blockchain POC Documentation

Release latest

Tomas Vukasovic

Nov 05, 2018

Contents:

This project is a POC (Proof of concept) of a public 1.0 blockchain, covering the requirements at the 'white paper' of bitcoin. We intended to understand more about how blockchains worked and how we could implement one using Python with Flask micro web framework and Docker.

CHAPTER 1

QuickStart

The project can be used in two ways either by using Docker or setting it up locally, we highly recommend using Docker as it removes complexity.

1.1 Requirements

- Docker
- Docker-compose
- Python3
- PIP
- GIT

1.2 Building the Blockchain

Before we run anything we need to download the project:

```
>> git clone https://github.com/tvukasovic/blockchain
```

Once the repository has been cloned we can either build the docker image or run it locally

1.2.1 Docker

Docker install only needs one command:

```
>> cd /path/to/project/blockchain
>> docker-compose build -f docker/Dockerfile blockchain
```

Thats it, we just have to wait for the image to finish building. The advantage on this approach is that we are not installing anything on our local machines we build everything inside the container, once we are done we can delete it without having any side effects or having packages that we won't use in the future.

1.2.2 Locally

To make the local installation we will need to run the following commands:

```
>> cd /path/to/project/blockchain/docker
>> pip install requirements.txt
```

Once PIP is done installing all the python packages, we will have to install textract. Following THIS tutorial.

Ater textract installation its done we are ready to run the blockchain.

1.3 Running the Blockchain

1.3.1 Docker

If you built the docker image, we just have to run it and thats that by doing:

```
>> docker-compose -f docker/docker-compose.yml up
```

And thats it. The blockchain is up and running at the port 5000.

You can try it by going to the browser or to Postman and making a GET request to:

```
localhost:5000/chain
```

You may also try to start a bash session inside the docker container to check the files by running the command:

```
>> docker-compose -f docker/docker-compose.yml run blockchain bash
```

1.3.2 Locally

Running the project locally demands a few extra steps:

```
>> cd /path/to/project/blockchain/src
>> export FLASK_APP=app.py
>> flask run
```

And thats it!

Congratulations you have build and run the blockchain POC, now its time to use it!

CHAPTER 2

Running the blockchain!

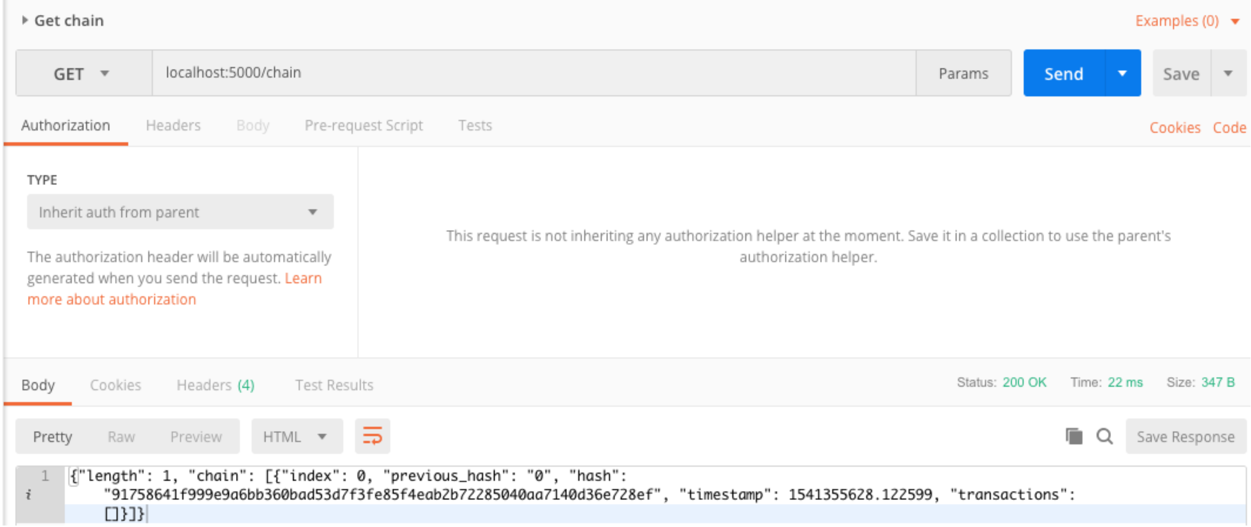
Now is where the fun begins! This example follows the local instalation result. If you are using the docker instalation the only difference is that the PDF files uploaded to the blockchain should be placed inside the files folder of the project. And at the body of the request at the new_transaction endpoint you will write something like:

```
"file": "files/<name-of-file>.pdf"
```

