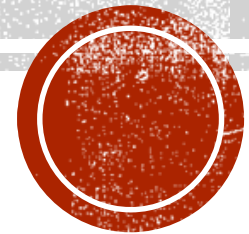


BLOCKCHAIN - PYTHON



Christian Camilo Urcuqui López, MSc

PRESENTACIÓN

Christian Camilo Urcuqui López

Ing. Sistemas, Magister en Informática y Telecomunicaciones

Big Data Professional

Big Data Scientist

Deep Learning Specialization

Cyber Security Data Scientist, LUMU Technologies

Líder de investigación y desarrollo, laboratorio i2t – U ICESI.

ulcamilo@gmail.com

OBJETIVOS DE APRENDIZAJE

Al final de esta actividad, podrá realizar una aplicación que hace uso de una red Blockchain:

- Aplicar el lenguaje de programación Python
- Aplicar las librerías de criptografía Python para el desarrollo de la red Blockchain
- Describir una aplicación web con Blockchain

SOFTWARE Y UTILIDADES

Para esta actividad deberá descargar o instalar las siguientes herramientas:

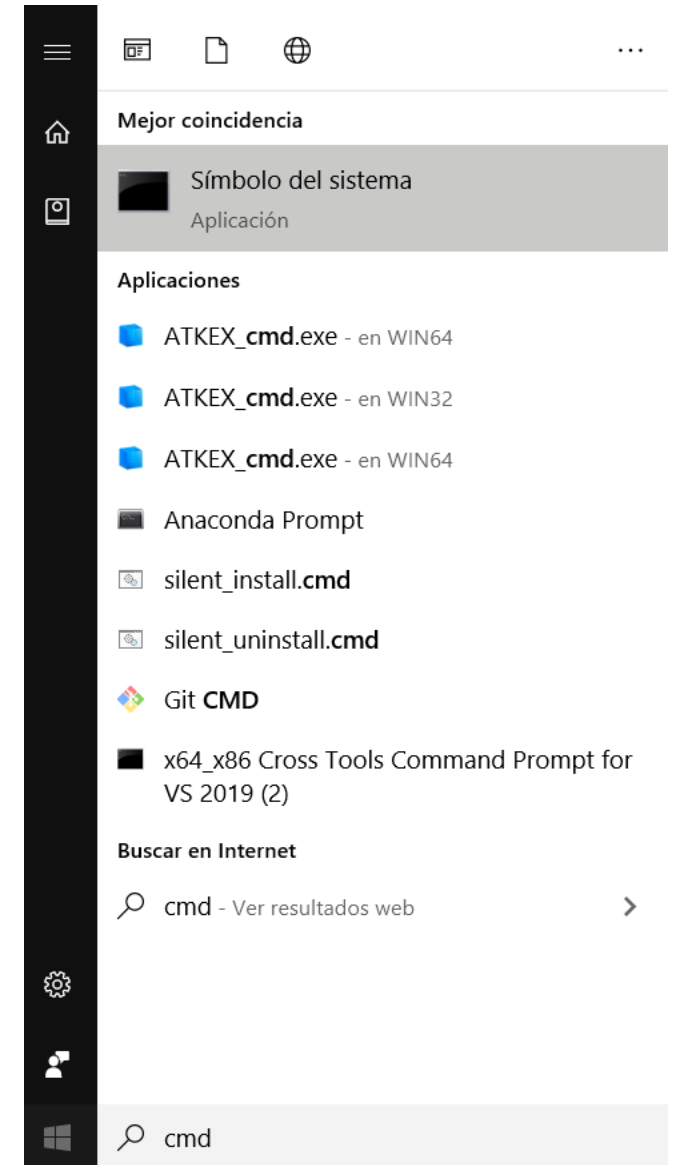
- Descargar e instalar el lenguaje de programación [Python 3.7+](#).
- Descargue de Moodle el zip Blockchain
- Como editor de código se le recomienda el software [Sublime Text](#)

DESCRIPCIÓN DE HERRAMIENTAS

- Python: es un lenguaje de programación
- Sublime Text: es un editor de texto con complementos para programación.
- Blockchain.zip contiene el proyecto de una aplicación web integrada por los componentes para un blockchain

VIRTUALENV

- **Virtualenv** es una herramienta en Python para la instalación de paquetes (librerías) de desarrollo en entornos aislados con el fin de no tener conflictos entre versiones.
- Para instalarlo por favor ingrese a la consola de su sistema operativo, por ejemplo, en Windows abra el menú y escriba “cmd”, posteriormente, de click en “símbolo del sistema”

