🎬 Roteiro Detalhado para Gravação do Vídeo

MC714 - Trabalho 2 - Sistema Distribuído com Lamport Clock e Bully

⚠️ ESTE ARQUIVO NÃO DEVE SER COMMITADO NO GIT

📋 Informações Gerais

- **Duração:** 5 minutos (máximo)
- **Formato:** Demonstração técnica + explicação
- **Participantes:** Sergio (298813) e José Victor (245511) - **AMBOS devem aparecer no vídeo**
- **Objetivo:** Demonstrar funcionamento dos algoritmos Lamport Clock + Bully em deployment real no GCP

🎯 Preparação ANTES da Gravação

1. Verificar que o deployment GCP está funcionando

```
export GCP_PROJECT_ID="trabalho2-477920"
./scripts/gcp/check-gcp-status.sh
```

Verificar que todos os 3 nós respondem:

- ✅ Node 1 (Iowa): HTTP respondendo
- ✅ Node 2 (São Paulo): HTTP respondendo
- ✅ Node 3 (Sydney): HTTP respondendo

2. Preparar navegador para mostrar outputs

Opcional: Abrir navegador para mostrar:

- Código fonte no GitHub/VS Code
- Documentação (README.md, ARCHITECTURE.md)

3. Preparar terminais

Terminal 1: Para comandos curl
Terminal 2: Para executar scripts de teste
Terminal 3: Código fonte aberto (VS Code com main.py e lamport_clock.py)

4. Ferramentas de gravação

- **OBS Studio** ou similar para gravar tela
- **Zoom** se vão gravar vocês dois falando
- **Microfone** com boa qualidade
- Testar áudio ANTES de gravar

🎬 ROTEIRO MINUTO A MINUTO

⏱️ MINUTO 0-1: Introdução e Arquitetura (60 segundos)

Quem fala: Sergio ou José Victor (decidir antes)

O que mostrar:

1. **Tela inicial:** README.md do projeto aberto
2. **Falar:**
    - "Olá, somos Sergio e José Victor"
    - "Este é nosso Trabalho 2 de MC714 - Sistemas Distribuídos"
    - "Implementamos um sistema de log distribuído com dois algoritmos clássicos:"
        - Relógio Lógico de Lamport (para ordenação causal)

        - Algoritmo Bully (para eleição de líder)
3. **Mostrar arquitetura:**
    - Abrir docs/ARCHITECTURE.md
    - Explicar: "Deployamos em 3 regiões do Google Cloud Platform:"
        - Iowa (EUA)
        - São Paulo (Brasil)
        - Sydney (Austrália)
    - Total: ~36.000 km de separação geográfica
4. **Mostrar código brevemente:**
    - Abrir src/lamport_clock.py
    - Mostrar a classe LamportClock
    - Explicar: "O relógio incrementa localmente e se atualiza com max(local, remote) + 1"

⏱️ MINUTO 1-2: Algoritmo Bully - Eleição de Líder (60 segundos)

Quem fala: O outro integrante (alternar)

O que fazer:

1. **Explicar o Algoritmo Bully:**
    - "O Algoritmo Bully elege como líder o nodo com maior ID"
    - "Temos 3 nodos: 8001 (Iowa), 8002 (São Paulo), 8003 (Sydney)"
    - "Portanto, Sydney (8003) é o líder atual"
2. **Demonstrar no terminal - Verificar líder em cada nodo:**

```
echo "Verificando líder em Iowa:"
curl http://34.55.87.209/leader
# Output: 8003

echo "Verificando líder em São Paulo:"
curl http://34.95.212.100/leader
# Output: 8003

echo "Verificando líder em Sydney:"
curl http://35.201.29.184/leader
# Output: 8003
```

3. **Explicar o resultado:**
    - "Todos os nodos concordam que 8003 é o líder"
    - "Isso demonstra que a eleição Bully funcionou corretamente"
4. **Explicar a arquitetura single-leader:**
    - "Todas as escritas vão para o líder"
    - "O líder replica para os followers"

