

# FRANCISCO JOSÉ DE CALDAS DISTRICT UNIVERSITY



*Bogotá C.D.*

*2024*



## I. OBJECTIVE

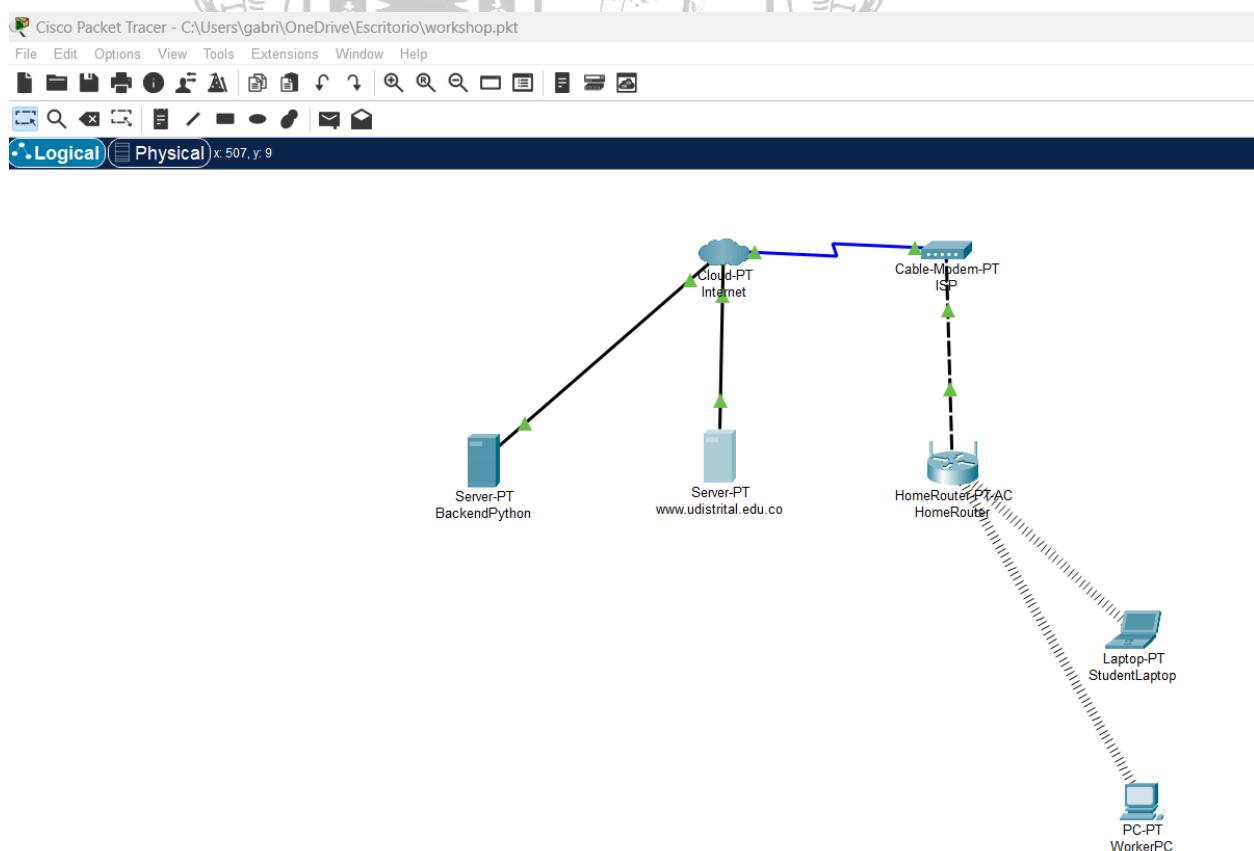
The main objective of this workshop is to understand and apply the use of sockets in network communication. Through practical exercises, you will explore how sockets function within the OSI model, gaining insights into the interaction between different network layers.

## II. WORKSHOP

The steps for carrying out the workshop will be described below:

### 1. OPEN *WORKSHOP 1* FILE AND ADD THE NEW SERVER

Let's open the file from the previous workshop and add the new 'BackendPython' server from end devices.





## 2. SETTING UP NEW SERVER “BackendPython” CONFIGURATION

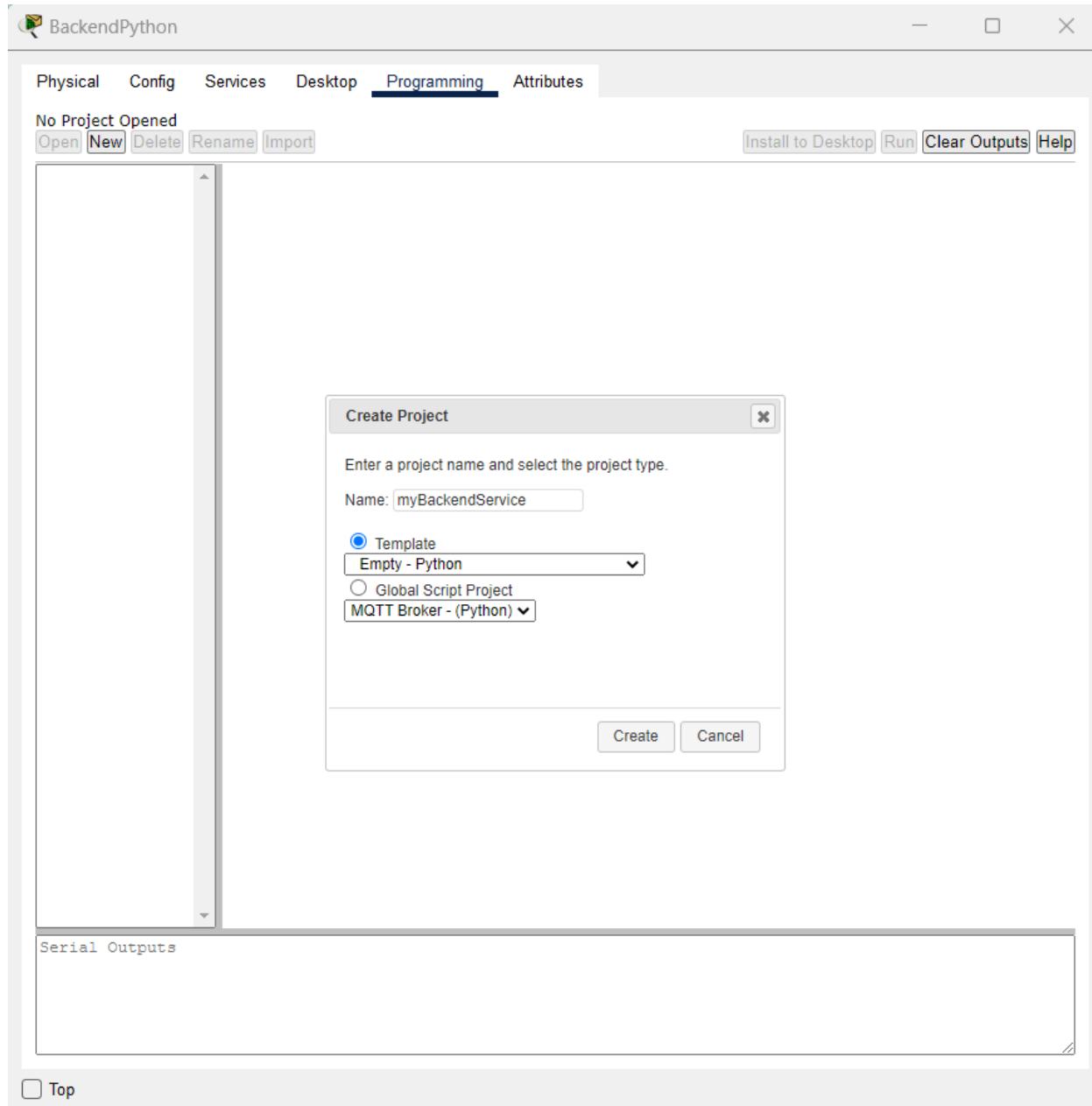
It is necessary setting up the data of the server: *IPv4*, *Subnet Mask*, *Default Gateway*.

