

PROGRAMAÇÃO WEB II

Curso Técnico Integrado em Informática

Lucas Sampaio Leite

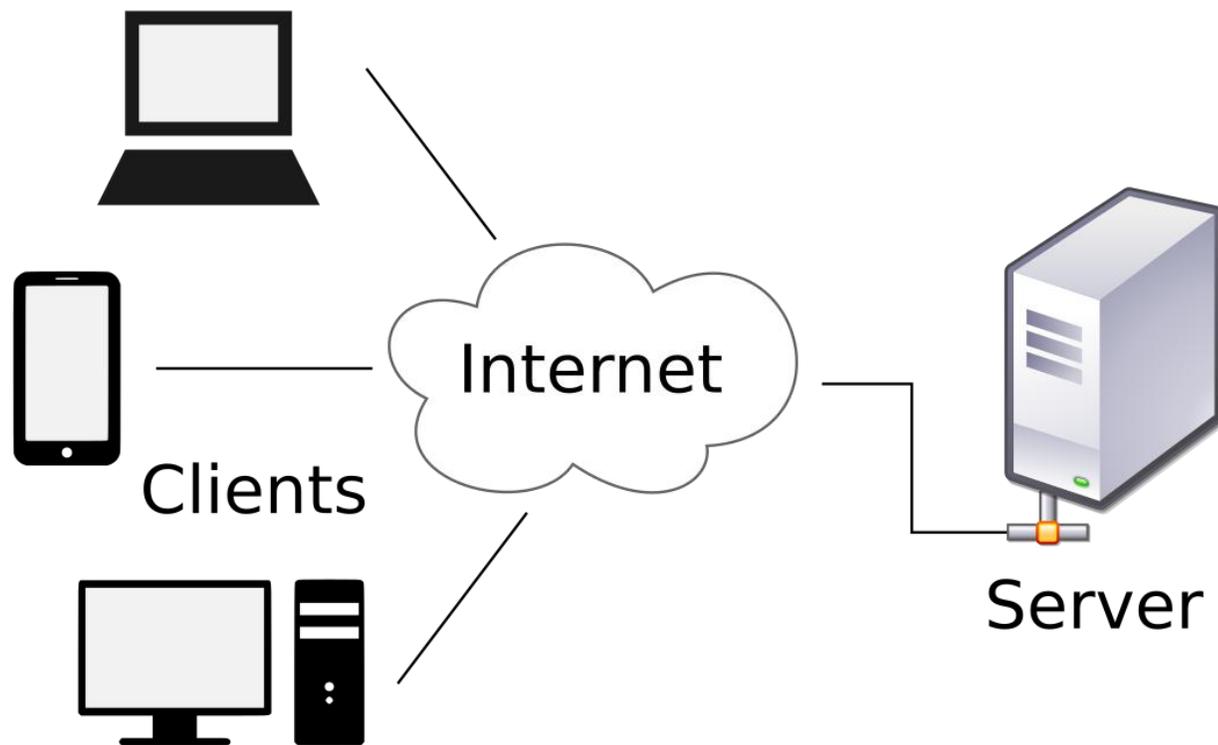


**INSTITUTO
FEDERAL**

Baiano

O Modelo Cliente-Servidor

- O modelo cliente-servidor é um dos pilares da arquitetura de redes e da computação distribuída. Ele descreve como dois sistemas (ou mais) se comunicam entre si, com funções bem definidas: um requisita e o outro responde.



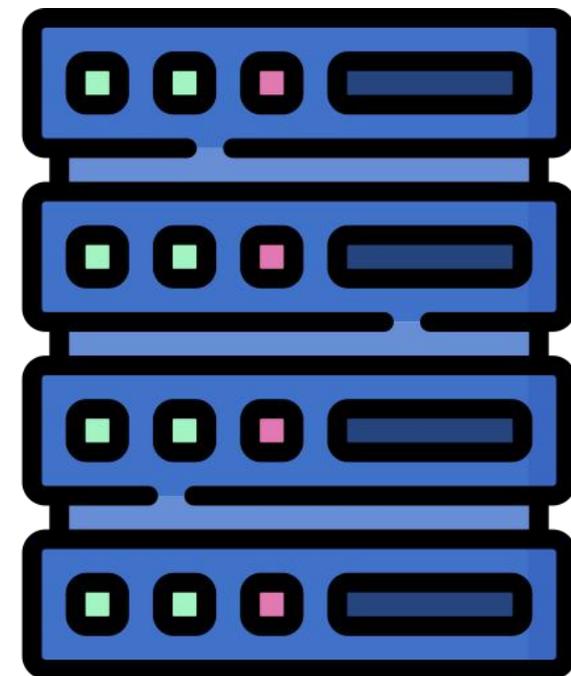
O Modelo Cliente-Servidor

- Cliente:
 - É o consumidor de serviços.
 - Envia requisições ao servidor pedindo algum recurso (ex: uma página web, um dado, o resultado de um cálculo).
 - Exemplos de clientes:
 - Navegadores (Chrome, Firefox)
 - Aplicativos móveis
 - Programas que usam uma API (como requests em Python)