⏱️ MINUTO 2-4: Relógio de Lamport - Ordenação Causal (120 segundos)

Quem fala: Ambos (dividir explicação)

PARTE 1: Envio de Mensagens Concorrentes (60s)

1. **Terminal - Enviar mensagens de diferentes regiões:**

```
echo "Vamos enviar mensagens concorrentes de 3 localizações diferentes"

# Terminal 1 - Iowa
curl -X POST "http://34.55.87.209/?message=Mensagem_Iowa_1"

# Terminal 2 - São Paulo
curl -X POST "http://34.95.212.100/?message=Mensagem_Brasil_1"

# Terminal 3 - Sydney (Líder)
curl -X POST "http://35.201.29.184/?message=Mensagem_Sydney_1"
```

2. **Verificar os Lamport timestamps:**

```
echo "Verificando Lamport time em cada nodo:"
curl http://34.55.87.209/lamport_time
curl http://34.95.212.100/lamport_time
curl http://35.201.29.184/lamport_time
```

3. **Explicar:**
    - Mostrar: time: 1, 2, 3, 4, 5...
    - "Mesmo sem relógio físico sincronizado, mantemos ordem causal"
4. **Verificar replicação com curl:**

```
echo "Verificando mensagens em Iowa:"
curl -s http://34.55.87.209/messages | grep -c '"id"'

echo "Verificando mensagens em São Paulo:"
curl -s http://34.95.212.100/messages | grep -c '"id"'

echo "Verificando mensagens em Sydney:"
curl -s http://35.201.29.184/messages | grep -c '"id"'
```

5. **Explicar:** "Todos os nodos têm o mesmo número de mensagens - replicação automática funcionando!"

PARTE 2: Testes de Carga e Métricas (60s)

1. **Executar script de coleta de métricas:**

```
./scripts/gcp/collect-metrics.sh
```

2. **Enquanto roda, explicar:**
    - "Este script executa 6 testes completos:"
        - TESTE 1: Latência HTTP entre regiões
        - TESTE 2: Throughput - 50 mensagens concorrentes
        - TESTE 3: Relógio de Lamport com escritores concorrentes
        - TESTE 4: Latência de replicação geográfica
        - TESTE 5: Throughput com diferentes cargas (10, 25, 50, 100)
        - TESTE 6: Estado final do sistema
3. **Mostrar outputs importantes:**
    - Apontar para: "✓ HTTP respondendo"
    - Apontar para: "✓ Mensagens enviadas"
    - Apontar para: "✓ Replicação consistente em todos os nós"
    - Mostrar Lamport Time crescendo: 1, 2, 3, 4...
4. **Explicar as métricas obtidas:**
    - "Obtivemos throughput de 26.19 msg/s com 50 mensagens concorrentes"
    - "Latências de rede refletem distância geográfica:"
        - São Paulo: 19ms (mais próxima de Campinas, ~90 km)
        - Iowa: 295ms (~8.000 km)
        - Sydney: 652ms (~18.000 km)
    - "O throughput cresce linearmente até 50 mensagens"
    - "Mas degrada 90% ao atingir 100 mensagens (apenas 2.70 msg/s)"
    - "Latência média salta de 38.2ms para 369.8ms - crescimento exponencial"
5. **Mostrar mensagens via curl:**

```
echo "Últimas mensagens com Lamport timestamps:"
curl -s http://34.55.87.209/messages | jq '.[-5:]'
```

- Explicar: "Ordem causal preservada: t=0, 1, 2, 3..."

