IMD0293

APRESENTAÇÃO

QUEM SOU EU?

Danilo Curvelo

danilocurvelo@imd.ufrn.br

UFRN/IMD/CIVT/**A216**

Entusiasta da tecnologia **Blockchain**

Carteiras:

Bitcoin 1AKE1bNd18tDVsvQGN4XxTMbA5DZQrpHst

Ethereum 0x75583f36A9F718477F6fa7AB088049c5a2CfbAeA

Quem são vocês?

INFO

IMD0293

Tópicos Especiais em Internet das Coisas D

Fundamentos de Blockchain

60h (35T56)

TECH TRENDS

CompTIA

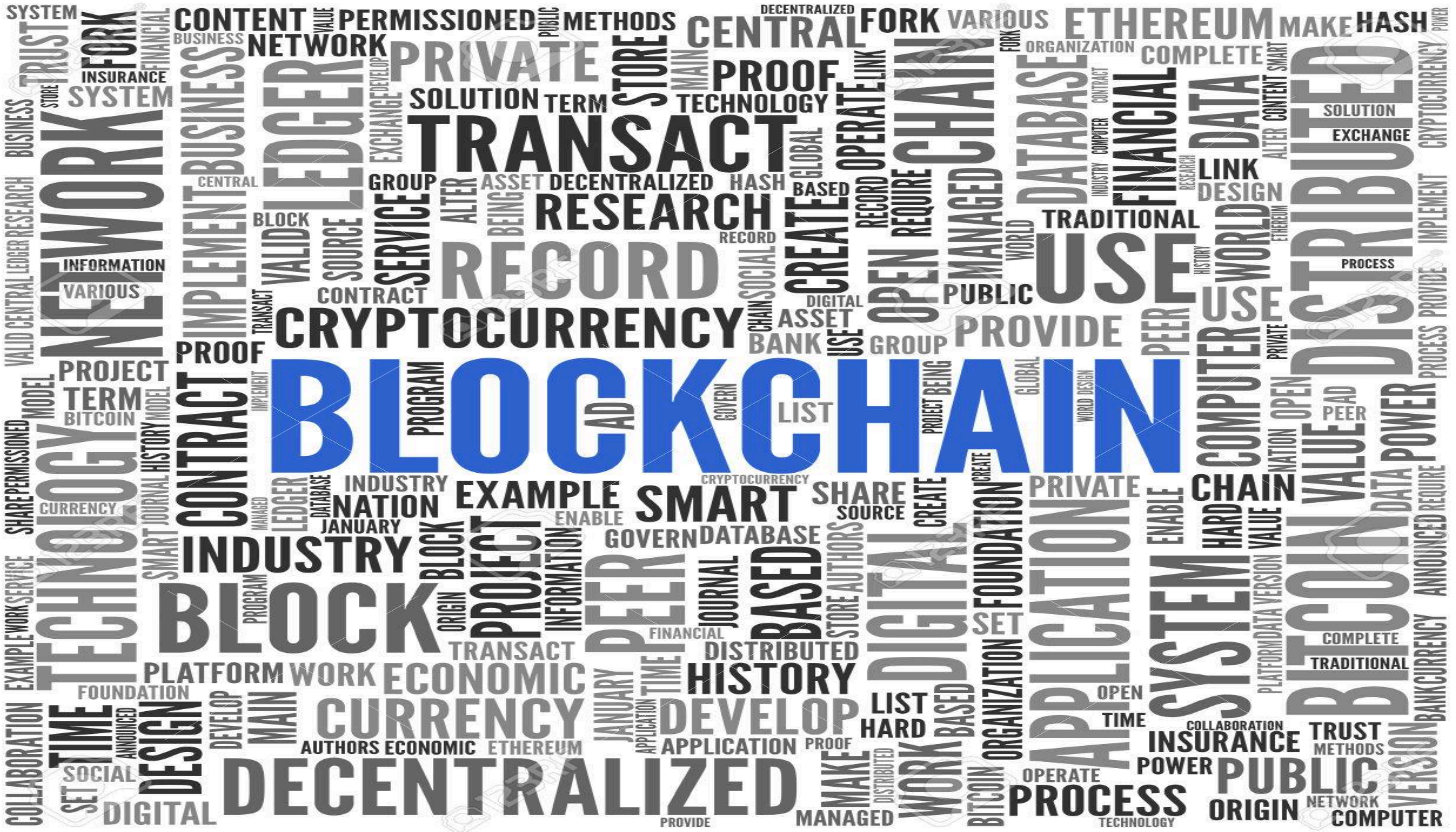
1. IoT
2. AI
3. 5G
4. Serverless Computing
5. **Blockchain**
6. Robotics
7. Biometrics
8. 3D Printing
9. VR/AR
10. Drones

Gartner

1. Autonomous Things
2. Augmented Analytics
3. AI
4. Digital Twins
5. Edge Computing
6. Immersive Technologies
7. **Blockchain**
8. Smart Spaces
9. Digital Ethics
10. Quantum Computing

Forbes

1. Increased Automation
2. **Blockchain**
3. Human/AI Collab
4. IoT
5. VR/AR
6. Cybersecurity with ML/AI
7. Solutions to Tech Backslash
8. Technology Convergence