The screenshot shows the BackendPython configuration interface with the "Desktop" tab selected. The "IP Configuration" section is open, displaying the following settings:

Setting	Value
DHCP	<input type="radio"/>
Static	<input checked="" type="radio"/>
IPv4 Address	193.168.100.201
Subnet Mask	255.255.255.0
Default Gateway	193.168.100.1
DNS Server	0.0.0.0
IPv6 Configuration	
Automatic	<input type="radio"/>
Static	<input checked="" type="radio"/>
IPv6 Address	/
Link Local Address	FE80::201:C9FF:FE02:1B9E
Default Gateway	
DNS Server	
802.1X	
Use 802.1X Security	<input type="checkbox"/>
Authentication	MD5
Username	
Password	



Subsequently, in the *Programming* services section, we create a new project using **Template:** empty -- Python.





Once created our project, let's add the corresponding code at *main.py* file

The screenshot shows the BackendPython application window. The title bar says "BackendPython". The menu bar includes "Physical", "Config", "Services", "Desktop", "Programming", and "Attributes", with "Programming" being the active tab. Below the menu is a toolbar with buttons for "Open", "New", "Delete", "Rename", and "Import". To the right of the toolbar are buttons for "Install to Desktop", "Run", "Clear Outputs", and "Help". The main area is a code editor titled "myBackendService (Python) - main.py". The code is as follows:

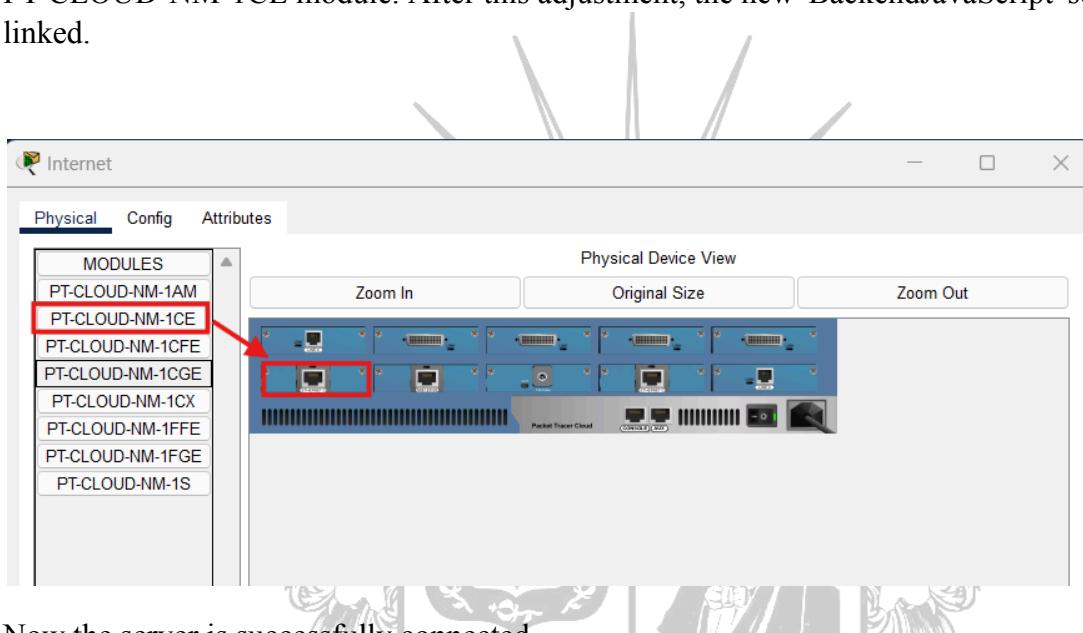
```
.. main.py
1 """
2 This is a simple example of a web service for Python into PacketTracer.
3
4 Author: Gabriela Martinez Eslava <gmartinez@udistrital.edu.co>
5
6 """
7
8 from http import *
9 from time import *
10
11 def on_route_networks(url: str, response):
12
13     """
14         This function is called when the URL is /healthcheck.
15
16     Args:
17
18         url (str): The URL of the request
19         response (HTTPResponse): The response object to send data to the client.
20
21     """
22     print("Test Services")
23     response.send("This is a verification about Python services.")
24
25 def main():
26
27     """This is the main function of the program."""
28     HTTPServer.route("/healthcheck", on_route_networks)
29
30     #start server on port 80
31     print(HTTPServer.start(80))
32
33     #don't let it finish
34     while True:
35         sleep(3600)
36
37 if __name__ == "__main__":
38     main()
```

Below the code editor is a "Serial Outputs" panel which is currently empty. At the bottom left is a "Top" button.