⏱️ MINUTO 4-5: Resultados e Conclusão (60 segundos)

Quem fala: Ambos (um fala resultados, outro conclusão)

RESULTADOS (30s):

1. **Executar verificação final do sistema:**

```
echo "Estado final dos 3 nodos:"
./scripts/gcp/check-gcp-status.sh
```

   - Mostrar que todos estão respondendo
   - Todos mostrando mesmo líder (8003)
   - Lamport timestamps consistentes

2. **Resumir métricas:**
   - "Conseguimos:"
     - ✅ Ordenação causal correta (Lamport Clock)
     - ✅ Eleição de líder automática (Bully)
     - ✅ Replicação geodistribuída (3 continentes)
     - ✅ Throughput de 26.19 msg/s (50 mensagens)
     - ✅ Latências: 19ms (São Paulo), 295ms (Iowa), 652ms (Sydney)
3. **Mostrar código brevemente:**
   - Abrir src/main.py
   - Mostrar onde incrementamos o Lamport clock
   - Mostrar onde verificamos o líder

CONCLUSÃO (30s):

1. **Limitações identificadas:**
   - "Sistema satura com 100 mensagens (degradação de 90%)"
   - "Arquitetura single-leader cria gargalo"
   - "Latência de Sydney (652ms) impacta throughput"
2. **Trabalho futuro:**
   - "Implementar multi-leader com CRDTs"
   - "Adicionar detecção de falhas completa no Bully"

    - "Otimizar replicação (atualmente ~28s, pode ser ms)"
3. **Demonstração de relatório:**
    - Mostrar rapidamente o relatorio.pdf
    - "Todos os detalhes técnicos, métricas e análises estão no relatório IEEE"
4. **Agradecimento:**
    - "Obrigado! Qualquer dúvida, estamos à disposição"
    - "Sergio Pezo (298813) e José Victor (245511)"
    - "MC714 - Sistemas Distribuídos - Unicamp 2025"

🎤 DICAS DE APRESENTAÇÃO

Para Falar Bem

1. **Não ler script palavra por palavra** - parecer natural
2. **Praticar ANTES** - pelo menos 2-3 vezes
3. **Falar com entusiasmo** - demonstrar que gostaram do projeto
4. **Ritmo:** Não muito rápido, não muito lento
5. **Pausas:** Dar pause quando mostrar algo importante no terminal

Para Gravar Bem

1. **Microfone próximo** - áudio é mais importante que vídeo
2. **Fechar outras aplicações** - evitar notificações
3. **Testar TUDO antes** - comandos curl, scripts check-gcp-status.sh e collect-metrics.sh
4. **Resolução:** 1920x1080 no mínimo
5. **Zoom no terminal** - fonte grande, legível (pelo menos 16pt)

Divisão de Falas (Sugestão)

Sergio:

- Introdução e arquitetura (0-1 min)

- Relógio de Lamport - parte 1 (2-3 min)
- Resultados (4-4.5 min)

José Victor:

- Algoritmo Bully (1-2 min)
- Relógio de Lamport - parte 2 (3-4 min)
- Conclusão (4.5-5 min)

OU alternativa: Revezar mais frequentemente (a cada 30s)

✅ CHECKLIST PRE-GRAVAÇÃO

- [ ] GCP deployment funcionando (todos os 3 nodos UP)
- [ ] Terminais preparados com comandos prontos para copy-paste
- [ ] Código fonte aberto (VS Code com main.py, lamport_clock.py)
- [ ] Documentação aberta (ARCHITECTURE.md)
- [ ] Relatório PDF compilado (para mostrar no final)
- [ ] Scripts testados: check-gcp-status.sh, collect-metrics.sh
- [ ] OBS/Zoom configurado e testado
- [ ] Microfone testado
- [ ] Script praticado (2-3 vezes)
- [ ] Cronômetro para controlar tempo (5 min MAX)
- [ ] Fechar Slack, email, notificações
- [ ] Fonte do terminal grande e legível (zoom in)

🎬 COMANDOS PARA COPY-PASTE DURANTE GRAVAÇÃO

Status do deployment

```
export GCP_PROJECT_ID="trabalho2-477920"
./scripts/gcp/check-gcp-status.sh
```