BLOCKCHAIN
CRYPTOCURRENCY
TRANSACT
RECORD
DECENTRALIZED
BLOCK
NETWORK
TECHNOLOGY
DISTRIBUTED
BITCOIN
APPLICATION
DIGITAL
LEDGER
USE
SYSTEM
PEER
INDUSTRY
CONTRACT
EXAMPLE
SMART
HISTORY
CURRENCY
DEVELOP
ECONOMIC
PROCESS
PUBLIC
PRIVATE
CENTRAL
ETHEREUM
RESEARCH
DATABASE
FINANCIAL
PROVIDE
PROOF
BUSINESS
IMPLEMENT
CHAIN
COMPUTER
VALUE
POWER
INSURANCE
TRUST
ORIGIN
DESIGN
TIME
MAKE
WORK
HASH
FORK
CONTENT
PERMISSIONED
METHODS
SOLUTION
TECHNOLOGY
PLATFORM
FOUNDATION
ORGANIZATION
MANAGED
DATA
WORLD
TRADITIONAL
LINK
COMPLETE
EXCHANGE

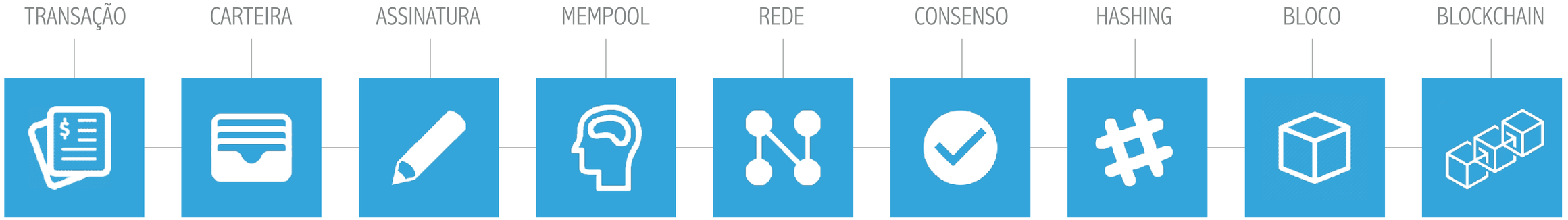
O QUE VAMOS APRENDER?

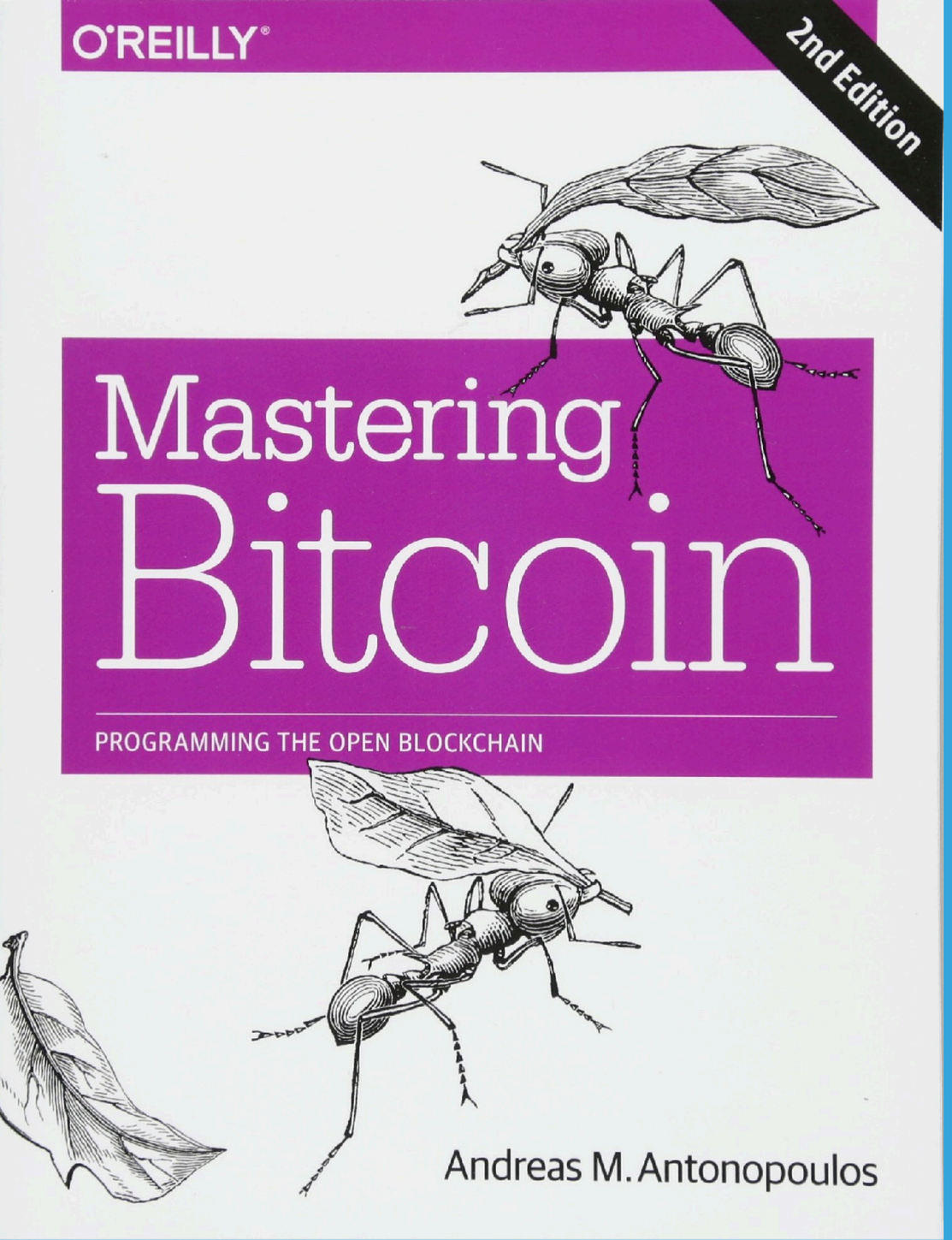
- Tecnologia Blockchain
- Fundamentos criptográficos
- Consenso distribuído
- Smart Contracts
- DApps
- Estudos de caso:
  - Bitcoin
  - Ethereum

O QUE NÃO VAMOS APRENDER?

