IMD0913
APRESENTAÇÃO

Quem sou eu?

Danilo Curvelo

`danilocurvelo@imd.ufrn.br`

UFRN/IMD/CIVT/**A216**

Entusiasta da tecnologia Blockchain

Carteiras:

Bitcoin bc1q8z8kkf24emy9577r7zmcyjc8t0ztk9xx49dsn5

Ethereum 0x6bE6ec844B5911476b03E5399147cA552702a471

Quem são vocês?

Sobre o curso...

IMD0913

Blockchain e Aplicações Descentralizadas

60h - **35T56**

CIVT/IMD **A101**

https://danilocurvelo.github.io/imd0913-2023/

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


DREX
by Banco Central do Brasil

*“Enquanto a nota de dinheiro está no papel,
o Drex vai estar em **blockchain**.”*

https://github.com/bacen/pilotord-kit-onboarding

BNDES

TRIBUNAL DE CONTAS DA UNIÃO

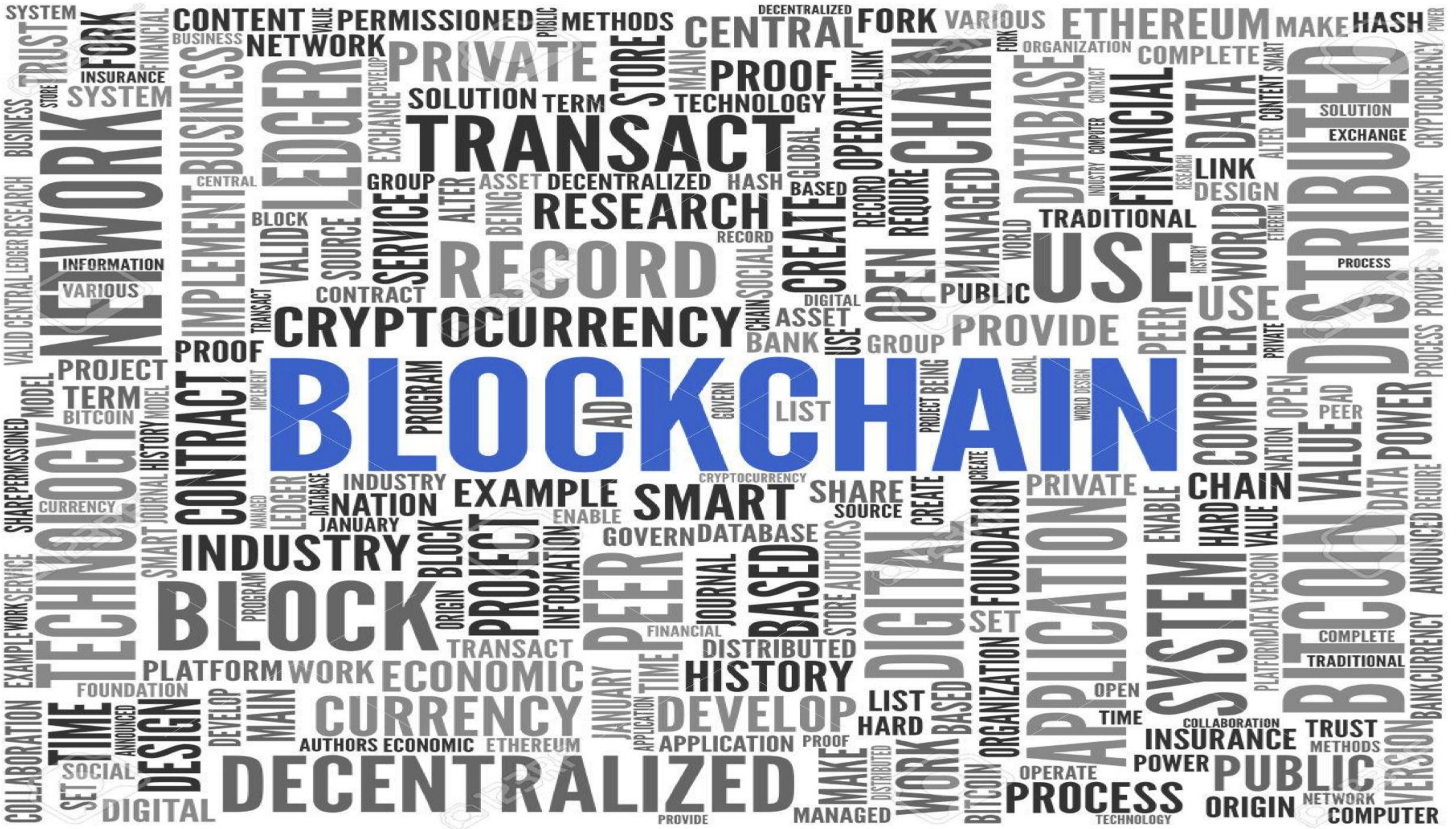
Classificação: Documento Ostensivo
Unidade Gestora: ATI/DESIS1 e AGOV/DEREG