Verificar líder em todos os nodos

```
echo "Verificando líder em Iowa:"
curl http://34.55.87.209/leader

echo "Verificando líder em São Paulo:"
curl http://34.95.212.100/leader

echo "Verificando líder em Sydney:"
curl http://35.201.29.184/leader
```

Enviar mensagens concorrentes de diferentes regiões

```
echo "Enviando de Iowa:"
curl -X POST "http://34.55.87.209/?message=Mensagem_Iowa_1"

echo "Enviando de São Paulo:"
curl -X POST "http://34.95.212.100/?message=Mensagem_Brasil_1"

echo "Enviando de Sydney (Líder):"
curl -X POST "http://35.201.29.184/?message=Mensagem_Sydney_1"
```

Ver Lamport time em todos os nodos

```
echo "Lamport time em Iowa:"
curl http://34.55.87.209/lamport_time

echo "Lamport time em São Paulo:"
curl http://34.95.212.100/lamport_time

echo "Lamport time em Sydney:"
curl http://35.201.29.184/lamport_time
```

Ver contagem de mensagens (verificar replicação)

```
echo "Mensagens em Iowa:"
curl -s http://34.55.87.209/messages | grep -c '"id"'

echo "Mensagens em São Paulo:"
curl -s http://34.95.212.100/messages | grep -c '"id"'

echo "Mensagens em Sydney:"
curl -s http://35.201.29.184/messages | grep -c '"id"'
```

Ver últimas mensagens com timestamps

```
echo "Últimas 5 mensagens com Lamport timestamps:"
curl -s http://34.55.87.209/messages | jq '.[-5:]'
```

Executar coleta completa de métricas

```
./scripts/gcp/collect-metrics.sh
```

Executar testes do sistema

```
./scripts/gcp/test-gcp-system.sh
```

📊 DADOS PARA MENCIONAR

Métricas Principais

- **Throughput:** 26.19 msg/s (50 mensagens concorrentes)
- **Throughput degradado:** 2.70 msg/s (100 mensagens - degradação de 90%)
- **Latência São Paulo:** 19ms ± 0.4ms (próxima de Campinas, ~90 km)
- **Latência Iowa:** 295ms ± 7.0ms (~8.000 km)
- **Latência Sydney:** 652ms ± 51ms (~18.000 km)
- **Latência média por mensagem:** 38.2ms (50 msg) → 369.8ms (100 msg)
- **Lamport timestamps finais:** T1=14, T2=4, T3=104 (líder processa maioria)

Distâncias Geográficas

- Iowa ↔ São Paulo: ~8.000 km
- Iowa ↔ Sydney: ~13.000 km
- São Paulo ↔ Sydney: ~15.000 km
- **Total:** ~36.000 km de separação

Configuração Técnica

- **VMs:** 3x e2-micro (1 vCPU, 1GB RAM)
- **Regiões:** us-central1-a, southamerica-east1-a, australia-southeast1-a
- **Tecnologias:** Python 3.9, FastAPI, Docker, GCP
- **Algoritmos:** Lamport Clock + Bully

🚫 O QUE NÃO FAZER

❌ **NÃO** ler o código linha por linha
❌ **NÃO** gastar mais de 5 minutos
❌ **NÃO** explicar teoria em excesso (relatório já tem)
❌ **NÃO** mostrar erros (testar TUDO antes)

❌ **NÃO** gravar com barulho de fundo
❌ **NÃO** esquecer de aparecer no vídeo (ambos)
❌ **NÃO** falar muito rápido ou muito devagar