- Economia;
- Investimentos;
- Mercado financeiro;
- Como ficar rico com criptomoedas.

ARQUITETURA DE UM **BLOCKCHAIN**

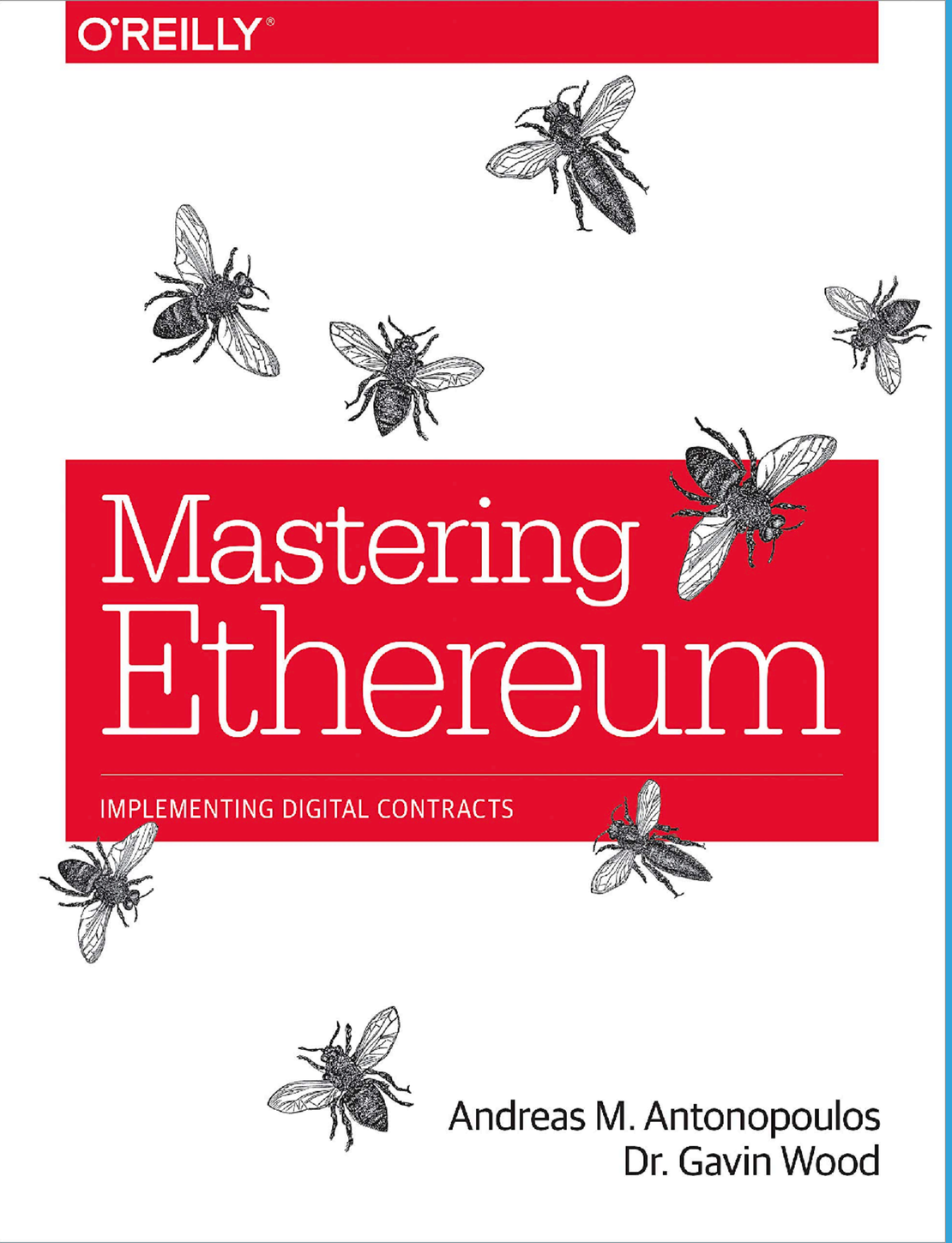




MASTERING BITCOIN

Andreas Antonopoulos