ACORDO DE COOPERAÇÃO Nº D-121.2.0014.22, QUE ENTRE SI CELEBRAM O BANCO NACIONAL DE DESENVOLVIMENTO ECONÔMICO E SOCIAL (BNDES) E O TRIBUNAL DE CONTAS DA UNIÃO (TCU), PARA COOPERAÇÃO COM VISTAS À FORMAÇÃO DA REDE BLOCKCHAIN BRASIL (RBB). (PROCESSO NO TCU: TC 039.840/2021-2)

O **BANCO NACIONAL DE DESENVOLVIMENTO ECONÔMICO SOCIAL (BNDES)**, empresa pública federal, regida pela Lei nº 5.662, de 21 de junho de 1971, com a denominação dada pelo Decreto-Lei nº 1.940, de 25 de maio de 1982, com sede em Brasília, Distrito Federal, e serviços no Rio de Janeiro, Estado do Rio de Janeiro, na Avenida República do Chile, nº 100, Centro, inscrito no CNPJ sob o nº 33.657.248/0001-89, doravante denominado simplesmente **BNDES**, neste ato representado nos termos de seu Estatuto Social; e

O **TRIBUNAL DE CONTAS DA UNIÃO**, com sede em Brasília, Distrito Federal, no Setor de Administração Federal Sul, Quadra 4, Lote 1, inscrito no CNPJ sob o nº 00.414.607/0001-18, doravante denominado simplesmente **TCU**, neste ato representado pela

https://github.com/RBBNet/rbb


BLOCKCHAIN
TRANSACT
RESEARCH
RECORD
CRYPTOCURRENCY
PRIVATE
CENTRAL
ETHEREUM
FORK
HASH
CONTENT
PERMISSIONED
METHODS
NETWORK
PROOF
SOLUTION
TECHNOLOGY
LEDGER
BUSINESS
CHAIN
DATABASE
FINANCIAL
DATA
DISTRIBUTED
USE
PUBLIC
PROVIDE
TRADITIONAL
LINK
DESIGN
CREATE
OPEN
MANAGED
WORLD
DECENTRALIZED
GROUP
ASSET
BANK
CONTRACT
INFORMATION
VARIOUS
SYSTEM
INSURANCE
TRUST
PROJECT
TERM
BITCOIN
TECHNOLOGY
IMPLEMENT
INDUSTRY
BLOCK
EXAMPLE
SMART
NATION
JANUARY
GOVERN
DATABASE
SHARE
SOURCE
PEER
BASED
DIGITAL
APPLICATION
FOUNDATION
ORGANIZATION
COMPUTER
VALUE
POWER
HARD
PLATFORM
WORK
ECONOMIC
TRANSACT
CURRENCY
HISTORY
DISTRIBUTED
DEVELOP
LIST
TIME
DESIGN
MAIN
AUTHORS
SOCIAL
COLLABORATION
MAKE
PROCESS
OPERATE
ORIGIN
COMPLETE
VERSION
COMPUTER
MANAGED

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;

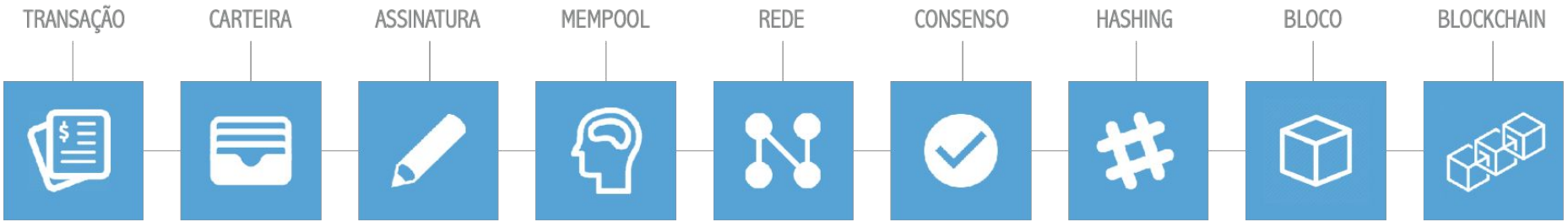
Como ficar **rico** com criptomoedas.