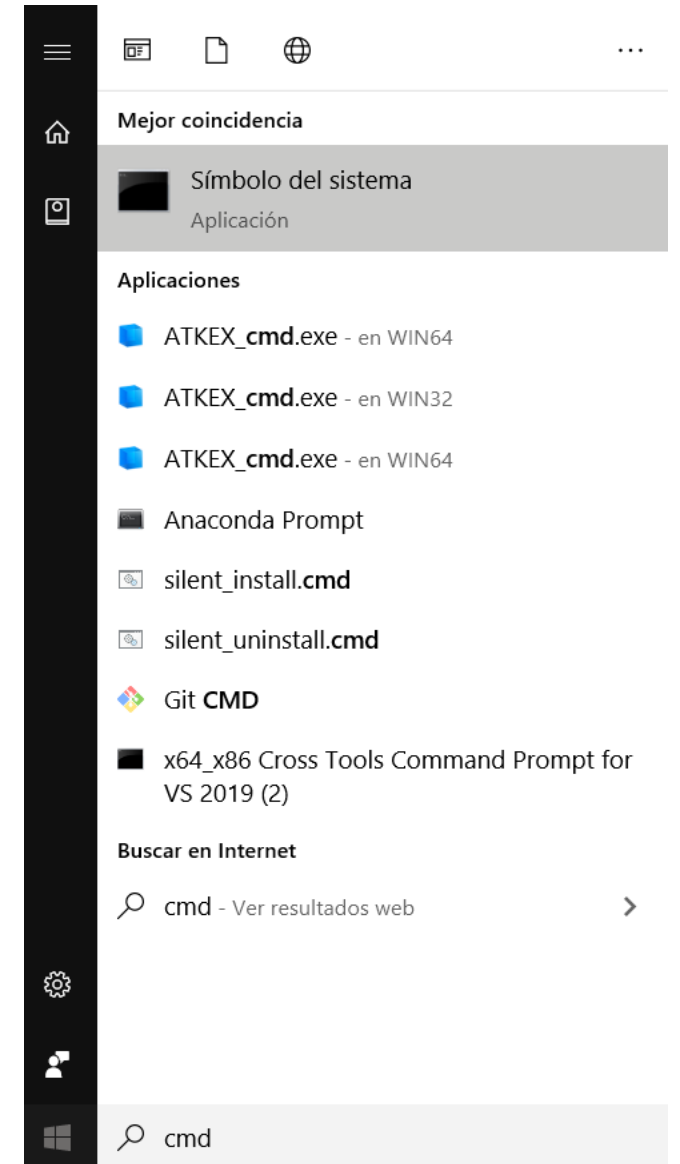


VIRTUALENV

- Una vez en la consola digite el comando
`pip install virtualenv`
- Debería conseguir un resultado como el siguiente

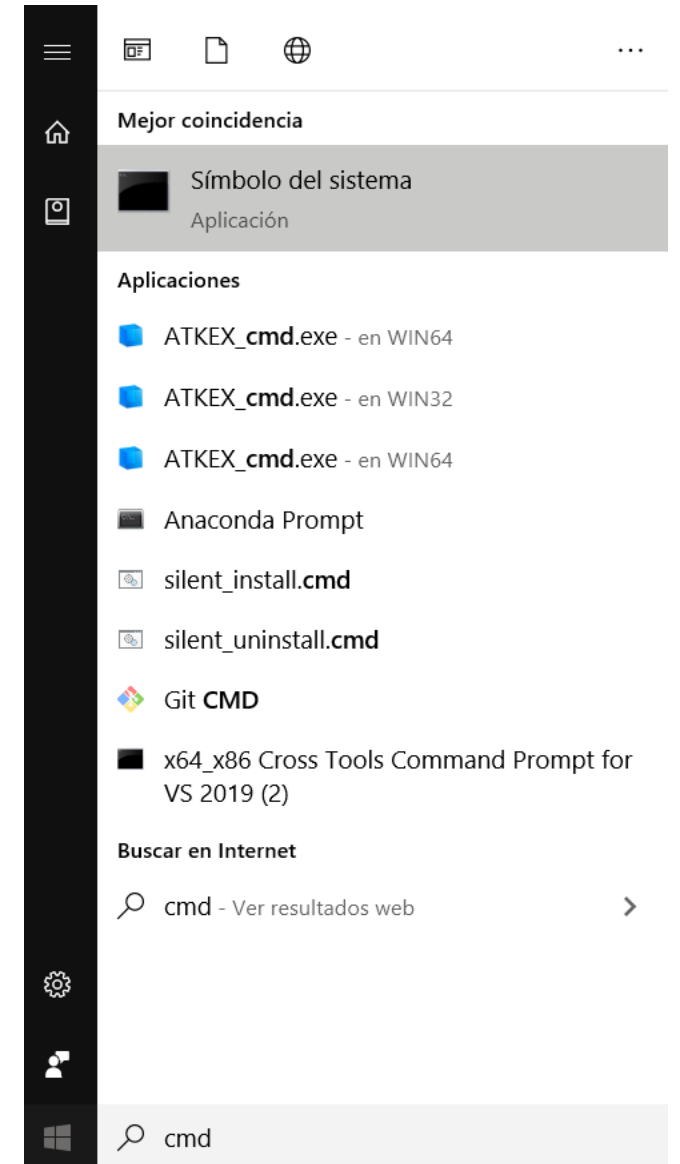
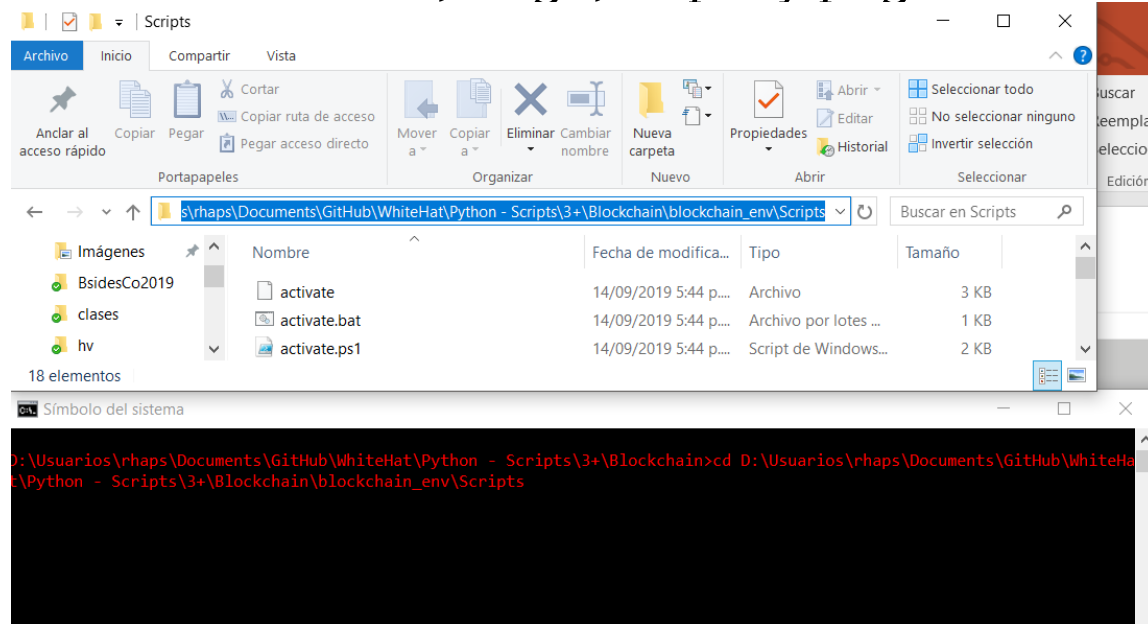
```
D:\Usuarios\rhaps\Documents\GitHub\WhiteHat\Python - Scripts\3+\Blockchain>pip install virtualenv
Collecting virtualenv
  Downloading https://files.pythonhosted.org/packages/8b/12/8d4f45b8962b03ac9efefe5ed5053f6b29334d83e438b4fe379d21c0cb8e/virtualenv-16.7.5-py2.py3-none-any.whl (3.3MB)
    100% |████████████████████████████████████████| 3.3MB 194kB/s
Installing collected packages: virtualenv
  The script virtualenv.exe is installed in 'd:\usuarios\rhaps\appdata\local\programs\python\python37\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed virtualenv-16.7.5
You are using pip version 10.0.1, however version 19.2.3 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

D:\Usuarios\rhaps\Documents\GitHub\WhiteHat\Python - Scripts\3+\Blockchain>
```