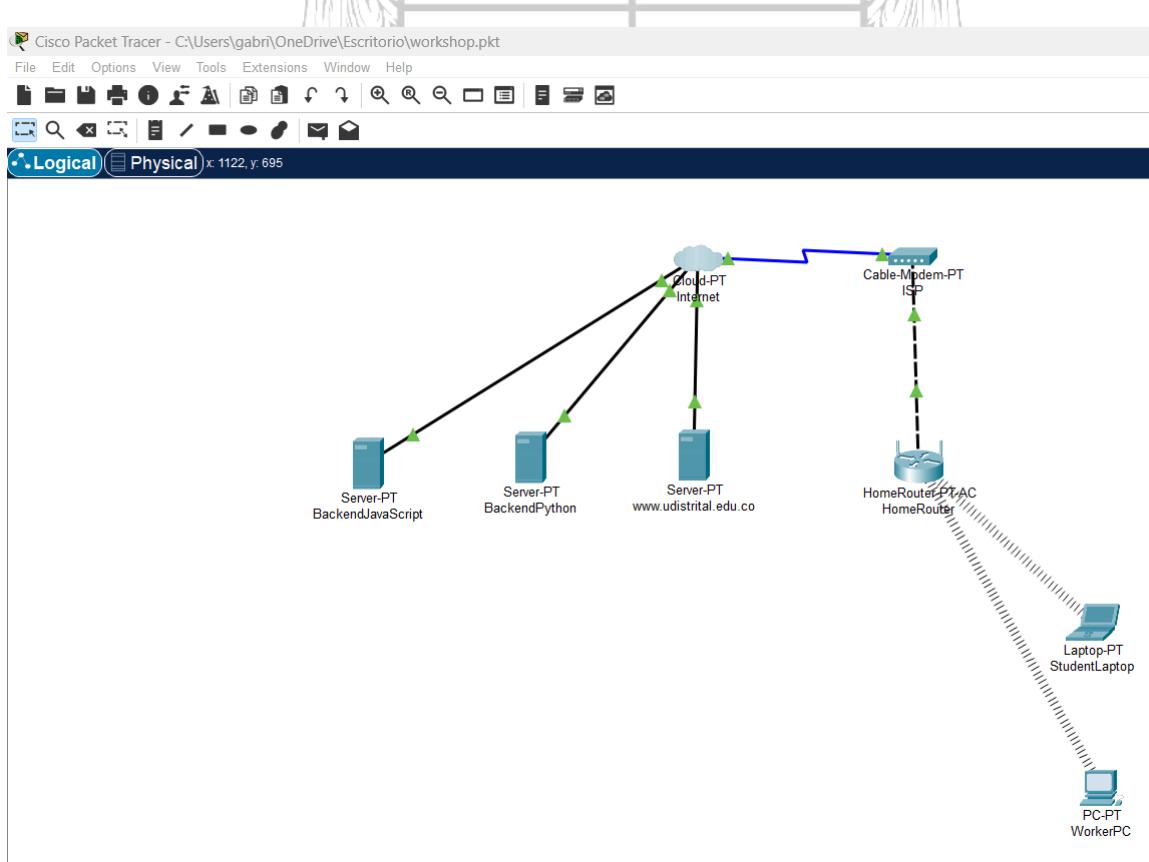


### 3. ADD ANOTHER NEW SERVER

add the new 'BackendJavaScript' server from end devices. In this particular case, the Cloud-PT Internet did not have enough ports to connect this third server, so it was necessary to add a PT-CLOUD-NM-1CE module. After this adjustment, the new 'BackendJavaScript' server was linked.

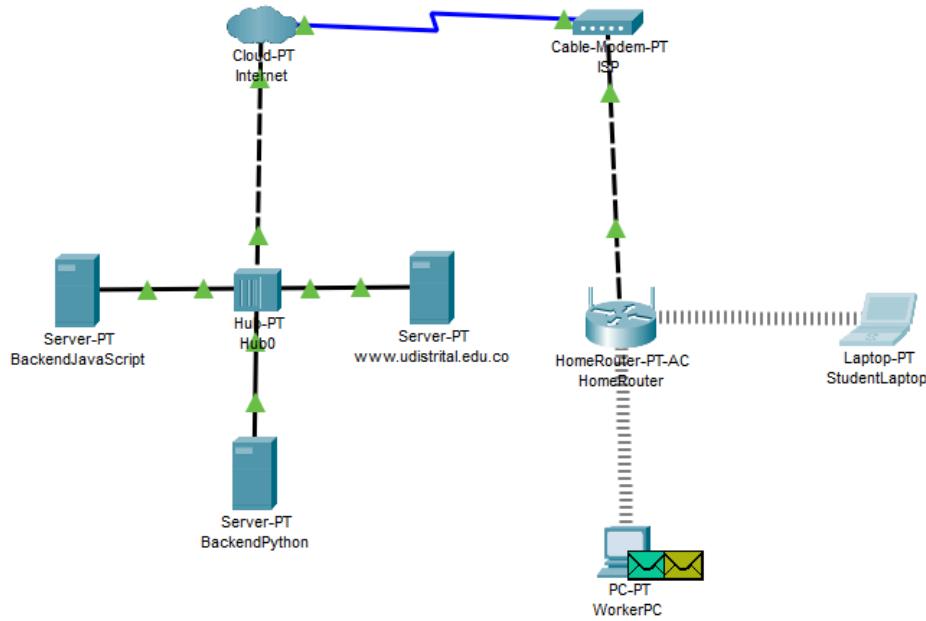


Now the server is successfully connected.





In the process of connecting the server I noticed that it will be better using a hub between servers to make the connection.