INTRODUÇÃO A
TECNOLOGIA
BLOCKCHAIN


ARQUITETURA DE
UM BLOCKCHAIN
bitcoin

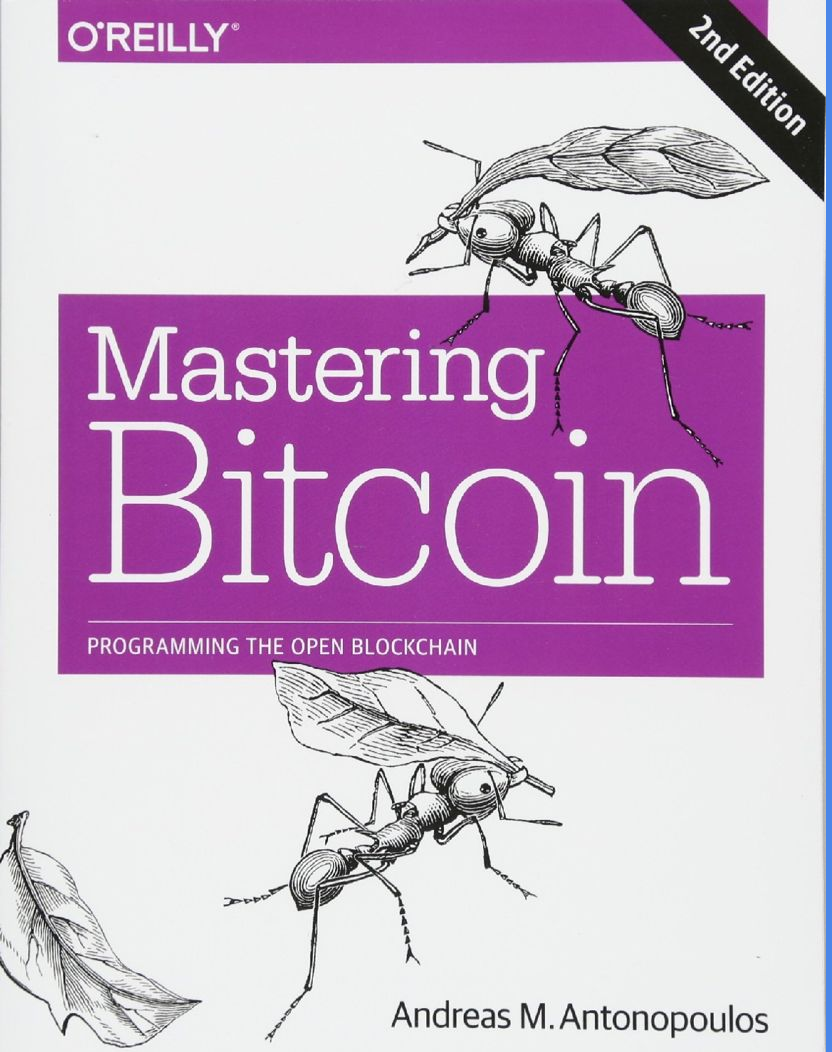

CONTRATOS INTELIGENTES
E APLICAÇÕES
DESCENTRALIZADAS
ethereum

ARQUITETURA DE UM **BLOCKCHAIN**