VIRTUALENV

- Dentro de la carpeta Blockchain encontrara una carpeta blockchain_env donde se encuentran los paquetes Python necesarios para ejecutar el proyecto web. Proceda a ir hasta la carpeta Scripts que esta dentro de este directorio en la consola de comandos. *!Sugerencia! Escriba el comando **cd**, luego, Copie y pegue la dirección*

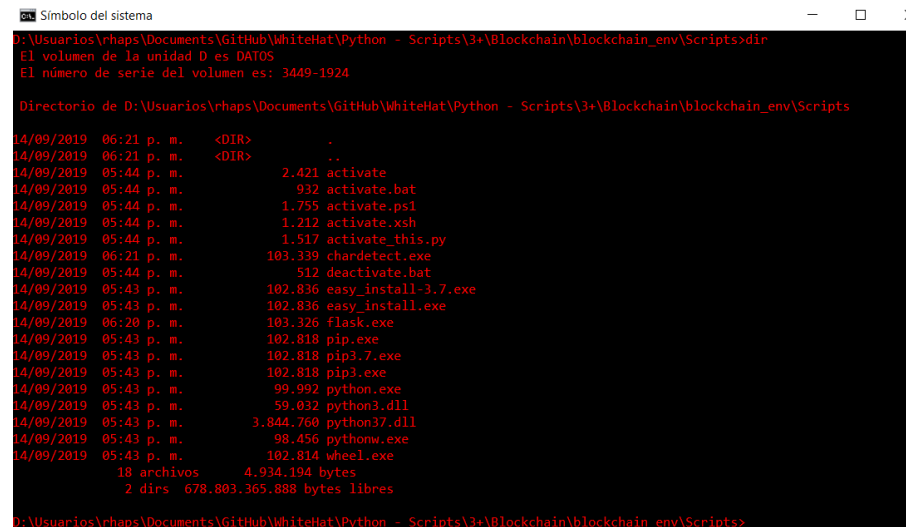


VIRTUALENV

En esta carpeta encontrara dos archivos:

- activate.bat
- deactivate.bat

El primer archivo nos ayudara a activar el entorno virtual y el segundo a desactivarlo, procedamos a ejecutar el ambiente



```
Símbolo del sistema
D:\Usuarios\rhaps\Documents\GitHub\WhiteHat\Python - Scripts\3+\Blockchain\blockchain_env\Scripts>dir
El volumen de la unidad D es DATOS
El número de serie del volumen es: 3449-1924

Directorio de D:\Usuarios\rhaps\Documents\GitHub\WhiteHat\Python - Scripts\3+\Blockchain\blockchain_env\Scripts

14/09/2019  06:21 p. m.    <DIR>          .
14/09/2019  06:21 p. m.    <DIR>          ..
14/09/2019  05:44 p. m.             2.421 activate
14/09/2019  05:44 p. m.             932 activate.bat
14/09/2019  05:44 p. m.             1.755 activate.ps1
14/09/2019  05:44 p. m.             1.212 activate.xsh
14/09/2019  05:44 p. m.             1.517 activate_this.py
14/09/2019  06:21 p. m.            103.339 chardetect.exe
14/09/2019  05:44 p. m.             512 deactivate.bat
14/09/2019  05:43 p. m.            102.836 easy_install-3.7.exe
14/09/2019  05:43 p. m.            102.836 easy_install.exe
14/09/2019  06:20 p. m.            103.326 flask.exe
14/09/2019  05:43 p. m.            102.818 pip.exe
14/09/2019  05:43 p. m.            102.818 pip3.7.exe
14/09/2019  05:43 p. m.            102.818 pip3.exe
14/09/2019  05:43 p. m.             99.992 python.exe
14/09/2019  05:43 p. m.             59.032 python3.dll
14/09/2019  05:43 p. m.            3.844.760 python37.dll
14/09/2019  05:43 p. m.             98.456 pythonw.exe
14/09/2019  05:43 p. m.            102.814 wheel.exe

               18 archivos          4.934.194 bytes
                 2 dirs  678.803.365.888 bytes libres

D:\Usuarios\rhaps\Documents\GitHub\WhiteHat\Python - Scripts\3+\Blockchain\blockchain_env\Scripts>
```