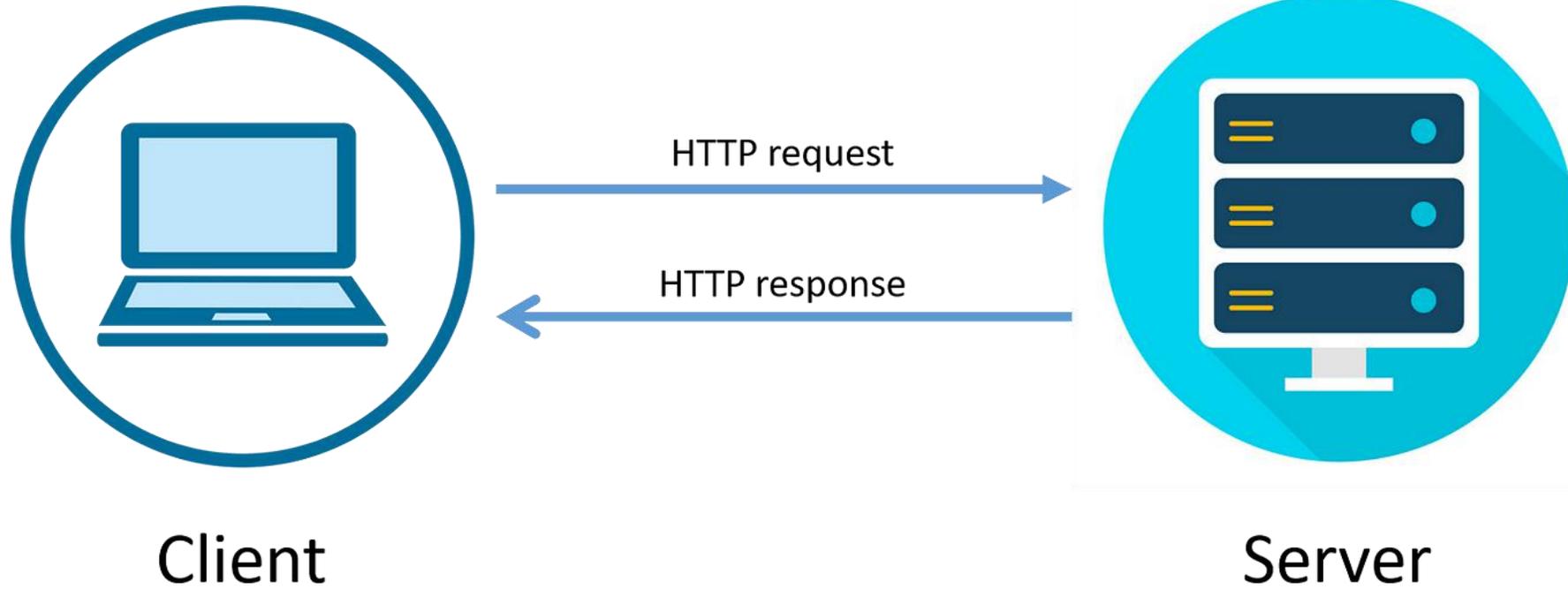


O Modelo Cliente-Servidor

- Servidor:
 - É o fornecedor de serviços.
 - Recebe as requisições dos clientes e responde com os dados/processamentos solicitados.
 - Pode atender múltiplos clientes ao mesmo tempo.
 - Exemplos de servidores:
 - Um servidor web (Apache, Nginx, Flask)
 - Um servidor de banco de dados (PostgreSQL, MySQL)