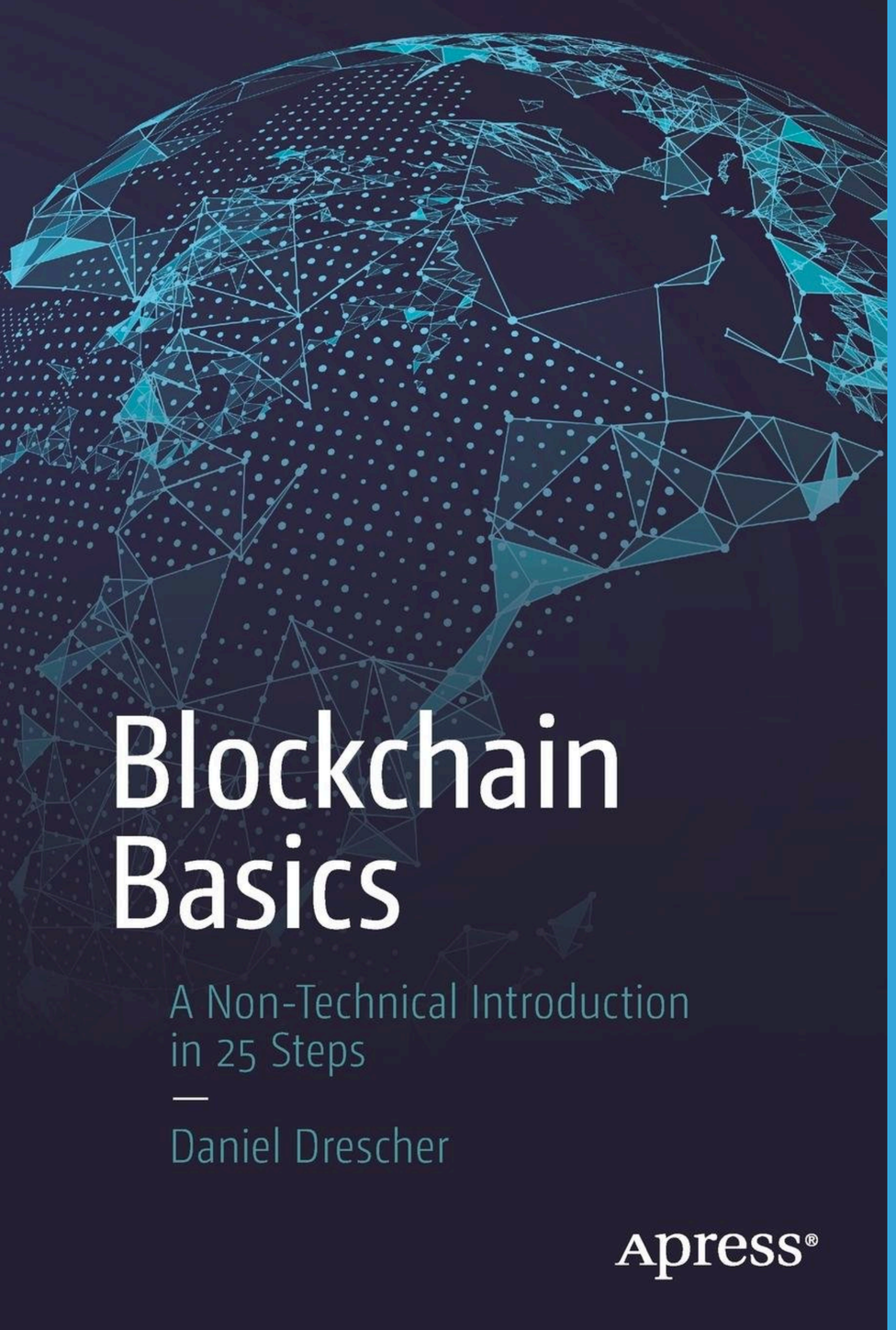
https://github.com/bitcoinbook/bitcoinbook



MASTERING ETHEREUM

Andreas Antonopoulos
Gavin Wood

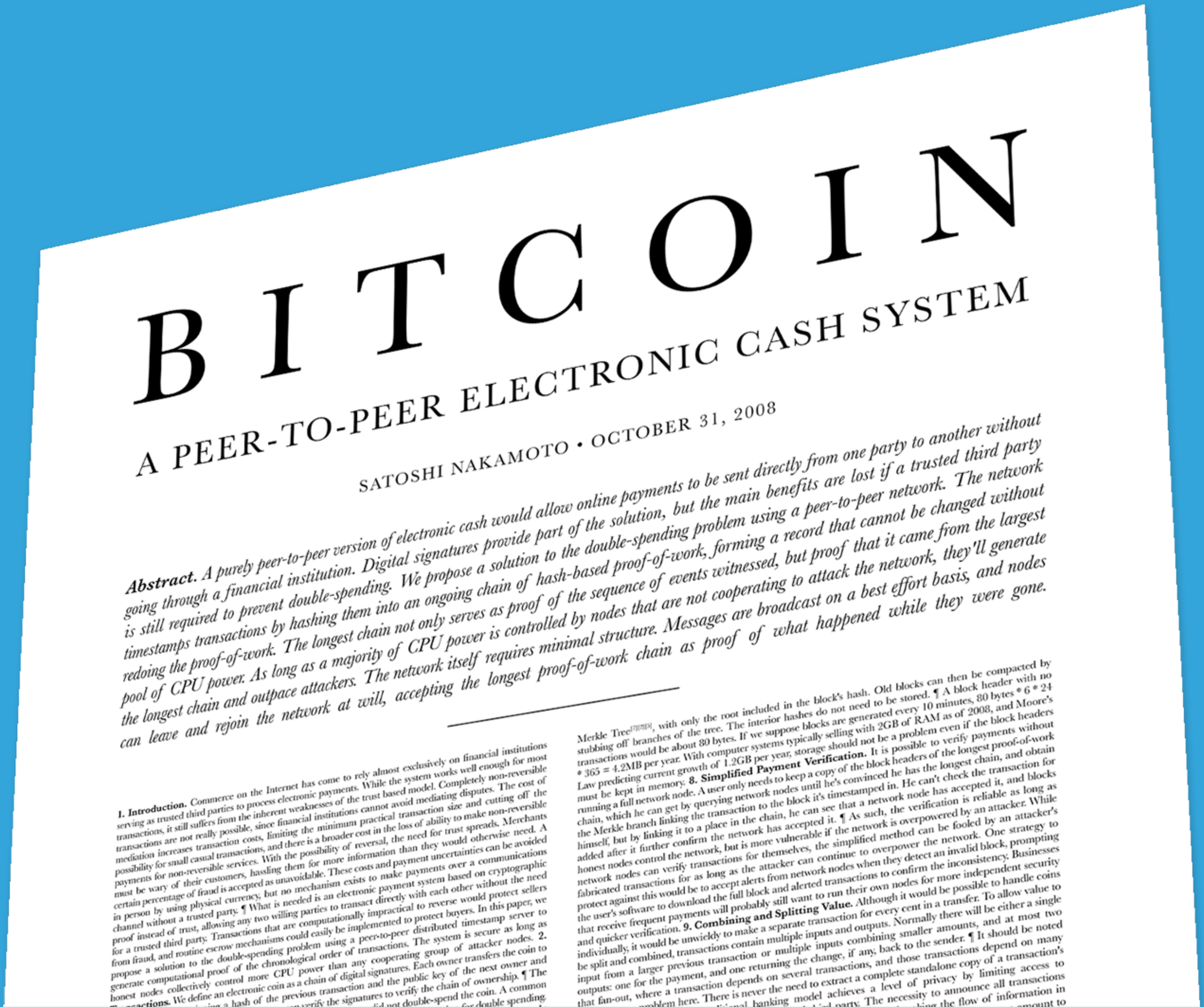
https://github.com/ethereumbook/ethereumbook