VIRTUALENV








```
Símbolo del sistema
D:\Usuarios\rhaps\Documents\GitHub\WhiteHat\Python - Scripts\3+\Blockchain\blockchain_env\Scripts>activate.bat
(BLOK~1) D:\Usuarios\rhaps\Documents\GitHub\WhiteHat\Python - Scripts\3+\Blockchain\blockchain_env\Scripts>
```

Observe el cambio, esto quiere decir que ahora nos encontramos en el entorno de trabajo.

EJECUTANDO EL PROYECTO

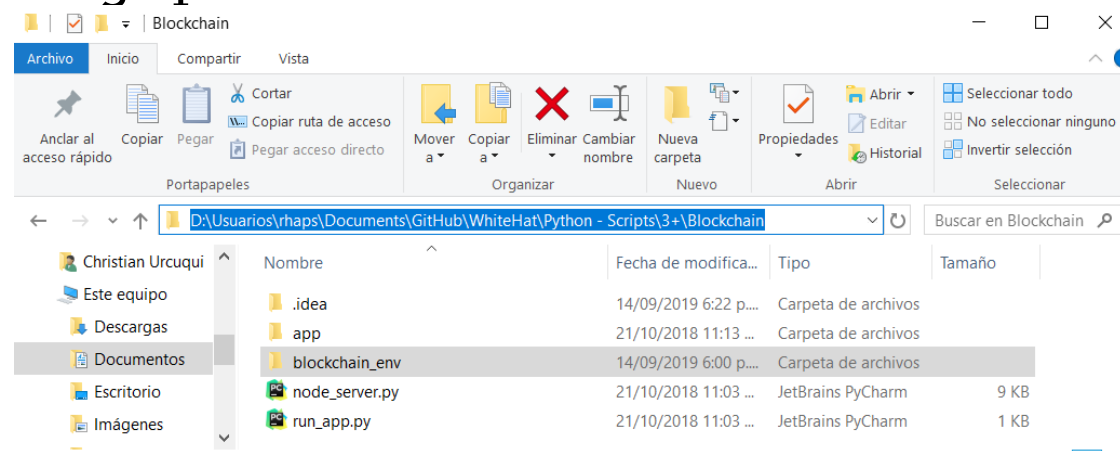
El proyecto web se encuentra dividido en dos secciones:

- El archivo `node_server.py` es el código Python que contiene la estructura de bloques y cadenas, observe su contenido con el programa de edición de texto.
- El archivo `run_app.py` es el encargado de ejecutar la aplicación web.
- La carpeta `app` contiene el archivo `views.py` y los `html`, es decir, el código encargado de recibir las interacciones de los usuarios y desplegar las visualizaciones de las páginas web.

-  `.idea`
-  `app`
-  `blockchain_env`
-  `node_server.py`
-  `run_app.py`

EJECUTANDO EL PROYECTO

Ahora proceda a abrir otra consola y haga el mismo proceso de activación, es decir, debería tener dos ejecutándose. En ambas ventanas vaya hasta la carpeta de blockchain que descargo previamente.



Símbolo del sistema

```
(BLOCKC~1) D:\Usuarios\rhaps\Documents\GitHub\WhiteHat\Python - Scripts\3+\Blockchain>
```

Símbolo del sistema

```
n_env\Scripts>cd ..  
(BLOCKC~1) D:\Usuarios\rhaps\Documents\GitHub\WhiteHat\Python - Scripts\3+\Blockchain\blockchai  
n_env>cd ..
```

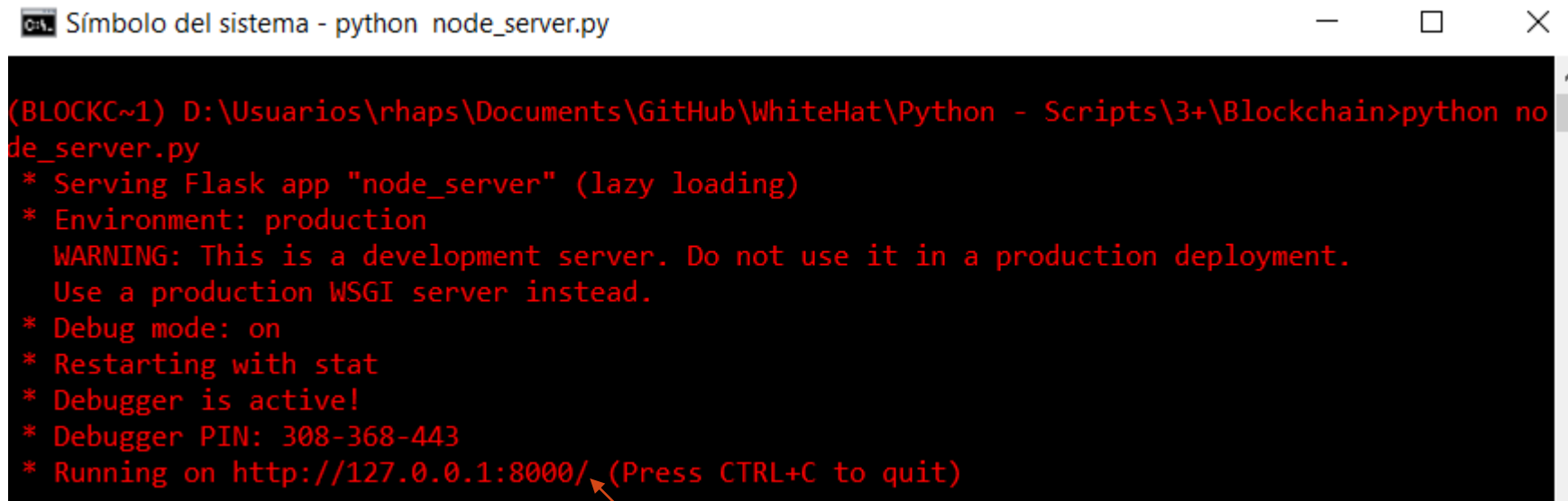
```
(BLOCKC~1) D:\Usuarios\rhaps\Documents\GitHub\WhiteHat\Python - Scripts\3+\Blockchain>
```