Lets start by obtaining a fresh chain. Go to postman and run this endpoint:

```
localhost:5000/chain
```

You should see something like this:



Get chain

GET localhost:5000/chain

TYPE

Inherit auth from parent

The authorization header will be automatically generated when you send the request. Learn more about authorization

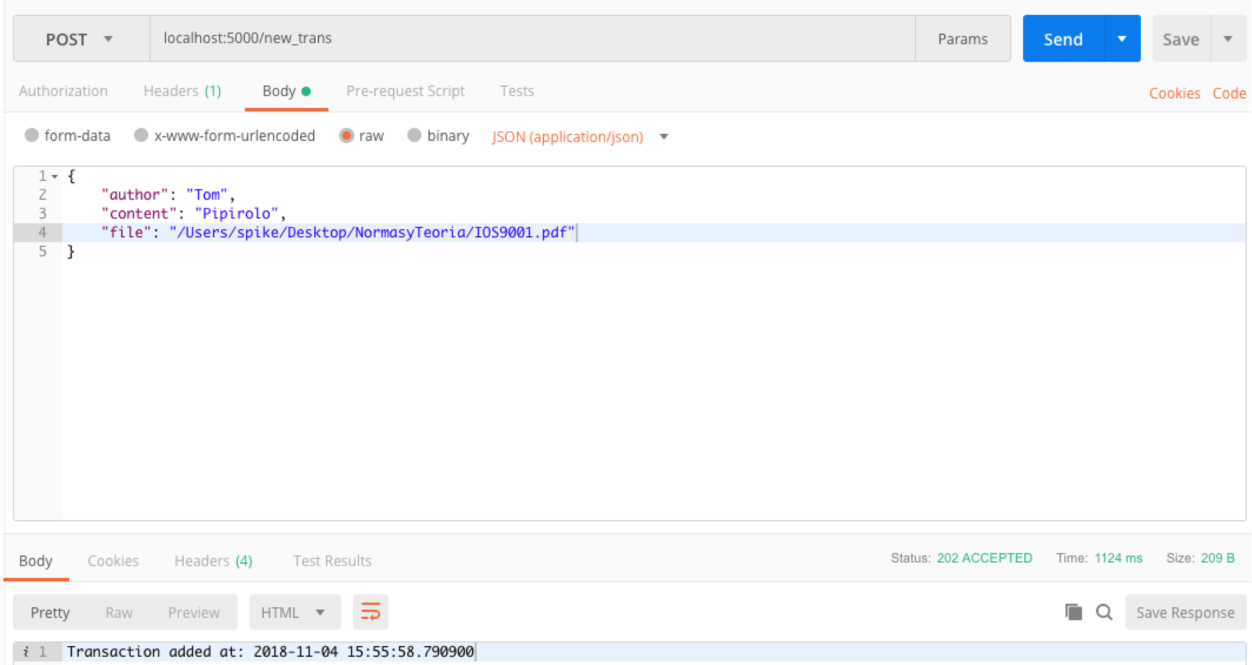
This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent's authorization helper.