O Modelo Cliente-Servidor

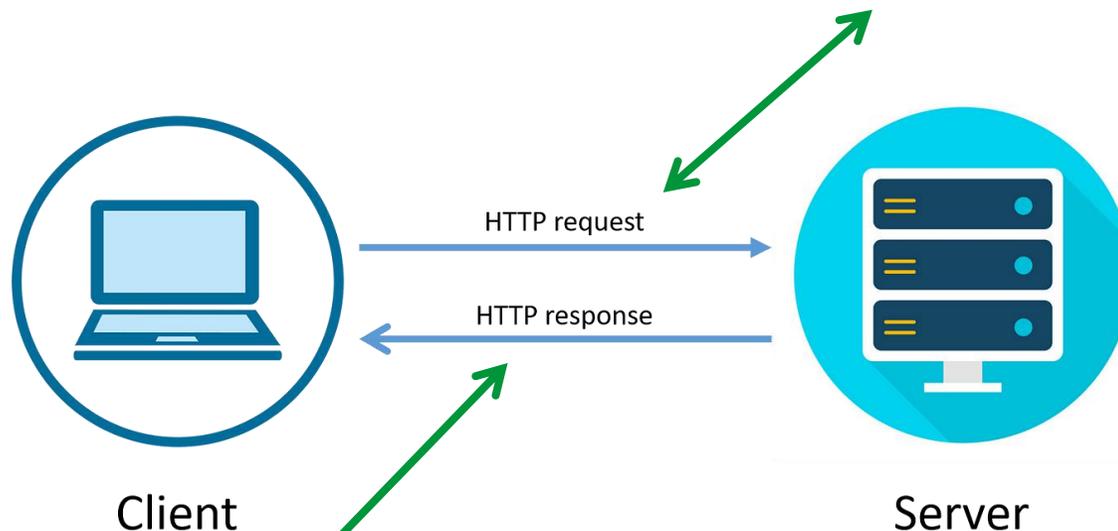


O Modelo Cliente-Servidor

1. O cliente envia uma requisição (ex: HTTP GET).
2. O servidor processa a requisição.
3. O servidor envia uma resposta com o resultado.
4. O cliente recebe e apresenta o resultado ao usuário.

O Modelo Cliente-Servidor

```
GET /usuarios/42 HTTP/1.1
Host: localhost:5000
User-Agent: Mozilla/5.0
Accept: text/html,application/json
Connection: keep-alive
```



```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "id": 42,
  "nome": "Maria",
  "email": "maria@example.com"
}
```

O Modelo Cliente-Servidor

1. O usuário interage com o frontend (interface gráfica):
 - Ex: clica num botão “Ver perfil”.

O Modelo Cliente-Servidor

1. O usuário interage com o frontend (interface gráfica):
 - Ex: clica num botão “Ver perfil”.
2. O frontend envia uma requisição para o backend:
 - Ex: GET /api/users/42
 - Essa requisição é feita usando JavaScript puro, formulários HTML, ou bibliotecas/frameworks do frontend como React, Vue ou Angular.

O Modelo Cliente-Servidor

1. O usuário interage com o frontend (interface gráfica):
 - Ex: clica num botão “Ver perfil”.
2. O frontend envia uma requisição para o backend:
 - Ex: GET /api/users/42
 - Essa requisição é feita usando JavaScript puro, formulários HTML, ou bibliotecas/frameworks do frontend como React, Vue ou Angular.
3. O backend processa a requisição:
 - Verifica permissões, busca dados no banco de dados, aplica regras de negócio, etc.

O Modelo Cliente-Servidor

1. O usuário interage com o frontend (interface gráfica):
 - Ex: clica num botão “Ver perfil”.
2. O frontend envia uma requisição para o backend:
 - Ex: GET /api/users/42
 - Essa requisição é feita usando JavaScript puro, formulários HTML, ou bibliotecas/frameworks do frontend como React, Vue ou Angular.
3. O backend processa a requisição:
 - Verifica permissões, busca dados no banco de dados, aplica regras de negócio, etc.
4. O backend envia uma resposta ao frontend:
 - Normalmente no formato JSON (ou HTML, XML, etc.)
 - Ex: `{"id": 42, "nome": "Maria", "idade": 25}`

O Modelo Cliente-Servidor

1. O usuário interage com o frontend (interface gráfica):
 - Ex: clica num botão “Ver perfil”.
2. O frontend envia uma requisição para o backend:
 - Ex: GET /api/users/42
 - Essa requisição é feita usando JavaScript puro, formulários HTML, ou bibliotecas/frameworks do frontend como React, Vue ou Angular.
3. O backend processa a requisição:
 - Verifica permissões, busca dados no banco de dados, aplica regras de negócio, etc.
4. O backend envia uma resposta ao frontend:
 - Normalmente no formato JSON (ou HTML, XML, etc.)
 - Ex: {"id": 42, "nome": "Maria", "idade": 25}
5. O frontend recebe os dados e atualiza a interface:
 - Ex: exhibe o nome e idade do usuário na tela.