EJECUTANDO EL PROYECTO

Realice las actividades en el debido orden

- En una de las consolas proceda a escribir el siguiente comando

`python node_server.py`



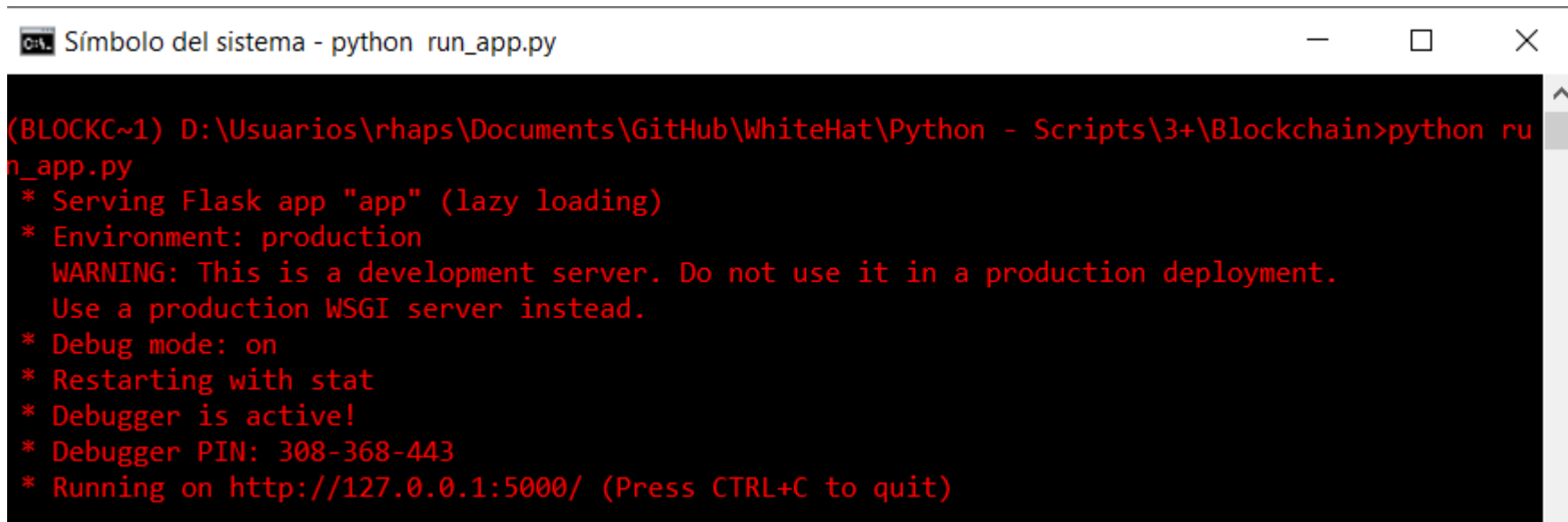
```
(BLOCKC~1) D:\Usuarios\rhaps\Documents\GitHub\WhiteHat\Python - Scripts\3+\Blockchain>python node_server.py
* Serving Flask app "node_server" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 308-368-443
* Running on http://127.0.0.1:8000/ (Press CTRL+C to quit)
```