Status: 200 OK Time: 22 ms Size: 347 B

```
{"length": 1, "chain": [{"index": 0, "previous_hash": "0", "hash":
    "91758641f999e9a6bb360bad53d7f3fe85f4eab2b72285040aa7140d36e728ef", "timestamp": 1541355628.122599, "transactions":
    []}]}
```

Now its time to add a new transaction, head to postman again and we go to the endopint and post a transaction as follows:

```
localhost:5000/new_trans
```



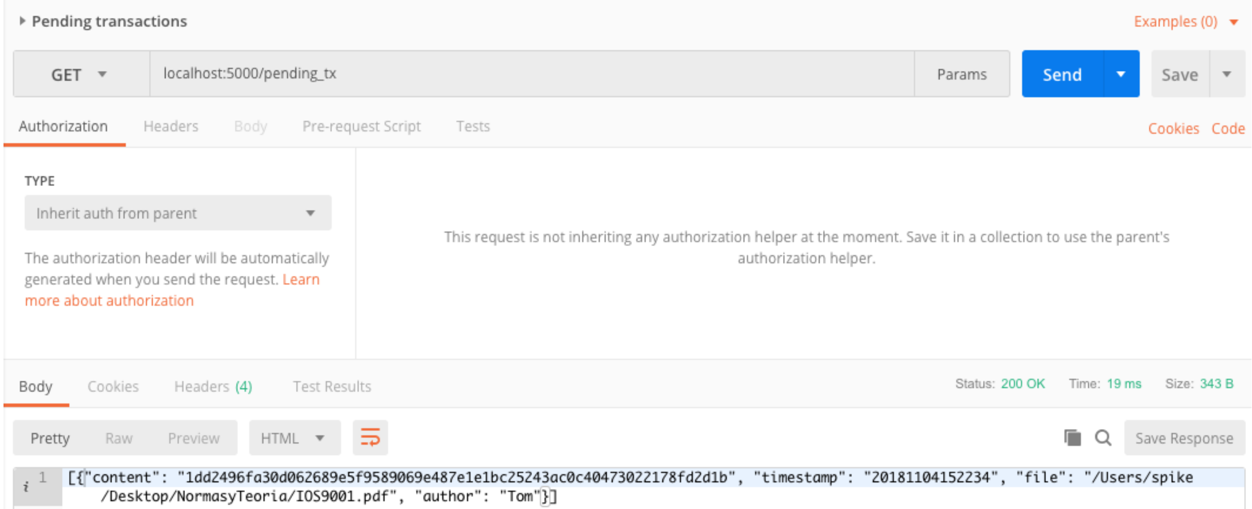
```
{
    "author": "Tom",
    "content": "Pipirolo",
    "file": "/Users/spike/Desktop/NormasyTeoria/IOS9001.pdf"
}
```

Status: 202 ACCEPTED Time: 1124 ms Size: 209 B

```
Transaction added at: 2018-11-04 15:55:58.790900
```

Lets check now that the transaction has been posted and its pending to be processed:

```
localhost:5000/pending_tx
```



The authorization header will be automatically generated when you send the request. Learn more about authorization

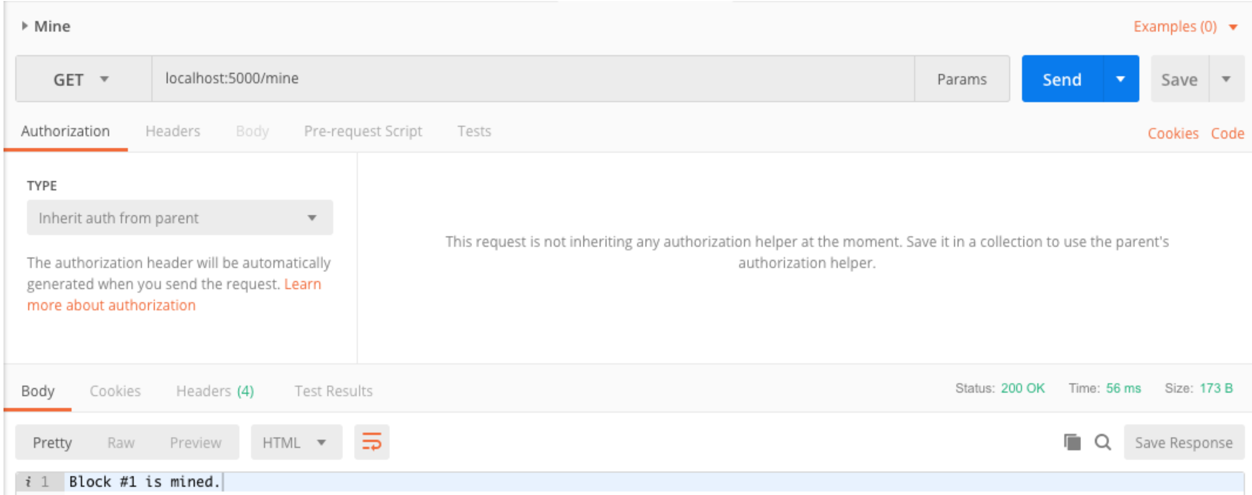
This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent's authorization helper.