BLOCKCHAIN BASICS

Daniel Drescher

https://bitcoin.org/en/bitcoin-paper



BITCOIN

A PEER-TO-PEER ELECTRONIC CASH SYSTEM

SATOSHI NAKAMOTO • OCTOBER 31, 2008

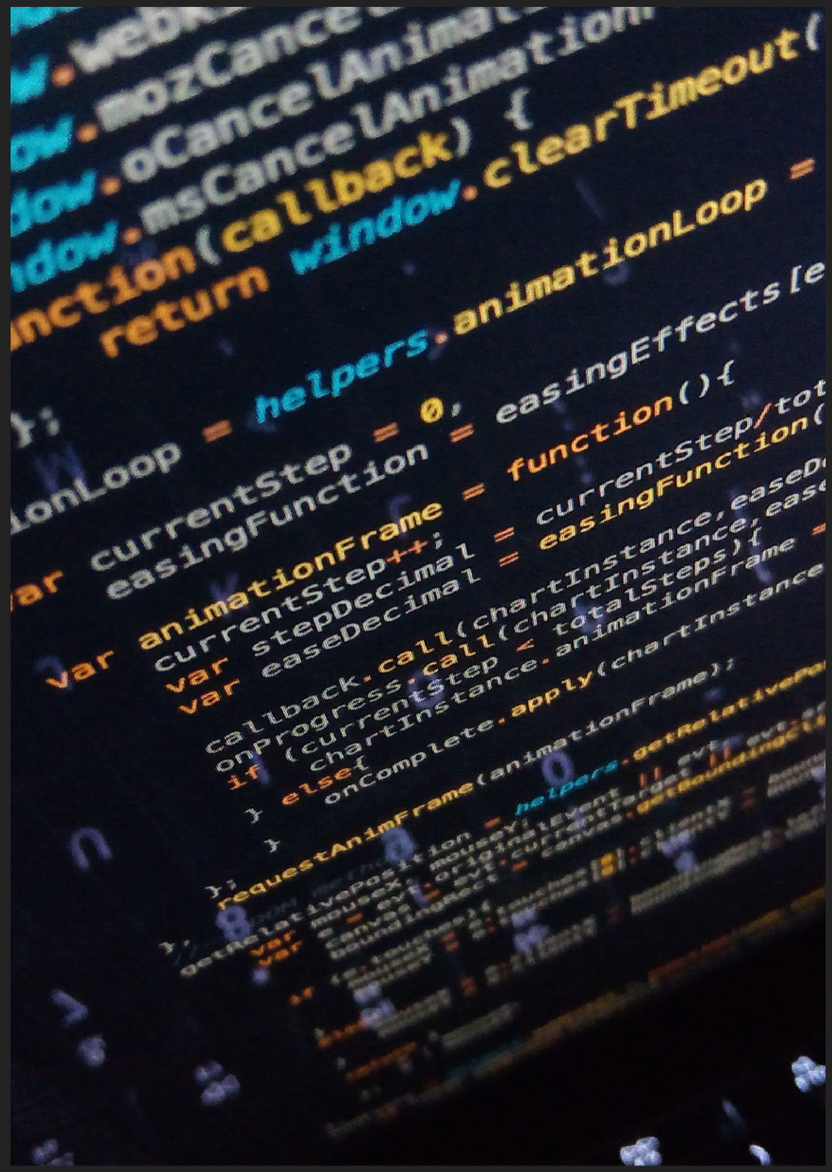
Abstract. *A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.*

1. Introduction. Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes. The cost of mediation increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions, and there is a broader cost in the loss of ability to make non-reversible payments for non-reversible services. With the possibility of reversal, the need for trust spreads. Merchants must be wary of their customers, hassling them for more information than they would otherwise need. A certain percentage of fraud is accepted as unavoidable. These costs and payment uncertainties can be avoided in person by using physical currency, but no mechanism exists to make payments over a communications channel without a trusted party. ¶ What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers. In this paper, we propose a solution to the double-spending problem using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes. **2. Transactions.** We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership. ¶ The