EJECUTANDO EL PROYECTO

Realice las actividades en el debido orden

- En la consola faltante escriba el siguiente comando

`python app_run.py`

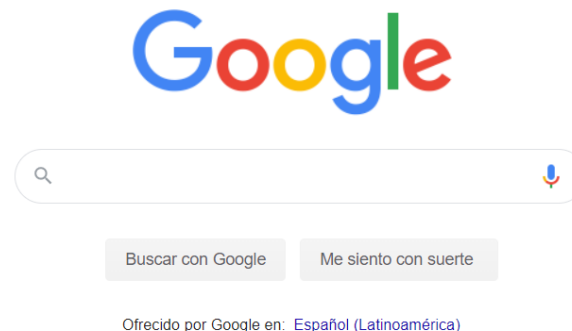
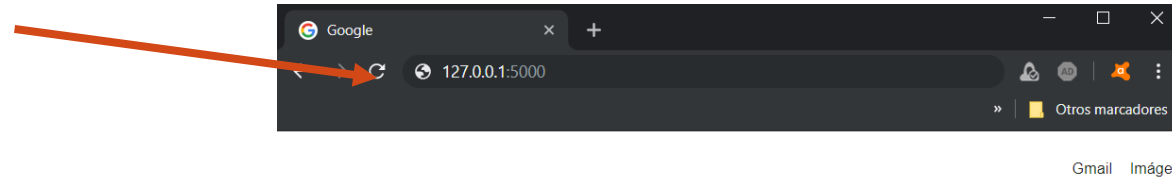


```
(BLOCKC~1) D:\Usuarios\rhaps\Documents\GitHub\WhiteHat\Python - Scripts\3+\Blockchain>python run_app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 308-368-443
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

EJECUTANDO EL PROYECTO

Si tenemos ambas consolas con los mensajes de **running**, esto nos quiere decir que la aplicación con Blockchain se encuentra en funcionamiento.

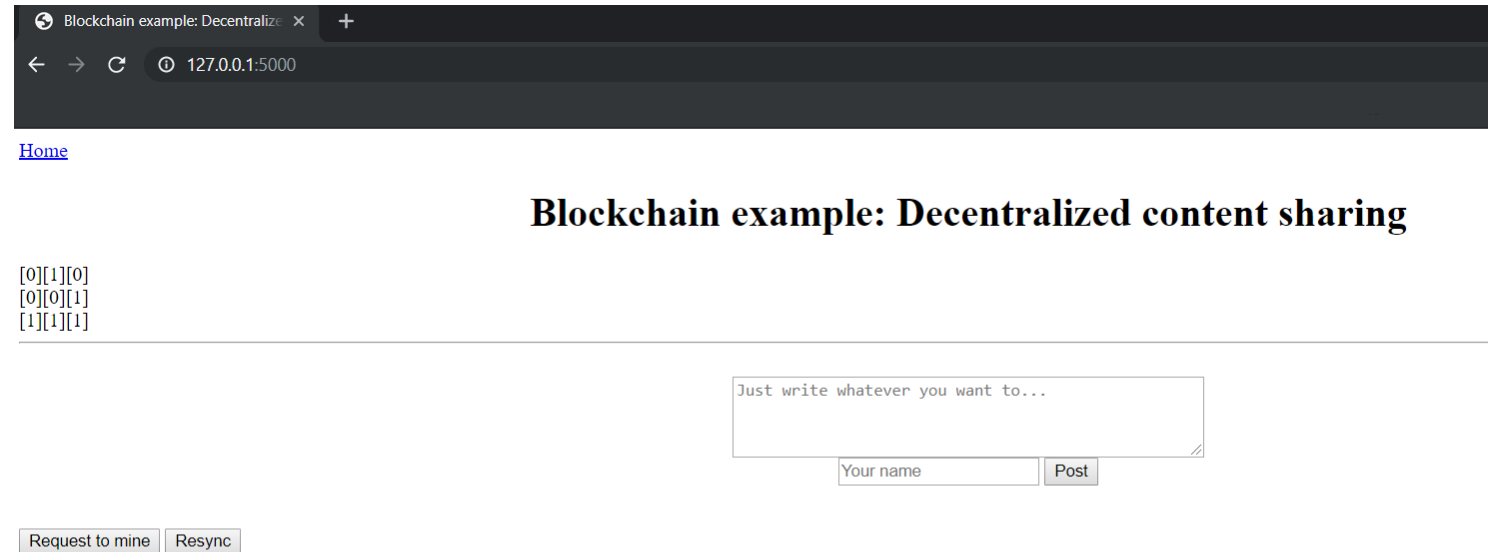
Para acceder a esta abra su navegador web de preferencia y escriba la dirección que despliega el resultado de app_run, en mi caso es la dirección 127.0.0.1:5000



EJECUTANDO EL PROYECTO

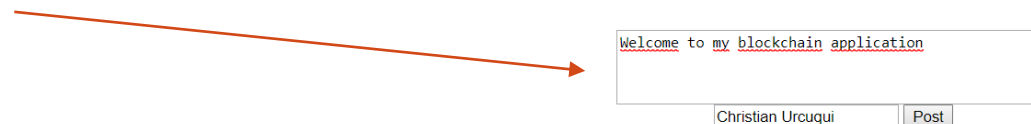
Obtendremos el siguiente resultado.

La página web de inicio nos permitirá ingresar post y enlazarlos a nombre.



- *Proceda a escribir un post y de click en botón "Post"*

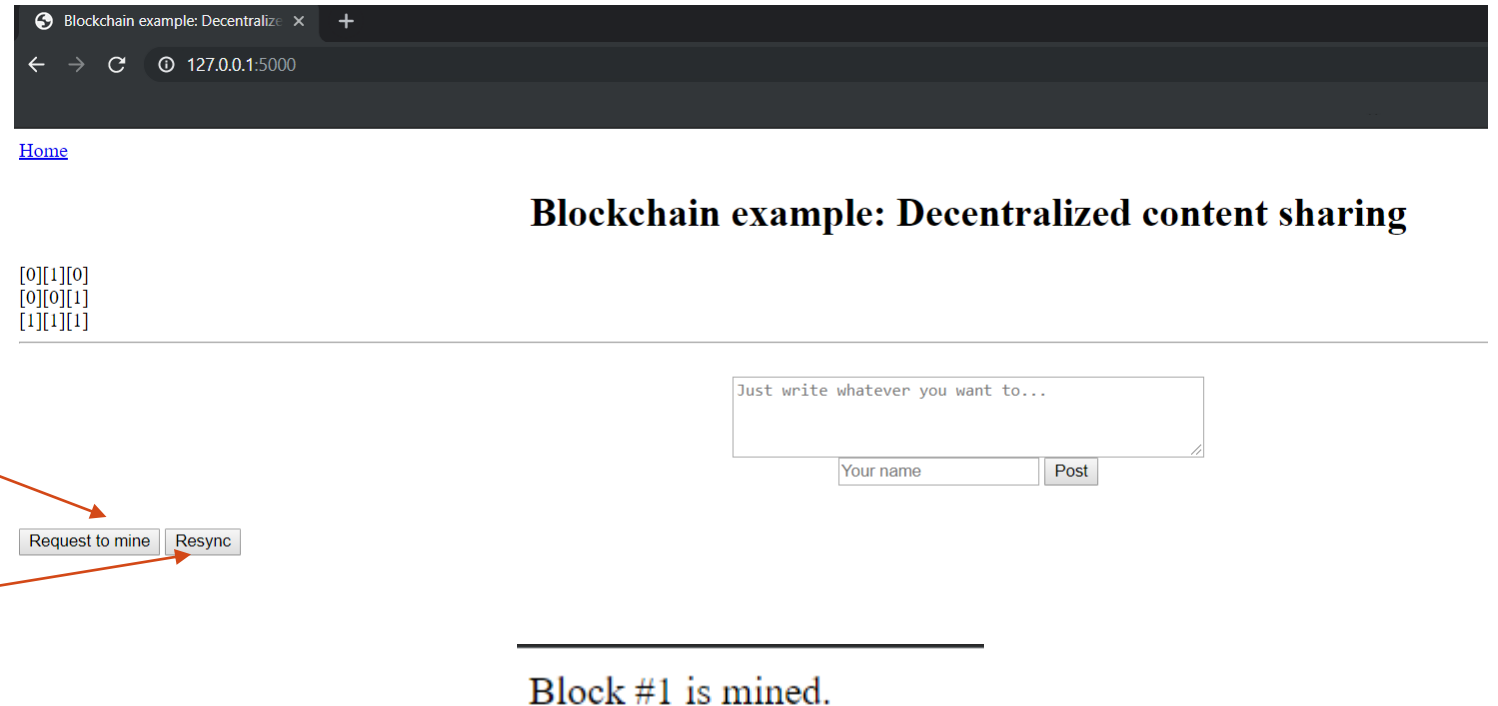
Blockchain example: Decentralized content sharing



EJECUTANDO EL PROYECTO

Por ahora el post no se ha propagado y validado por los integrantes de la red de blockchain.

- Proceda a dar click en “Request to mine”.
- Luego de refresque la página en “Resync”



EJECUTANDO EL PROYECTO

[Home](#)

Blockchain example: Decentralized content sharing

[0][1][0]
[0][0][1]
[1][1][1]

Just write whatever you want to...

Your name

Post

Request to mine

Resync



Christian Urcuqui
Posted at 21:48

Reply

My second post



Christian Urcuqui
Posted at 21:37

Reply

welcome to a blockchain application

Ahora contamos con una aplicación web para postear comentarios en una red Blockchain, es decir, todos los mensajes están asegurados y a su vez notificados a los participantes.

REVISANDO EL CÓDIGO

Procedamos a abrir el archivo “node_server.py”

- *hashlib* librería que importa el método de encriptación sha256
- *json* para manejo de los servicios web
- *flask* es una tecnología para el servidor web (backend)
- *requests* para envío y recepción de peticiones web