Status: 200 OK Time: 19 ms Size: 343 B

```
[{"content": "1dd2496fa30d062689e5f9589069e487e1e1bc25243ac0c40473022178fd2d1b", "timestamp": "20181104152234", "file": "/Users/spike
    /Desktop/NormasyTeoria/IOS9001.pdf", "author": "Tom"}]
```

Now that there is a transaction pending, lets mine it so it is appended into a new block that will be part of the blockchain:

```
localhost:5000/mine
```



localhost:5000/mine

The authorization header will be automatically generated when you send the request. Learn more about authorization

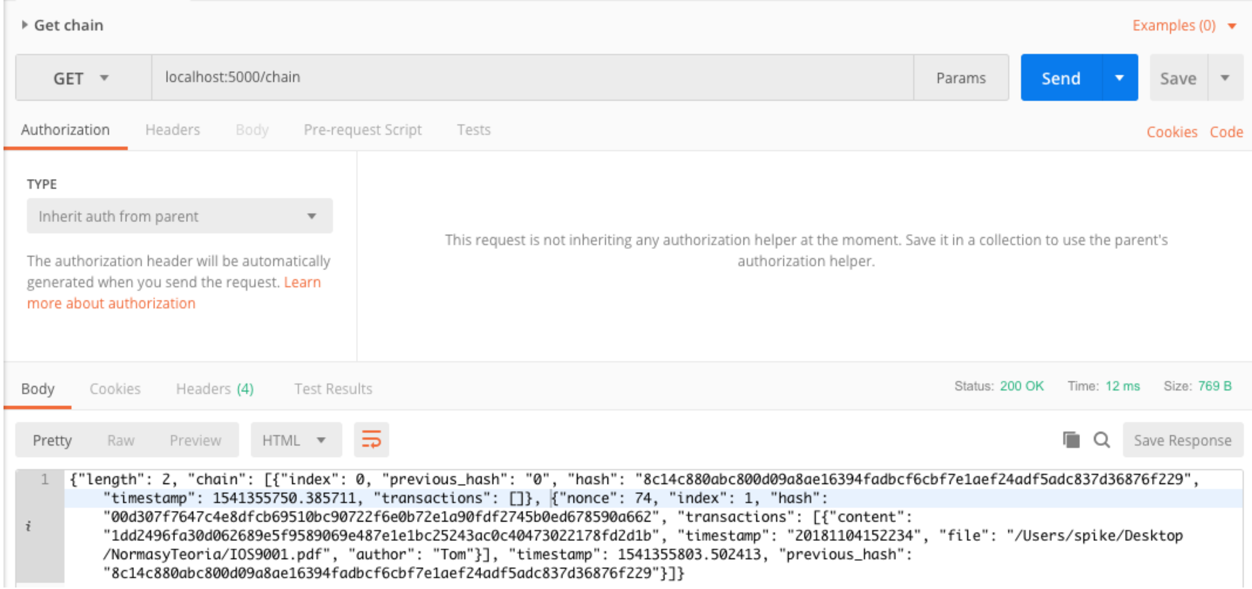
This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent's authorization helper.

Status: 200 OK Time: 56 ms Size: 173 B

```
Block #1 is mined.
```

Finally lets see the results! Head to the first endpoint and see the result. A new block has been added and it has our transaction!:

```
localhost:5000/chain
```



localhost:5000/chain

The authorization header will be automatically generated when you send the request. Learn more about authorization

This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent's authorization helper.