📤 DEPOIS DA GRAVAÇÃO

1. **Editar** (se necessário):
    - Cortar introdução/final desnecessários
    - Adicionar título/créditos (opcional)
    - Verificar áudio está bom
2. **Exportar:**
    - Formato: MP4
    - Resolução: 1920x1080
    - Qualidade: Alta
3. **Upload:**
    - YouTube (unlisted ou public)
    - Google Drive (dar permissão de visualização)
    - Copiar o link
4. **Adicionar link em ENTREGA.md:**
    - Substituir `[ADICIONAR URL DO VÍDEO NO YOUTUBE/DRIVE]`
    - Pelo link real

💡 SUGESTÕES EXTRAS (OPCIONAL)

Se sobrar tempo ou quiserem incrementar:

1. **Mostrar falha do líder** (se tiverem tempo):

```
# Derrubar Sydney
gcloud compute ssh log-node-3 --zone=australia-southeast1-a \
  --command='docker stop distributed-log'

# Aguardar 10s
# Mostrar que outro nodo vira líder
curl http://34.55.87.209/leader
```

2. **Mostrar detalhes de uma mensagem específica:**

```
# Ver mensagens completas com todos os campos
curl -s http://34.55.87.209/messages | jq '.[0]'
```

3. **Mencionar bônus implementados:**
    - Scripts de automação completos (deploy, test, monitoring)
    - Métricas detalhadas (6 testes diferentes)
    - Deployment geodistribuído (3 continentes)
    - Dashboard web interativo (em desenvolvimento)

🎯 OBJETIVO FINAL

Ao final do vídeo, o espectador deve entender:

✅ O que é Relógio de Lamport e como funciona
✅ O que é Algoritmo Bully e como funciona
✅ Como deployamos em 3 continentes
✅ Que o sistema FUNCIONA (demonstração real)
✅ Quais métricas obtivemos
✅ Quais as limitações identificadas

BOA SORTE NA GRAVAÇÃO! 🎬🚀

Sergio (298813) e José Victor (245511)

MC714 - Sistemas Distribuídos - Unicamp 2025

📖 GUION LITERAL DE LECTURA (PARA COMPLETAR TIEMPO)

🎤 GUION COMPLETO - 5 MINUTOS EXACTOS

[0:00 - 0:45] INTRODUCCIÓN (45 segundos)

PERSONA 1 (Sergio o José Victor):

> "Olá, bom dia! Nós somos Sergio Pezo, RA 298813, e José Victor Santana, RA 245511. Este é o nosso Trabalho 2 da disciplina MC714 - Sistemas Distribuídos, ministrada pelo Professor Carlos Astudillo na Unicamp.
>
> Neste projeto, implementamos um sistema distribuído de log que garante ordenação causal de mensagens. Para isso, utilizamos dois algoritmos clássicos da literatura: o Relógio Lógico de Lamport, proposto em 1978, e o Algoritmo Bully para eleição de líder, proposto por Garcia-Molina em 1982.
>
> O sistema foi deployado em três regiões geograficamente distantes do Google Cloud Platform: Iowa nos Estados Unidos, São Paulo no Brasil, e Sydney na Austrália. Isso representa uma separação geográfica de aproximadamente 36 mil quilômetros, permitindo avaliar o comportamento dos algoritmos sob latências realistas de redes de longa distância."

[Mostrar na tela: README.md ou ARCHITECTURE.md com diagrama]

[0:45 - 1:30] ALGORITMO BULLY (45 segundos)

PERSONA 2 (alternar):

> "Vamos começar demonstrando o Algoritmo Bully. Este algoritmo garante eleição de líder de forma determinística: o processo com maior identificador sempre assume a liderança.
>
> No nosso sistema, temos três nodos com os seguintes IDs:
>
> - Node 1 em Iowa: ID 8001
> - Node 2 em São Paulo: ID 8002
> - Node 3 em Sydney: ID 8003
>
> Portanto, Sydney com ID 8003 é o líder atual. Vamos verificar isso consultando cada nodo."