ethereum
BLOCKCHAIN APP PLATFORM

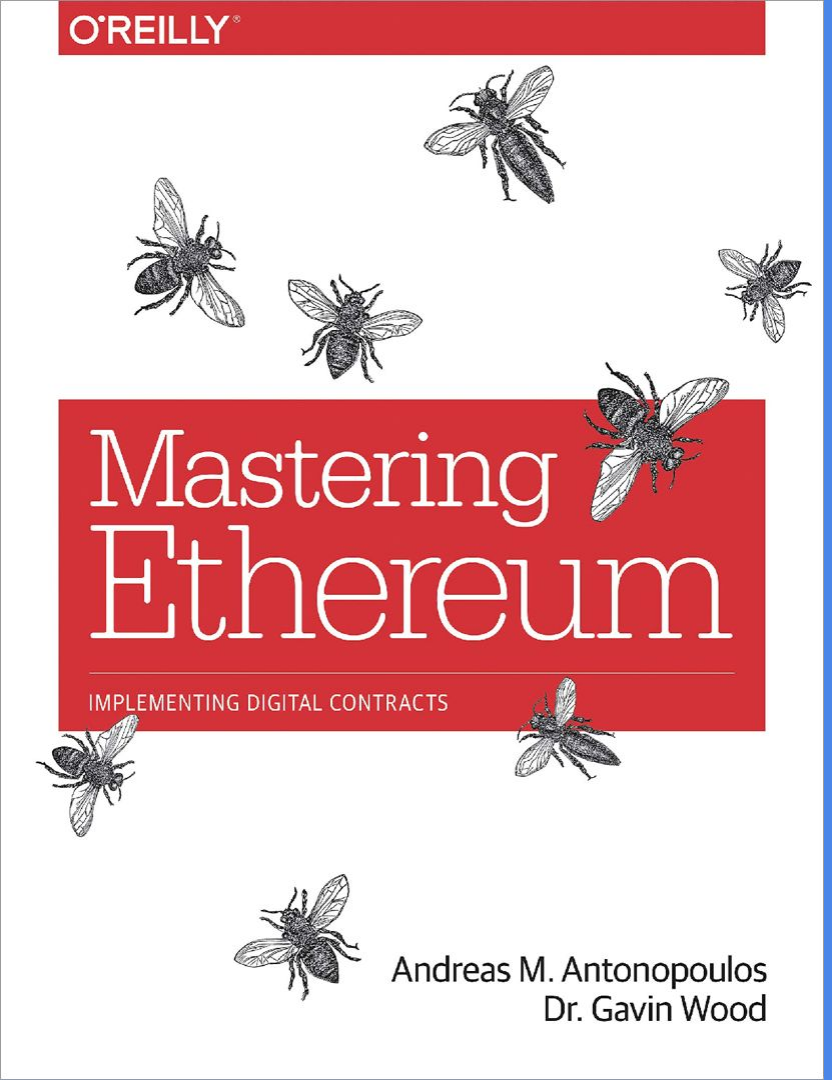

remix



MASTERING BITCOIN

Andreas Antonopoulos

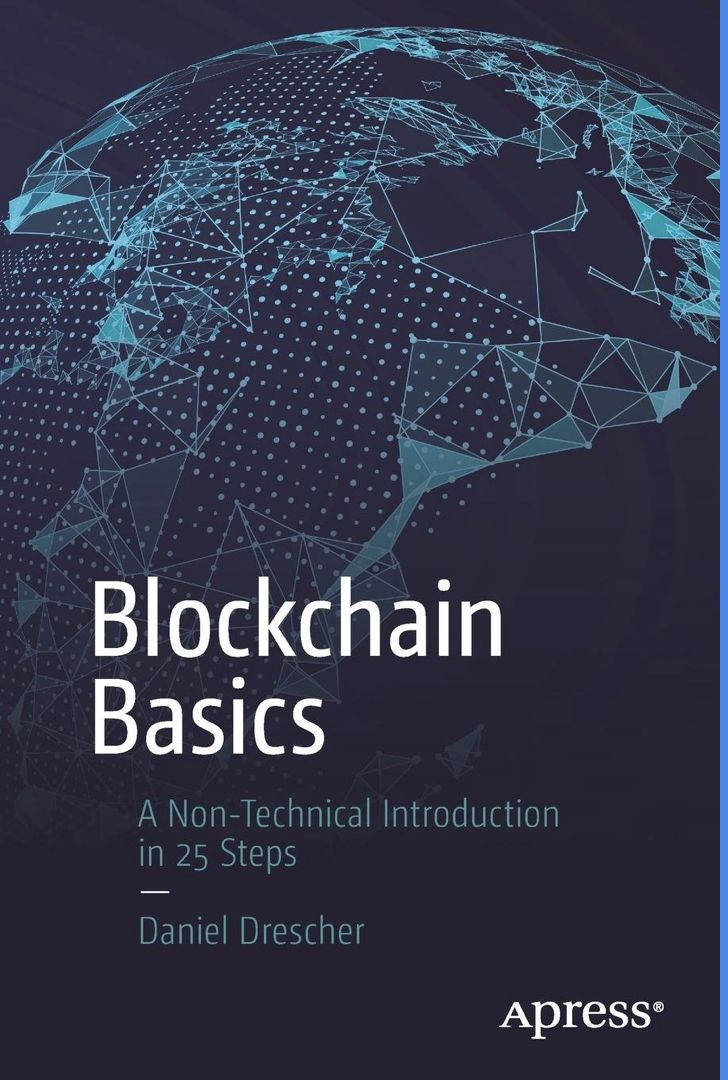
https://github.com/bitcoinbook/bitcoinbook