#### 4. SETTING UP NEW SERVER “BackendJavaScript” CONFIGURATION

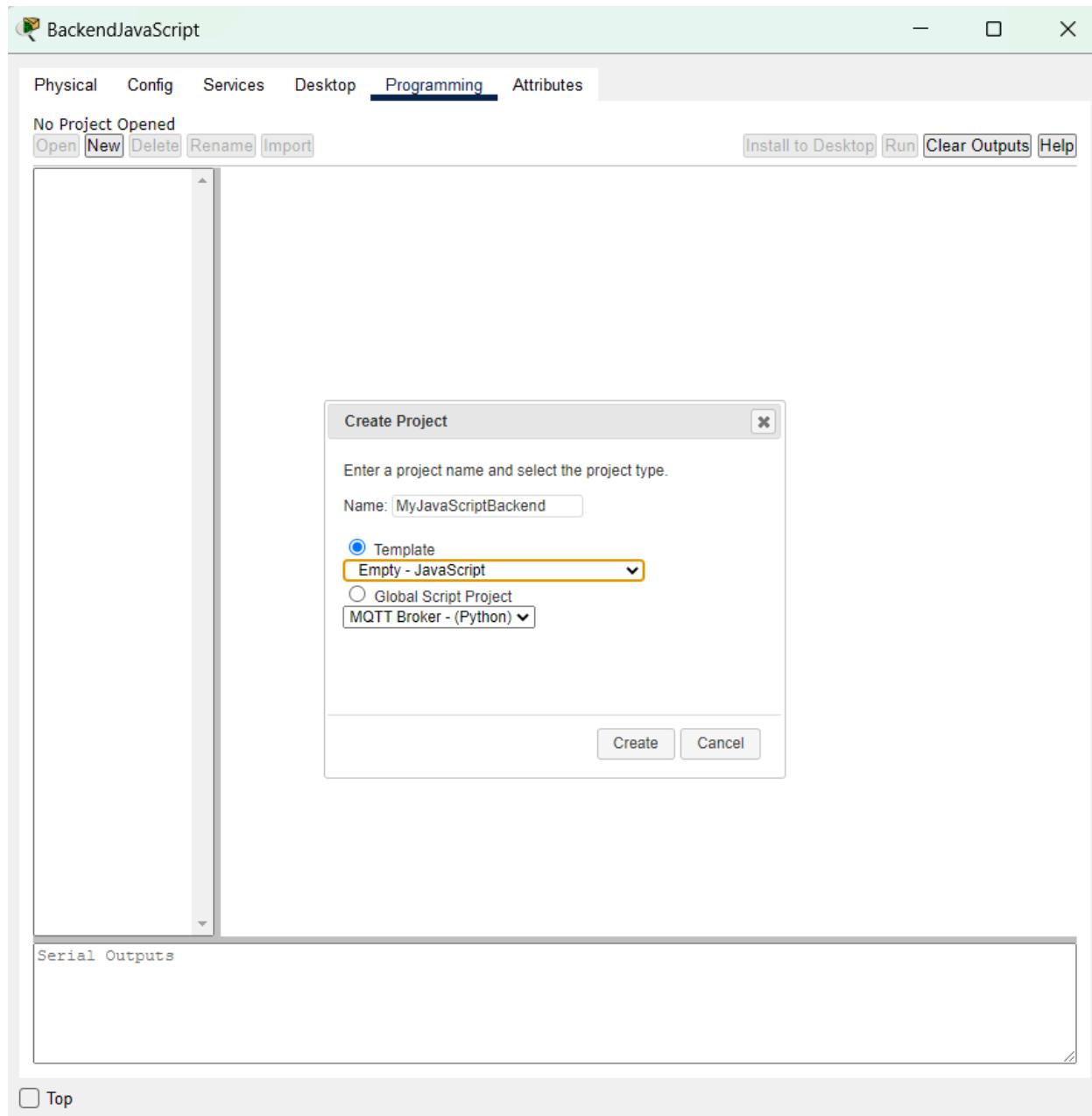
set up the data of the server: *IPv4, Subnet Mask, Default Gateway*.

A screenshot of a server configuration interface titled "BackendJavaScript". The interface has a tab bar with "Physical", "Config", "Services", "Desktop" (which is selected), "Programming", and "Attributes". A red box highlights the "IP Configuration" section under the "Desktop" tab.

IP Configuration	
IP Configuration	<input type="radio"/> DHCP <input checked="" type="radio"/> Static
IPv4 Address	193.168.100.202
Subnet Mask	255.255.255.0
Default Gateway	193.168.100.1
DNS Server	0.0.0.0
IPv6 Configuration	
IPv6 Configuration	<input type="radio"/> Automatic <input checked="" type="radio"/> Static
IPv6 Address	[Empty Field] / [Empty Field]
Link Local Address	FE80::206:2AFF:FE90:AA50
Default Gateway	[Empty Field]



Then, in the *Programming* services section, we create a new project using **Template:** empty -- JavaScript.





Once created our project, let's add the corresponding code at *main.js* file

The screenshot shows the 'BackendJavaScript' application window. The title bar reads 'BackendJavaScript'. The menu bar includes 'Physical', 'Config', 'Services', 'Desktop', 'Programming' (which is selected), and 'Attributes'. Below the menu is a toolbar with buttons for 'Open', 'New', 'Delete', 'Rename', 'Import', 'Install to Desktop', 'Run', 'Clear Outputs', and 'Help'. The main area is a code editor with the file 'main.js' open. The code is as follows:

```
1 /*  
2 This is an example of a web service for Python into PacketTracer.  
3  
4 Author: Gabriela Martinez Eslava <gmartinez@udistrital.edu.co>  
5 */  
6  
7 function setup(){  
8     HTTPServer.route("/healthcheck", function(url, res){  
9         Serial.println("Test services");  
10        res.setContentType("text/plain");  
11        res.send("This is a verification about javascript services");  
12    });  
13  
14    //Start server on port 80  
15    HTTPServer.start(80);  
16  
17 }|
```

At the bottom left, there is a 'Serial Outputs' panel which is currently empty. At the bottom right, there is a 'Top' button.



Then, it is necessary to build the frontend service using HTML, CSS and JavaScript.

### HTML file:



```
<html>
  <head>
    <title>Workshop 4 - Networks</title>
    <meta charset="UTF-8">
    <meta name="description" content="This is a simple example to explore OSI">
    <link rel="stylesheet" type="text/css" href="styles.css">
    <script src="functions.js"></script>
  </head>
  <body>
    <h1>Workshop 4 - Networks</h1>
    <h2>OSI Layers</h2>
    <p>Click on the buttons to explore services on the network</p>
    <div>
      <button onclick="callPython()">Python Message</button>
      <button onclick="callJavaScript()">JavaScript Message</button>
    </div>
    <div id="result"></div>
  </body>
</html>
```

### CSS file:

I changed it to make it prettier, nothing personal :P



```
# styles.CSS > body
body {
  font-family: "Arial", sans-serif;
  font-size: 16px;
  background-color: #f8f4f9;
  color: #5a5a5a;
  text-align: center;
  padding: 20px;
}

h1 {
  color: #ff6b6b;
  font-size: 28px;
  margin-bottom: 20px;
}

h2 {
  color: #6b7aff;
  font-size: 22px;
  margin-bottom: 15px;
}

button {
  background-color: #ffcb77;
  color: #5a5a5a;
  border: none;
  padding: 12px 24px;
  font-size: 16px;
  font-weight: bold;
  cursor: pointer;
  border-radius: 12px;
  transition: background 0.3s, transform 0.2s;
  box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.1);
}

button:hover {
  background-color: #ffd97d;
  transform: scale(1.05);
}
```

```
# styles.CSS > body
button {
  border: none;
  padding: 12px 24px;
  font-size: 16px;
  font-weight: bold;
  cursor: pointer;
  border-radius: 12px;
  transition: background 0.3s, transform 0.2s;
  box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.1);
}

button:hover {
  background-color: #ffd97d;
  transform: scale(1.05);
}

.result {
  margin-top: 20px;
  padding: 15px;
  border: 2px solid #b8c0ff;
  background-color: #e3e8ff;
  border-radius: 12px;
  box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.1);
  display: inline-block;
  min-width: 200px;
}
```