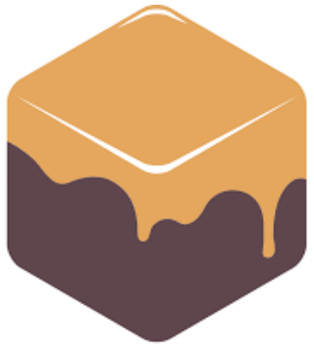
Merkle Tree[7][2][5], with only the root included in the block's hash. Old blocks can then be compacted by stubbing off branches of the tree. The interior hashes do not need to be stored. ¶ A block header with no transactions would be about 80 bytes. If we suppose blocks are generated every 10 minutes, 80 bytes * 6 * 24 * 365 = 4.2MB per year. With computer systems typically selling with 2GB of RAM as of 2008, and Moore's Law predicting current growth of 1.2GB per year, storage should not be a problem even if the block headers must be kept in memory. **8. Simplified Payment Verification.** It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction to the block it's timestamped in. He can't check the transaction for himself, but by linking it to a place in the chain, he can see that a network node has accepted it, and blocks added after it further confirm the network has accepted it. ¶ As such, the verification is reliable as long as honest nodes control the network, but is more vulnerable if the network is overpowered by an attacker. While network nodes can verify transactions for themselves, the simplified method can be fooled by an attacker's fabricated transactions for as long as the attacker can continue to overpower the network. One strategy to protect against this would be to accept alerts from network nodes when they detect an invalid block, prompting the user's software to download the full block and alerted transactions to confirm the inconsistency. Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification. **9. Combining and Splitting Value.** Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender. ¶ It should be noted that fan-out, where a transaction depends on several transactions, and those transactions depend on many