MASTERING ETHEREUM

Andreas Antonopoulos
Gavin Wood

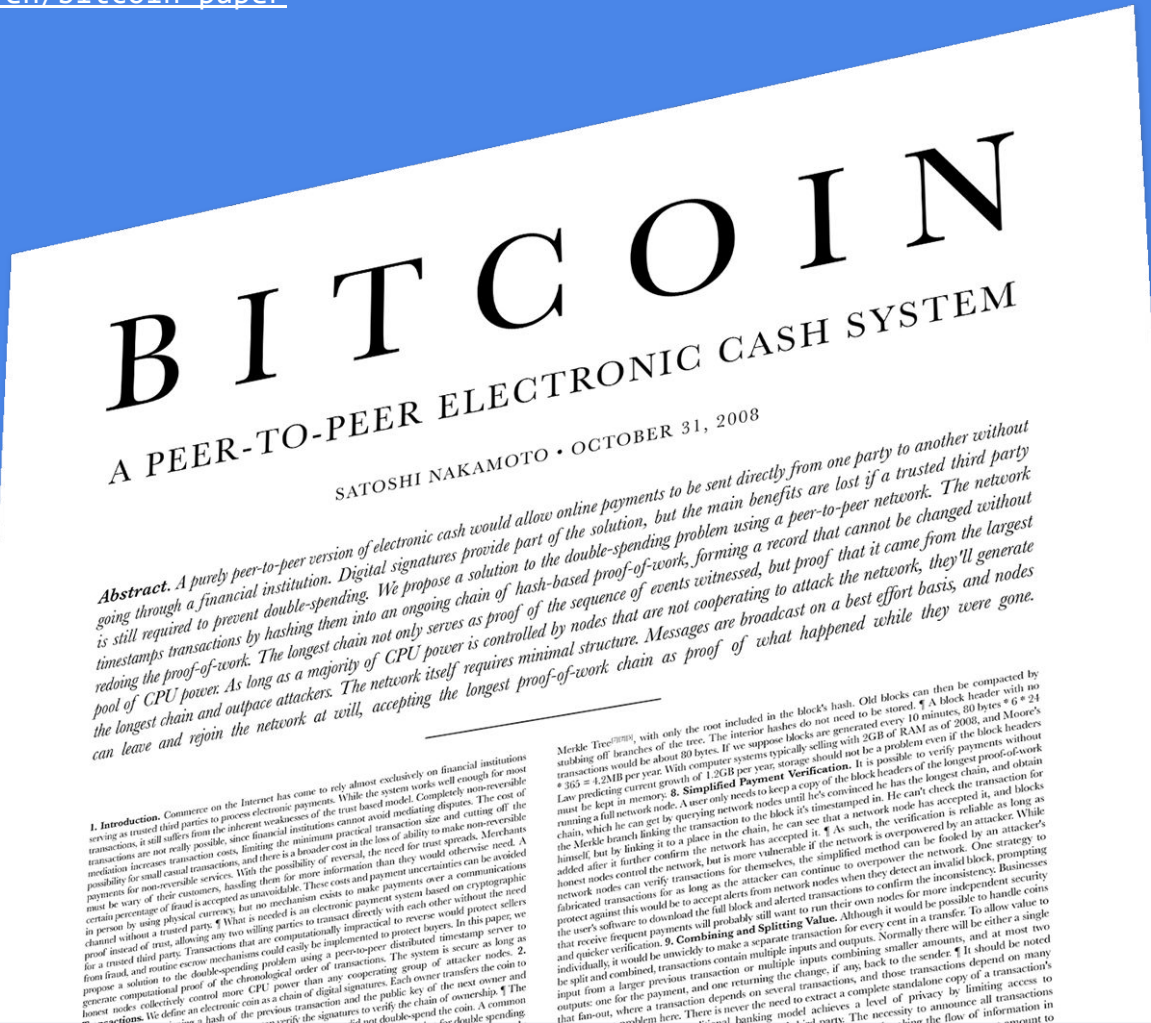
https://github.com/ethereumbook/ethereumbook



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 problem here. There is never the need to extract a complete standalone copy of a transaction's

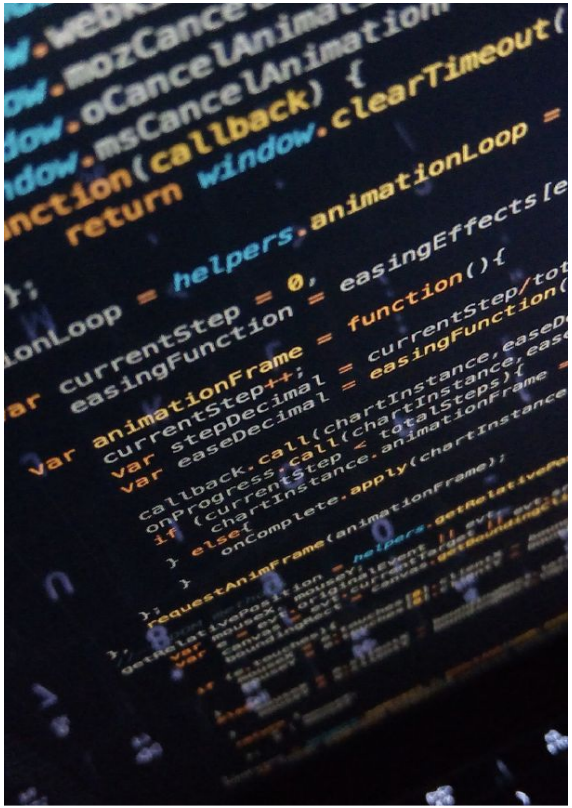
Merkle Tree[7][8][9], with only the root included in the block's hash. Old blocks can then be compacted by stubbing off branches of the tree. The interior hashes do not need to be stored. ¶ A block header with no transactions would be about 80 bytes. If we suppose blocks are generated every 10 minutes, 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