```
node_server.py x
4  from hashlib import sha256
5  import json
6  import time
7  from flask import Flask, request
8  import requests
9
10
11  # All the transactions are packaged in blocks and their information are hashed
12  # the blocks are chained with their hash, that is the reason that we are using
13  # it in the __init__ method as a parameter
14  class Block:
15      def __init__(self, index, transactions, timestamp, previous_hash):
16          self.index = index
17          self.transactions = transactions
18          self.timestamp = timestamp
19          self.previous_hash = previous_hash
20          # nonce is a number that is going to change until the hash is going to be correct
21          self.nonce = 0
22
23      def compute_hash(self):
24          """
25          This function returns the hash of the block contents
26          :return:
27          """
28          # dumps allows us to serialize obj to a JSON formatted str
29          block_string = json.dumps(self.__dict__, sort_keys=True)
30          return sha256(block_string.encode()).hexdigest()
31
32
33  class Blockchain:
34      # difficulty of our PoW algorithm
35      difficulty = 2
36
37      def __init__(self):
38          self.unconfirmed_transactions = []
39          self.chain = []
```

REVISANDO EL CÓDIGO

- Método para iniciar una red blockchain, es decir, tiene un conjunto de cadenas vacías y transacciones vacías.
- Método encargado de crear el bloque padre con un índice de cero, sin transacciones y enlazarlo a una primera cadena encriptada con sha256

```
class Blockchain:
    # difficulty of our PoW algorithm
    difficulty = 2

    def __init__(self):
        self.unconfirmed_transactions = []
        self.chain = []
        self.create_genesis_block()

    def create_genesis_block(self):
        """
        A function to generate genesis block and appends it to the chain.
        The block has index 0, previous_hash as 0, and a valid hash
        :return:
        """
        genesis_block = Block(0, [], time.time(), "0")
        genesis_block.hash = genesis_block.compute_hash()
        self.chain.append(genesis_block)

    @property
    def last_block(self):
        return self.chain[-1]
```

REVISANDO EL CÓDIGO

- Retorna la última cadena creada dentro de la red
- Agrega un nuevo bloque a la cadena siempre y cuando el hash o el código de encriptación no se encuentre en la red

```
@property
def last_block(self):
    return self.chain[-1]

def add_block(self, block, proof):
    """
    A function that adds the block to the chain after verification
    Verification includes:
    * Checking if the proof is valid
    * The previous_hash referred in the block and the hash of latest block
    in the chain match
    :param block:
    :param proof:
    :return:
    """
    previous_hash = self.last_block.hash

    if previous_hash != block.previous_hash:
        return False

    if not Blockchain.is_valid_proof(block, proof):
        return False

    block.hash = proof
    self.chain.append(block)
    return True
```

REVISANDO EL CÓDIGO

- Encargado de aplicar y comprobar la complejidad en el descifrado
- Adiciona una nueva transacción
- Valida si es un hash valido