Flask

- O Flask é um microframework web escrito em Python, usado para criar aplicações web, desde APIs simples até sites completos.
- Principais características:
 - Leve e minimalista: começa pequeno, sem exigir estruturas complexas.
 - Extensível: você adiciona apenas o que precisa (banco de dados, autenticação, etc.).
 - Rápido para aprender e usar: ótimo para iniciantes e projetos pequenos/médios.
 - Baseado em WSGI (o padrão de comunicação entre servidores e aplicações web em Python).

Flask

- Documentação: <https://flask.palletsprojects.com/en/stable/>

Project Links

[Donate](#)
[PyPI Releases](#)
[Source Code](#)
[Issue Tracker](#)
[Chat](#)

Contents

[Welcome to Flask](#)
[User's Guide](#)
[API Reference](#)
[Additional Notes](#)

Quick search



Flask

Welcome to Flask's documentation. Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications.

Get started with [Installation](#) and then get an overview with the [Quickstart](#). There is also a more detailed [Tutorial](#) that shows how to create a small but complete application with Flask. Common patterns are described in the [Patterns for Flask](#) section. The rest of the docs describe each component of Flask in detail, with a full reference in the [API](#) section.

Flask depends on the [Werkzeug](#) WSGI toolkit, the [Jinja](#) template engine, and the [Click](#) CLI toolkit. Be sure to check their documentation as well as Flask's when looking for information.

User's Guide

Flask provides configuration and conventions, with sensible defaults, to get started. This section of the documentation explains the different parts of the Flask framework and how they can be used, customized, and extended. Beyond Flask itself, look for community-maintained extensions to add even more functionality.

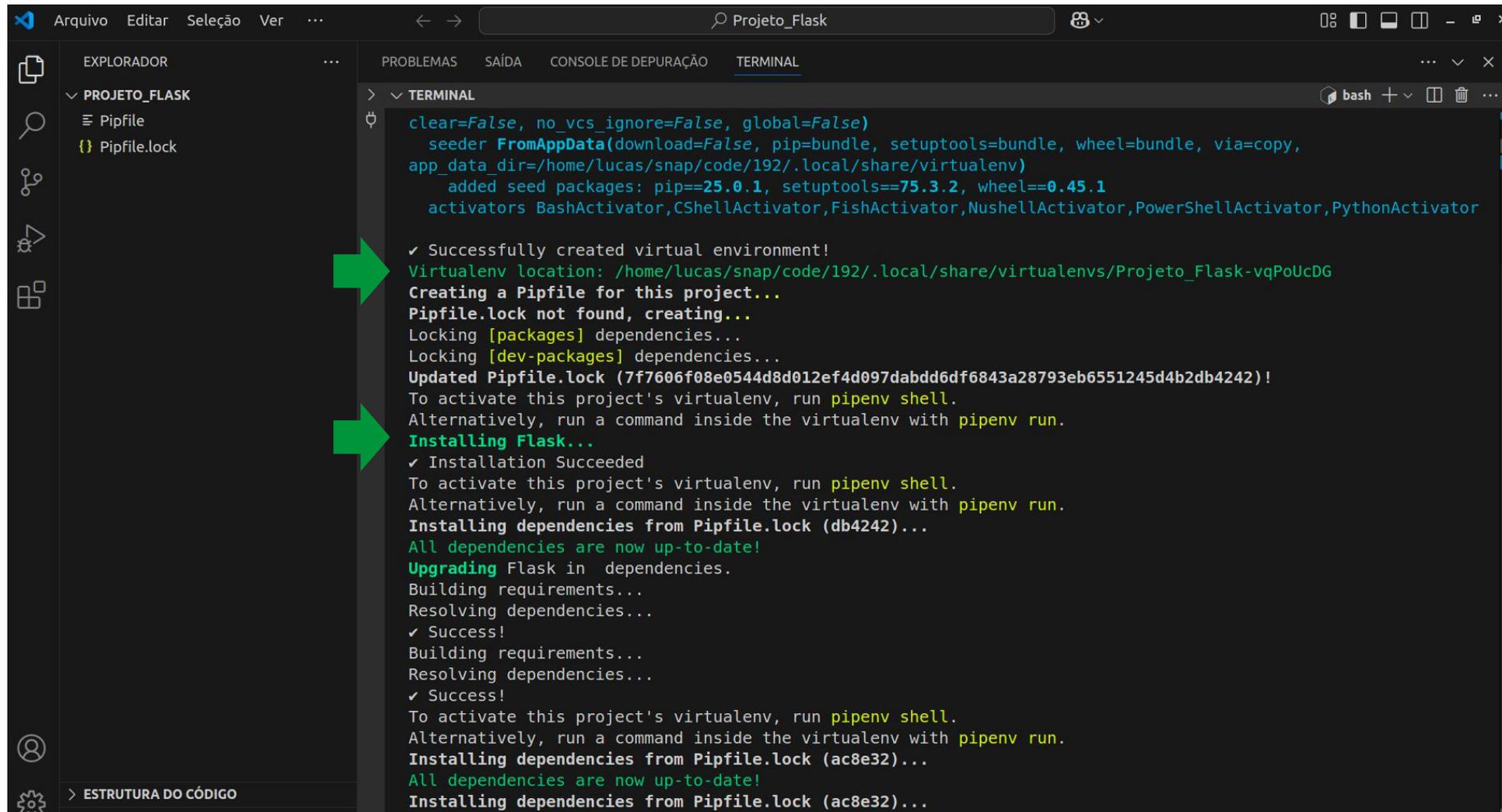
- [Installation](#)
 - [Python Version](#)
 - [Dependencies](#)
 - [Virtual environments](#)