[Executar comandos curl mostrando que todos respondem leader: 8003]

```
curl http://34.55.87.209/leader    # Iowa → Output: 8003
curl http://34.95.212.100/leader   # São Paulo → Output: 8003
curl http://35.201.29.184/leader   # Sydney → Output: 8003
```

PERSONA 2:

> "Como podemos ver, todos os três nodos concordam que 8003 é o líder. Isso demonstra que a eleição Bully funcionou corretamente. Em nossa arquitetura single-leader, todas as escritas são direcionadas ao líder, que então replica as mudanças para os seguidores."

[1:30 - 2:45] RELÓGIO DE LAMPORT - PARTE 1 (75 segundos)

PERSONA 1:

> "Agora vamos demonstrar o Relógio Lógico de Lamport. Este algoritmo estabelece uma ordenação parcial de eventos em sistemas distribuídos através de timestamps lógicos que respeitam a relação de causalidade 'aconteceu-antes'.
>
> O relógio funciona da seguinte forma: cada processo mantém um contador local que incrementa antes de executar eventos. Ao enviar uma mensagem, o processo inclui seu timestamp atual. E ao receber uma mensagem, o processo atualiza seu relógio para o máximo entre o valor local e o recebido, mais um.
>
> Vamos enviar mensagens concorrentes de diferentes regiões para demonstrar isso na

> prática."

[Executar comandos de envio de mensagens]

```
# Enviar de Iowa
curl -X POST "http://34.55.87.209/?message=Mensagem_Iowa_1"

# Enviar de São Paulo
curl -X POST "http://34.95.212.100/?message=Mensagem_Brasil_1"

# Enviar de Sydney
curl -X POST "http://35.201.29.184/?message=Mensagem_Sydney_1"
```

PERSONA 1:

> "Agora vamos verificar os Lamport timestamps em cada nodo."

[Executar comandos para ver lamport_time]

```
curl http://34.55.87.209/lamport_time
curl http://34.95.212.100/lamport_time
curl http://35.201.29.184/lamport_time
```

PERSONA 1:

> "Observe como os timestamps crescem monotonicamente: 1, 2, 3, 4, 5... Mesmo sem um relógio físico sincronizado, conseguimos manter uma ordem causal consistente entre os eventos."

[2:45 - 3:45] MÉTRICAS E ANÁLISE (60 segundos)

PERSONA 2:

> "Realizamos experimentos extensivos a partir de Campinas para avaliar o desempenho do sistema. Vou apresentar as métricas mais importantes que coletamos.
>
> **Latências de rede:**

- São Paulo: 19 milissegundos, com desvio de apenas ±0.4ms. Isso é esperado, pois Campinas está a apenas 90 quilômetros de São Paulo.
- Iowa: 295 milissegundos ±7ms, refletindo a distância de aproximadamente 8 mil quilômetros.
- Sydney: 652 milissegundos ±51ms, a região mais distante com 18 mil quilômetros.

A latência para São Paulo é 15 vezes menor que Iowa e 34 vezes menor que Sydney. Isso demonstra claramente o impacto da distância geográfica e valida a importância de estratégias de edge computing.

Throughput:

- Com 50 mensagens concorrentes, atingimos throughput máximo de 26.19 mensagens por segundo.
- No entanto, ao aumentar a carga para 100 mensagens, observamos degradação severa: o throughput cai para apenas 2.70 mensagens por segundo.
- Isso representa uma degradação de 90%, indicando saturação do sistema.

Latência média por mensagem:

- Com 50 mensagens: 38.2 milissegundos
- Com 100 mensagens: 369.8 milissegundos

Este crescimento exponencial, de 38ms para 369ms, confirma a formação de fila de espera no servidor e indica os limites da nossa arquitetura single-leader com VMs e2-micro de apenas 1 vCPU."