REQUISITOS

- Lógica de Programação (Python 3+)
- HTML+CSS+JS
- REST-APIs



TECNOLOGIAS



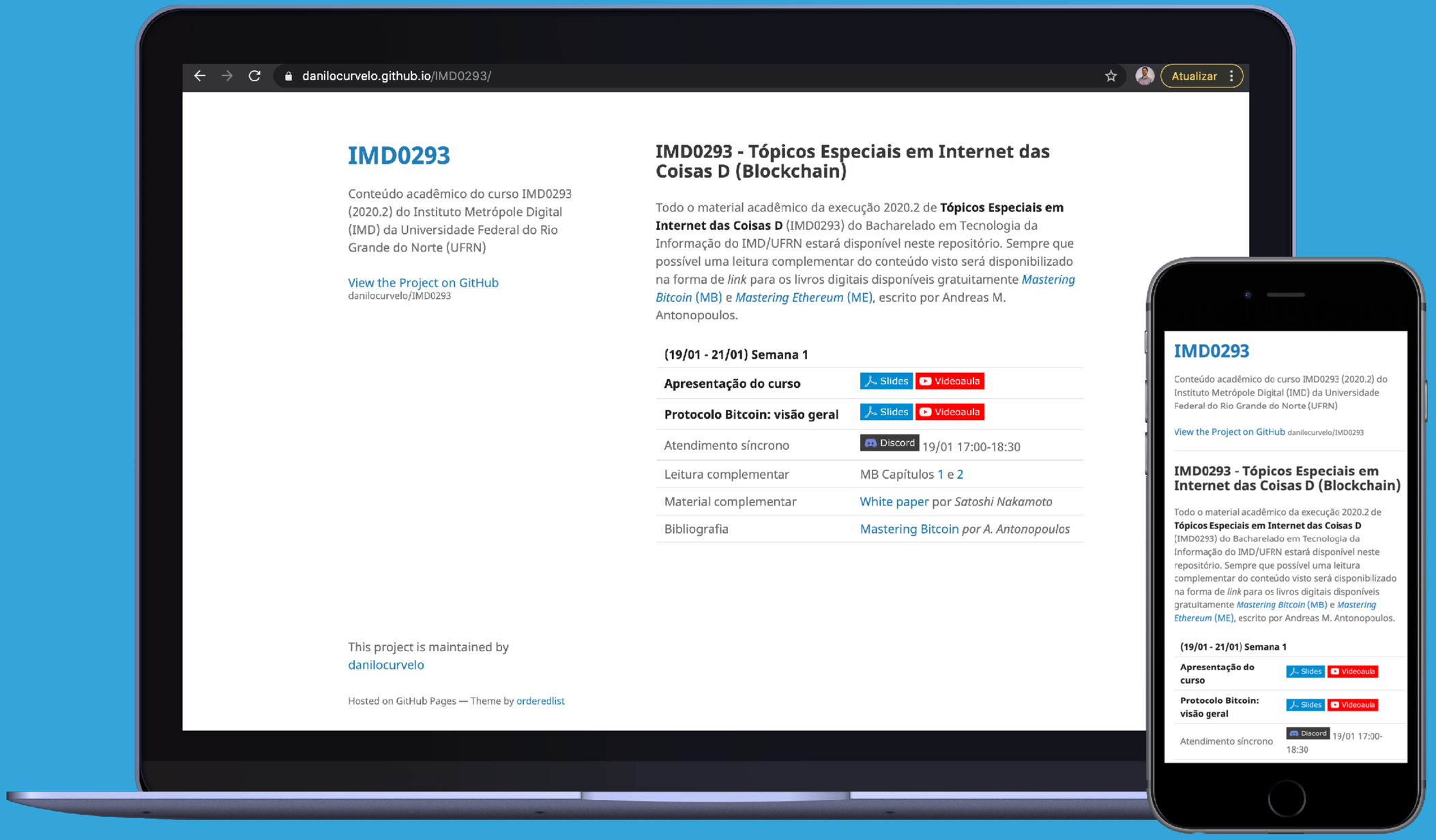
AVALIAÇÃO

Atividades práticas

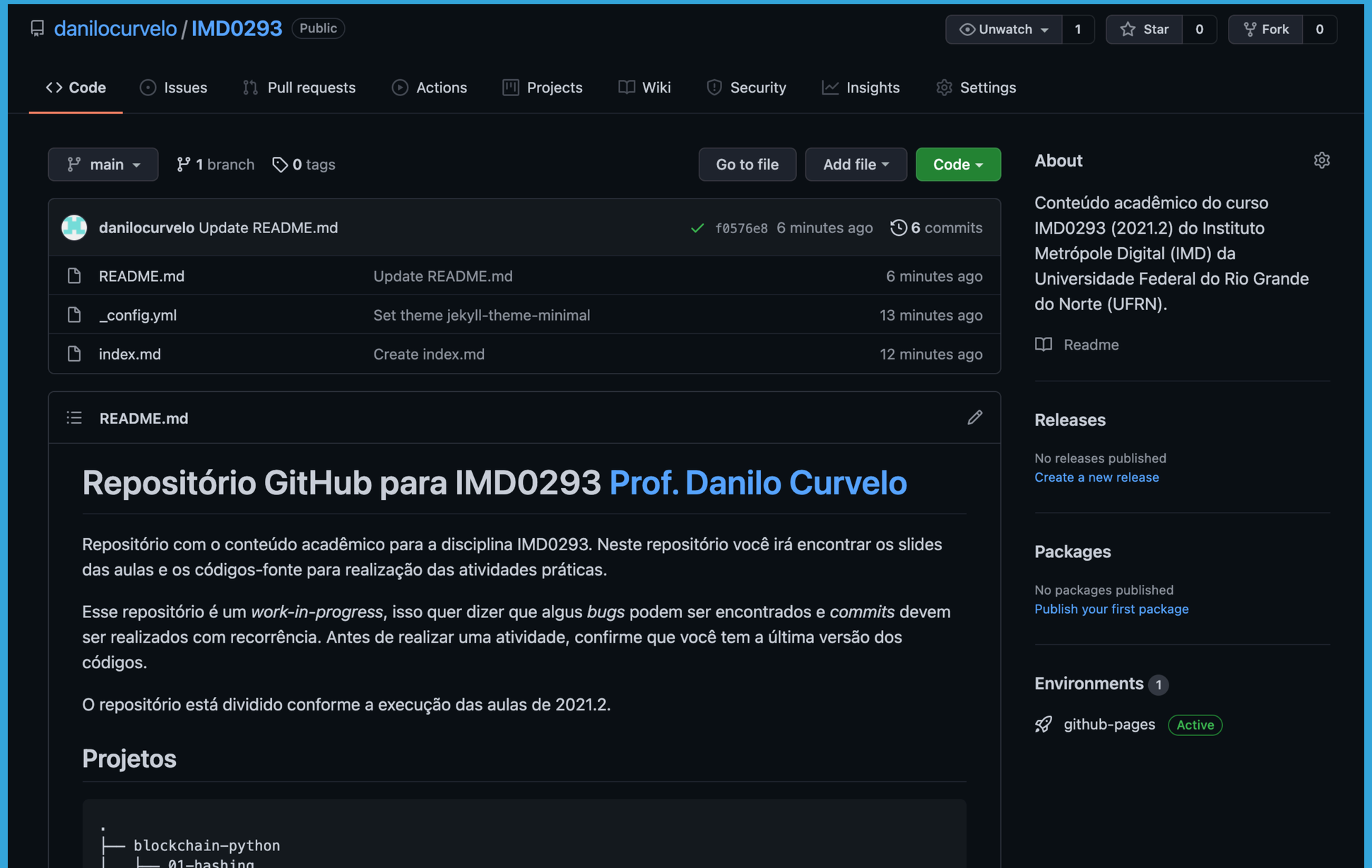
~~Prova(s) teórica(s)~~

Projetos

danilocurvelo.github.io/IMD0293



github.com/danilocurvelo/imd0293



GitHub Classroom

Classroom

GitHub Education

Classrooms / imd0293-2021.2

imd0293-2021.2

imd0293

Assignments 0 | Students 0 | TAs and Admins 1 | Settings

Assignments

Create an assignment to get started.

Create an individual assignment to generate an assignment repository for each student to work from. Or, create a group assignment and have students work collaboratively in groups from team repositories.

Create an assignment

Learn more about individual and group assignments.

Need to teach Git & GitHub fundamentals?

The Classroom team has created an assignment for you to use to teach your students the fundamentals of Git & GitHub.

Use starter assignment

Learn more

Assignments

New assignment

The Classroom team has created an assignment for you to use to teach your students the fundamentals of Git & GitHub. Click here to try.

01-hashing
Individual assignment
Invite link

02-blocks
Individual assignment
Invite link

03-pow
Individual assignment
Invite link

05-transactions
Individual assignment
Invite link

06-consensus
Group assignment for **Individual ou em dupla**
Invite link

imd0293 / 01-hashing Private template

Watch 0 Star 0 Fork 0

Code Issues Pull requests Actions Projects Security Insights Settings

main 1 branch 0 tags Go to file Add file Code Use this template

danilocurvelo Update README.md dd7c9a1 on 28 Jan 7 commits

README.md	Update README.md	9 months ago
blockchain.py	Add files via upload	9 months ago

README.md

ALUNO: <EDITE AQUI COM O SEU NOME COMPLETO!>

Atividade: Hashing (01-hashing)

Esta atividade tem como objetivo implementar o primeiro método no desenvolvimento do nosso **blockchain**. Este método estático será amplamente utilizado em várias etapas do processo, uma vez que *hashing* é uma das técnicas essenciais para o funcionamento deste modelo de blockchain.

Metodologia e Avaliação

O desenvolvimento das atividades avaliativas deve ser realizada individualmente, em computador pessoal ou em computador do laboratório, com livre consulta a recursos na internet (*consulta != cópia*) e discussão entre colegas. Utilize a IDE de sua preferência (sugestão: Visual Studio Code).