```
# the next method is going to calculate the hash through the nonce value, but, it is like a brute force operation
def proof_of_work(self, block):
    """
    Function that tries different values of nonce to get a hash
    that satisfies our difficulty criteria
    :param block:
    :return:
    """
    block.nonce = 0

    computed_hash = block.compute_hash()
    while not computed_hash.startswith('0' * Blockchain.difficulty):
        block.nonce += 1
        computed_hash = block.compute_hash()

    return computed_hash

def add_new_transaction(self, transaction):
    self.unconfirmed_transactions.append(transaction)

@classmethod
def is_valid_proof(cls, block, block_hash):
    """
    Check if block_hash is valid hash of block and satisfies
    the difficulty criteria.
    :param block:
    :param block_hash:
    :return:
    """
    return (block_hash.startswith('0' * Blockchain.difficulty) and
            (block_hash == block.compute_hash()))
```

REVISANDO EL CÓDIGO

- Este es el método encargado de agregar las transacciones a un nuevo bloque de la red Blockchain

```
def mine(self):  
    """  
    This function servers as an interface to add the pending  
    transactions to the blockchain by adding them to the block  
    and figuring out Proof Of Work  
    :param self:  
    :return:  
    """  
  
    if not self.unconfirmed_transactions:  
        False  
  
    last_block = self.last_block  
  
    new_block = Block(index=last_block.index + 1,  
                      transactions=self.unconfirmed_transactions,  
                      timestamp=time.time(),  
                      previous_hash=last_block.hash)  
  
    proof = self.proof_of_work(new_block)  
    self.add_block(new_block, proof)  
  
    self.unconfirmed_transactions = []  
    # announce it to the network  
    announce_new_block(new_block)  
    return new_block.index
```

REVISANDO EL CÓDIGO

Los siguientes métodos interactúan directamente con la aplicación web

- Obtiene información de una cadena
- Retorna información acerca si la transacción fue creada como un bloque en la red.
- Adiciona nodos a la red a partir de una solicitud desde la web

```
@app.route('/chain', methods=['GET'])
def get_chain():
    # make sure we've the longest chain
    consensus()
    chain_data = []
    for block in blockchain.chain:
        chain_data.append(block.__dict__)
    return json.dumps({"length": len(chain_data),
                      "chain": chain_data})

# endpoint to represent the node to mine the unconfirmed
# transactions (if any). We will be using it to initiate
# a command to mine from our application itself
@app.route("/mine", methods=['GET'])
def mine_unconfirmed_transactions():
    result = blockchain.mine()
    if not result:
        return "No transactions to mine"
    return "Block #{0} is mined.".format(result)

# endpoint to add new peers to the network
@app.route('/app_nodes', methods=['POST'])
def register_new_peers():
    nodes = request.get_json()
    if not nodes:
        return "Invalid data", 400
    for node in nodes:
        peers.add(node)

    return "Success", 201
```


REVISANDO EL CÓDIGO

Los siguientes métodos interactúan directamente con la aplicación web

- Obtiene información de la transacción para agregarla a la red
- Encargado de dar información a la aplicación web sobre los datos de las cadenas

```
# endpoint to submit a new transaction. This will be used by
# our application to add new data (posts) to the blockchain
@app.route('/new_transaction', methods=['POST'])
def new_transaction():
    tx_data = request.get_json()
    required_fields = ["author", "content"]

    for field in required_fields:
        if not tx_data.get(field):
            return "Invalid transaction data", 404

    tx_data["timestamp"] = time.time()

    blockchain.add_new_transaction(tx_data)

    return "Success", 201

# endpoint to return the node's copy of the chain.
# Our application will be using this endpoint to query
# all the posts to display.
@app.route('/chain', methods=['GET'])
def get_chain():
    # make sure we've the longest chain
    consensus()
    chain_data = []
    for block in blockchain.chain:
        chain_data.append(block.__dict__)
    return json.dumps({"length": len(chain_data),
                      "chain": chain_data})
```

REVISANDO EL CÓDIGO

- Encargado de validar las cadenas
- La función del siguiente método es de notificar a los otros nodos cuando un nuevo bloque se ha agregado.

Los otros archivos del proyecto están dedicados a soportar la aplicación web que se encuentra sobre el marco de trabajo Flask.

Ahora cuenta con una aplicación web basada en una red Blockchain, un desarrollo que se encuentra bajo un esquema de código abierto.

```
def consensus():
    """
    Our simple consensus algorithm. If a longer valid chain is
    found, our chain is replaced with it
    :return:
    """

    global blockchain

    longest_chain = None
    current_len = len(blockchain.chain)

    for node in peers:
        response = requests.get('http://{}/chain'.format(node))
        length = response.json()['length']
        chain = response.json()['chain']
        if length > current_len and blockchain.check_chain_validity(chain):
            current_len = length
            longest_chain = chain

    if longest_chain:
        blockchain = longest_chain
        return True
    return False

def announce_new_block(block):
    """
    A function to announce to the network once a block has been mined.
    Other blocks can simply verify the proof of work and add it to their
    respective chains.
    """

    for peer in peers:
        url = "http://{}/add_block".format(peer)
        requests.post(url, data=json.dumps(block.__dict__, sort_keys=True))
```

OTROS ENLACES POR SI ES DE SU INTERÉS

- Introducción a Python

<https://github.com/urcuqui/Data-Science/tree/master/My%20Courses/Python%20Introduction/0%20-%20Basic%20functions>

- Repositorio del código de Blockchain

<https://github.com/urcuqui/WhiteHat/tree/master/Python%20-%20Scripts/3%2B/Blockchain>

- Página oficial de Ethereum

<https://www.ethereum.org/>