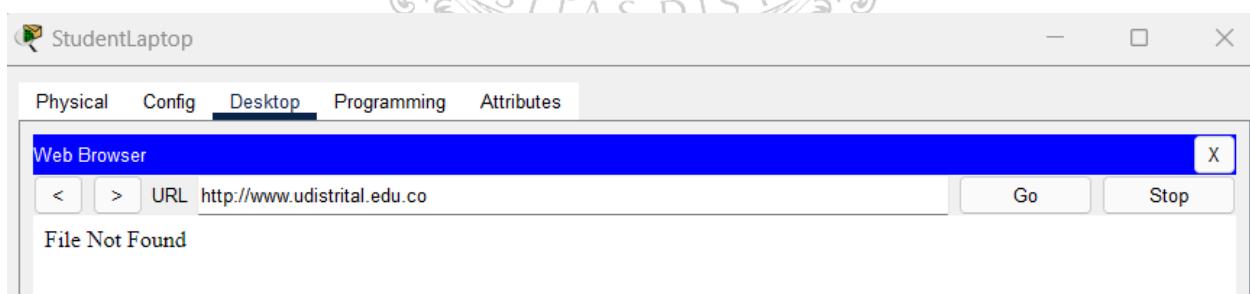


### JavaScript file:

```
index.HTML X # styles.CSS JS functions.js X
JS functions.js > callJavaScript
1 function callPython(){
2     fetch('http://193.168.100.201/healthcheck')
3         .then(Response => Response.text())
4         .then(data => {
5             const resultDiv = document.getElementById('result');
6             resultDiv.innerText= data;
7         })
8         .catch ( error => {
9             console.error('Error:', error);
10        });
11    }
12
13 }
14
15 function callJavaScript() {
16     fetch('http://193.168.100.202/healthcheck')
17         .then(response => response.text())
18         .then(data => {
19             const resultDiv = document.getElementById('result');
20             resultDiv.innerText = data;
21         })
22         .catch(error =>{
23             console.error('Error:', error);
24         });
25 }
```

### 5. TEST ACCESS IN THE DEVICES

First, when I try to access [www.udistrital.edu.co](http://www.udistrital.edu.co) from *StudentLaptop* or *WorkerPC* file was not found.



So I checked the .HTML, .CSS, and .js files, and the issue was the extension, which must be in lowercase. The problem was fixed by renaming the file from index.HTML to index.html, and the same for style.CSS



The screenshot shows a network management interface with a sidebar containing services like HTTP, DHCP, DNS, and IoT. The main area has tabs for Physical, Config, Services (selected), Desktop, Programming, and Attributes. A file named "styles.CSS" is being uploaded, displaying its contents:

```
body {  
    font-family: "Arial", sans-serif;  
    font-size: 16px;  
    background-color: #f8f4f9;  
    color: #5a5a5a;  
    text-align: center;  
    padding: 20px;  
}  
  
h1 {  
    color: #ff6b6b;  
    font-size: 28px;  
    margin-bottom: 20px;  
}  
  
h2 {  
    color: #ff6b6b;  
    font-size: 24px;  
    margin-bottom: 10px;  
}  
  
button {  
    background-color: #5a5a5a;  
    color: #fff;  
    border: none;  
    padding: 12px 24px;  
    font-size: 16px;  
    font-weight: bold;  
}
```

A modal dialog titled "File Edit Warning" appears, stating "File extension is not supported. Supported file extensions: .html .htm .php .css .js".

After this, it finally worked 😊

The screenshot shows a web browser window titled "StudentLaptop". The tabs are Physical, Config, Desktop (selected), Programming, and Attributes. The URL bar shows "http://www.udistrital.edu.co". The page content includes:

## Workshop 4 - Networks

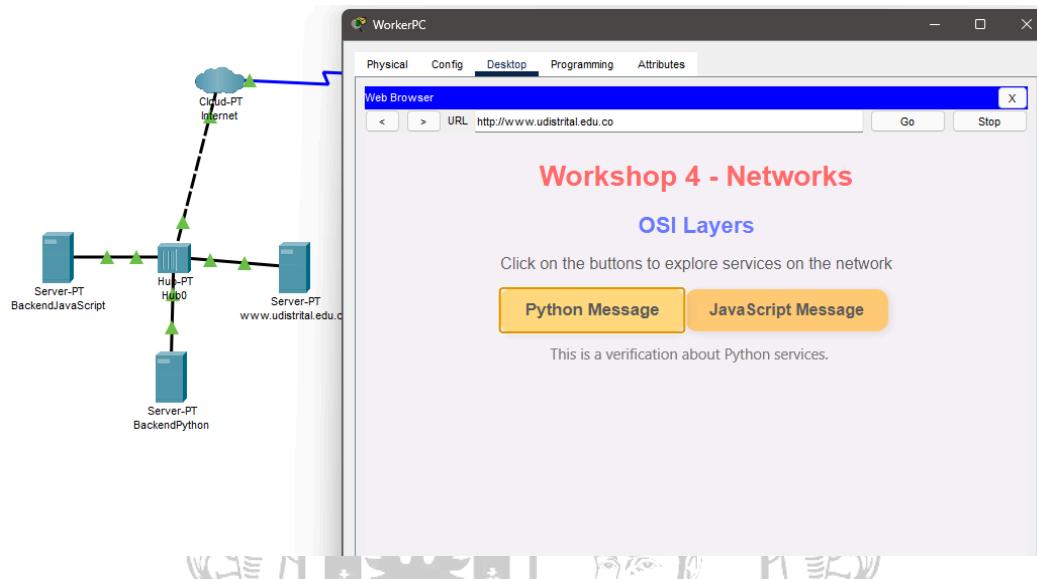
### OSI Layers

Click on the buttons to explore services on the network

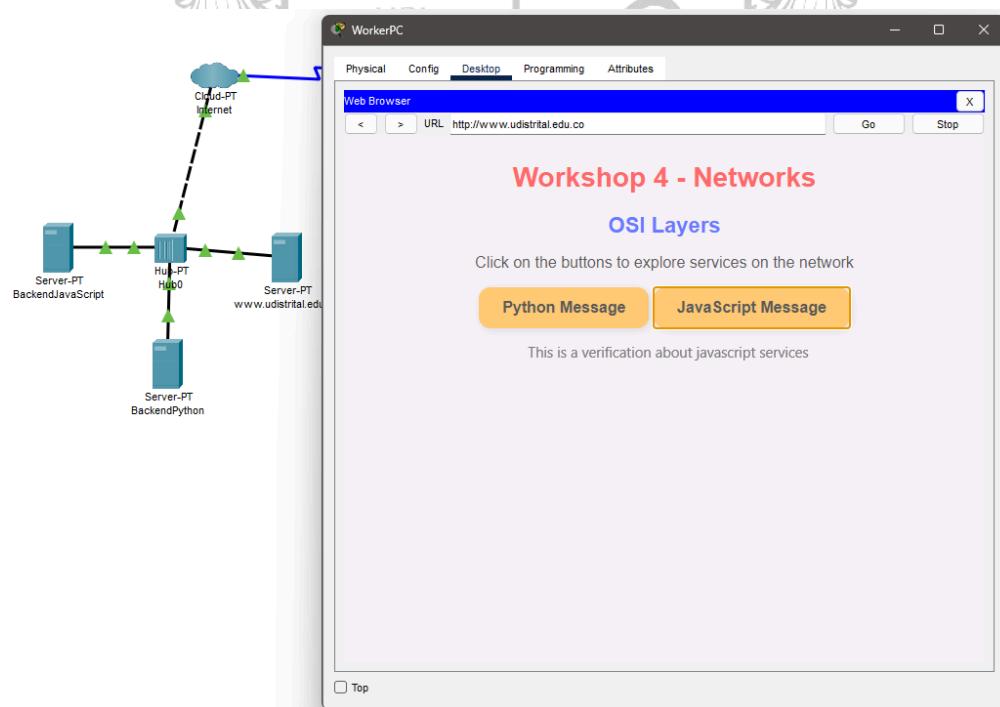
[Python Message](#)   [JavaScript Message](#)



