

# **CI1061 - Trabalho I**

Davi Campos Ribeiro

Fernando de Barros Castro

Novembro 2025

# 1 Método de Desenvolvimento

- Método de Colaboração: Reuniões online para ideação e programação em pares
- Plataforma: Discord
- Linguagem: Golang

# 2 Registros de Desenvolvimento

## 2.1 8 de novembro de 2025

- **Escolha da Hash:** A *hash* escolhida foi `sha512_256`, pois se trata de uma função que combina a segurança do `sha512` com a eficiência espacial do `sha256`.
- **Decisões de Implementação:**
  - Múltiplos Clientes.
  - Formato do nodo da *blockchain*, contém um identificador, data, e valor inteiro que indica uma movimentação, valores negativos indicando retirada e positivo depósitos.
  - Definição da interface de comunicação que será usada para comunicar entre servidor e cliente. Foi definido um pacote que envia tipo e tamanho do *payload*, e baseado no tipo o recipiente decide como interpretar o *payload*, que pode assumir uma serie de estruturas definidas para diferentes tipo de interações.
  - Definição das funcionalidades providas por cada domínio da aplicação (Cliente, Servidor, *Blockchain*)
  - Definição do cliente como cli, permitindo fácil testagem por meio de *scripts* e interação por linha de comando
- **Implementação:**
  - Implementado estruturas de dados do projeto
  - Protótipos de todas as funções a serem implementadas
  - Definição de uma hierarquia de diretórios
  - Sistema de Build

## 2.2 10 de novembro de 2025

- Alterações de Projeto:
  - A dupla decidiu usar uma codificação em JSON para envio dos dados para melhor compatibilidade com a forte tipagem de Go que não permite um *cast* de um *array* de bytes para uma struct.
  - O campo de tamanho de *payload* foi removido em favor de strings terminadas em 0.
- Implementações:

- CLI que permite personalizar endereço, porta, ações e incluir valores para ações que os demandem.
- Loop principal do servidor concorrente que aceita novas conexões e cria uma rotina de Go para lidar com as requisições do cliente.
- Cliente resolve endereço do servidor e envia uma requisição ao servidor (ainda falta fazer cliente esperar por uma resposta).

- Testes:

- Teste de envio de uma mensagem singular em uma mesma máquina e entre máquinas com IPs distintos.

### 2.3 11 de novembro de 2025

- Implementações

- Função que verifica a integridade dos nodos, comparando o valor esperado de *hash* com o valor presente no nodo.
- Função que verifica integridade da lista.

### 2.4 12 de novembro de 2025

- Decisões de Implementação

- O id do cliente não pode ser zero, já que esse é o valor padrão e usado internamente para identificar quando o usuário não forneceu seu identificador.
- Não existe uma criação explícita de um cliente. A criação é identificada no primeiro depósito.
- A *blockchain* inicia com um único nodo zerado.
- Caso o cliente tente retirar dinheiro, dois erros podem ocorrer: cliente não foi encontrado ao percorrer a *blockchain* interna do servidor ou cliente tem saldo insuficiente.
- Em toda operação ao servidor, a integridade da *blockchain* é verificada. Caso essa esteja corrompida, nenhuma operação além da própria requisição de verificação de integridade é executada.

- Implementações

- Função para obter o saldo de um determinado cliente percorrendo todos os nodos.
- Simulação de corrupção da *blockchain* após  $N$  inserções de nodos onde  $N$  é definido por parâmetro do servidor.

- Testes

- Um *script* para testagem e demonstração da totalidade das funcionalidades implementadas, este cria um servidor configurado para falhar apos 6 inserções e então realiza por meio da cli uma serie de operações que demonstram as funcionalidades implementadas.