As atividades são cumulativas, de forma que ao final teremos um blockchain funcional usando as técnicas e os conceitos téoricos vistos em sala de aula.

Instruções de submissão

About

Esta atividade tem como objetivo implementar o primeiro método no desenvolvimento do nosso **blockchain**. Este método estático será amplamente utilizado em várias etapas do processo, uma vez que *hashing* é uma das técnicas essenciais para o funcionamento deste modelo de blockchain.

Readme

Releases

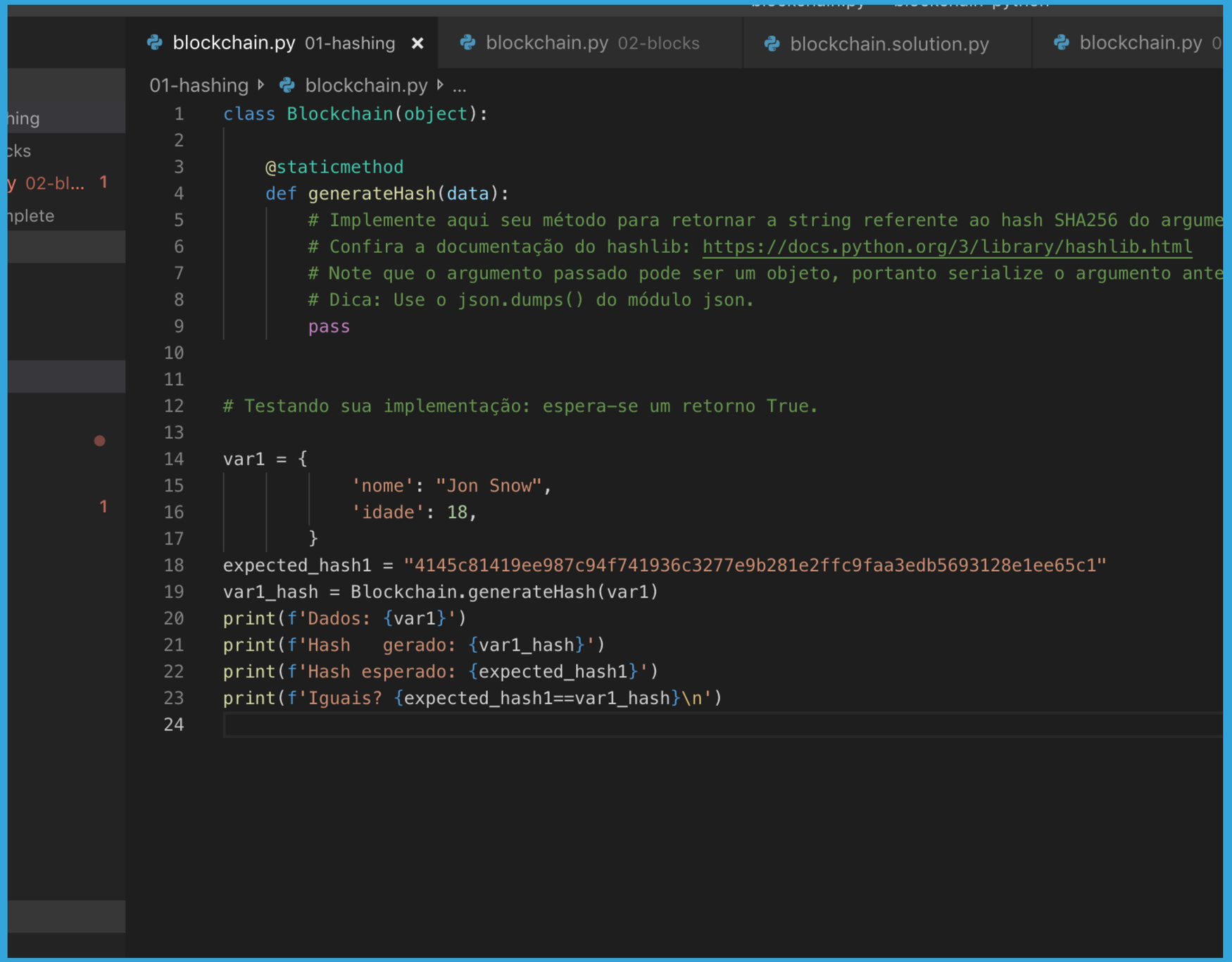
No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

Python 100.0%


blockchain.py 01-hashing
blockchain.py 02-blocks
blockchain.solution.py
01-hashing ▸ blockchain.py ▸ ...
class Blockchain(object):
@staticmethod
def generateHash(data):
# Implemente aqui seu método para retornar a string referente ao hash SHA256 do argume
# Confira a documentação do hashlib: https://docs.python.org/3/library/hashlib.html
# Note que o argumento passado pode ser um objeto, portanto serialize o argumento ante
# Dica: Use o json.dumps() do módulo json.
pass
# Testando sua implementação: espera-se um retorno True.
var1 = {
'nome': "Jon Snow",
'idade': 18,
}
expected_hash1 = "4145c81419ee987c94f741936c3277e9b281e2ffc9faa3edb5693128e1ee65c1"
var1_hash = Blockchain.generateHash(var1)
print(f'Dados: {var1}')
print(f'Hash gerado: {var1_hash}')
print(f'Hash esperado: {expected_hash1}')
print(f'Iguais? {expected_hash1==var1_hash}\n')

Atendimento **síncrono** toda quinta-feira das 17:00-18:30

**presença não é obrigatória*

https://discord.gg/w8zPgquWxg