First I tried calling the python backend through its button, and the backend answered by sending the corresponding message.

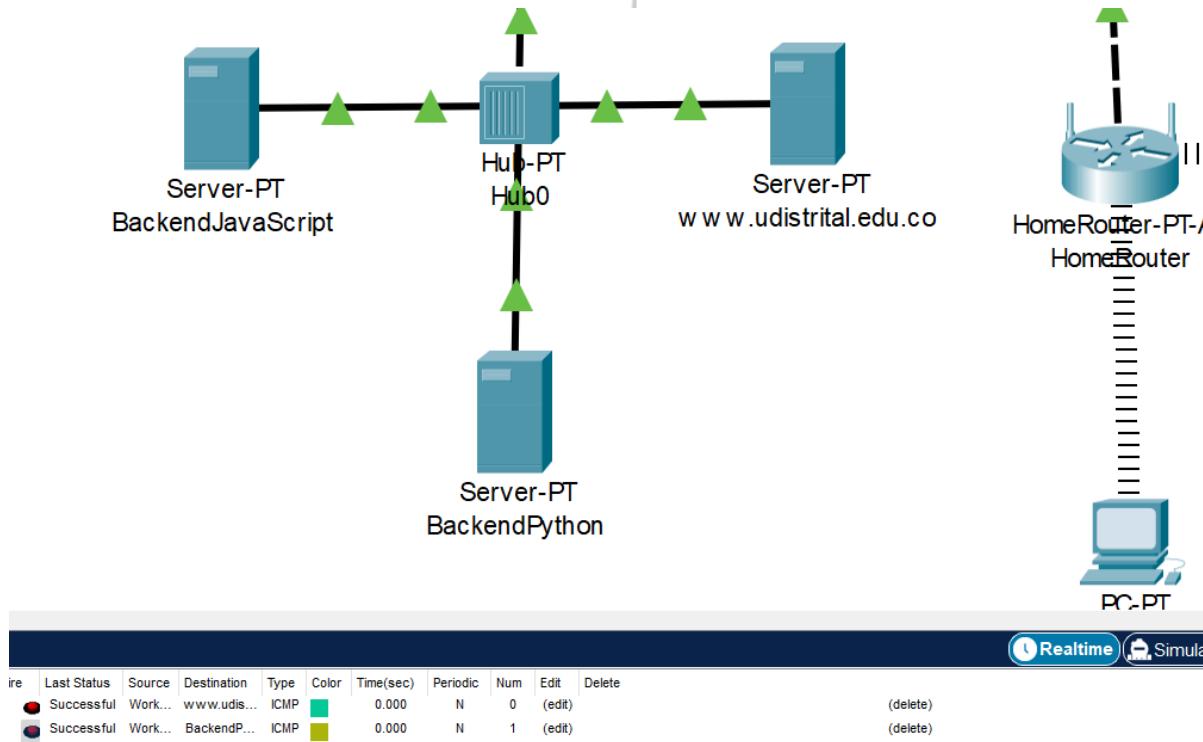


Then i tried calling Java Script backend and it worked as well 😊

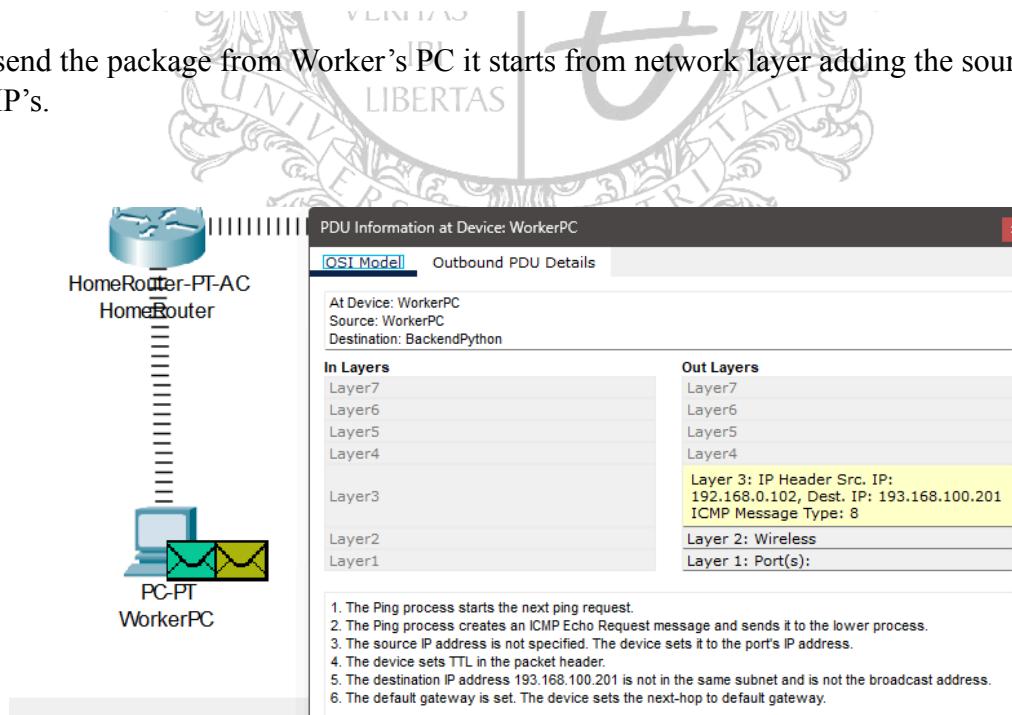




I made the simulations trying to call the frontend from Worker's PC and both of them worked.