Status: 200 OK Time: 12 ms Size: 769 B

```
{"length": 2, "chain": [{"index": 0, "previous_hash": "0", "hash": "8c14c880abc800d09a8ae16394fadbcf6cbf7e1aef24adf5adc837d36876f229",
    "timestamp": 1541355750.385711, "transactions": []}, {"nonce": 74, "index": 1, "hash":
    "00d307f7647c4e8dfcb69510bc90722f6e0b72e1a90fdf2745b0ed678590a662", "transactions": [{"content":
    "1dd2496fa30d062689e5f9589069e487e1e1bc25243ac0c40473022178fd2d1b", "timestamp": "20181104152234", "file": "/Users/spike/Desktop
    /NormasyTeoria/IOS9001.pdf", "author": "Tom"}], "timestamp": 1541355803.502413, "previous_hash":
    "8c14c880abc800d09a8ae16394fadbcf6cbf7e1aef24adf5adc837d36876f229"}]}
```

CHAPTER 3

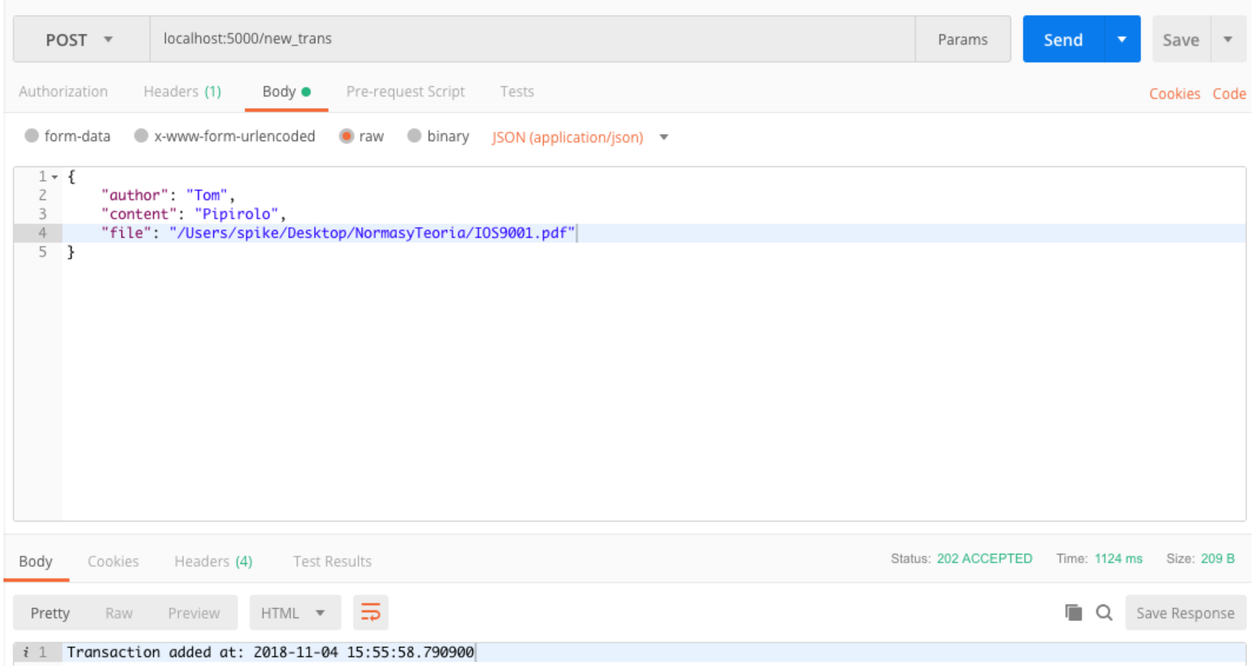
Integration Test for files

We are going to make a small test, when we add a file we obtain a hash of its content but if we change the integrity of the files content we should obtain a different hasn. Lets make the test!

