
installation finished.  
Do you wish the installer to initialize Anaconda3  
by running conda init? [yes|no]  
[no] >>> 
```

- 7- Após terminar o programa, executamos o código de instalação do jupyter, através do comando *conda install -c anaconda jupyter*.




```
osorim@titas-Elite-8300-SFF: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
osorim@titas-Elite-8300-SFF:~$ conda install -c anaconda jupyter
```

- 8- Após a execução da instalação, o notebook poderá ser aberto com o comando *jupyter notebook*.




```
osorim@titas-Elite-8300-SFF: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
Collecting zipp>=0.5
  Downloading zipp-3.6.0-py3-none-any.whl (5.3 kB)
Installing collected packages: zipp, typing-extensions, pyrsistent, importlib-me
tadata, attrs, wcwidth, tornado, pyzmq, pyparsing, pycparser, parso, nest-asynct
o, jupyter-core, jsonschema, entrypoints, webencodings, pygments, prompt-toolkit
, pickleshare, packaging, nbformat, MarkupSafe, jupyter-client, jedi, cffi, back
call, async-generator, testpath, ptyprocess, pandocfilters, nbclient, mistune, j
upyterlab-pygments, Jinja2, ipython, defusedxml, dataclasses, bleach, argon2-cff
i-bindings, terminado, Send2Trash, prometheus-client, nbconvert, ipykernel, argo
n2-cffi, notebook, widgetsnbextension, qtpy, jupyterlab-widgets, qtconsole, jupy
ter-console, ipywidgets, jupyter
Successfully installed MarkupSafe-2.0.1 Send2Trash-1.8.0 argon2-cffi-21.3.0 argo
n2-cffi-bindings-21.2.0 async-generator-1.10 attrs-21.4.0 backcall-0.2.0 bleach-
4.1.0 cffi-1.15.0 dataclasses-0.8 defusedxml-0.7.1 entrypoints-0.4 importlib-met
adata-4.8.3 ipykernel-5.5.6 ipython-7.16.3 ipywidgets-7.7.1 jedi-0.17.2 Jinja2-3
.0.3 jsonschema-3.2.0 jupyter-1.0.0 jupyter-client-7.1.2 jupyter-console-6.4.3 j
upyter-core-4.9.2 jupyterlab-pygments-0.12 jupyterlab-widgets-1.1.1 mistune-0.8
.4 nbclient-0.5.9 nbconvert-6.0.7 nbformat-5.1.3 nest-asyncio-1.5.5 notebook-6.4
.10 packaging-21.3 pandocfilters-1.5.0 parso-0.7.1 pickleshare-0.7.5 prometheus-
client-0.14.1 prompt-toolkit-3.0.29 ptyprocess-0.7.0 pycparser-2.21 pygments-2.1
2.0 pyparsing-3.0.9 pyrsistent-0.18.0 pyzmq-23.2.0 qtconsole-5.2.2 qtpy-2.0.1 te
rminado-0.12.1 testpath-0.6.0 tornado-6.1 typing-extensions-4.1.1 wcwidth-0.2.5
webencodings-0.5.1 widgetsnbextension-3.6.1 zipp-3.6.0
osorim@titas-Elite-8300-SFF:~$ jupyter notebook
```

- 9- Depois disso, irá abrir uma aba no seu navegador padrão com o sistema de arquivos do Jupyter Notebook, e pronto, está instalado :)

 Quit Logout

Files Running Clusters

Select items to perform actions on them. Upload New ▾ ↻

<input type="checkbox"/> 0 ▾	 /	Name ▾	Last Modified	File size
<input type="checkbox"/>	3D Objects		6 dias atrás	
<input type="checkbox"/>	Contacts		6 dias atrás	
<input type="checkbox"/>	Documents		6 dias atrás	
<input type="checkbox"/>	Downloads		uma hora atrás	
<input type="checkbox"/>	Favorites		6 dias atrás	
<input type="checkbox"/>	Links		6 dias atrás	
<input type="checkbox"/>	Music		6 dias atrás	
<input type="checkbox"/>	OneDrive		6 dias atrás	
<input type="checkbox"/>	Saved Games		6 dias atrás	
<input type="checkbox"/>	Searches		6 dias atrás	
<input type="checkbox"/>	Videos		4 dias atrás	

REFERÊNCIAS

1. ALMEIDA, João Paulo A. **The doce water quality ontology**. Disponível em: <https://nemo.inf.ufes.br/projetos/integradoce/> >
2. ALMEIDA, J. P. A. et al. **gUFO: A Lightweight Implementation of the Unified Foundational Ontology (UFO)**. 2019. Disponível em: <<https://nemo-ufes.github.io/gufo/>>.
3. BROETTO, Dhiego Santos. **Uma Apresentação Web para Apresentação de Dados de Qualidade de Água do Rio Doce**. Centro Tecnológico da Universidade Federal do Espírito Santo – UFES. Vitória, 2022.
4. FERREIRA, Thiago Castro. **Introdução a Ontologias e à Web Semântica (LIG948D)**. Faculdade de Letras - Universidade Federal de Minas Gerais (UFMG), 2020. Disponível em: <<https://www.youtube.com/watch?v=XLV416Gjl4g&list=PLlRlHSmC0Mw6lEj060psXrt3BSATBFVzV>>
5. ISOTANI, Seiji. BITTENCOURT, Ig Ibert. **Dados Abertos Conectados**. Novatec, Creative Commons. 2015. Disponível em: <<https://ceweb.br/livros/dados-abertos-conectados/>>
6. **Jupyter Project**. Jupyter Trademark, 2015. Disponível em: < <https://jupyter.org/> >
7. Miroir, Jean-Claude. (2018). Guia prático de construção de ontologias - Protégé v. 5.2. 10.13140/RG.2.2.31922.96969.
8. STANFORD. 2014. Protégé Ontology Editor. [Online] 2014. <http://protege.stanford.edu/>.
9. RABBI, Vinicius Teixeira. **Portal De Dados Sobre A Qualidade Da Água Do Rio Doce: Um Protótipo Funcional**. Curso de Pós-graduação lato sensu em Conectividade e Tecnologias da Informação. IFES – Campus Colatina, 2019
10. **rdflib 6.1.1 documentation**. RDFLib Team, 2021. Disponível em: < <https://rdflib.readthedocs.io/en/stable/> >