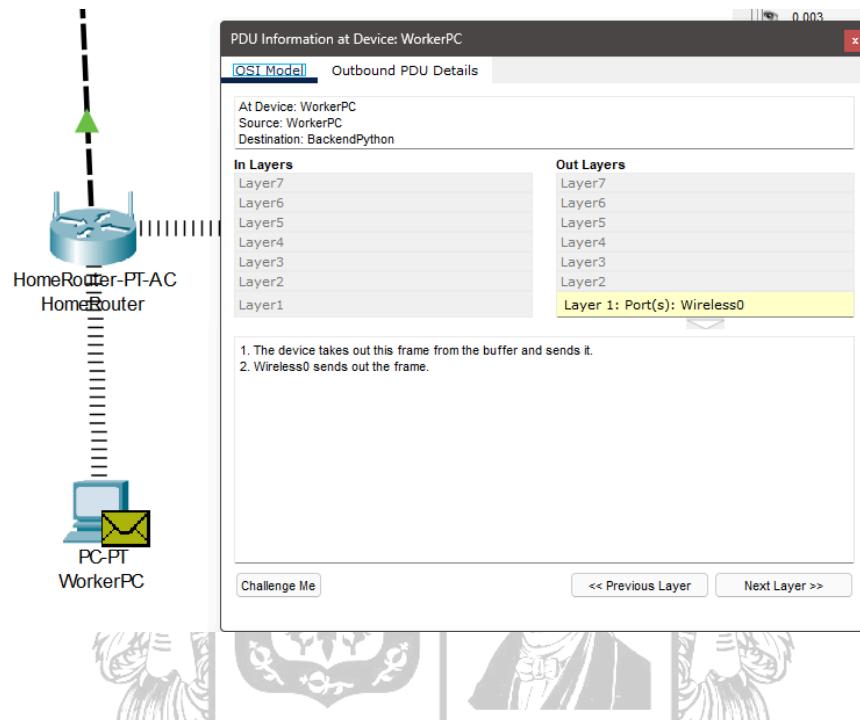


When i send the package from Worker's PC it starts from network layer adding the source and destiny IP's.

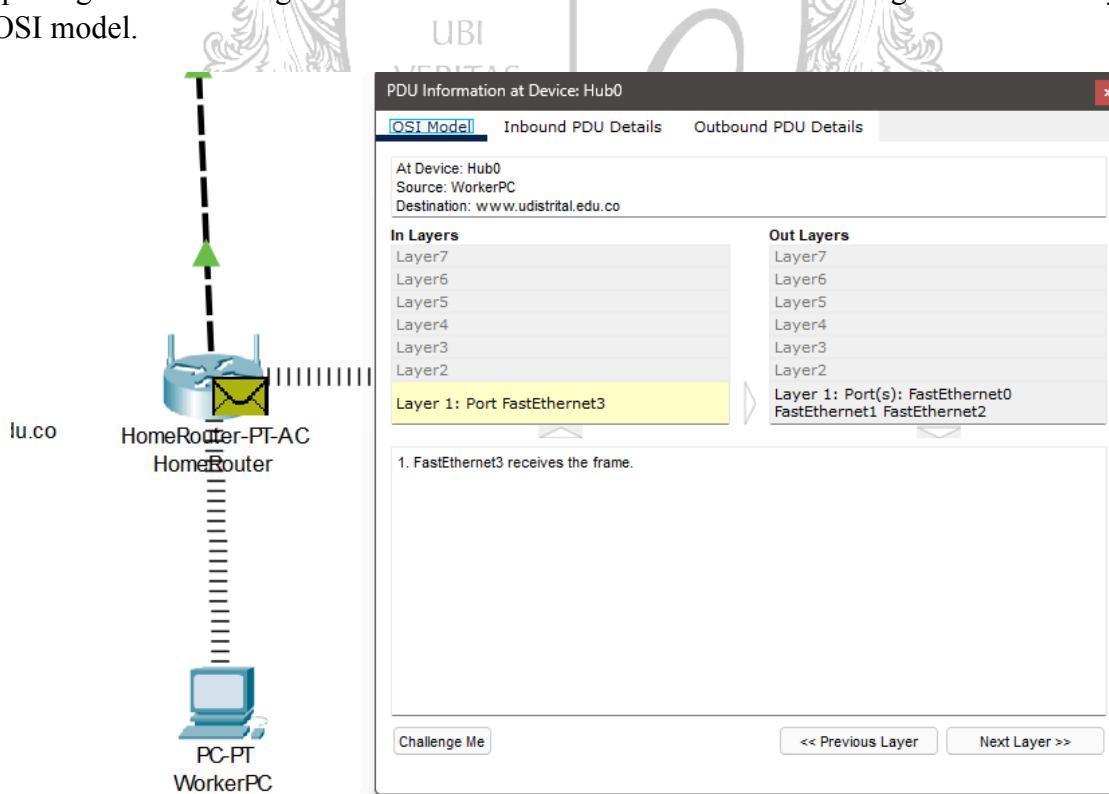




Then it goes through data link layer adding the MAC and goes through the wires in the physical layer.

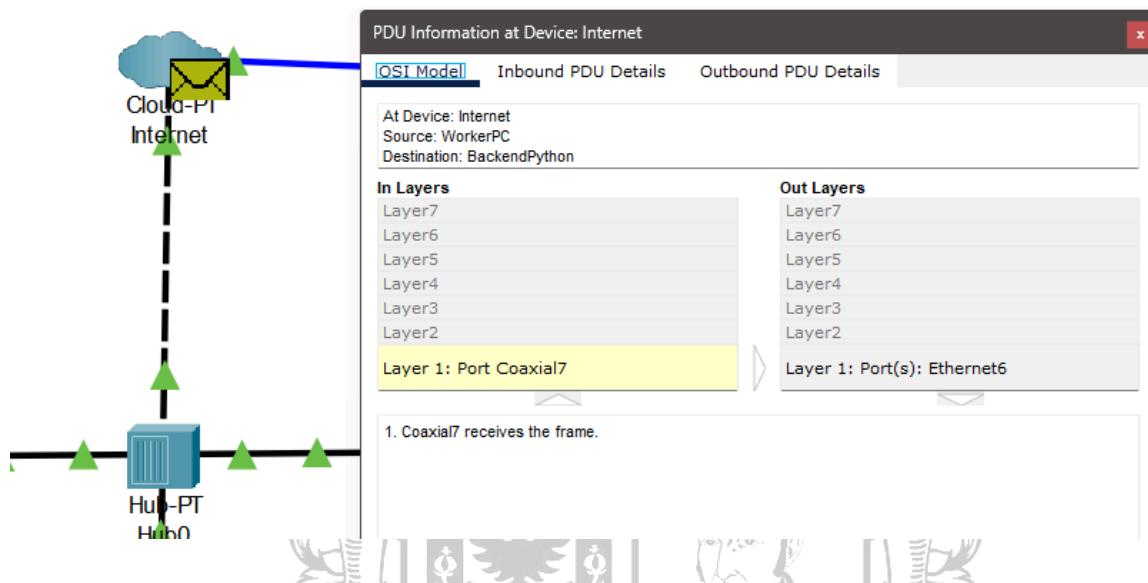


The package travels through wireless connection to the Home Router using the first “in” layer of the OSI model.