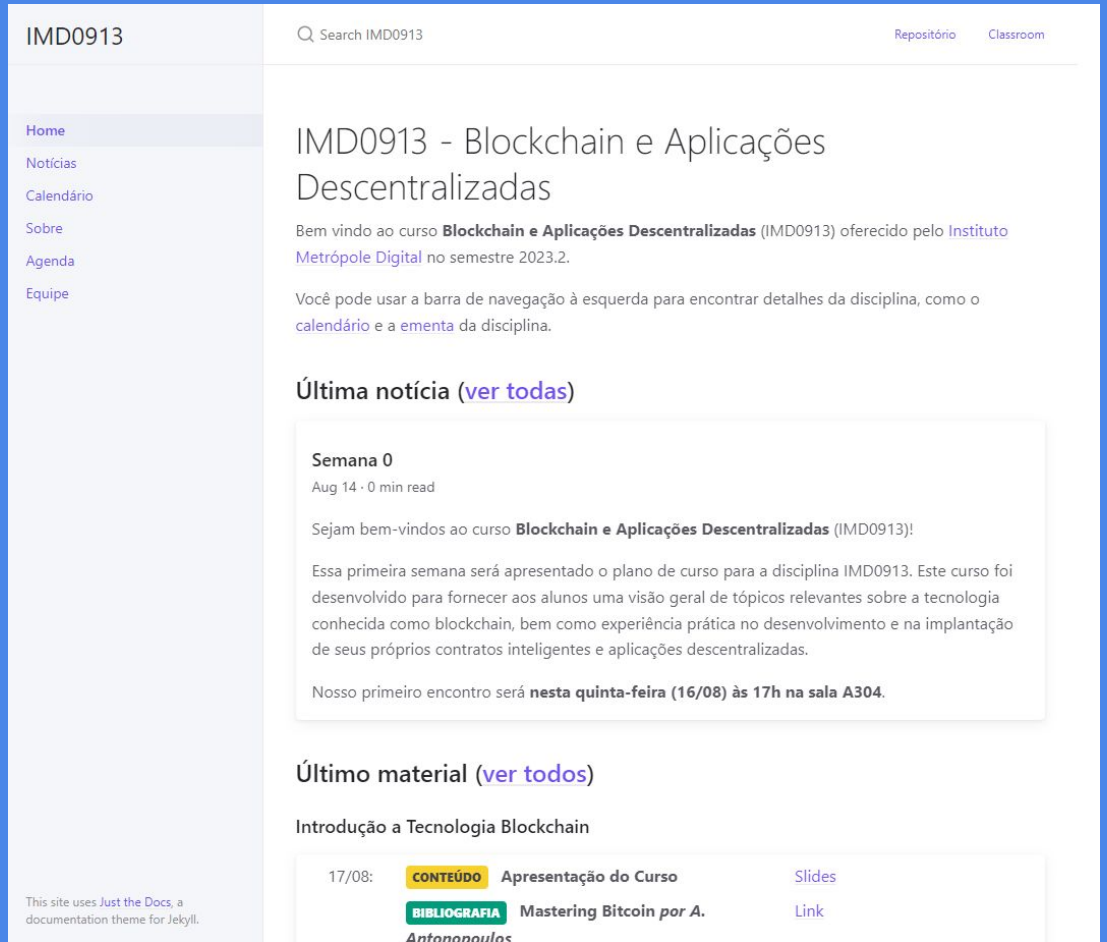


Avaliação

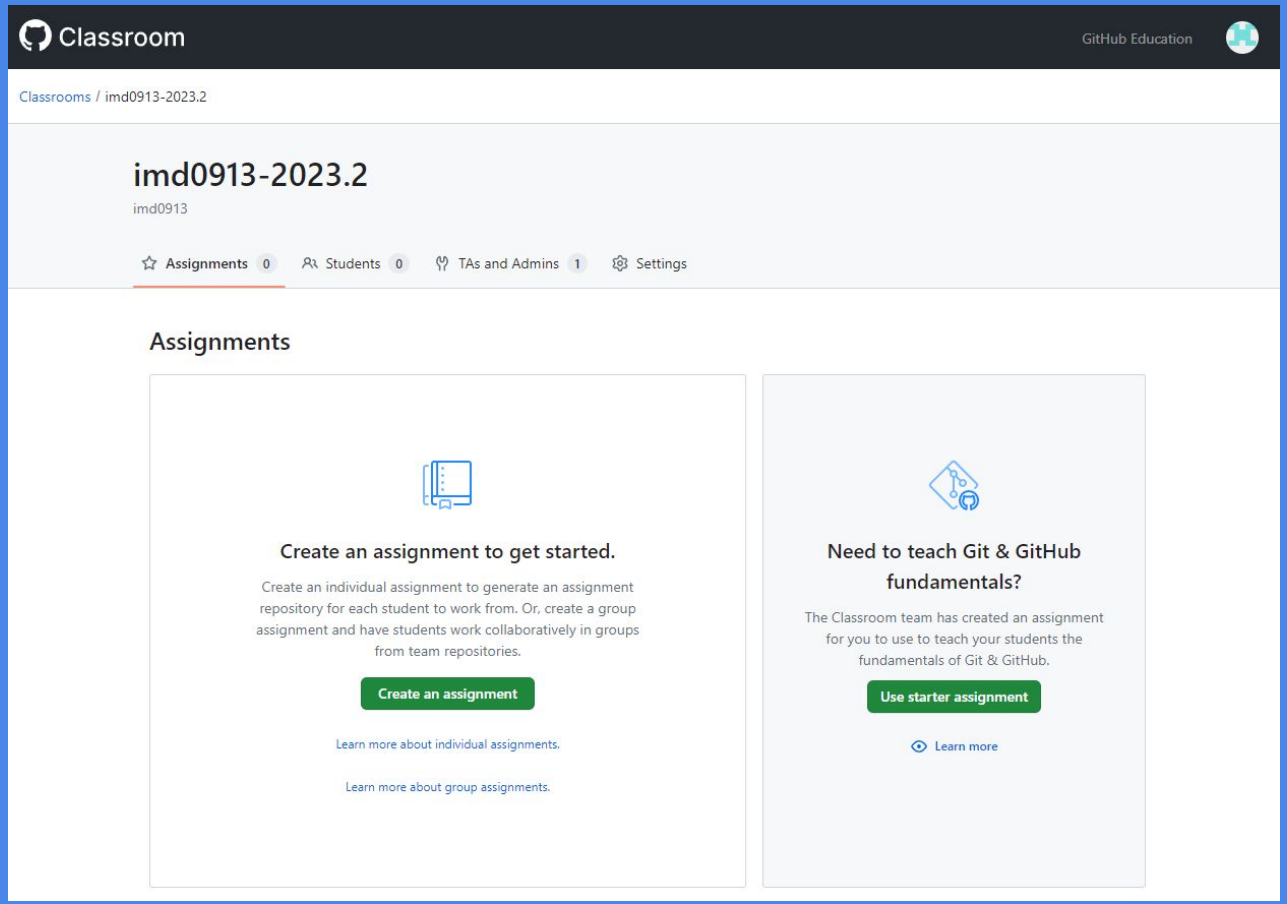
Atividades práticas

Projeto final

https://danilocurvelo.github.io/imd0913-2023/




GitHub
Classroom


Classroom
GitHub Education
Classrooms / imd0913-2023.2
imd0913-2023.2
imd0913
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 assignments.
Learn more about 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

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	
07-smart-contracts Group assignment for **Smart Contracts**	Invite link	

01-hashing

Individual assignment - Deadline Passed Active VS Code https://classroom.gi1 Edit Download

Rostered students	63
Added students	0
Accepted students	52
Assignment submissions	52
Passing students	47/52

Search by GitHub username or student identifier

Classroom roster	Unlinked accounts	Accepted	Submitted	Passing	Sort
ABRAAO VITOR LOPES DANTAS @abraaod	Latest commit passed	10/10	1 commit	Submitted	
ALEF EMANNUEL TRIGUEIRO DIAS @AlefEmannuel	Latest commit passed	10/10	1 commit	Submitted	
ALEX BARBOSA FELIX DA SILVA @alexbarbosafs	Latest commit passed	10/10	1 commit	Submitted	
ALEXANDRE ALVES ANDRADE @alexandreand	Latest commit passed	10/10	1 commit	Submitted	
DORGIVAL DA ROCHA FILHO @Dojak220	Latest commit failed	0/10	2 commits	Submitted	
ENZO LOPES D'ANJOUR DE SOUZA @enzodanjour	Latest commit passed	10/10	2 commits	Submitted	

imd0913 / 01-hashing Private template

Edit Pins | Watch 0 | Fork 0 | Star 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 test_github_classroom.py b535d09 on 1 Nov 2021 17 commits