1. Lets add our original file and check the hash of the content! The hash we obtained for the file is:

```
1dd2496fa30d062689e5f9589069e487e1e1bc25243ac0c40473022178fd2d1b
```

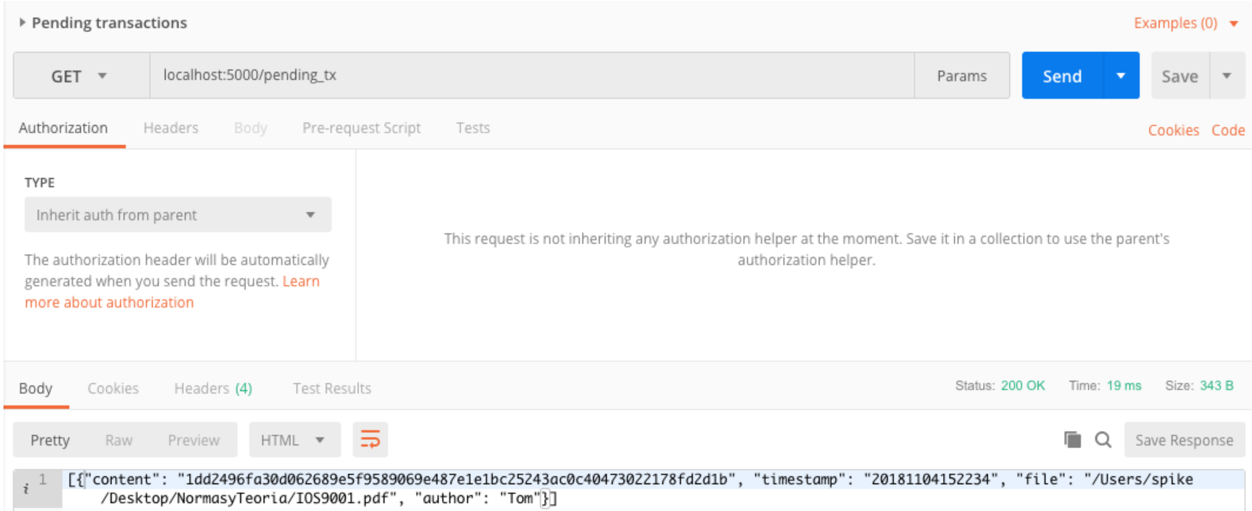
Adding the transaction



```
{
    "author": "Tom",
    "content": "Pipirolo",
    "file": "/Users/spike/Desktop/NormasyTeoria/IOS9001.pdf"
}
```

```
Transaction added at: 2018-11-04 15:55:58.790900
```

Checking the transaction



localhost:5000/pending_tx

The authorization header will be automatically generated when you send the request. Learn more about authorization

This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent's authorization helper.

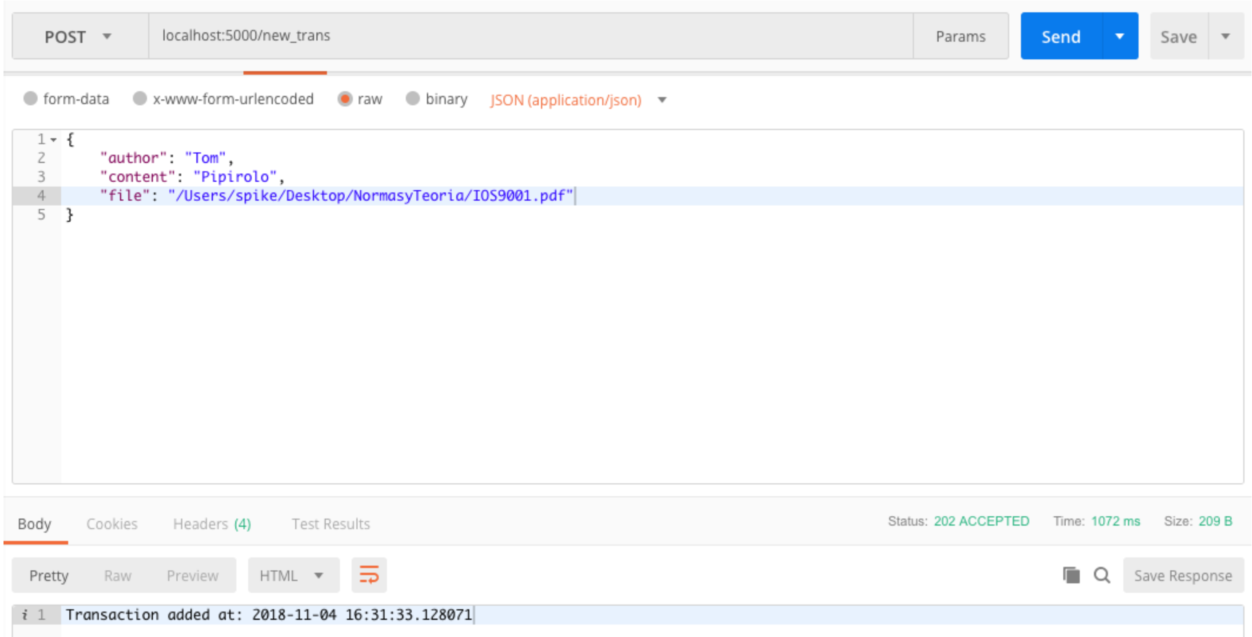
Status: 200 OK Time: 19 ms Size: 343 B

```
[{"content": "1dd2496fa30d062689e5f9589069e487e1e1bc25243ac0c40473022178fd2d1b", "timestamp": "20181104152234", "file": "/Users/spike
    /Desktop/NormasyTeoria/IOS9001.pdf", "author": "Tom"}]
```