Criando um ambiente virtual e instalando o Flask

- Verificar a versão do python: Python: ≥ 3.8
 - `python --version`
 - Exemplo de retorno: Python 3.8.10
- Verifique se o pipenv está instalado:
 - `pipenv --version`
 - Exemplo de retorno: pipenv, version 2024.4.1
- Se o pipenv não estiver instalado:
 - `pip install pipenv`

Criando um ambiente virtual e instalando o Flask

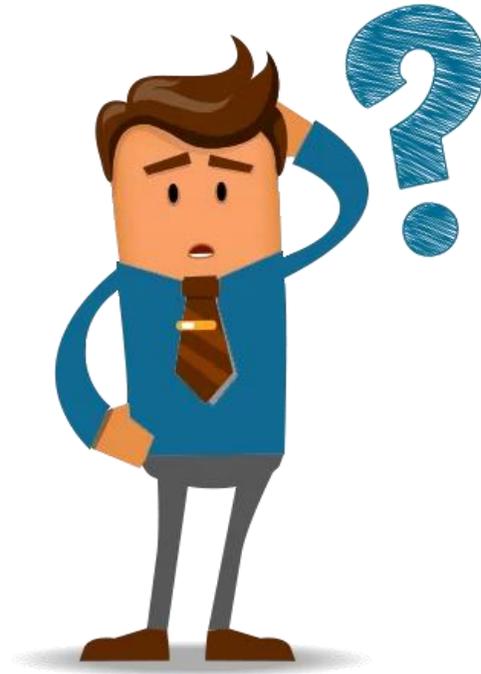
- Criar o ambiente virtual e instalar o Flask: **pipenv install Flask**



```
clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy,
app_data_dir=/home/lucas/snap/code/192/.local/share/virtualenv)
  added seed packages: pip==25.0.1, setuptools==75.3.2, wheel==0.45.1
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

✓ Successfully created virtual environment!
Virtualenv location: /home/lucas/snap/code/192/.local/share/virtualenvs/Projeto_Flask-vqPoUcDG
Creating a Pipfile for this project...
Pipfile.lock not found, creating...
Locking [packages] dependencies...
Locking [dev-packages] dependencies...
Updated Pipfile.lock (7f7606f08e0544d8d012ef4d097dabdd6df6843a28793eb6551245d4b2db4242)!
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing Flask...
✓ Installation Succeeded
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing dependencies from Pipfile.lock (db4242)...
All dependencies are now up-to-date!
Upgrading Flask in dependencies.
Building requirements...
Resolving dependencies...
✓ Success!
Building requirements...
Resolving dependencies...
✓ Success!
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing dependencies from Pipfile.lock (ac8e32)...
All dependencies are now up-to-date!
Installing dependencies from Pipfile.lock (ac8e32)...
```

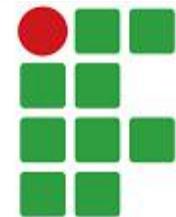
Dúvidas



PROGRAMAÇÃO WEB II

Curso Técnico Integrado em Informática

Lucas Sampaio Leite



**INSTITUTO
FEDERAL**

Baiano