README.md	Update README.md	10 months ago
blockchain.py	Update blockchain.py	10 months ago
test_github_classroom.py	Update test_github_classroom.py	10 months ago

README.md

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

Submissão deve ser feita a partir do GitHub Classroom até às 23:59 do dia 07/11/2021. Basta realizar o *commit* do seu arquivo `blockchain.py` no repositório privado criado para você a partir do link disponibilizado (associe sua

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
0 stars
0 watching
0 forks

Releases

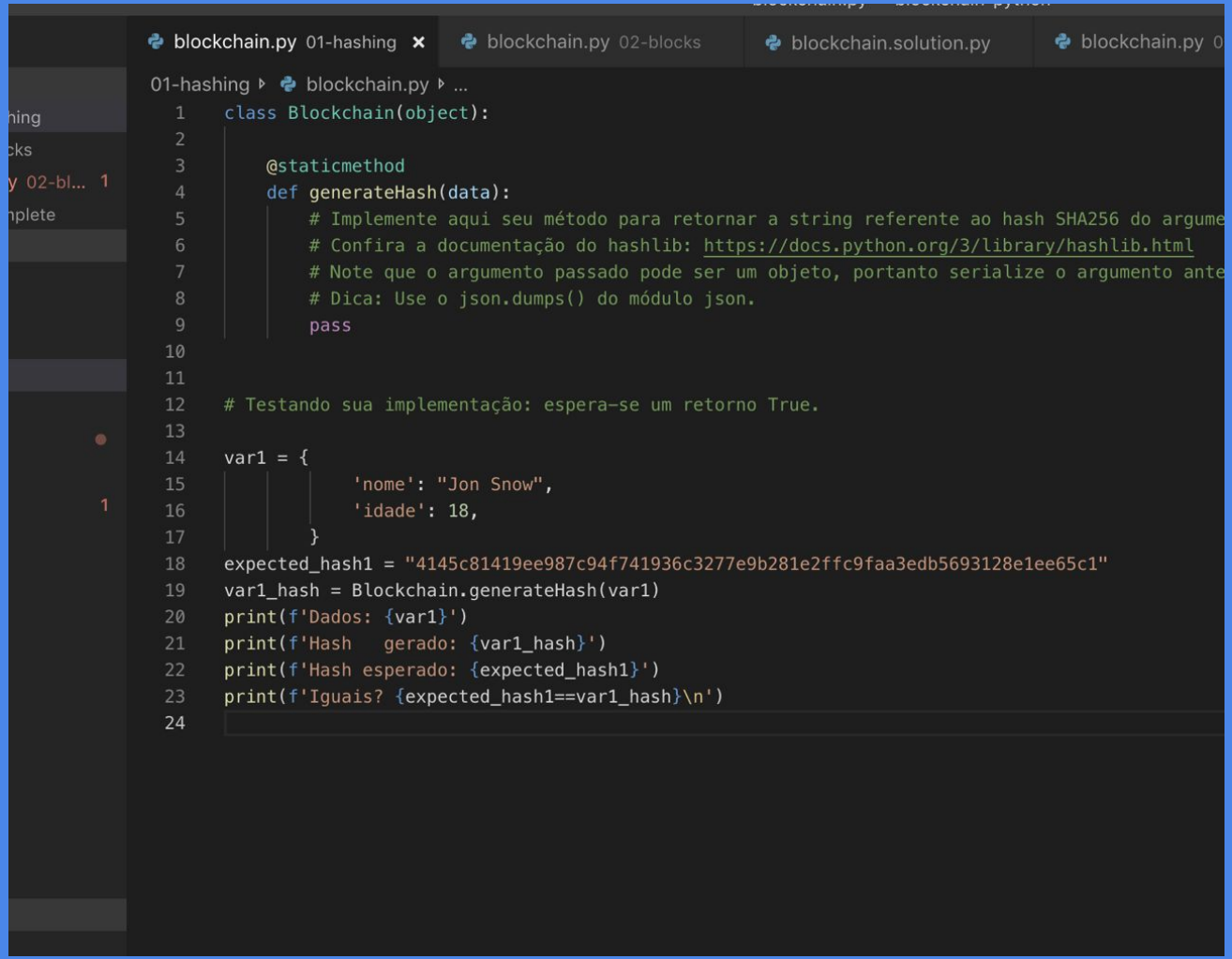
No releases published
Create a new release

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')