2. Now lets modify the file by removing some pages and lets upload it as a new transaction, the hash we now obtained is:

```
684bd9509d71f417a914db7a4514d247a8fa347f79e657122914e6562361a7ee
```

Adding the transaction



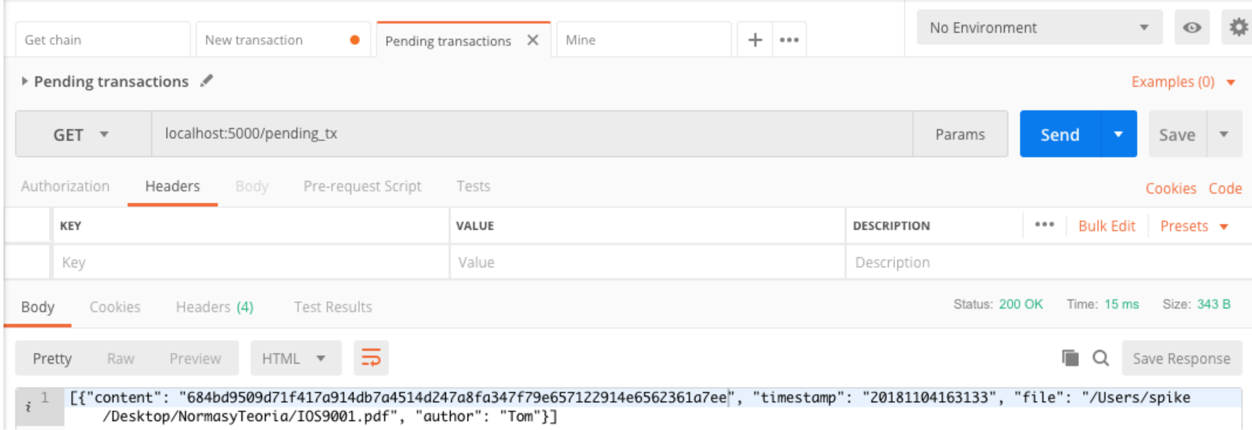
localhost:5000/new_trans

```
{
    "author": "Tom",
    "content": "Pipirolo",
    "file": "/Users/spike/Desktop/NormasyTeoria/IOS9001.pdf"
}
```

Status: 202 ACCEPTED Time: 1072 ms Size: 209 B

```
Transaction added at: 2018-11-04 16:31:33.128071
```

Checking the transaction



localhost:5000/pending_tx

Status: 200 OK Time: 15 ms Size: 343 B

```
[{"content": "684bd9509d71f417a914db7a4514d247a8fa347f79e657122914e6562361a7ee", "timestamp": "20181104163133", "file": "/Users/spike
    /Desktop/NormasyTeoria/IOS9001.pdf", "author": "Tom"}]
```

Thats it! Our hashes are different! Therefor we maintain the integrity and with the timestamp of our original transaction and the one we received later one we can check that the content of our files are different and the integrity has been lost at our file.

CHAPTER 4

Indices and tables

- genindex
- search