[3:45 - 4:30] CONVERGÊNCIA E CONSISTÊNCIA (45 segundos)

PERSONA 1:

"Para avaliar a convergência dos Relógios de Lamport sob concorrência, executamos três rodadas de escritas simultâneas nos três nodos, totalizando 9 mensagens. Ao final, os timestamps foram:

- Node 1 em Iowa: timestamp 14
- Node 2 em São Paulo: timestamp 4
- Node 3 em Sydney: timestamp 104

A disparidade nesses valores é significativa e revela um comportamento importante: o líder Sydney processou a grande maioria das mensagens, atingindo timestamp 104, enquanto os followers ficaram em 14 e 4. Isso confirma que a arquitetura single-leader concentra a carga no nodo líder.

No entanto, o mais importante é que a propriedade de ordenação causal foi mantida em todos os experimentos: eventos locais incrementam o relógio monotonicamente, mensagens recebidas sempre têm timestamps maiores que mensagens anteriores do mesmo processo, e a relação 'aconteceu-antes' é preservada através dos timestamps."

[Mostrar rapidamente as mensagens com timestamps via curl ou na tela]

[4:30 - 5:00] CONCLUSÕES (30 segundos)

PERSONA 2:

"Em conclusão, implementamos com sucesso um sistema distribuído que valida os algoritmos clássicos de Lamport e Bully em um ambiente geodistribuído real.

As limitações identificadas incluem: saturação com 100 mensagens devido ao gargalo de CPU nas VMs e2-micro, e o ponto único de falha da arquitetura single-leader.

Como trabalho futuro, propomos: implementar detecção de falhas completa no Bully, explorar arquiteturas multi-leader com CRDTs para resolver conflitos, e otimizar a replicação assíncrona.

Todos os detalhes técnicos, código-fonte, scripts de deployment e análises completas estão disponíveis no nosso repositório e no relatório IEEE de 4 páginas que acompanha este trabalho.

Muito obrigado pela atenção! Ficamos à disposição para perguntas."

[Mostrar na tela: Relatório PDF ou README.md]

AMBOS:

> "Sergio Pezo, RA 298813, e José Victor Santana, RA 245511. MC714 - Sistemas Distribuídos - Unicamp 2025. Obrigado!"

⏱️ CONTROLE DE TEMPO

Segmento	Tempo	Duração
Introdução	0:00 - 0:45	45s
Algoritmo Bully	0:45 - 1:30	45s
Relógio de Lamport Parte 1	1:30 - 2:45	75s
Métricas e Análise	2:45 - 3:45	60s
Convergência e Consistência	3:45 - 4:30	45s
Conclusões	4:30 - 5:00	30s
TOTAL		**5:00**

💡 DICAS PARA LEITURA DO GUION

1. **Pratique 2-3 vezes antes** de gravar para pegar o ritmo
2. **Leia com naturalidade** - não precisa decorar, pode ler diretamente
3. **Faça pausas** entre parágrafos para dar tempo de processar
4. **Enfatize números importantes:** 26.19 msg/s, 90% degradação, 36.000 km
5. **Mostre na tela** os outputs dos comandos enquanto lê
6. **Alterne quem fala** conforme indicado (PERSONA 1 e PERSONA 2)
7. **Use cronômetro** - se estiver passando de 5 min, fale um pouco mais rápido
8. **Se faltar tempo** - corte a seção "Convergência e Consistência" (4:30)
9. **Se sobrar tempo** - adicione pausas maiores entre seções

---

🎬 CHECKLIST FINAL

- [ ] Praticar leitura do guion 2-3 vezes (cronometrar)
- [ ] Preparar terminais com comandos prontos
- [ ] Abrir arquivos para mostrar (README, ARCHITECTURE, código)
- [ ] Testar todos os comandos curl funcionando
- [ ] Configurar OBS/Zoom com boa qualidade de áudio
- [ ] Ambos aparecerem no vídeo (ou alternando)
- [ ] Cronômetro visível durante gravação
- [ ] Fonte do terminal grande e legível

BOA GRAVAÇÃO! 🎥🚀