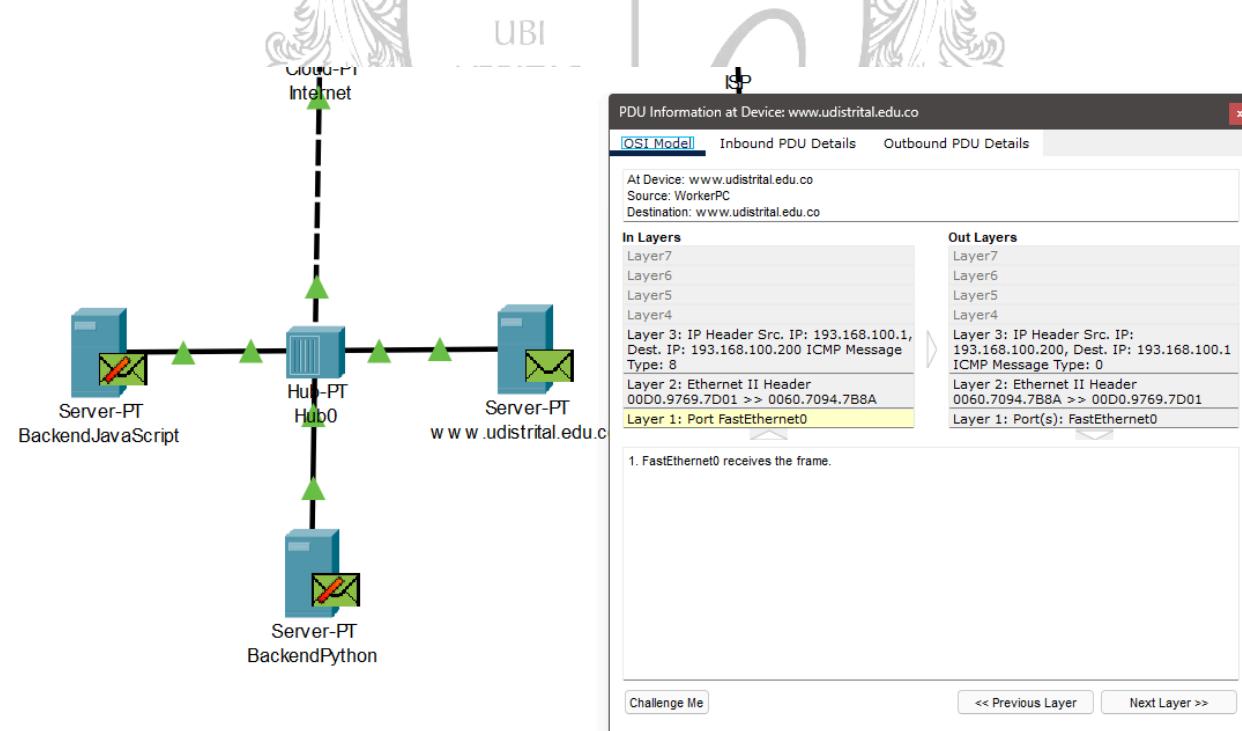




It continues travelling through the physical layer, searching for the destination device

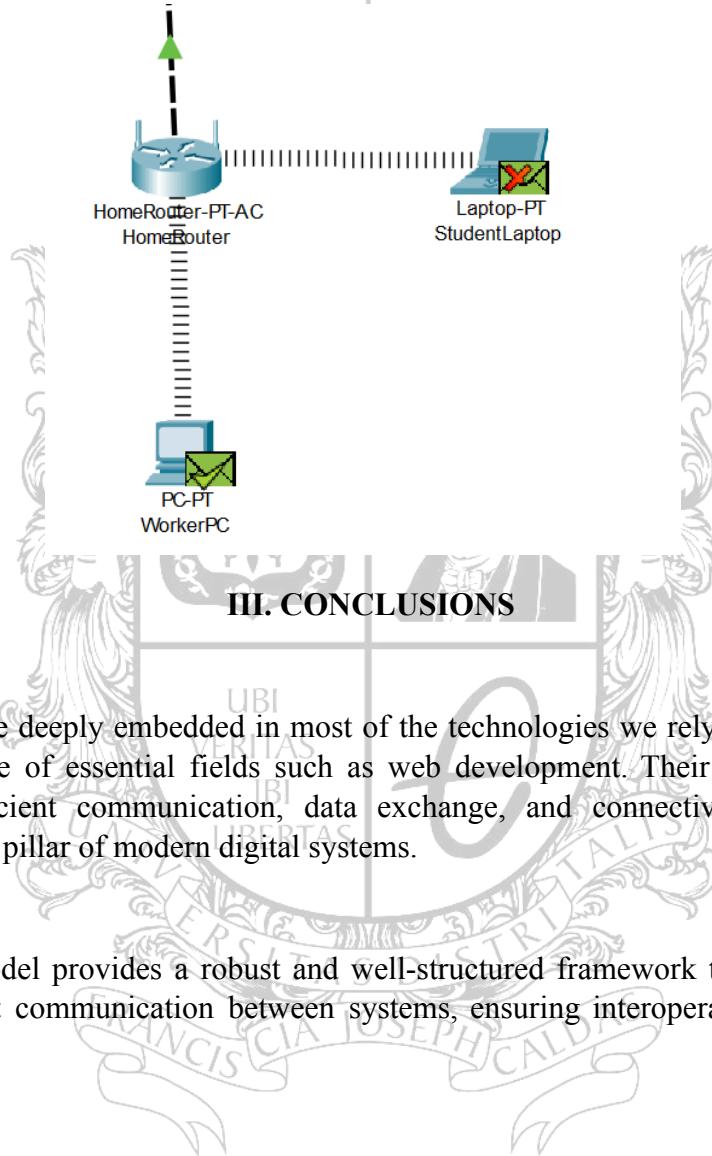


Then after reaching the hub, it starts unpacking the payload and search for the device's IP and MAC in the datalink and network layer.





Then it sends back an answer to the source device notifying a successful connection.



### III. CONCLUSIONS

- Networks are deeply embedded in most of the technologies we rely on today, serving as the backbone of essential fields such as web development. Their seamless integration enables efficient communication, data exchange, and connectivity, making them a fundamental pillar of modern digital systems.
- The OSI model provides a robust and well-structured framework that enables seamless and efficient communication between systems, ensuring interoperability across diverse networks.